ggplot2 is a package that allows you to make pretty plots beyond the ordinary **plot, boxplot**, etc.

Installing Packages in R

Previously, all the functions we've used in R have been part of the base package (i.e. they are already loaded in R). As you progress to doing "fancier" stuff, you will need to install and load packages from outside sources. This is done simply by using the command **install.package("nameofpackage")**. Let's try installing "ggplot2". //
It will require you to choose a CRAN mirror. I usually just choose one of the USA options (but I don't think it really matters). The package should then install automatically. //
To load the library into R to begin using it, type **library("package name")**.
Also helpful:

- To check which packages have been installed, type **installed.packages()**. This will also tell you under which version of the package is installed (some packages are periodically updated.) You can also look at the "packages" window in RStudio (probably the same window as files, plots, and help unless you've modified your panes.) Here is a list of "popular" packages. If you check the box, this will also load the library into your console for use.

ggplot2

I will go through a few examples of the flexibility of **ggplot2** with you. However, I encourage you to explore on your own: `http://docs.ggplot2.org/current/` does a great job of going through examples and the resulting graphs. Try to make some pretty stuff on your own!

One note before beginning is that **ggplot2** only works with **data frames**. Remember that you can transform a matrix into a dataframe simply by using the command **df.name = data.frame(matrix.name)**. Don't forget to name the variables using the **names(df.name) = c("var1", ...)** command.

qplot()

The most straightforward option in **ggplot2** is to use a "quick plot" i.e. **qplot()**. This allows you to make x-y plots, histograms, and boxplots depending on the argument. The "geom" argument will specify what type of "geometric" object will be plotted. See the following examples

- **x-y plot** - this is the default. The basic command is **qplot(yvar, xvar, data =)**. The default is to display points, but you can change it to lines using the argument **geom = "line"**
- **boxplot - qplot(x (box type), yvar, data = nameofdata, geom ="boxplot")**
- **histogram - qplot(yvar, data = nameofdata, geom = "histogram")**. You can specify "breaks" just like the **hist** function or alternatively use the "binwidth = " argument
- **bar chart - qplot(yvar, data = nameofdata, geom = "bar", weight = xvar)**

Great. So now you can do all the stuff you knew how to do with separate commands in R.

Using the "color" and "shape" options

One way **qplot** can make your life easier is by using the **color** and/or **shape** options. This will automatically plot the data using different color or shape options and create a legend. Therefore, you can avoid using the **points()** function over and over. See "ggplot2_ex.R" for an example.

Arguments that are the same as plot()

Certain arguments are the same in both **plot()** and **qplot()**. These include xlim, ylim, xlab, ylab, log, and main.

Faceting

One nice feature of **ggplot2** is that it can automatically create "facetted" plots, i.e. create a plot for each factor simultaneously.

Layering plots

ggplot2 also allows you to "build up" a plot layer by layer. This allows for more plot flexibility than **qplot()**.

You begin by creating a graphical object and assigning it to a variable, e.g. **p = ggplot(sm.diamonds, aes(x = carat, y = price, color = cut))**. **aes** stands for "aesthetics" and you can assign x, y, color, shape and size. Note that if you type "p", nothing happens. You need to add layers using the **layer** command. The basic syntax for layering a plot is **p + layer(...) + ...**.

Layer options:

- **geom** - geometric object like "line", "histogram" etc.
- **geom_params** - geometric parameters
- **stat** - statistical function to apply (e.g. lm object to plot the fit for a linear model)

The best way to understand layering is to look at examples. See the "ggplot2_ex.R" code and `http://docs.ggplot2.org/current/index.html`
I believe the following sections on that website are especially helpful. Take a look through them in your spare time.

- Geoms - geom_abline, geom_blank, geom_boxplot, geom_dotplot, geom_histogram, geom_line, geom_point, geom_text, geom_hline, geom_vline
- Statistics - stat_bin, stat_boxplot, stat_function, stat_quantile
- Scales - guides, guide_legend, scale_color_gradient, scale_linetype
- Themes - theme, theme_bw
- Aesthetics - aes, aes_auto, aes_group_order