DataGraph

Version 5.3

A SIMPLE AND POWERFUL GRAPHING PROGRAM

VISUAL DATA TOOLS, INC.

DataGraph

Version 5.3

The DataGraph manual was written by Pamela Schultz and David Adalsteinnson.

DataGraph was created by David Adalsteinnson.

Released March 2024

For the latest updates and corrections to this manual visit visualdatatools.com/DataGraph.

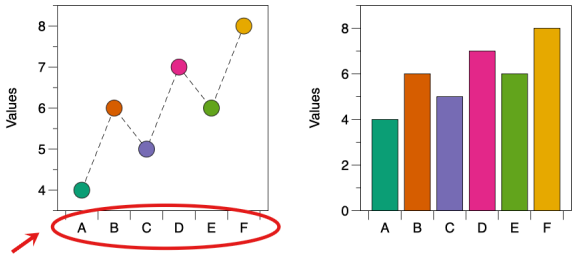
Send questions and comments to help@visualdatatools.com.

Table of Contents

What's New in version 5.3?

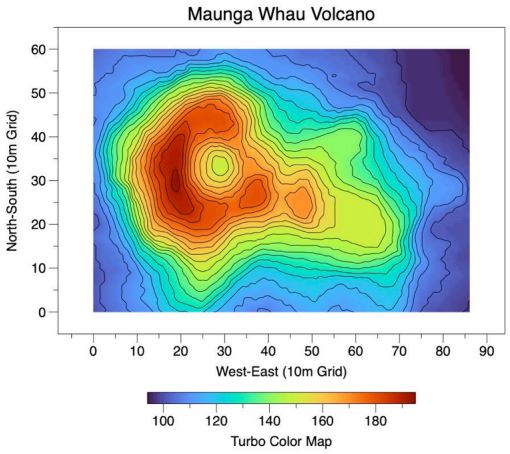
Easier to Plot Categorical Data

You can now use Text columns as direct input to the Plot, Points, and Bar commands, making it easier to make graphs that combine categorical and numerical data.



New Color Ramp Options

The Scalar field command has been upgraded with new color ramps. The old Rainbow color ramp was replaced with 'Turbo', an improved rainbow color ramp.



Other Updates

- New Feature to Create a QR Code. Use the Graphic command to create.
- Easier Centering for Split Graphs. Use Text command "Graph Center" option.
- Better Importing from Excel. The default import now groups data from tabs.
- And more!

For a complete list of updates, visit the **What's New** page on the Community website.

You may also notice some updates to this document. We split the sections on Data and Commands and changed the overall formatting for easier scrolling. Enjoy!

1. Introduction

DataGraph is a feature-rich drawing and data analysis environment for numerical data, categorical data, and analytic functions. The program can be used by anyone, from a middle school student learning about graphs and trends to a researcher analyzing millions of data points and creating graphs for publications.

DataGraph Highlights

DataGraph is designed to represent two-dimensional data sets. In addition, you can compute statistics and fit data using analytic expressions. DataGraph allows you to combine plots, scatter plots, fits, analytic functions, bar graphs, stock charts, and box plots in the same graph. You can attach labels, lines, add graphical elements, highlight regions, and magnify regions to point out details. All of this is done interactively and updates automatically as you change data or settings.

DataGraph uses default rules to create a consistent, clean and professional-looking graph. You can adjust every aspect of the graph and overwrite the default rules as needed. Using color schemes and font styles, DataGraph makes it easy to create consistent graphs and change the look and feel of a graph with a minimal number of clicks.

Exploring Data

DataGraph is not just a drawing program. The program contains a number of exploratory data analysis features. You can map columns by using analytic expressions to combine columns, compute maxima, accumulate rows, etc. You can compute statistics and fit data using linear and nonlinear functions. You can compute histograms and box plots to drill into your data set and explore patterns and trends. You can change variables using sliders.

Beyond DataGraph

Some things are outside scope of DataGraph, such as the three-dimensional representations of data. This does not fit into the two-dimensional focus of DataGraph, and a tool like ImageTank would be better in this case. Faux 3D, such as extruded bars or shadows, is inconsistent with the focus on exact data representation and graph cleanliness. Use KeyNote when this need arises.

Our Beta Version

You can download the DataGraph beta from our website, and stay on the bleeding edge. User suggestions and quick fixes to issues are available in the beta to all licensed users.

The DataGraph Community

Be part of our active community that doesn't hesitate to suggest new features and point out things that they feel should be improved. A large part of the development of the program has been driven by user feedback. Additional actions and drawing methods are constantly being added.

Use the following link to access the community, or use the **Help** menu.

https://community.visualdatatools.com/datagraph

The DataGraph Channel

Make sure to subscribe to our YouTube Channel. We have lots of tutorials videos of paste webinars, and much more to come.

https://youtube.com/DataGraph

Contact Us

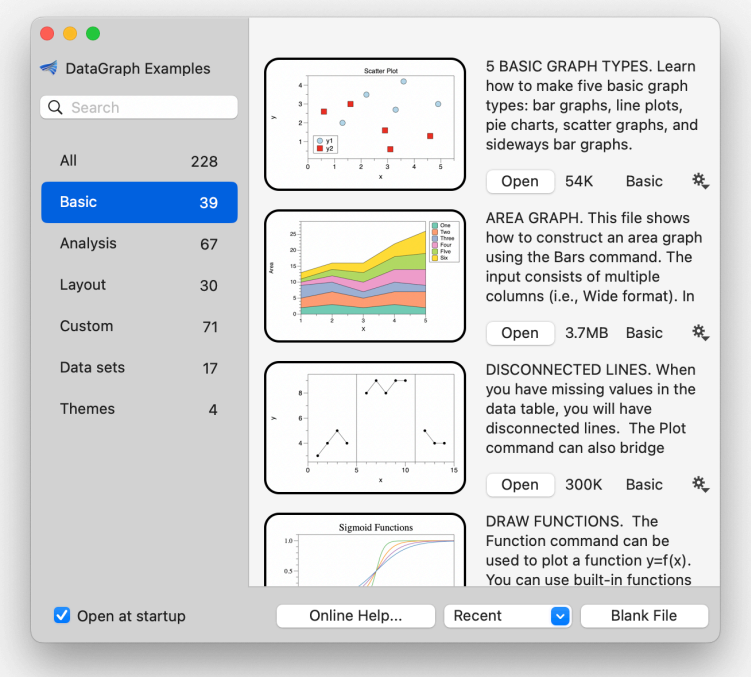
In addition to the user forum on the community, you can always email us with any questions you have at **help@visualdatatools.com**.

2. Overview

DataGraph files can contain data, calculations, graphs, and annotations all in a single file.

Example Files

When you first open DataGraph, you will see the DataGraph Examples window. This window gives you access to an online repository of files that you are free to use and modify. These are also intended as a learning tool. Many of the files contain notes and links to help.



When you click **Open,** the file is downloaded to your computer, and a new Untitled file is created. When you edit that file, you do not change the example file.

Use the search bar in the top left to filter the files or find specific graph types. You may want to change to view **All** when searching. For example, here the word

The example files window will open every time you open DataGraph. If you don't want to see this when you open, deselect the '**Open at startup**' check box in the bottom left. To open the example files window, select **File > Online examples**.

User Interface

The user interface (UI) is an adjustable layout that has five main sections. Here we show the default UI when you first open the program, but note that there are many ways to customize the UI.

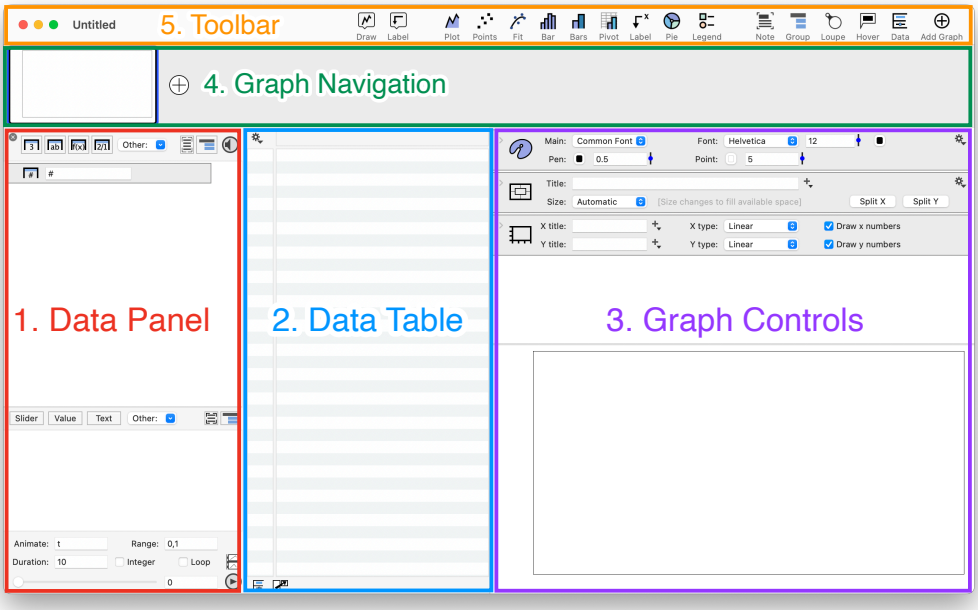
1. Data Panel - Left of the screen.

2. Data Table - Middle of the screen

3. Graph Controls - Right side

4. Graph Navigation - Across the top

5. Toolbar - Very top of the UI



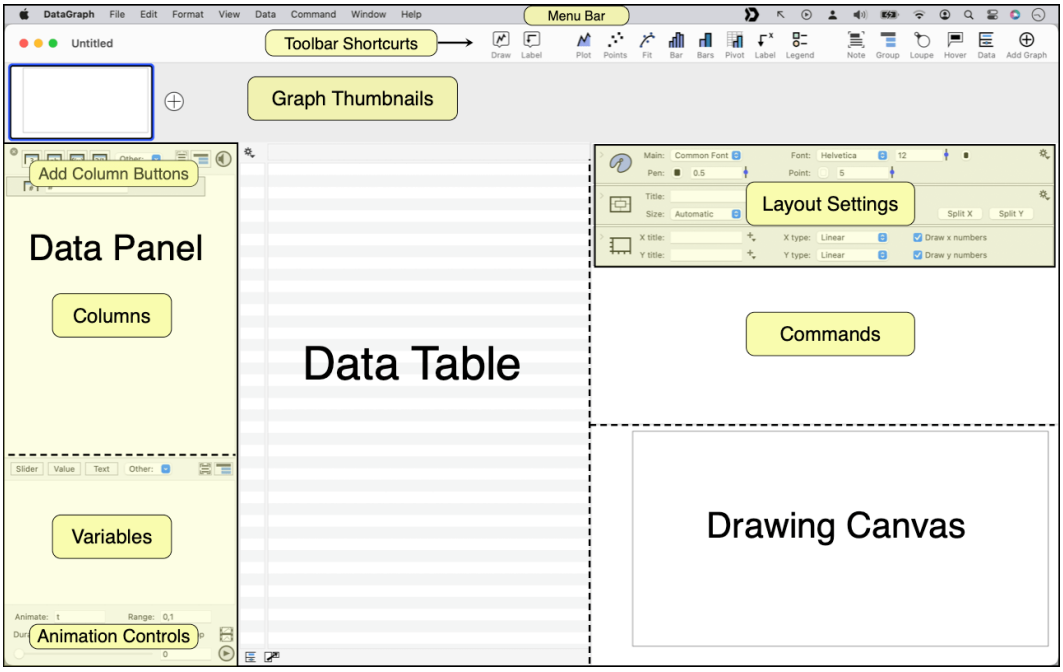
Detailed View

Below is a more detailed view of the UI.

- Menu Bar — Top of the screen.
- Toolbar – Across the top of the file.
- Graph Thumbnails – Below the toolbar.
- Data Panel — Left of the screen.
- Data Table — Middle of the screen.
- Drawing Canvas — Bottom right.
- Settings & Commands — Above the canvas.

The size of these sections can be adjusted by selecting and dragging the sliding bars between them. The image at the bottom of this page shows each section and the sliding bars are highlighted with a blue line.

In the following sections we will discuss the components of the UI.



A Blank DataGraph File. Dashed lines are adjustable. The Data Panel can be toggled open/closed.

Data Panel & Data Table

By default, the left-side of the user interface has a side-panel that contains a list of all data in the file. This **Data Panel** can be toggled from view using **View > Show Definitions** (⌘-**D**) .

You can also close the side panel by clicking on the **Data** icon in the top right of the toolbar or the bottom left of the data table. When the data panel is hidden, you can also double click right side of any column header to open the side panel to that entry in the column list.

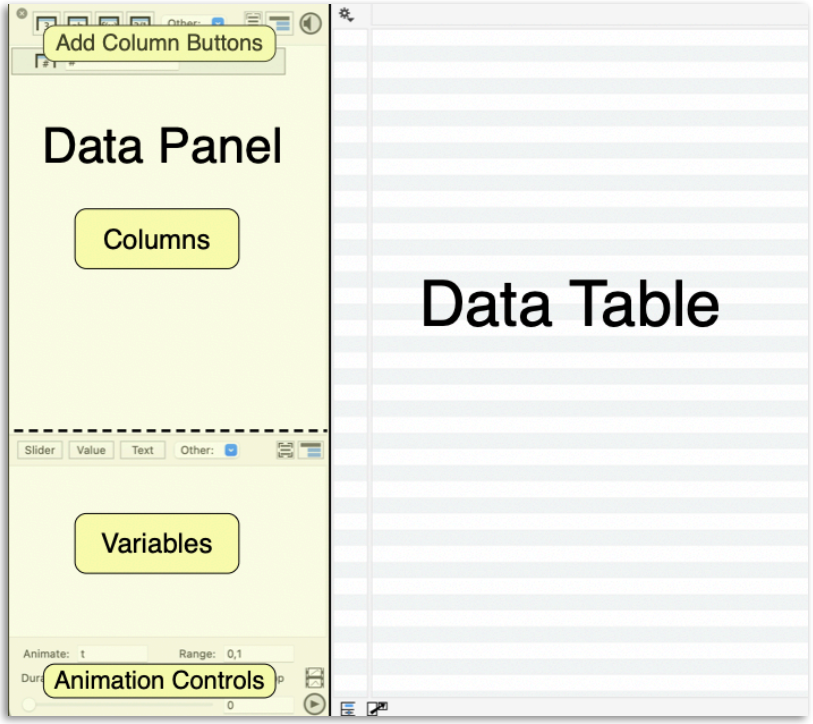
The Data Panel has four main sections:

1. **Add Column Buttons** - This is where you can create new columns for entering data. Note when you import data, columns are automatically created.

2. **List of Columns** - Each column or column group is shown as an object

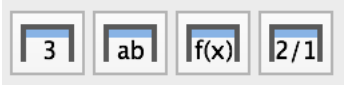
3. **List of Variables** - Add slider variables, color schemes, and more

4. **Animation Controls** - Use the animation variable to create movies



Add Column Buttons

You can add additional columns by clicking on the buttons at the top of the data panel to add the four main column types: numbers, text, expressions, and date.



Columns

When the data panel is open, you see the structure of the data table on the left of the screen. Here the columns of data are listed and can be reordered by clicking and dragging. Every column in the file has an entry in the list. Each column also has a specific type, indicated by its icon.

For example in the above screen shot, there are four columns defined, '#', 'x', 'y', and 'label'. The 'label' column is a text column. This is primarily used for labeling data rows but can also be used in drawings. The first column is simply called # and contains the row number. This cannot be removed but you can change its name. The x and y columns are number columns.

For more detail, see the **Working with Data** and **Column Types** chapter.

Variables

In the lower left corner of the window, you can define variables. Variables allow you to define numbers and text to which columns and fields can refer. Variables can be used in columns, including the standard numerical column, as well as expression columns. Variables can also be used in text fields, such as legends, titles, labels, etc.

For more detail, see the **Variables** chapter.

Animation Controls

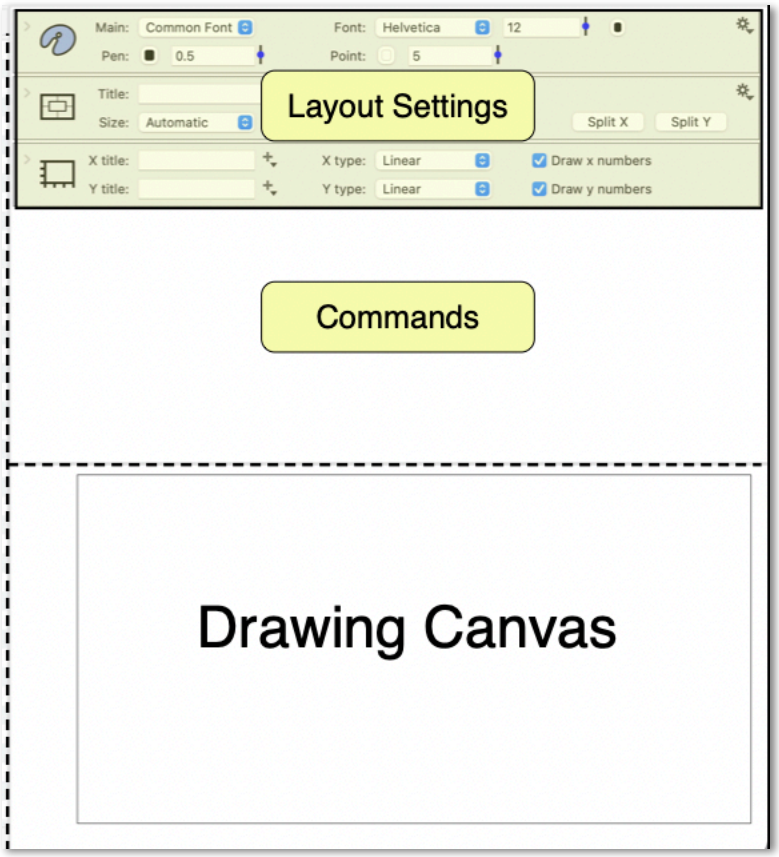
The bottom of the data list are settings for creating animations. By default the name of the variable is 't', but you can change this in the Animate entry box.

Graph Controls

The top half of the graph controls consists of layout settings, we control things, like the fonts, and commands where you specify what you want to draw, such as line graphs, bar graph, and histograms.

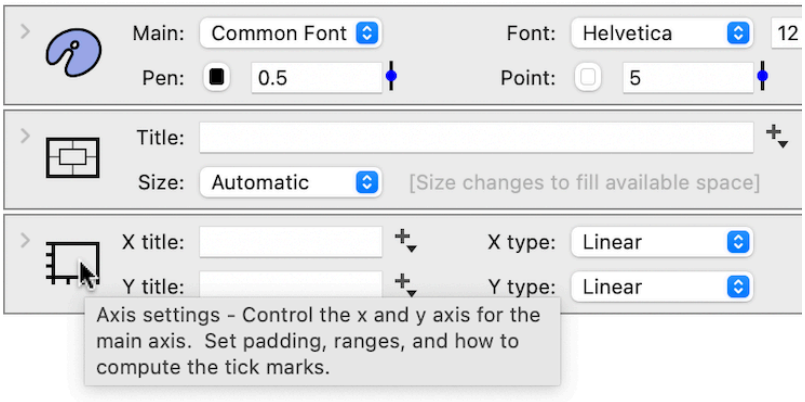
The layout settings are fixed in place, and are grouped into style settings, access settings, and canvas settings. The commands are objects that you can click and drag to move up and down and are drawn on the canvas in layers from top to bottom.

The drawing canvas is also interactive. By default the size of the canvas will change when you click and drag the sliding bars around it. But you can also fix the size in the layout which is recommended for publication graphs.



Layout Settings

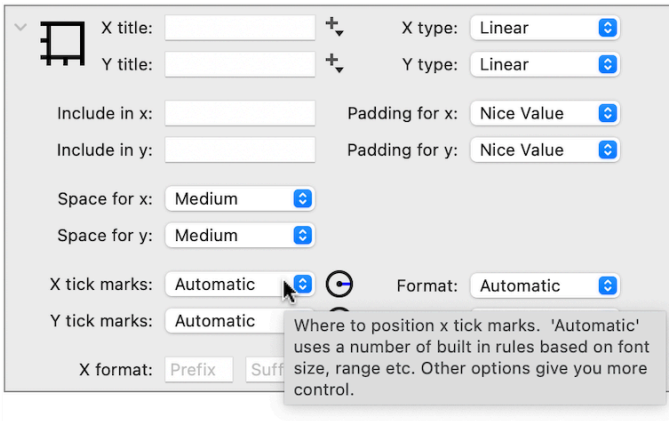
Every graph has **Layout settings**. The settings are the control center for your graph. They are comprised of three rectangular objects: Style, Canvas, and Axis. Hover the mouse over the icons or menus and tooltips will appear with the description of that item. Here you can see the tooltip describing the Axis settings themselves.



Menu Options

The layout settings and drawing commands all contain menus where you can select from various options. Many settings are by default set to ‘Nothing’ or ‘Automatic’, but you can adjust practically every aspect of a graph.

As an example, consider how tick marks are specified. The tick marks are set using the A**xis** settings. After clicking the top left corner of the axis command, the tick mark settings are shown (third entry down). By default **X tick marks** is set to ‘Automatic’ and DataGraph picks tick marks and formats labels to give a good visual appearance.



The tick mark menu contains several more settings. When you pick a different X tick marks option (e.g., ‘Categories’), the options displayed to the right of the menu change.

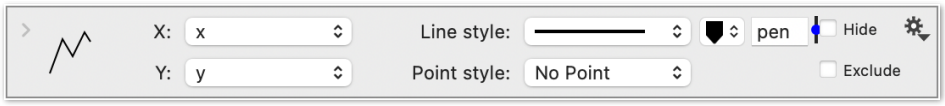
Other menu selections, such as ‘Angles’ or ‘Labels’, will have different options. DataGraph compacts the user interface like this in several places. It is recommended that you spend some time exploring the menus a bit to get more and more familiar with the many options that are available.

Drawing Commands

DataGraph does not present you with a choice of predefined graph types but rather gives you a canvas where you build a graph by adding and combining drawing elements.

We call these elements **Drawing Commands**. These are not text-based commands, instead, they are visual objects that have menus for inputting data and show sliders and other menus for varying input options.

When you add a drawing command, the drawing command object is placed below the axis settings. For example, if you click the **Plot** icon in the toolbar, DataGraph adds a plot command object to the command list. Once you have selected the data, a corresponding plot will immediately appear in the graph.



A Plot Command Object

Working with Commands

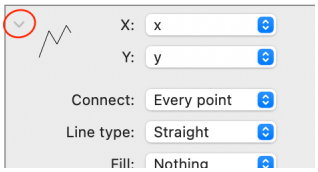
If you add a second plot command, the object for that command is also added to the command list. Each command is independent, but the data is drawn on the same canvas. DataGraph adjusts the x and y-axis ranges to fit both plots together.

A drawing command refers to the data by columns. You select columns using a pop-up menu or by dragging the column definition onto the drawing command. You can rename columns and reorder them without breaking the connection to your graph.

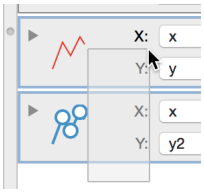
You can also drag a selection from one menu to another by holding down the command key when you click on the menu. When you pick a different data column or change values in a column, the graph is updated immediately.

You can adjust the drawing style, such as line width and color, marker size, and marker type, inside each drawing command. Each drawing command shows the most frequently accessed settings at the top, and they are always visible.

Click the small disclosure triangle in the top left corner to access more options.



To delete a command, you select it and hit the delete key. You can select it by clicking on the icon, or by clicking and dragging to select multiple commands.



There are over 20 drawing commands. The chapter entitled **Drawing Command Elements** explains aspects of the drawing commands that are common to all of them (e.g., how to specify a mask).

The **Drawing Commands** chapter explains each individual drawing command in more detail.

Drawing Canvas

The lower right corner shows the result from the drawing commands. This is not just a visual representation of your drawing, but an interactive window that you can click and drag to zoom in, add draggable text elements, and use hover labels and the loupe tool to explore data.

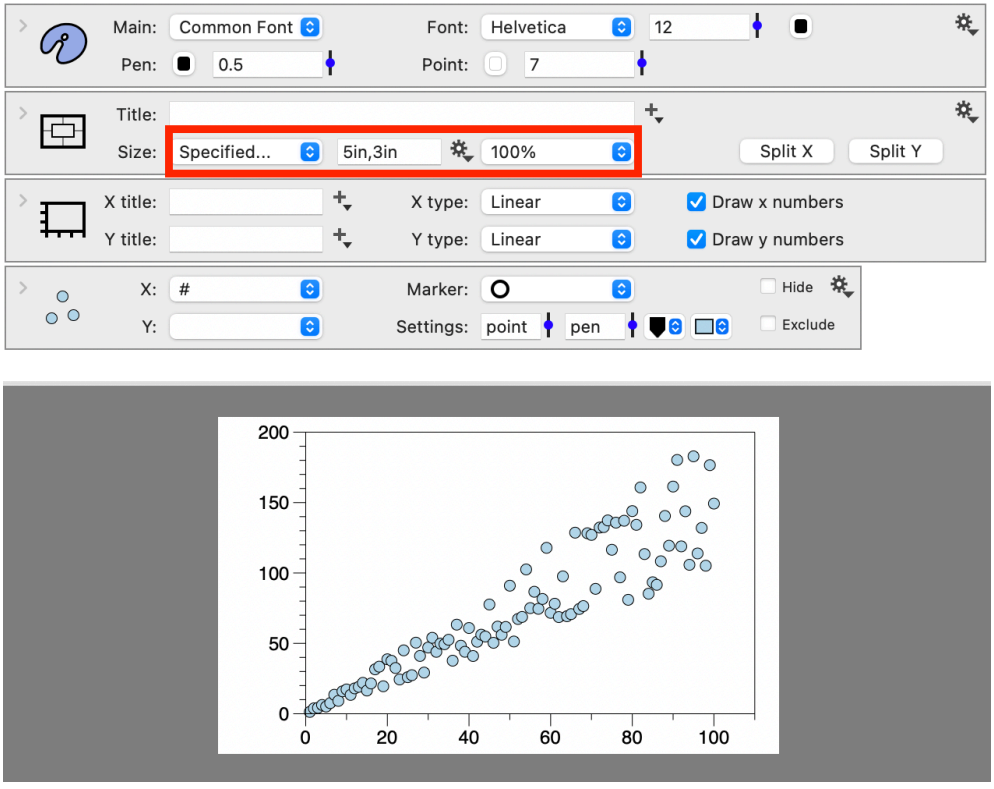
Graph Size

By default the size of the graph changes when you resize the window or adjust the split views. By default the size is set to 'automatic'. This means that the screen resolution is used to figure out the display scaling; thus, when you print a graph, it will print out the same size as it appears on the screen.

When you want to create graphs for publication you typically want to specify the size of the graph exactly. This is controlled using the **canvas**. If you switch the size to 'Specified', as shown here, you get two additional user interface settings.

One sets the size in pixels, the other controls the viewing resolution. You can use the units in, cm or mm to specify the size. No units means you are specifying the size in pixels.

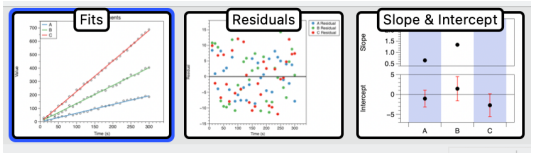
Here the size is set to the default of 5in by 3in.



Graph Navigation

You can have a number of graphs in the same file. This is done by clicking on the **Add Graph** button in the top right corner of the toolbar or **File > New Graph** or simply click the plus symbol next to the thumbnail.

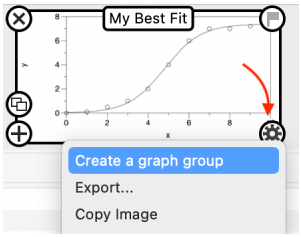
When you add multiple graphs, DataGraph adds a thumbnail for each one at the top of the screen. These can be named as shown below.



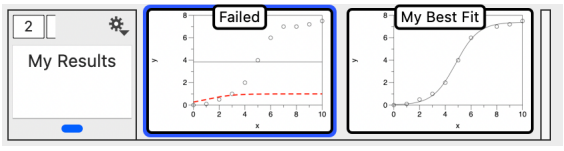
Click to navigate between graphs using the thumbnails. The currently selected thumbnail is shown as the graph and has a blue outline around the thumbnail. All of the graphs update automatically, so when you change the data you will see an update in the thumbnail.

Grouping Graphs

When you hover the mouse over a thumbnail, you will see controls around the graph. Click the bottom right gear menu to create a **Graph Group**.



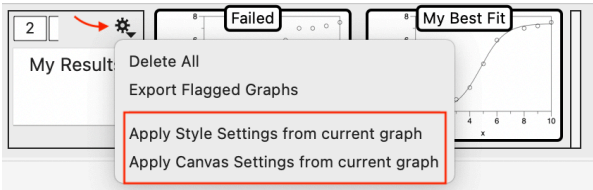
You can drag graphs in and out of the group. The graph group object has a beginning, where you can enter a name, and the end is a vertical bar. The number of graphs in the group are in the top left.



Click the top left to open and close. You can nest groups within groups. The top level group will will show the total number of graphs including any subgroups.

Applying Settings from the Active Graph

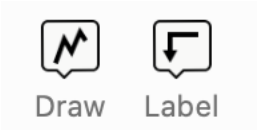
The graph group object also has a gear menu in the top right corner. One handy action is to apply the settings from the current graph to the entire group of graphs, eliminating the need to format each graph.



The current graph is the one that has a blue outline around the thumbnail. When the current graph is in the group, you will see a blue pill shaped object on the bottom of the group object. This way you can collapse the group, and keep track of the current graph thumbnail.

Toolbar

From the toolbar, you can access them using the **Draw** and **Label** pop-up menus. Click these icons to reveal a list of commands.



The commands in the **Draw** pop-up menu are used to display and analyzed data. The commands in the **Label** pop-up menu are used to add labels and annotations. The default toolbar also has several command shortcuts you can access directly.

You can customize the toolbar to include shortcuts for commands you frequently use. To learn how see:
https://community.visualdatatools.com/datagraph/knowledge-base/toolbar/

Make Your First Graph

STEP 1: Add a number column. You can use **Data > Add Number Column** (⌘-1) or click the icon as shown below. This creates a corresponding object in the Data List, where you can name the column.

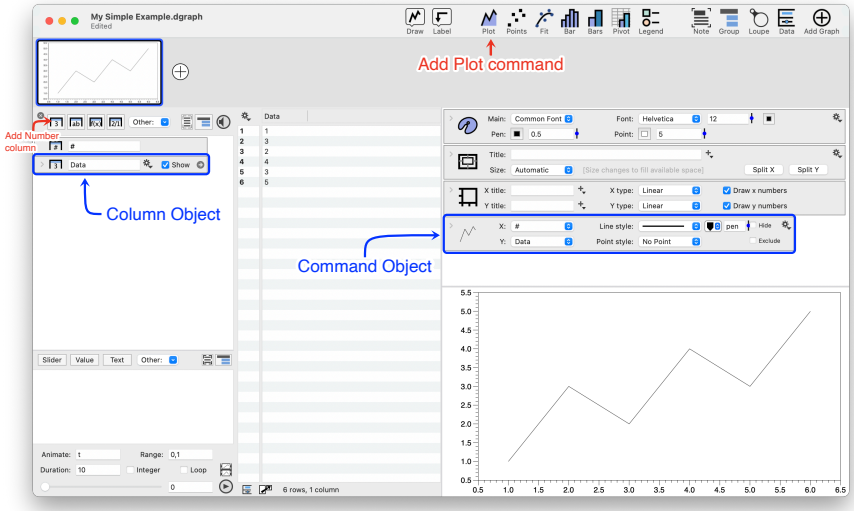
STEP 2: Enter data in the Data Table. Double click in Row #1 to start editing. After entering a value, hit return to move down the column to the next row. The row counter is populated as you moved to each new row.

STEP 3: Select the Data. Click the column header to select the data column. When you do, you will see the entire column is highlighted. This is different from a spreadsheet where you have to select a range. Here you chose the entire column.

STEP 4: Add a Plot command. You can use **Command > Add Plot** or click the **Plot** command short-cut in the toolbar. This creates the plot command object and draws a line in the graph.

The command has a drop-down menu for the x and y input. In this case, x is set to the row number '#' and y is the data column.

For a more detailed example, see the [10 Step Getting Started Guide] in the DataGraph Community.



Exploring Data

Quick Graph

Click on any column and hit the space bar to get a preview of the data.

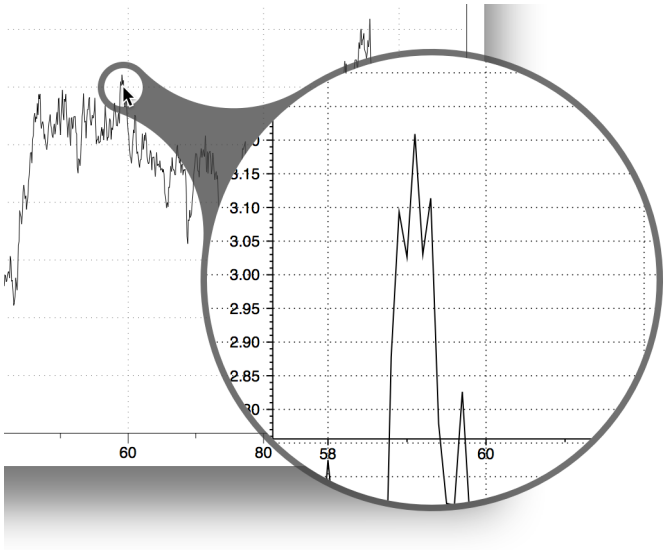
The type of graph you see will change depending on the data type of the column. Text columns will result in a count of each unique entry displayed in a sorted, sideways bar graph, using the Pivot command. Number columns will result in a histogram. If you chose more than one number column the Quick Graph will show the relationship between the variables (i.e., linear regression).

If you have multiple number columns selected the Quick Graph will show a matrix plot with each variable compared. Up to four number columns can be selected. You can change the columns selected with the Quick Graph window open and the preview will update as needed.

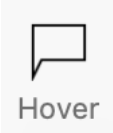
Loupe Tool

When a graph gets very busy it is sometimes necessary to zoom in on the detail. There are several ways to do this in DataGraph. You can zoom in by changing the axis ranges, you can use the Magnify command to draw an inset, or you can use the Loupe tool, accessible in the tool bar. Activate the **Loupe** tool using the toolbar shortcut or under the View menu (⇧⌘M).

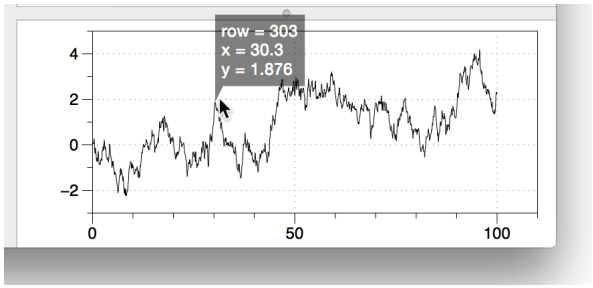
When you move the mouse over the graph, you see a magnified view, and can change the source and destination sizes by using the scroll wheel/gesture.



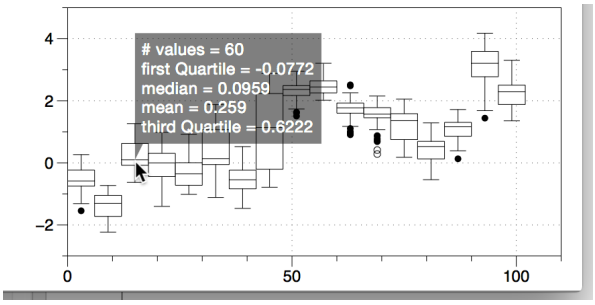
Hover mode



DataGraph has a Hover mode that will show the value and corresponding row in the Data Table for a given point on a graph as the mouse is moved over that section of the graph. Activate the **Hover** mode using the toolbar shortcut or under the **View** menu (⇧⌘H.)

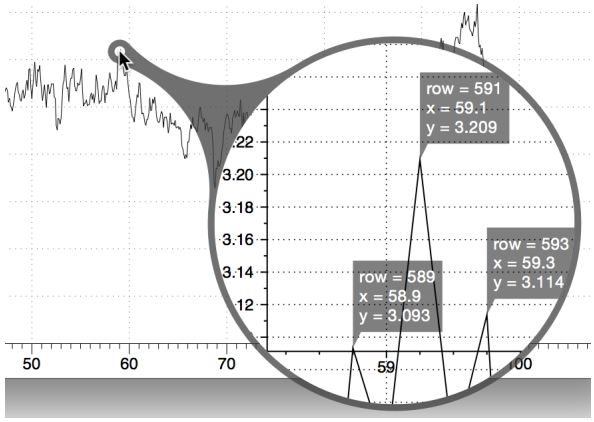


The Hover tool provides different information depending on the type of plot. In the **Box** plot example below, the Hover tool provides the statistics used to generate the plot based on the underlying dataset.



Hover and Loupe together

When both that Hover mode and the Loupe Tool are activated the Hover information is shown inside the loop tool instead of over the graph. Another difference is that multiple hover tags are shown at the same time.



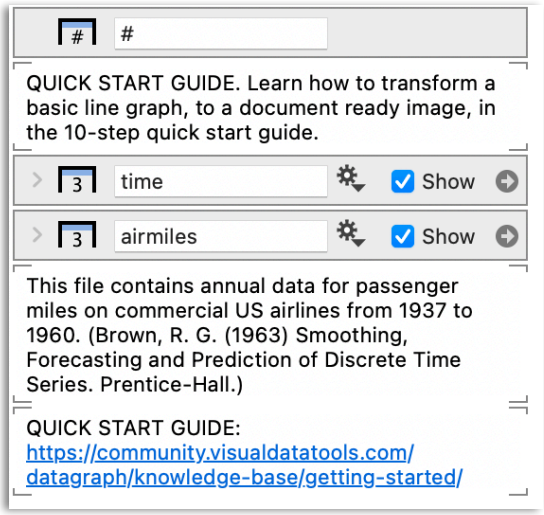
Annotating Your File

Your DataGraph file can also contain notes and annotations throughout. These notes can be both in the data panel and listed along with the commands. The comment blocks are elements that can be tracked up and down within the lists. The pop-up notes remain attached to whatever object they are created with. You can add a pop-up note to column objects, variable objects, and command objects.

Comment Blocks

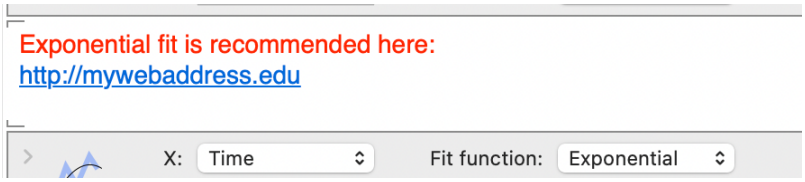
Add plain text comment blocks in the column list or variables list by clicking the note icon. These comment blocks can be moved around freely in the list.

You will find comments throughout the example files that are included with DataGraph. When you enter a URL into a comment block, DataGraph will automatically convert that to a clickable link.



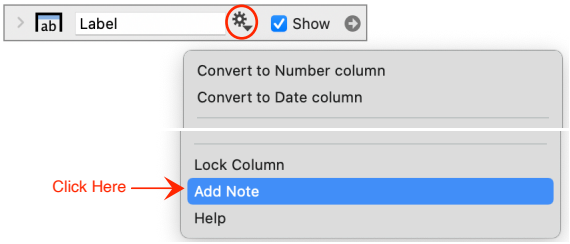
In the command list, rich text comment blocks can be added by selecting **Command > Add Text Description** or click the note icon in the tool bar.

The text can be formatted using the **Format** menu. For example, you can change the font size and color. You can also add URL links or even paste in images or screen shots.

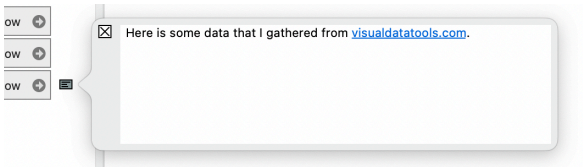


Pop-up Notes

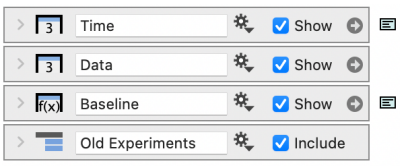
Another option is to attach a pop-up note. To create, click the gear menu on an object and select **Add Note**.



Pop-up notes can be attached to data columns, variables, commands, and groups. Simply type some text in the note and click the top left corner to close.



When closed, the notes show on the right side as a small icon next to the object.



If you export your data into an Excel document (**File > Export Data**), the Pop-up notes are also exported in a separate tab in the Excel file. See Exporting Data for more details.

To delete a pop-up note, open the note and delete the text in the note. The note will disappear when it is closed.

Saving Your File

Save your DataGraph file using **File > Save** or ⌘**-S**. The Save As dialogue box will appear for new files, as shown in the image below. Here, you must specify the name of the File. You can also select the file format, choosing between 'Package' and 'Single File.'

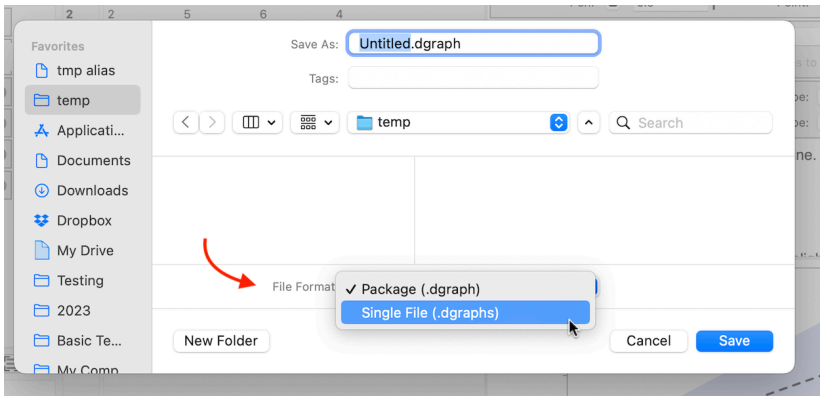
Package Format

Files with the extension *.dgraph use an Apple standard called a package file. These will appear as a file in the Finder but are a series of folders. For any *.dgraph file, you can control-click on the File in the Finder and ask to show package contents. Then, you will see the contents as a list of folders.

Single File Format

Starting with DataGraph version 5.2, you can save in a new single file format. Files saved in this format will have the extension *.dgraphs.

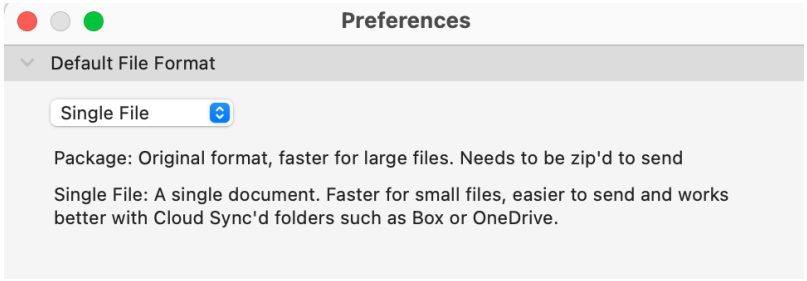
The single file format is recommended when using a cloud service (e.g., Google Drive, One Drive, etc.) where the file location is the cloud. This type of 'Files on Demand' service means that files are only downloaded to your computer when the File is requested.



Save As Dialogue Box

Set the Default Format

Open the Preferences Panel, **DataGraph > Settings** (⌘,), to set your default DataGraph file format.



Comparing Formats

Some users may prefer the Single file format due to the better compatibility with cloud file services; however, the Package file format has some advantages:

- Saves may be faster for files with large amounts of data (>10GB) or a large number of graphs.
- The contents of a *.dgraph file can be viewed as folders by control-clicking in the Finder. Then, you can extract data and graphics imported into the File.
- Files can be programmatically created by assembling folders in the proper format.

To use Package files with a file syncing service, do not use a 'Files on Demand' option and take regular backups.

Converting Between Formats

To convert an existing file, duplicate the File and save a copy in the specified format. You can do this in one step.

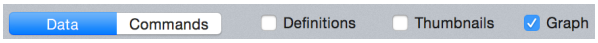
Open the File menu, then press the Option key ⌥. The **Save** menu changes to **Save As**, and you can specify the file format.

Customizing the UI

Focus Mode

Focus Mode (formally called Alternate Screen Mode) provides an alternate user interface that was designed for small screens. With this mode, you can focus on either the **Data** or the **Commands**, at one time.

You can active this mode under **View > Focus Mode**. At the top of the window is a narrow bar that allows you to control what is displayed.



You can show either the data or the commands, but not both. The **Definitions** checkbox will add a column/variable definitions list on the left side of the screen. This will work for both the **Data** and the **Command** option. The **Thumbnails** checkbox will show the thumbnail list of all the graphs in the file at the bottom of the screen.

The **Graph** checkbox allows you to show/hide the graph window. The graph window floats on top of everything. This means that you can cover the data/commands, definitions, and thumbnails. You can also have the graph displayed on a second monitor.

You can make **Focus Mode** your default user interface for any file you open using the Preferences Panel.

Separate the Graph Window

By default the graph window is embedded in the bottom right of the user interface. If you have a second or larger monitor you may want to separate the graph window. Select **View > Separate Graph Window**.

The separated window is not designed for projecting or remote sharing (e.g., Zoom) and may appear small on some larger monitors/projectors. Instead, use the Presentation Mode when sharing with remote users.

Customize the Toolbar

If you control, click or right, click on the toolbar, a menu will pop up and you can choose to customize the toolbar. This will show you an interactive view of all of the command icons that you can drag in or out of the toolbar location.

Under the **View** menu, you can also hide the toolbar or access the customization options there.

Presentation Mode

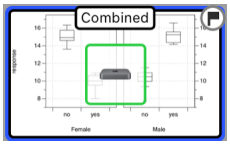
There are two ways to present graphs directly from DataGraph. Either present to a window that can be shared in a video conference, or present to an external monitor.

Present to a Window

To help DataGraph users connect and share their work we have added a new Presentation Mode for video conferences, such as through Zoom.

Select **View > Presentation > Present in Window**.

This creates a presentation window with the selected graph scaled to that window. Click in the center of a graph thumbnail to change the displayed graph.



The icon you see depends on the computer. In the image above, a mac mini was being used for the presentation.

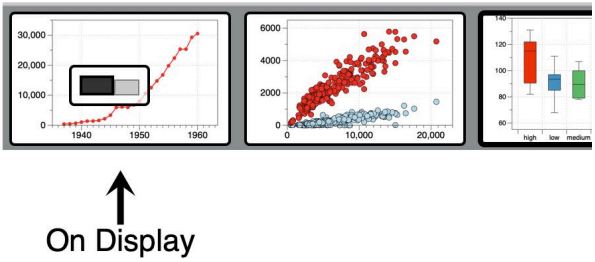
Present to an External Monitor

In Presentation Mode, DataGraph will take over an external monitor to display a selected graph. You control what is shown on the external monitor using the primary display (e.g., laptop).

Step 1: Select an External Monitor

Under **View/Presentation** menu, select the external monitor to control. The options you see listed will depend on the number of monitors you have connected and the monitor you are using to set-up the presentation.

The graph on display is indicated using an icon over the thumbnail.



Step 2: Change the Displayed Graph

Hover your mouse over a different thumbnail. After a moment, a representation of the screens will appear. Click the screen icon in the center of the thumbnail to send the graph to that monitor.

Step 3: Control Presentation

While using Presentation mode, DataGraph will continue to look and operate in the same way as before. The only difference is that you will have the small image displayed on graphs that are shown on external.

Since you control what is displayed using the thumbnails, you can continue to work in DataGraph and make changes to other graphs in your file, while a graph is being displayed on the external monitor.

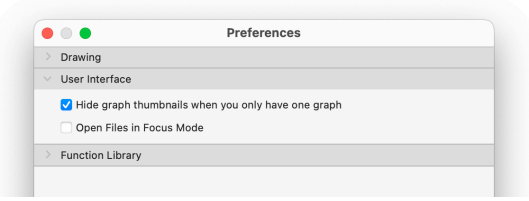
NOTE: You are not limited to one DataGraph file during the presentation. You can move between files, open files, or even create new graphs to display.

Step 4: End Presentation Mode

Under View/Presentation, select the same external monitor to stop displaying OR hit the ESC key.

Preferences Panel

Open the preferences panel from the menu bar using **DataGraph > Settings (⌘,)**. In this panel, you can modify the default behavior of the program. For example, you can hide thumbnails until you add a second graph to a file.



You can also chose Focus Mode as the default way to open files or create a **Function Library**, accessible from the Fit and Function commands, the Expression column and Expression variable. Import/Export your function lists to share with other DataGraph users.

As of DataGraph 5.2, you also have the option to set your default DataGraph file format.

3. Importing Data

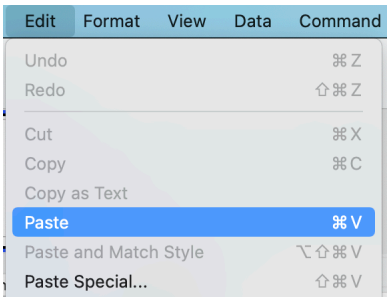
There are several ways to quickly import data into a DataGraph file. When you import data, the data is separated into columns and a data type is assigned to each column.

It's important to recognize that the data type is a way of interpreting the data. Changing the data type will not change your data. If you did, it was not assigned to the right type upon import, use the gear menu on the column object to assign the appropriate type.

For an overview with short demos, see **How to Import Data** in the Community.

Copy & Paste

If the data is copied to the clipboard you can select between two options — **Paste** or **Paste special…** The **Paste** option uses default rules to figure out what separator is used for the columns and to determine the column type (i.e., number or text). **Paste Special...** brings up a sheet where you can control the separator, transpose the data, and trim the data as it is pasted in the data table. Note that if you drag in a text region from Text Edit it is handled as a **Paste**.



Automatic Data Parsing

When you paste and or import data, DataGraph tries to understand the format of the data without any additional user input. This is sometimes referred to as data parsing. Many sources of data contain column entries separated by tabs. Clipboards from spreadsheets are typically stored this way. But even in that case, sometimes the first row is just numbers, sometimes it is column names.

Another common separator is the comma, so called CSV (comma separated values). Here the problem is that sometimes the commas are part of a label and

in Europe a comma is used as a decimal separator in numbers. Also, sometimes numbers are separated by spaces, so '4 2 2.3' are three numbers.

DataGraph tries to handle all of these cases automatically when you import a file or use the **Paste** action.

- **Separator**: If there is a tab character in the text, this is used as the separator. If there is no tab, DataGraph then checks for commas, semi colons or pipes (|) and then finally uses spaces.

- **Title row**: If the first line has no numbers, the row is viewed as a title row. Otherwise, the first line is considered a data row and the columns don't have any name.

- **Column type**: If DataGraph needs to create a table column to fit the input data, the column type is determined from the first row. If the first row is a number, the column that is created is a standard number column. Otherwise it is a text column. If this is incorrect, you need to convert the column. But if you are overwriting columns or appending rows, the existing types are used.

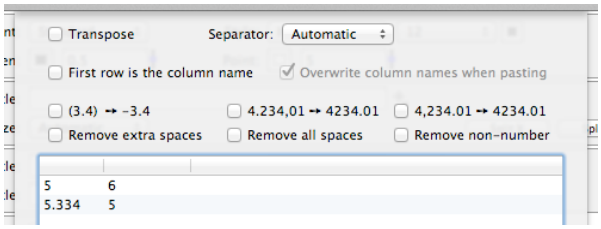
Paste Special

The paste and import functions use the rules explained above along with some automatic substitutions, to convert text blocks from files or the clipboard into a table of numbers or text. But sometimes these rules might not import the data in the way you would like. For example, what does 4,5 mean? DataGraph assumes that you are using a comma separator between two columns and would interpret this as the number 4 and the number 5; however, in many countries the comma is used to separate the decimal, and this is the number four and a half.

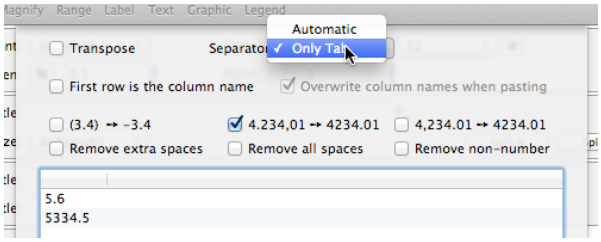
If these rules are not what you want, and that will unfortunately happen more often in Europe than the US, the solution is to use the *Paste Special...* option.

For example, take the following numbers
5,6
5.334,5

If you have these in the clipboard and you select **Paste Special...**, you will see the following screen, in which the data is interpreted as two comma separated columns.



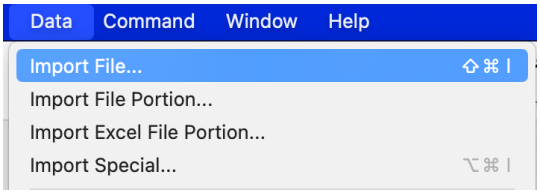
If the decimal mark here is the comma, there are two settings you need to modify in order to import these numbers correctly. These are illustrated in the following graphic. First, set the separator to 'Only Tab' so that the comma (,) will not be used as a separator. Next, click on the number conversion so that 5.334,5 is converted to the standard form for DataGraph. Note that you can still display numbers in the graph using , as a separator. You just need to adjust the separator setting in the style sheet.



Paste Special... also has additional features such as being able to paste in a transposed table. So if you have two rows with a lot of numbers, you can convert them into two columns. This is important since drawing commands refer to the columns and use all rows in that column.

Importing Files

Under the **Data** menu are four options for importing files.



Import File... will import the entire contents of a file. This option is similar to the standard paste, using the same rules to determine what separator is used (tab or comma). Any file type understood by DataGraph can be imported this way including: text, csv, Excel, Matlab (-v4), NetCDF.

Import File Portion is for text files where you want to to select a sub set of the lines in the file. This is helpful when you want to trim off a header or footer before importing.

Import Excel File Portion ... gives you a preview of Excel file before you import. You only need to use this option if you want to select a portion of a sheet. To learn more about importing data from Excel, see the **Excel File** section in this chapter.

Import Special... option is a very flexible import method, and gives you a tremendous amount of control over how data is imported.

If you drag and drop in a file on the data table, it is imported using the **Import File...** option. You can drag the file either onto the data table or into the list of columns. If you drag it into the list of columns it creates a group of columns.

Data Destination

There are two different import destinations, the data table and the column list. The column list only accepts a dragged file or columns dragged from other DataGraph files. This adds the content as columns or as a group of columns, and is explained at the end of this chapter. When you import a file, the content is either appended or overwrites data that is already there. There are three different ways that data is added.

1. If nothing is selected in the table, the data is appended as new columns. The exception is that if there are no rows in the columns that are displayed, the data overwrites the existing columns.

2. If you select columns (or no rows are displayed), the selected columns are overwritten with the data, and additional columns are inserted to the right of the last selected column. If the data source has column names, the existing column names will be overwritten. Note that drawing commands that use the selected columns will still refer to those columns even if the name changed.

3. If you select rows, the data is added to the columns and overwrites the selected row data. This means that if you want to add data above the first row, you first create an empty row at the top (menu entry in the Data menu) and then overwrite that row with the content. Any additional rows will be added below the last selected row. No columns will be added, and if there is a different number of columns in the incoming data, or the column names don't match, DataGraph will ask for a clarification. This is explained further in a later section with an example.

File formats

In addition to **Text** and **Excel** files (described in the previous sections), DataGraph can import other data types. The file is selected just like the text file. For file formats with binary data, DataGraph creates binary columns so no information is lost. The file formats that DataGraph 3.0 supports are:

- **Matlab**. Matlab is widely used in engineering and sciences, and you can save data in Matlab and import it into DataGraph. The key here is that you need to save the file with the -v4 flag. For example, if you want to save the workspace so that it can be loaded into DataGraph type

  ```
  >>save import.mat -v4
  ```

  Don't forget the -v4 flag. See help save in Matlab for more info about how to save only particular variables. All row or column vectors are read in as columns (binary).

- **netCDF**. DataGraph can import netCDF version 3 files. Files that contain 2D or 3D arrays will be imported as "flattened" arrays. All lists are read in as columns (binary).

- **JSON.** DataGraph can read files using the JSON format via a file link. These can be either online or local files. For more details see: **Link to Files**.

- **Plot**. This is for an earlier version of Plot, another plotting program on macOS. DataGraph has not been tested with files from the latest version of this program which is now called Plot2.

- **Cricket Graph**. Cricket Graph was widely used years ago, but the last OS that supported it was OS9. Intel machines cannot run the classic compatibility layer, but DataGraph can import these data files.

Import File Portion

The **Import File Portion** option can be used to select rows that you want to import from a text file. Select a portion of the file, then select File Import.

Importing Excel Files

It is easy to copy data from Excel and paste it into DataGraph, but DataGraph can also import data directly from an Excel file without requiring you to have Excel installed.

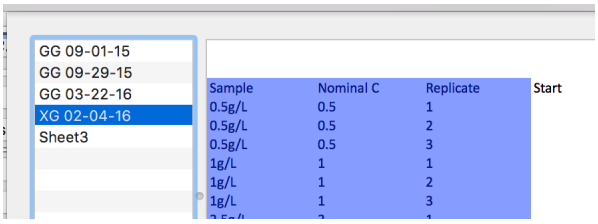
There are four different ways to import data from Excel:

1. From the Finder, you can control-click on a file and select **Open with > DataGraph**. This imports all the data from the file.

2. Drag an Excel file onto the data table or select **Import File…** from the Data menu and select the file. This imports all of the sheets from the file and appends each sheet as additional columns. Any rows or columns that you have selected will get overwritten in the same way as when you import a text file or paste data in.

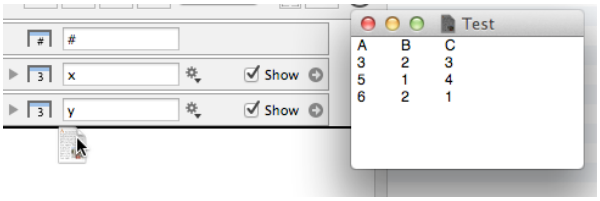
3. Drag an Excel file into the column list. This will create a group for the file. If there are multiple sheets in the Excel file, DataGraph will also create a group for each sheet. This maps the structure of the Excel file into the DataGraph hierarchical group structure.

4. You can import data from a single sheet by using the **Import Excel File Portion…** option in the Data menu. Either import all of that sheet or a rectangular region. This is particularly useful when the sheet contains a lot of header text or you don't need all of the values.



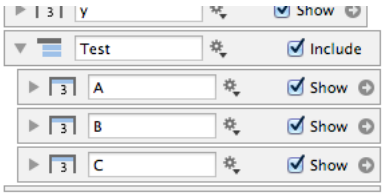
Dragging a File

If you drag a file onto a table, the table gets a blue highlight. If you release the file DataGraph responds in the same manner as when you select Import... from the data menu. This is handy when you see the file in the Finder. You can also drag a file into the column list.



This action works in a slightly different way. When you drag a file into the column list, you will see a drag line. If you release the mouse, the content of this file is

added as a group, with the columns coming from the data file. The name of the group comes from the file name.



If you drag a file onto a group, the content will overwrite the columns inside this group. This is the same as selecting all of the columns in the group and then dragging the file onto the table.

Import Special

DataGraph can automatically understand files coming from various platforms and using various separators (tab, comma, space delimited, etc.). In some cases, you may have more complicated data to import or you may only want to import a portion of data from a file. Rather than depend on other tools to manipulate your data into the appropriate format, you can use the Import Special method to define rules for how to import records from a file.

The DataGraph interface for creating custom import methods is called Import Special. When you create an import method, you can save the method to use on other files of the same format.

For examples showing how to use the **Import Special** interface, please see the following online documentation.

How to Create a Custom Import Method:
https://community.visualdatatools.com/datagraph/knowledge-base/import-special/

Connect to Files

For files that are either standard text (e.g., txt, csv) or a JSON file format, you can connect to them through a URL or by specifying a local file location.

You can connect to files that are comma separated or tab delimited text. Such files typically have the an ending of *.csv or *.txt, but you can use any file extension for text files.

You can also connect to files that are formatted using a JSON format. Note that JSON is a very flexible format and not all sources of JSON data may work in our

JSON importer. One example of a JSON protocol that works well with DataGraph is published by the World Bank.

For detailed instructions and examples see:
https://community.visualdatatools.com/datagraph/knowledge-base/link-file/

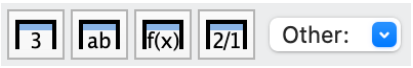
4. Working with Data

The data table gives you access to individual rows in a column. You can enter data, delete rows/columns, sort data, find data, and even have DataGraph read back the numbers to you.

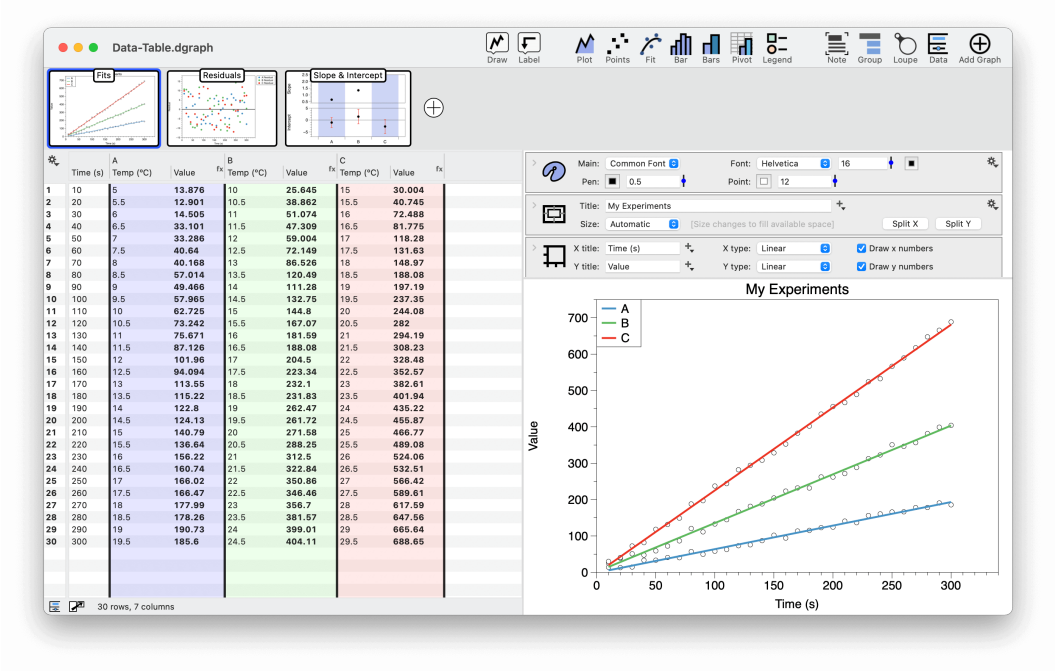
Data Side Panel

By default, the user interface includes a side panel on the left side of the screen that lists all the data in the file.

When the data panel is open, the top section shows a list of columns and/or groups of columns. Based on the icon, you can see the data type and the name of each column. The four most frequently used column types are: **numbers**, **text**, **expressions**, and **date**. To create an empty column, click on one of the buttons above the column list or select it from the **Other** menu.



As is the case in the screenshot below, this **Data Panel** can be hidden from view, giving more room to view the data table. To toggle the panel open/closed click the **Data** icon in the toolbar or use ⌘-**D**.

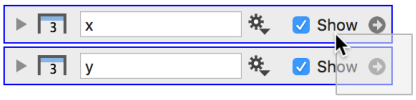


Column List

The objects in the column list, give you a quick way to organize re-order and group your data.

Selecting Columns

Selecting columns is similar to how you select drawing commands. If you click on the icon of the column entry, the border changes to blue to indicate that this entry is selected. Another way is to click outside the command and drag a selection rectangle to select multiple entries.



You can also add to the selection by holding down the shift key when you select. If you click outside of a column, or select a command the column is deselected. Now actions like cut, copy and delete will act on the selection and not just this column.

Deleting Columns

To delete the selection, select the column or columns you want to remove. Next, hit the delete key. If you delete a column accidentally, just select undo.

Reordering Columns

You can select and drag columns, or groups of columns, to reorder them in the list. When you reorder them in the list, the order is reflected in the data table. You can also click and drag column headers in the data table.

Copy/Cloning Columns

When a column is selected, you can use a standard Copy & Paste to make a copy. If you hold down the option key while dragging an option you can clone (i.e., copy) the current selection. You will see a green plus symbol. This is one way to quickly create a column with the same settings.

You can also drag columns or groups of columns between DataGraph files. For that you need to drag them from the column list and drop them into a column list in a different DataGraph file.

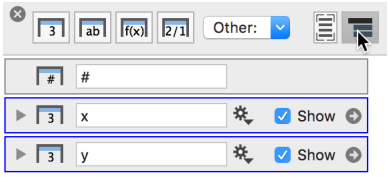
Data Groups

When a table becomes complex, or you have multiple logically separate clusters of columns, you can use the group structure to keep them together and make it easier to select columns in a drawing command.

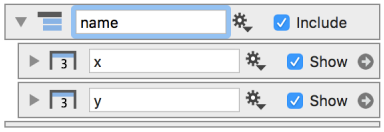
The Groups are somewhat analogous to tabs in a spreadsheet. The advantage, however, of using groups is that you can see all the data across multiple groups at the same time. It is also easy to navigate between multiple groups in the way that is more difficult with times.

Note that when you import a spreadsheet, the tabs in the sheet will be organized into groups.

In order to create a group, select the columns that you want to group and click on the Group icon.



The selected columns are now indented and you can add a group name.

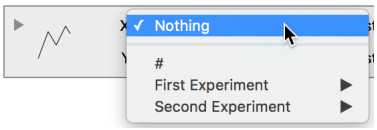


You can nest groups within groups. You can also drag entries in and out of a group.

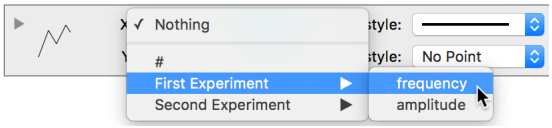
Select Grouped Data in Command Menus

The column selectors in the drawing commands will show the groups as sub-menus.

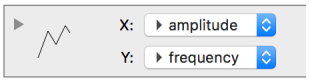
For example, if your data is in two groups, named ‘First Experiment’ and ‘Second Experiment’ when you create a new drawing command the drop-down menu for selecting the data to plot will show the group names. The small right facing triangle to the right indicate this is a group, as shown below.



Who you move the cursor over one of the group names, you will see the data columns inside those groups.



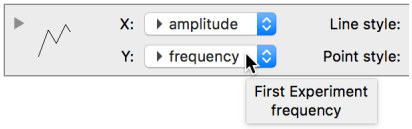
In the example above, the ‘First Experiment’ and ‘Second Experiment’ both have two entries, named ‘amplitude’ and ‘frequency’. Once you create a drawing command with a group entry, a small right facing triangle will show to the left of the column name, as shown below, to indicate the data is in a group.



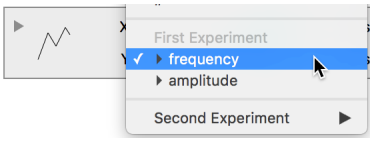
Showing the Group Location

To reveal the source of the data from any drawing command, you can:

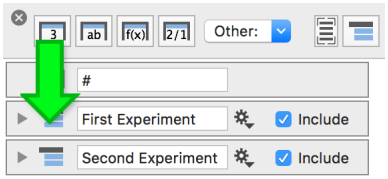
1. Hover over the name to display a pop-up with the group name displayed.



2. Click on the menu item. The group name is shown in gray and the column selected has a check mark next to it.

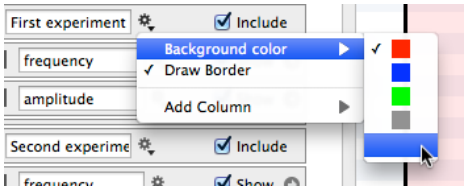


3. If you continue to hold the mouse in place over the menu item a green arrow will also appear in the column definition list pointing to the group or column name.



Background Colors

By default, a group has a different background in the data table to separate it visually. In the column definitions list, you can change this background color, or specify no color at all. This is done from the small action menu (gear menu) to the right of the group name.



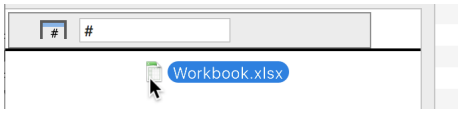
Organizing Groups

The same actions that work for selecting, deleting, and copying columns, also work for groups.

You can drag groups by clicking on the group icon and dragging. You can use that to reposition the group or drag it into a different DataGraph file to copy it. If you hold down the option key during the drag you will see a small plus button next to the mouse arrow that indicates that the group will be cloned when you release the mouse. Thus, you can replicate the structure of the table and then overwrite the content.

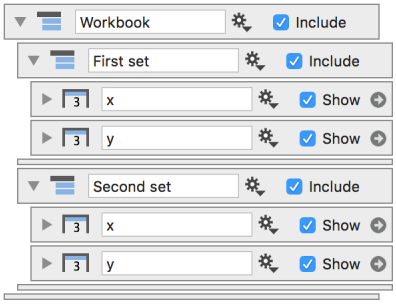
Importing into Groups

If you drag a data file directly into the column list, DataGraph will create a group with the name of the file and create columns in that group.



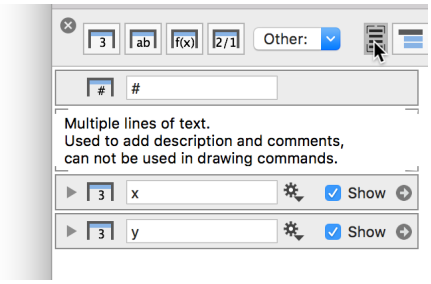
If you drag an Excel file that contains more than one sheet each sheet will be put into a separate group.

Note that if you drag the Excel file onto the table itself no groups are created and all of the columns from all of the sheets are imported.



Adding a Note

Allows you to add comments. Create a comment block by clicking on the icon next to the group icon. You can drag this comment around, but need to click on one of the four corners to select the box.



Copying Over Data

When you overwrite columns, the connections that you have made in drawing commands will not be broken. This means that you don't have to repeat any of the steps to convert data into a graphical/statistical representation.

Copying Columns & Groups

You can select columns or groups from the data side panel and copy them.

When you copy using the objects from the side panel, DataGraph only places the native DataGraph format in the clipboard. Thus, if you want to copy and paste data within the same or to a different DataGraph file, it is recommended to use the objects in the data side panel, as it will be much faster, particularly for large data sets.

When you select columns from the data table to copy, DataGraph adds a text representation of the data to the clipboard. This means that you can paste the data into a spreadsheet or different application, and the string representation will be used. However, if you have a few million entries, this can take a few seconds.

Copying Large Data Sets

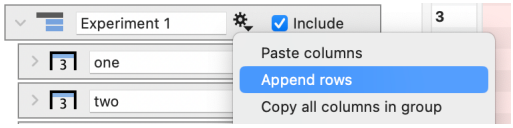
The clipboard on Mac OS X can have multiple representations. The application that receives the data determines which representation it uses. The default copy operation (⌘-C) copies two representations, One is the internal DataGraph representation that includes settings such as the column type. The other is a textual representation where the column title is in the first row and the columns are separated with a tab.

Note, that if the total size for the clipboard is larger than 40MB, DataGraph only copies the internal representation. This is done because most applications can't handle that large of a text block, and the assumption is that you are not moving the data out of DataGraph. If you want to export more than 40MB use the **File > Export Data** option to save it to a file or use **Edit > Copy as Text**, which does not copy the internal representation and has no cap.

The **Copy as Text** option is in the **Edit** menu, and does not have a shortcut key. This only copies the textual representation and is not limited to 40MB. In addition to allowing you copy large amounts of data into other programs, this command can be useful when you want to convert an expression column to text. **Copy as text** will effectively convert the result of the expression to a text column when it is pasted in either DataGraph or other programs.

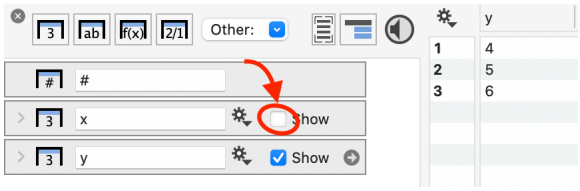
Paste into Groups

You can paste data into existing groups using the group object. First, copy the data to get it into the clipboard. Next, select the gear menu to the right of the group name. Select **Paste columns**, to add the data into new columns. Select **Append Rows**, to add the data to the existing columns.

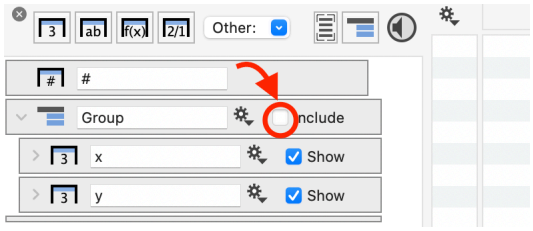


Hiding Data

Hide individual columns using the **Show** check box to the right of gear menu.

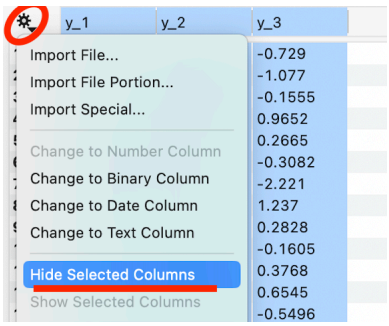


Hide an entire column group by deselecting the Include check box.



You can also hide data from the data table. You can control-click on individual columns and select **Hide column** from the bottom of the context menu that appears.

To to hide multiple columns at once from the data table, select all the columns we want to hide. Next click the gear menu in the top, left corner of the data table and select **Hide selected columns**.



Data Entry

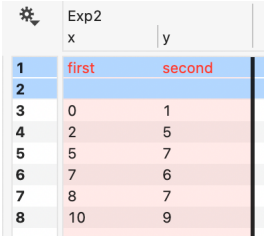
To start editing an entry, double click on it. If you press return (enter) you close this cell and start editing the cell right below it. If you hit tab you go to the next editable cell to the right or to the next line if you are at the end. By holding down the shift key when you hit the tab or enter key you go backwards to the next editable cell or up to the previous line.

Editing Data

When a column is visible, you can edit the individual values, but only for the number, text, date or binary columns. As you start using other columns, such as expressions, these derived values, are not editable, and are shown in bold in the data table.

Selecting Rows

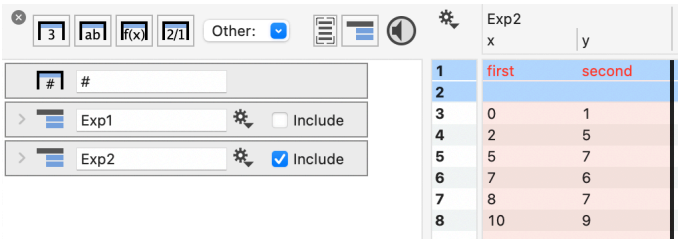
Click on any row in the data table to select. You can hold the Shift key to select a range or the Command key to select multiple. Select rows are highlighted.



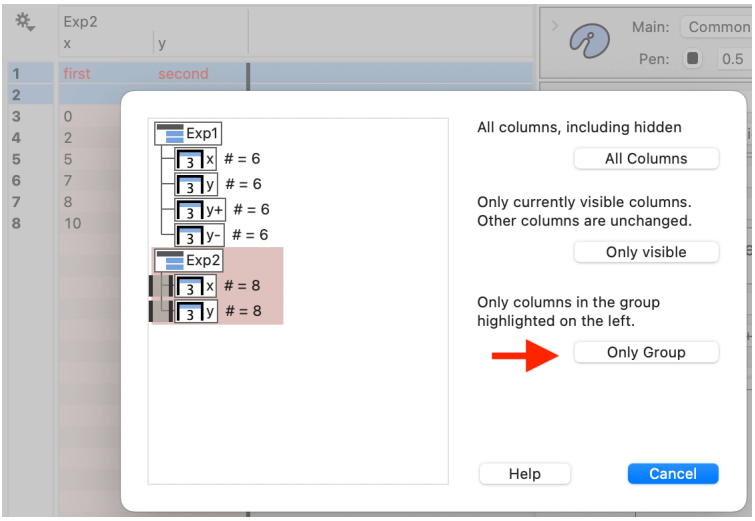
Deleting rows

You can also select rows and hit the delete key.

If you have hidden columns or groups, then a box will appear confirming what data you want to delete. For example, here we have two rows to selected and there is group of data hidden from view.



When you hit delete, you can choose to delete all the rows across all columns, only the visible columns, or the rows in the current group.

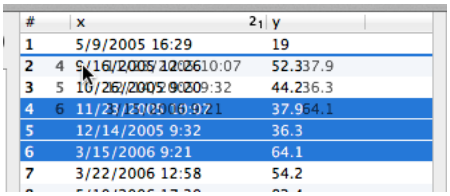


Thus, if you just want to delete the rows of a sub-set of columns, hide (uncheck the Show or Include box) for the columns you don't want to change and hit the delete key.

Reorder rows

The simplest way to reorder rows is to drag them around in the table. This is done as follows:

1 - Select the rows.
2 - Click and drag the mouse to the left or right.
3 - Drag the selection up/down.



When you start dragging, a gray version of the selection moves with the cursor. A blue line will show where this block will be moved when you release the mouse. If you hold down the option key, you clone the selection.

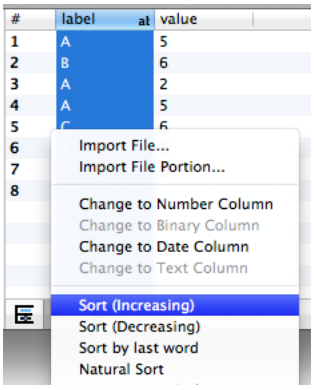
If you click and drag the mouse more up/down in step 2, the system interprets this as a change in the selection and does not drag the selection around. If you are not displaying all of the columns, the same window pops up as when you are deleting rows, since it is not clear from context which columns you want to be affected.

Reorder columns

You can reorder columns in the column definition list or in the table. To drag columns around in the table, just select the column and drag them. When you release the mouse, the column will be moved in the column list.

Sorting rows/columns

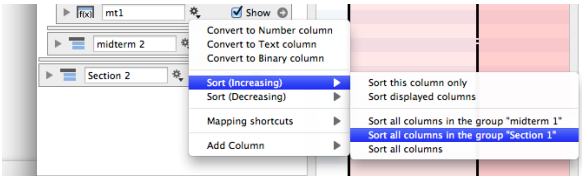
You can sort data based on a column. The simplest way to do this is to select the column you want to sort, and then go to the action menu below the table, or the Data menu in the main menu bar. There you can select the sorting method.



Note that the action menu gives you more options, and the available options depend on the column type. For example, the text column can be sorted according to last name and natural sort puts 'value 2' before 'value 10'.

The sorting options are also available in the context menu for a column. You get this menu by either control clicking (right clicking) on the column header in the table or on the small action menu icon to the right of the column name in the column list.

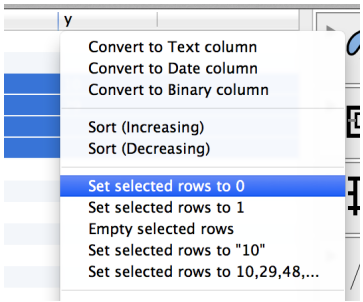
When you have groups, or not all of the columns are displayed, the action menu uses a sub menu for sorting. Using this sub menu, you can specify exactly which columns you want to include in the sort. This is important because sorting only the visible columns might corrupt the data. For example, say you have a class list of ‘student ID’ and ‘final grade’. You can hide ‘student ID’ and then sort the table according to name or a grade.



For example, say you have nested groups. The highest level is a group for ‘class section’. Inside that, you have groups for each midterm. Inside that, you have the grades for that midterm. If you select sort for the ‘final grade’, you need to specify what you want to sort. Here it makes sense to sort everyone in the section, since sorting the midterm group only will certainly corrupt the data.

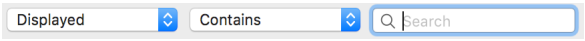
Editing Shortcuts

Shortcuts are available through the context menu. You get to the context menu by either right clicking or control clicking. If you select rows and control click on the column header you trigger the context menu for that column. This allows you to fill the entries with a fixed value, or use the first two rows to fill out the remainder of the row selection. This only affects the column that is selected.



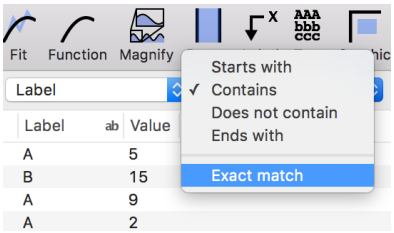
Find & Filter

DataGraph has a find action, triggered with ⌘F. This command provides the user a search bar above the data table that can be used to filter the display of the current table based on a search string.



Type in the search string to search all the data shown in the data table. To specify one column to search, change the left-most drop-down menu from 'Displayed' to the column name.

By default, the search returns any rows that contain the search criteria but you can also specify the type of search to conduct as shown below.

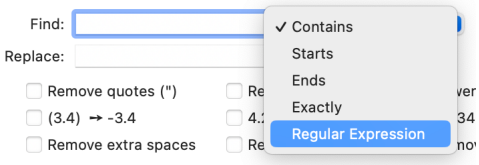


Using Find action, does not affect any drawing or change the data, but you can select rows to copy/cut or delete them. You can also edit individual cells in the selection. Note that this command works on all the data columns that are displayed in the data table. To end the search and restore the data table click the Done button in the search bar.

Find & Replace

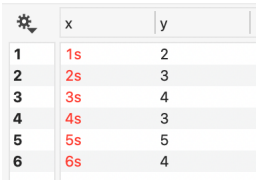
DataGraph can also be used to find and replace data as needed. For example, if you have extra text in a number or quotes where they shouldn't be you can easily isolate this data and update the data table.

By default, Find and Replace will search for any cell that contains type in text. You can also search for entries that start, end, or are an exact match. There is also the option of using Regular Expressions.



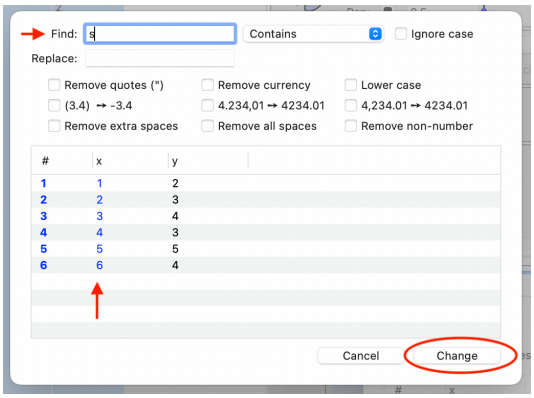
First, you should select the columns or rows that you want to modify. Next, open the find and replace window, you can either hit ⇧⌘R or select **Edit > Find and Replace.**

Find & Replace shows you a preview of how your selected options will change the selected data. For example, the entries in column x are shown in red, as they are not understood as numbers given the letter 's' in each entry.

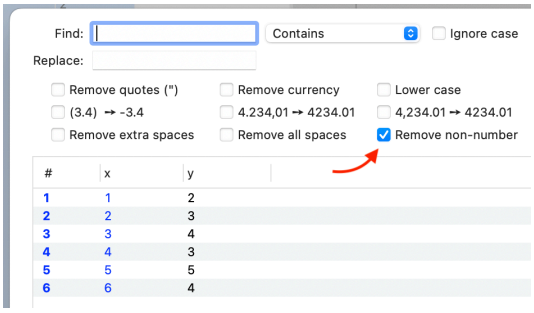


	x	y
1	1s	2
2	2s	3
3	3s	4
4	4s	3
5	5s	5
6	6s	4

If "s" is entered in **Find** and nothing is entered in **Replace**, you can see the result of this action. Hit **Change** to apply.



There are also several shortcuts for common actions. For example, we could achieve the same result as above by checking **Remove non-number**.



Entries that are changed are displayed in blue, unchanged entries in black. You can pick several filter options, as well as using a find-replace string match.

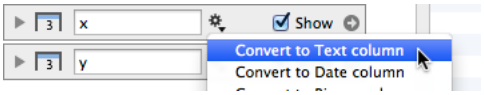
5. Column Types

The number, text, date and expression are by far the most frequently used column types. The Other menu has additional types, which will come in handy as you start using DataGraph for more advanced actions such as extracting residuals from function fits, etc.

The small disclosure triangle in the top left corner of each column type reveals additional settings. For some like the number column, the settings are for optional formatting. For other columns, such as date and expression, these settings are important input.

Converting Between Types

Sometimes you have a column that doesn't have the proper type. Maybe you pasted in a column, or began with a standard numerical column and want to convert it into a label column. Instead of creating a new column, you can convert the type of the existing column using the gear menu, as shown below. When you convert a number column to a label or date, the entries stay exactly the same.



In addition to the four primary column types, many other types of columns can be used to process and manipulate data in various ways. This chapter starts by explaining how to organize your data, followed by a section on each column type.

Primary Column Types

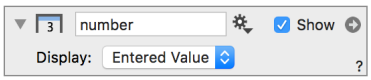
- Number — For standard numbers like 3.2 or 1e-5. Also, for expressions like 3/4.
- Text — Used for labels, names, etc. You can also assign numerical values to the labels.
- Expression – For performing calculations.
- Date — Convert calendar dates into numbers that can be used in a graph or for statistics.

Other Column Types

- Binary — Numerical column type for importing data from programs like Matlab with no loss due to rounding.
- From Command – Extracts values from a command.
- Text expression – Combine strings from text columns.
- Plot Action – Apply actions on x-y data.
- Interpolate by category – Interpolate between points in a given category of data.
- Map – Map text of numerical values to other values. Can define mapping within the column or use other columns to map data.
- Mask – Use conditional statements to hide row data.
- Redirect – Use to create an alias for a column.

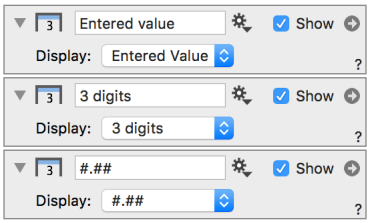
Number

The **Number** column is a standard column for numerical data. Below is a **Numerical** column entry as it is shown in the column definition list.



When you create a numerical column, you can enter numbers in the data table in the form 3.41, 3.51e3 etc, but you can also type in expressions like 5/2, sqrt(5), 5e, ... If DataGraph cannot understand the expression it will display the content in red.

You can change the format of the data shown in the data table using the **Display** drop-down box. For example, three **Number** columns are shown below that have varying options for **Display**.



Below, the same exact values are entered into each column, with the 'Entered value' column showing exactly what was entered. The other two columns show the numerical representation of the same data rounded in different ways.

#	Entered value	3 digits	#.##
1	4.56788	4.57	4.57
2	5E-2	0.05	0.05
3	4π	12.6	12.57
4	sqrt(3)	1.73	1.73
5	4 44	4 44	4 44
6	none	none	none

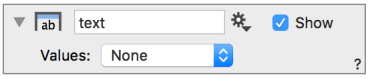
When you change the display, the entered valued is always stored. You can click on the cell to reveal what was originally entered, as shown below, or you can always change the **Display** setting back to **Entered Value**.

3 digits	#.##	y
4.57	4.57	
0.05	0.05	
12.6	4π	
1.73	1.73	

By using variables (See the **Variables** chapter for more information), you can define mm = 0.001, cm = 0.01, m = 1 so you can enter in 5m, 44cm, 22mm and this will be understood as 5*m = 1, 44*0.01 = 0.44 and 22*0.001 = 0.022.

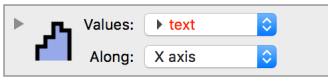
Text

The **Text** column is intended for labels or categorical data. Below is a **Text** column entry as it is shown in the column definition list.

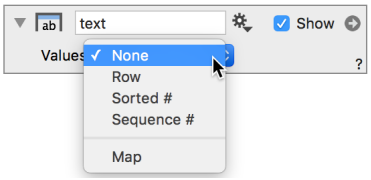


When you import data, DataGraph tries to determine whether or not the data is text. If you end up with a column where all the entries are red, this is most likely because you are using a number column. You can convert to a **Text** column using the gear menu to the right of the column name.

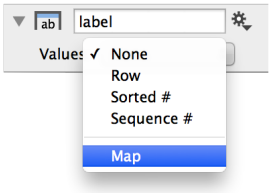
When the **Values** drop-down box is set to 'None', you will simply see the text entries in the data table. You can also map the text entries to numerical values. This can be useful since many drawing commands require a numerical value. For example, if we try to create a histogram using a **Text** column it will show the column name in red (see below).



There are several ways to define this mapping, which are listed in the Values drop-down box.

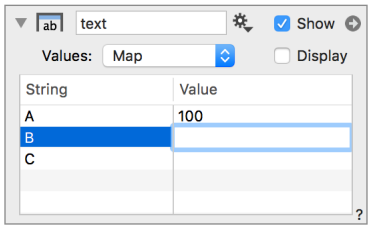


For example, say you have a table with entries that include ‘A’, ‘B’, and ‘C’. ‘Row’ simply returns the row number, #. ‘Sorted #’ will assign ‘A’ to 1, ‘B’ to 2, and ‘C’ to 3. ‘Sequence #’ means that the value is assigned based on when the label first appears. For example, if the rows are C,A,C,B the values are 1,2,1,3.



The ‘Map’ option allows you to specify the mapping exactly by assigning values to each label. For example, take a text column with the entries C,A,C,B. When you select ‘Map’ from the **Values** pull down menu, a mapping table appears. This table is initially empty and you have to specify the value for each label.

To simplify that task, you can ask DataGraph to populate the left column in the mapping table with the unique entries from the data table. First, select ‘Make sure all entries are mapped’ from the gear menu. After that, all the unique entries from the data table are automatically populated in the String column, as shown in the following image.

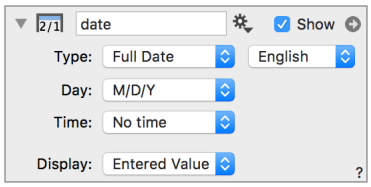


Finally, you can type the numerical Value for each String. In this example, the label 'A' is mapped to the numerical value '100'. The gear menu has additional menu items to simplify this process and automatically assign values, sort entries , etc. To show the numerical values in the data table, simply click the **Display** check-box.

When a text column has a numerical mapping, it can be used where ever DataGraph expects a numerical column. For example, you can use this to map grades into a numerical score for computing averages.

Date

The **Date** column type is intended for calendar dates and understands a variety of formats. Below is a **Date** column entry as it is shown in the column definition list.

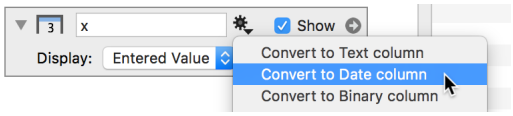


Instead of trying to guess that a numerical column is a date and require you to use a particular format for dates, DataGraph asks you to make that explicit by converting the column into a date column and specifying the format.

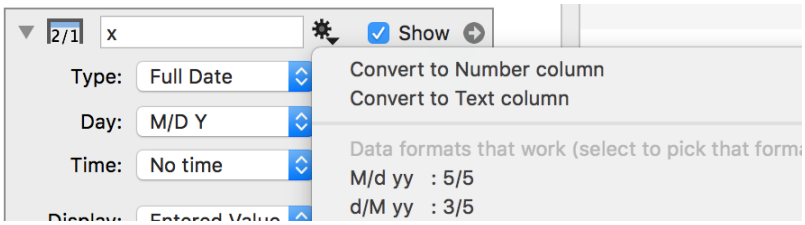
For example, if you have five numbers in a standard numerical column of the form month/day year, these entries are not understood as numbers and will show up in red.

#	x
1	12/2 2015
2	4/5 2015
3	6/15 2015
4	7/9 2015
5	10/31 2015
6	

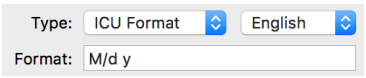
To convert this column to a date, use the gear menu to the right of the column name.



When the column is converted, DataGraph makes a best guess at the date format. If the format is not correct you can select the gear menu again. Then you will see a list of possible date formats to choose from that DataGraph has selected as working with your data.



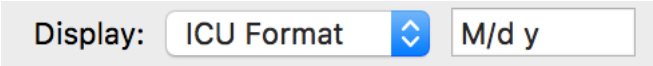
You can also specify the date format manually. There is a long list of built-in formats in the **Day** drop-down box. For expert users, you can specify a format string for formats that are not already supported. Specifically, in the **Type** drop-down box select 'ICU Format'. ICU stands for International Components for Unicode and provides standard methods for specifying date and time formats.



After changing the **Type** to 'ICU format', a **Format** entry box appears, as shown above, where you can specify the date format. If you hover over the **Format** field the following help will appear summarizing the ICU format string options.

ICU format string. Click on the question mark in the lower left corner to find a more detailed description, but a fast description is:
y, yy, yyyy = year
M = Month as a number
MMM = month as a string
d = day
h = hour in AM/PM 1-12
K = hour in AM/PM 0-11
H = hour in day 0-23
k = hour in day 1-24
m = minute in hour
s = second in minute
S = millisecond

Using the **Display** drop-down box, you can specify a different display format. The original data will be saved as this only changes how the date is shown in the data table.



Binary Column

This column stores numbers as IEEE binary numbers. This differs from the number column (discussed previously) that stores the textual representation entered in a column and converts that to a number when you open the DataGraph file. For example, if you enter '1/10' in a binary column it gets converted to 0.1 immediately.

When you import a Matlab binary file (saved with the -v4 flag) DataGraph creates binary columns so that the import is completely loss less. To convert a number column to a binary column, use the small gear menu to the right of the column name.

Expressions

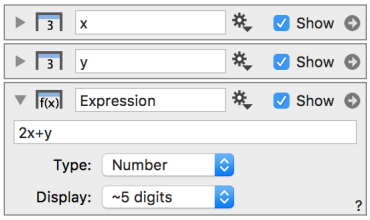
A standard spreadsheet typically handles expressions by referring to individual cells of the form A5, A10, B4 where A,B,C... indicates the column and the number indicates the row. This gives you a way to reference any cell, but there are problems when you have a lot of rows and it is sometimes hard to know what each cell does.

DataGraph handles this completely differently. Some actions are done by using drawing commands or columns, such as the plot action column, but the main workhorse here is the expression column.

Syntax

The expression column uses arguments that are either columns or variables. An expression like 2x+y, where 2x is understood as 2*x, has two arguments 'x' and 'y'. If x and y are names of numerical columns, the expression column will use the rows where the x and y columns have values.

For example, below is an expression column named 'Expression' using x and y.



In the Data Table, the result of the expression is shown.

#	x	y	Expression fx
1	1	2	**4**
2	2	5	**9**
3	3	1	**7**

If one of the arguments is not found as a numerical column, DataGraph will look for a variable with that name. If x is a column and y is a constant, the expression is evaluated wherever x is defined and y is the same for each row.

Column names that contain a space must be placed in double quotes in order to refer to them as arguments in an expression. Note that argument names cannot start with a number, since 3x is understood as 3*x. If you have a column that is called "3x", you can use this in an expression, but will need to enclose it in quotes, e.g., y-"3x". This allows you to also include spaces and even operators, so "3x - 5" + "another column" will subtract the two columns named "3x - 5" and "another column".

DataGraph uses the following symbols for standard operations: plus '+', minus '-', multiply '*', divide '/', and raise to a power '^'. DataGraph also contains a number of built-in functions and logical operators.

NOTE: For more information on built-in functions, see the Function Reference in the Appendix.

The constants π, pi, e, ∞, NaN, inif, Inf are defined, and can be used in expressions. For example, x*π/180 converts from degrees to radians. The keyboard short cut for π is option-p and for ∞ is option-5.

Column References

In the previous example, it was clear that there was one x column and one y column that were being used in the expression. However, DataGraph allows you to have multiple columns with the same name and to nest the columns using groups. When you use arguments in an expression, DataGraph searches the columns, groups, and variables based on a specific set of rules and picks the first one that matches.

The search order is as follows:

1. At the same level as the expression, search through all columns in the order listed, while also searching through each group or subgroup.

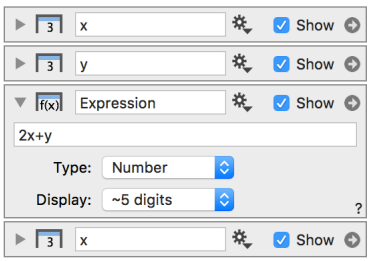
2. If no match is found and the expression is nested within a group, go up one level to the parent group and search through only the number columns in this group in the order listed. Do not go into subgroups.

3. If no match is found and you are not at the top level, move up one level and search as explained in step 2. Continue up until you have checked the top level.

4. See if the variable exists as a variable.

5. If the variable is not found, the expression is considered invalid and DataGraph puts a red dot in the top left corner of the column definition.

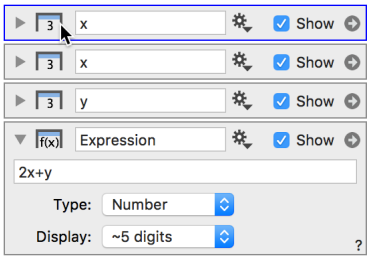
Building upon the example from the previous section, we can add another column 'x' to the end of our column definition list.



In the Data Table, we set the new 'x' column equal to all zeros. Since the expression uses the first x column listed, the value of the expression does not change.

#	x	y	Expression	fx	x
1	1	2	4		0
2	2	5	9		0
3	3	1	7		0
4					

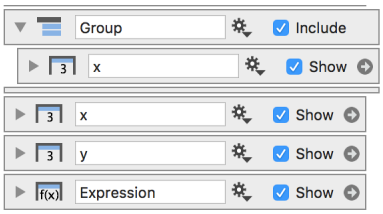
Now, let's move the new x column to the top of the list.



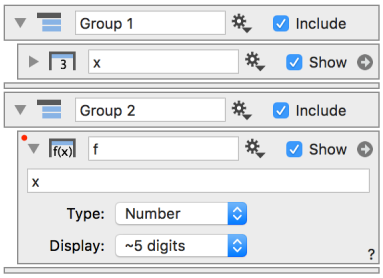
Notice how changing the order in the column definition list automatically updated the location of the column in the Data Table. The expression column will now use the first x column listed. As a result, the value of the 'Expression' column changes.

#	x	x	y	Expression fx
1	0	1	2	2
2	0	2	5	5
3	0	3	1	1

As noted in step one of the search order, DataGraph will search through groups and subgroups that are at the same level as the expression. Thus, if we put the first x listed in a group, our expression will still use this x, even though it is in a group.



Based on the search rules discussed above, columns that are in a different group nested at the same level would not be found. For example, in the following screenshot, the f function cannot directly refer to x because f and x belong to different groups.

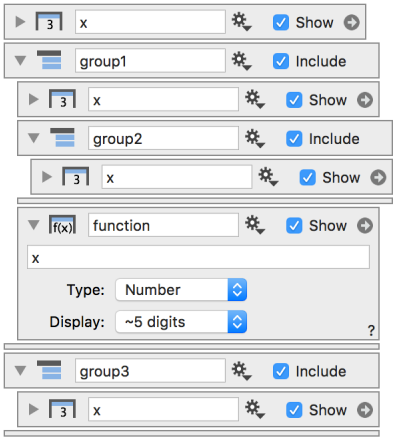


In this case, we could use path references to refer to columns outside of the search path, discussed in the following section. You could also use a Redirect column, using the x column from Group 1 as input, and the output name can also be 'x'.

Path References

It is possible to refer to columns in a specific group by using a syntax similar to path names in unix: '.' refers to the current level and '..' refers to the parent group, one level up. You can refer directly to the top level using '/'.

For example, consider the following three groups and four data columns all named x. An expression column is nested contained in group1 that returns the value for x.



Inside the expression, x alone will refer to the x column inside group1, the second x. The following table explains how to refer to any of the x columns using relative or absolute references.

Reference	Location
x	second
../x	first
/x	first
./x	second
/group1/x	second
/group3/x	fourth
/group1/group2/x	third
./group2/x	third
../group3/x	fourth

The # column

The row number column, named # by default, is a special column. The column gives the row number corresponding to the data shown in the data table but has no inherent length of its own.

If you use an expression that only has the # column and variables or constants there is no way to know how many rows should be in this column. When this

happens, DataGraph adds a field to the expression column definition called **# of rows**.

The number # column can be used to generate a list of numbers. For example, the following expression would generate a list of 10 odd numbers: 1, 3, 5, 7, … , 19

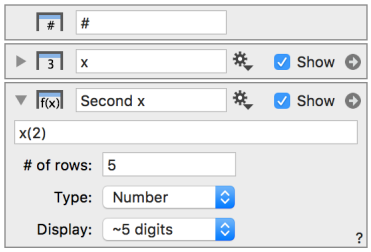
If you use the expression '#*x' and x is a column, the **# of rows** field is not displayed since the length can be deduced from the x column. Putting 10 in the **# of rows** field specifies the length. This field can use a variable, so you can specify the length as 'N' as long as one of the variables is called 'N'.

The # column can also be used to refer to the column as a subscript, which is described in the next section.

Referring to rows

DataGraph allows you to refer to a specific row in a given column by using the column name followed by the row number inside parentheses, what is referred to as a subscript operator.

For example, the expression column below, named 'Second x' is set the value of x at row 2, or #=2. Since the length cannot be deduced from the expression you need to specify the length.



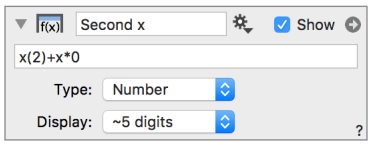
If our x data is simply the list 1,2,3,4 the result of this expression is the number 2.

#	x	Second x fx
1	1	2
2	2	2
3	3	2
4	4	2
5		2

The number in the parentheses is referring specifically to the row number. Thus, if we sort the x column descending the value of 'Second x' changes accordingly.

#	x	Second x fx
1	4	3
2	3	3
3	2	3
4	1	3
5		3

If we wanted our expression column to have the same length as the x column we could simply add x*0 as shown below. Since the length of the column can be deduced from the x column the **# of rows** field is not used or displayed.

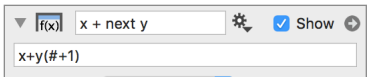


If we modified our expression to be x(#), that is 'x(#)+x*0', note how now we just return the value for x at that row number.

#	x	Second x fx
1	4	4
2	3	3
3	2	2
4	1	1

Next, consider an example where we want to stagger the rows that are used in an expression. By default, the expression x+y uses x and y from the same row. In spreadsheets, if x is column A and y is column B, this means that the rows are evaluated by using A1+B1, A2+B2, A3+B3,… But what if you want row 1 to be A1+B2, row 2 to be A2+B3, etc. That is, you want to refer to the next row in the column, not the same row.

In DataGraph, you can specify the row number by using the subscript operator. To get the row number, use the row number column #. The following expression adds x(1)+y(2) and x(2)+y(3)….

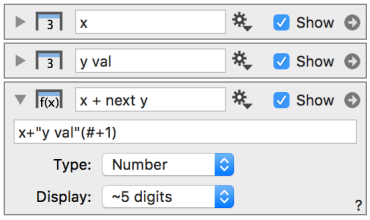


An example result is shown below.

x	y	x + next y fx
10	1	12
20	2	23
30	3	

Note that for the last row y(#+1) is not valid since it is out of bounds. That just means that the result there is empty.

To refer to a column that has a space in the name or cannot be understood as a constant for other reasons (e.g., has a + in the name), put quotes around the name, but not around the parenthesis.

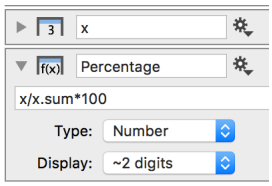


In DataGraph, a column cannot refer to itself, since that will lead to an infinite recursion. In spreadsheets, this approach is often used to do an incremental sum. That is, column B is defined such that, row 1 is A1, row 2 is A2+B1, row 3 is A3+B2, etc. In DataGraph, this type of summation is accomplished using the 'isum' column property, explained in the next section.

Properties

Several built-in properties are defined for every numerical column, including expression columns, plot action columns, or columns coming from drawing commands. They are referred to by using the 'name.property' syntax, where name is a column name.

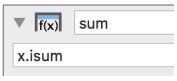
Some column properties are single values or constants for a given column. For example, x.sum is the sum of all values in column x. You can use this inside an expression. For example, the expression shown below uses the x.sum to calculate the percentage.



x	Percentage
2	**9.1**
4	**18**
15	**68**
1	**4.5**

Note that the format in the above table only shows a few of the digits, but the full resolution is used in any drawing commands or expressions that use this column. Also, for percentages, to round percentages in such a way that the sum of the rounded numbers is 100% use the Pie command.

There are also a number of column properties that return a column of numbers. For example, the 'isum' property is the incremental, or accumulative, sum of a column.



x	Sum
2	**2**
4	**6**
15	**21**
1	**22**

This is the incremental/accumulative sum of a column. You can also use column properties in more complicated expressions, such as 2*x.isum+1. You can even refer to particular row values of a property, such as x.isum(#-1).

You can also use column properties as a subscript, or row reference inside expressions. Consider the sortwith property, a column property that returns the row number for sorting. That is, x.sortwith returns the row number of each entry in x that corresponds to a sorted x. That means that x(x.sortwidth) will sort the x list, and y(x.sortwith) will apply the same reordering to the column y.

x	y	x(x.sortwith) fx	y(x.sortwith) fx
4	5	**1**	**4**
3	2	**2**	**3**
5	1	**3**	**2**
1	4	**4**	**5**
2	3	**5**	**1**

For more information on column properties see: The Column Property section of the Appendix.

The full list of columns properties can be found online at:
https://community.visualdatatools.com/datagraph/knowledge-base/properties/

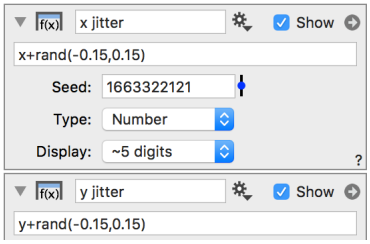
Random Numbers

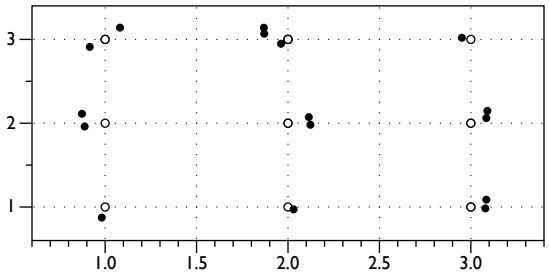
DataGraph includes support for pseudo random variables. They are pseudo random because given the same starting seed they will produce the same sequence every time. There are two random number functions, rand and nrand. The function rand creates a uniformly distributed random number. It takes in two arguments, the minimum and maximum values that define the range. The function nrand creates normally distributed random numbers. If you use a single argument, i.e., nrand(a), you get a normally distributed number with standard deviation ‘a’ and mean 0. If you use nrand(a,b) you get a normally distributed random number with mean ‘a’ and standard deviation ‘b’.

When you use either of the random number generators, a new field is added to the column definition containing the seed for the random number generator. Each time you add a random number column, the seed itself is created using a different pseudo random number generator. You can change the seed to any value you prefer. Once set, the seed will not change. Thus, the next time you open the file the seed will be the same so the random sequence will be the same. This is why it is referred to as a pseudo random number generator, since it

is a repeatable sequence of numbers that behaves statistically like a random number sequence.

An example use for a random number generator would be to add a slight change, or jitter to data for plotting purposes. Consider a case where the x and y values are integers between 1 and 3. There are only nine different combinations possible. If you draw the x,y values as points you will only see these nine points. If you add a random jitter to x and y and draw the jittered result you will get point clouds around each point. Thus, you can see where there are many points close to a certain combination.





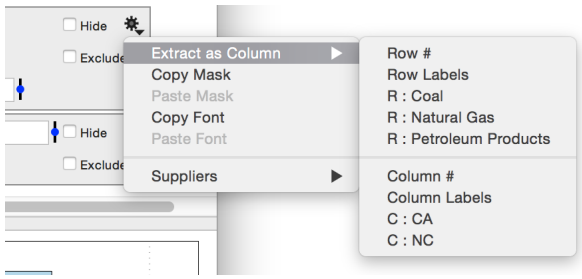
The random number generator used is called the Mersenne Twister, a very widely used pseudorandom number generator. For more information, see Wikipedia at https://en.wikipedia.org/wiki/Mersenne_Twister.

From Command

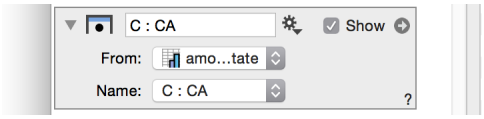
This column type allows you to extract a column of data from a drawing command. Thus, you can use the output from one drawing command as the input to another drawing command.

Typically, you create this column by using the gear menu on the right hand side of the drawing command. There are several drawing commands that have extractable data including: the **Pivot**, **Histogram**, **Box-plot**, and **Fit**.

In the following graphic, the gear menu is being used to extract an individual column or row from a *Pivot* command.



This will create a **From Command** column in the column definition list, as shown below. Once the column is created, you can use the **From** drop-down box to change the drawing command from which you are extracting data and the **Name** drop-down box to change the specific column extracted.



By default, the label assigned to the **From command** column matches the name from the drawing command, but the label can be changed in the column definition list. Note that if you change the selection in the **From** drop-down box to a command that does not have an extractable column matching the entry in the **Name** drop-down box, the entry in the **Name** drop-down box turns red (to denote the mismatch) and the column is empty.

The column type assigned to the **From Command** depends on the type of data extracted. For example the category names for a **Pivot** command are text labels. If you select that data to extract the column changes into a text column.

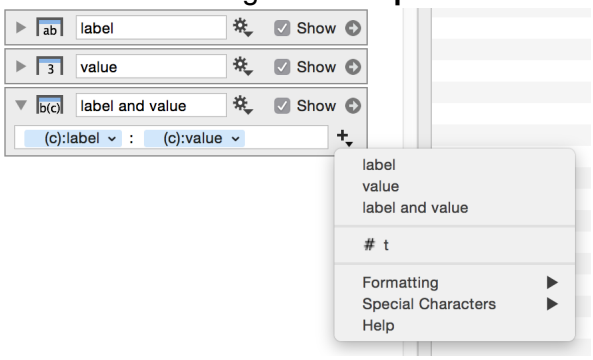
The content in a **From Command** column changes dynamically. Thus, if the input to the drawing command changes the result in the extracted column changes accordingly. This allows you to use DataGraph as a programming environment. For example, you can take a pair of columns, use the Fit command to compute the closest fit, extract the residual back as a column by using the **From Command** column, then pass that residual into a **Histogram** command. You could then take the result from the **Histogram** command back into the data table and continue.

Note that if you want to only extract a single number from a drawing command you can do that by using a variable and use the **From Command** variable type.

Text Expression

The **Text Expression** column can be used to combine columns and variables into a text column. It is similar to using tokens in a text field (See section on **Tokens** in the chapter on **Drawing Command Elements**), except that you can select other columns as well as variables. This column type is useful for labels, such as hover labels or labels for points and bars.

In the following example, a **Text** column ('label') and a **Number** column ('value') are combined using a **Text Expression** column ('label and value').



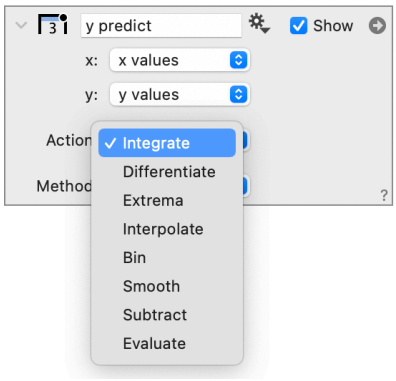
An entry in the **Text Expression** column is created in every row where both values are defined. Thus, if any of the entries in either the 'label' or the 'value' columns are empty the result in the 'label and value' column is also empty, as shown below.

#	label	value	label and value
1	A	3	**A : 3**
2	B	2	**B : 2**
3	C		
4	D	5	**D : 5**
5		4	

Plot Action

The **Plot Action** column allows you to compute/derive other mathematical values from data, using one or two columns as input. Inside this column, you can specify computational actions that can be described as taking an xy plot as input, accepting additional arguments as either columns or numbers and returning a single column. Rather than creating different column types for each action, they are all grouped together into a single column type.

To use this action, the first step is to select columns for the xy plot. This requires two columns. The next step is to select the computational action from a menu, as shown in the following screen shot.



The entries below the **Action** menu change depending on what you select. In the following sections, each selection under the **Action** menu is explained.

Integrate

$$F_n = \int_{x_0}^{x_n} f(x)dx$$

The **Integrate** action computes the accumulative integral. Specifically, for a particular row the value is the integral of the xy plot up to that row. As demonstrated below, there are three options for how the integral is calculated: Trapezoid, Left and Right.

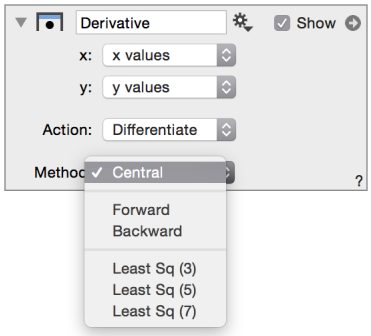
x values	y values	Trapezoid	Left	Right
0	1	**0**	**0**	**0**
1	2	**1.5**	**1**	**2**
2	1	**3**	**3**	**3**
4	2	**6**	**5**	**7**

Note that the first entry is always 0. For each interval, the integral changes by what is considered the integral for that interval. The first interval in the table is between 0 and 1 in the x direction. The y values are 1 and 2, so the trapezoid approximates the integral by computing width times the average of 1 and 2. The Left option uses the value at the starting point and multiplies the width. The Right option uses the value at the ending point and multiplies by the width.

The trapezoid method is more accurate when you have a smooth function, but the left and right options are useful when you want to use this method to compute an accumulative sum. In that case you would use the row column (#) as the x coordinate.

Differentiate

Computes the derivative at each x location. There are several ways to approximate the derivative.



The central derivative approximation uses a second order polynomial through the current point and the point before and after and differentiates that approximation. If the x values are uniform it simplifies into what is typically called the central derivative approximation, namely for row i the derivative is computed by

$$\frac{y(i+1) - y(i-1)}{x(i+1) - x(i-1)}$$

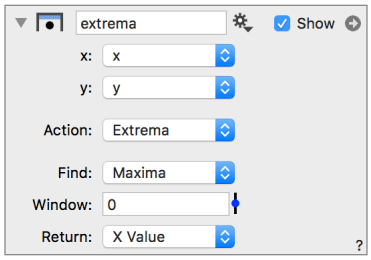
Forward and backward use one sided derivatives. They are less accurate for smooth functions. The Forward difference for row 'i' is computed by

$$\frac{y(i+1) - y(i)}{x(i+1) - x(i)}$$

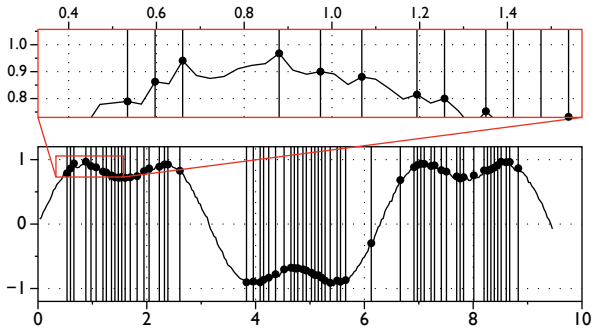
The least square options are computationally a little bit more involved. They use three, five or seven neighboring rows (one, two or three before and after), compute a least squares linear fit to the (x,y) values at those points and then return the slope of that line.

Extrema

To find the overall maxima or minima for a column, you can use a **Number from Column** variable; however, if you want to find local minima or maxima you can select the 'Extrema' option from the **Action** menu.

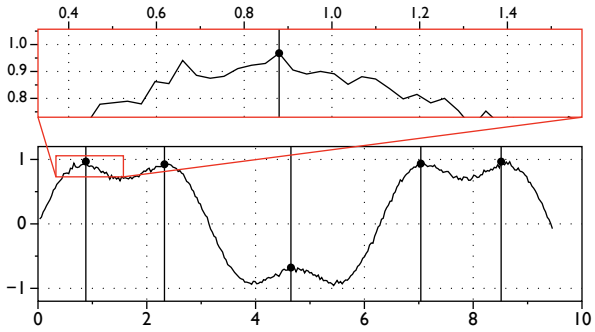


When the window is 0, the action finds every local maxima. A local maxima is defined as a row where the previous row is less than or equal and the next row is strictly less. The output column has a non-empty row at the maxima, and you can choose to either return the x or y entry at that maxima. If the y column has a lot of noise you get a lot of local maxima as is shown in the next figure. The graph is created by using the result from the Extrema action in a Lines command to draw the horizontal lines associated with each maximum. The top graph is an inset that is zooming in on a range with a lot of maxima.



The window setting can be used to specify a minimum distance between the extrema. In this case, the maximum has to be both a local maximum and no value within the given window can be larger.

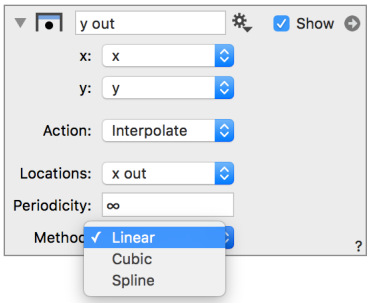
When the window is set to one, the local maxima due to noise are removed, as shown below. The local maxima at x=1.2 is not included because the value at 0.9 is larger.



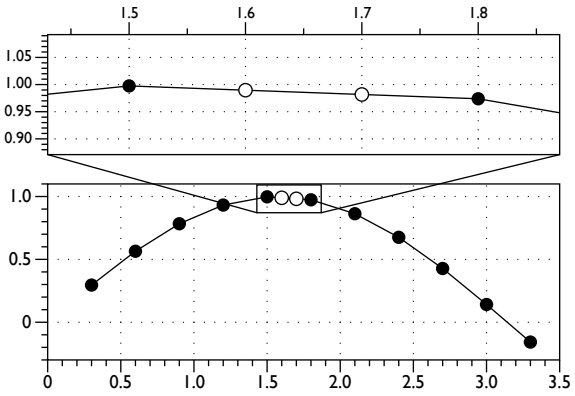
Interpolate

The **Interpolate** action allows you to use known data to approximate values at locations where the data is not known. For example, if you are given a sampling of a function at points x1,…,xn, you can approximate at points in between those values. There are multiple ways of interpolating between data points that typically involve approximating a function by fitting a line or polynomial through neighboring points. The function can then be used to approximate values at locations were there is no data.

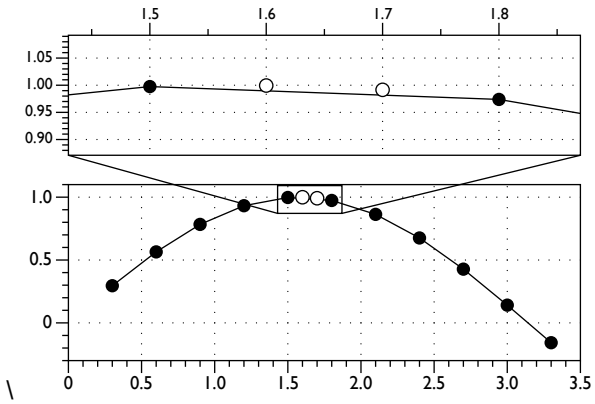
The **Interpolate** action allows you to choose between a number of different approximations methods as shown below.



You select the x locations where you want to interpolate the x,y data sets as a column. The linear option means that the approximation uses the two closest points and approximates the function with a straight line between the points.



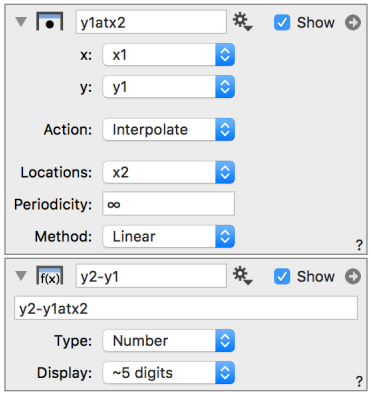
The cubic option uses four points to fit a third degree polynomial through the points and evaluates that polynomial at the point.



Spline uses a cubic polynomial on each interval such that the second derivative is continuous across the interval. This is same as the spline option in the Plot command.

Example:

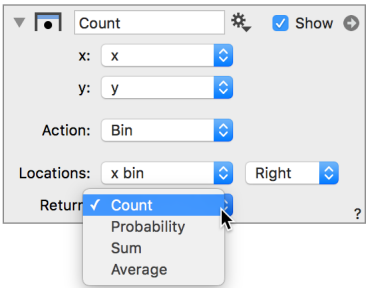
If you have two data sets (x1,y1) and (x2,y2) and want to subtract the two y values, you can use the expression column as long as the x1 and x2 values are the same and aligned. If the values for x1 and x2 are not all the same locations, you can use the **Interpolate** action to compute the y1 values at the x2 locations.



Bin

In order to group data based on either numerical or categorical groups, DataGraph contains drawing commands such as the Histogram and Pivot command. These drawing commands also allow you to visualize and extract the results of the binning; however, there may be some cases where you want to bin a set of data without plotting or you want to use non-uniform bin sizes. In these cases, the **Bin** action can be used to analyze data while specifying the bin locations directly.

You must have a column set to specify the locations of the bins and whether these locations define the Right, Left, or Center points of the bins. You must also specify what should be returned as shown below.

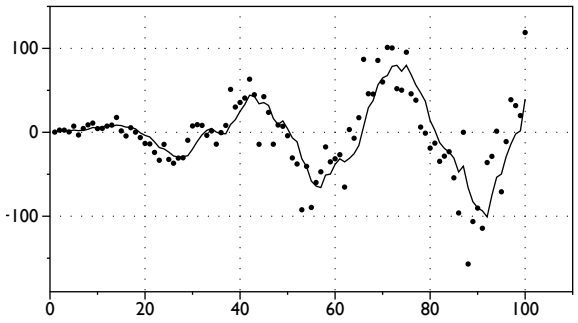


Below is an example using the Bin action on the following x y data to determine counts within each bin and averages for the y data.

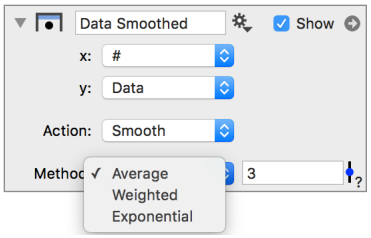
#	x	y	x bin	Count	Average
1	0.5	2	1	1	2
2	1.2	2.1	5	2	2.8
3	2.5	3.5	15	2	8.1
4	6.2	3.9	25	0	NaN
5	12.2	12.3			

Smooth

The **Smooth** action requires two columns (e.g., x,y), and generates a single column of data. Smoothing can be used to reduce noise in a data set as shown below.



There are three options available for data smoothing.

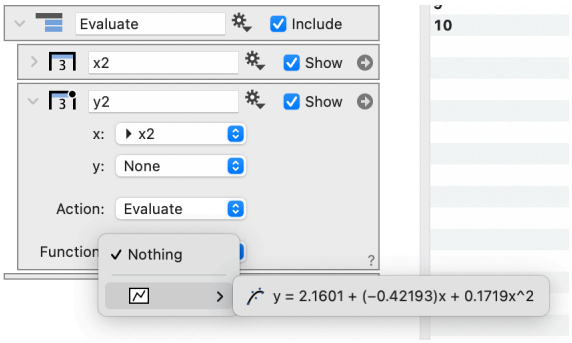


For a moving average, you specify how many previous values to use in the average. This functionality will ignore the x column and just look at the order for the y column. For the first column, it uses an ever larger average, so the first value is the same, the second value is the average of the first and second, the third is (y(1)+y(2)+y(3))/3. The fourth is the average (y(2)+y(3)+y(4))/3, etc.

For more information on these techniques, see: Wikipedia - Simple Moving Average.

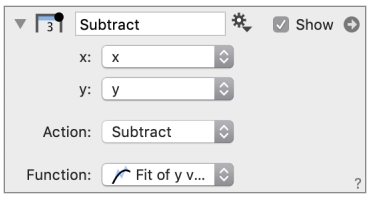
Evaluate

The **Evaluate** option allows you to evaluate a function, y=f(x), at discrete x locations. It required requires one column (x) and either a function from a Fit command or from a Function command.



Subtract

The **Subtract** option allows you to subtract a function, y=f(x), from discrete y values. It requires two columns (x,y) and either a function from a **Fit** command or from a **Function** command.

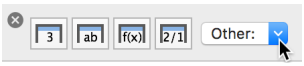


This column can be useful for calculating residuals from a function fit, since the residuals extracted from the **Fit** command directly will only provide residuals for the data used in the fit.

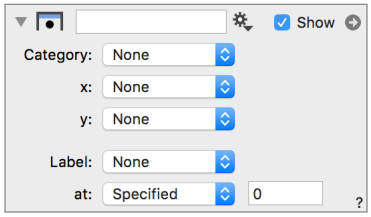
Interpolate by Category

To interpolate values between known data points you can use the **Plot Action** column as described in the previous section. The **Interpolate by Category** column goes a step further as it allows you to interpolate data using a category to group data.

You can add a new **Interpolate by Category** column using the **Other** drop-down menu above the column definitions list.



A new column of this type will look as shown below. The first three drop-down boxes define the data set that you want to interpolate. The last two drop-down boxes define the output.



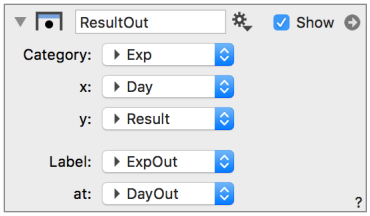
For example, let's say we have raw data from two experiments 'A' and 'B' where the sampling days are not always the same days.

#	Exp ab	Day	Result
1	A	1	8.637
2	A	2	12.26
3	A	5	28.96
4	A	6	32.37
5	A	8	41.45
6	B	1	1.222
7	B	4	4.622
8	B	6	7.753
9	B	7	8.207
10	B	8	10.36

You can use the **Interpolate by Category** column to estimate the 'Result' column for missing days using linear interpolation.

For this example, you can first create the **Interpolate by Category** column, called 'ResultOut'. Next, specify the input data, **Category** equals 'Exp', **x** equals

'Day' and **y** equals 'Result'. We have also defined two additional columns 'ExpOut' and 'DayOut' that specify the category of data to interpolate and the day.



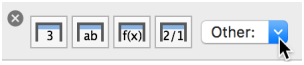
The result of the interpolation is shown in the column 'ResultOut'.

#	ExpOut	DayOut	ResultOut
1	A	1	8.6374
2	A	2	12.258
3	A	3	17.827
4	A	4	23.395
5	A	5	28.964
6	A	6	32.368
7	A	7	36.91
8	A	8	41.452

For days in the original data (1,2,5,6,8), DataGraph does not need to interpolate. For days not in the original data (3,4,7), the values in 'ResultOut' are interpolated.

Map

The **Map** column allows you to lookup items in a list based on a key value. You can add a **Map** column using the **Other** drop-down menu above the column definitions list.

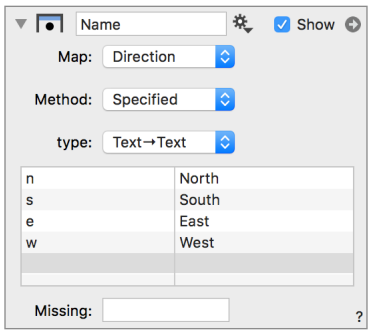


There are three methods available for mapping entries:

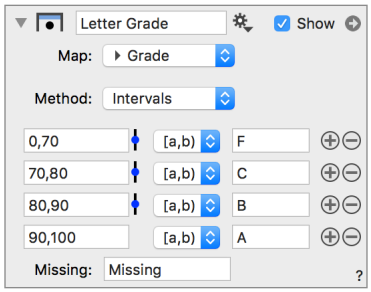
1. **Specified** — map a column by specifying key-value pairs directly in the column definition.
2. **From Columns** — map a column by using other columns to specify the key-value pairs.
3. **Intervals** — map a numerical column based on intervals.

For each of these methods, you must specify the column to map using the **Map** drop-down box. When you add a new **Map** column, the mapping method is set to ‘Specified’ by default. Below is an example using the specified method, where a column named 'Direction' is mapped from text to text.

Direction	ab	Name
n		**North**
e		**East**
s		**South**
s		**South**



You can do the same type of mapping using the 'From Columns' method but you must have two columns that define the key-value pairs. The 'Intervals' method is useful for mapping intervals of numbers to letters. For example, you could map number intervals to grades.



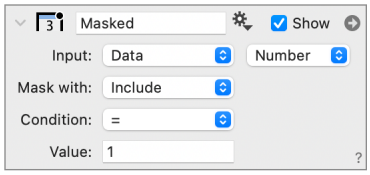
Note that you can also specify what should be shown when an entry is missing. Thus, the value entered in **Missing** is shown when there is no match.

Mask

The **Mask** column allows you to filter a column in the data table. Specify the column you want to filter using the **Input** menu. The mask can depend on values in the **Input** column, or use a different column. For example, we can create a column that only includes values from 'Data' when the column 'Include' is equal to 1.

	Data	Include	Masked
1	1	1	**1**
2	2	0	
3	3	1	**3**
4	4	0	
5	5	1	**5**

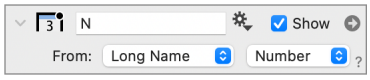
Here is how the options are set in the column 'Masked'.



NOTE: You can also use Masks directly in commands.

Redirect

The **Redirect** column allows you to reference a column using a different name. For example, you may want to shorten a name, or use a column from a different group in an expression. For both uses, the Redirect column can be useful. Here a column named 'Long Name' is in a new column called 'N'.



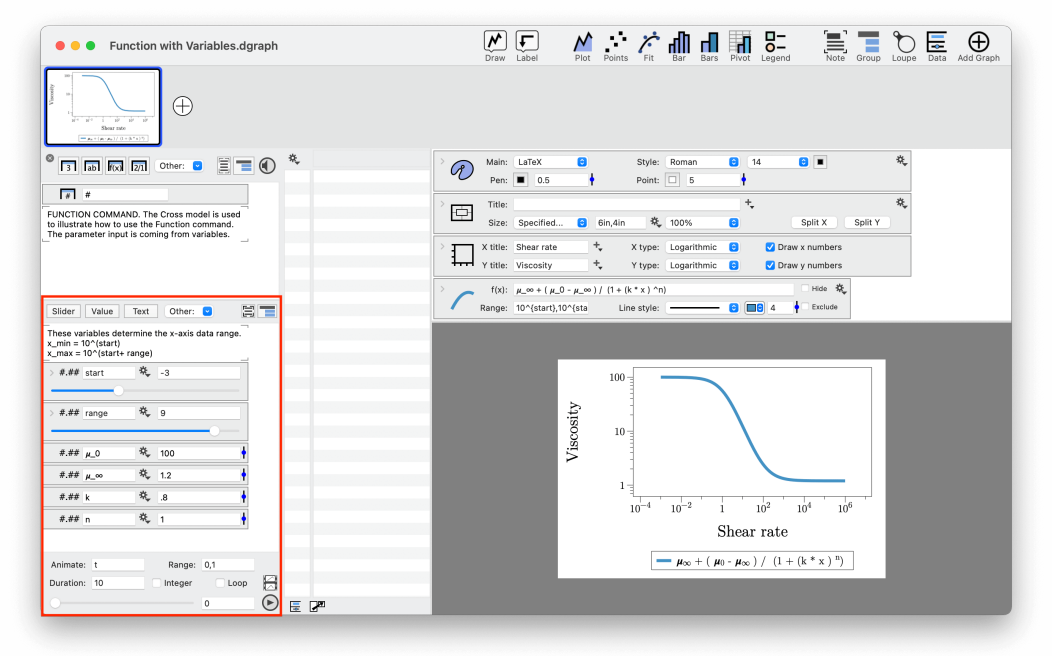
5. Variables

Variables allow you to assign a variable name to a number or text that can be used in other places in DataGraph. For example, numeric variables can be used within formulas or to control the line width in a plot. Text variables can be used in titles or data labels.

You can also use **Variables** to create Font, Color schemes, and Marker schemes, to use across multiple graphs and commands, and they are easily copied between files. There is even a way to have the current date and time set using a **Variable**.

V**ariables** are must have a unique name. Note that by organizing variables into groups you can have variables with the same name, since the group name is used when referring to the variable in an expression, as described in the next section.

The V**ariables** list is located in the bottom half of the data panel. You can create variables using the buttons above the list. The three main types are shown here: **Slider**, **Value**, and **Text**. Additional variable types can be accessed in the **Other** drop down menu.

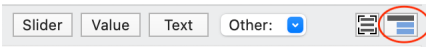


Organizing variables

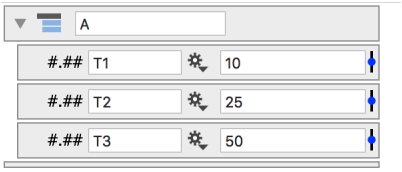
When you add variables to the list, you can organize them by selecting and dragging. You can also group variables and add annotations to help organize and document your data.

Grouping

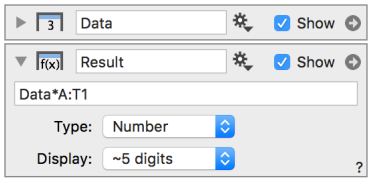
Add a group using the icon to the right of the **Other** drop-down menu. You can add variables to a group by dragging them into the group or by highlighting the variables before clicking the group icon, as shown below.



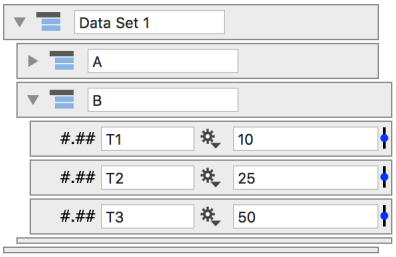
This creates a group containing the selected variables. In the example below, the group is named 'A' and the variables are 'T1', 'T2', and 'T3'.



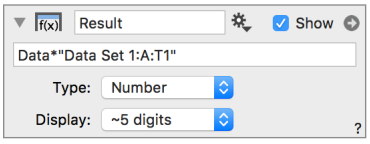
You can refer to variables in expressions by combining the group name with the variable name, using a colon to separate the two. For example, to use the value for 'T1' in an expression you would use 'A:T1', as shown below. This will multiply every entry in the column 'Data' by the value of 'A:T1'.



You can also nest groups within groups. Group 'A' and 'B' can contain the same variables names since the full name also includes the name of the group.

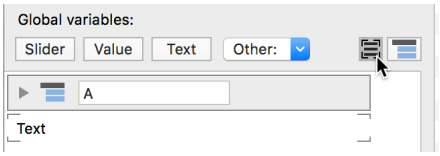


If your variable name contains a space you need to place the entire name in quotes to use it in an expression.

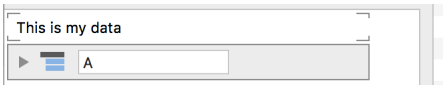


Annotating

You can also add annotations using a comment block. Add the comment block using the icon just to the left of the group icon.

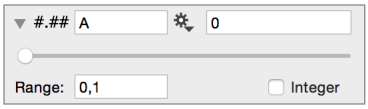


By default, the comment block will simply contain the word 'Text'. You can edit this entry and drag the comment block to a new location.

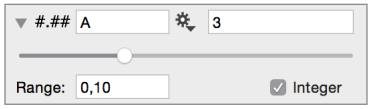


Slider

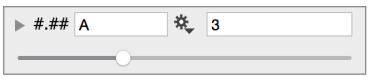
The **Slider** is a numeric variable that has a default value of zero. You can create a **Slider** by clicking the **Slider** button and a variable will be added to the **Variables** list. You must assign a name to the variable. In the example below, we have named the **Slider** variable 'A'.



As you move the slider bar back and forth, the value of 'A' will vary from 0 to 1. Note that the value of the slider is limited to the specified range, but you can modify the range to any values you choose. You can also select the **Integer** check box on the bottom right to limit the numbers that the slider moves between integers only.



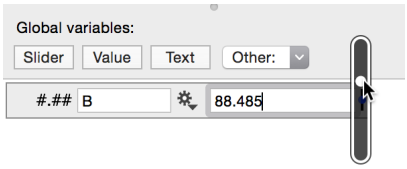
You can also minimize the space the **Slider** requires using the disclosure triangle.



Value

Value is another numeric variable. In the example below, we have a created a **Value** variable, changed the name to 'B' and assigned a value to '52.65'.

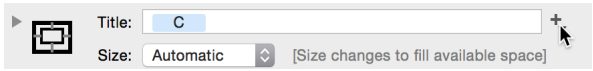
Value also has a toggle bar to the right of the number that allows you to vary the numeric data. This can be useful when you are using the variable in a function and want to understand how the function changes with a given variable. The range that you have available depends on its latest value.



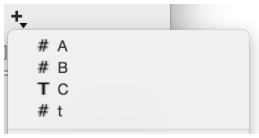
Text

The **Text** variable can be used in titles and labels and can also be used to mask data. Below we have created a *Text* variable called 'C' and changed its value to 'My Title'.

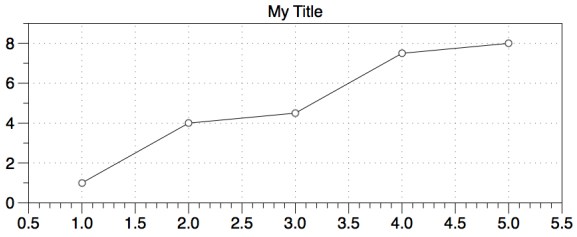
You can place the variable in an expression, as shown below for the title field for a graph. In this case, **Title** is set equal to the 'C' variable.



To select a variable, click the plus symbol to the right of the **Title** field. A drop down list displays the three variables in this file along with a symbol denoting the variable type, where 'A' and 'B' are numeric and 'C' is the text variable we just created.



The graph using this **Text** variable now has 'My Title' as the title.

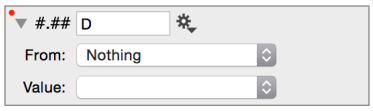


Number from Command

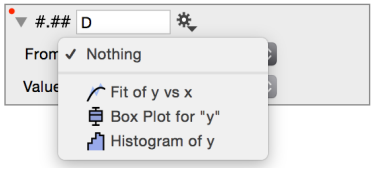
Additional variable types are listed in the **Other** drop-down list. The first one is called **Number from Command**. This variable type allows you to extract a single numeric value from a drawing command. The drawing commands that have extractable parameters include: **Fit**, **Multivariable Fit**, **Box plot**, and **Histogram**.

In the **Number from Command** variable, there are two dropdown boxes. In the **From** dropdown box, you specify a drawing command. In the **Value** dropdown box you specify what to extract.

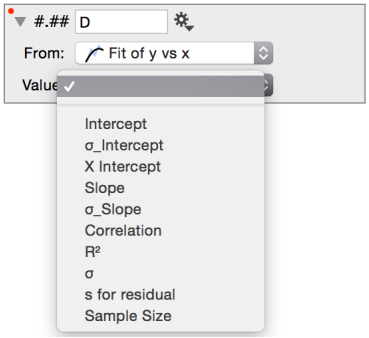
For example, below is a **Number from Command** variable called 'D' in which we have yet to specify a drawing command or the value to extract.



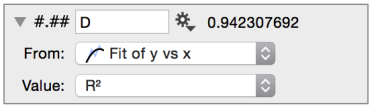
When we select the **From** drop down box, we see a list of all the graphing commands in the specific DataGraph file in which we are working. In the example below, the file has a **Fit**, **Box plot**, and **Histogram**. If a DataGraph file does not contain any drawing commands with extractable parameters, the **From** drop down menu has nothing to select.



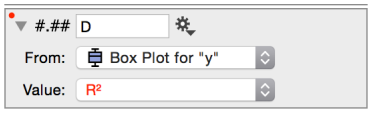
For each type of drawing command, there are different values that can be extracted. For example, if we select 'Fit of y vs x' the **Value** dropdown box contains a list of variables corresponding to that drawing command.



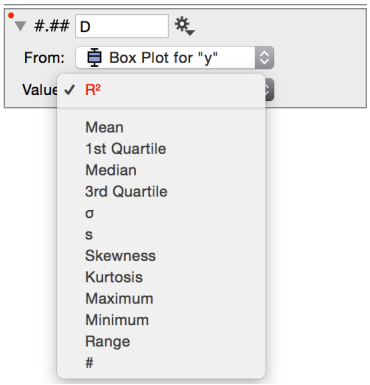
Once you select a **Value**, the red dot in the left corner of the variable goes away and the value of the parameter is shown.



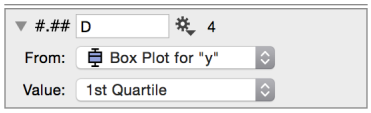
If you changed the drawing command specified in the drop down box to something that does not have the same list of extractable values a red dot appears indicating the variable is no longer valid. Here we have changed the selected drawing command to a 'Box Plot for y'. Since R^2 is not a parameter generated from box plots, the name changes to red, indicating that this is no longer a valid choice.



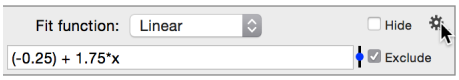
When we now select the **Value** dropdown box, we have a different list of possible values to extract that correspond to the **Box Plot** command.



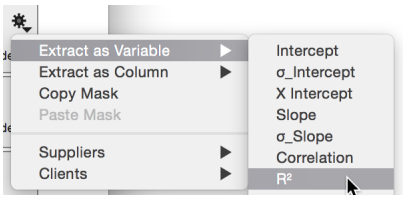
Once you select a valid **Value** for the specific drawing command, 'D' is now set correctly and the red dot disappears again.



You can also use the gear menu on the right hand side of any drawing command to access the list of variables you can extract. In the example below, a variable is extracted from a **Fit** drawing command.



From that gear menu, you will see the **Extract as Variable** shortcut.

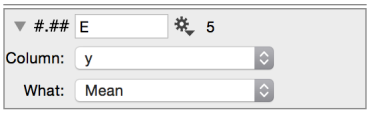


When you select a variable in this manner, the variable is automatically created.

You can also extract entire columns of data from a drawing command such as the residuals in a regression. This is discussed in more detail in the **Data Table** chapter. See the section on the **From Command** column type.

Number from Column

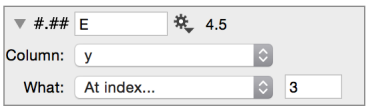
The **Number from Column** variable allows you to extract various summary statistics about a column of data. When you create a **Number from Column** variable, you have to specify the column of data you want to use and the type of value you want to display. In the following example, the Mean value is being calculated based on the 'y' column and the variable is called 'E'.



The values you can extract include sum, maximum, mean, range, count, standard deviation, which are described in the table below.

Name	Description
sum	Sum of all values
sum of ^2	Sum of the squares of the values
max	Maximum of a column
min	Minimum of the column
range	Maximum minus the minimum
count	Number of the non-empty rows
mean	Mean of the column
median	Median of the column
σ	Standard deviation (sigma)
s	Unbiased estimate of the standard deviation
sem	Standard error of the mean

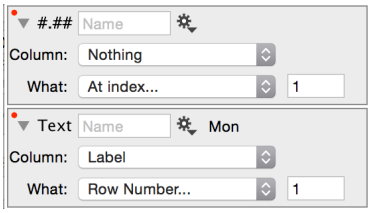
You can also extract a single numeric value from a row by specifying the index for the row.



This feature can be useful for animations since you can set the index to a slider variable like 't' and restrict that to an integer. Then the slider could be used to scrub through the rows.

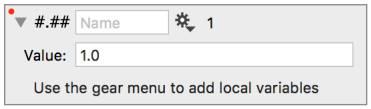
Text from Column

Text from Column works in a similar manner as **Number from Column** discussed in the previous section. You can use this to select a particular row as well as return the name of the column.



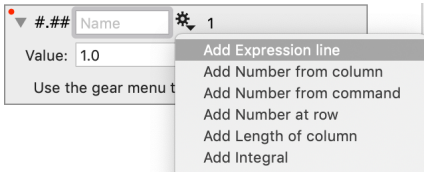
Expression

The **Expression** variable allows you to create a numeric variable that uses a formula or expression. You add an **Expression** column from the **Other** drop-down menu.

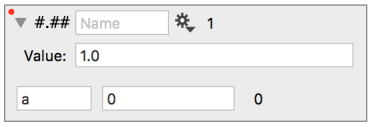


The numeric value of the expression is shown to the right of the name, which is set to '1' by default.

As noted at the bottom of the variable, you can use the gear menu to add local variables, such as extracting numbers from columns or commands (See the previous sections on **Number from column** and **Number from command** to learn more about these types).

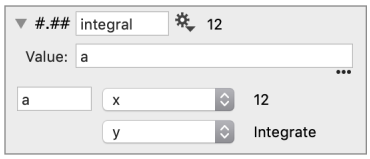


When you add an entry using the gear menu, a section is appended to the bottom of the variable, where you can add a local variable, in the form of a number or expression.



This allows you to handle more complicated expressions by breaking them down into multiple steps, or intermediate calculations.

The **Expression** variable can also be used to integrate a discrete set of data, given two columns as input, using the Trapezoidal rule (https://en.wikipedia.org/wiki/Trapezoidal_rule).

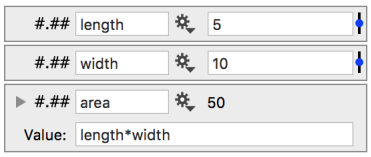


Once created, you can use this **Expression** variable in the same way as any other numeric variables. Expressions themselves are formatted in the same way as the **Expression** column, but the **Expression** variable is used to calculate a single number, as opposed to a column of numbers.

Example 1: Using Global & Local Variables

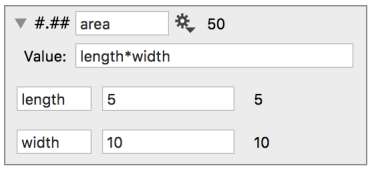
An **Expression** variable can include other variables or local variables that you create using the gear menu.

For example, say that you have two variables called 'length' and 'width'. You want to use these existing variables to calculate 'area'. To do this enter 'length*width' in the **Value** field, as shown below.



The result of the calculation is shown to the right of the variable name (i.e., 50).

Another way to compute the same value would be to add two expression lines using the gear menu to create two local variables, one for length and one of width.



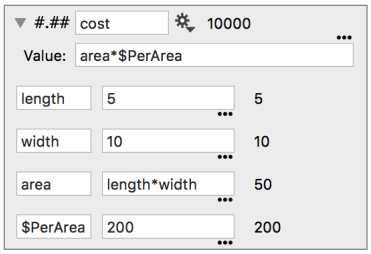
Each of these are local variables and would not be recognized outside of the variable called 'area'.

Example 2: Using Multiple Steps

You can add multiple steps to your expressions by using the gear menu to add sections to the **Expression** variable. This can be very useful when you have a complicated expression that you want to break up into smaller equations.

To illustrate, let's say that the area is needed to calculate a cost, using a conversion factor '$PerArea'. You can add additional expression lines and calculate the area within the **Expression** variable.

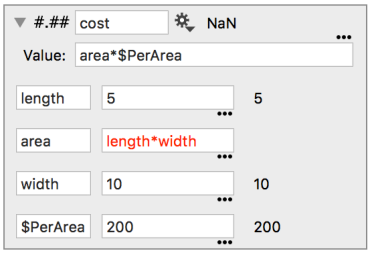
In the **Value** field, the calculated value for 'area' is multiplied by '$PerArea'.



You can select an entry by clicking with the mouse. Hit the delete key to remove. You can also change the order of the entries by clicking and dragging.

Note that the calculations are computed using the order of the entries. If you move a local variable below an equation that is using that value, the color of the equation will be red, to indicate an error.

For example, if we moved 'width' below the calculation for 'area', the 'area' field changes color to red to indicate that this calculation is missing a variable.

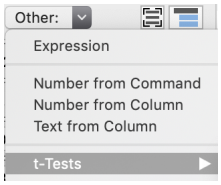


t-Test

The t-Test variable allows you to calculate the p-value from three standard types of tests:

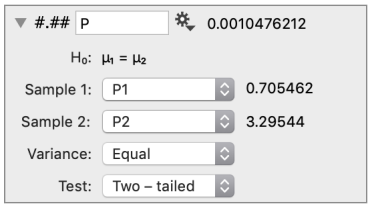
1. One-sample
2. Two-sample
3. Paired

Create a t-Test using the Other menu. Select t-Tests to get a menu of the available tests.



Once you create a test, an entry will be created in the variables list to specify the data you want to compare.

The hypothesis being tested is provided at the top of the variable. Further options are shown below.



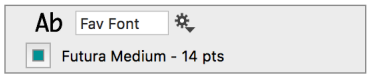
The p-value can then be used as a label. See the Bracket command.

There are also three functions related to the t-Test that can be used to construct your own test as needed.

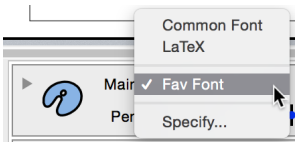
tcdf	tcdf(x,df) the cumulative distribution function from the Student's t distribution, at specified degrees of freedom (df)
tpdf	tpdf(x,df) the probability density function (pdf) from the Student's t distribution, at specified degrees of freedom (df)
tinv	tinv(p,df) inverse of the cumulative distribution function from the Student's t distribution, at specified probability (p) and degrees of freedom (df)

Font

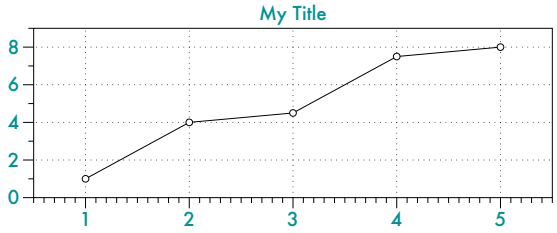
The Font variable allows you to create a font that is added to every font selector menu. In the example below, we have created a variable called 'Fav Font'.



Now, when we are selecting the Main font in the Style settings of a graph, 'Fav Font' is listed as an option.



After selecting 'Fav Font', the title and axis labels are updated accordingly.



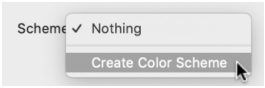
For any new graphs that are created, the **Main** font is automatically set to the font selected from the last graph created; thus, additional graphs would use the 'Fav Font' as the **Main** font until we select another font for a particular graph.

Note that you can also customize every font in a graph individually using the style settings. See the Chapter entitled *Layout* for more information.

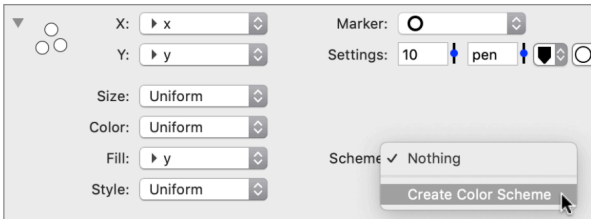
Color Scheme

Color Schemes allow you to have a consistent color pallet across graphs and can even be copied and pasted between files.

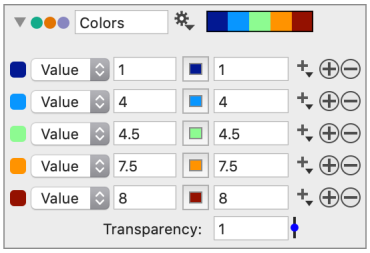
To quickly create a color scheme, select Create Color Scheme from within a command that has the color scheme option.



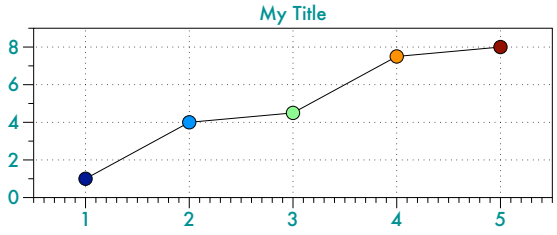
For example, we can create a color scheme for the Fill in a points command.



This will create the scheme based on the selected data column.



The graph will automatically reflect the new color scheme.

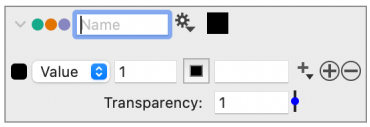


The **Color scheme** can be used in the **Points**, **Pivot**, **Pie**, **Box**, **Lines**, **Connect**, **Range** and the **Bar** command.

Using the **Color Scheme** variable, you can set the color used in a drawing command based on numerical values or on text labels.

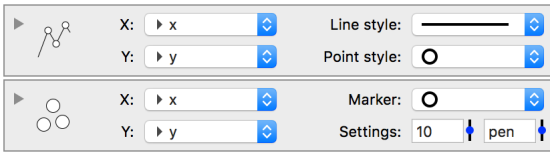
Manually Create a Color Scheme

You can create a global **Color Scheme** variable manually under the **Other** dropdown box. Select **Color Scheme** from the **Other** dropdown box to create a new **Color Scheme** variable..

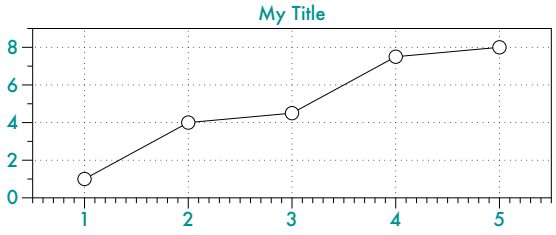


Color Scheme variables have a list of conditions and corresponding colors. When you first create a color scheme, it only has one condition set. Specifically, for data points with a numerical value of 1. You can give it a name, change the color, and add a title.

To use the example from the previous section, we first have to add a **Points** command to the drawing commands; the color scheme does not work with the **Plot** command. Our graph now has both a **Plot** and a **Points** command.

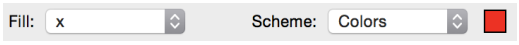


We also increased the default point size to 10 in the Settings for the **Points** command. Since the points command is listed last, these points cover the points from the line command.

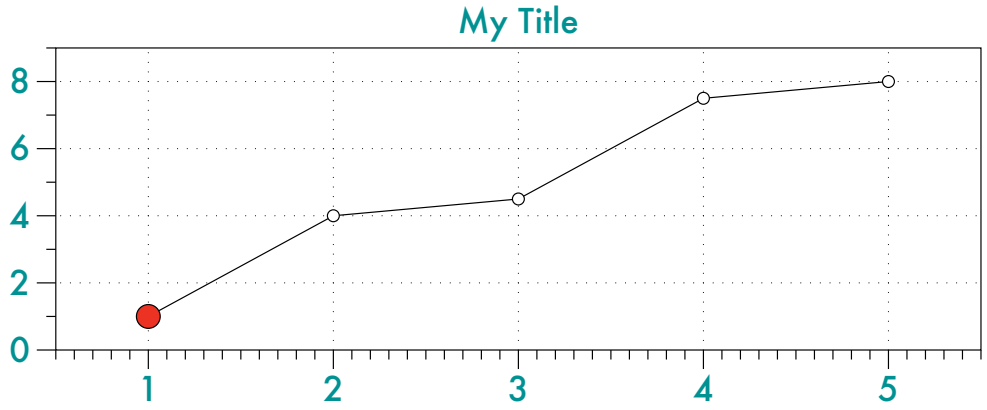


We can now change the fill for the **Points** command using our new color scheme, 'Colors'. First you will need to open the disclosure triangle for the **Points** drawing command. In then **Points** command, you have several options for defining the **Size**, **Color**, **Fill**, and **Style** of the points.

For this example, we will change the Fill of our points to be based on the value for 'x' and to use the color scheme 'Colors'.

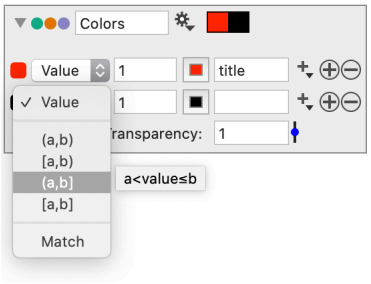


Notice how the only point that is now drawn from the **Points** command is at x=1 because this is the only **Value** specified in our color scheme.

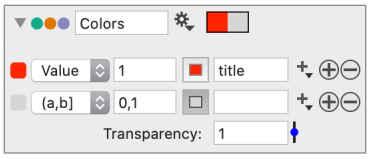


Next, we can add more colors to our color scheme by clicking the right plus symbol to the right of the first entry.

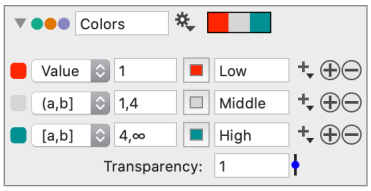
The colors can be set not only based on individual values but also based on ranges of values. By selecting the drop down menu for an entry, a list of options is given. As shown below, hovering over each option, describes the type of range.



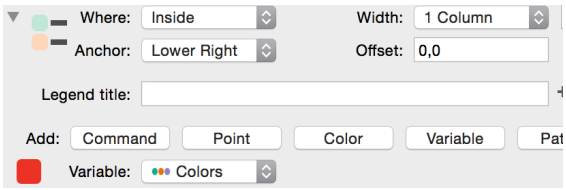
The color is set by selecting the box to the right of the specified value or range.



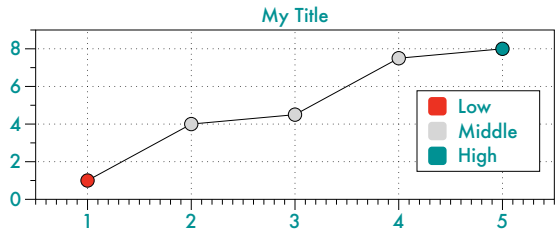
In the example below, we have three possible colors assigned to data in our graph. The range can also be set with an upper value of infinity by selecting Option-5. We have also added a better name for each color as 'Low', 'Med', and 'High'.



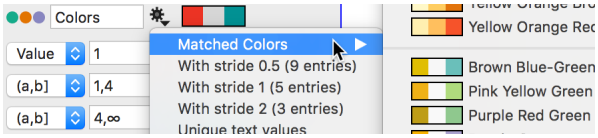
We can also create a **Custom Legend** and add the variable 'Colors'.



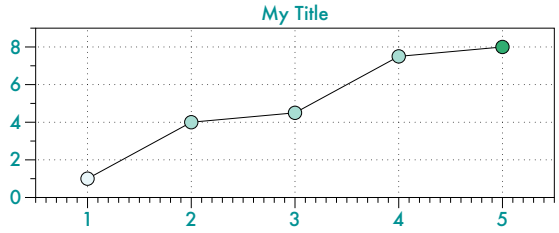
Below, we now see the graph based on the updated **Color scheme** and the **Custom legend**. Note that the color is assigned to each point based on the first line item that is a match in the color scheme.



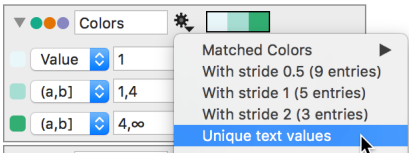
In the color scheme, the gear menu allows you to select between a list of color presets. In this example, we have three colors so the **Matched Colors** also has three colors.



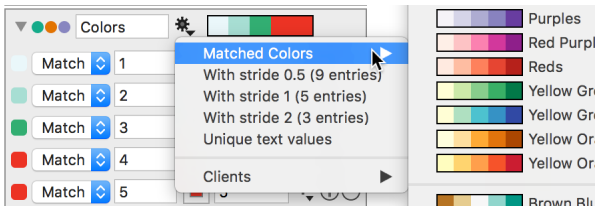
For this example, we have selected the blue-green color scheme and our graph updates accordingly.



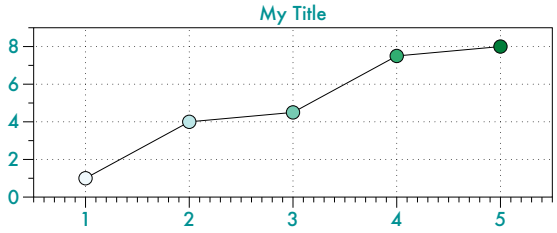
The gear menu can also be used to suggest strides or unique text values based on the current content in the graph where the scheme is being used, as shown below. Note that if the color scheme is not being used by any drawing command you will not see these selections in the gear menu.



After selecting 'Unique text values', the color scheme is automatically modified to correspond to the five values in our 'x' variable. The two new colors in our scheme are set to the default of red. If we want to reset the color scheme, we can select Matched Colors again.

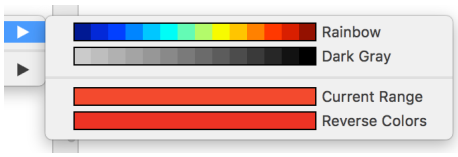


In this example, we again select the ‘Blue green’ color scheme. Now each point is a unique color.



Large Color Schemes

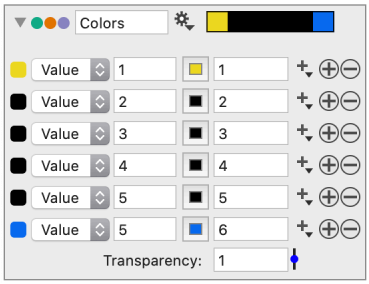
The number of color suggestions will vary depending on number of items in your color scheme, such that you have fewer options when your list is longer. When you exceed 12 entries only the 'Rainbow' and 'Dark Gray' scheme will be shown as options.



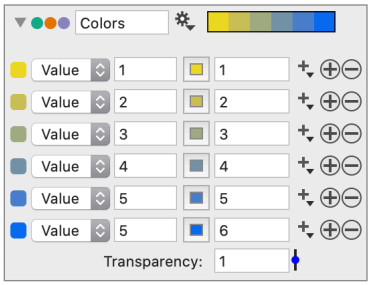
These options will create a discrete number of colors based on the number of items in your list with no limit.

Custom Color Schemes

When you have six or more entries in your list you have the option of creating a custom color scheme based on two colors, using the 'Current Range' option. Start by creating a color scheme with the number of entries you require. You need to have at least six entries for this option to be available.

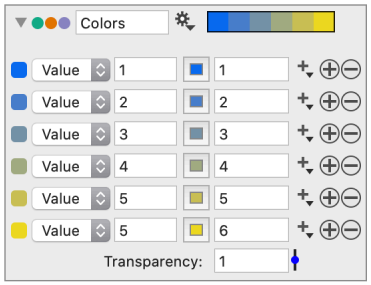


Next, select the 'Current Range' option from under the gear menu. You will see a color scheme that is uses the first and last colors you specified. These colors will be populated into your color scheme variable.



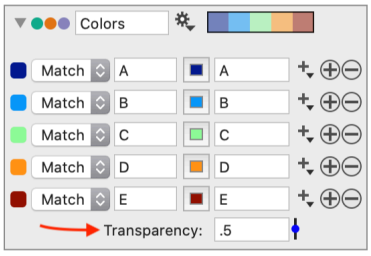
Reverse Colors

The 'Reverse Colors' option simply uses the current color scheme and reverses the order of the colors. The values stay the same but the order of the colors changes.



Transparency

At the bottom of every color scheme is an option for changing the transparency for all of the colors. You can type in a value, use the slider to adjust from 0 to 1, or set to a variable.

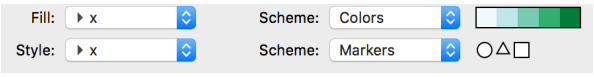


The individual tiles can still have transparency added. The value at the bottom of the scheme will apply in addition.

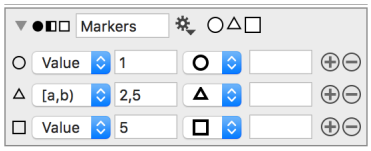
Marker Scheme

The **Marker Scheme** command allows you to map markers to values in your data. The way it works is very similar to the color scheme.

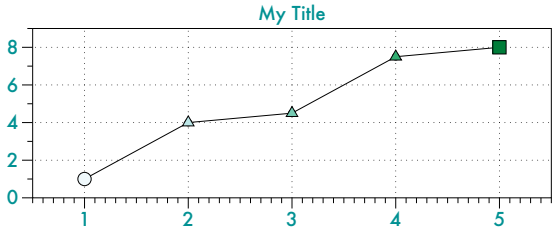
You can specify an existing **Marker Scheme** or create a new one in the Style section of the drawing command, which is directly under the Fill section.



You can also create a **Marker Scheme** directly in the variable section..

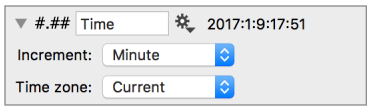


Here is what this graph looks like with these marker types.

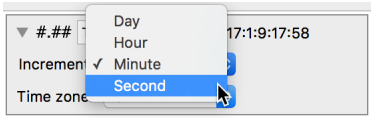


Current Time

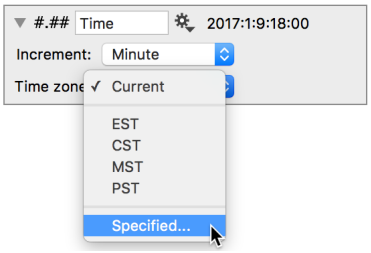
The **Current Time** variable tracks time in a given time zone. An example is shown below, which has been named 'Time'.



By default, a **Current Time** variable shows the year:month:day:hour:min. You can change the **Increment** to track either days, hours, mins or seconds, as the smallest increment.

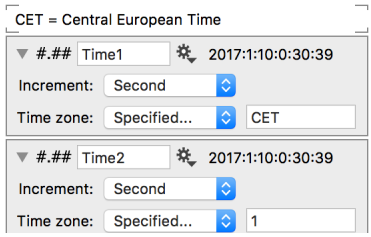


The **Time zone** will be based on the time zone in which you reside, known as 'Current'. You can change this to any of the built-in options or specify exactly the Time zone you would like.



When you select **Specified**, an entry box appears to the right. In this box, you enter the offset from UTM. This can be either in the form of a time zone abbreviation or the actual numerical offset from UTM.

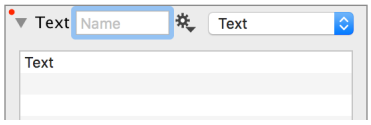
In the following example, two time variables are created, 'Time1' and 'Time2'. Both of these times use the 'Specified' option for central European time, but 'Time1' is specified using the abbreviation 'CET' while 'Time2' is specified using the UTM offset of +1.



Not all abbreviations are coded into DataGraph. If you want to specify a time zone that does not have the abbreviation built-in specify it using the numerical UTM offset. For a list of time zone abbreviations and offsets see: https://en.wikipedia.org/wiki/List_of_time_zone_abbreviations.

Text Menu

The **Text Menu** variable can be used as any other text variable. It consists of a drop-down box on the top right that allows you to change the value based on the entries in the list below. By default, there is only one entry, the word 'Text'.

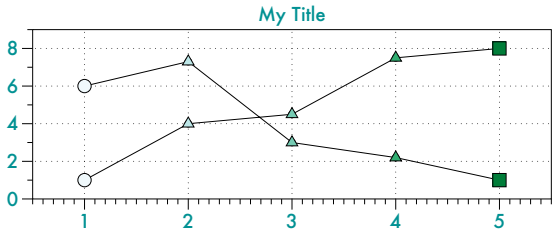


Using the **Text Menu**, you can create a variable that allows you to easily toggle between datasets in a graph. You can manually set up the **Text Menu** but it is much easier to use shortcuts that have been created for this purpose.

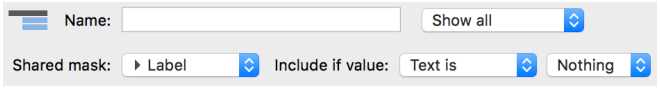
To illustrate, consider the following data where we have two x and y sets of points that are labeled either ‘A’ or ‘B.’

#	Label	ab	x	y
1	A		1	1
2	A		2	4
3	A		3	4.5
4	A		4	7.5
5	A		5	8
6	B		1	6
7	B		2	7.3
8	B		3	3
9	B		4	2.2
10	B		5	1

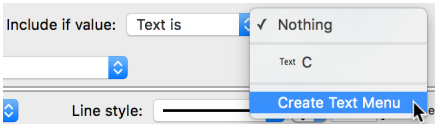
We modified our example plot from the last section to include this data.



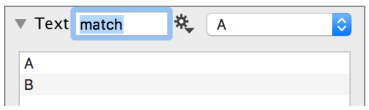
Let’s say that we only want to display one set of data at a time. First, we can create a group mask for the drawing commands. Set the **Shared mask** to the 'Label' column and **Include if value** to 'Text is'.



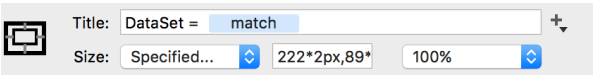
Next, select the drop-down menu 'Nothing' and select 'Create Text Menu'.



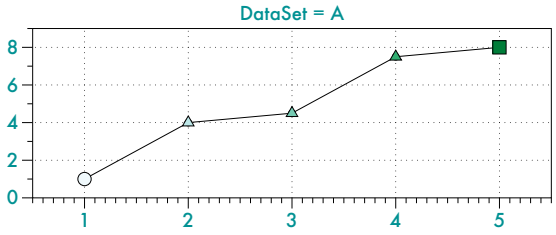
Once selected, DataGraph will create a **Text Menu** variable named ‘match’ that contains the two unique values from the 'Label' column.



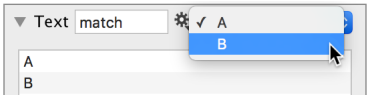
The **Text Menu** variable can then also be used as a token in the title of the graph. As shown below, we replaced the **Text** variable that we created earlier in this chapter with the **Text Menu** variable.

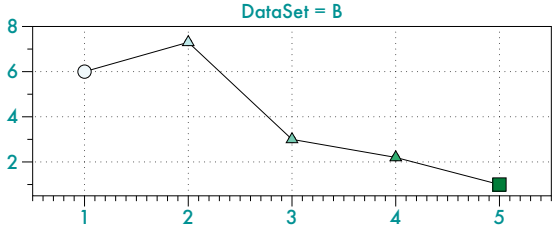


Now, the **Text Menu** variable is displayed in the title and the graph only contains data where ‘label’ equals ‘A’.



If we toggle the **Text Menu** variable to equal 'B' the data and the menu are updated in our graph.

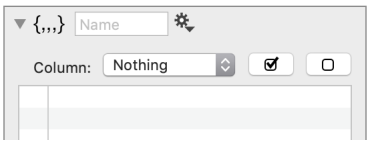




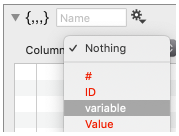
Text Set from Column

The **Text Set** variable allows you to create a menu of choices for using in a Mask. Unlike the other variables, the **Text Set** allows for multiple selections. The list is also dynamic.

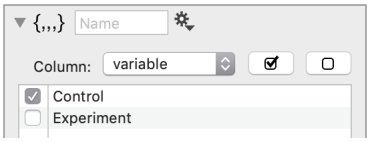
1- Create from the **Other** menu in the variables.



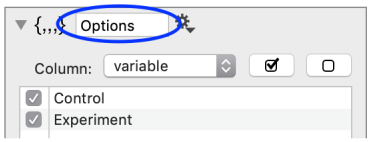
2 - Select a **Column** using the menu. Only text columns will work, all other column types will show up in red.



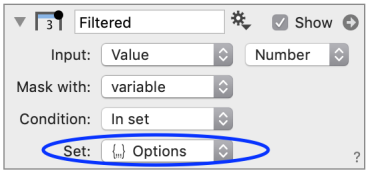
Note: The set of entries is automatically populated by including every unique entry. In this example, we have a text column that contains just two entries "Control" and "Experiment".



3 - Give the set a name.



Now you can use this variable in a **Mask**. Thus, you can change the values for a **Mask** by changing the selected items in the **Text Set**. The **Text Set** can also be used in the **Masked** column.



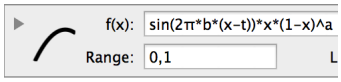
The Animation Variable

The bottom of the data list are settings for creating animations. By default the name of the variable is 't', but you can change this in the Animate entry box. You can also specify the Range and Duration.

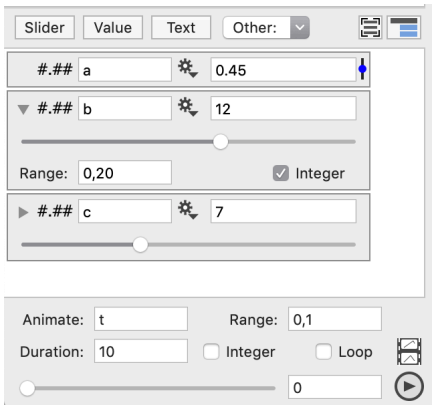
The animation variable can be used in any location a number or other variable can be used. To begin the animation, press the Play button. To export as a movie (mp4) click the Movie icon above the Play button.

Example: Use Variables to Animate a Function

Variables can be used in a variety of ways to create animations. For example, you can create a **Function** command and specify the function to be sin(2π*b*(x-t))*x*(1-x)^a,where 'a' and 'b' are defined as Variables.

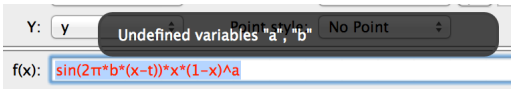


In the above example, note that one of the variables is specified as an animation parameter (default name is 't'). This is the animation parameter set in the bottom left.



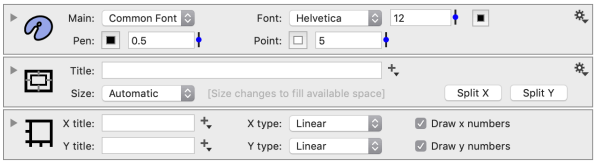
You can have DataGraph animate this on the screen or save it into a movie. When you click the play button to the right of the slider, DataGraph runs the animation on the screen. If you click the Movie button above the play button, you create an mp4 file.

Note that if the variables 'a' or 'b' are not defined, DataGraph will not understand the expression and display it in red.



6. Layout Settings

There are three groups of settings that are used to define the overall layout of a graph: the **style**, the **canvas** and the **axis** settings. These are always located above the drawing commands and are shown in the following image



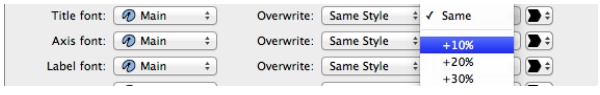
The **style** settings are used to set overall rules, line thickness, font, color scheme and look of the axes. The **canvas** settings are used to position the drawing area. The **axis** settings determine how the data is mapped to the drawing area defined in the canvas settings. The axis setting also allows you to set tick marks and axis labels. This chapter focuses on these three settings. The content of a graph is created by using the drawing commands, which are described in the next chapter.

Style

The style setting is the top entry, and the icon looks like an artist palette.

At the top you select the Main font style, the default pen color, line thickness, fill for markers and marker size. By default, drawing commands will refer to these settings but they can be overwritten.

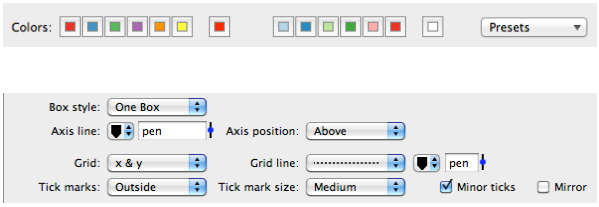
Once you open the detailed portion by clicking on the disclosure triangle, there are a number of additional settings you can change. These are grouped together by function, with small spaces to separate them visually. The first group defines the built in font styles. Some are used by the axis settings, such as title and axis.



The drawing commands that draw text also refer to these fonts by default. How to use them and overwrite style and color is explained in the next chapter. By

default, they all refer back to the Main font defined in the top line, so if you change that font everything in your graph gets updated in a consistent manner.

Below the fonts, you set the color theme. You can use the built in colors or you can overwrite the colors freely. The colors that are selected in the style settings are then accessible to the drawing commands. The drawing commands can pick those colors from a menu rather than having to pick every color separately. The intent is to make it easy to limit the number of colors you use and change the color of several objects at once. By defining them here you can quickly change the color scheme of the entire graph.

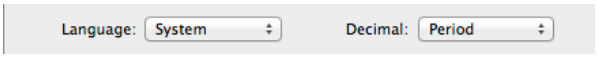


Below the color list you specify the drawing style for the axis as shown below. Note that by default the axis line is set to be ‘pen’. The pen size is set in the top left corner of the style settings. This works since the style defines a variable ‘pen’ that is accessible from all command entries in the graph. When you change the pen thickness you change the pen variable at the same time, and this will therefore change the axis line.

Since pen is a variable, you can also use it to create expressions. For example, you can set the axis line to be pen*2 to make it twice as wide as the pen. You can also set it to '1.0' when you don’t want it to depend on the pen thickness and always be one pixel. This approach is used in the drawing commands to make the default line thickness be the same as the pen thickness.

In the style settings, you set the grid style and the drawing style for the tick marks. The position of tick marks is controlled by the axis setting or by using the Extra Axis drawing command.

Below the axis style is the language and localization settings. The date tick marks use names such as Jan, Feb, and date formatters allow you to use week day names as well. The default option is to use the settings in the system, but if you want to share a file with someone in a different country or use a different language for other reasons, you can select that here. Here you can also specify whether or not numbers should be formatted with a decimal or a period.

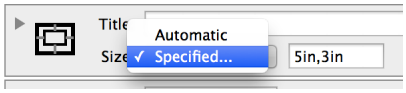


You can save the settings by using the style set and quickly switch between styles. For example, you can set up a grayscale vs color theme or thick presentation theme vs thin scheme for printing.

You can copy a style and paste it into a style setting in a different graph. You do that by selecting the style setting entry (thick blue border) and select Copy. Then go to the graph you want to change and select Paste. You don't have to select anything.

Canvas

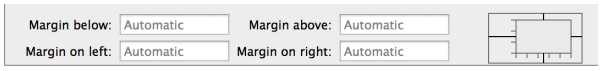
The canvas sets the drawing area for the graph and the title of the graph.



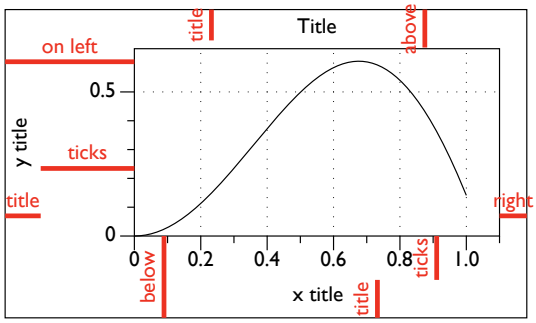
In the canvas, you can specify the figure size by using the 'Size' menu. Once you select the size you can set the working magnification. This does not affect the output but makes it easier to work on small graphs or on machines that have very high resolution.

The *Actual* setting in the magnification menu makes the graph appear at the same size as when you print it. Otherwise a graph will look a lot smaller on the screen than when you include it in a text document. It is better to set the size of the graph to the final size in DataGraph rather than scale the figure in another program (e.g., word processor).

Another common setting you need in the canvas detail is the margin settings. If you leave them blank, DataGraph uses automatic rules to determine the space around the data area. The automatic margin size depends on the graph title, font, axis titles, and any drawing commands set to draw outside the data area, etc.



The following figure shows how the margins are computed.

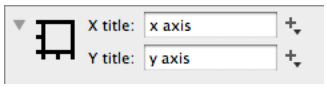


If you leave the margins as automatic in the canvas, the margin is the sum of the tick mark space and title space. Otherwise the margin space is exactly what you specified, but this will not affect the tick or title space. Note that the margins are set in pixels, but DataGraph has defined conversion constants so that you can type in 1.1in and it gets converted to pixels.

Note that the tick spacing is adjusted in the axis settings. The title space depends on what is drawn there and is not set directly. If you want to tweak it, you do that by setting the corresponding margin.

Axis

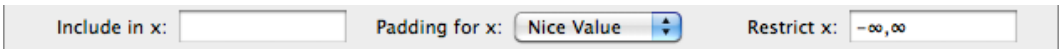
Once the size of the drawing area has been determined by the canvas settings, the axis settings control how the data is mapped onto the graph. This setting is partly also a drawing command, in that it will draw the x and y titles when specified and will draw tick marks when the Draw x/y numbers is checked. This can also be accomplished using the Extra Axis drawing command.



The axis range in x and y is computed in three steps as described below.

Step 1: Compute the union of all the drawing commands. The drawing commands will affect the x and y ranges based on the data that they are representing.

Step 2: Add to that the numbers you specify in the 'Include in x/y' field inside the axis settings (see image below). By default this is empty, which means that the range from step 1 is unchanged.

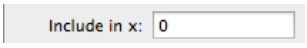


Step 3: Pad this range so that the drawing commands will not go right up against the boundary. The default is 'Nice Values' which uses a number of heuristics to pick the padding. You can turn this padding off and pick other methods using a menu.

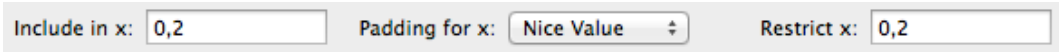
Step 4: Restrict this range, i.e., crop, by using the range specified. By default this is -∞,∞ which means that the range from step 3 is unchanged.

For example, consider the case where you have a plot with the x range set to [0.2,1.5]. This will be the result from step 1. If you leave the **Include** field blank, step 2 is not changed. If the padding in step 3 is set to 'Nice Values', DataGraph will change the range to [0.1,1.6] to add space to the left and right. Step 4 is by default set to -infinity to infinity and will therefore not restrict it at all. Note that this padding is independent of the graph size.

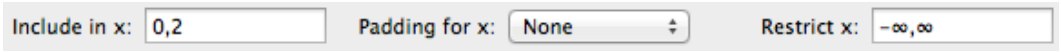
In this case, you might want to include the origin. Do that by typing the number 0 into the **Include in x** field. This means that step 2 will make the range [0,1.5] and then the padding will make it [0,1.7] to keep things symmetric.



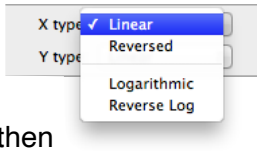
You can also specify the range exactly. Assume for example that you want the range to be exactly [0,2] independent of what data you have. Then you set the **Include** and **Restrict** fields to 0,2. Here the padding option doesn't matter.



This means that any data outside that range is not visible. If you are zooming in on data, this is what you want. If you want to see whether or not there is anything outside, but still ensure the range includes at least [0,2], then you leave the **Restrict x** as is and turn off the padding.



Axis types



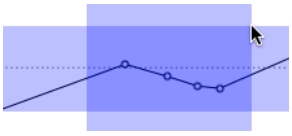
The axis type determines how a coordinate is mapped. The most common is linear, which means that each unit on the axis has the same length in the output. Logarithmic means that first DataGraph computes the logarithm of the data and then draws the result linearly. Logarithmic is better when you draw stock

prices and other data where relative growth/reduction is more important than absolute size.

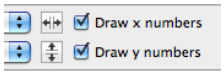
Crop using the mouse

When determining the region for an axis, and this also holds for the split axis described below, the last step above (4) is to crop the range using the **Restrict** field. It is certainly possible to enter in your own numerical value here, but if you just want to zoom in, this is not a very fast method. DataGraph has a number of methods to set and modify the cropping range using the mouse and scrolling. The most common method is to click in the background of the axis and drag the mouse.

Once you have dragged it a few pixels, you see a blue selection interval.

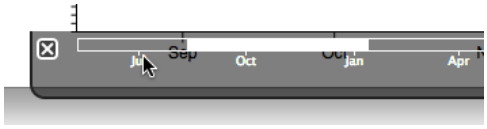


As shown above, if you drag the mouse back towards the starting point it will vanish. When you release the mouse, these intervals will overwrite the current restriction range for the x and y axes. By moving the mouse horizontally, you only crop the x-axis, and only vertically, and you crop the y-axis. If you do this accidentally, the simplest thing is to move the mouse towards the starting point to not crop anything or hit undo after you release the mouse.

If you crop the axes, small icons are displayed to the left of the 

Draw x/y check boxes. They indicate the range is restricted. By clicking on either of these buttons you reset the range to -∞,∞.

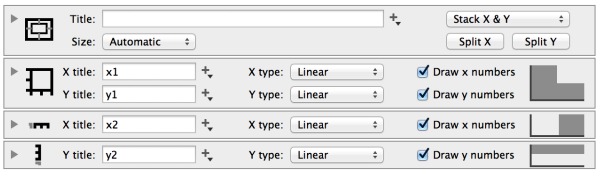
If you then move your mouse over the axis, a floating window pops up on top of the axis, as shown in the following image. This window allows you to adjust the range.



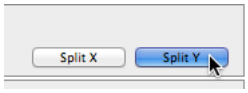
You can remove the crop, click on either end of the white bar to adjust one of the ends or click and drag the entire bar to translate the range. You can also use the scroll functionality to translate the range. Note that if you scroll up/down for the x-axis and left/right for the y-axis, you zoom in or out.

Split Axis

By default, the entire drawing region is used by the axis specified in the axis setting. To have more than one graph in the drawing region, you can split the x or y-axis. You can split either axis using the Split X or Split Y buttons to the right side of the canvas setting.



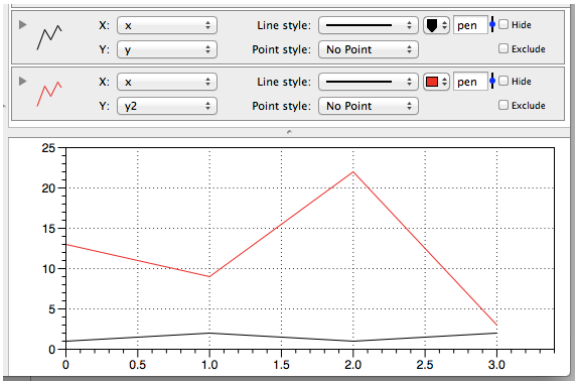
When you click the Split X or Split Y button you get additional settings just below the axis settings.



A small axis selector is also added to the right hand side of each drawing command that represents the drawing region and highlights the location for the drawing command. You can easily move a drawing command by simply clicking a different axis.

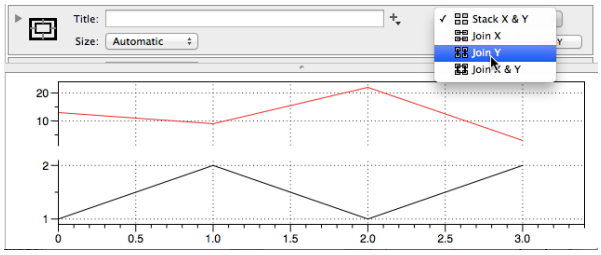
Double Y Graph

The simplest and a common use for this is to create what is typically called Double Y graph. For example, consider the following line plots that have very different y ranges.



To overlay them, you first need to split the y-axis. In the plot command for the red line, click on the small axis selector that was added to the right side. Now the graphs show up one on top of the other, as shown below. The final step is to

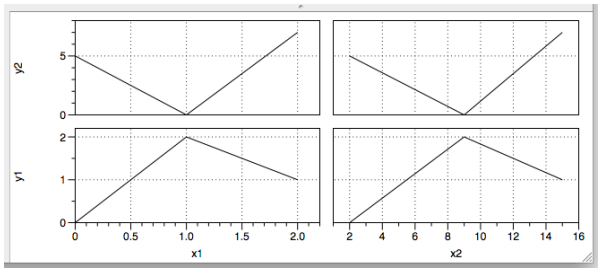
overlap the two y axes. This is done by selecting 'Join Y' on the right hand side of the canvas (see above image).



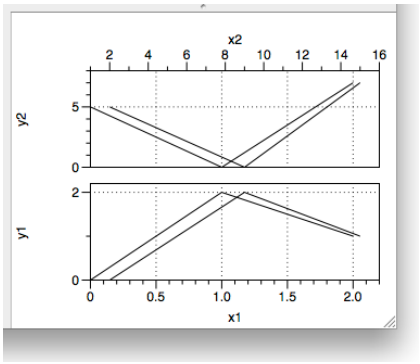
Split in X and Y

If you split the x-axis once and the y-axis once you get a 2x2 matrix of graphs.

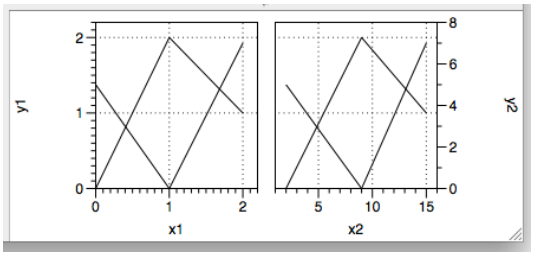
Each row shares the same y-axis and each column shares the same x-axis. By default, the ranges are computed independently for each axis. Thus, x1 and x2 do not have the same range, and neither do y1 and y2.



You can overlap the two x axes by picking Join X.

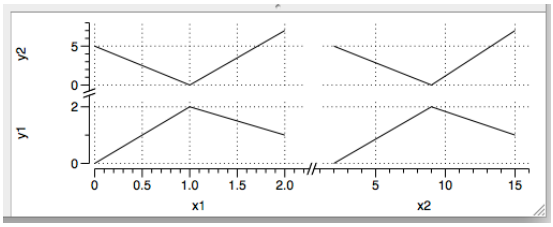


Joining the Y-axis instead gives you



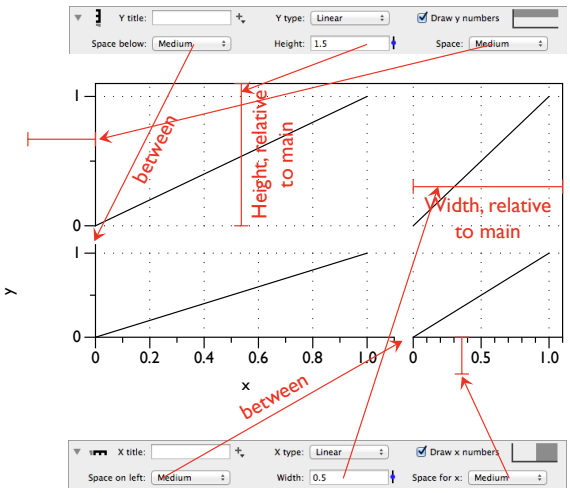
If you want fully independent graphs, you should create them as separate graphs (explained in a later chapter) and combine them using the Graphic command.

You can adjust how the split axis is joined. In the graphs above, the Box style in the style setting is set to Axis Box instead of the standard 'One Box'. You can change that style, and inside the Split axis entries you can adjust how the break in between the sub-axes is drawn. For Offset box style and break separator you get the result shown below.



Split axis spacing

Inside the split axis you can adjust the spacing and layout. The following figure shows what each field controls.

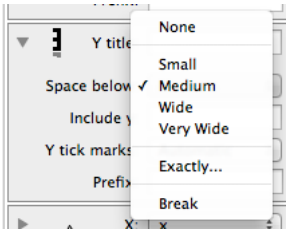


The main axis controls the space for x and y tick marks. The overall size is controlled by the canvas setting. The width/height fields are relative to the main axis. The space between axes can be adjusted in several ways. If you set it to Exactly... you can type in the exact space in pixels/mm/inches, etc.

Axis Labels

If you specify the x or y title, this command will draw them centered with the y title rotated 90 degrees counter clockwise. This is the most common way to draw axis labels, but if you want a more flexible method you can leave the labels here blank and use the Text command to draw the labels. This command allows you to draw it non-centered, rotated left aligned, etc.

The spacing for the x and y tick marks are controlled in the axis entry. By default, the space for the y-axis expects the number to be about 3-4 digits. If you only have a single digit, such as the integers 1,2,3 this is too wide, so switch the space to 'Narrow'. This scales with the font size. This is used as part of the automatic margin calculation and to find how big the tick mark space is, and this will move the title.



7. Drawing Command Elements

One of the main differences between DataGraph and traditional drawing environments is how settings for drawing elements are handled. In most other graphing programs, when you see a line or text in a graph, the way to change the line thickness, etc. is to click on an element of the drawing, expect it to highlight in some way, and then, either use a dialog box (menu entry or double click) or an inspector window to change the properties behind that element. This same inspector would be used for each line, and you change them one at a time. These settings are stored somewhere behind the scenes for every entry, and just a sub-set is shown at any given time.

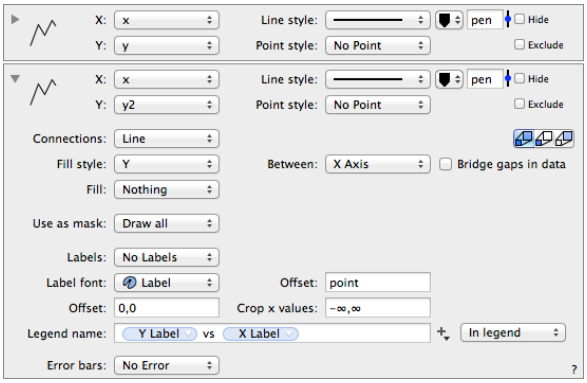
In DataGraph, every drawing element is visible in the 'drawing command list'. When you click on a drawing element, the drawing element does not highlight on the screen. Instead, the applicable drawing command is selected (highlighted with a blue border). The UI is always visible, even when the drawing element is not selected. You can copy settings between commands, compare them side by side and view all of the settings at the same time. Every change you make is applied immediately, and if you didn't mean to make a change, you can use the undo mechanism or change it back.

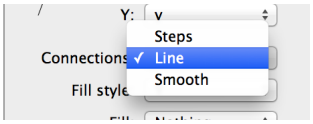
This chapter goes through the common building blocks used in all of the drawing commands. The next chapter explains the commands in more detail.

Level of Detail

There are two levels of detail for each drawing command object. The main view is always visible and is typically just two lines. These are the settings that you frequently adjust and gives enough information to get a sense of what is drawn. For example, in the case of a Plot command, the top two lines show the x and y columns that are used, the line style and point style.

When you open the little disclosure triangle you see the rest of the settings, such as if it should be filled, how to connect the markers, if you want a label printed at each data point, set the error bars, etc. In the lower right corner is a small ? button which will bring up the online help for that drawing command.



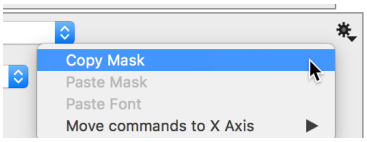


The UI is fairly compressed, and throughout the UI, as you select menu entries, you can expect additional details to show up to the right of the menu. As an example, consider the line options for the Plot command. The default is to connect the points by straight lines. But there are two other options, Steps and

Smooth. If you pick Steps there are three different step styles, so to the right of the Connections menu you get a second menu which allows you to pick which style. The smooth connection is a spline, and there are two different types of splines that you can pick from a menu. With very few exceptions, selecting an entry in a menu will only add or remove user elements to the right or below the menu.

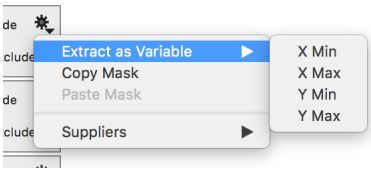
Gear menu

The gear menu on the right hand side of the drawing group also has some very handy short cuts. You can use the gear menu to copy a shared mask to another drawing group. If you have a split axis on your graph you will also be able to move all the drawing commands in the group to another axis.



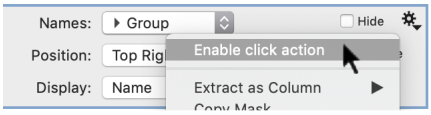
Several commands also have the ability to extract data from the gear menu. For example, the gear menu in the **Label**, **Bracket**, **Region**, and **Range** commands

can be used to extract the locations that define these annotation elements into a variable.

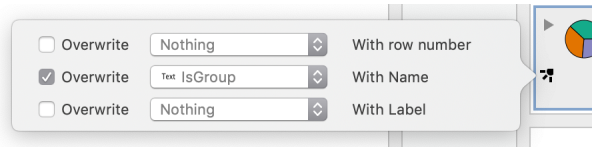


Click Events

For the some commands, you can set up Click Events using the gear menu, to create interactive graphic and dashboards.

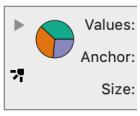


After you select "Enable click action", a pop-up menu is shown with the options available, which depend on the type of command..



In the above example, the Click action is set up to overwrite a text variable called "IsGroup". That variable can be used as a Mask in other commands. This means a click on a Pie Chart can be used to select the data that is drawn in another graphic.

When you move the mouse away, the pop-up menu vanishes but the small icon remains. Click this icon anytime to activate the pop-up menu.

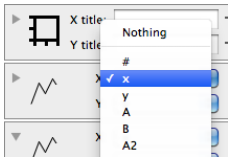


You can also deactivate the click events under the gear menu. Your last selections will be remembered if you toggle it back on.

Click Events can be created for the following commands: Points, Bars, Pivot, and Pie.

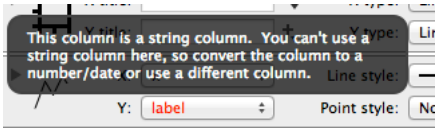
Selecting Data Columns

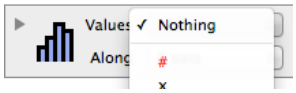
For drawing commands that use input from the data table, the way you refer to the data is by selecting the column from a menu. The menu shows the columns in the order that they appear in the column list. Any column groups will show up as a sub-menu.



You can also drag a column from the column list and on top of the column menu. This selects that column.

Once a column has been selected, you can change the name of the column and move it around. The column selector keeps track of it. Columns that cannot be used, such as a Text column if you need a numerical column, are drawn in red. You can still select them, but they won't be used, and typically nothing will be drawn. When you select the column, DataGraph pops up an error message that explains why the entry is red. Many drawing commands cannot use the # column. This is because # differs from a normal column in that it has no length.





Hide/Exclude

Each drawing command has a **Hide** and **Exclude** check box in the right hand corner of each drawing command that allows you to remove the command from being drawn without having to delete the command.

Hide means that even if the graphic is not visible it affects the bounding ranges for the axes. One common use is to use the hide check box to toggle a plot or point on and off.

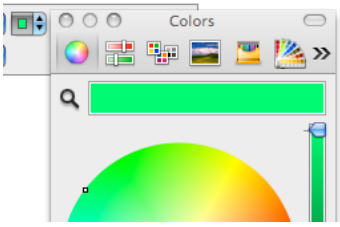
The **Exclude** check box means that the range of the lists is not used to determine the range of the axis. That means that the content is essentially part of the background. Thus, if there is a line at x=10 it won't be visible unless the range includes the value. If the check box is not selected then x=10 is considered

part of the range and unless it is explicitly restricted the x-axis will include that value.

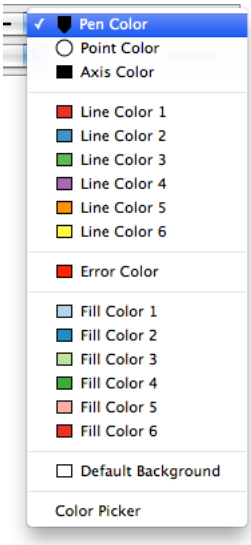
Histograms and fits are still computed even if they are hidden or excluded, so you can use the results in text fields.

Color

You can tweak the color for every drawing element. For a good graph you should use color sparingly and consistently. Often you want to use the same color for a number of different elements, and rather than having to set the same color over and over, DataGraph by default sets up a color theme and you can pick colors from there. You can still specify colors for a single entry and have hundreds of colors in the same graph.

To explain this mechanism, consider the color picker that controls the line color  for a plot. This is a menu, and when you click on that menu you will see 17 preset colors and at the end the 'Color Picker'. If you select the color picker you get 

a small rectangular color tile. When you click on the border around the tile, you bring up the standard color picker from the operating system and you can pick the color. Click on the tile again, or click on a different tile to deselect the color and stop sending color changes to this color selector. You can drag the color tile onto another color tile. The problem with setting each color by using the color picker is that it is tedious to change a color if you decide that you want a slightly different hue of green. You then need to change several drawing elements. This is why it is better to use one

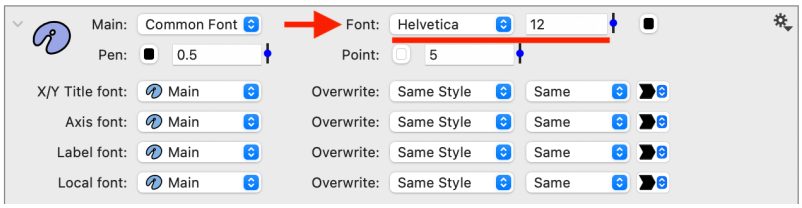


of the 17 predefined colors by selecting them from the menu. The colors are defined in the style settings. The pen and point are defined at the top, and the line, error, fill and background color are defined there as well. These are defined with the standard color pickers. So consider referring to those colors, and then set those using the color picker.

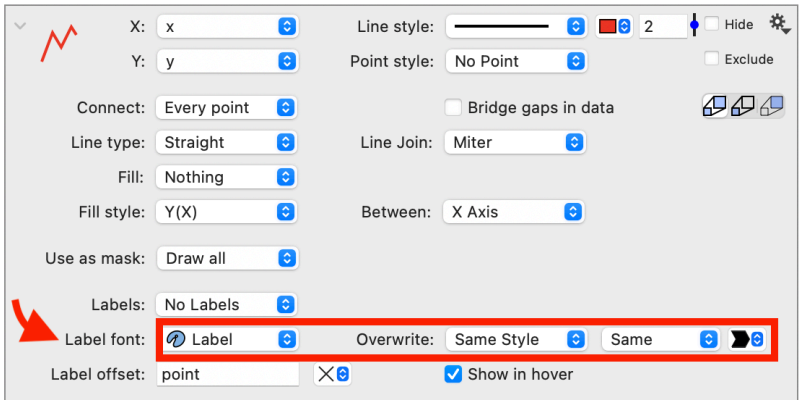
Font Style

Font specification is set up similar to the color picker in that by default, DataGraph sets up font styles that make it easy to use a uniform font, sizes that are consistent and when you change the main font everything changes with it.

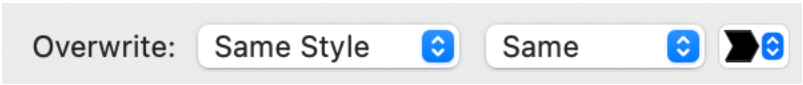
At the very top, the first line is the main font. In the detail settings are four more font styles - X/Y Title, Axis, Label and Local. Drawing commands will then refer to those fonts and allow you to overwrite aspects of them.



When a drawing command uses a font, the first step is to select the font style from a menu. There you can pick from the font styles defined in the style settings or variables. You can then overwrite certain aspects of the font. This is done by the next three menus to the right of the font menu.

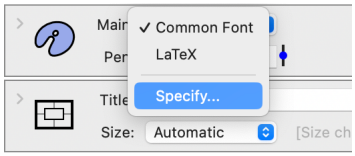


There are three things you can **Overwrite**. The first is the font style, as in changing it to italic or bold. The next menu allows you to overwrite the size. This is relative, so you specify how much the font size should be changed up or down, and says "Same" by default.

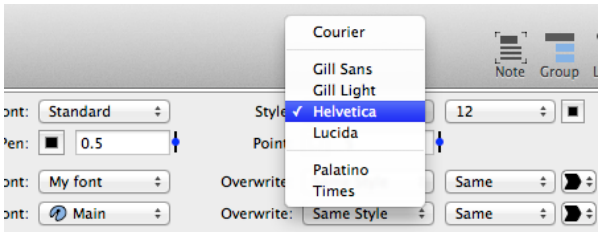


The third menu allows you to overwrite the color. The top entry just uses the color you defined in the parent font, but you can choose a color from the presets defined in the style settings or overwrite it using a color picker.

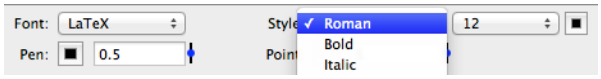
The style settings also allows you to overwrite aspects of the main font when specifying a font style in the same way. Here 'My font' is defined in the variable list. **Main** is the font defined in the first line.



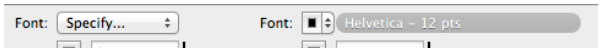
The **Main** font also has a font selector that allows you pick from common fonts that are included with the system. Since the other fonts typically are all based from this font, everything changes consistently.



Another option for the main font is the LaTeX font. This is intended for people that use the LaTeX package to typeset papers, and this way the graph will have a consistent look.



All font selectors then have a 'Specify...' option that allows you to specify the font by using the standard font picker from Apple. Just click on the font name to bring up the font selector.

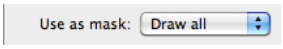


Mask

When you select columns, the drawing command will look at all of the rows in those columns. As you add/remove columns the drawing command is immediately updated. However, in some cases you might want to only use a sub-

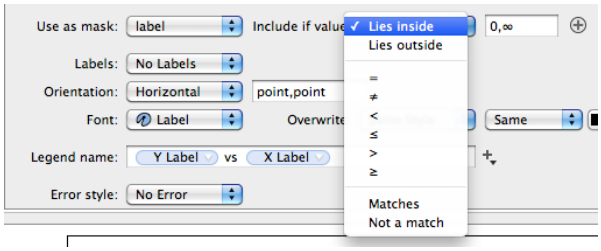
set of the data. For example you want to only create a scatter plot for rows that have a particular label, take histograms only when a given value lies in a specified range, etc. This is what DataGraph calls a mask.

You can specify which rows to include, and any other rows are removed for the purposes of that drawing command. For example in the Points command, which is used for scatter graphs, by default the mask entry is 'Draw all', indicating that every row will be used.

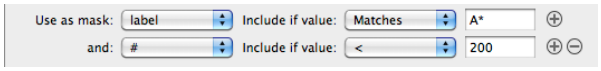


You can select any column here, so you can mask based on a column that is different from the x or y coordinate. Once you select the column, new entries will show up on the right of this menu.

The next entry you change is how to determine if a row in the column should be included or not. The first two entries use a range, but for <, > ,... this range changes into a specific number. Note that both the number and the range can use variables, so you can specify variables a,b and set the range to 'a,b' so that you can easily change a and b using a slider.

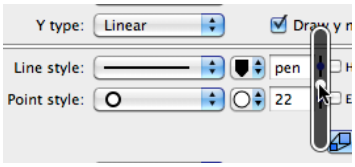


Using the small + button to the right you can add additional masking lines. This allows you to limit the mask further.



Pop up slider

There are a lot of numerical fields in DataGraph. They control point sizes, line widths, offsets, histogram bin width, etc. When you enter in a new number you have to close the field before the change takes effect. You can close it by either hitting return, tabbing out of it or clicking with the mouse somewhere to change the focus. Changes are made immediately, but to make this even faster and more immediate DataGraph allows you to change many of those values by using a pop up slider.



Next to most of the numerical fields is a small icon of a slider, a black slider with a small blue button. When you click on that, a small black slider pops up. The range of this slider is automatically determined based on the current value and the valid range for that parameter. For example to change the marker size, just click and drag the mouse until you get a size that you like. If the range is not sufficient, move the mouse all the way to the top/bottom of the slider, release the mouse and click again. Since the range is determined by the current value, the slider range has changed the next time you click it.

Text labels

Text labels are used in a number of places including: graph titles, axis titles, and labeling specific points within a graph. You can put labels at individual data points, to mark connecting lines, inside legends, etc.

There are two sources for text, either you specify it in a text field inside the drawing command or it comes from a data column. A text field allows you a lot of flexibility to build up the text, and you can use variables and properties from inside drawing commands. For example, you can put the result of a fit command inside a text label, axis title etc using the token mechanism. When you use a column, either numerical or text, you can adjust the formatting to set the number of digits for numbers, append and prepend text such as currency, etc.

The text layout is explained in more detail in the online documentation. The next sections explain some of the most important and frequently used aspects.

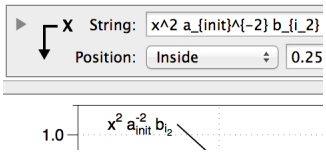
Greek/Special Characters

Every text field as well as the data table supports unicode characters. That means that in addition to cyrillic, greek, accented characters etc you have access to a lot of the symbols such as daggers, copyright signs, etc. The standard method to insert these characters into text is to bring up the Font panel and click

on the Character... palette. This is certainly possible in DataGraph as well, but involves a number of clicks. DataGraph speeds this up by using notation borrowed from TeX (LaTeX). LaTeX is not what you would call a WYSIWYG editor, and even though it is used extensively in math and physics it isn't used that much. But TeX introduces a very compact notation to insert characters and commands using the \ character. So the character α is written as \alpha followed by a space. Similarly \beta, \gamma, \delta, \nu, \mu, etc.

This is used for a lot more special characters, and they are listed in the online help. Or you can look for a LaTeX reference. Some examples are \euro, \pound, \permille, \leftarrow, \longleftarrow.

Super/Subscript



Super and subscripts are implemented by using the special characters ^ and _. This means that if you want to type x squared you use x^2, And sub scripts are done with _, so if you want to create a subscript 0, type a_0. The sub/super script only works for a single character. To apply it to a longer text string, enclose the text in curly braces, such as a_{10}.

Since \, ^, _, {, } have special meaning, you have to use a workaround if you want to use them in a label. You do that by putting \ in front, that is \\, \^, _, \{, \}

Math symbols

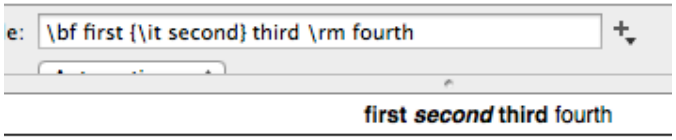
In addition to the greek characters and super/sub scripts, DataGraph contains some basic equation structures such as square roots, fractions and sums. If you have very complicated equations you might want to use a small LaTeX to PDF utility such as LaTeXiT where you can use the full power of LaTeX, but in order to use that utility you need to install the entire TeX package. The subset that DataGraph implements is completely independent of any TeX distribution.

The main equations structures that are implemented in DataGraph are \frac for fractions, \sum for sums and \sqrt for square roots. So for example \frac 14 is the 1/4th fraction. If you want multiple characters, enclose them in { }, for example \frac {17}{19}. \sqrt 2 is the square root of 2, but \sqrt{23} is the square root of 23. For sums you can for example use \sum_{n=0}^{\infty}. Note that the expression parser is also fully unicode compliant.

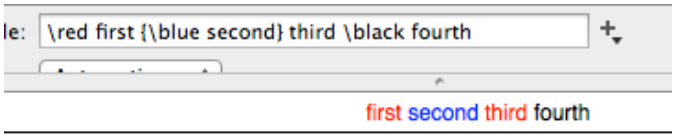
Style and color

You specify the font for the text label by using a font. This means that you can't use different fonts in the same label. Where people typically need that is if they want to make a word bold or change the color of a part of the text. DataGraph

supports this using a different mechanism, and one that works better with the token mechanism. To switch to bold use the \bf or \bold command. Similarly \it (\italic) and \rm (\plain). This is a switch that is then in effect until something else is specified. You can use {} to limit the range, for example 'this {\bf box} is bold' is equivalent to 'this \bf box \plain is bold'.

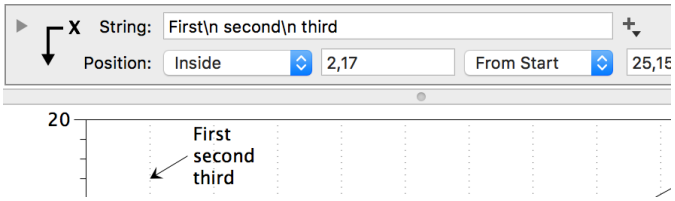


Color is specified similarly. DataGraph understands \red, green, \blue, \black, \white



Multiple Lines

If you paste in multiple lines from a word processor, DataGraph will draw multiple lines for the label. But typing in a multiple line text label is not immediately obvious, since hitting the return key will close the text field. DataGraph does not wrap long lines automatically, so you have to specify where the line break should be. Typing the line in a different application and pasting it in is tedious, so DataGraph has a command to insert a line break. This is the \n command. So 'First line\n Second line' will give you a two line label. Note that the space after the n in \n is crucial since there are several commands that start with n.

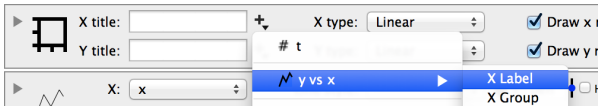


The space after \n is absorbed so if you want a space at the beginning of the next line you need to put two spaces after \n.

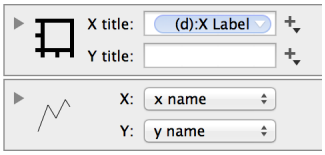
Tokens

Tokens give you a way to include a dynamic link in a text label. To the right of the text field there is a small + menu where you can create a link. The link points to either a variable or a property inside a drawing command.

For example, the axis settings entry has fields for the x and y titles. If you click on the + button to the right of the X title, there is a sub menu for the plot command inside the graph. If you select the 'X Label' entry you get a blue token in the text field.



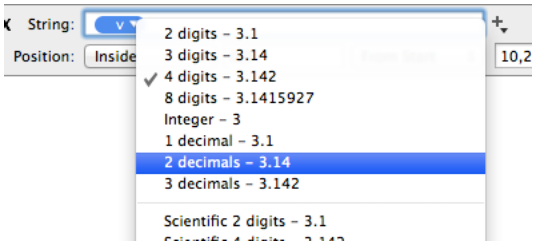
This token is the name of the column that is selected. So if you select a different column in the plot command, the x title will change.



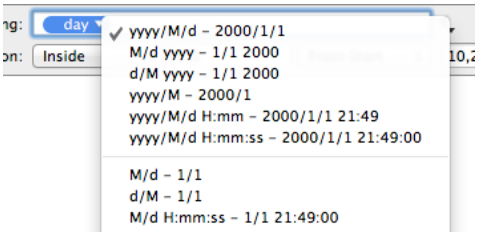
You can also use tokens to extract a fit parameter from a Fit function or statistic from a Histogram command. The legend field inside the plot drawing command uses tokens to create the 'y vs x' name, as shown below.

Formatting

Number tokens, either from a variable or a fit parameter etc need to be converted into text. You can adjust the format that is used by clicking on the small pop-up menu in the token. This also previews what the number will look like formatted.



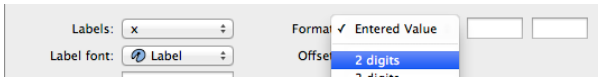
If the parameter is a date, the format options are different.



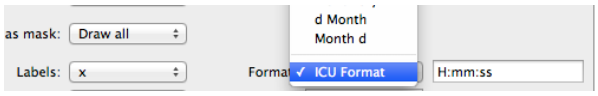
Labels

Several drawing commands allow you to display values from the data table or computed results as text labels. For example, the Plot and Point commands allow you to draw a label at each x,y coordinate. The Bar command allows you to draw a label at each bar in a graph. The Pivot command can draw the underlying value on top of each bar.

For each of the above examples, you can set the format that is used to convert numerical values to text. There are two types of formatters, one for numbers and another for dates.



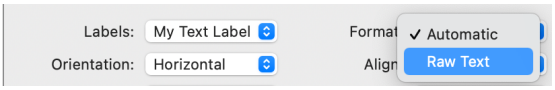
For example, you can adjust the number format and set a prefix/postfix for every label. You can add a currency label such as \euro , \pound, or $ before or after each number. If the column is a date column, the formatting options change so that you can specify the day format from a preset or your own format string.



Text Format

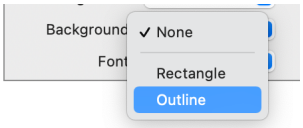
When a text column is selected as a label, the Format menu shows the default option of ‘Automatic’. When this option is used the character formatting described in the previous sections will be applied to display the label (e.g., math symbols, subscripts, etc.).

If you to show the text as is, change the **Format** menu to ‘Raw Text’.



Background

In several commands, labels have a **Background** option. By default the background is set to ‘None’, but can be changed to show a rectangular shape or an outline around the text, referred to as a stroke in graphic design.

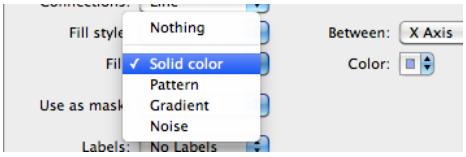


You can control the color and the width of the outline. The font outline is available in the **Text**, **Label**, **Points** and **Scalar field** commands.

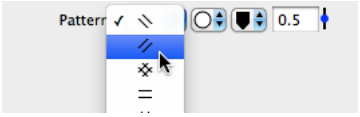
Text Text Text

Fill Style

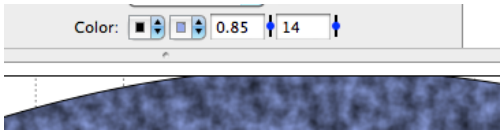
Several drawing commands allow you to fill in a solid region. The fill settings are similar between drawing commands. The default is typically no fill, and as you select the fill method, the view to the right of the menu contains additional details. For example, the 'Solid color' setting shows a single color selector.



The pattern allows you to vary the hash pattern, background color, line color and width. This pattern is done by drawing lines and not a bitmap.



The Noise fill option is a fill based on a procedural texture. This is a fractal noise function, called Perlin Noise. You can adjust the background and foreground color, the weight and noise size.

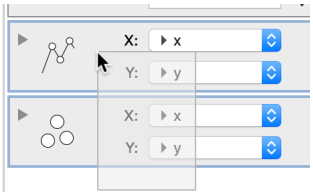


Perlin noise is a spatially coherent noise function between 0 and 1. If the weight (field on the left) is 1, the noise is used to switch between the colors. If the weight is 0.5, the noise is scaled by 0.5 and then used to switch between the colors. The field on the right is the grain size.

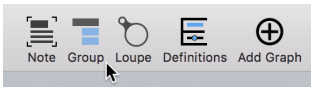
For interesting effects you can set the colors to be semi-transparent. The noise is computed dynamically and when you print or export to pdf the noise is computed at a higher resolution than the screen resolution so it will print properly. This is implemented as a bitmap, so the pdf/eps output file can be large.

Drawing Groups

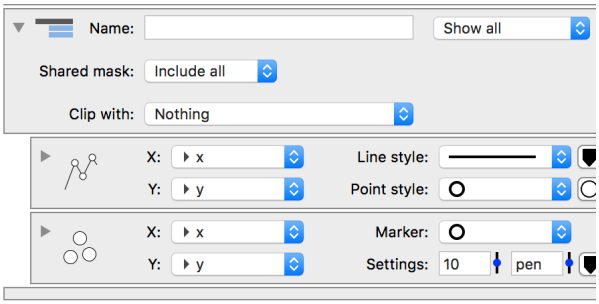
When you have more than one drawing command for a graph, you can combine them into a Drawing Group. There are two ways to create a drawing group. You can first drag the cursor over the commands you want to group, as shown below.



Next you add a group using either the icon in the top right hand corner, as shown below, or by selecting **Command\Add Drawing Group** from the file menu.



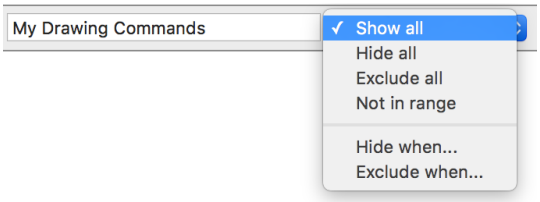
The highlighted drawing commands will now be in the drawing group. You can also create the group first and then drag the commands into the group. Either way, the group is created as shown below with the discloser triangle showing so you can see the contents of the group.



You can name the group and close the disclosure triangle as a way to organize your work space.

You also have the ability to very easily toggle data from being displayed on your graph. By default the group is set to 'Show all', meaning that all the drawing commands in the group are included in the graph. Note that they are also used to define the extent of the axis range, along with any other drawing commands that may be in the graph.

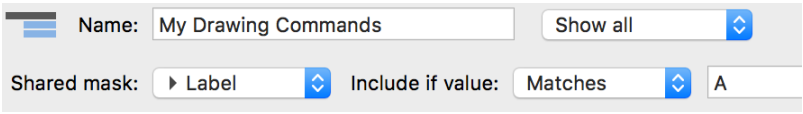
If you want to exclude a group of drawing commands from being drawn DataGraph gives you several options: 'Hide all' hides the drawing commands but will not change the extent of the axis; 'Exclude all' removes the drawing command from the graph completely and will not consider them in determining the extent of the axis; and 'Not in range' will still draw the drawing commands in the group but does not consider them when determining the range of the axis for the graph.



There are also two conditional Hide and Exclude options that can be useful for animating your graphs.

Groups can also be very useful for modifying the data that is being drawn by your commands. You can drag and drop a data group onto a drawing group and the column references will all be replaced, assuming you have columns in the data group that correspond to the names used in the drawing commands.

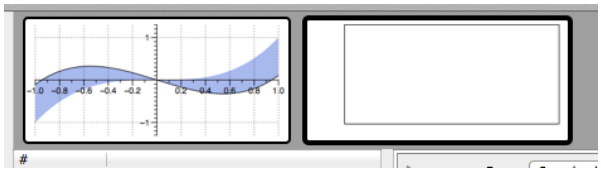
You can also add a mask at the drawing group level that allows you mask the data used in the drawing commands without having to add a mask to each drawing command individually. For example, you can have a categorical column used to limit the data that is drawn based on a text value. These can also be tied to variables as described further in that Chapter.



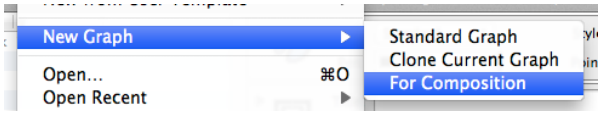
Multiple Graphs

DataGraph allows you to create multiple graphs from the same data set, and create composite graphs. To create a new graph, click on the **Add Graph** button that is in the toolbar.

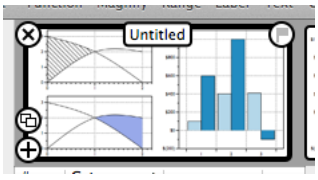
This adds a new empty graph to the DataGraph file.



If you hold down the option key you get a duplicate of the current graph. You can also use the New Graph sub menu in the File menu to create the graph.



At the top of the window you get a horizontal list of the current graphs. Each graph is updated in real time so as you change the data you see the changes in all of the graphs. For graphs that do not have a specified size, the size is given by the size of the current drawing view. As you change this view, either by changing the window or changing the splitters, the other graphs are updated.



When you move the mouse over a graph thumbnail you will notice a few controls. On the top left side you see a button to delete this graph (undo if you accidentally hit this). Below that is the clone graph button and the button to create an empty graph. At the top is a text field so you can name the graph. This is partly for organization, but also to be able to export multiple graphs at the same time (explained in the next section). If you don't specify a name the 'Untitled' tag will vanish when you move your mouse away. In the top right is the flag check box. This is also currently only used for exporting graphs.

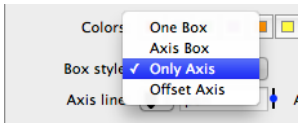
You can also drag the thumbnails to reorder the graphs or drag a command onto a thumbnail to add a copy of the command to that graph. Note also that you can select the style, canvas and axis and copy them to the clipboard. Then you can go to a new graph and paste in the settings. This is one way to propagate style changes to other graphs.

Combining graphs

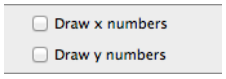
The split axis allows you to create multiple axes which share an x/y range. But in some cases you want to put two graphs next to each other. To do this in DataGraph, create each graph as a separate graph page, and then create an additional graph to combine them.

This takes a little bit of tweaking, since by default a new graph is set up to draw a graph so it has an axis. The 'For Composition' option in the **New Graph** menu does the following three steps.

Step 1: Set the axis style to 'Only Axis'. This stops you from drawing the box around the graph.

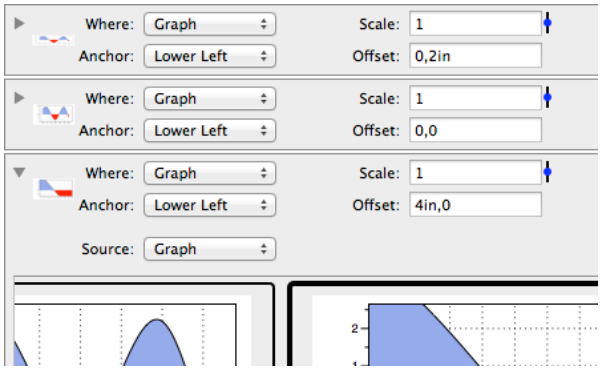


Step 2: Turn off numbers for the x and y-axis in the axis settings. This removes the remaining two lines.



Step 3: Set the margins around the axis to 0. This means that the underlying axis fills the entire graph region.

There is still an axis in the graph, and if you draw any plots it will show up. But for compositing other graphs, all you need are a number of **Graphic** commands. The **Graphic** command is then set up so that the source is one of the other Graphs in the file. Use the offset field to position them.

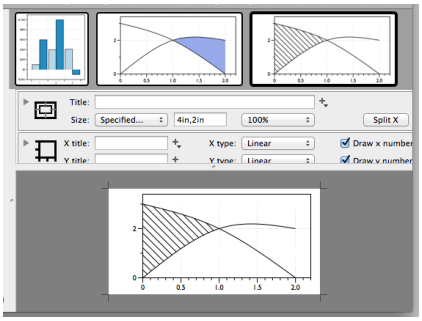


It is recommended that you specify the size for all of the graphs, including the composite graphs so that the positioning and size of the graphs is more predictable. This also allows you to align the graphs. You align them along the boundary of the drawing canvas, but if the margin computations are the same the axes are aligned as well.

Example: Compositing Multiple Graphs

Take as an example a case where you want to draw three graphs in one image. If they have the same y-axis range, you might want to use the Split axis functionality; however, if they are truly independent create three separate graphs and combine as described below.

First step is to create three separate graphs in the same DataGraph file.



By default a graph fills the lower right corner of the DataGraph window and will resize as you change the window size. For compositing graphs this is not helpful, so specify the graph sizes exactly by using the Specified... option in the canvas.

This will set the canvas size exactly. If the result is going to be printed the graph size will be surprisingly small on the screen so set the magnification level so that you can view things either in actual magnification (same size as printed) or in 2x,4x view.

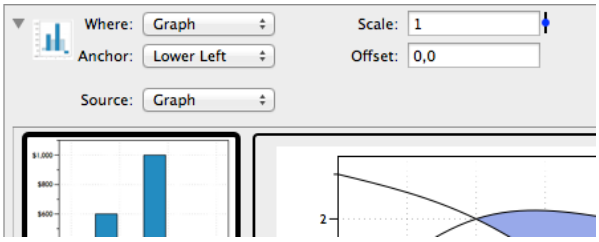
In this example, the first graph has size 4x4 inches, and the other two are 4x2 inches. This means that the composite graph will be 4 inches high and 8 inches wide.

Step 1: Create a composite graph from the File menu.

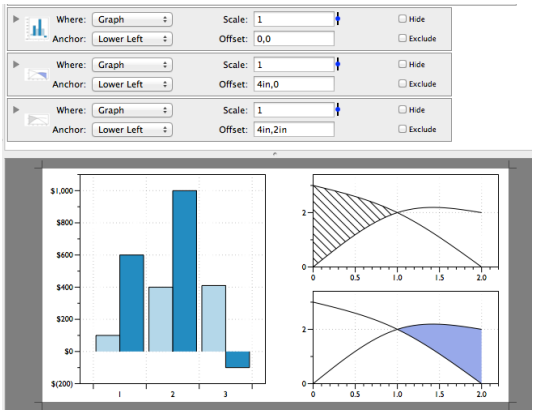
Step 2: Set the canvas size to 8in,4in

Step 3: There is one Graphic command already in the list, and select Source as 'Graph' and click on the graph that you want to display.

Step 4: Set the Anchor to lower left and the offset to 0,0. The graph command will look like this:



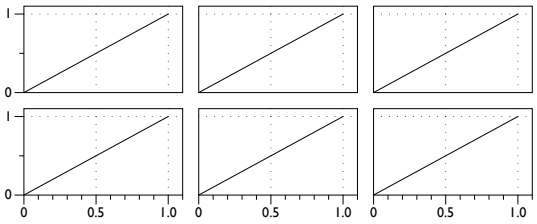
Step 5: Now create two more drawing commands. Set the anchor to lower left but in this case set the offset to '4in,0' and '4in,2in'.



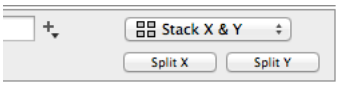
If the drawing commands have the same font sizes axis padding the axis will align exactly. If you change the style in one of the graphs, you will most likely want to copy that style over to the other graphs. To make this quick, go to one of the graphs, select the Style settings (top entry), select Copy and then go to the other graphs and select paste. You can toggle between the graphs by using the left and right arrow keys while holding down the command key.

Axis selector

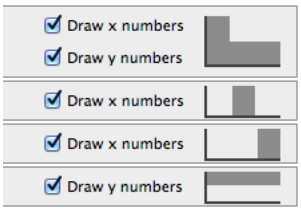
DataGraph allows you to break down a graph into sub-axes.



Each sub-axis is created by clicking on the Split X or Split Y buttons in the canvas settings.

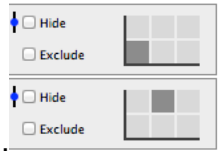


The standard axis settings is for the axis in the lower left corner, and each split axis gets it own settings where you can specify the axis type, ranges etc, just like the axis settings. To indicate what you are changing, there is a small schematic drawing on the far right side. You cannot change anything, it is just to indicate what portion of the graph you are adjusting.



Each drawing command changes a little bit. It gets wider and on the right you get a user interface that looks similar to the graphic in the split axis, but this is a controller you can change.

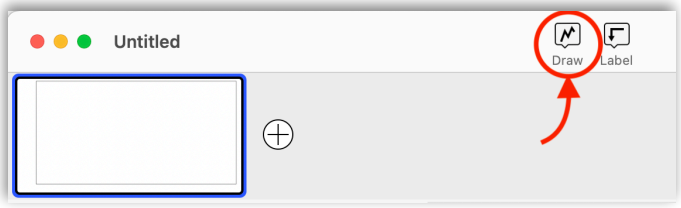
Each axis is drawn as a tile, and this does not change even when you join the X and Y-axis in the canvas settings. When you click on the tile, the drawing command is now drawn in this axis, and the axis ranges are adjusted accordingly



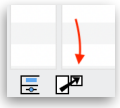
8. Graphing Data

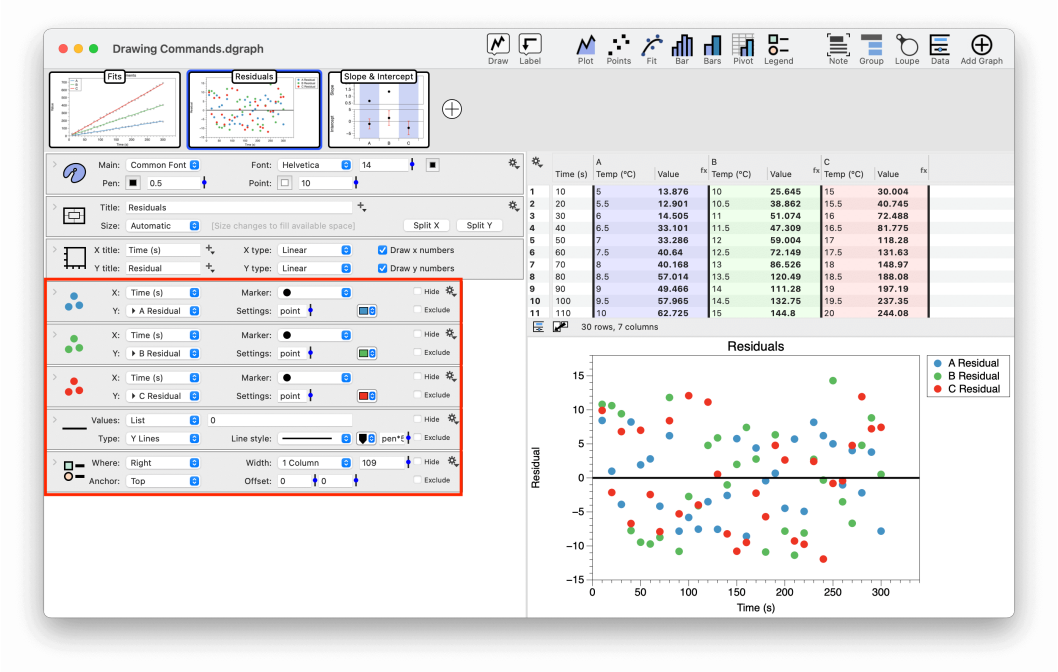
The previous chapter explained the building blocks for drawing commands and common elements that drawing commands share. This chapter goes through each commands that is listed under the **Draw** menu in the toolbar.

In the UI the commands are divided into two groups. The command that are used to graph and analyze data are can be created using the **Draw** icon in the toolbar.



The screenshot on this page has the data sidebar closed (⌘D) and the layout swapped. When you are using the commands, it can be helpful to swap the commands with the data, using **View > Swap Layout** (⇧⌘D), or click the Swap icon in the bottom left of the data table.





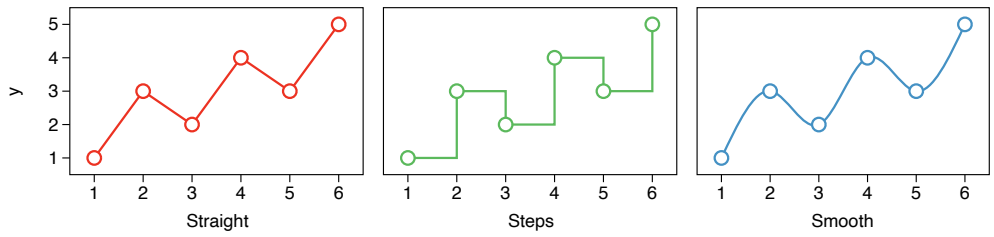
Commands in the Draw Menu

Icon	Name	Description	Example Types
	**Plot**	Draw straight, stepped, or smooth lines between a sequence of numbers (y) or points (x,y).	Trend lines Time series Line segments Fill Shapes
	**Plots**	Draw straight, stepped, or smooth lines between points with one x column and multiple y columns (x, y1, y2, y3, ... , yn).	Multiple lines Multiple colors Offset lines
	**Points**	Draw markers at (x,y) locations. Vary marker style, color, and size. Good for scatter plots and bubble graphs.	Scatter plots Bubble graphs
	**Bar**	Draw bars at specific x locations (numbers or dates). Vary start location. Use color schemes.	Bar plots Mosaic plots
	**Bars**	Draw grouped bars for categories of data or at row locations. Draw in X or Y direction.	Bar graphs Stacked bars Area graphs
	**Pie**	Create a pie chart using a single column of numbers. Vary start location. Can pin to a graph corner or an (x,y) location.	Pie chart Donut charts Windrose
	**Stocks**	Draw lines between start, end, high, and low. Useful for stock time series.	Stock graphs Candlesticks
	**Lines**	Draw horizontal or vertical lines line at a specified values (x or y) or input a column of values. Lines cross multiple stacked graphs.	Reference lines Stick graphs Gantt charts
	**Connect**	Draw line segments between x and y locations. Specify individual values or a column of values.	Slope graphs Vector fields Dendogram
	**Pivot**	Group data into rows/columns. Output counts, min, max, mean, SD, etc Output is similar to the Bars command, but for computed values.	Bar graphs Stacked Bars Area graphs
	**Scalar field**	Draw 2D fields over regular grids using color ramps and schemes to represent values. Use text or numbers to define grid.	Heat maps Stripped graph Conformal shapes
	**Histogram**	Represent a column of numbers using a counts or probabilities. Can draw stepped or smooth representations.	Histograms Density diagrams CDF/PDF

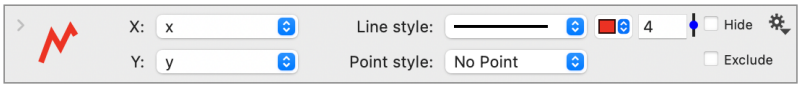
Icon	Name	Description	Example Types
	**Box**	Draw a variety of types for comparing distributions of data. By default shows a Box plot. Group data for multiple plots. Calculates summary statistics.	Box plots Point distributions Violin plots Sideways histograms Confidence intervals
	**Function**	Graphs an analytical function, y = f(x), over a specific intervals. Form continuous or discontinuous functions.	Line plots
	**Fit**	Conduct linear and nonlinear curve fitting for y = f(x). Use built-in functions or specify any arbitrary function. Fits trends with low order least squares (LOESS).	Line plots Regression LOESS
	**Multi-variable Fit**	Conduct linear and nonlinear curve fitting for y = f(x1,x2,x3,.. xn). Use built-in functions or specify any arbitrary function.	Point + Distance

Plot

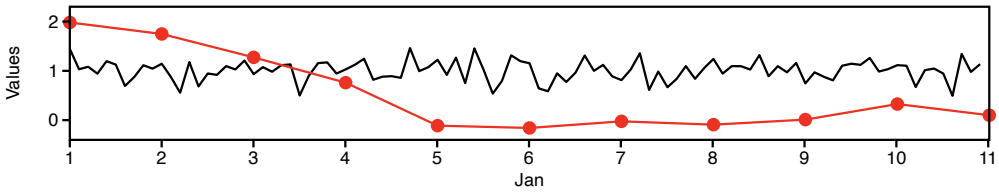
The **Plot** command has a variety of options for drawing lines between points. Draw a standard line plot with straight lines, or use steps or a smooth line.



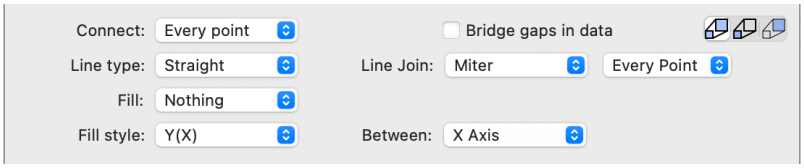
The main view of the command has a menu for the x and y columns. You can use number or date columns. The icons on the command will change with the style of the line. If you add solid points the color with match the line. If you add open points, you can vary the fill color.



Add multiple **Plot** commands to the same graph to draw lines for different x and y ranges. You can format them independently, with different colors, line styles, or markers.



Expand the command to control **Connections**, **Line Type**, and **Fill**.

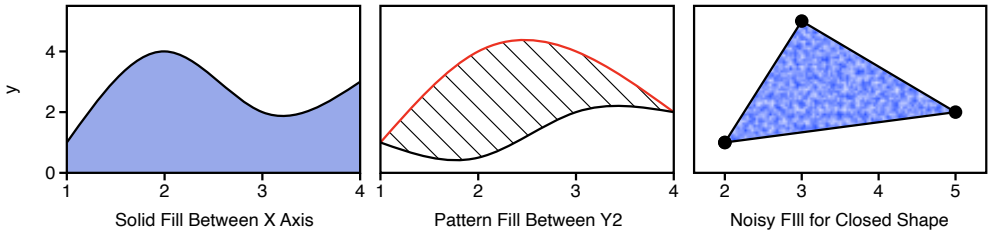


- By default every point is connected, but if you have a dataset where the x sequence repeats, you have to option to **Connect** only when x is increasing or decreasing, giving the look of multiple lines from one command

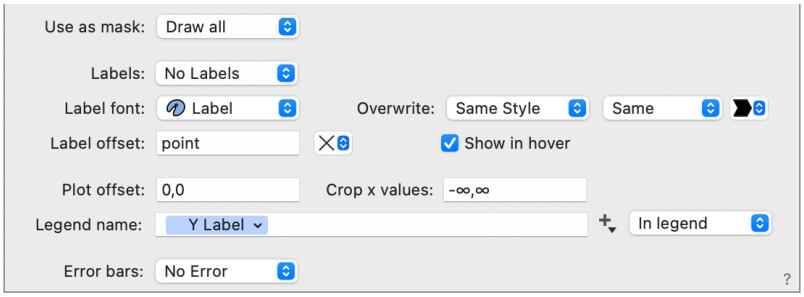
- If you have noisy data, change the **Line Join** from 'Miter' to 'Bevel' or 'Round'. Use the menu to the right to show points intermittently. Change from 'Every Point' to 'Stride'.

- The x values don't have to be increasing, but missing data or empty rows will cause gaps in the line. Check **Bridge gaps in data** to connect across missing data.

- **Fill** can go from the line to the x or y axis. Fill can also be added between two sets of Y values. If your coordinate locations begin and end at the same (x,y) location, the fill will draw inside the shape.



The bottom half of the command contains options for the **Mask**, **Labels**, **Offset**, **Crop**, **Legend**, and **Error bars.**

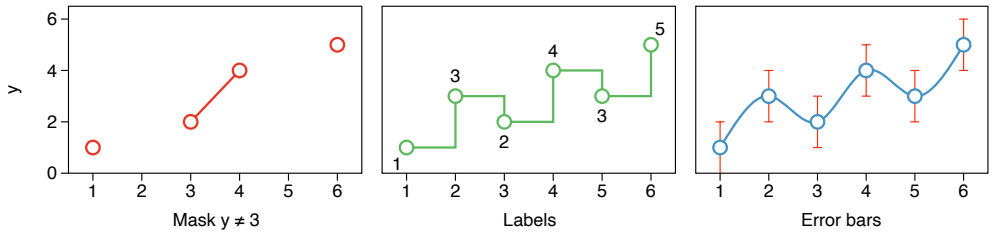


- If you add a **Mask**, a gap in the data is treated just like a blank row; thus, if **Bridge gaps in data** is not checked you might get a very spotty line between the points, or no line at all.

- **Labels** are above or below to avoid crossing the lines. Adding an offset moves the label further away from the line.

- **Legends** can either show 'In Legend' or at the "End of the Line' on the graph.

- **Crop** to hide a section of the line without changing the range. Useful for animated reveals using variables.

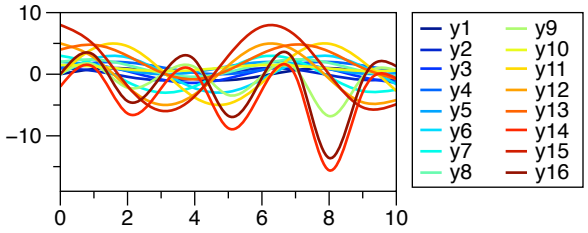


Plots

The **Plots** command draws lines from numeric data in multiple columns where you have a single x and multiple y's (as shown below) **OR** a single y and multiple x's.

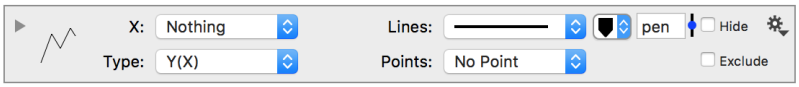
The **Plots** command allows you to create line plots with varying colors for each line more easily than using only the **Plot** command. For example, to create the graphic below you would need to use sixteen different "**Plot**" commands, while you only need one "**Plots**" command.

x	y1	y2	y3	y4
0	0	1	1	1
1	0.84	0.54	1.4	1.8
2	0.91	-0.42	0.49	1.9
3	0.14	-0.99	-0.85	1.1
4	-0.76	-0.65	-1.4	0.24
5	-0.96	0.28	-0.68	0.041
6	-0.28	0.96	0.68	0.72



- The Plots command has built-in color suggestions using Color Brewer color schemes.

- The Plots command is designed for data where you have one x column with multiple y columns in a table. Or one Y column with multiple x values.

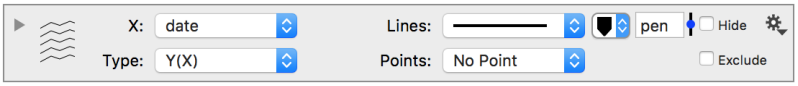


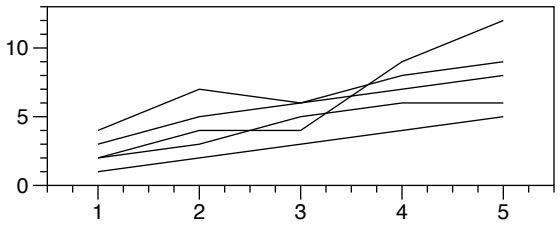
Use the drop-down menu to specify the 'X' column. The **Type** is used to toggle between a graph where the y values are a function of x, 'Y(X)', or x values as a function of y 'X(Y)'. In other words, you can easily change your lines to be oriented horizontally or vertically.

Another shortcut is to highlight all the columns in the DataTable that you want to include with the leftmost column being your x column.

#	date	1	2	3	4	5
1	1/1/2010	1	2	3	4	2
2	1/2/2010	2	3	5	7	4
3	1/3/2010	3	5	6	6	4
4	1/4/2010	4	6	7	8	9
5	1/5/2010	5	6	8	9	12
6						

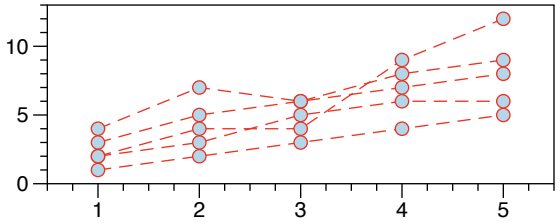
The following command is created, along with the following graph with a line for each column of data.





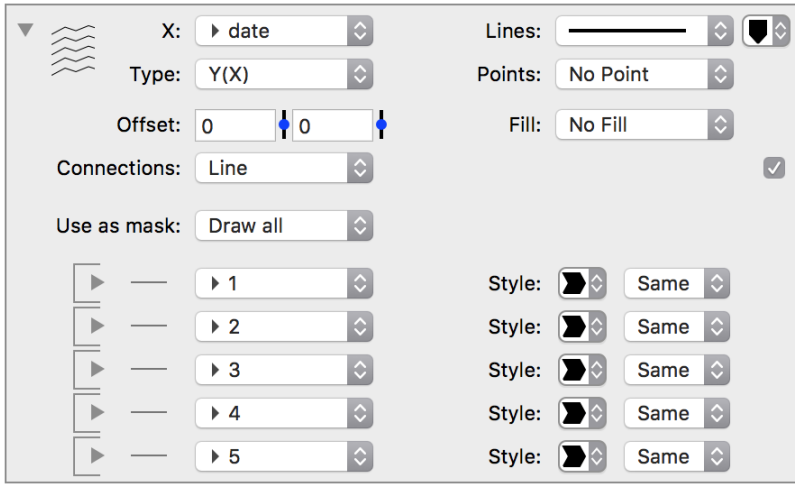
On the **Main view**, you can customize the **Lines** or **Points** settings.

This will change the setting for every line in the graphic.

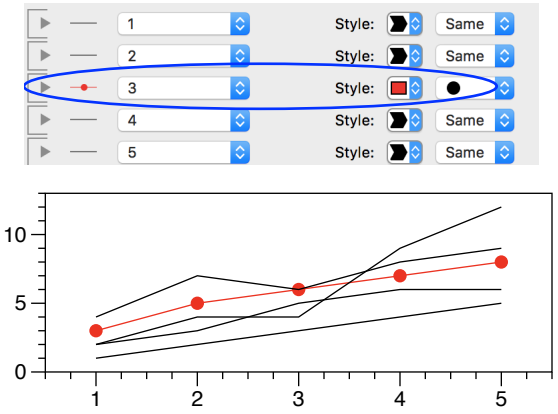


Detail View

If you only want to change one of the lines you need to open the **Detail view**. Here you will see a list of each of the lines being drawn.



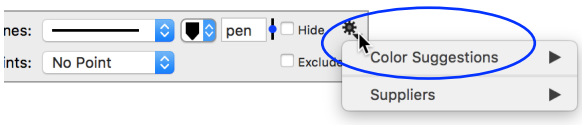
To the right, the **Style** settings are noted for each line. When the drop-down menu is set to 'Same', the points style is the same as indicated in the Main view. You can override these settings to customize the look of a particular line.



Example: Using Color Schemes

The real benefit of the **Plots** command is the ability to create multicolored line plots very quickly using the built-in color suggestions.

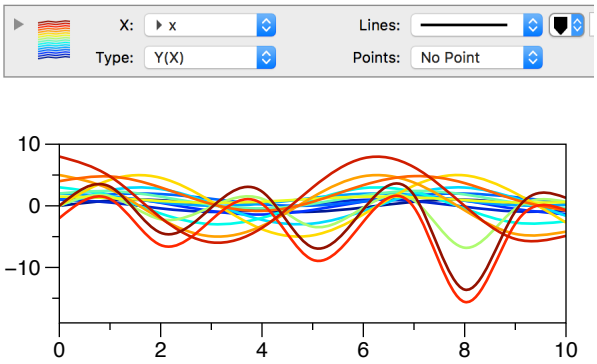
Click the gear menu in the right corner of the drawing command to reveal the Color Suggestions.



When you mouse over the Color Suggestions menu a list of possible color schemes will be provided. Note that this list will vary depending on the number

of items in your data set, such that there are more options available when you have fewer entries.

Selecting the **Rainbow** color scheme causes the custom icon to reflect the colors and results in the following graphic.



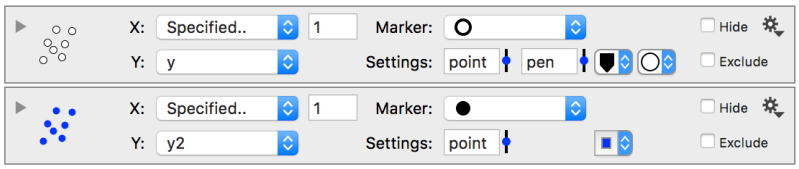
You can also create a custom color scheme using the 'Current Range' option. This works the same way as is described in the **Custom Color Schemes** section of the **Variables** Chapter**.**

Points

The **Points** command only draws the points at x,y locations and does not connect the points like the **Plot** command does. The **Points** command differs from drawing points using the **Plot** command in that the **Points** command has a lot more control over the markers. For example, the **Plot** command draws all of the points using the same color, fill, type and size while the **Points** command allows you to vary all of that based on additional columns.

Points command basics

The simplest way to create a scatter graph is to select the two columns you want to draw and hit the **Points** command shortcut.

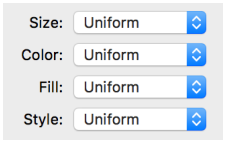


You can adjust the marker style using the **Marker** menu in the top right. Below the marker menu is the drawing style. There are two possible size fields, the size of the marker and the size of the line. Note that the solid markers don't have a line width, so the line width field is hidden along with the color of the line.

The icon on the left of the command is a custom icon that provides a preview of what the marker will look like. The size is in points, and by default the marker size is 'point' and line thickness is 'pen' which are values defined in the top entry, the **Style** settings.

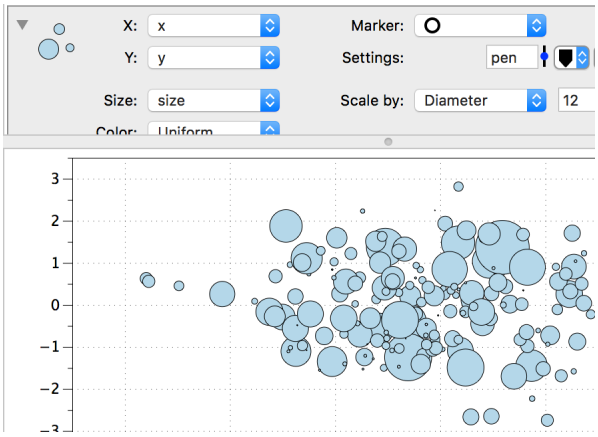
Additional settings

By opening the disclosure triangle on the left corner of the drawing command you access settings to customize the look of the markers based on the size, color, fill, and style.



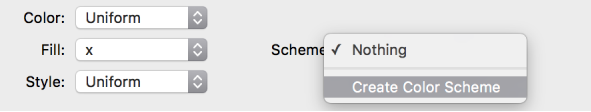
Marker Size

The first block in the detail settings allows you to change the marker based on additional columns. You can specify the size of the marker by using a column, and this is typically referred to as a bubble chart. When you specify a column, you get additional controls which set the scaling from data units to pixels. You can also specify the size to be scaled by the area or diameter of the marker.

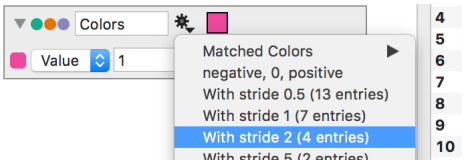


Using Color Schemes

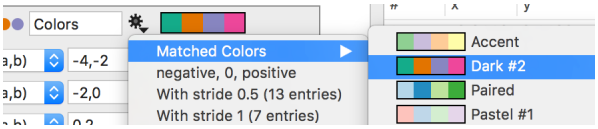
The **Color** and **Fill** menus allow you to vary the fill color and the line color based on a column. This functionality depends on a color scheme variable to set the color. Once you have selected a column you get a menu for 'Scheme'. Here you select the color scheme you want to use. The last entry allows you to create a new color scheme.



When you select 'Create Color Scheme', a new entry is added to the variable list. Here you can add matching rules that are very similar to the mask mechanism. You can also use the small gear menu to the right of the name to ask DataGraph to suggest matching rules.



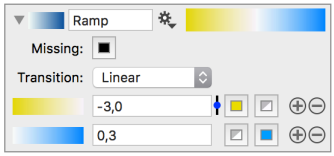
You can specify the colors manually, or use the gear menu again to pick from a predefined color set



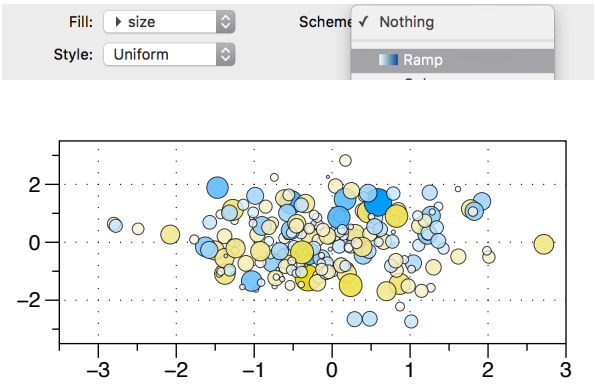
You can set up multiple coloring schemes and select between them by using the color scheme menu in the Points command. You can use a color scheme for the line color and fill color.

Using a Color Ramp

Color Ramps can be used to apply a continuous color scheme to points. Color ramps are created in the Variables section. Below is a Color Ramp variable named 'Ramp'.

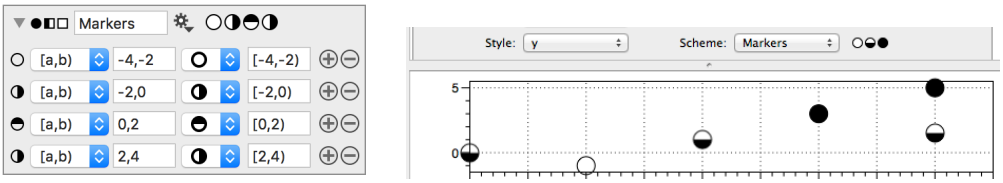


Once created, the Color Ramp will be shown as an option in the **Scheme** drop-down menu from the **Points** command.



Using a Marker Scheme

For the marker style you can use a marker scheme that works similarly, but allows you to put a white circle for negative, half filled between 0 and 3, and solid above three.



Bar

There are three commands that can be used to create bar graphs: the **Bar** command, the **Bars** command, and the **Pivot** command. Each of these commands take a different approach to handling data and which command will work better depends on the type of data you want to graph. In general, the **Bars** command has more functionally to handle multiple data sets while the **Bar** command can be used to create bars with varying color and size for a single data set. The **Pivot** command is for flattened data, organized in a minimal number of columns, and can calculate summary statics.

In the following sub-section, the use of the **Bar** command is compared further to the **Bars** command. For more detailed information on the **Pivot** or **Bars** command, refer to those sections later in this chapter.

Bar vs. Bars

There are two main benefits of the **Bar** command over the **Bars** command. One is that you can specify the location of the bars using a numerical column, such as

dates and non-integer values (e.g., 2.5). The **Bar** command also allows you to vary the fill and color of the bars for a single data set. For example, you can vary the color based on a separate column by using the Color Scheme fill option.

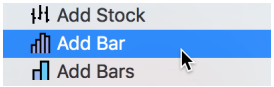
The **Bars** command is better for creating side-by-side bars or stacked bars where the x-axis is a text label. A summary of the main differences between the two commands is given in the following table.

Bar	Bars
Single data column only	Multiple data columns possible
X Locations are specified	X Locations are categories
Not easy to make side by side	Side by side or stacked bars are simple to create
Start is at X=0, can customize	Start location varies depending on categories
Width of each bar can be specified	Width is the same for all bars
Can vary color by bar	Bars from a single column all have the same color

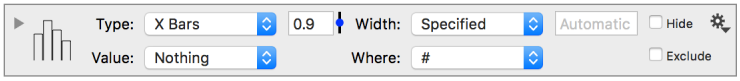
Bar command basics

The **Bar** command allows you to create a bar graph from one column of numbers, which is specified in the **Value** drop-down box. The X-axis location is specified using the **Where** drop-down box. By default, **Where** is set to the row number, or **#** column.

To add a Bar command to a graph, select Add Bar from the command menu.



The following drawing command will be created.

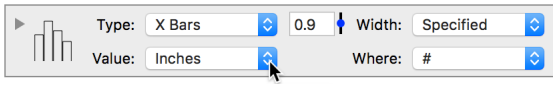


The height of the bars, or y-axis, is specified using the **Value** drop-down box. The x-axis location is specified using the **Where** drop-down box. By default, **Where** is set to the row number, or **#** column.

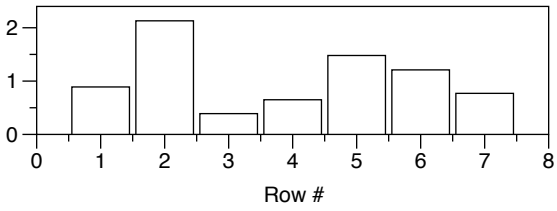
Consider the following data where we have values of a parameter (i.e., inches of precipitation) that occur intermittently.

#	Day	Inches
1	1	0.89
2	2	2.13
3	8	0.39
4	9	0.65
5	10	1.48
6	19	1.21
7	20	0.77

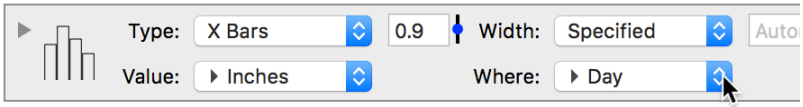
We first need to change the **Value** column to 'Inches'.



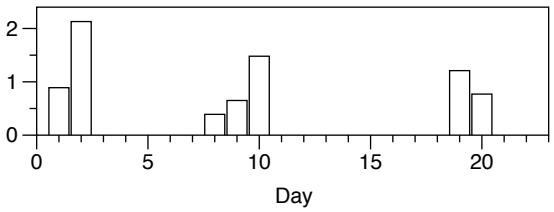
This creates a bar graph using our data but the x-axis is set to the default row location, or # column.



Next, we can change the setting for **Where** to the 'Day' column.



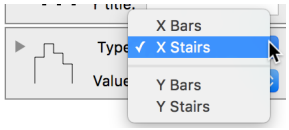
The graph now shows our data according to the numerical value for 'Day'.



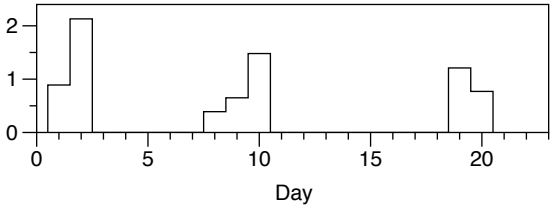
Note that the example shown here uses all integer values but they do not need to be. The bars will be centered on the exact number entered.

Bar Type

The top of the drawing command has a setting for **Type**. The **Type** of bar can be modified to change from 'Bars' to 'Stairs'. You can also change the orientation to have the values displayed on the y-axis.



This creates a graph where adjacent bars are combined as shown below.



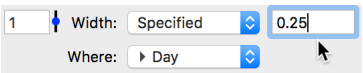
Bar Width

By default, the width of the bars is set to the smallest increment between data points multiplied by 0.9. In other words, the width of the bars is 90% of the distance between the closest adjacent points

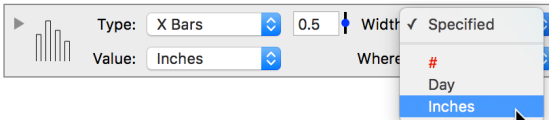
There are three ways to vary the width of the bars:

1. Use the toggle bar to the left of the **Width** drop-down box. The value in the box represents the fraction of space between points that the bars occupy. It can vary from zero to 1.

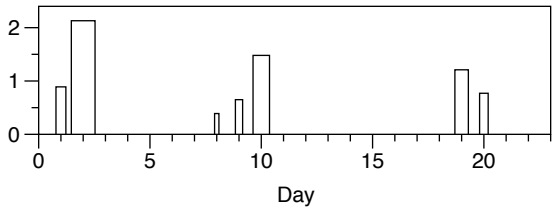
2. Specify the value of the width to the right of the **Width** drop-down box. If you also set the fraction to 1 the bar width will be exactly this length.



3. Set the width of each bar based on a column in the data table. You can even use one of the same columns as **Value**.

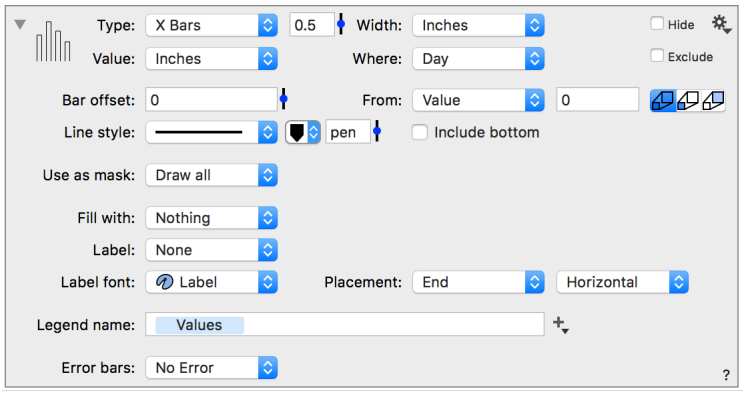


In the following graph, the ‘Inches’ column is used for both the **Value** and the **Width**, which has the effect of adding emphasis to the larger values.



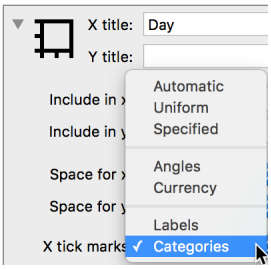
Additional Settings

Opening the disclosure triangle reveals additional ways that you can customize the look of the bar graph. This includes adding an offset, changing the outline of the bar, adding a fill, and adding data labels.



Specifying X-axis Labels

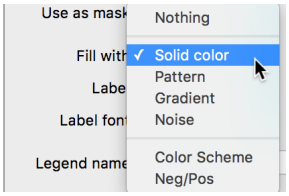
If you want to change the x-axis to use labels rather than displaying numbers, you need to adjust the tick-mark setting in the **Axis** settings to 'Categories' and select the Label column there.



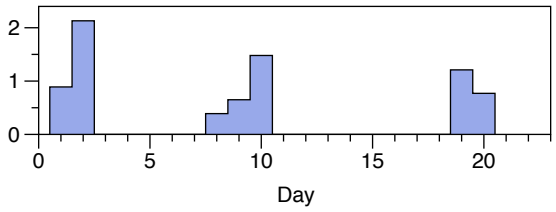
Note that in most cases, the **Bars** command is better suited for creating a graph with categorical data on the x-axis.

Bar Fill

The fill can be adjusted using the **Fill With** drop-down box.



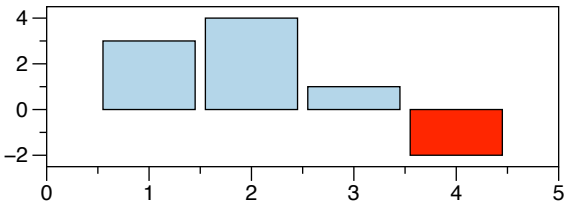
The fill can be set to a solid color as shown in the following graph. You can also vary the fill based on the data being plotted as discussed more in the following section.



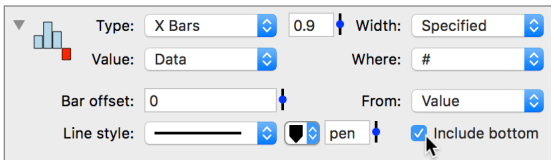
Negative/Positive

The **Bar** command allows you to use a different solid color for negative and positive values. Do that by choosing the fill method 'Neg/Pos' within the drawing command.

An example is given below using the **Bar** command with negative and positive set to different colors.

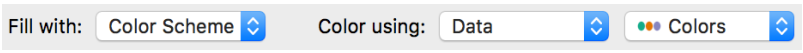


Note that for the previous graph, the **Include bottom** checkbox was selected, as shown below, to create bars with lines drawn all the way around them.

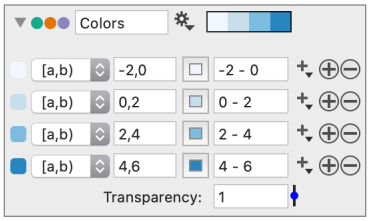


Color Scheme

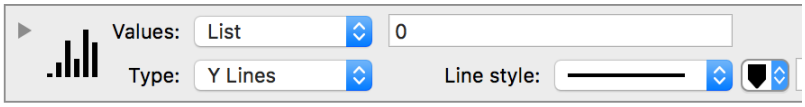
You can use a separate column to vary the color of the bar. To do that, select 'Color Scheme' as the fill method in the drawing command. Next, select the column you want to use as input to the color scheme. Lastly, select a color scheme variable to map the rows from the information column to a color.

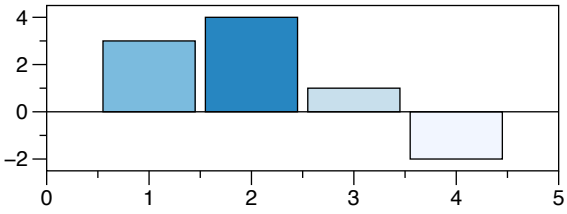


For this example, we are using the following color scheme variable named 'Colors'. Please see the **Variables** chapter for more information on creating a color scheme, and how you can use the gear menu and +/- buttons to specify the entries.



Our graph now has colors that are varying based on the value in the column named 'Data'. Note that we also added a drawing command to create a line at Y=0.



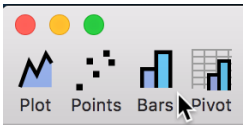


Bars

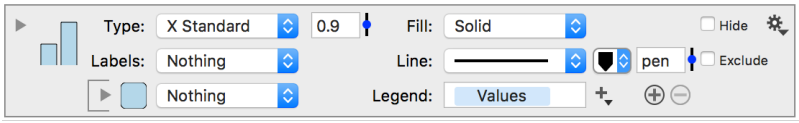
Bar graphs are the standard way to represent categorical data. The **Bars** command is specifically designed to work with this type of data. Note that DataGraph also includes the **Bar** command, which is useful for placing bars on a numerical x-axis. Please refer to the previous section for a description of the **Bar** command and more in-depth comparison between the **Bar** and **Bars** command.

Bars command basics

You can create a **Bars** command using the command menu or using the shortcut in the toolbar across the top.



The Bars command looks as follows where the default **Type** is an 'X Standard' graph.



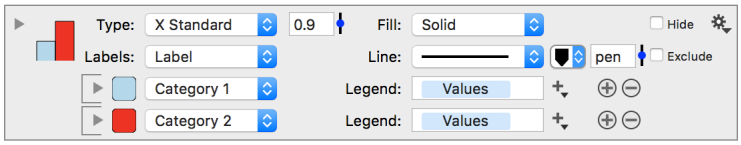
The **Labels** setting is used to designate the axis labels. Below the **Labels** drop-down box is a single line where you designate the data to plot. By default, these are both set to 'Nothing'.

To create the bar graph, you could manually use the drop-down boxes to specify the columns of data. You can also drag a column from the column definition list (far left) onto the **Bars** command to add one or more bars.

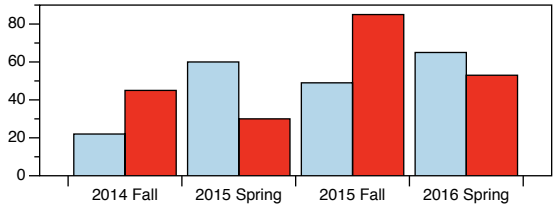
A short-cut is to have the columns of data selected before you click the Bars command icon. For example, consider the following data table where the 'Label' column is a text column and we have two data sets to plot.

#	Label	ab	Category 1	Category 2
1	2014 Fall		22	45
2	2015 Spring		60	30
3	2015 Fall		49	85
4	2016 Spring		65	53

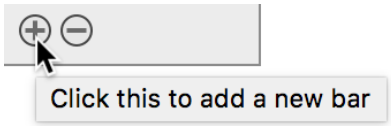
You can highlight all three columns and then click on the **Bars** command shortcut. The following drawing command is created where the **Labels** drop-down box is set to the first column, 'Label', and two lines are created below the **Labels** drop-down box for each column of data, 'Category 1' and 'Category 2'.



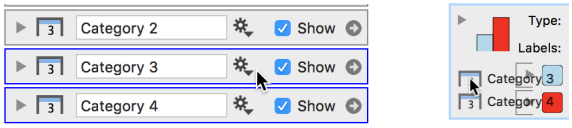
The above command produces the following graph.



To add or remove data, you can use +/- button pair next to each line of data in the drawing command.



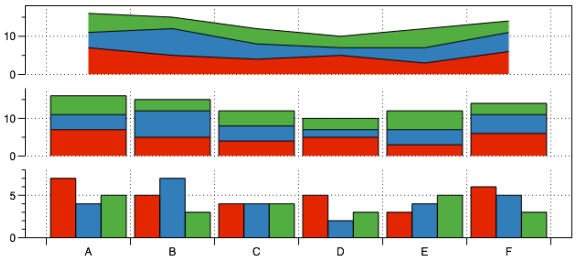
Another way to add data is to first highlight a column or columns on the column definition list (far left) and drag them onto a **Bars** command (as shown below from left to right).



You can also select a data entry in the **Bars** command (click on the small icon) and hit the delete key to remove or drag it around to reorder the bars. If a legend is added the order in the legend will be the same as the order in the drawing command.

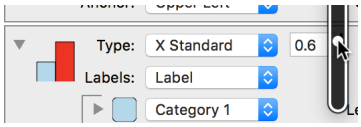
Bars Type

The type of bar graph drawn is specified in the first line in the command using the **Type** drop-down box. By default, the type is 'X standard', which has the bars drawn side by side. As demonstrated below, you can also select stacked and stacked area. You can also orient the bars along the y-axis.



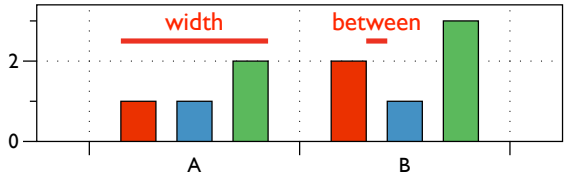
Bars Width

The graphical representation can be further tweaked by using the bar width to the right of the **Type** drop-down box. This changes the width for all the bars with the same label.



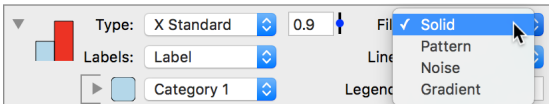
By default, bars with the same label are draw without any space between them. You can offset bars and add space between bars using commands located in the detailed view (See the Additional settings section below). The values for these settings are in grid units.

The graph below indicates what controls the bar width versus the **Between bars** setting.



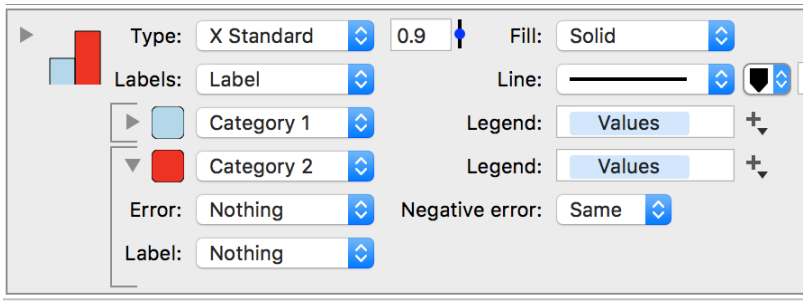
Bars Fill

The fill style is set through the **Fill** menu. There are four types of fill styles, and for each style you set the details further inside the drawing command.

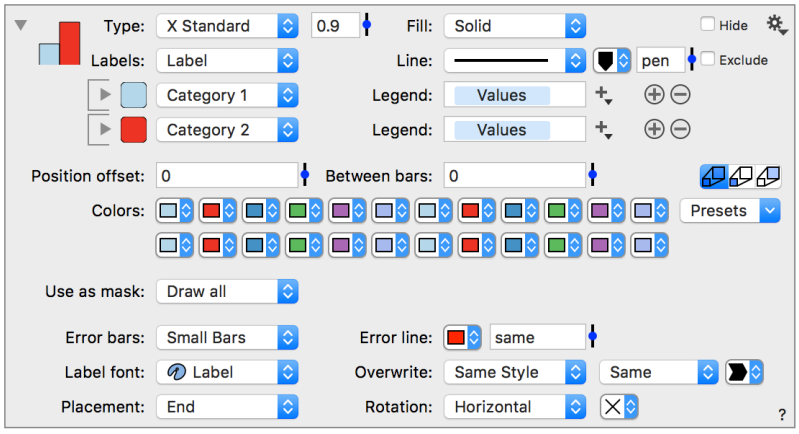


Additional settings

There are additional settings for each data entry that you can access by opening the disclosure triangle to the left of the color icon, as shown below for 'Category 2'. Here you can specify a column to use for error bars or specify a label.

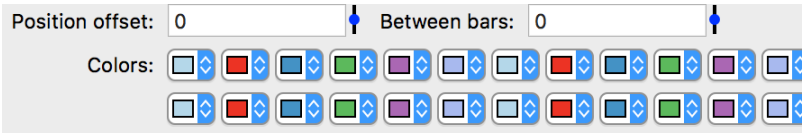


By opening the disclosure triangle, you reveal additional settings for the graph. At the bottom, you set the drawing style for settings that are set for each bar. For example how error bars are drawn, where the label is drawn in each bar, etc.



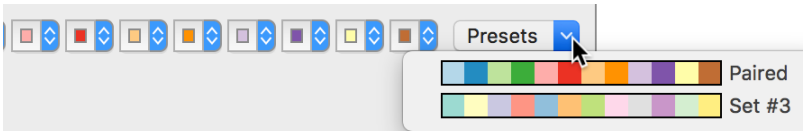
More Fill options

The detail in the settings is partially dependent on the **Fill** style selected. As shown below, the 'Solid' setting gives you the option of having 24 separate fill colors. These are shown to the right of the word **Colors,** going from right to left and continuing on the second line. By default, these colors are set to the six colors in the style sheet repeated four times.

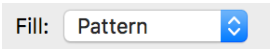


You can change the colors by using the color picker or using the color Presets (see the section on **Color** in the **Drawing Command Elements** chapter).

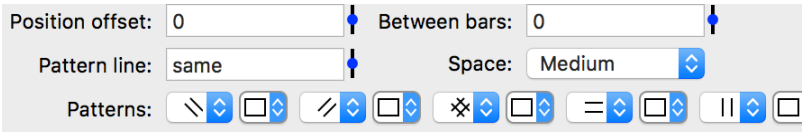
Note that the **Presets** to the right of the colors work for up to 9 data entries for the single-color schemes, up to 11 data entries for the double-color schemes, and up to 12 data entries for 'Paired' and 'Set#3'. Thus, when you have a large number of data entries, you will see a more limited number of options in the Presets as shown in the example below. There are currently no **Presets** for more then 12 data entries.



If you change the **Fill** setting to 'Pattern' the graph settings in the drawing command change to reflect the options available to you for that setting.



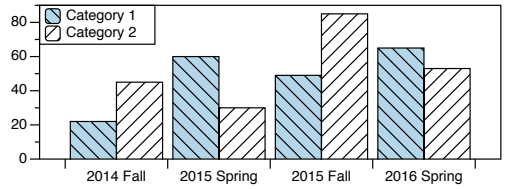
The screen shot below shows the settings that are now displayed. We no longer see the **Colors** settings, instead we now have the **Patterns** settings displayed.



This fill setting has six unique patterns that are preset. If you exceed six data entries they will repeat. You can set both the patterns and the background color separately. For each data entry, a setting for the background color is just to the right of the pattern setting.

To illustrate, the settings below show the pattern and background color for the first two data entries. By default, all of the background colors are set to white. In this example we have changed the background color for the first data entry to light blue.

This results in the following graphic.



Pie

The **Pie** command is used to draw a single pie or donut type of graphic using a single numerical column to control the size of each slice.

Pie command basics

The quickest way to create a pie chart is to first highlight the column you would like to graph in the table and click the **Pie** command shortcut, which is in the tool bar by default.

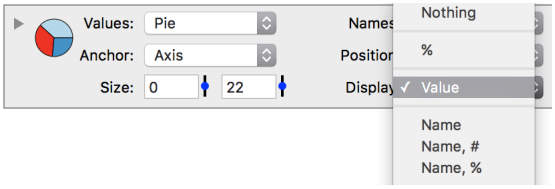
In the example below, we used the ‘Cost’ column.

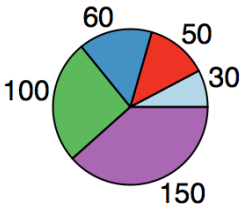
#	Item	ab	Cost
1	A		30
2	B		50
3	C		60
4	D		100
5	E		150

The following drawing command is created where the **Values** drop-down box is the set to the column 'Cost'.

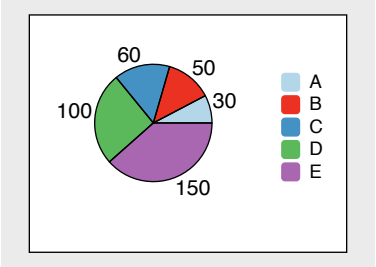
The drawing canvas now contains the following pie chart.

You can add labels to a pie chart using the **Display** drop-down box.



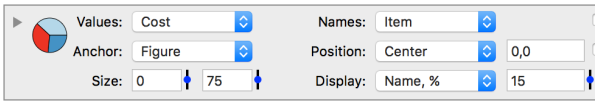


The **Name** drop-down box allows you to specify a corresponding text label for the data. This label can be displayed directly on the pie or by adding a **Legend** command, as shown below.



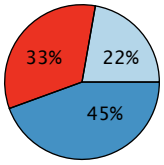
Percentages

The **Display** drop-down box can be used to automatically shown the % or numerical value for each category.



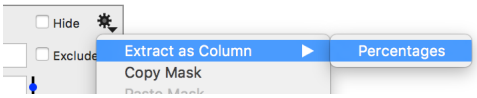
The percent is computed based on the sum of the values in the column, and rounded to get the total sum to be 100%. This is slightly different from standard rounding to the nearest integer.

For example, if you have the numbers 2,3,4, the percentages are 22.22…, 33.33…, and 44.44… If you round to the nearest integer you get 22%, 33%, 44% which sums up to 99%. Instead, the DataGraph will round up the largest fractions until the sum reaches 100%. That means that 44.44… is rounded up to 45, since 0.44… is larger than 0.22… and 0.33…



In the detail view of the command, you can adjust the number of digits after the decimal point via the **Percentages** drop-down box. This setting is located at the bottom of the Pie command.

If you want to use these percentages elsewhere you can extract them as a column back into the data table using the gear menu.

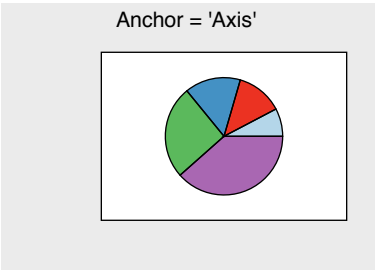


Positioning

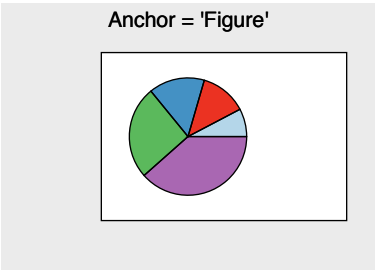
You can modify the position of the pie by clicking and dragging it around the canvas. You can also specify the location using the **Anchor** and **Position** settings.

The **Anchor** drop-down box has three options:

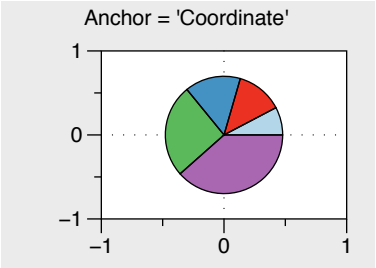
1. 'Axis' - The default option positions the pie relative to the axis box. The following graph shows the pie in the center of the axis box (the inner white region).



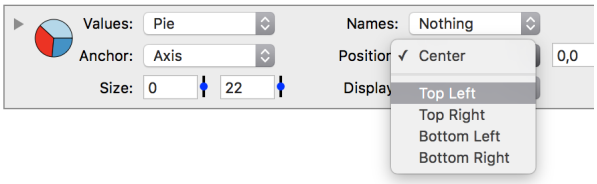
2. 'Figure' - Positions the pie relative to the entire drawing canvas, defined by the light grey region. Below the pie is in the center of the figure.



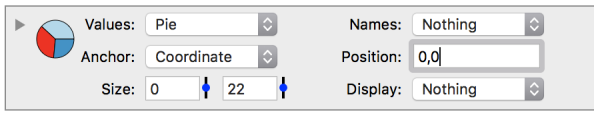
3. 'Coordinate' - Positions the pie based on an x and y location in the axis box. Below the pie is located at (0,0).



When the **Anchor** is set to 'Axis' or 'Figure', the **Position** setting allows you to specify an offset in pixels from either the center or one of the corners.



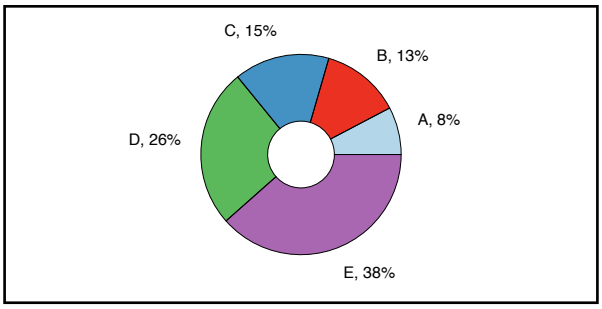
When the **Anchor** is set to 'Coordinate', you can specify the exact x and y location of the Pie. You no longer see a position drop-down box. Instead, you have a single entry box to the right of **Position** where you enter the location as an (x,y) coordinate pair.



Sizing

The size of the pie is determined based on two entry boxes in the drawing command. The first box sets the size of the inner circle such that a value greater than zero results in an annulus. The second box is added to the first to determine the overall size of the pie.

For example, the following graphic has an inner circle set to 25 and an outer circle set to 50. Thus, the total size is 75.



The pie chart is now what is sometimes referred to as a doughnut chart.

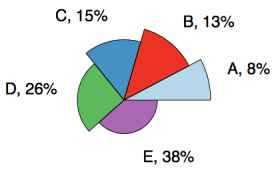
Custom Wedges

In the detail view of the **Pie** command, there are settings to control the size of the inner circle and the width of each wedge individually using columns in the data table.

For example, below we have set the **Width** drop-down box to a column, also called 'Width'.

#	Pie	Width	fx
1	30	65	
2	50	55	
3	60	45	
4	100	35	
5	150	25	

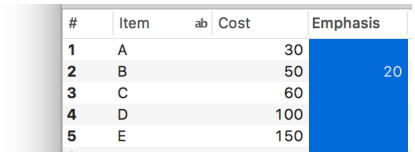
The width changes for each wedge as shown in the following pie chart.



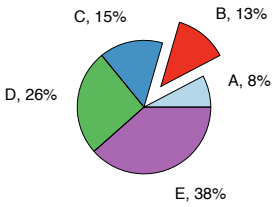
In this case, the **Inner** drop-down box is set to 'All the same' but we could also use a column to vary this length.

Adding Emphasis

You can change the offset for an individual wedge to add emphasis. Set the **Offset wedge** drop-down box to a column. In this example, we will use the column 'Emphasis'.



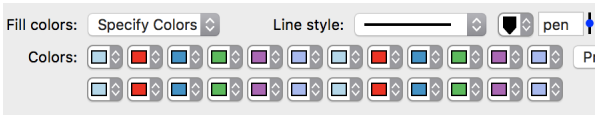
Now the data for 'B' is slightly pulled away from the center of the chart.



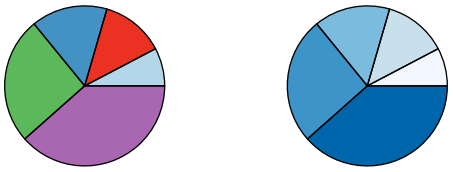
We could also animate the movement of 'B' by setting the value of '20' in the data table to a variable that is varied in an animation.

Specifying Colors

In the detail view of the command, you can control the colors of the pie chart.

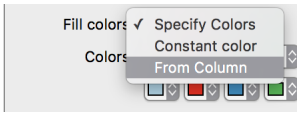


Similar to other commands, you can use the default colors, change individual color tiles, select new colors using **Presets**, or use **Color Schemes** or **Color Ramps.**



Default Color Scheme

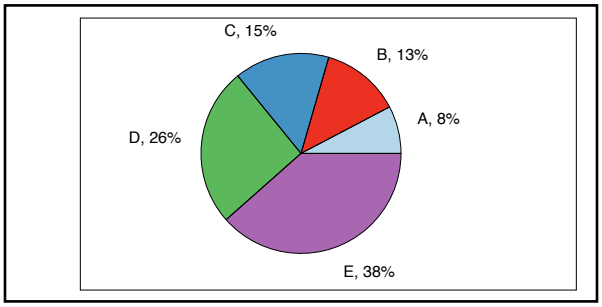
To change to a **Color Scheme** or **Color Ramp**, select 'From Column' from the **Fill colors** drop-down menu.



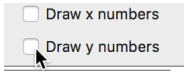
You can either create a new **Color scheme** or select an existing **Color scheme** or **Color ramp**.

Single Pie Chart

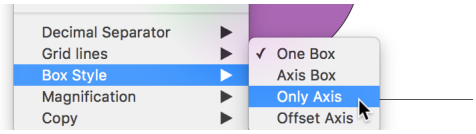
If you have a single pie chart, it is located in the center of the axis box by default.



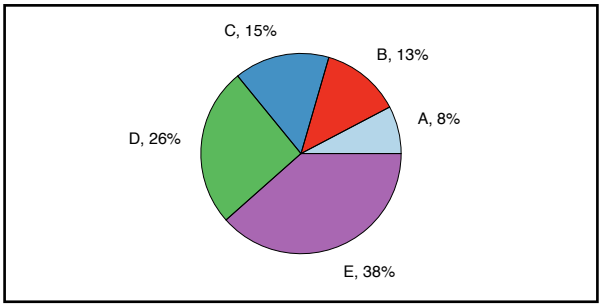
If you want to only show the single pie chart without the axis lines there are two steps. First, uncheck the **Draw x and y numbers** check box from the axis settings. The box is still drawn but it will no longer leave space to draw the axis labels.



Second, change the **Box style** to 'Only Axis'. You can access the **Box style** setting in the **Style** settings or by right-clicking on the graph.

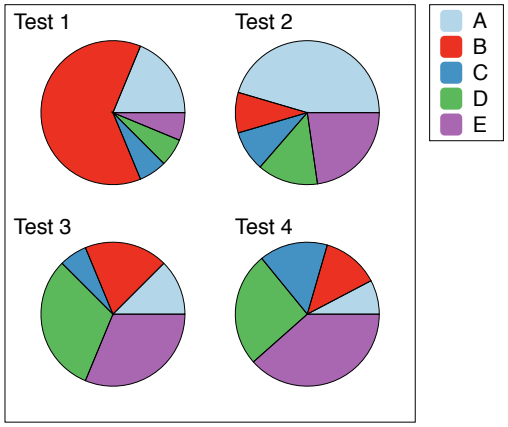


The pie chart is now the only thing drawn in the window.

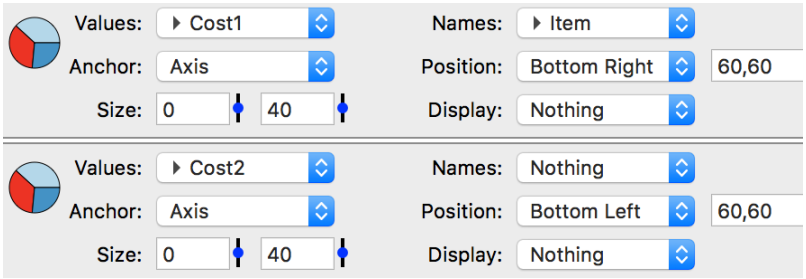


Multiple Pie Charts

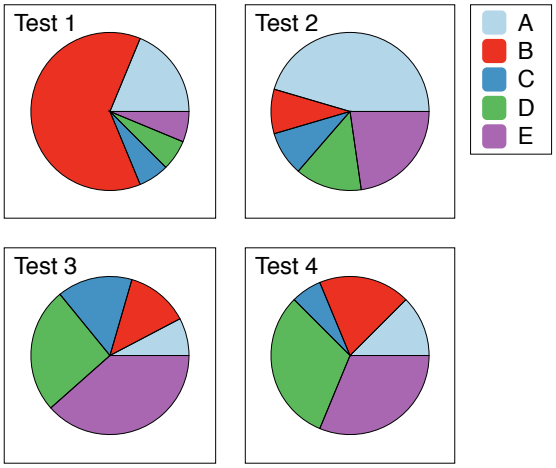
You can also create multiple pie charts. The figure below was created by first creating multiple pie charts and then dragging them into position.



Note that when you drag a pie chart on the drawing window, the position automatically updates to indicate the general location of the graphic. Thus, for each of the pie charts above, the **Position** was updated as the charts were dragging across the screen. The coordinate locations are relative to specific corners of the graph (e.g., Bottom Right, Bottom Left, …). To align the locations, these are all set to the same x,y values of 60,60.



You could also split the x and y-axis and have a single pie chart in each axis. Taking that approach, you need to set the **Anchor** drop-down box back to 'Axis'. Using Split axes, allows you to easily have individual boxes around each chart, as shown in the following example.



Stocks

The **Stocks** command can be used to visualize stocks using two different styles, Candlestick and OHLC (open high, low close). For more information on Candlestick charts see https://en.wikipedia.org/wiki/Candlestick_chart.

Copy and paste the data, or import from a file. A typical data set will have columns similar to the following example.

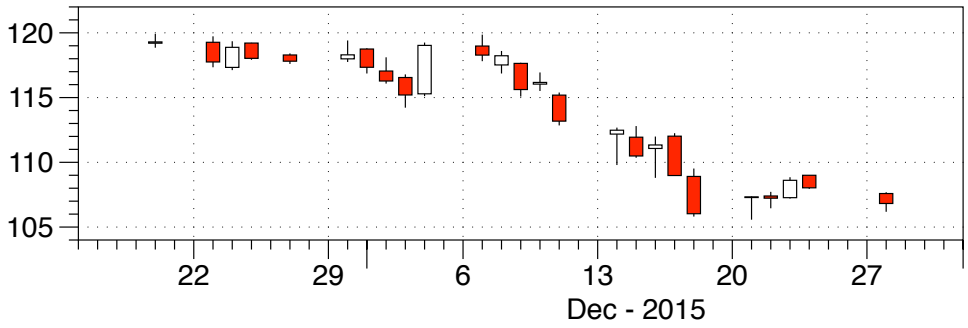
Date	Open	High	Low	Close	Volume
Dec. 28, 2015	107.59	107.69	106.18	106.82	26.64M
Dec. 24, 2015	109.00	109.00	107.95	108.03	13.60M
Dec. 23, 2015	107.27	108.85	107.20	108.61	32.61M

When the Date and Volume are not understood as numerical columns, these can be converted using the gear menu. Since the data is in one of the standard date formats, DataGraph will suggest the right format for the conversion.

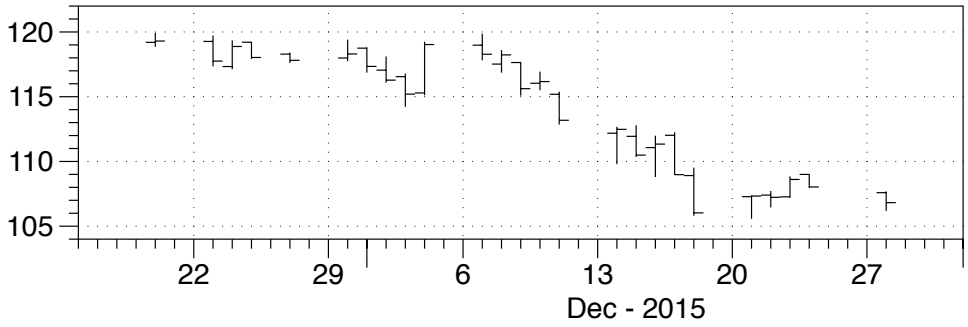
Although the Volume is technically not needed for the Candlestick graph, you may want to use this in another graph. To convert Volume to a numerical value is a little bit trickier since 'M' is used to denote the volume in millions. You can convert this to a numerical column, but it will not be understood immediately because of the M. You can use find and replace to either remove the M or replace it with *10^6 or *1E6. One option is to define 'M' as a variable as 10^6. This works because DataGraph interprets 26.64M as 26.64*M.

Once the data is converted, create a Stocks command using the Command menu. Select the columns from the table as shown below.

Below is an example using the Candlestick option.



You can also choose the OHLC option.



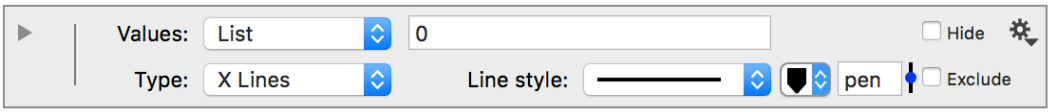
For the Candlestick, you can change the fill colors. By default the width of the marker is one day. If the Date column is a date column, this is the proper width, but if the Date column is a list of days (e.g., 1,2,3,…) the width will be much too wide, since Day is 24*3600 seconds. In that case switch the marker width to either specified (coordinate) or pixels (graphical).

Lines

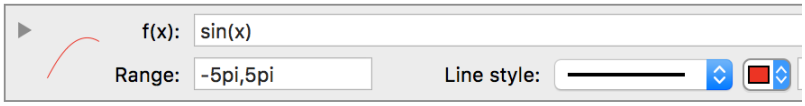
The **Lines** command allows you to draw horizontal or vertical lines.

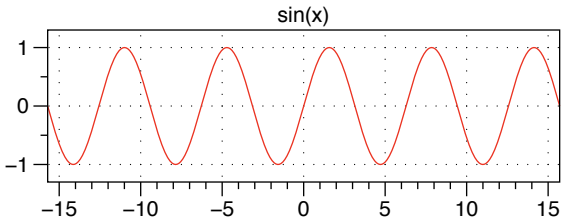
Lines Command Basics

When you create the command the default is to draw a vertical line at x=0, extending the full range of the y-axis. If you switch the type to 'Y lines' you get a horizontal line along the x-axis. One example where you might want to use this is to indicate the location of the x=0 or y=0 when the axis range includes both positive and negative values.

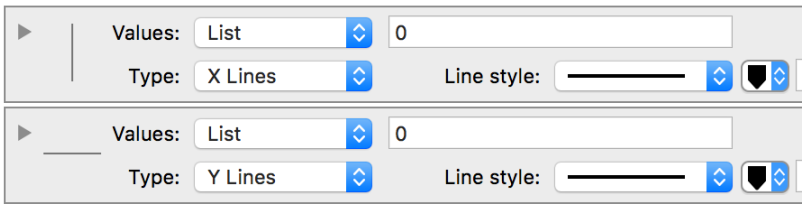


For example, if you plotted the function sin(x) you would get the flowing graph.

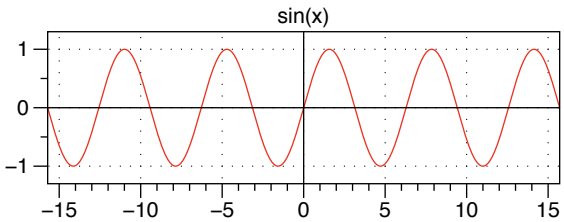




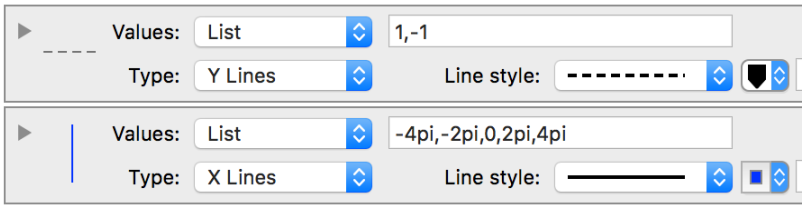
You can easy add x and y-axis lines with two **Lines** commands. The first **Lines** command has the default settings. The second **Lines** command has the **Type** changed to "Y Lines".

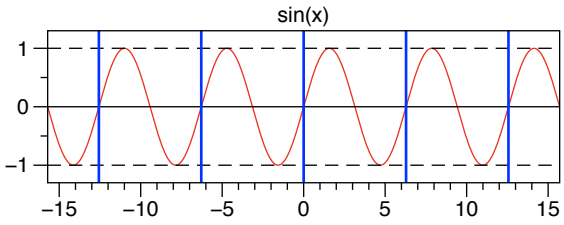


Now you get the following graph.



You can also type in a comma separated list of numbers in the field, for example 0,e,pi. Note that this list can be an expression such as (0,1,2,3,4)/2, etc. In our example, we addd two additional 'Y lines' and 'X lines' at every 2pi increment.



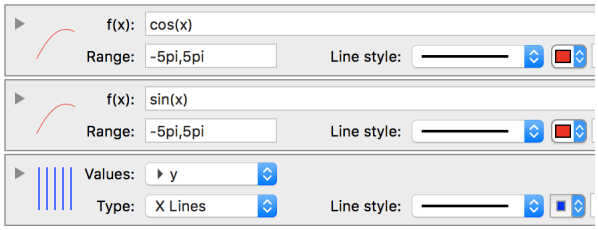


You can also include variables. For example, if the variable t is the current time and the value is ‘t’ you get is a vertical line that changes as you vary the time.

If you want to draw a large number of lines you select a column instead. In that case, you can also specify the lower and upper ranges for the lines, either as a constant or separate column. For example, for x lines you can set x lines to have a lower value of 0 and upper value to come from a column ‘y’. This will create a collection of thin bars that start at (x(i),0) and end at (x(i),y(i)).

Detail View

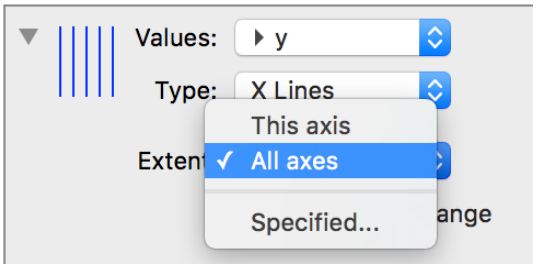
In the next example, we will draw two functions and y lines and customize the **Lines** command using settings in the detail view.



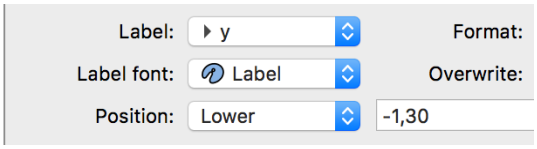
We specified the lines from the following 'y' column using the **Values** drop-down menu (where opt-p gives us the symbol for pi).

#	y
1	-4π
2	-2π
3	0
4	2π
5	4π

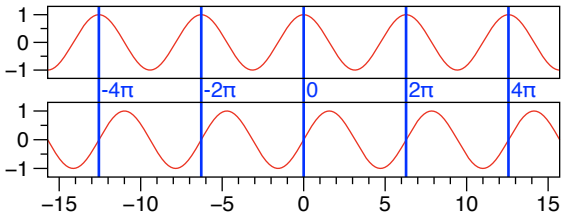
Note that we split the Y-axis to draw to each line separately. To have our lines cross both axes, we changed the **Extent** of the lines to cross 'All axes'.



Lastly, we specified the 'y' column to be used as a **Label** for each line and we also specified the **Label** color.

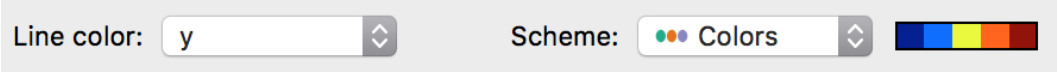


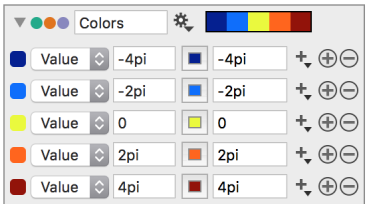
This results in the following graph with multiple x-lines and a label that comes from a column.



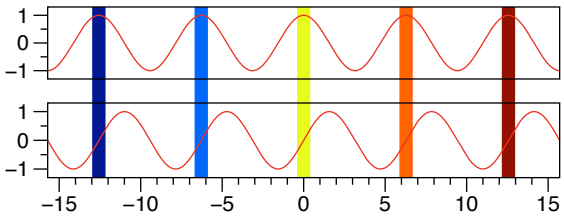
Color schemes

In the detail view, the **Line color** drop-down menu can be set to a color scheme.

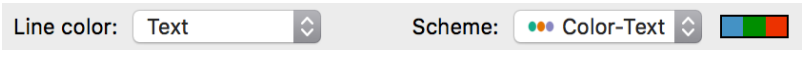




If we apply the above color scheme to the graph from the previous section, we get the following graph.

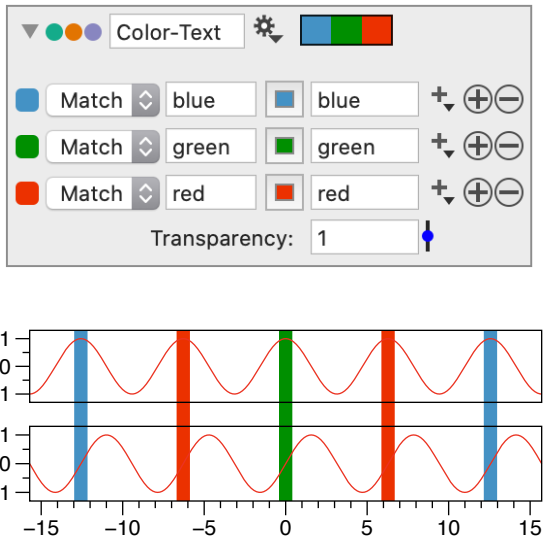


Color schemes can also be based on a text field. In this example, we set the color scheme based on the column called 'Text'.



#	y	Text ab
1	-4π	blue
2	-2π	red
3	0	green
4	2π	red
5	4π	blue
6		

The color scheme is set to match the colors based on the text.



See the Variables Chapter for more information on creating color schemes.

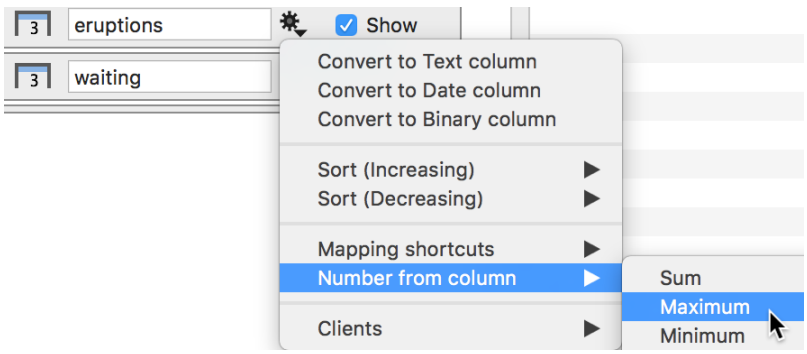
Example: Using Variables

The location of the lines can also be set using variables.

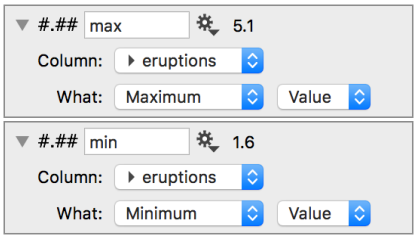
Let's take the Old Faithful data set where we have a column with the time for a number of eruptions, with a total of 272 rows entries.

#	eruptions
1	3.6
2	1.8
3	3.333
4	2.283
5	4.533
6	2.883

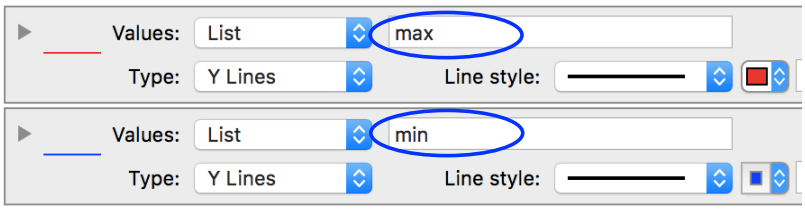
We can create variables for the maximum and minimum value using the **Number from column** shortcut in the gear menu of the eruptions entry in the column list.



We will call these variables max and min, which look as follows in the variables list.

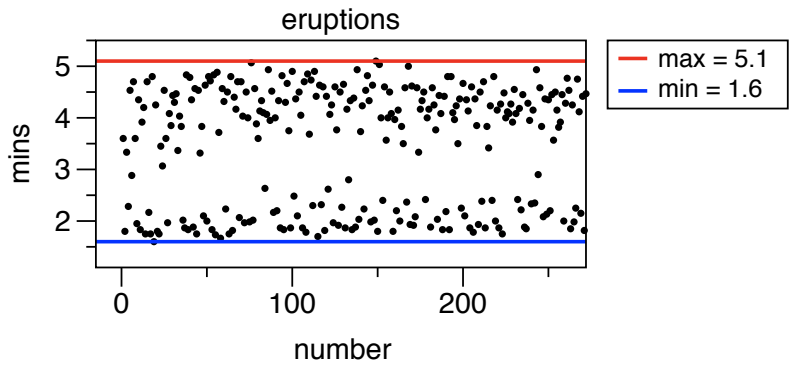


We can use these variables 'max' and 'min' as the **Values** in two **Line** commands with the **Type** set to 'Y Lines'. Simply type the name of the variables in each of the **Values** list.



Finally, we will also set the legend in the detail of the **Lines** commands to use a token. For more on using tokens, see the **Tokens** section in the chapter on **Drawing Command Elements**.

Along with a **Points** command and a **Legend** command we get the following graphic, where the values in the legend are dynamically linked to the data.



Connect

The **Connect** command is a basic drawing command, similar to the Lines command. There are three parts to the graphic that gets produced:

1. A starting point (x1,y1) and an ending point (x2,y2)

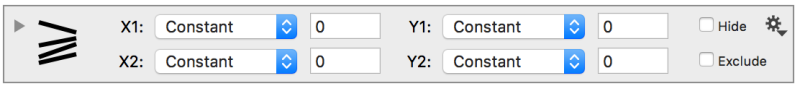
2. A line that connects the points and an optional symbol at the start and end

3. A label (also optional)

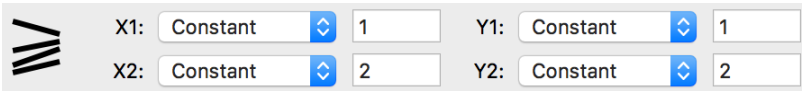
When one or more of the x1,y1,x2,y2 values is given by a column, you create multiple lines.

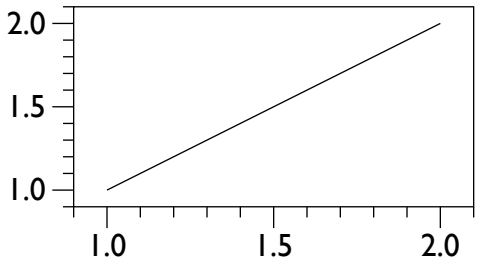
Connect Basics

When you first create a **Connect** command, nothing is drawn as the starting x1,y1 and x2,x2 pairs are all at 0,0. You can change any of these values and a line will be drawn between the points.



You can change these values in order to draw a simple line. Here we draw a line from (1,1) to (2,2).



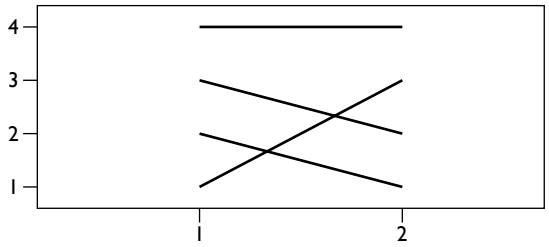


You can use the drop-down box to specify a column as input for any of the values. You can also leave some as constants and use a column for others. For example, consider the following y1 and y2 values where we want our lines to show the change between.

	y1	y2
1	2	1
2	3	2
3	1	3
4	4	4

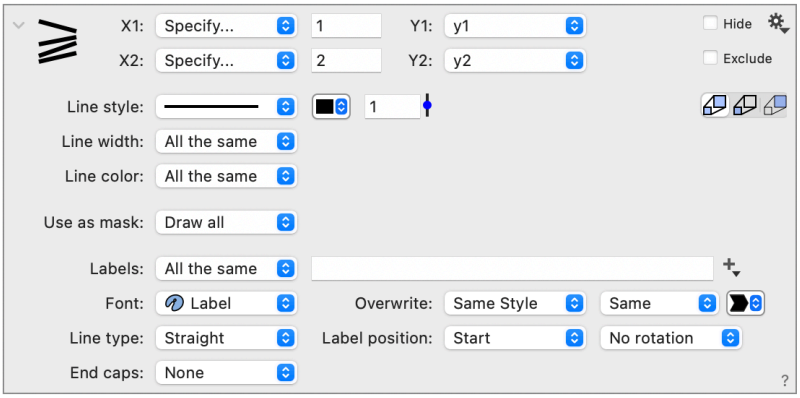
The drawing command would be set as follows to use the data in the columns.

Now we have four lines drawn.



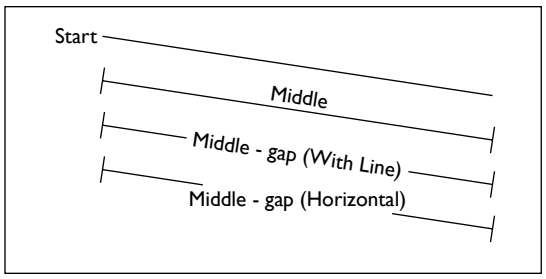
Custom Options

Opening the disclosure triangle reveals additional options such as line style, line color, adding labels, and line type.



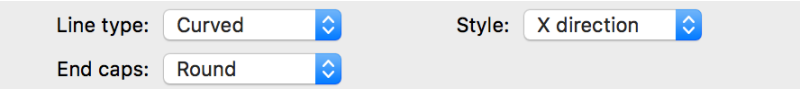
Label Options

The label can be a constant for all lines or specified with a column. For curved and step lines, the label is at the start of the line, and horizontal. For the straight line there are other options, namely that the label can be in the middle of the line and be rotated with the line or horizontal.

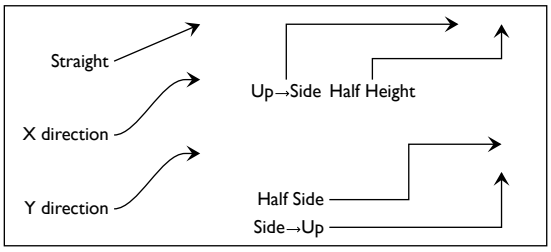


Line Types

By default, the line is drawn straight between the x,y coordinates. The **Line type** menu gives you a lot of options for how the points are connected. Choose from 'Steps, 'Curved', or 'Straight'.



To the right of the **Line type** menu other options will be available that depend on the type chosen. For example, curved lines have a Style menu for changing the direction of the curve (i.e., X or Y). Stepped lines have several options for how the step is oriented. These options are illustrated below.

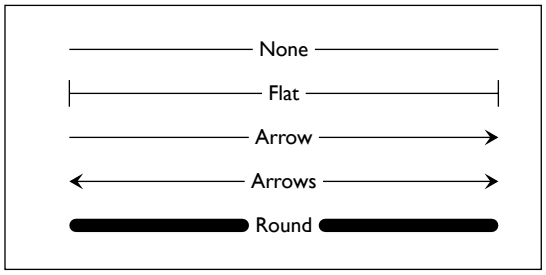


Specifying end caps

There are several ways to draw the end caps. Each is listed below and demonstrated in the following figure.

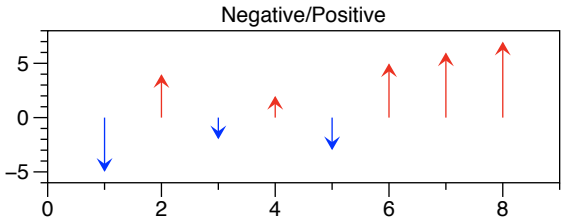
1. None — Straight line, with sharp ends.
2. Flat — End caps at the beginning and end, perpendicular to the line. Useful when you are indicating length.
3. Arrow — An arrowhead at the (x2,y2) point.
4. Arrows — Arrowheads at both the start and the end.

5. Round — Rounded edge, this is only visible once the line width is a few pixels.



Using Color Schemes

The example graph shown below uses the **Connect** command to create vertical lines where the color and location of the arrows is determined by the direction of the data. This requires the use of a **Color Scheme** variable, as described in the **Variables** Chapter.



Pivot

The **Pivot** command is a powerful tool for displaying and analyzing data. This command can be used to create bar graphs, while also binning data according to categories, number ranges, or dates.

Pivot is short for Pivot Table, and is a standard data analysis technique (see wikipedia). **Pivot** requires one to three input columns. One or two columns are used to specify categories and an optional column can be used to specify values.

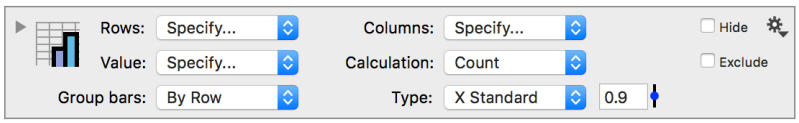
At its core, this command allows you to group data based on one or two categories and calculate a number of summary statistics based on the data (e.g., mean, standard deviation). These calculated values can be extracted out of the **Pivot** command to use in other drawing commands.

This section begins with some very basic examples to illustrate how pivoting is done in DataGraph; however, note that the **Pivot** command is a powerful tool for analyzing large sets of data with categorical information.

Pivot Command Basics

You can create a **Pivot** command using the short cut on the toolbar or by selecting the **Command < Add Pivot** in the menu or using one of the toolbar shortcuts.

The following command will be created where you can specify the **Rows** and **Columns** in the top line of the command. In the second line, you specify the **Value** used in the table and the **Calculation** you want to display.



Example 1: One or two columns of data

In this first example, we are only going to change the **Rows** and **Value** drop-down boxes. Technically, neither of these will 'pivot' the data but we will be able to easily calculate summary statistics using the **Pivot** command.

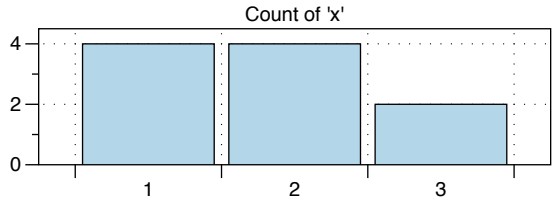
The simplest **Pivot** command you can create only has one setting. Specifically, you can create a graph by setting the **Rows** drop-down box to a column in the data table to count the unique entries in that column.

For example, let us consider the following data where we have an ‘x’ and ‘y’ column.

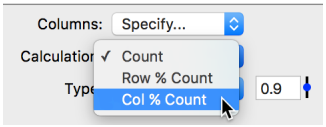
#	x	y
1	1	1
2	2	2
3	1	4
4	2	5
5	1	6
6	2	1
7	1	3
8	2	5
9	3	8
10	3	7

We are going to set **Rows** to the 'x' column. You can use the drop-down box to specify it manually or you can highlight the 'x' column in the data table and then hit the **Pivot** command short-cut. Either approach will create the following command.

A bar graph is created showing the count for unique entries of 'x'. As illustrated in the graph, the numbers 1 and 2 both appear in the 'x' column 4 times and the number 3 appears in the column 2 times.

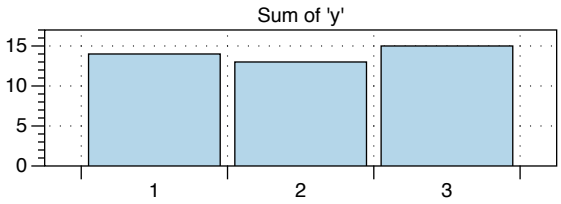


You can also select to display the results as a percentage of the total.

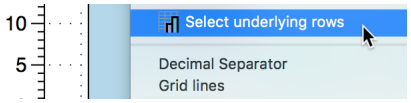


Next, let's modify the Pivot command to specify the Value drop-down box as 'y'. The **Calculation** drop-down box is automatically updated to provide the 'Sum'.

As a result, our graph now displays the sum of y for each unique entry in x.



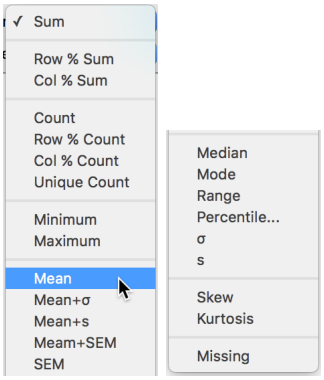
To see the exact data used in the calculation, you can right-click on any of the bars in the graph itself and choose **Select underlying rows**.



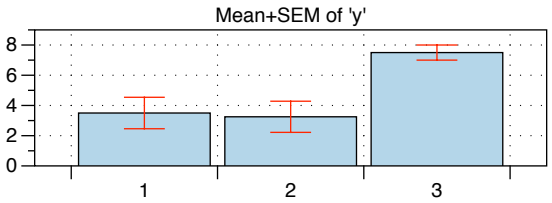
In the following example, **Select underlying rows** was chosen for the bar where x=1. In the data table, the data is now highlighted where x=1. Thus, the value of 14 comes from summing up the values of y where x=1 (i.e., 1+4+6+3 = 14).

#	x	y
1	1	1
2	2	2
3	1	4
4	2	5
5	1	6
6	2	1
7	1	3
8	2	5
9	3	8
10	3	7

In addition to calculating a sum, a long list of options is available under the **Calculation** drop-down box shown below.



One convenient option is to select both a Mean value and a measure of the deviation around the mean. In this case, the Pivot command automatically provides the errors bars around the mean, as shown in the example below.

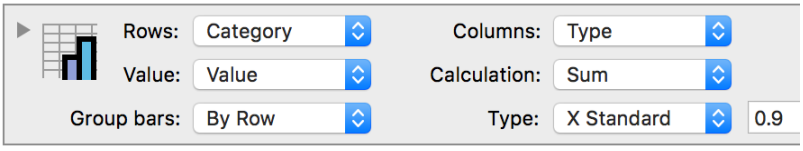


Example 2: Three columns of data

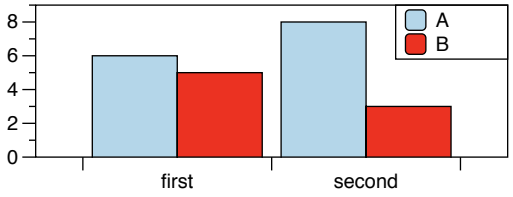
To explain this further, let's walk through another basic example. In the table below, there are two text columns, 'Category' and 'Type', and one numerical column, 'Value'.

Category ab	Type ab	Value
first	A	4
first	B	2
second	A	4
first	B	2
first	B	1
second	A	4
second	B	3
first	A	2

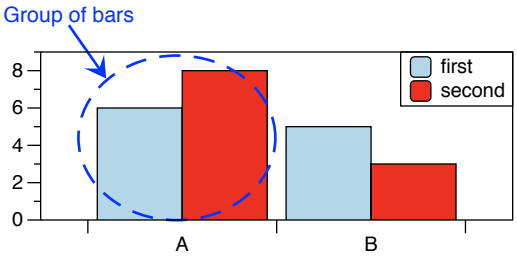
If you select all three columns in the data table and then click the Pivot command short-cut, the following drawing command is created with the **Rows**, **Columns**, and **Value** fields filled automatically. Since the **Value** drop-down box is specified, the **Calculation** is set to 'Sum'.



The above command creates the following graphic where each bar represents the sum of the values grouped by both the 'Category' and 'Type'. Note we also added a **Legend** command.

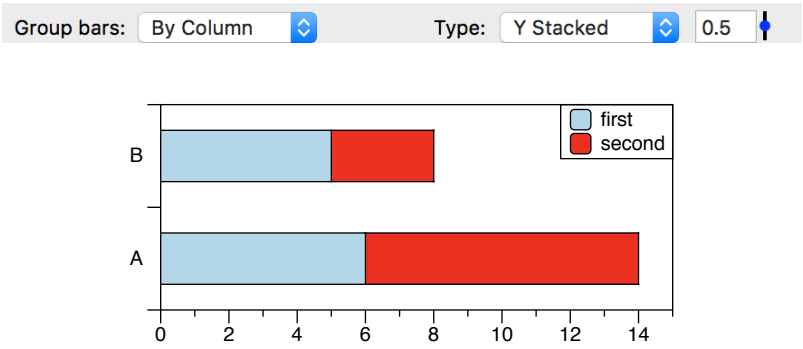


By default, the **Group bars** drop-down box is set to 'By Row'. If you change this to 'By Column' the data is displayed differently but the sum for each combination of 'Category' and 'Type' is the same. Basically, changing this command swaps what is displayed on the x-axis with what is displayed in the legend.



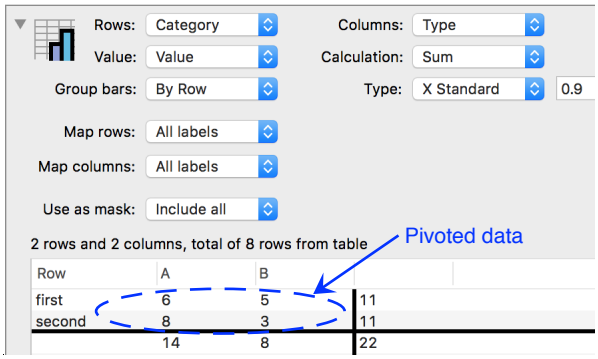
The last line of the drawing command main view allows you easily modify the look of the bar chart. In addition to the **Group bars** setting, you can change the **Type** from 'Standard' to 'Stacked' or 'Area', oriented on the x or y-axis. Just to the right of the **Type** drop-down box is a toggle button for adjusting the width of the bars.

These settings were adjusted for the previous graph as shown below.



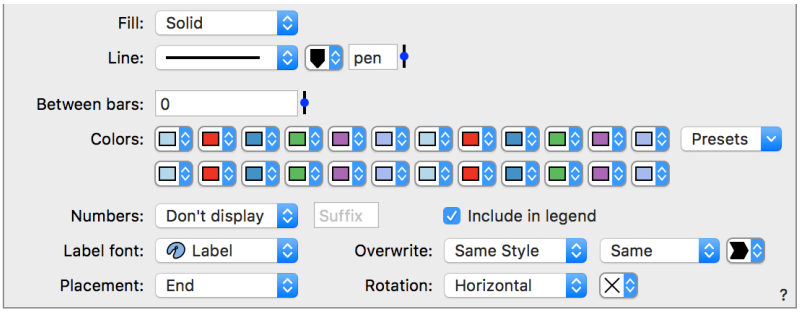
Additional settings

You can open the disclosure triangle to reveal the detailed settings for the **Pivot** drawing command. Not only does this give you additional options for modifying the display of the graph, but you will also see a table view of the data being graphed. The values are displayed in a small table inside the drawing command.



For several types of calculations, the value is given across and down the table as is shown in the previous screen shot. These include: sum, count, min, max, range, and missing.

Below the pivoted data, you can vary the **Fill**, **Line**, **Labels**, etc. Note that the **Fill** options change depending on the selection in the Fill drop down box in the same way as in the **Bars** command. For more information, see **More Fill Option**s in the **Bars** command section of this chapter.



Mapping and Grouping Data

The detailed view of the **Pivot** command has two drop-down boxes called **Map rows** and **Map Columns** that allow you to further group and map data. The menu options for these drop-down boxes vary depending on the type of data specified in the **Rows** and **Columns** drop-down boxes. Specifically, these vary for text columns, numeric columns and date columns. Each of these is explained in more detail in the following subsections.

Text columns

Below are the options shown in the **Map** drop-down box when the data we are pivoting is text. The default setting is 'All labels', meaning that all of the data in the pivoted data is included in the resulting bar graph.

Using the 'Specified' option you can:

1. Change the order of entries
2. Limit or add entries

In addition, using the Columns option you can:

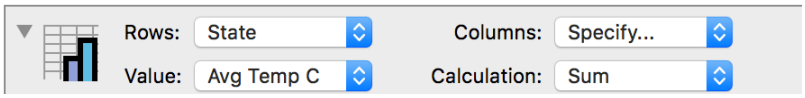
3. Replace an entry with a label

4. Group data based on the new label

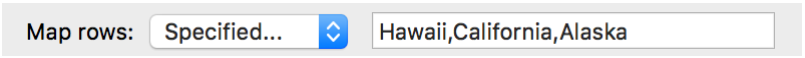
If you change **Map rows/columns** to 'Specified' an entry bar appears where you can manually enter the data groups you want displayed. To illustrate how this works consider a data set with the average annual temperature of the 50 US states. The first few rows are shown below.

State	ab	Avg Temp C
Alabama		17.1
Alaska		-3.0
Arizona		15.7
Arkansas		15.8
California		15.2

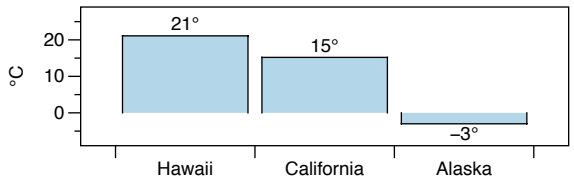
Create a **Pivot** command with the **Rows** set to 'State' and the **Value** set to 'Avg Temp C'. This will create a bar graph with every state included.



Next, change **Map rows** to 'Specified' and enter the states of interest, as shown below. Note that spaces are included in the search so do not put a space between entries unless it is part of the data.



Map rows will limit the data to the specified entries and create the following bar graph. Essentially, this is working similar to a mask, where we are limiting the data in the graph.

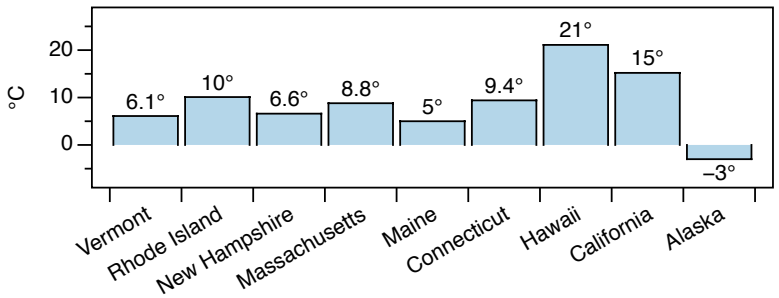


If we have a long list of categories we would like to include the 'Specified' option will not work well. For example, say we are also interested in the New England states. You could create a column that contains the states of interest

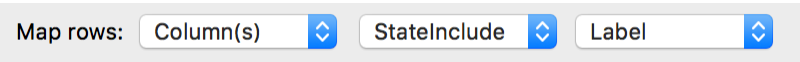
StateInclude	ab	Label
Vermont		New \n England
Rhode Island		New \n England
New Hampshire		New \n England
Massachusetts		New \n England
Maine		New \n England
Connecticut		New \n England
Hawaii		Hawaii
California		California
Alaska		Alaska

First, change the **Map rows** to 'Column(s)'. Next, select 'StateInclude' as the column we are using to map our data.For now leave the last drop-down box as 'Same'.

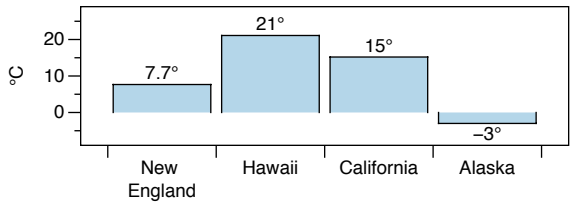
This creates a figure with all of the states listed in 'StateInclude'. Note the axis settings were modified to angle the x-tick mark labels.



To group all the New England states together, change the third drop-down box to the right of **Map rows** to the 'Label' column.

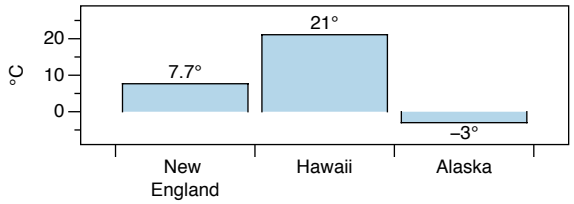


To calculate the average across all the New England states, you also have to change the **Calculation** drop-down box to 'Mean'. This results in the following graph where all the New England states are now grouped. Note how the '\n' in the label for the New England states creates a line break such that we do not have to angle the tick marks labels.



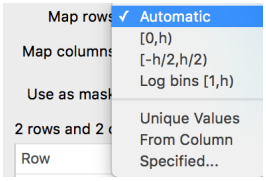
If we decided to remove any of the entries we could delete them from the either ‘State Include’ or ‘Label’ columns. For example, if you decide you no longer wanted to include California you could delete the associated entry in ‘Label’.

This removes California from our graph as shown below.



Numeric columns

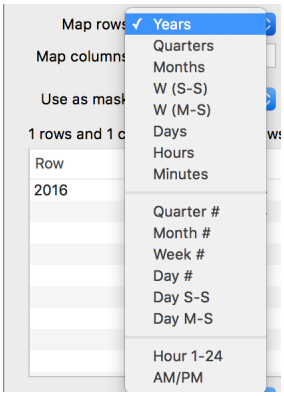
Below are the options shown in the **Map rows/columns** drop-down box when the column or row data specified in the **Pivot** command is numeric.



The default setting is ‘Automatic’. In this setting, DataGraph will bin the numeric data based on the input. If you want to change how the data are binned several options are available. You can also specify numeric values to include using the ‘Specified’ or ‘From Column’ options, which are similar to those described in the previous section for text columns.

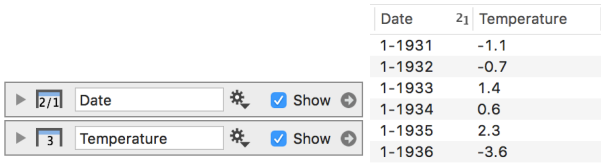
Date columns

Below are the options shown in the **Map rows/columns** drop-down box when the column or row data specified in the **Pivot** command is a date column.

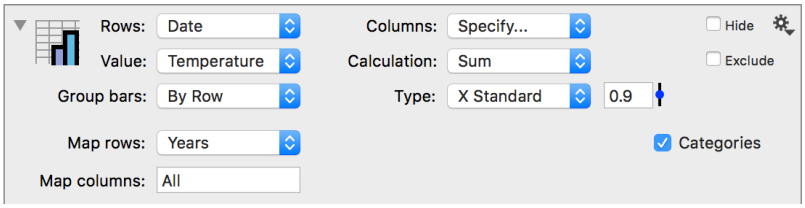


In this case, you can group the data over time by years, quarters, months, etc., or you can group the data by a number representing quarters, months, weeks, and days.

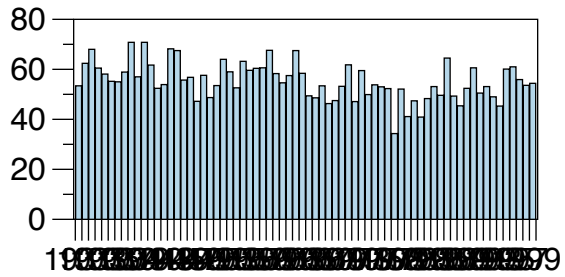
For example, let's consider a temperature data set where we have the average monthly temperatures over several decades. For this example, monthly average temperature data was downloaded for Reykjavik, Iceland from 1931-2000. The data are in two columns, ‘Date’ and ‘Temperature’. The date column is in the format month-year, as shown below for a subset of the data.



To analyze this data using the **Pivot** command, create a new blank graph, highlight the ‘Date’ and ‘Temperature' columns in the data table, and click the **Pivot** command shortcut. The following drawing command will be created.

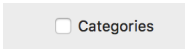


The default settings produce the following graph where the rows are being mapped to ‘Years’ and **Calculation** is set to the ‘Sum’ for each year.

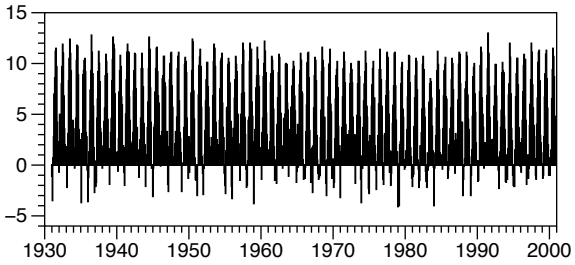


Change **Map rows** to 'Months', such that the data for each month is be displayed.

Also, uncheck the **Categories** checkbox. When this box is checked, each bar is considered its own category and every label is shown.



Now the x-axis shows the years using tick-marks and the data is for each month.

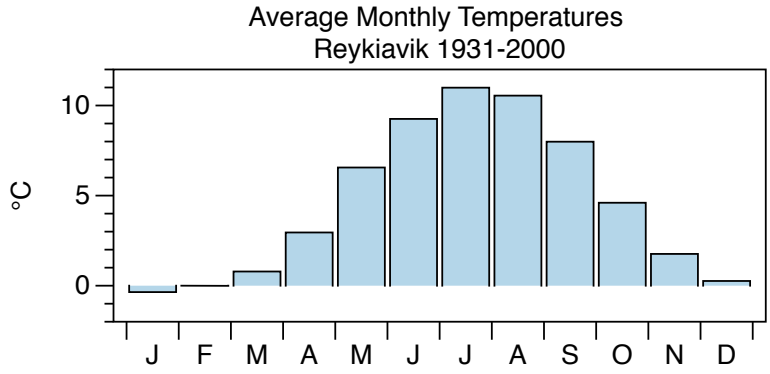


Next, change the calculation to the 'Mean'.

If you change **Map rows** to 'Month #' you can get the average for each month over the entire data set, where January=1, February=2, etc.

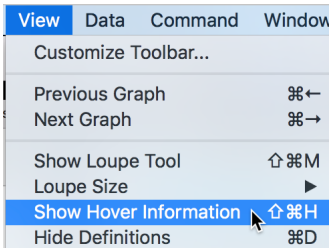
To the right of the **Map rows** dropdown box, there are built-in text labels for the months. Note you must also re-check the **Categories** check-box to use these labels.

These changes result in the following graphic where we have also added titles to describe our data.

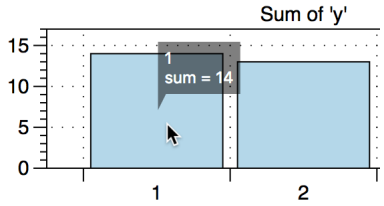


Using the Hover Tool

The Hover tool can be very useful for data exploration, particularly for complex data sets. Turn on the Hover tool by typing shift-command-H or via the View menu.



Now when you move the cursor over a bar, it will display the value in the bar as shown below for x=1.



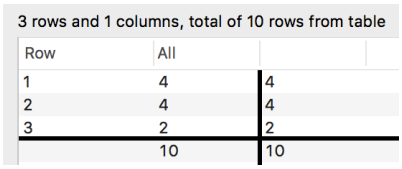
Extracting data

You can extract data from the pivoted table to the main data table. When you click on the gear menu on the top right corner of the command, you will see a list of options for what you can extract. You can also right click on the pivoted data.

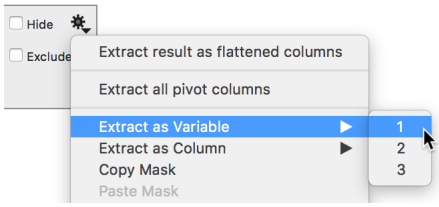
The extracted columns are dynamic, so when the input data changes, the pivot columns change automatically. You can then use these columns as input for any other drawing commands.

Example 1: One column of data

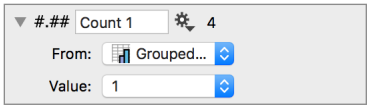
Let's consider the example from the beginning of the section on pivoting where we had one column of data being counted. The resulting calculations in the **Pivot** command are as follows:



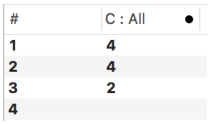
When you have only one column of data specified (as in this example), you can extract individual numbers using the gear menu by selecting **Extract as Variable**, or the entire column by selecting E**xtract as Column**.



Extract as Variable will create a variable. In this case the variable '1' was extracted, which has a value of 4.



Extract as Column will create a new column variable in the data table containing all the data.



Example 2: Three columns of data

Next, let's consider the example where we had three columns of data being counted. The resulting calculations in the **Pivot** command are as follows:

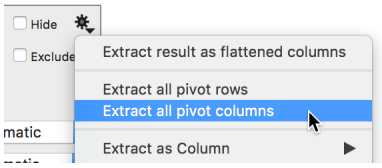
2 rows and 2 columns, total of 8 rows from table

Row	A	B	
first	6	5	11
second	8	3	11
	14	8	22

When you click the gear menu, you get the following options:

- Extract result as flattened columns — extracts every entry in the table as an individual row,
- Extract all pivot rows or columns — this also extracts all the pivoted data, either as shown or the transpose of the data, and
- Extract as Column — reveals a list of individual columns that can be extracted.

To illustrate, we will select **Extract all pivot columns.**



The following columns are added to the data table, which are identical to the pivoted data, shown above.

#	Row Labels	A	B
1	first	6	5
2	second	8	3
3			

For comparison, if we selected **Extract result as flattened columns** we would get the following data where the column headers are now row identifiers. All the numerical data is in the 'Flat Value' column.

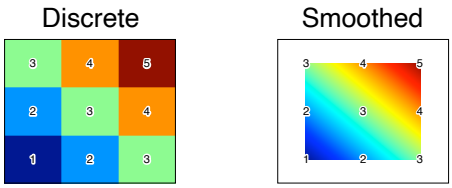
Flat Row	Flat Column	Flat Value
first	A	6
second	A	8
first	B	5
second	B	3

The format you choose to extract data depends on how you intend to use the data. In general, the flattened format often is more flexible as we can continue to sort on either categorical identifier (Row or Column). You can even use your extracted data from one **Pivot** command as an input to another **Pivot** command.

Scalar Field

A scalar field is a representation of a 2-D array of numbers, where colors are used to represent the numerical values, sometimes referred to as a heat map.

The values can be represented using a single, discrete color over a grid or as a continuous, smoothed surface (i.e, 2D interpolation).



Scalar field command basics

Select **Command > Add Scalar Field** to add the scalar field command.

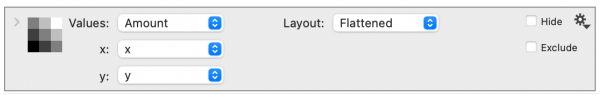
The command has three menu selectors:

- Values — must be numeric (number or date column).
- X — can be text, number or date.
- Y — can be text, number or date.

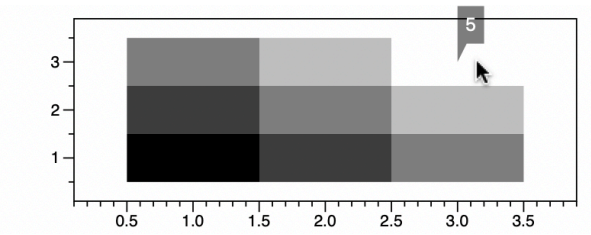
Here are three input columns, where color in the scalar field will be based on ‘Amount’.

x	y	Amount
1	1	1
2	1	2
3	1	3
1	2	2
2	2	3
3	2	4
1	3	3
2	3	4
3	3	5

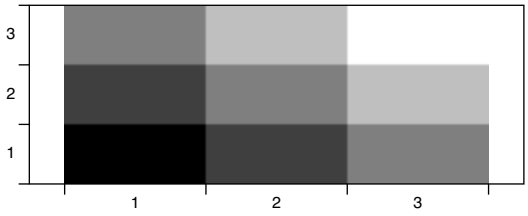
Select the data as shown.



This creates a simple scalar field, where the color of each square is based on the value of ‘Amount’. Toggle on the hover (**Shift-⌘-H**) to view the corresponding value for each square.



If columns x and y are converted to text columns the axes are formatted as categorical.

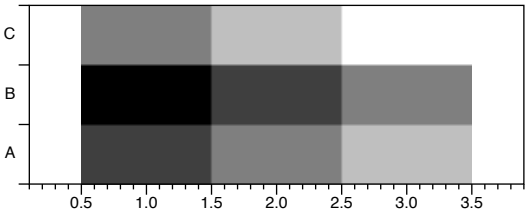


You can combine text and numeric axes. When the entries are alphanumeric characters, they are sorted alphabetically.

For example, here the y data is changed a text column ‘Group’.

x	Group	Amount
1	B	1
2	B	2
3	B	3
1	A	2
2	A	3
3	A	4
1	C	3
2	C	4
3	C	5

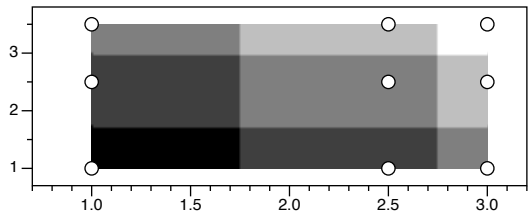
Although the ‘Amount' column did not change, the resulting graph is different because the y direction is sorted by the alphanumeric characters (A,B,C).



The distance between x-y locations can vary resulting in non-uniform grids. In this example, every x or y value that was 2, has been changed to 2.5, and 3 in the was changed to 3.5.

x	y	Amount
1	1	1
2.5	1	2
3	1	3
1	2.5	2
2.5	2.5	3
3	2.5	4
1	3.5	3
2.5	3.5	4
3	3.5	5

This results in a non-uniform grid. In the graph below, the grid locations are shown by plotting the x and y points with a Points command.



Note how the edge of each shaded cell is now the mid-point between the nearest grid locations.

Layout Menu

By default, **Layout** menu is set to 'Flattened', where the grid locations are specified using an input column for x and y, as in the prior examples. The values of x and y do not have to be in any particular order, but the program expects a rectilinear grid, oriented along x and y.

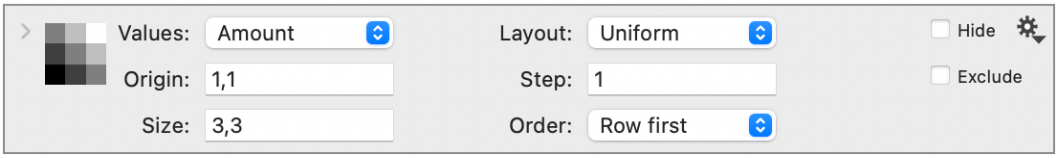
Two other **Layout** menu options are available that allow the x and y location to be specified via settings in the command, instead of in the data table.

Uniform

By setting the **Layout** = 'Uniform', you can specify the grid locations without entering an x or y column. This option only works for grids with a uniform step size between points. Here are the input parameters:

- **Origin** — starting x,y location
- **Step** — distance between points
- **Size** — number of points (x,y)
- **Order** — starts with row or column (across x or up y)

For example, the following settings would work for a 3 by 3 grid, starting at (1,1).



Non-uniform

By setting the **Layout** = 'Non-uniform', the x and y locations are specified in a column but you don't have to provide the x and y locations for every row.

This setting has two types. By default, **Type** = 'Two lists', where you must input the unique x and y values.

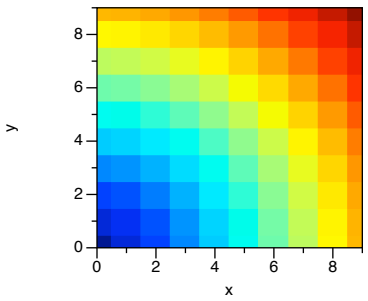
Technically, this setting could be used for Uniform grids also.

x	y
1	1
2	2
3	3

Specifying x and y locations with uneven distances between points will result in non-uniform grids.

x	y
1	1
2.5	2.5
3	3

By setting **Layout** = 'Arrays', you can draw scalar fields that are not aligned with the x and y-axis. To illustrate, first let's look at an example that is aligned.



In this case, x and y are in an ordered list similar to the flattened format, but here the list **must be ordered** and the dimensions of the grid have to be specified. In this example, the grid is (n,n) and where n is a variable =10.

The color is based on column 'd' which is calculated as follows:

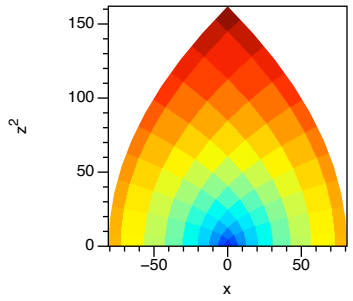
$$d = norm(x, y)$$

The x and y locations can then be mapped to other grid locations to form other shapes (Conformal mapping). In this case the values u and v are mapped using x and y:

$$u = x^2 - y^2$$

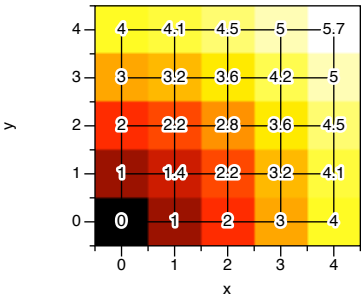
$$v = 2xy$$

The resulting scalar field is still rectilinear, but it is no longer aligned with the x and y axes.

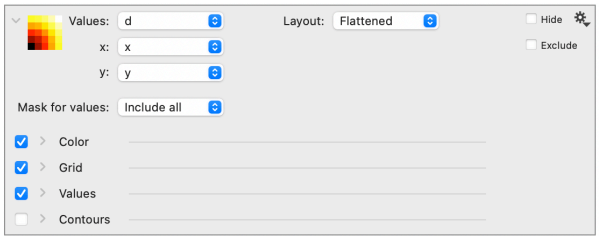


Color, Grid, and Values

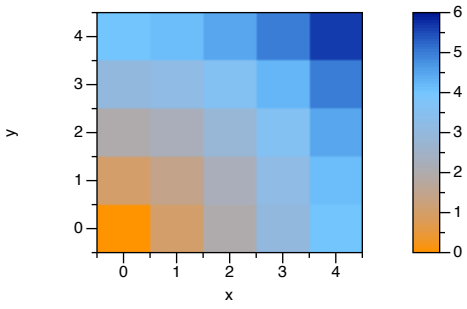
The Scalar Field command also has several built-in color ramps, the option of showing a grid, and the ability to display a label for each value.



Each of these overlays can be toggled on and off using check boxes. Expand each section for formatting options.

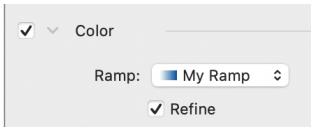


Use your own custom Color Ramp variables and a Color Ramp Legend to display on the graphic.

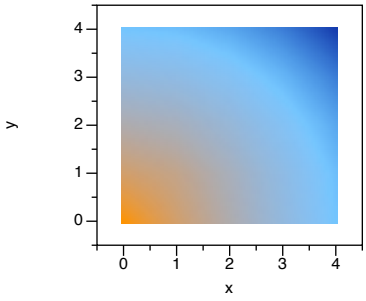


Refine

In the Color section, there is a check box called ‘Refine’.

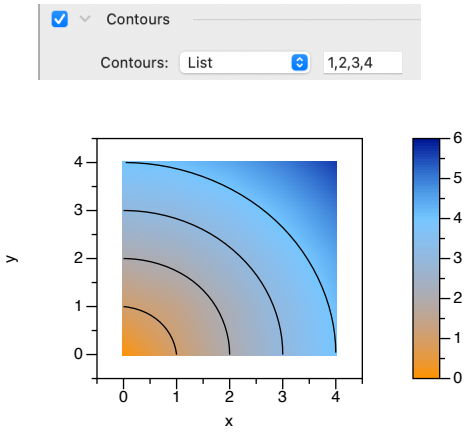


By checking this box, the values will be interpolated between each grid, resulting in a smooth, continuous representation of the data.

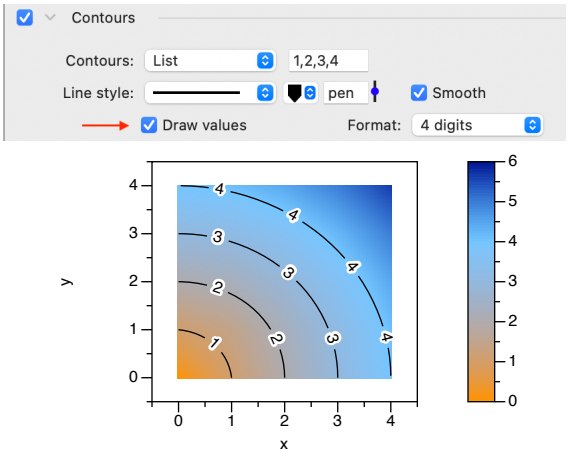


Contours

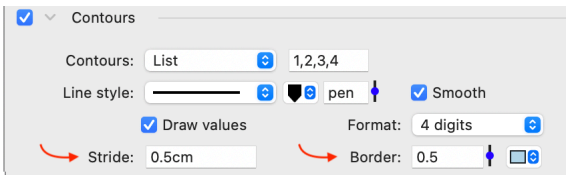
Contour lines can also be drawn. Select the contours check box. Then use the Contours menu to either select a column that contains the numerical values where you want to draw contours, or select 'List' and then type the values in, as shown below.

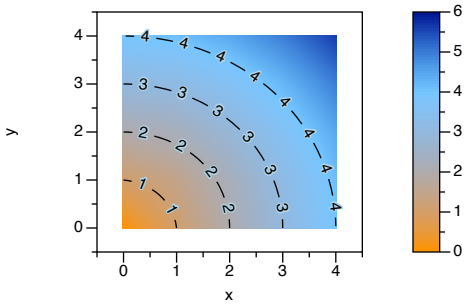


By checking **Draw values**, the numeric values for each contour are shown.



You also can control the distance between each value on the contour, the font size, the number format, and the border around each number. The border around the number is white be default but you can customize to any color.





Histogram

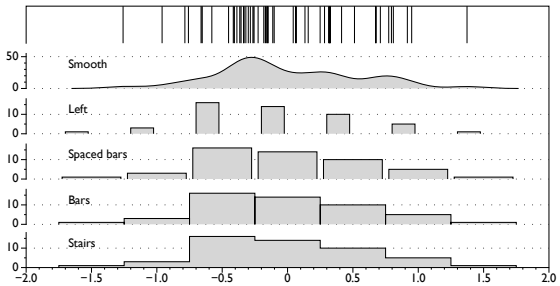
The **Histogram** command is used to analyze statistics for a list of values. You can create a histogram by first, selecting a column of data. Next, add Histogram command under the Draw pop-up menu in the toolbar or from the menu **Command > Add Histogram**.

Histogram Basics

In the default setting, the histogram command requires a column of numbers or dates that are specified in the **Values** drop-down box, determines a bin size, and counts how many values are in each bin.

The data are represented graphically based on what histogram **Type** is selected. The default option is 'Stairs', which does not draw individual bars but instead just the tops, and approximates the probability density function. Use the **Along** drop-down box to draw the histogram along the x-axis or y-axis.

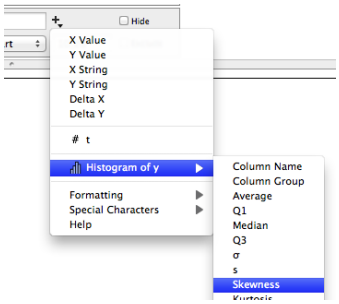
The **Type** menu allows you to select how to draw the histogram. These options are illustrated below. A very common method is 'Bars', but 'Stairs' is useful when you have a large number of bins, and is the default setting for this reason.



'Left/Right' will allow you to compare two distributions, one command will draw the histogram on the left side of each bin, the other on the right side of each bin.

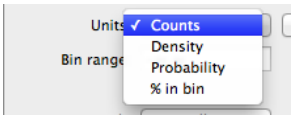
The 'Smooth' option adds together Gaussians. You can adjust the width of the Gaussians using a slider or by specifying the value exactly.

The histogram command also computes a number of basic statistics for the list such as median, mean, standard deviation, etc. These numbers are displayed in a table inside the drawing command, but can also be used in a label.



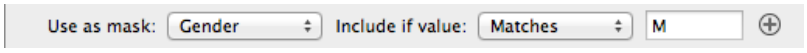
Histogram Units

The **Units** drop-down box is located in the detail view of the command. 'Count' means that the y values are how many numbers are in each bin. 'Density' means the value is count/bin width and probability means count/(width*total count). This is best understood if you consider the integral of the curve. For Density the total integral is the number of values. For Probability the total integral (area) is one.



Mask

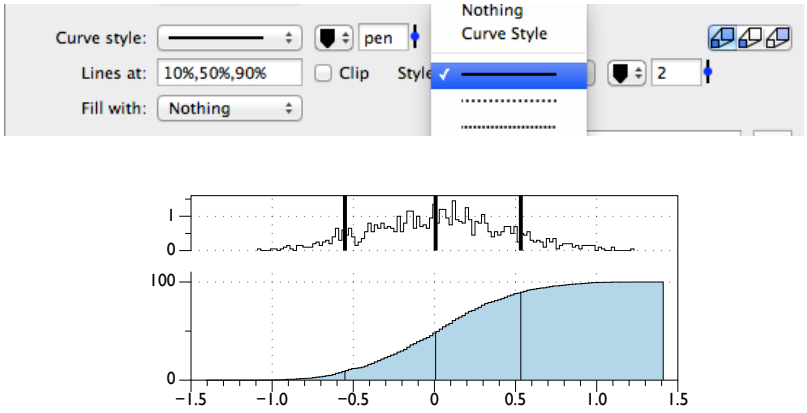
If you want to compute a statistic for a sub-set of the rows you can use the mask command. For example, let's consider an example where you have values and genders (M or F). The data consists of two columns 'Value' and 'Gender' and a number of rows. Select the Gender column as the mask column and include if it matches the string M. This only computes the histogram of the corresponding rows in Values.



Indicate Percentages

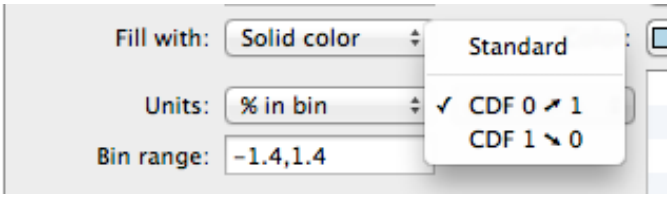
It is possible to draw lines at particular points in the distribution. For example if you want to draw lines where the median is, or where you have 10% or 90% of

the values to the left, you can use the **Lines at** field. By default this field is empty, but you can type in a list of numbers between 0 and 1.



For convenience the symbol '%' is defined as a variable with the value 0.01. Thus, 50% is the same as 50*0.01 = 0.5. So you can enter either 0.1, 0.5, 0.9 or 10%, 50%, 90%. This works for all histogram types, even the cumulative density option which is selected to the right of the units menu.

The **Clip** check box determines that the line should be clipped by the distribution shape. This is what the lower graph does in the figure above. The default line style for these lines is the same as the line for the distribution, but you can specify a different line style.



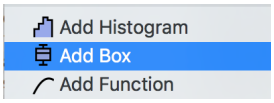
Box

The **Box** command is used to represent data using non-parametric approaches, which do not make any assumption about the underlying shape of a distribution of data.

Box plots, also called Box and Whiskers (see wikipedia), are a way to describe a distribution of numbers, and provide a graphical way to compare two distributions.

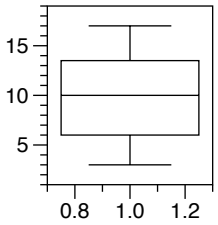
Box Command Basics

In the simplest case, you specify a single column of numbers when creating the **Box** command. To create the command, select from the **Draw** menu in the toolbar or use **Command > Add Box** from the menu bar or use a tool bar shortcut.



Given a column named 'Data' (as shown below) the following command will be created along with the graph below.

#	Data
1	3
2	4
3	8
4	10
5	12
6	15
7	17

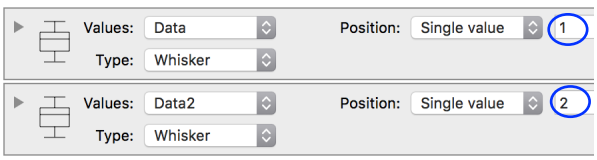


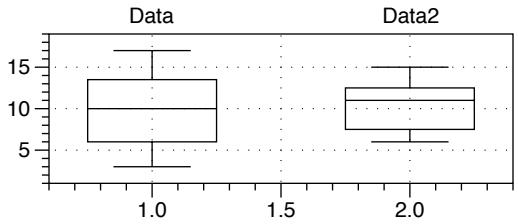
Summary statistics are calculated on the data specified in the **Values** drop-down box. The box plot shows: the minima, the maxima, the median, and the first and third quartile.

The **Position** drop-down box on the main view indicates the location of the box on the X-axis. When the **Position** is set to 'Single value', the number to the right indicates the numerical location on the X-axis.

As illustrated in the following examples, the **Position** setting is used to specify the location of a single box plot, or to group data, resulting in multiple box plots.

1. If your numeric data is in multiple columns, you can create multiple **Box** commands and modify the **Position** of each plot.

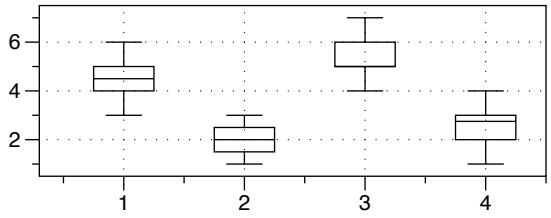




2. If your numeric data is in one column and you also have a second column that defines the bin for the data, you can create multiple box plots with one single **Box** command.

Location	x
1	4
1	5
1	6
1	3
1	4
1	5
2	2
2	3

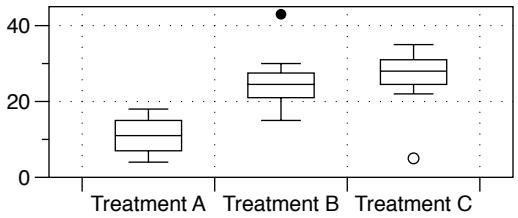
This makes it easy to create multiple boxes with one command.



3. In the third example, the **Position** drop-down is used to select a text column that is used to bin the data.

#	Treatment ab	Value
1	Treatment A	10
2	Treatment A	12
3	Treatment C	30
4	Treatment A	4
5	Treatment B	22

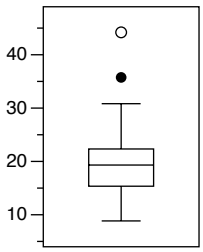
The **Box** command groups the data and changes the numeric labels to text labels on the X-axis.



Outliers

Outliers are drawn as filled in circles. An outlier is a value that is larger than 1.5 times the Inner Quartile Range (IQR) from the median. IQR is the difference between the first and third quartile. The whiskers are only drawn to the smallest/ largest non-outlier. Extreme outliers that are 3 times the IQR are drawn as open circles.

Here is a data set with one outlier and one extreme outlier. Use the gear menu to extract a column that identifies outliers with a value of 1 and extreme outliers with a value of 2.



	Values	Outliers
1	11.5	0
2	35.8	1
3	20.3	0
4	44.2	2
5	17.9	0
6	18.9	0
7	24.9	0
8	16.7	0
9	30.8	0
10	19.9	0
11	14	0

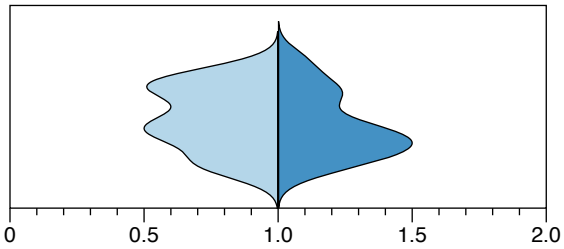
Type

In addition to the standard box plot, the **Box** command has several different types of plots that can be selected from the **Type** drop-down box, including:

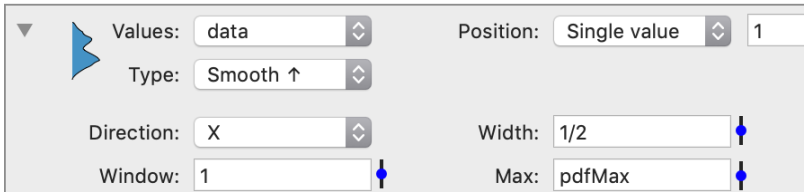
- Whisker — the default setting.
- Points — a point cloud of the data.
- Point + Interval – a display the mean or median along with different intervals, such as the min/max, percentiles, or confidence intervals around a mean.
- Probability — a probability density function (PDF).
- Histogram — a sideways histogram.
- Violin — a violin plot that scales either like a PDF (Violin PDF) or based on the number of data points (Violin Count). When you have multiple violin plots drawn with the same command, these two options will give different results when the number of underlying data points varies.

- Smooth — a smooth histogram that can be in either the x or y direction. The smooth type has four general options, that are used to specify the direction of the histogram along the chosen direction. The smooth options with the '#' symbol will scale histograms based on the number of data points.

Here is an example using the 'Smooth' option in two commands at the same location but one is left and one is right.



The direction is specified in the expanded view.



Specifying Labels

When **Position** is set to a column, the **Labels** drop-down box menu appears beneath it in the main view of the drawing command.

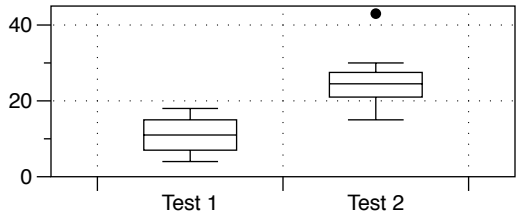
By default, **Labels** is set to 'All,' indicating that all the data are plotted and the label on the X-axis is taken from the **Position** column (i.e., the column called 'Treatment' in the above example).

Using the **Labels** drop-down box, you can change the X-axis labels without modifying the data. First, set up a column that contains the bins you want to include. When you also want to change the labels, create a second column that specifies the name you want to use.

bin ab	name ab
Treatment A	Test 1
Treatment B	Test 2

Next, set the **Labels** drop-down box to the column that contains the bins you want to include. To the right, a second drop-down box will appear that allows you to select a display name for the X-axis.

By default the display name is 'same' but you can specify a different column, as shown above. These settings result in the following graph with only two categories included (i.e., 'Treatment A' and 'Treatment B').



Note that you can also change the order of the boxes by changing the order in the column specified in the **Label** drop-down box.

Setting the Stride

When the **Position** is set to a numerical column, the **Bins** drop-down menu appears beneath. From this menu, you can set the bin stride to specify how to group the data.

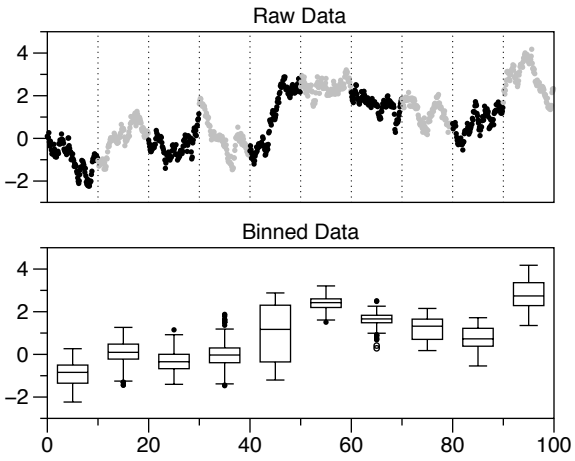
The data is assigned to a bin such that

$$a \leq x < a+\text{stride}$$

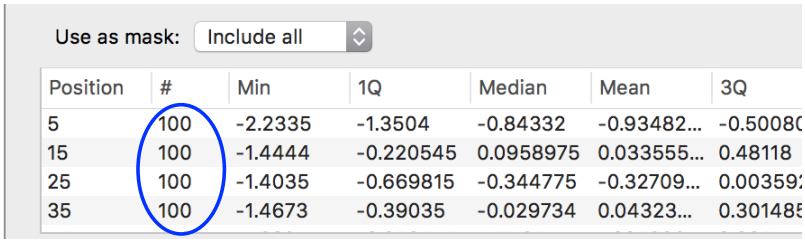
where a is the minimum for each bin, starting with zero. In the example below, we have 1000 data points, where the **Position** is set to x, which starts at 0, increases by 0.1, and ends at 99.9.

#	x	v
1	0	0.10103
2	0.1	0.057234
3	0.2	0.041808
4	0.3	0.26635
5	0.4	-0.16775
6	0.5	-0.36212

By setting the **Stride** to 10, the data is divided up into 10 bins, 100 points per bin.

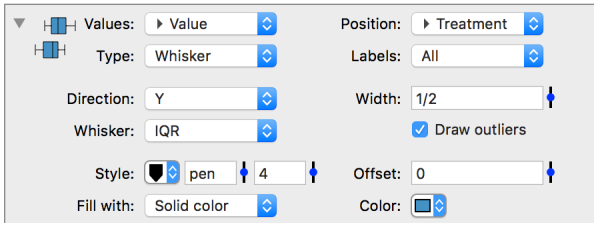


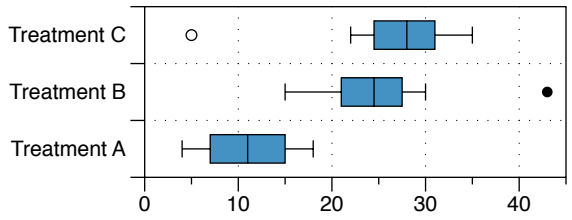
You can confirm the number of points in each bin in the detail view.



Detail View

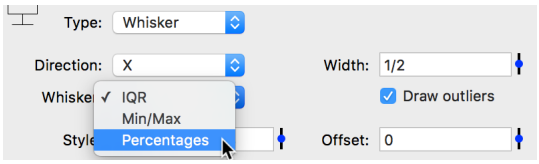
In the detail view, you can customize your box plot. You can vary the **Width** of the box plot. The **Direction** drop-down box allows you to position the boxes on the Y-axis. The **Fill** with drop-down box has options for solid colors, patterns, and gradients.





The options available in the detail view will vary slightly depending on the **Type** of graph selected.

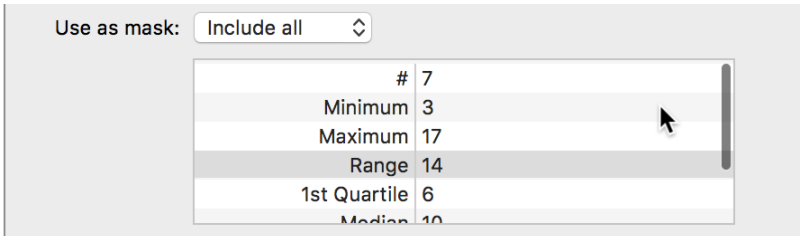
For the 'Whisker' type, you can specify displaying your data in various formats such as 'IQR', 'Min/Max', or 'Percentages'. You can also opt not to show outliers by unchecking **Draw outliers**.



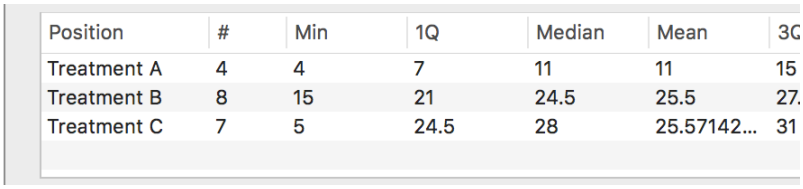
Summary Statistics

In the detail view of the **Box** command, a scrollable table is shown that provides a list of summary statistics.

If you have a single box, the statistics will be in a single column.

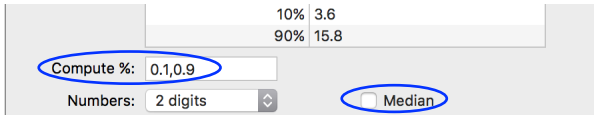


If you have multiple boxes drawn by a command, each box is listed as a row in the table.

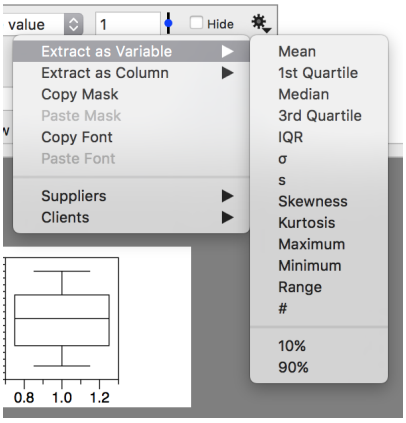


Position	#	Min	1Q	Median	Mean	3Q
Treatment A	4	4	7	11	11	15
Treatment B	8	15	21	24.5	25.5	27.
Treatment C	7	5	24.5	28	25.57142...	31

Directly below this table, you can specify additional percentages that you would like to compute. There is also a check box that will add a label with the numerical value of the median to the box plot.

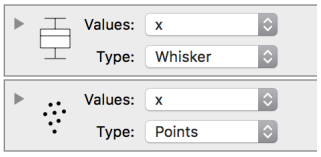


Click on the gear menu to see a list of summary statistics that can be extracted into variables or into the data table as columns.

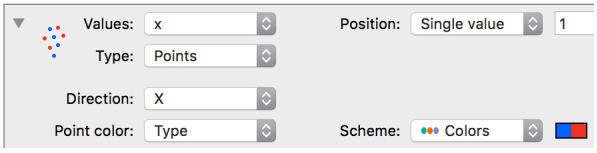


Example: Using a Points Cloud with a Color Scheme

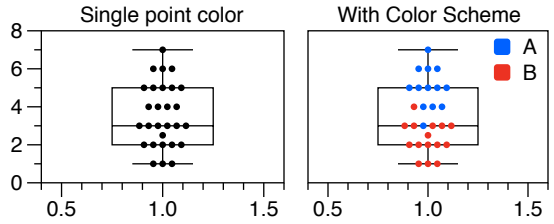
When you set the **Type** to 'Points', a point cloud is created. This allows you to overlay the box plot with the underlying data by combining two commands.



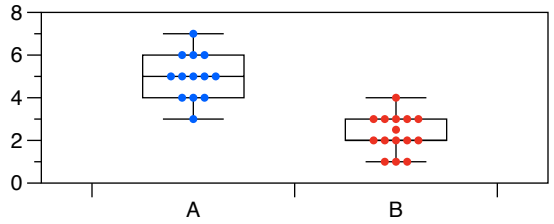
You can use a **Color Scheme** or **Color Ramp** to change the color of points based on an additional column.



The **Color Scheme** can be used within a single box plot as shown below.

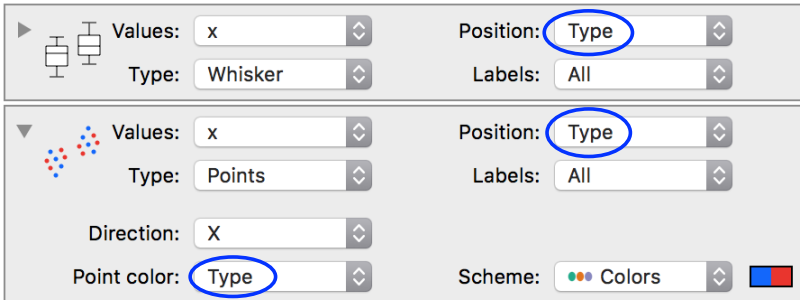


You can also use the **Color Scheme** as way to indicate categories for individual boxes.



In the graphs above, the **Position** and **Point Color** were set to the same columns.

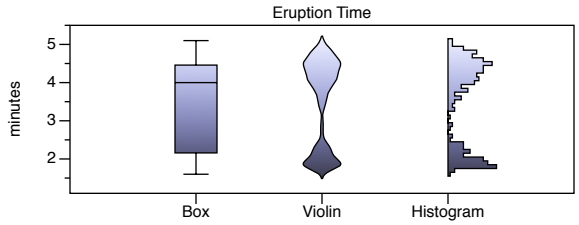
Postion and Point Color set to the same Column



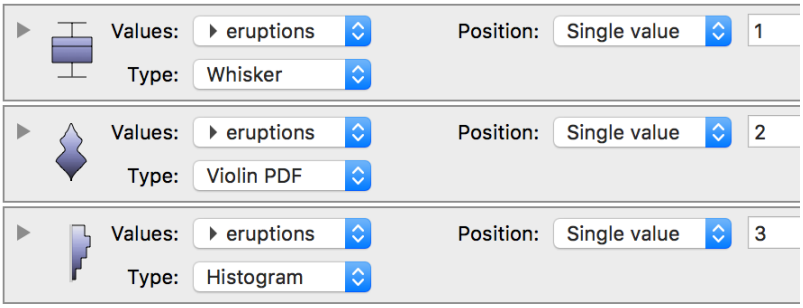
Example: Bimodal distribution

The standard box plot does not reveal the shape of data distributions. For example, bimodal data will be represented more completely using histograms or violin plots.

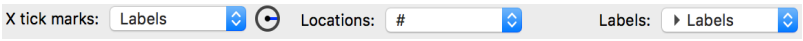
For example, the eruption time of Old faithful Geyser in Yellow Stone National Park, U.S., is a classic example of a bimodal set of data. In the following figure, the eruption time is represented using a box plot, a violin plot, and a sideways histogram.



Three **Box** commands are needed to create this figure, one for each **Type** (i.e., 'Whisker', 'Violin', 'Histogram'). The **Position** is set to 1, 2 and 3 for each command, respectively. (Source: Old Faithful dataset, faithful.csv)

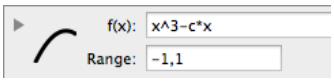


Note that the labels on the X-axis were specified in the **Axis** settings. (See the **Axis** section of the **Layout** Chapter for more information.)



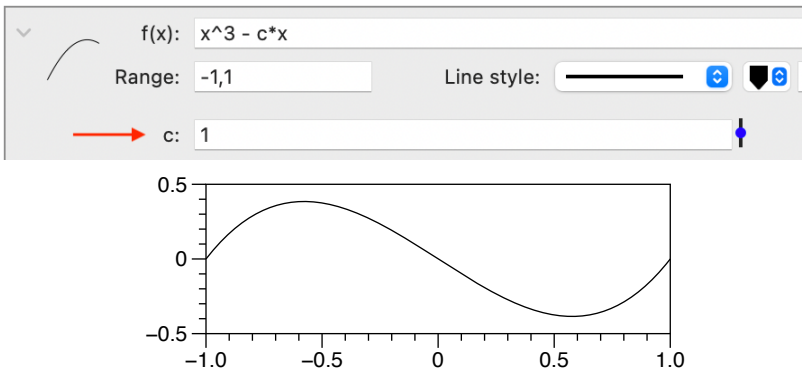
Function

This allows you to specify an analytic function and a range where it should be drawn. The function needs to be a function of x.

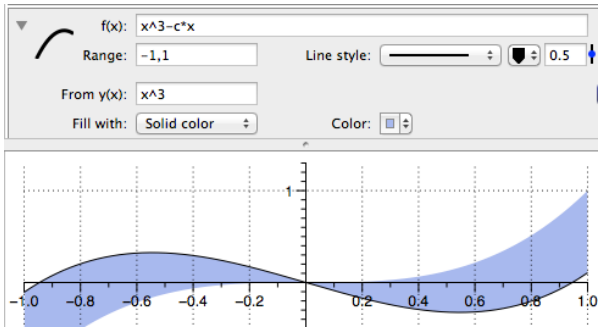


If you use additional variable names that are not already defined (See the Variables section), an entry box for each will be listed below.

For example, here the variable ‘c’ was not defined. An entry box for ‘c’ is automatically added with an assigned value of 1. You can type in a new value or use the slider to the right of the entry box to vary. You can also enter names of other variables.



You can also specify a 'From y(x)' function. This allows you to fill between two analytic functions.



To learn more about the many built-in functions you can use in expressions, see the Function Reference section of the Appendix.

Fit

The **Fit** command is used to compute functional relationships between x and y variables in a data set. This is often also referred to as regression. This command computes the fit, draws it and also gives you access to fit parameters and quality measurements. Even if you exclude the command, and therefore don't draw it in the graph, DataGraph still computes the fit and allows you to use the variables in token fields anywhere in that graph, or extract it as a variable and use it in other graphs, expression column or numerical fields.

The fit works by minimizing a term of the form

$$\sum (y_i - f(x_i))^2,$$

where f depends on the parameters that you want to optimize.

The exception is the exponential and logarithmic fit, where the routine minimizes:

$$\sum(\log(y_i) - \log(f(x_i)))^2.$$

You can specify a weight column, such that each term is weighted by the corresponding row in the weight column. In this case, the term that is minimized is:

$$\sum(w_i(y_i - f(x_i)))^2.$$

Thus, where the weight is larger, relative to other points, the error is smaller.

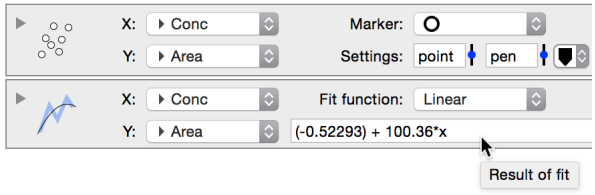
Fit Command Basics

At the top of the command object, you select the x and y columns and what fit function to use including: 'Linear', 'Quadratic' and 'Cubic', which are all are polynomials.

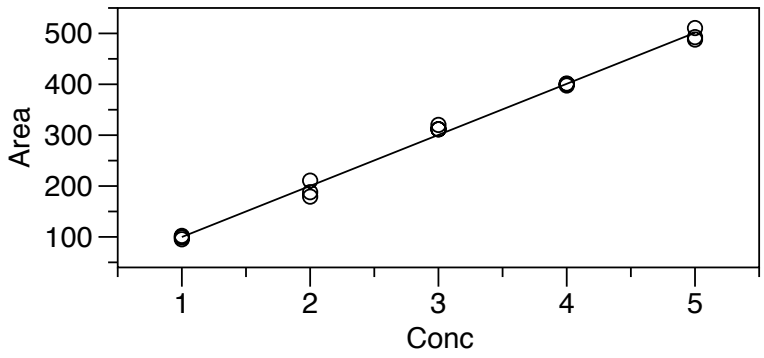
Selecting 'Polynomial' allows you to quickly vary the order for a polynomial fit and to fit higher-order polynomials. For high-order polynomials, the coefficients themselves don't generally have a lot of meaning or predictive capability, but the approach can be useful for showing trends. To most accurately evaluate a high-order polynomial, use the **Plot action** column 'Evaluate' option.

If you add the **Fit** drawing command to an empty graph the X and Y selection are not populated and you would have to manually select the X and Y columns.

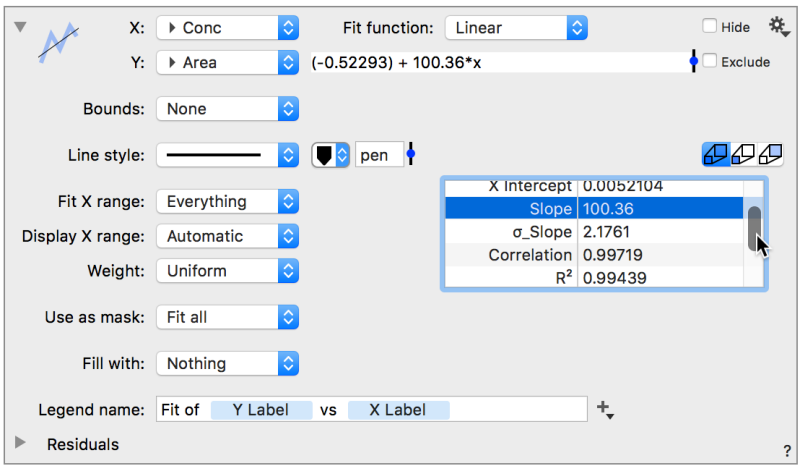
A short-cut is to first create a **Points** command using the data you want to fit. Next, click the **Fit** command. The X and Y columns will automatically be populated and the equation for the linear fit is shown in the **Fit** command.



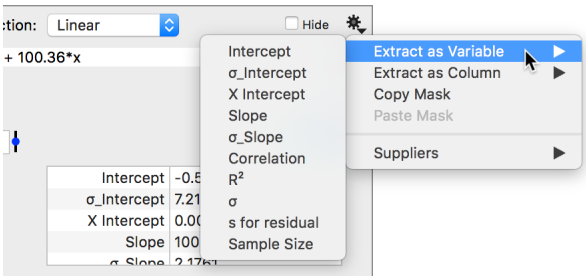
The Graph now includes both the points and the fitted line.



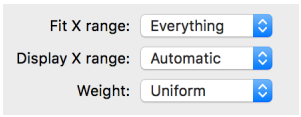
If you open up the disclosure triangle you will be able to see summary statistics related to the goodness of the fit. You can use the mouse to move the list up and down.



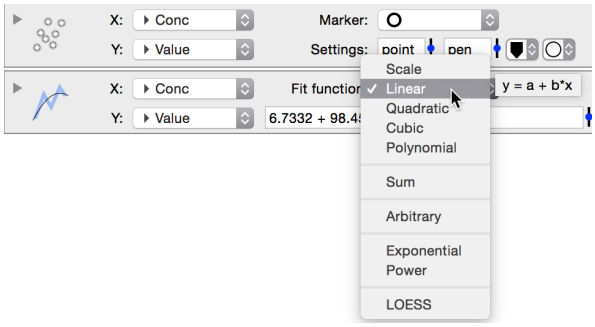
Each of these values can also be extracted as a variable using the gear menu, allowing you to use them in an **Expression** column. Click on the gear menu in the top right hand corner of the command. Highlight the **Extract as Variable** option and a list of variables you can extract is shown.



The command also gives you further options to apply **Weights** to particular data points or to specify which data to include in a fit. The **Display X range** option will allow you to extend or limit the best fit lines to any specified range.



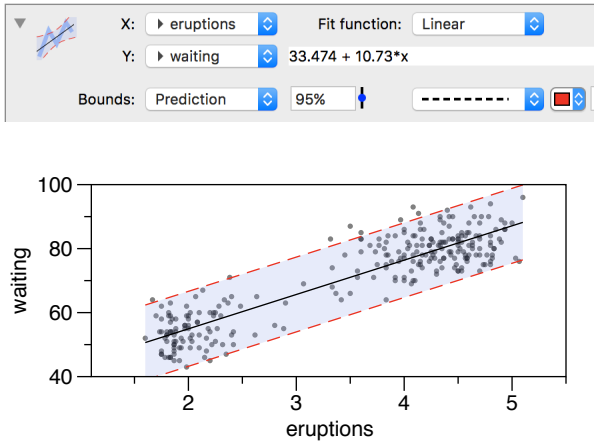
Other fit types can be selected as shown below.



The **Sum** options allows you to specify a comma-separated list of functions you want to use. DataGraph finds the best linear combination of those functions. For example, if you want a quadratic fit that goes through the origin, you can use the functions x and x^2. For a more general fit, use the Arbitrary fit option.

Confidence and Prediction Intervals

You can easily add confidence and prediction intervals to a Linear fit. The **Bounds** option will be shown in the detail view (see below). Select the type of interval to display along with the statistical level of significance.

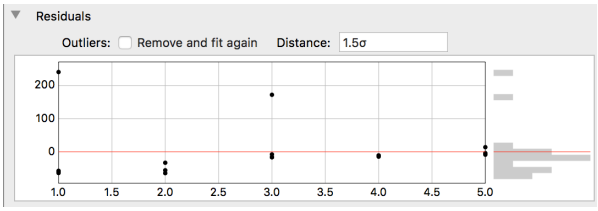


If you want to show both intervals create two commands with the same data and specify one as 'Predication' and one as 'Confidence'.

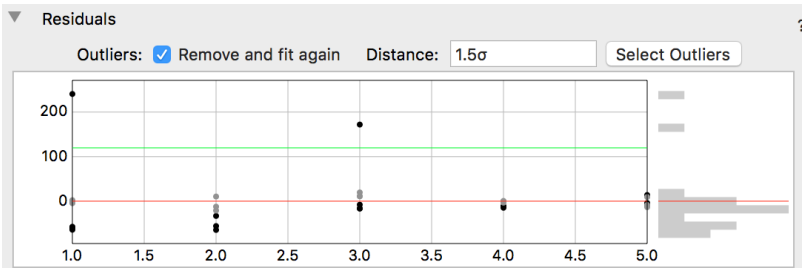
When you display either interval, the **Fill** option will fill-in the region of the interval. The **Fill** option is also located in the detail view of the command.

Determining Outliers

There is an additional disclosure triangle at the bottom left hand corner of the **Fit** command that can be used to test the impact of outliers. When you open this triangle you see the following, where the points plotted are the residuals from the fit function versus x.



If you click the **Outliers** check-box a green line appears that corresponds to 1.5*σ, where σ is the standard deviation of the residual. The points outside this green line are considered outliers according to this criteria.



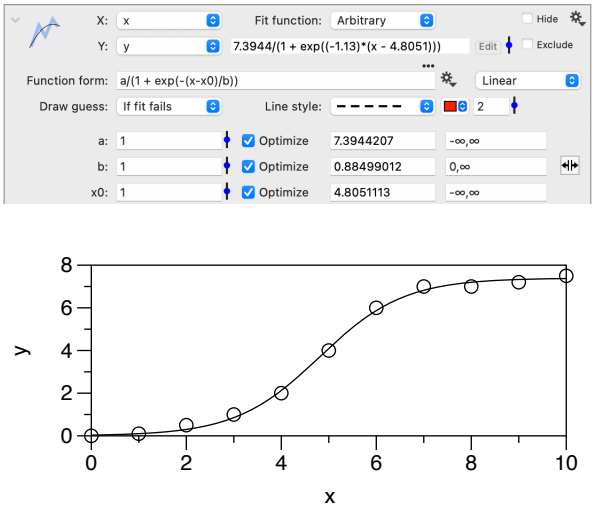
The fit is then recomputed with these points excluded from the fit. To identify the outliers in your data, click the 'Select Outliers' button in the right corner. Any outliers will then be highlighted in the Data Table.

Arbitrary Fit

The **Arbitrary** fit allows you to specify a function with unknown parameters and DataGraph computes the parameters to minimize the difference between the function and the data. Mathematically, this is called generalized least squares and allows you to specify a nonlinear function.

When you specify the function form, unknown variables are automatically shown listed below the function along with an initial guess. You can change the initial guess and also have the option of providing bounds on the parameters. By default, the range is set to (-∞, ∞). You can constrain the maximum value, the minimum value, or both.

In the following example, the lower value for b is set to zero, limiting the coefficient to positive values, and providing a good fit of the data.

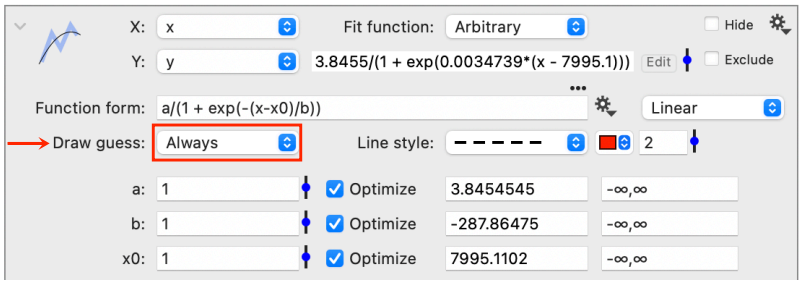


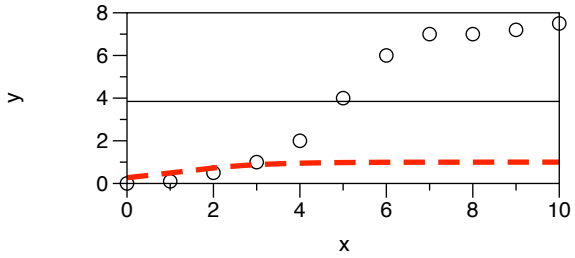
NOTE: For information on built-in functions you can use to create expressions see the Function Reference section in the Appendix.

Unlike linear functions that always have a unique solution, nonlinear functions can have multiple solutions. An iterative process is used to find the fit and may find a solution, or not converge at all. Thus, it is important to provide a good initial guess of the parameters.

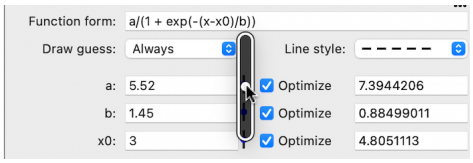
When the fitting routine is not finding a solution or you think the best solution has not been found, you can assess your initial parameter values by changing **Draw guess** to 'Always'. This draws a red dashed line showing the solution at the initial parameter values.

In this example, a best-fit line was identified (the solid horizontal black line) but it is not a good fit for the data. **Draw guess** is set to 'Always'; thus, you can see the initial guess (red dashed line). In this case, the red line corresponds to the function evaluated where a=1, b=1, and c=1.

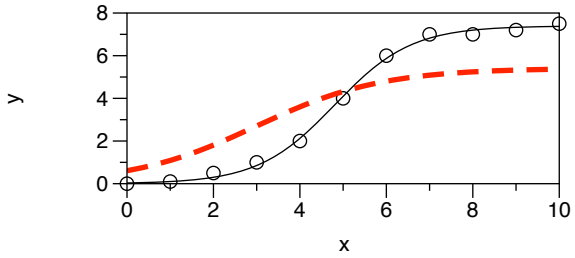




Using the sliders to the right of the initial values, you can explore the impact of changing the initial values.



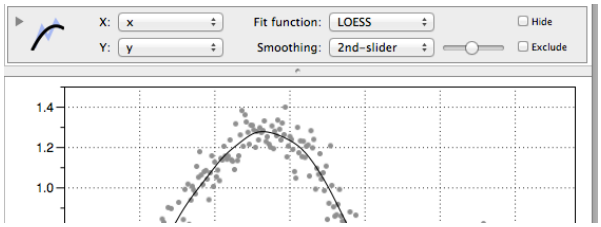
In this example, changing the initial values provides a much better initial guess, resulting in a good fit of the data.



In addition to the visual representation of the fit, the program also outputs goodness of fit results such as the RMSE and the standard errors for the parameters.

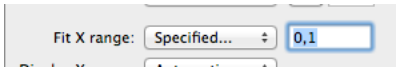
LOESS Fit

The **LOESS** fit is a non-parametrized fit. That means that the fitting function is not an analytical function, but is rather computed by a lower order fit at each data point. Use Wikipedia to get a more detailed description, but the short explanation is that at each point on the interval, DataGraph computes a lower order least squares fit in that neighborhood and uses the value of that fit as the function value. You can adjust the interval width, and select if the local fit is linear or quadratic. The smaller the interval, the noisier the fit function and there might not be enough points to do a local fit.



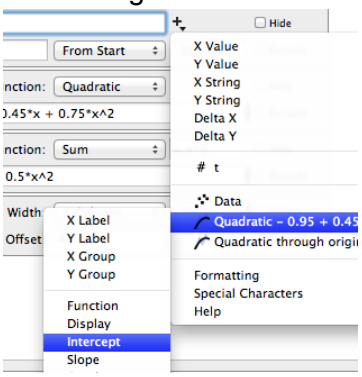
Restricting Data to Fit

By default all the rows are used to compute a fit. Rows that are invalid or empty are ignored. You can select a sub-set using two methods or a combination. The first is the 'Fit X range' option. The default is ‘Everything’, but if you select ‘Specified…' you can type in a range for the x values. This will restrict the fit based on a range along the x-axis. You can also use the mask mechanism. This allows you to base the selection on other columns.



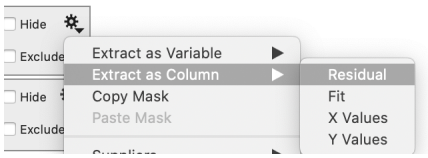
Displaying Fit Results

It is easy to display the results of the fit function in labels, legends, etc. In any text entry field, click the plus symbol to the right of the field and select a result from any of the fit functions. This adds a token that is updated automatically when the fit changes.

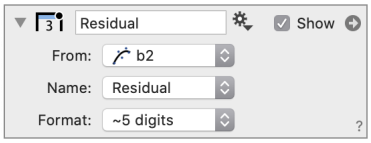


Extracting Residuals

If you want to view or plot the residuals, or which rows were used for the fit, click the gear menu on the right side of the Fit command. Select **Extract as Column > Residual**.



This creates a column entry in the data table.



Multivariable Fit

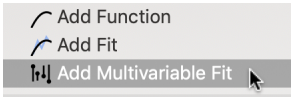
In addition to the **Fit** command described in the previous section, DataGraph also includes a **Multivariable Fit** command. The **Fit** command uses two columns x and y, and a function f(x) that has one or more unknown parameters.

The **Multivariable Fit** is very similar, but instead of just using a single column x, you can have multiple columns. That is you have n input columns x1,x2,…,xn and a single output value y. The function depends on n arguments and is parametrized with one or more unknown parameters. **Multivariable Fit** command minimizes the sum of the residuals as follows

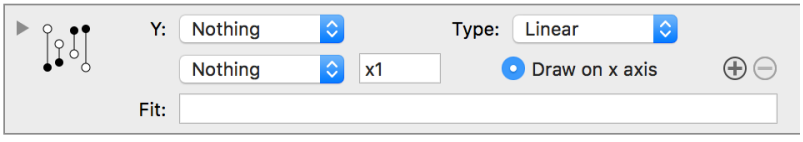
$$\sum \Big(y(i) - f\big(x_1(i), x_2(i), \ldots, x_n(i)\big) \Big)^2 .$$

Multivariable Fit Command Basics

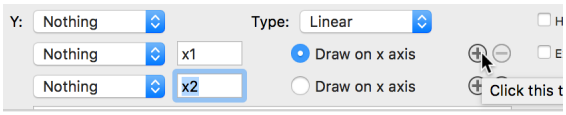
By default, the **Multivariable Fit** is not included as a shortcut in the tool bar. Thus, to conduct a multivariable fit, you first create a **Multivariable Fit** drawing command from the Command menu.



The command has a drop-down box to specify the column that corresponds to y and to one x variable, x1.



To add more x variables, click the plus symbol to the right of the existing x1 variable. Assume x1, x2, and y are columns in the data table.



Conducting a Fit of a Linear Function

One common application for a multivariable fit is a linear function. For example, if you have experiments where the input is x1 and x2 and you measure y. A similar linear relationship can be expressed as

$$y = f(x_1, x_2) = C + a_1 x_1 + a_2 x_2.$$

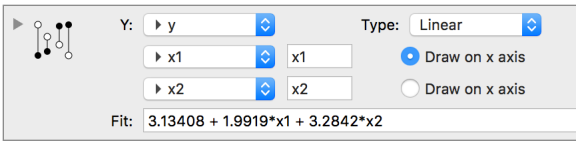
In this case, the goal is find the values for a1, a2, and C that minimize either the square of the residuals

$$\sum \Big(y(i) - \big(C + a_1 x_1(i) + a_2 x_2(i) \big) \Big)^2$$

or the weighted square.

$$\sum w(i)^2 \Big(y(i) - \big(C + a_1 x_1(i) + a_2 x_2(i) \big) \Big)^2$$

In the **Multivariable Fit** command, select y as the top entry, select the x1 column in the top entry and add a second entry for x2. The result of the fit is shown below the parameter list

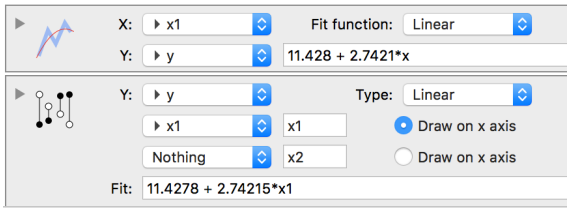


In this case, the best fit parameters are C=3.13408, a1= 1.9919 and a2 = 3.2842.

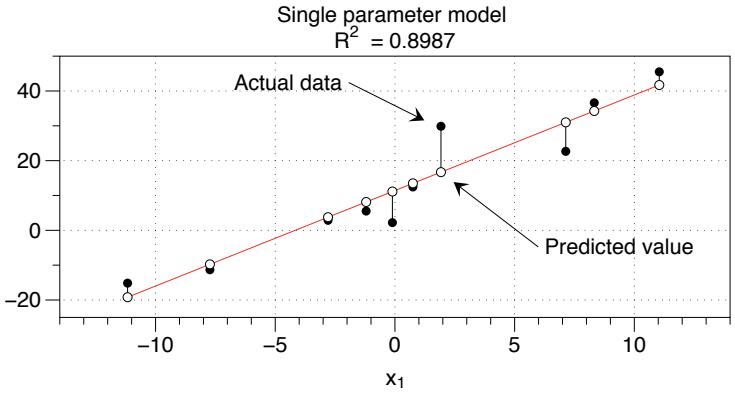
Viewing Fit Results

Since a multivariable fit is not a function of one argument, it is not possible to draw it completely on an xy graph. The drawing command computes the optimal set of parameters, but the representation that is drawn on the screen shows each data point along with the predicted value based on the fit.

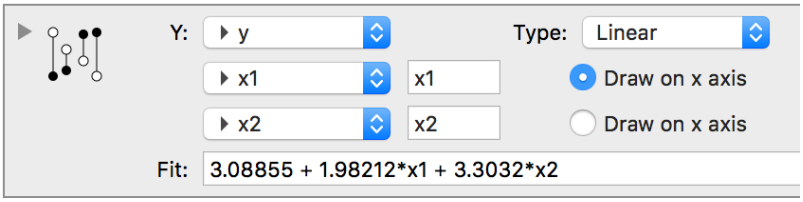
For example, consider a data set that is dependent on x1 and x2. Initially, we will determine a linear regression based only on x1. We can fit this equation using either of the Fit command or the **Multivariable Fit** command as shown below.



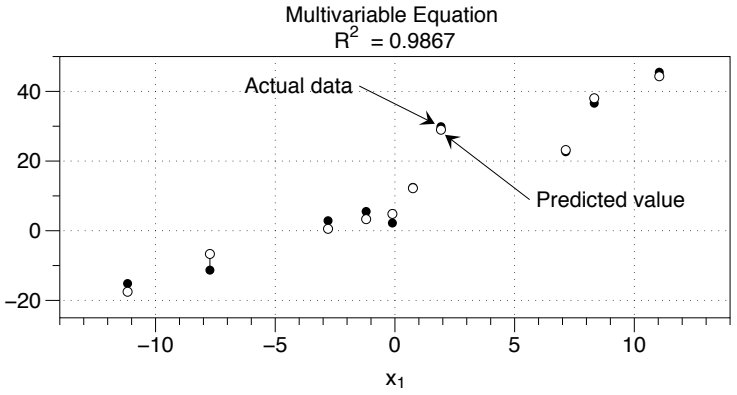
In the figure below, the result of the **Fit** command is drawn in red. The points are the representation of the fit as drawn by the **Multivariable Fit** command. By default, the actual data are drawn as solid points and the predicted values are drawn as open points. The line between the actual and predicted corresponds to the residual. In this case, the predicted points from the **Multivariable Fit** command fall on the same line drawn by the **Fit** command since they predicted the same equation.



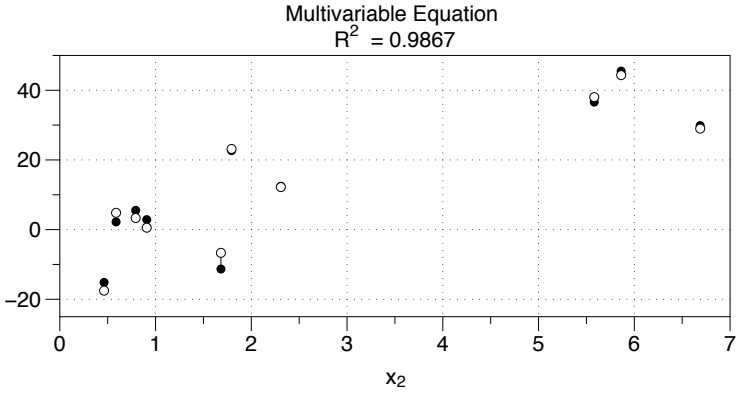
The **Multivariable Fit** can determine the dependency on multiple variables. To further illustrate, we will now add x2 as a parameter in our fit equation.



Since the data depend on both x1 and x2, the difference between the actual and predicted values are now much less. The graph below shows the data with x1 on the x-axis.



To the right of each parameter listed in the drawing command, there are little radio buttons indicating which argument is used for the x-axis. We can use these to show the result with x2 on the x-axis.



Different Fit Functions

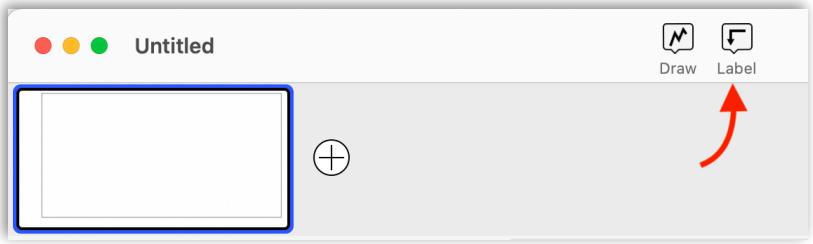
There are three different fit function types that can be specified using the **Type** dropdown box.

The previous sections used only the Linear fit of the form c + a1*x1 + … an*xn. In this case, there are n+1 scalars that are optimized. The ‘Scale’ does not have the initial constant, c, so it is of the form a1*x1 + … an*xn. Thus, function fits using the ‘Scale’ type have an intercept that goes through zero.

The Arbitrary setting allows you to specify the exact function form. Any parameter names in the expression that are not column names or variables are treated as optimization parameters. This works similarly to the ‘Arbitrary Fit’ option for the **Fit** command described in the previous section.

9. Annotating Graphs

There are several commands that are designed for adding labels, legends, highlighting regions and more. These commands are grouped under the Label icon in the tool bar.



One big difference between the commands in this section and the previous section is that these commands add elements that you can drag and adjust on the drawing canvas interactively.

Overall Position

One of the common options these commands have is a **Position** or **Where** setting. By default, these are typically set to "Inside", meaning inside the coordinate box. You can also locate many elements above or below the coordinate box.

If you want to locate an element on the graph without considering for the coordinate box, use the "Figure" option for the position.

Anchor and Offsets

When you add an element such as a text command or legend, they are located in the graph using an anchor and an offset from that anchor point. The approach to locating elements in the graph with an anchor and offset is used for most of the commands in this section.

The anchor points are something that you can specify relative to the locations in the graph. For example, you may want to anchor a legend to certain corner of the graph. Or perhaps place the legend below the graph and anchor to the center.

The when the offset value is set to (0,0) that means the element is at exactly the anchor location. You can drag many of the annotations elements such as legend and labels on the graph. This will change the offset from the anchor point.

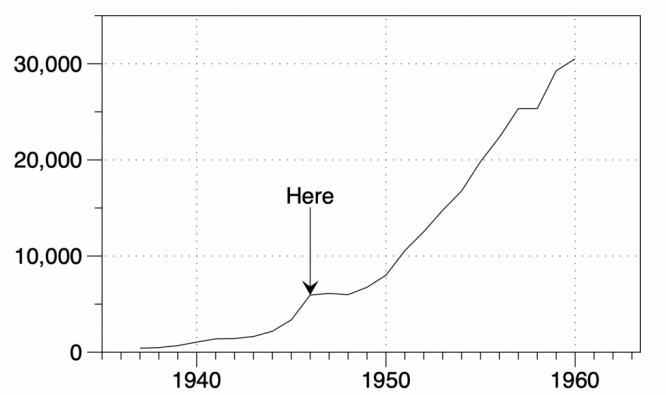
The offsets are specified in pixel coordinates and the two values in the offset are the x offset and the y offset.

Commands in the Label Menu

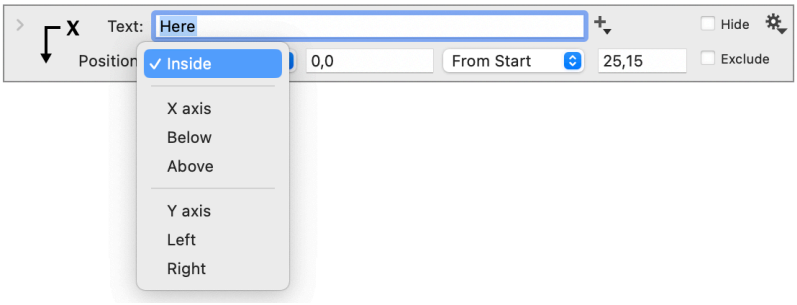
Icon	Name	Description
	**Label**	Attach a label to a single point on a graph.
	**Text**	A text label, typically anchored relative to the boundary.
	**Bracket**	To indicate a relationship between two points. Can use curvy or straight lines.
	**Region**	Highlight a region using a shape. Can choose box, rounded box or circular.
	**Range**	Highlight ranges in X or Y with one or more rectangular regions.
	**Legend**	Picks up information from drawing commands in the graph to create a legend.
	**Custom Legend**	A more flexible. Legend but not as automatic. You specify every element and label in the legend..
	**Color Ramp Legend**	To draw color ramps from scalar fields or for continuous colors.
	**Extra Axis**	More flexibility than the standard axis. Can use to convert units.
	**Graphic**	Add a reference to another graph or paste in a bitmap/PDF.
	**Magnification**	Automatic region of interest or overview. Zoom in on a detail. Create an inset.

Label

The **Label** command allows you to point out a particular point on a graph. When you create a new Label command, it defaults to an arrow that points to the center of the axis region. You can drag the arrow around the screen, and you can drag the label and keep the same point.



There are three parts to a label. The first one is the anchor point. This is the location that the arrow is pointing. There are several different places to base an anchor. The default setting is ‘Inside’, which allows you to specify an x and y coordinate for the location of the arrow.

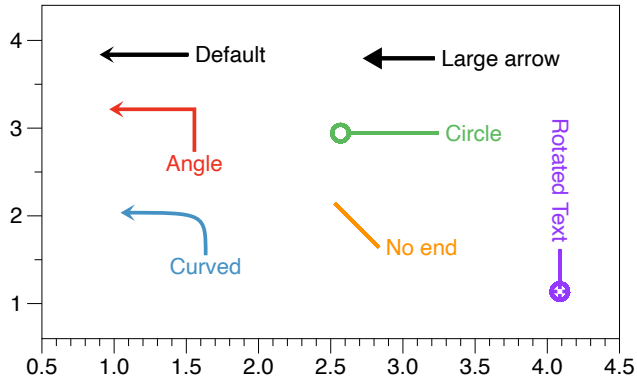


For ‘X axis’, ‘Above’, and ‘Below’, you specify the x value as a coordinate, but the y as an offset. You can drag the Inside arrow up to the x-axis, but the difference is clear when you resize the graph or the data changes, since the ‘Inside’ point has a y coordinate and that coordinate might not work for different data or scaling.

You can drag these arrows by using the mouse, but arrows along the x or y-axis are restricted to only move in that coordinate location, and you need to adjust the offset by entering in a numerical value or using the pop-up slider.

Formating Options

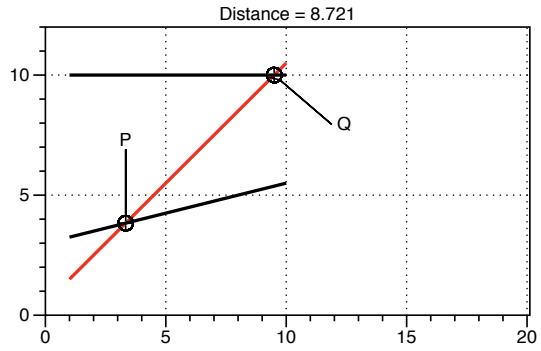
There are options for adjusting the end of the arrow and the size of the arrow. Apart from line thickness and color you can specify the line style. You can for example hide the line, make a straight line or angular/curved arrows.



Drag and Snap to a Location

At the bottom of the **Label** command, are options for **Drag** and **Snap**.

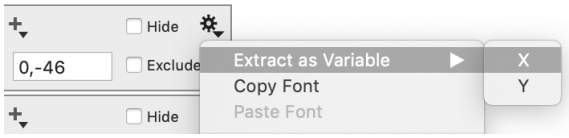
When dragging, the label will also **Snap** to locations on functions or data points (e.g., minima, maxima, intersecting points). For example, here P and Q are highlighted using **Label** commands at the intersection of the lines. The lines are drawn using **Function** commands.



To freely drag the labels, without snapping, uncheck the **Snap** check box.

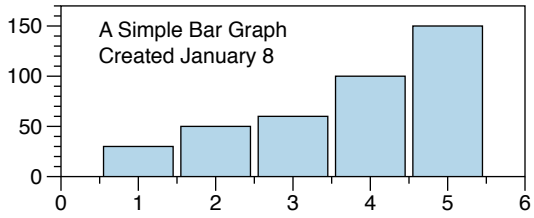
Extract X/Y Location

Extract the value of a label location, x or y, into a variable, directly from the gear menu in the **Label** command.



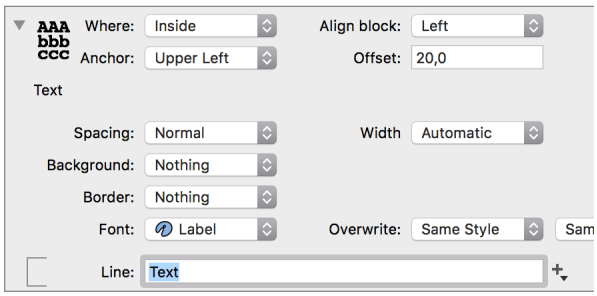
Text

The **Text** command is an annotation tool that is similar to the **Label** command, but has additional functionality that allows you to enter and align multiple lines.

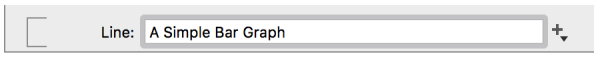


Text Command Basics

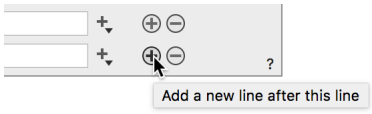
By default, the **Text** command is created with the detail view open.



Type the text you want to display at the bottom of the command.



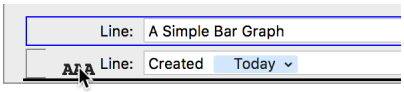
Add or remove lines by clicking the plus or minus buttons the right of the text.



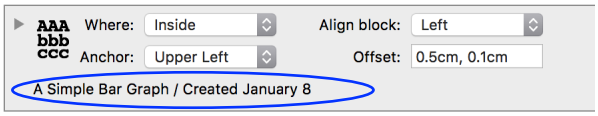
You can also add variables by clicking the plus symbol directly the right. Below, a variable named 'Today' was added as a **Token** containing the date.

(For more on using tokens**,** see the **Tokens** section in the chapter on **Drawing Command Elements**.)

In a graph, the order of the items corresponds to the order in the command. You can click and drag entries to reorder them.

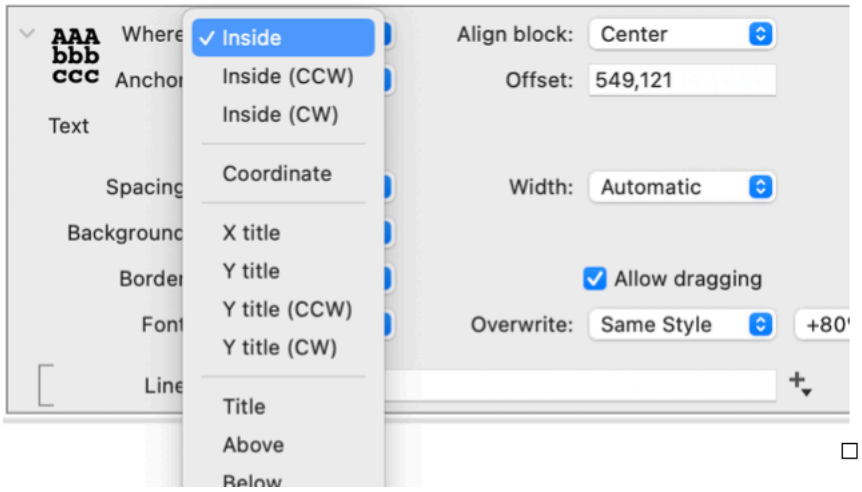


When you close the detail view of the command, the text label will be visible from the main view. You must open the detail view again to edit.

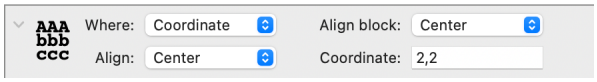


Positioning

The **Where** menu specifies the region of the graph the text is positioned. By default, **Where** is set to 'Inside', but you have a lot of options for the placement of the text. You can also change the orientation to clockwise (CW) or counter clockwise (CCW).

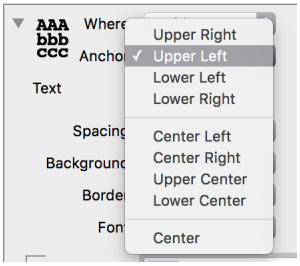


You also have the option of locating the text at a specific coordinate location. When you select **Coordinate**, you have an entry box for entering the location in (x,y).

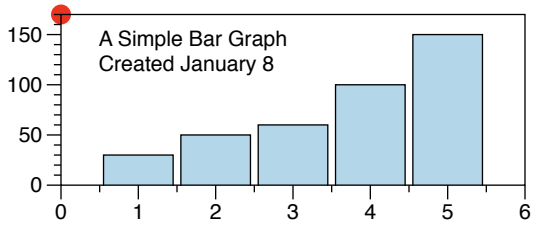


Every other selection requires an anchor to pinpoint the text to a location of the graph, and an offset which is in pixels. Using this approach your text labels stay in the same relatively location, even when the size of the graph changes or the data changes.

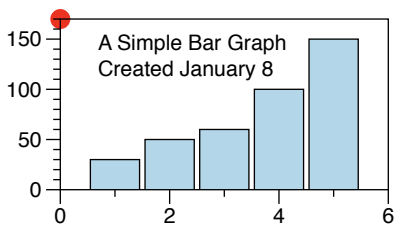
The **Anchor** specifies where in the region the text is placed.



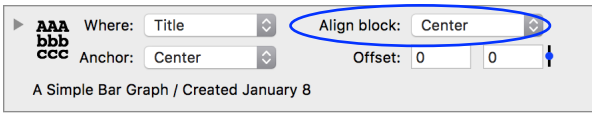
In the following graph, the Anchor point is the default location, such that **Where** is set to 'Inside', and **Anchor** is set to 'Upper Left'. The Anchor location is highlighted with a red point.

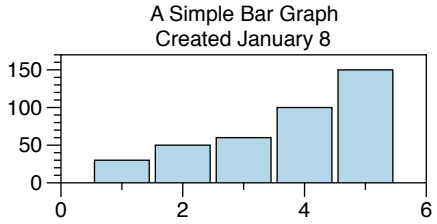


The **Offset** setting changes where the text is located relative to the **Anchor**. In the previous graph, the **Offset** in the x direction is '0.5cm' and the y direction is '0.1cm'. Thus, the label is always the same distance from the top left corner even as you resize the graph.



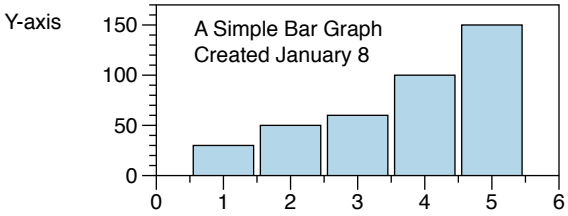
The **Align block** menu allows you to change the alignment.





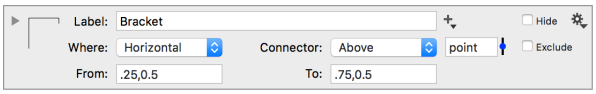
The **Text** command allows you to insert references to variables and values inside drawing commands. This for example allows you to create a table of results, where the values are pulled from fit functions or histograms, even if those commands are excluded from the graph.

Another use of the **Text** command is to create more flexible titles for graphs. The standard axis title will always center the label for the x or y-axis and rotate the y title. If you want more flexibility, create a **Text** command and use that to position the text.

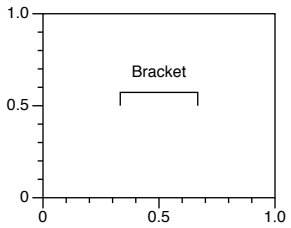


Bracket

The **Bracket** command draws several types of brackets and a text label. The **Bracket** consists of two lines, or arms, with a middle line, or connector, between them. The text label is located in relation to the connector.

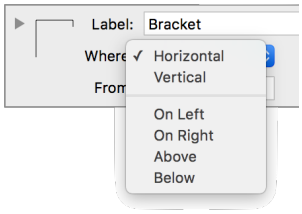


When you create a new **Bracket** command, DataGraph will create a 'Horizontal' bracket with arms of equal length and place it near the center of the graph. The label is placed above the connector.



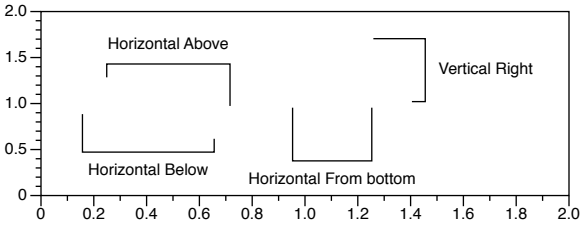
Bracket Command Basics

The **Bracket** can either be located within the graph or in the margins around the graph. The **Where** drop-down menu controls the overall location and the orientation of the connector.

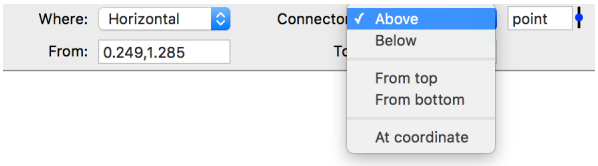


Brackets in the Graph

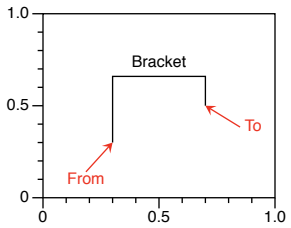
When **Where** is set to 'Horizontal' or 'Vertical,' the Bracket is located within the graph and you can create brackets with various orientations and label directions.



The **Connector** drop-down box allows you to modify how the middle line is anchored on the graph. The options in this box vary slightly depending on whether or not the line is horizontal or vertical.



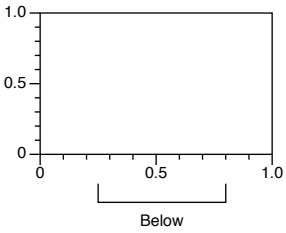
The location of the bracket is also controlled by the **From** and **To** variables at the bottom of the command. These are (x,y) coordinate pairs that define the beginning and ending of the bracket.



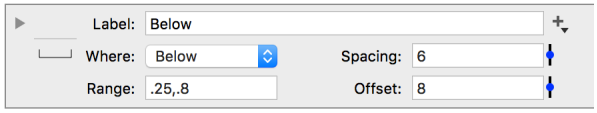
You can also think of the bracket as being a connector between the two points **From** and **To**. If you change the **From** or **To** variables to points outside the current axis range, the range will be expanded. Thus, the range will consider **From** and **To** but not the text label.

Brackets outside the Axes

When **Where** is set to any of the bottom four options (i.e., left, right, above, or below), the bracket is located outside of the axis. These brackets are useful for highlighting a section of the axis and will always have equal arms.



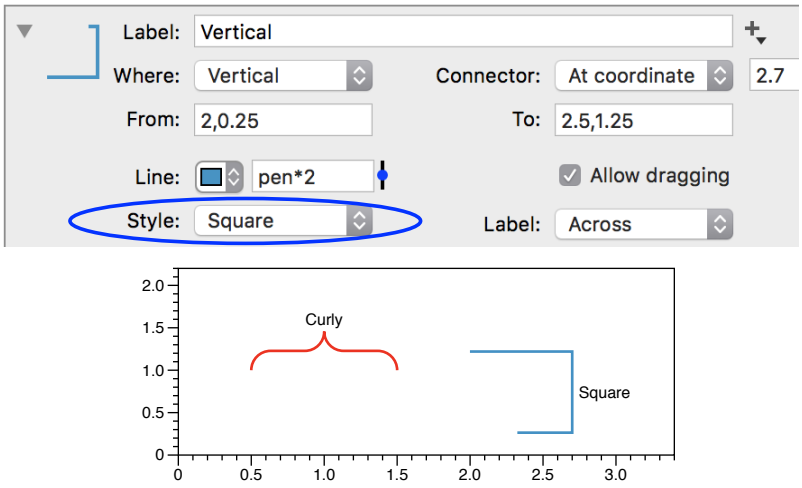
When you place a bracket outside of the axes, The available settings on the drawing command change. Specifically, the available settings include: **Range**, to set the distance between the arms; **Spacing**, to set the length of the arms; and **Offset** to set the distance to axis.



Note that if you change **Where** back to a previous selection (e.g., horizontal), your settings for that selection are recalled. Also, for Brackets outside of the axes range, the graph will expand to accommodate the label.

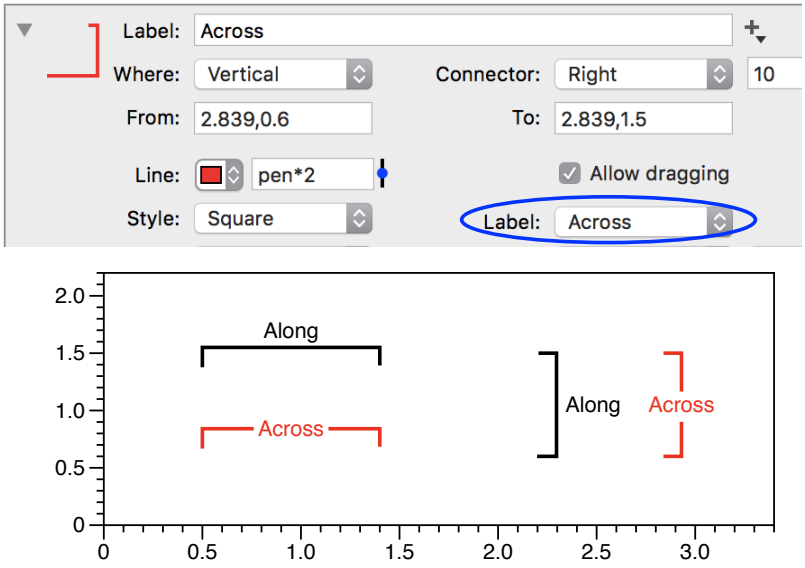
Alternate Styles

You can modify the style of the bracket from the default of 'Square' to 'Curly'.



Label location

You can modify the label location from the default of 'Along' to 'Across' the line.



Font

The font and color of the label is specified at the bottom of the command.

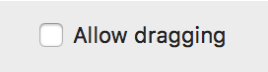
Dragging

The location of a bracket can also be modified by dragging. Dragging works differently depending on whether your bracket is inside or outside of the axes.

When the bracket is inside the axes range, clicking near the middle of the bracket drags the entire bracket. Clicking near the end points of the bracket will move just that end, modifying either **From** or **To**.

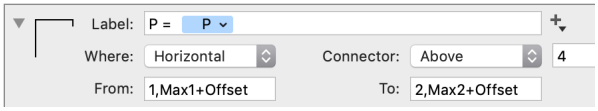
When the bracket is outside of the axes range, you can click and drag an edge to make the bracket wider or narrower. In this case, you cannot drag the bracket by clicking in the middle.

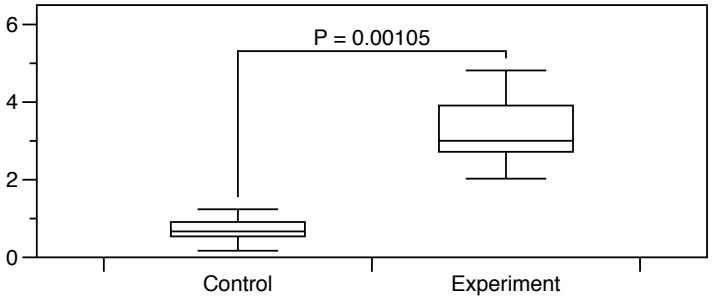
You can also turn off dragging by deselecting **Allow dragging** in the detail view.



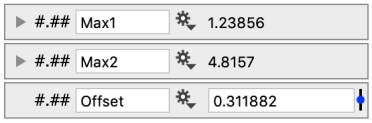
Example: Using Box Plots

The **Bracket** command can be used to indicate the statistical significance between two populations using a Box plot. The p-value is determined using the t-test variable, and is shown in the label by using token.



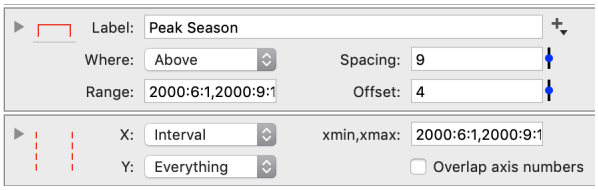


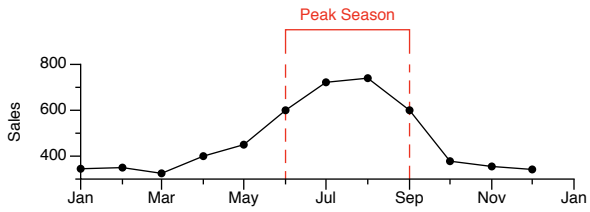
In this example, the values for **From** and **To** use **Variables:** Max1, Max2, and Offset. The values for Max1 and Max2 are pulled directed from the data (i.e., **Number from column** variables). The Offset is a slider to allow the y location of the ends of the bracket to be adjusted simultaneously.



Example: Highlighting an Axis Range

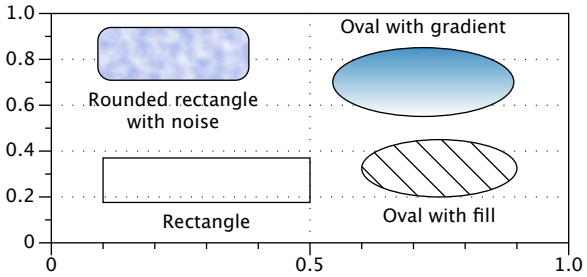
The **Bracket** command can be used to highlight a region of the axis. Here the **Range** command was also used to create the dashed red lines within the graph.





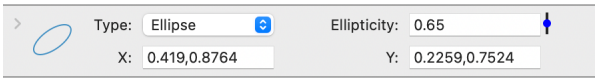
Region

The **Region** command allows you to highlight a region of space within a graph. You must specify a region in x and y, and the shape. Note that you can use variables in the interval definition.

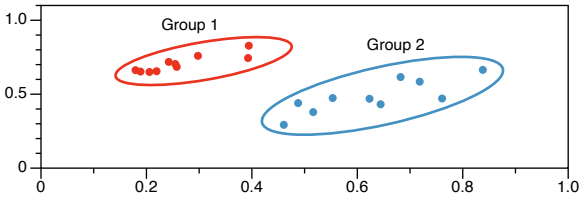


You can resize the region by using the mouse. Click anywhere inside the region to drag it to a new location without changing the width or height. If you click on the boundary you can resize the region. For example, for the 'Rectangle' option if you click close to the upper left corner you change start in x and the end in y.

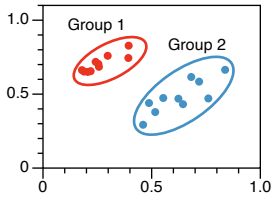
The regions are defined in the x-y coordinate system so you can use them to outline points.



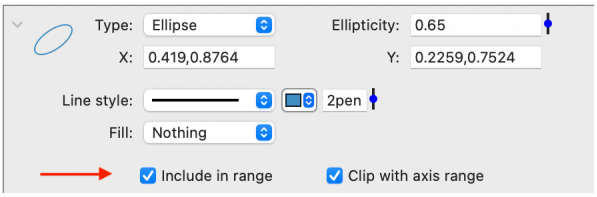
This example shows the “ellipse” option.



If you significantly change your aspect ratio, the region will still be represented appropriately in the x-y coordinate system.

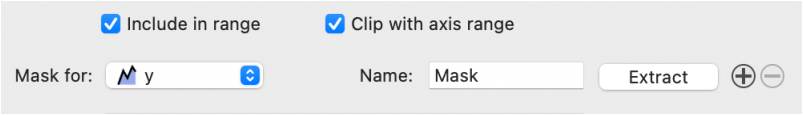


The **Include in range** checkbox determines whether or not the region should be used when computing the bounding box for the axis. If you drag the shape out of current axis range, the extent of the axis will increase when this box is checked. You can uncheck this to use the **Region** command to create a background that extends beyond the boundary of the axis.

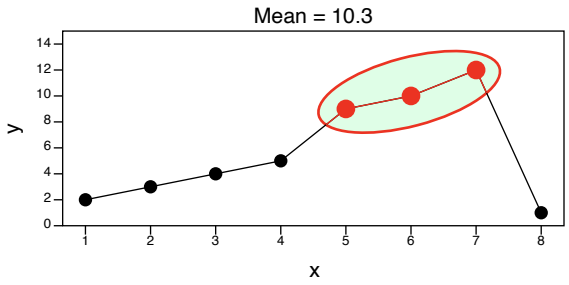


The **Clip with axis range** checkbox specifies whether or not to clip the shape when it does go beyond the boundary of the axis. If you uncheck this, the drawing command can be moved outside the axis extent. This allows you to use the **Region** command as an annotation tool, or to ensure that the label is not clipped by the axis.

The **Mask for** option allows you to extract a column that indicates whether or not a point is inside or outside of the shape. Click the **Extract** button to create the column. Use the "+" button to the right of Extract button to add more commands.



Use this functionality to identify clusters and perform calculations on a subset of values. Works with Point and Plot commands.

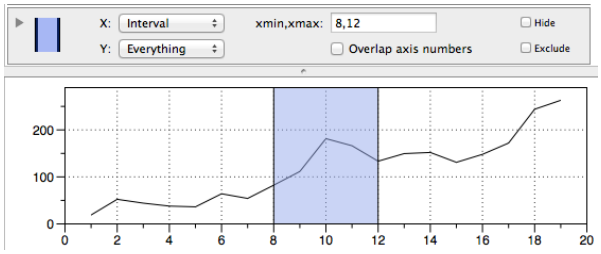


At the bottom of the command is a **Label** text box where you can type a label or use the + symbol to the right of the box to select a token. The **Position** menu lets you specify the label location around the shape or in the center.

Range

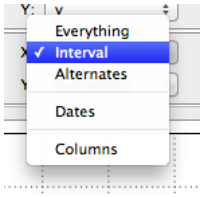
Range can be used to highlight a coordinate region in x and/or y. It is also possible to use it as a basic building block.

In the simplest case you specify a single interval in either the x or y direction.

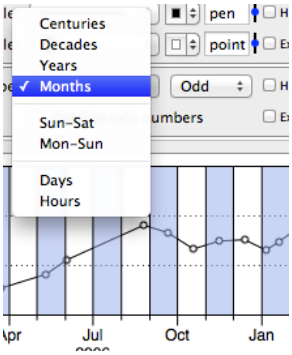


You can drag the edges of the interval with the mouse. As with all numerical fields, you can use variables when specifying the range. You can specify the range in both x and y directions, so you can use this to draw rectangles, and semi- or in-definite rectangles by specifying one or both ends to be infinity.

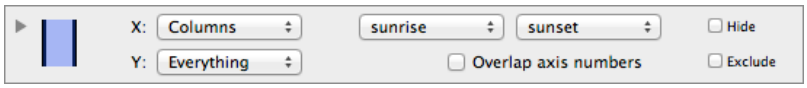
There are also other ways to specify intervals using the menu on the left side. Alternates means that you can color the intervals [0,1], [2,3], [4,5],... without having to use multiple range commands.



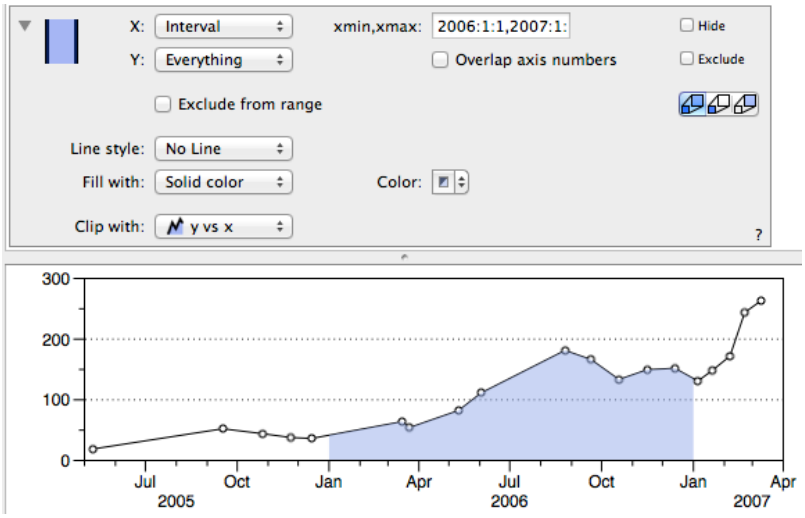
The Dates option allows you to specify alternating fills based on dates.



For full flexibility use the Columns option. This allows you to specify columns for the start and end of each interval. Each row gives you an interval.

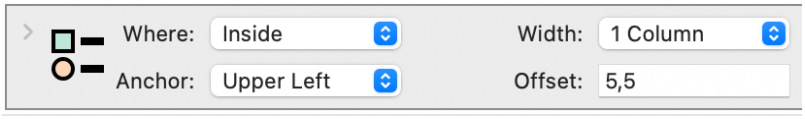


A slightly advanced use, is to use the clipping action to fill a portion of a plot. This is done by using two drawing commands. One draws the plot, the other draws the region but clips the result with the output of the plot command. This is done by selecting the command from the 'Clip with' menu at the bottom of the Range command.

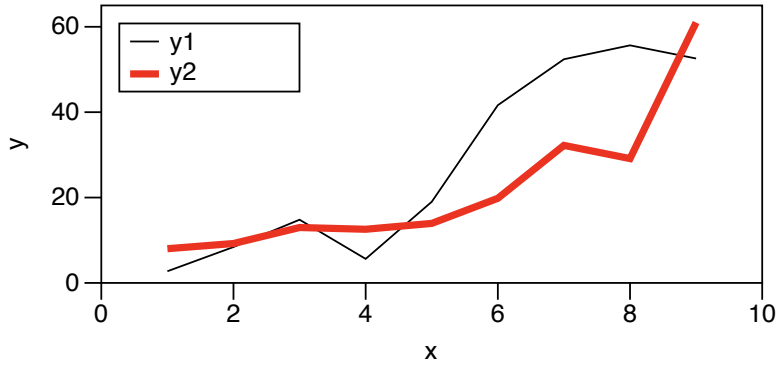


Legend

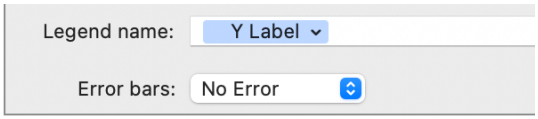
This is the standard way to give a visual key to label lines, fills, etc. Add a **Legend** command from the toolbar. This adds the following object to the list of commands.



Once added, a legend will appear in your graph that shows a key for the commands. For example, here is a graph with two Plot commands.



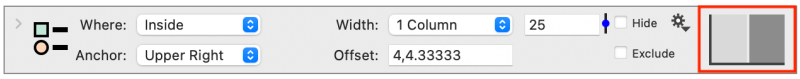
The legend name is specified in the drawing commands, typically at the bottom. For example, here is the **Legend name** entry at the bottom of a **Plot** command.



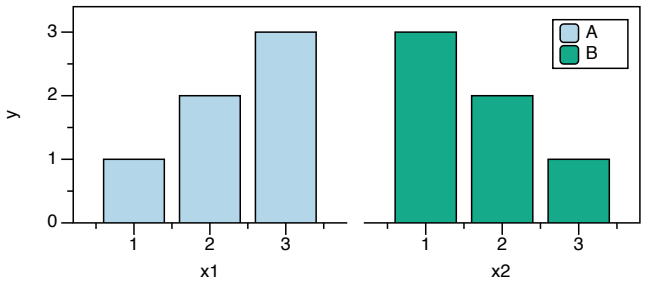
By default, the name that shows up in the legend is based on the blue token that will show the name of the Y column. You can change the legend entry by changing the name of the column, or delete the blue token and type in a different name.

You can adjust the drawing style, how many rows to use and where to position the legend. You can for example put the legend above the graph or to the right. If you have a second legend it will draw exactly the same entries unless you use the **From axis** menu.

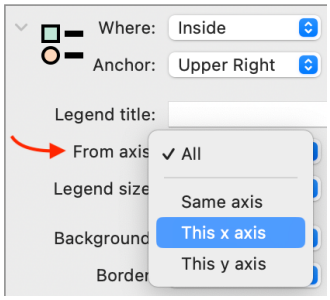
When you have a split axis, use the select boxes on the right to indicate the x or y sub-axis where the legend should be displayed. For example, if you have a split x-axis then clicking the right side places the legend on the second x-axis.



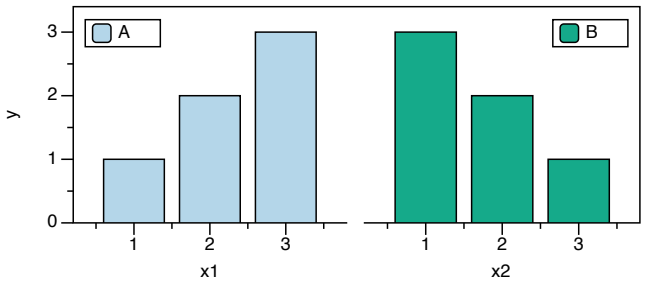
By default, the Legend will show an entry for every command in the graph. So if you have commands in each sub-axis, they will all be shown in one legend.



If instead you want to show multiple legends only include the commands in that axis, you can specify that in the **From axis** menu. In the case of the bar graph above, you would set **From axis** to 'This x axis'.



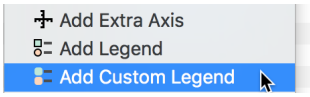
To create the graph as shown below, you would have two **Legend** commands, one for each split-axis.



Custom Legend

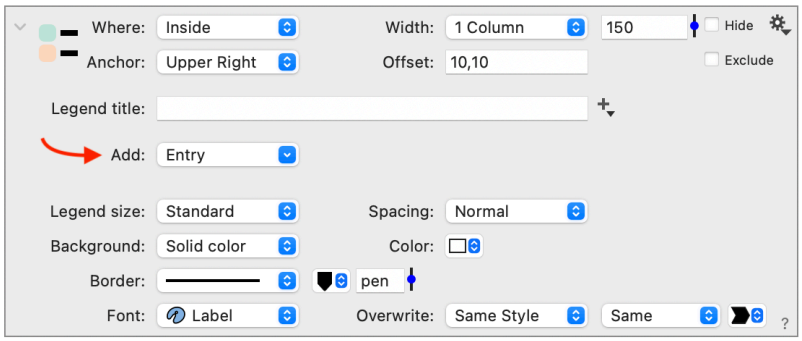
As discussed in the previous section, the **Legend** command uses the information from the drawing commands to automatically create an appropriate legend. In cases where the automatically created legend does not produce the desired effect, you can use the **Custom Legend** command.

The **Custom Legend** is created from the **Label** menu in the toolbar or **Command** menu.



This will add the following command to your graph but the legend itself will be blank since you haven't specified the entries.

To add entries to the custom legend you must open the disclosure triangle on the top left corner to reveal additional settings.



From the **Add** menu. there are six different types of entries that you can add to a custom legend:

1. **Command** - entries based on a specified drawing command
2. **Point** - a point that you can customize
3. **Color** - a color that you can customize
4. **Variable** - the value for a variable including color scheme variables

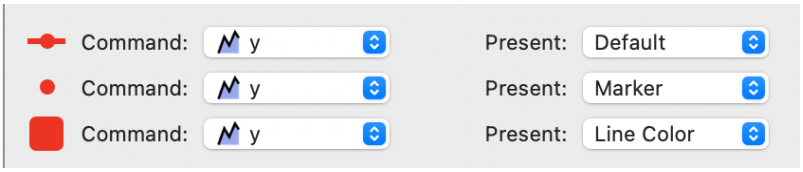
5. **Pattern** - a pattern that you can customize

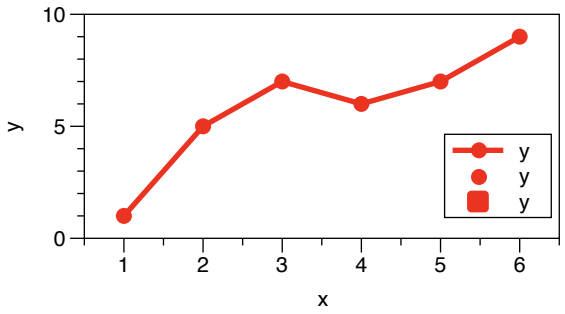
6. **Text** - a label that you can use as a divider between entries.

Each time you add an entry, a line is added that represents that item in the legend.

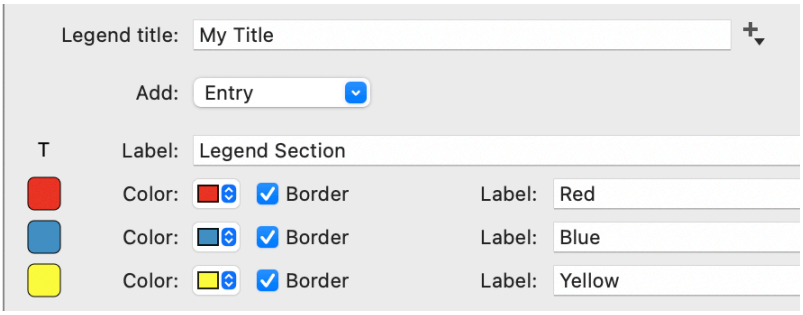
If you select **Command**, an entry is added where you can select from the commands in your graph. The **Present** menu gives you several options for what is displayed.

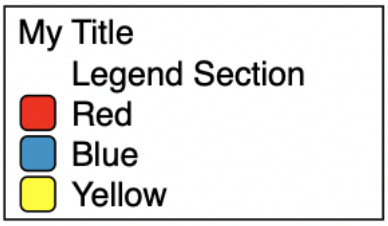
For example, here the same Plot command has been selected three times, but different options are displayed. The default shows the line and marker, but you can show only the marker or line color. The text corresponds to the Legend text entry from the command, similar to the standard **Legend** command.





You can also construct a legend completely from scratch, without referring to any commands. Here one line was added for **Text**, and three lines for **Color**. For the color tiles, you have the option of including a border, which will match the pen color defined in the Style settings.

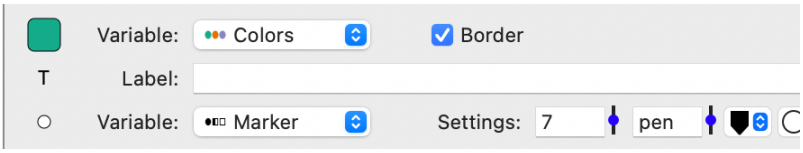




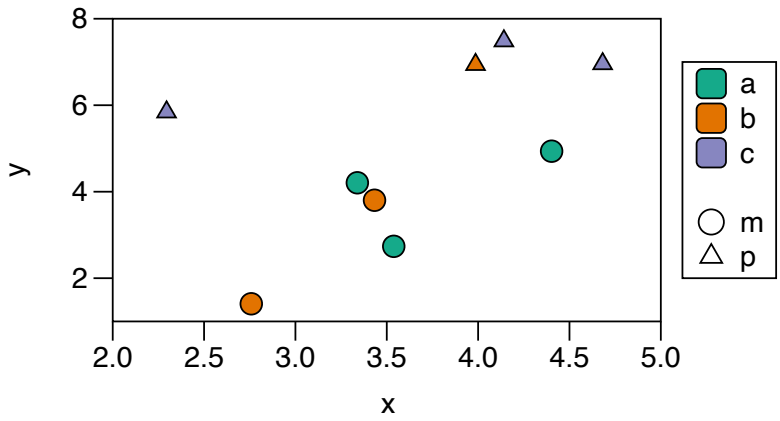
You can reorder the entries by clicking and dragging in the list of entries.

Another use of the **Custom legend** is to display Color scheme and Marker scheme variables, which are not displayed in the standard **Legend** command.

In the example below, three entries have been added: a **Variable** set to a color scheme, a **Text** entry, left blank to leave a space, and a second **Variable** set to a marker scheme.

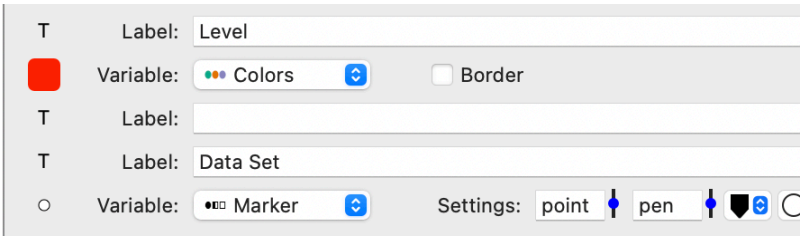


In this example, the color scheme had three colors and the maker scheme had two types.

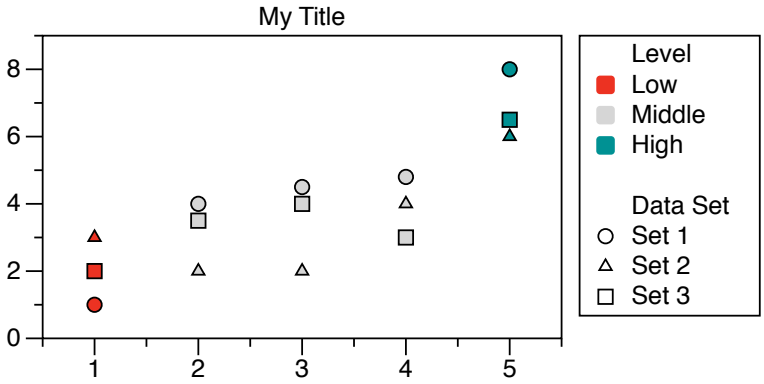


In the example below, five entries have been added to the legend: a **Text** entry that is set to 'Level'; a **Variable** that is set to the variable 'Colors'; a **Text** entry

that is set blank to leave a space; a **Text** entry that is set to 'Data Set'; and a **Variable** that is set to a marker scheme variable called 'Markers'.



The following figure shows the resulting graph, where there are three colors and three markers, coming from the schemes.

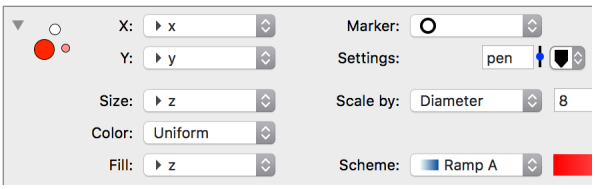


Color Ramp Legend

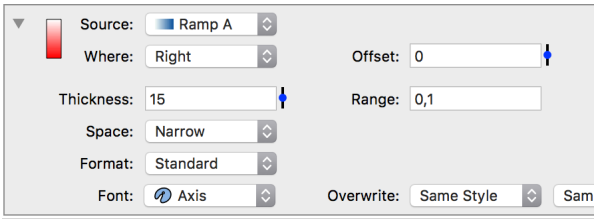
The **Color Ramp Legend** command allows you to display a color ramp on your figure.

Any command that allows a color scheme can also use a color ramp. The color ramp allows you to create a continuous color scheme, rather than specifying discrete colors for a given value or range of values.

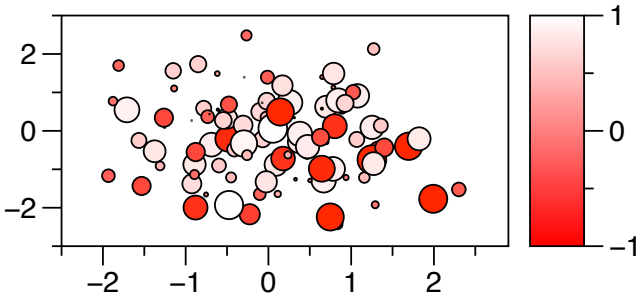
Here a **Color ramp** called 'Ramp A' was selected for the **Fill** color in this **Points** command.



To display the color ramp on the graph, create a **Color Ramp Legend** command and select the color ramp you want to display from the **Source** menu.



By default, the color ramp will be displayed to the right of the graph, but you can modify the location using the **Where** menu.



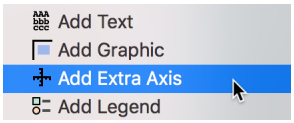
Extra Axis

By default, graphs in DataGraph contain an x and y-axis with tick marks and ranges that are automatically generated based on the data selected in the drawing command.

The **Extra Axis** command allows you to customize the look and location of the axis, to add a second axis, or to show the data in different units without having to modify your data.

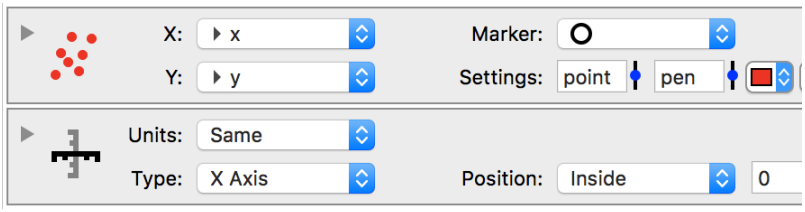
Extra Axis Basics

You can add an extra axis drawing command using the Command Menu.

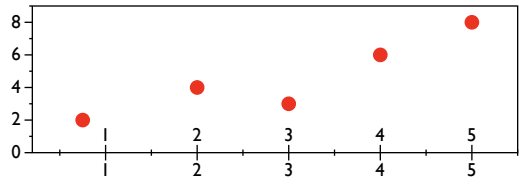


This will add the following command to your list of drawing commands, where the axis is added to inside the x-axis by default.

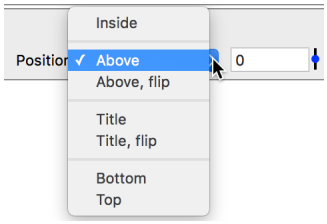
For example, let's say we added an extra axis to a graph containing a points command. Note how the **Type** drop-down box is set to 'X axis' and the **Position** is set to 'Inside'.



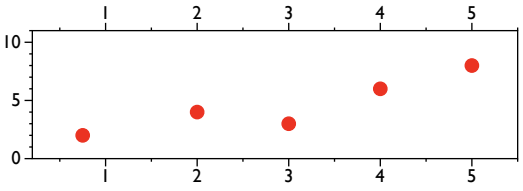
The resulting graph is shown below, where we now have two x-axis, the default axis and the extra axis.



You can change the location of the axis using the **Position** drop-down box.



Now the extra axis is above the graph.



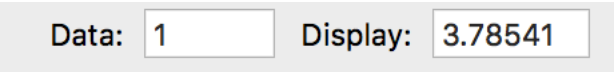
You can move the extra axis to the y-axis by using the **Type** drop-down box.

Unit Conversions

You can also use the extra axis as a way to display your data in different units. DataGraph contains a large number of built-in units to allow you to convert between units symbolically. The full list is included in the appendix.

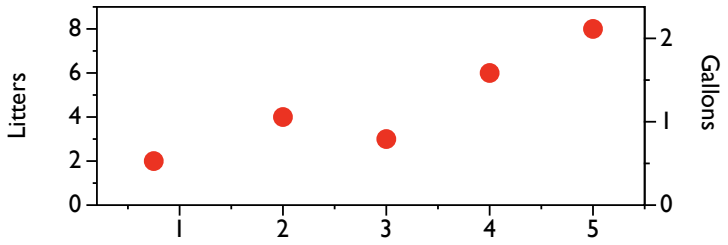
For example, say your data is in liters but you also want to display it in gallons. First, change the **Units** drop-down box to 'Convert'. Next, enter the symbols for the units of measure in the **Data** and **Display** entry boxes that are now visible. In **Data**, enter the units of the data as given in the data table. In **Display**, enter the units you would like to shown in the extra axis.

In addition, you can also enter your own values directly into the **Data** and **Display** entry boxes. For example, this combination shown below would also result in a conversion from liters to gallons since there are 3.78541 liters per gallon. The axis is be scaled by multiplying by Data/Display, which in this case is 1 divided by 3.78541.

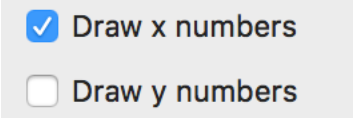


Extra Axis Placement

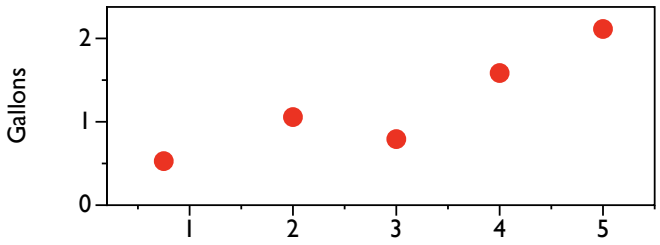
The extra axis can be placed anywhere on the drawing window. In the example below, the extra axis is on the right and the axis labels are automatically updated to show the units based on the **Data** and **Display** settings.



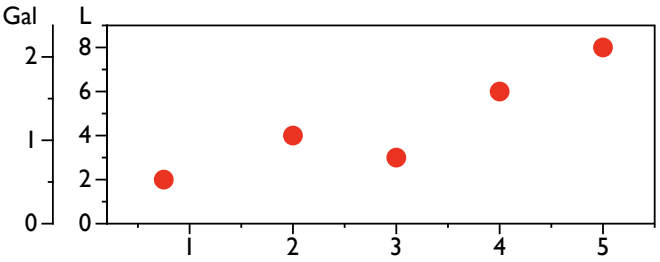
You can also use the extra axis to replace the default scale on the x or y-axis. In the *Axis* commands, you can hide the default axis by unchecking **Draw x/y numbers**.



Then you can move the extra axis to the desired location as shown below for the y-axis.



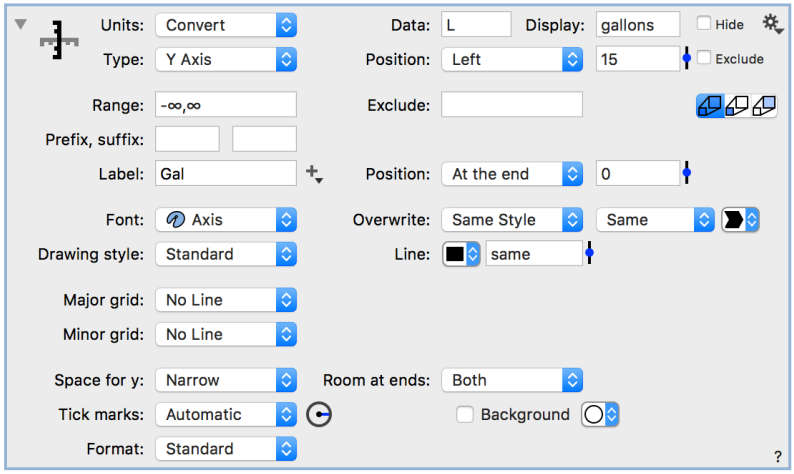
You can also set up multiple scales on the same axis using the position offset. In the following example, both y-axes are set up as an Extra axis, then the labels can be easily added above the axis as discussed further in the next section.



Labels, Range, and More

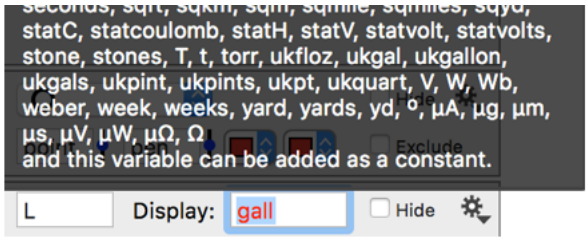
By opening the disclosure triangle on the left of the drawing command, you can reveal commands to control almost every aspect of how the extra axis is displayed. This includes adding labels, specifying the range, and changing the location of tic marks.

Note that the labels for the y-axis are added above the axis as shown in the last example plot from the last section, where both y-axes are set up using an extra axis.



Symbolic Constants

DataGraph includes symbolic constants for converting the units in the **Extra Axis** command. Some of the constants have more than one symbol defined. For example, gallon can be understood as ‘gal’, ‘gallon’, or ‘gallons’. If you enter a symbol that is not defined a warning is displayed, listing all of the symbolic constants that have been defined.



You can think of each of these constants as built-in variables that represent a conversion factor based on SI units. For example, the value for meters 'm' is one since meters is the SI unit for length. The value for ‘cm’ is 0.01, one-one hundredth of the meter. The conversion of the data is done by multiplying by the ratio of Data/Display to show the requested units.

Note: DataGraph will not warn you if you pick a combination of units that don’t make physical sense (i.e., converting from volume to mass).

For more information see the Defined Constants section in the Appendix.

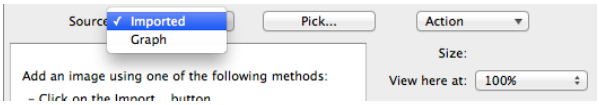
Graphic

The **Graphic** command allows you to add a graphic to the figure. The graphic can either be pasted into a graph window, or can come from another graph. Once

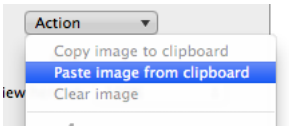
the graph is inside the command you can position and scale it. Some examples of what you can use this command for are:

- Add a logo in a corner of a graph
- Add an equation that is done in LaTeXiT or some other equation editor and pasted in. DataGraph does support expressions, but doesn't cover as much as TeX does.
- Stack graphs into a figure. Each graph can be done in a separate graph in DataGraph and remain dynamic, or just pasted.
- Scale bitmaps. Applications like TextEdit don't have a lot of control over the output size, and DataGraph can be used to scale a bitmap to an exact size.

As shown in the image below, the **Source** menu provides two ways to specify a graphic. The 'Imported' option means that the image needs to be copied from a file or come from the clipboard. The 'Graph' option is for specifying a graph in the same DataGraph file.



If the clipboard contains a figure the paste option is enabled int the action drop-down box.



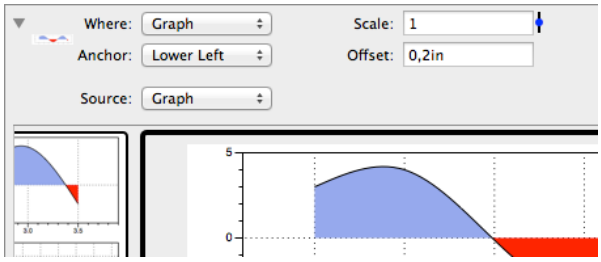
If you select this option, the clipboard contents overwrites the current figure content. Note that if you have a figure in the clipboard and paste it, DataGraph creates a **Graphic** command for it. This menu also allows you copy the graphic back into the clipboard.

This is useful since DataGraph retains any LinkBack (http://linkbackproject.org) information so for some applications such as LaTeXiT you can paste it back into the application and continue to edit it.

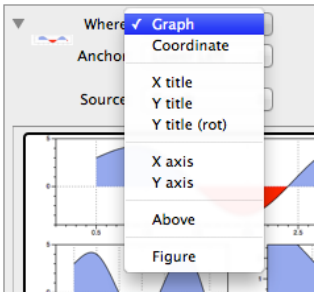
If you have a graphic file, you can click on the **Pick...** button and select it, or drag the file to the command list to create a Graphic command for that file.

Creating multiple graphs is explained in the next chapter, but the short version is that a single DataGraph file can have multiple graphs that use the same data set.

The graphs show up at the top of the file window, and if you select Graph from the Source menu, you see the same list of miniatures inside the Graphic command. Click on a graph to select it. You cannot select the current graph or any graph that directly or indirectly depends on this graph since that would cause an infinite recursion.



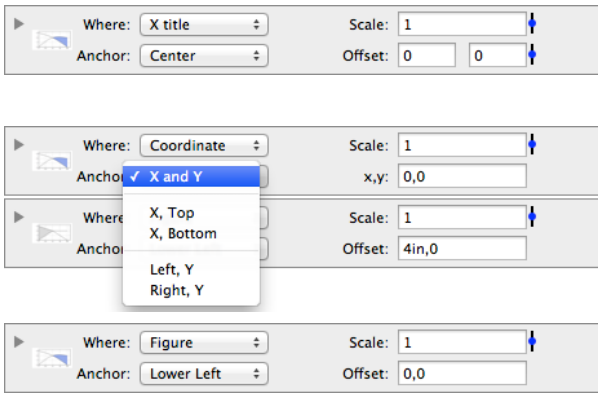
The graphic can be positioned relative to different parts of the graph, just like the **Label** and **Text** commands. The **Graph** command positions the graphic relative to one of the corners of the current axis. The Coordinate positions the graph at a particular x,y location and this means that the graph is positioned at a particular coordinate in space, and if you crop the axes it might not be visible.



You can also position the graph in the title or along the axes. For example, if you want a graphical label instead of a text label. The ‘Figure’ option means that the placement is relative to the overall figure, and therefore not dependent on an axis.

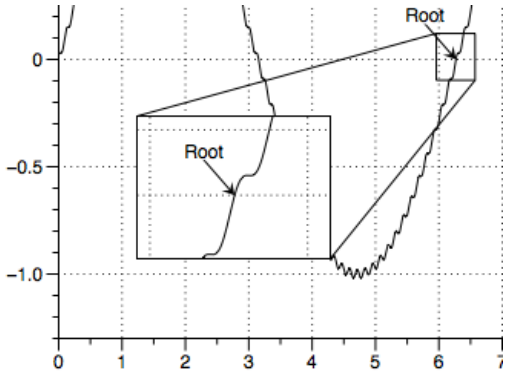
The option you pick really should depend on what should happen as the data changes and you resize the graph. The second line will change depending on what is selected in the Where menu.

For example, for the ‘Graph’ option you can pick which corner the graphic should be anchored to and what the offset should be. When you pick the x title, you can specify the x coordinate and the offset in the y direction.



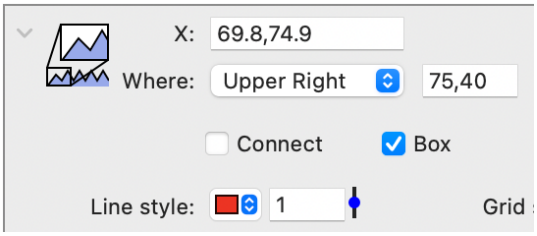
Magnify

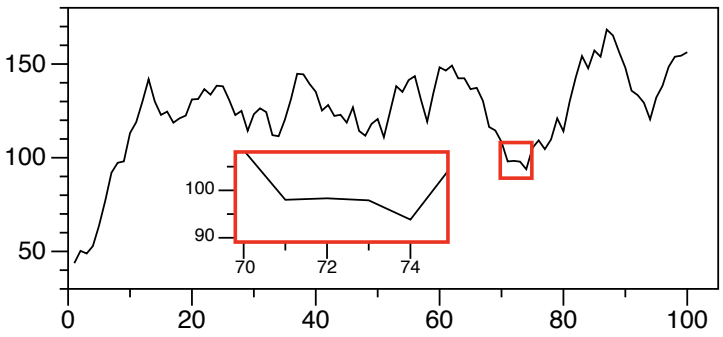
The loupe tool allows you to inspect details of the graph without changing the axis ranges. The Magnify command allows you to include that type of view in the output.



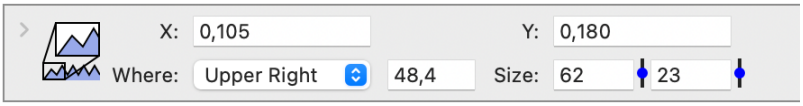
Drag the edges of the magnified area to change where you are zooming into. Drag the edges of output area, to grow or shrink the inset graph.

There are many options for how to draw inset. For example, you can turn off the connect lines or color the box.

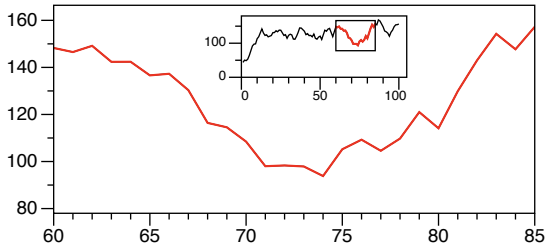




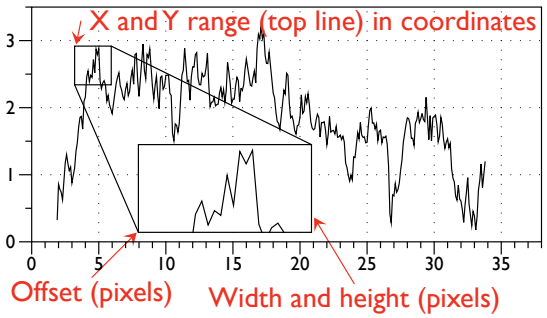
You can also have the inset show the entire range of the data, and instead, have the main axis magnify the data. For example, here the X and Y range has been set to the entire data set.



The inset now shows the specified range and the graph is the data highlighted in the inset.



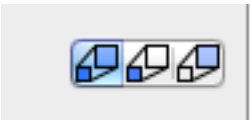
By default, the X/Y range is inside the current axis bounds. Lines are used to connect these areas visually. When you are magnifying a region, you can change the X/Y ranges using the mouse by dragging the source rectangle.



You change the position and size of the inset by dragging it around using the mouse or by adjusting the numerical fields for size. The size is in pixels, but you can use units such as in, cm, mm. The anchor is one of the four corners of the axis, just like the Graphic and Text command.

You can adjust the drawing style inside the drawing command. For example, you can adjust whether or not you want to draw the box, connecting line, tick marks, etc.

Most commands have a small magnify selector in the top right corner of the detailed view.



This selects how the command interacts with the magnify command. The entry on the far left, where the destination and source are both solid blue means that this command is drawn both in the standard view and magnified view. The next means that it is not drawn in the magnified view, i.e., it is not magnified. The one on the right where only the magnified view is solid blue means that the command is not drawn unless the area is magnified.

10. Export

Exporting data

In addition to exporting data graphically as a figure or a movie, DataGraph can also export the values from the data table. Copying data from the table is the simplest way to get data from DataGraph for use in other program; however, this assumes that the external program accepts data through the clipboard and that the data is not too large. Some programs might not handle a few tens of megabytes of data through the clipboard. For that purpose, you can use the **Export Data** menu entry in the **File** menu.

The **Export Data** method will export only the data displayed in the data table. If you select specific rows or columns, only the selected data is exported. In each case, the column headers are included in the first row.

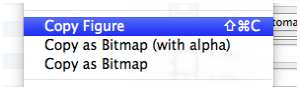
To begin the export, select **Export Data** from the **File** menu. You can export the data as text, either tab delimited or comma separated. You can also export the data as an Excel document (xlsx). Any notes you have in your DataGraph file will be placed in a separate Note sheet in Excel.

Data is exported in the background, so you can work on other DataGraph documents in the meantime. A progress bar will show the status of the export; however, this operation is typically very fast, so unless your data set is a few hundred megabytes or saved on a slow disk, you will not see a progress bar.

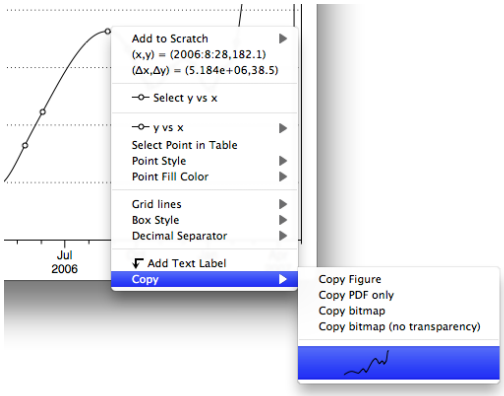
Exporting figures

Copy

In the edit menu you can select from three copy options. The top option copies the figure as a pdf and bitmap, allowing the program you paste into to pick the preferred format. Programs such as KeyNote will use the pdf version.



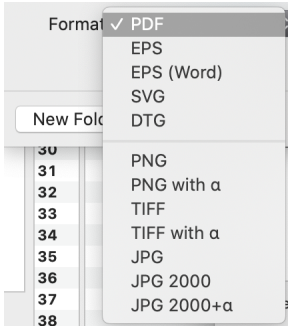
You can also copy the figure as bitmap only, with or without transparency. You can also use the context menu (right click/control-click) to copy the figure. You can also copy a particular drawing element from the graph.



Export to a file

In the **File** menu, you can export the graph as a file.

You can export the graph using multiple formats.



The first section of output formats are all vector graphics (PDF, EPS, SVG, DTG). The second section of output formats are bitmap or pixel based formats (PNG, TIFF, JPG).

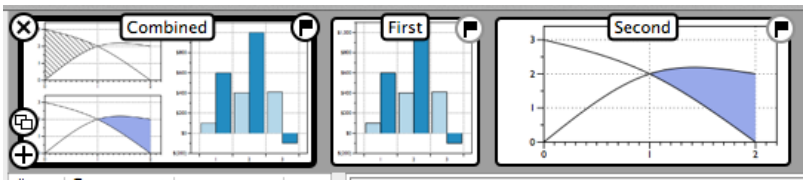
For bitmaps you can also specify the resolution. The default (screen resolution) is 72dpi. Note that the EPS file format does not support semi-transparent objects. What DataGraph does is to automatically replace the transparent color by an opaque one that will show up the same based on a white background. PDF does support transparent fills, but if you are printing on laser printers, they will typically draw that region using a low resolution bitmap so quality will suffer. PDF does not however support semi-transparent gradients, and DataGraph automatically adjusts the gradient colors so that they will look the same on a white background. This is done since otherwise the EPS/PDF file that gets generated cannot be opened.

The SVG format is a vector graphics format that is intended for web pages, and is an xml file. The limitation here is that you cannot use an external figure (using the Graphic command) or a textured fill since that is implemented with a bitmap. DTG is a graphic file that is unique to DataGraph and DataTank. This format keeps all of the information, since it is just a file version of the graphic that DataGraph uses internally. The main use of this option is if you want to save a graphic that you intend to include in a different DataGraph file.

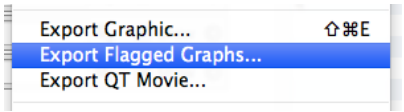
The bitmaps don't have any of these limitations, but for high resolutions they will take a lot more disk space than the vector graphic version. PNG and TIFF can be semi-transparent, which is typically called an alpha channel. Note that even though Mac applications tend to all handle alpha channels properly, some applications do not.

Exporting multiple graphs

The standard **Export graph** menu only exports the selected graph. If you name and flag the graphs you can then export all of them at the same time by using the 'Export Flagged Graphs...' option.

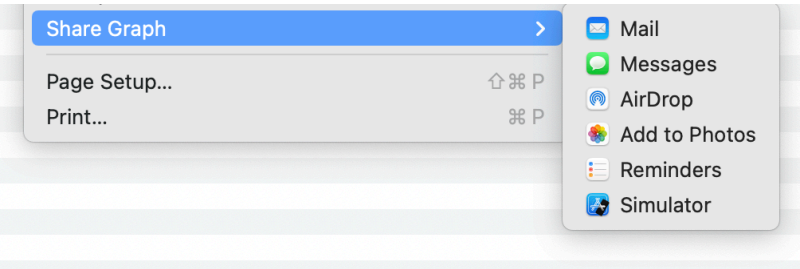


This brings up a dialog box to ask you for the folder where the images should be saved.



Sharing Figures

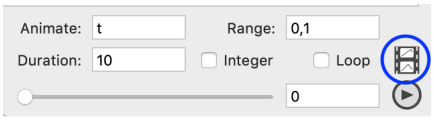
You can easily share graphics via Mail or several social media web sites. These options are available under the **File > Share Graph**. First you must have the graph you would like to share as the selected graph. Also, you must already have an account configured on sites such as Twitter for this to work. These are configured in your overall System preferences under Internet accounts.



Movies

You can set up animations in DataGraph to create MP4 files. This is done through the animation variable called 't'. You can change the name of the variable and use it in any numerical field such as cropping regions, line thicknesses, masking rules, etc.

The variable is a slider, so you can set the range and restrict the value to integers. Then you set the duration in seconds and can click the small play button to animate the variable on screen. Once you have set up an animation that you like, click the small Film button circled or use the File menu.



This will open the movie interface. You can create movies in retina format. Click the "Create MP4" button, to create the movie file. While this is running you can continue to use DataGraph.

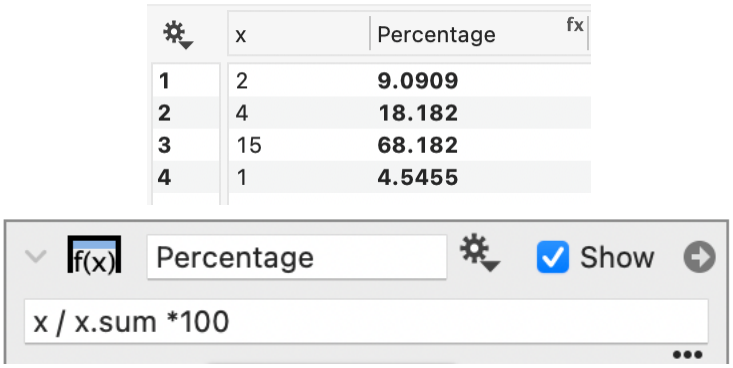
Appendix

Column Properties

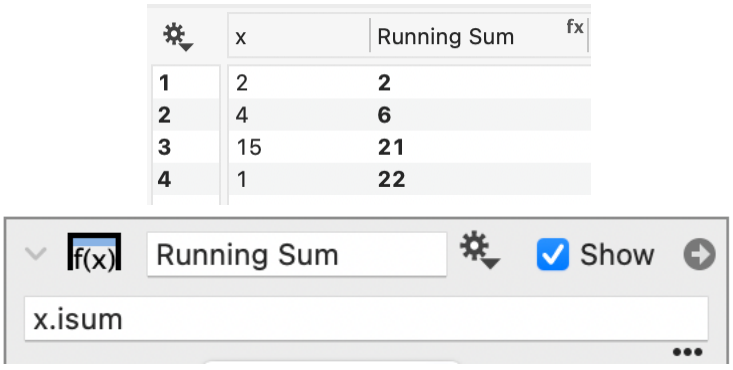
Column properties are a handy way to build mathematical expressions using a range of numbers, specifically a column of values.

To refer to them, type the name of the column followed by a '.' and the property name.

For example, '*name*.sum' returns a single value, the sum of a column. In the table below, the column **Percentage** is calculated using the expression: x / x.sum * 100. In this case, x.sum = 22.



Other column properties return a column of values. For example. '*name*.isum' returns to the running sum of a column. Enter the column property alone in a expression column (as shown below) or in more complicated expressions/ calculations.



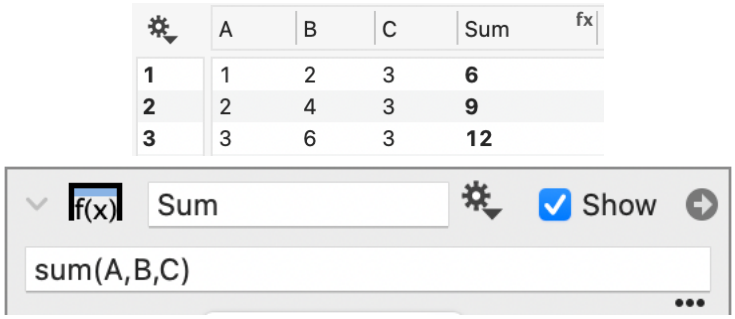
For a full list of properties visit the on-line List of Properties.

Function Reference

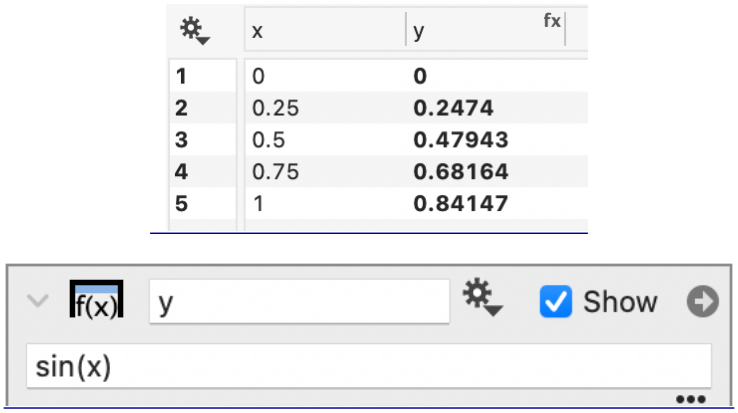
Functions can be used anywhere in DataGraph that accepts numerical values. Use them in Expression Variables or Columns. Function input can be numbers, variables, or column names.

Some functions accept multiple inputs (i.e., a list of numbers). For example, the sum function can accept multiple inputs with commas in between such that: sum(1,2,3) = 6.

These functions are useful for row operations across multiple columns. For example, the row by row sum of three columns A, B, and C can be computed as: sum(A,B,C).



Other functions have a specified number of input arguments. For example, most of the standard functions have one input and one output, sin(0.5) = 0.47943. If a column with multiple entries is input, then the output will be computed on a row by row basis.



There are several types of functions in DataGraph including:

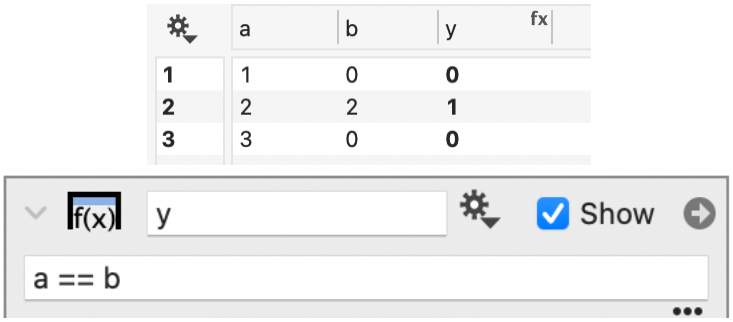
- List of Inputs
- Standard Functions
- Integers & Rounding
- Logical Operations
- Special Functions
- Piecewise Functions
- Ternary Functions
- Date & Time Functions
- Statistical Functions

For a full list of Functions visit the on-line Function Reference.

Mathematical Operators

The standard operators such as +,-,*,/ are all defined as well as the following operators.

The logical operators can be used with the if function or all alone. For example, the double equal sign (a == b) tests when two values are equal. These comparisons return 1 when true and 0 when false. This is a quick way to compare values in columns.



For a full list of Operators visit:

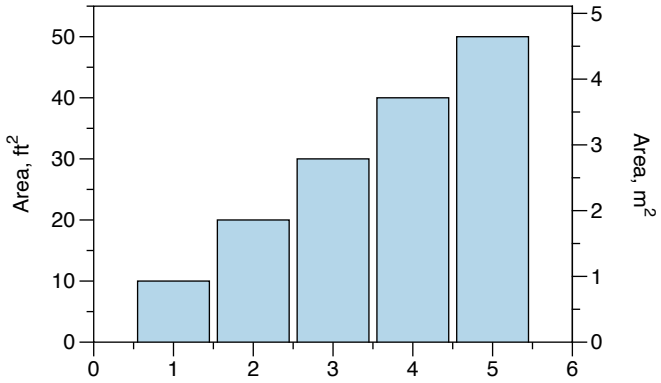
https://community.visualdatatools.com/datagraph/knowledge-base/operators/

Defined Constants

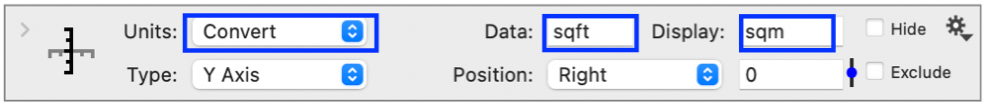
DataGraph has over 100 defined constants that can be used to convert axis values from one unit of measure to another, without you having to enter any conversion factors. They include, length, area, volume, time, and more.

These constants are used in the unit conversion in the **Extra Axis** command. To use this functionality, add an **Extra Axis** command to your graph and change the **Units** to 'Convert'. Then you will have two new entry boxes named **Data** and **Display**.

One use case for this functionality is to provide two units of measure for a set of data. For example, below we are graphing data in feet square, but we also want to show meters square on the right.



To create this graph, the **Extra Axis** options are **Units** = 'Convert' and the **Position** = 'Right'. The defined constants are entered such that **Data** is set to 'sqft' and **Display** is set to 'sqm'.



Note that it's up to you to make sure the combination of units you enter makes sense, so always double check!

For more information, see the **Extra Axis/Symbolic Constants** section of the **Drawing Commands** Chapter.

Command Line

DataGraph works with a command line utility called dgraph that allows you to generate graphs. This is intended for automation and shell scripts and assumes some knowledge of unix commands and working in the terminal.

If you are using the Mac App store version you need to download dgraph separately, since it cannot be included in the application. To download it, use the following command:

```
curl https://www.visualdatatools.com/DataGraph/dgraph -o dgraph
```

If you are using the version downloaded from the Visual Data Tools web site, you can use the above command or just get to it inside the DataGraph application wrapper. Go to the DataGraph.app/Contents/Library/ subfolder.

If you run the dgraph utility that is inside the application wrapper, it will launch that application. If the dgraph utility is outside the application wrapper it assumes that the DataGraph application is inside the /Applications folder.

Running dgraph

There are three crucial parts to the argument list. The first is the DataGraph file that you want to use as a template. The second is the data file that you want to import into the table. The third is the output that you want to create.

```
dgraph action.dgraph file.txt out.pdf
```

Note that typically one or more of those files will have full path names, so the argument list will be a lot longer.

What this does is to open action.dgraph and then overwrite the table with file.txt. This is done non-graphically, but what happens is exactly the same as if you select all of the visible columns in the table and then import file.txt. The output graph is saved in out.pdf. Only the current figure (the one selected when you saved the DataGraph file) is saved.

You can overwrite variables, change sizes, etc by using command line arguments. To get further explanation, use the -help flag.

```
dgraph -help
```

How dgraph works

What happens when you use the dgraph application is that the underlying DataGraph application is launched with the same arguments. It will not open any windows, and it doesn't matter if you already have a copy of DataGraph running. When it is finished this instance of DataGraph quits.

For The Mac App Store version

The Mac App store version is pretty much identical to the version downloaded from the web site, except that the Mac App store version is running in a sandbox (google Mac OS Sandbox for more detail). This means that access to the file system is severely restricted and is limited to files and folders that you either drag in or open directly. Because the command line utility refers to files by path names this leads to a problem. But starting in DataGraph 4.4 there is a workaround.

If you go into the Preferences panel in DataGraph there is a way to explicitly add read or read/write access to particular directories or files. This has to be done once and is then saved in the preferences. You can pick what folders you give access to. The non Mac App Store version has access to every file you can navigate to in the Finder, and you can of course do that here as well. But a more common approach would be to allow access to a more limited portion of your home directory.