**Getting Started Notes**

**Installing R:**

**For Windows :**

1. Download the binary setup file for R from the following link.( [R for Windows](http://cran.r-project.org/bin/windows/base/R-3.1.2-win.exe) )
2. Open the downloaded .exe file and Install R

**For Mac :**

1. Download the appropriate version of .pkg file form the following link. ( [R for Mac](http://cran.r-project.org/bin/macosx/) )
2. Open the downloaded .pkg file and Install R

**For Linux :**

1. For complete R System installation in Linux, follow the instructions on the following link ( [Link](http://cran.r-project.org/bin/linux/ubuntu/README) )
2. For Ubuntu with Apt-get installed, execute sudo apt-get install r-base in terminal.

**Installing Rstudio:**

On the following link [Download R Studio](http://www.rstudio.com/products/rstudio/download/) choose the appropriate installer file for your operating system, download it and then run it to install R-studio.

**Installing Packages:**

Many useful R function come in packages, free libraries of code written by R's active user community. To install an R package, open an R session and type at the command line

install.packages("<the package's name>")

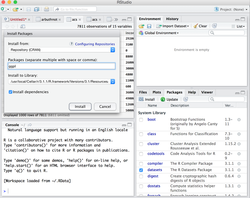
R will download the package from CRAN, so you'll need to be connected to the internet. Once you have a package installed, you can make its contents available to use in your current R session by running

library("<the package's name>")

There are thousands of helpful R packages for you to use, but navigating them all can be a challenge. In each lab, I’ll curate a list of helpful R packages. But if you find one you think is better, feel free to share on our R Google Doc!

Installing Packages in Rstudio

1. Run R studio
2. Click on the Packages tab in the bottom-right section and then click on install. The following dialog box will appear



1. In the Install Packages dialog, write the package name you want to install under the Packages field and then click install. This will install the package you searched for or give you a list of matching package based on your package text.

Once a package has been installed, you then must load it each session by using the library(*package*) command.

**Finding/Downloading Data**

One of the hardest things to do in data analysis is accessing data. We will be exploring different ways to find usable data in this course. I’ve created a page in our class to link to data sources.

**Prepping Your Data Set**

Before you start thinking about how to load your Excel files and spreadsheets into R, you need to first make sure that your data is well prepared to be imported. If you would neglect to do this, you might experience problems when using the R functions

Here’s a list of some best practices to help you to avoid any issues with reading your Excel files and spreadsheets into R:

* The first row of the spreadsheet is usually reserved for the header, while the first column is used to identify the sampling unit;
* Avoid names, values or fields with blank spaces, otherwise each word will be interpreted as a separate variable, resulting in errors that are related to the number of elements per line in your data set;
* If you want to concatenate words, do this by inserting a .. For example:

Sepal.Length

* Short names are preferred over longer names;
* Try to avoid using names that contain symbols such as ?, $,%, ^, &, \*, (, ),-,#, ?,,,<,>, /, |, \, [ ,] ,{, and };
* Delete any comments that you have made in your Excel file to avoid extra columns or NA's to be added to your file; and
* Make sure that any missing values in your data set are indicated with NA.
* Use a separate spreadsheet to manipulate data, in order to keep your master copy intact.

With larger datasets this can be difficult to do in Excel, so there are other tools for cleaning the data within R that we will address in other classes.

**Loading Data into R**

Once you have your dataset saved in Excel, you still need to set your working directory in R.

To do this, try to find out first where your working directory is set at this moment:

getwd()

Then, it could be that you want to change the path that is returned in such a way that it includes the folder where you have stored your dataset:

setwd("<location of your dataset>")

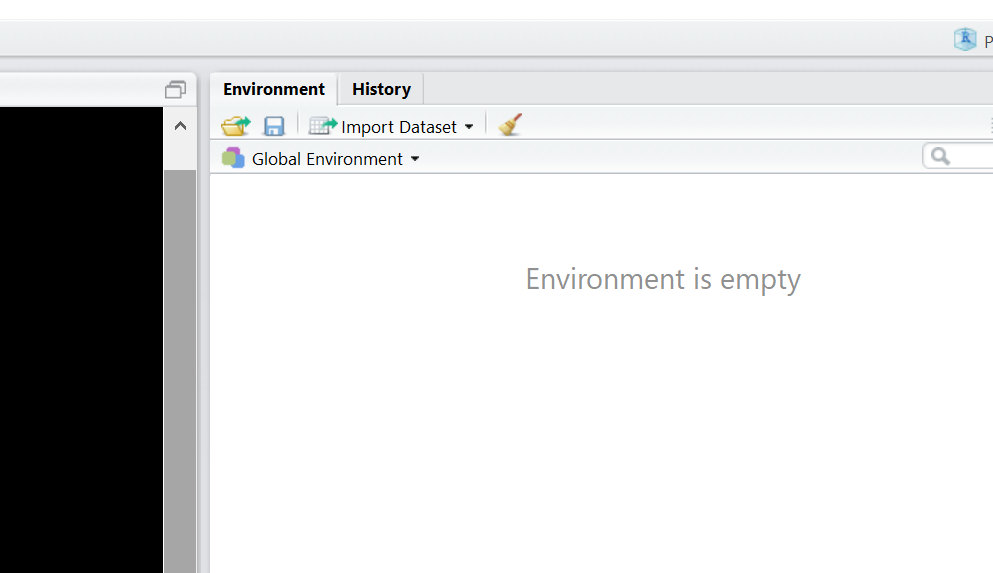
[RODBC](https://cran.rstudio.com/web/packages/RODBC), [RMySQL](http://www.rdocumentation.org/packages/RMySQL/functions/RMySQL-package), [RPostgresSQL](http://www.rdocumentation.org/packages/RPostgresSQL), [RSQLite](http://www.rdocumentation.org/packages/RSQLite) - If you'd like to read in data from a database, these packages are a good place to start. Choose the package that fits your type of database.

[XLConnect](https://cran.rstudio.com/web/packages/XLConnect), [xlsx](https://cran.rstudio.com/web/packages/xlsx) - These packages help you read and write Microsoft Excel files from R. You can also just export your spreadsheets from Excel as .csv's. The xlsx package is for Excel 2007 and earlier.

[foreign](http://www.rdocumentation.org/packages/foreign) - Want to read a SAS data set into R? Or an SPSS data set? Foreign provides functions that help you load data files from other programs into R.

R can handle plain text files – no package required. Just use the functions read.csv, read.table, and read.fwf. If you have even more exotic data, consult the CRAN [guide](https://cran.rstudio.com/doc/manuals/R-data.pdf) to data import and export.

If you are using RStudio, once these pacakages are installed, you can use a drop down menu to load the data:



In the preview menu, you can also update the type of data (numeric, character, etc) as well as determine if you have prepped the data appropriately.

**Basic Statistical Operations**

*Statistical Summaries*

summary(*data*) will provide a basic summary of all your data

*Plotting/Charts*

The ggplot2 package has a remarkable suite of tools for creating graphical displays of your information. Here is a great link on how to use this tool. <http://ggplot2.tidyverse.org/>

You will be best served by trying to figure out how to use the R commands by yourself first. I will provide a step by step key if you can’t figure something out, but learning by doing is the ideal.

**R Hints:**

Rename your variables to make them easier to use in a function

**Updates**:

Here are some of the functions needed to complete the getting started lab:

To rename a variable (to make it easier to use in a function)

NewVariableName <-dataset$oldvariablename

To parse the dataset by some attribute (such as size)

dataset$Colorvariable <-cut(dataset$Variable, breaks=c(-Inf, q, Inf), labels=c("color1", "color2"))

To use the above subsetted data in a graph

ggplot(data=dataset) + geom\_point(mapping=aes(x=X, y=Y), color=dataset$Colorvariable)

To use a variable as part of ggplot’s aes function (to adjust components of the graph, or to put three variables on a two-dimensional graph)

thirdvariable <-dataset$variable

ggplot(data=dataset) + geom\_point(mapping=aes(x=Xvariable, y=Yvariable, size=thirdvariable))