R for Assessors

R Core Team (2013). R: A language and environment for statistical computing.

R Foundation for Statistical Computing, Vienna, Austria.

URL http://www.R-project.org/

IAAO

INTERNATIONAL ASSOCIATION *of* **ASSESSING OFFICERS**

WE are a professional membership organization of government assessment officials and others interested in the administration of the property tax. We were founded in 1934, and we have more than 7,000 members worldwide from governmental, business, and academic communities.

WE are the internationally recognized leader and preeminent source for innovation, education, and research in property appraisal, assessment administration and property tax policy.

WE are IAAO, and **WE** value the world!

Printed in the United States of America.

www.iaao.org

Contents

R for Assessors

What is R

In the summer of 2008 a group of mass appraisal practitioners met in Kansas City to discuss the future of mass appraisal education at IAAO. Specifically, they discussed the roll hands on education would play in future courses and how best to implement that idea. The result was an outline for four new courses aimed at moving students from a rudimentary knowledge of data analysis through to mastering basic modeling skills. This would be accomplished using Excel[1] in the early stages and SPSS[2] for the highest-level courses.

One of the major concerns that grew from that early meeting focused on the cost of the SPSS software. There was nearly unanimous agreement that this was the best overall platform on which to base advanced modeling techniques. The concern was that the cost of a license would discourage all but the largest jurisdictions from participating in the upper level courses.

Then in January of 2015 I took a MOOC (massive open online course) on statistics and the professor used a software package called "R". It is best described in the "About R" section of the R Project for Statistical Computing website (http://www.r-project.org/):

> *R is a language and environment for statistical computing and graphics. It is a GNU project which is similar to the S language and environment which was developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues. R can be considered as a different implementation of S. There are some important differences, but much code written for S runs unaltered under R.*
>
> *R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, ...) and graphical techniques, and is highly extensible. The S language is often the vehicle of choice for research in statistical methodology, and R provides an Open Source route to participation in that activity.*
>
> *One of R's strengths is the ease with which well-designed publication-quality plots can be produced, including mathematical symbols and formulae where needed. Great care has been taken over the defaults for the minor design choices in graphics, but the user retains full control.*
>
> *R is available as Free Software under the terms of the Free Software Foundation's GNU General Public License in source code form. It compiles and runs on a wide variety of UNIX platforms and similar systems (including FreeBSD and Linux), Windows and MacOS.*

The R environment

[1] Microsoft. Microsoft Excel. Redmond, Washington: Microsoft, 2010. Computer Software
[2] IBM SPSS Statistics for Windows. Armonk, NY: IBM Corp

R for Assessors

R is an integrated suite of software facilities for data manipulation, calculation and graphical display. It includes

- *an effective data handling and storage facility,*
- *a suite of operators for calculations on arrays, in particular matrices,*
- *a large, coherent, integrated collection of intermediate tools for data analysis,*
- *graphical facilities for data analysis and display either on-screen or on hardcopy, and*
- *a well-developed, simple and effective programming language which includes conditionals, loops, user-defined recursive functions and input and output facilities.*

The term "environment" is intended to characterize it as a fully planned and coherent system, rather than an incremental accretion of very specific and inflexible tools, as is frequently the case with other data analysis software.

R, like S, is designed around a true computer language, and it allows users to add additional functionality by defining new functions. Much of the system is itself written in the R dialect of S, which makes it easy for users to follow the algorithmic choices made. For computationally-intensive tasks, C, C++ and FORTRAN code can be linked and called at run time. Advanced users can write C code to manipulate R objects directly.

Many users think of R as a statistics system. We prefer to think of it of an environment within which statistical techniques are implemented. R can be extended (easily) via packages*. There are about eight packages supplied with the R distribution and many more are available through the CRAN family of Internet sites covering a very wide range of modern statistics.*

R has its own LaTeX-like documentation format, which is used to supply comprehensive documentation, both on-line in a number of formats and in hardcopy.

So far, I have found the basic version of the R software to be more than adequate to perform normal analysis functions in an assessment office. The user interface is neither elegant nor user friendly when compared to other commercially available statistical packages. The software is unforgiving and demands exact spellings, punctuation and is case sensitive, but ease of use is more than offset by the zero-purchase price.

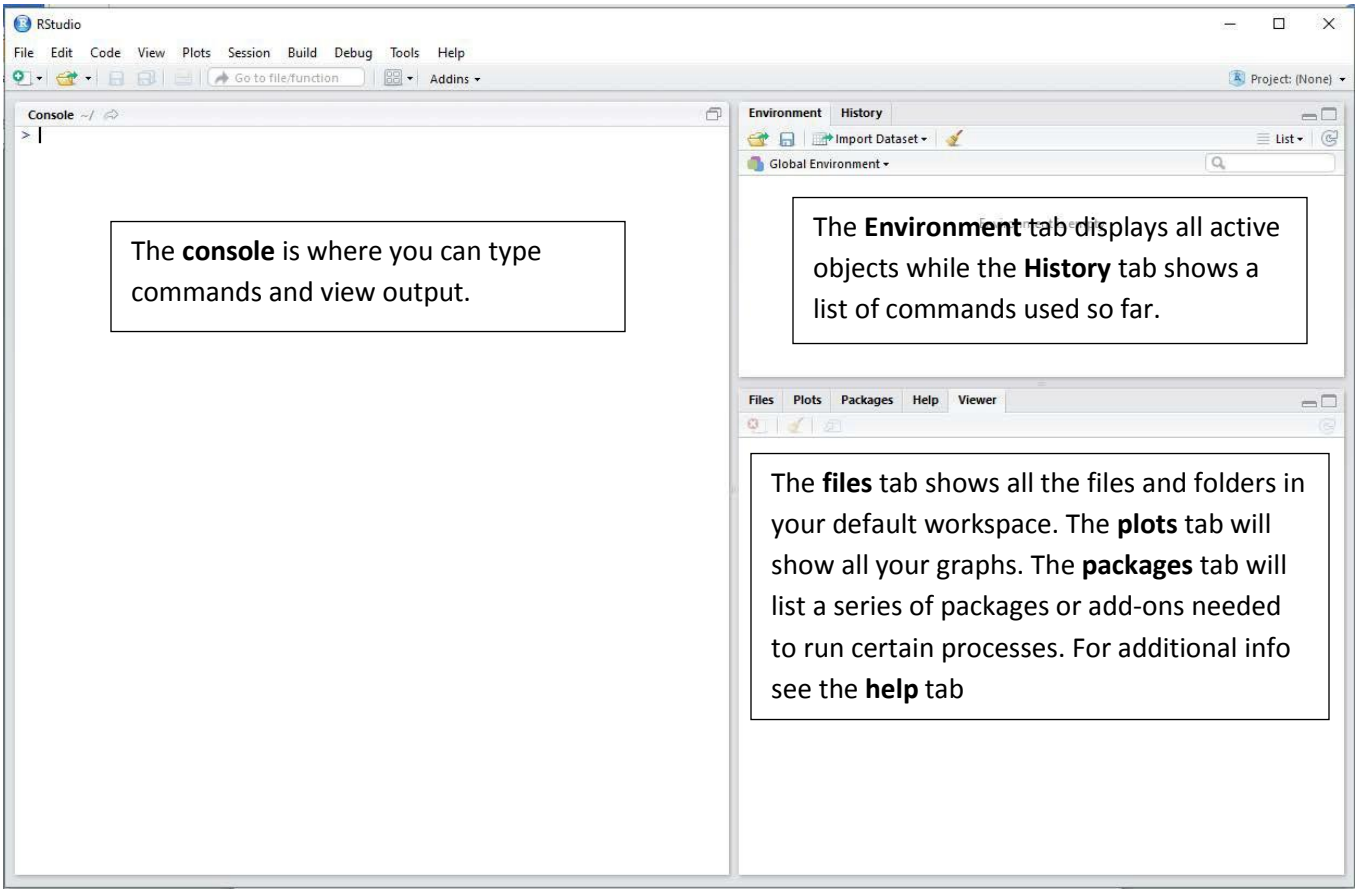
One cost that cannot be avoided is the price of learning the R language. Since this is a command driven environment, the user must learn the commands needed to accomplish desired tasks. The R site has an extensive help file the user can access directly through the interface and there are a wide variety of sources on the internet and in print to provide assistance. Although there is an extensive user group, most are involved in some way in the

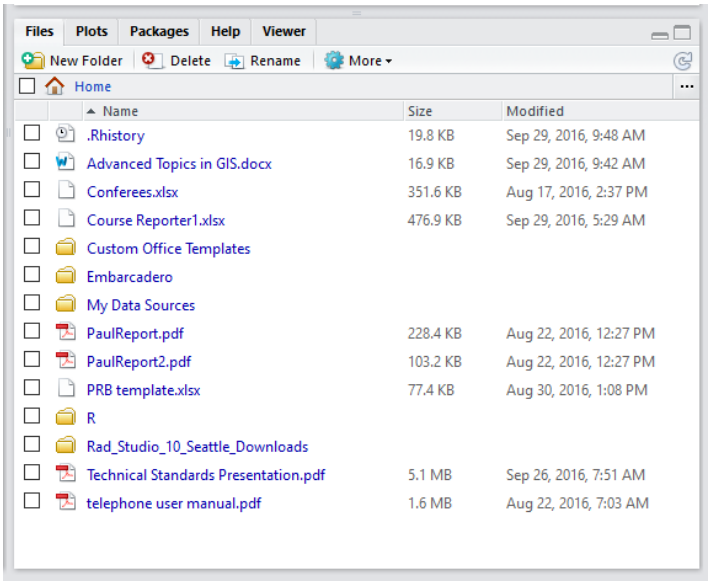
academic arena and not directly associated with appraising. That is the reason for this document. It is hoped that users within the assessment profession will share the syntax they have developed to solve particular problems and thereby build a library of functions that meet all the analytical requirements of any assessment office.

R Studio

One last item before we dive into the actual analysis and that is R Studio. I originally wrote this tutorial using only the R interface to minimize the amount of computer resources the user needed and to keep the approach as simple as possible. However, in only a short period of time I have come to appreciate the value of the flexibility and utility of the R Studio interface.

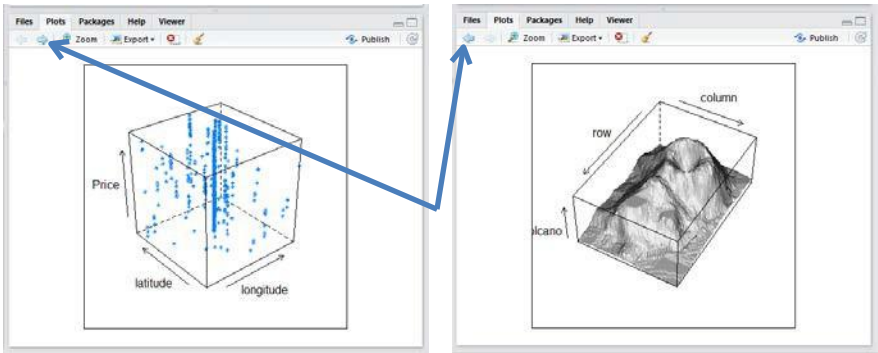
This interface is also an open source product, which means it is free to download and use. Its integrated development environment (IDE) is easy to learn and provides the user with tools that make it a valuable addition to the basic R package. Therefore, illustrations in this tutorial will make extensive use of this software. An explanation of the basic interface begins on the following page.



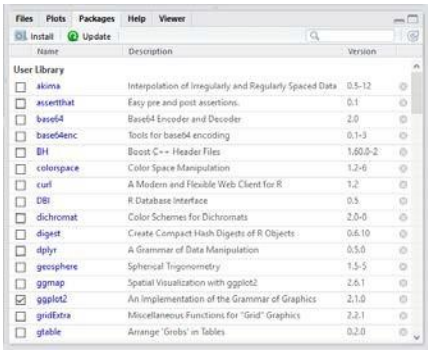


Use of the Files tab

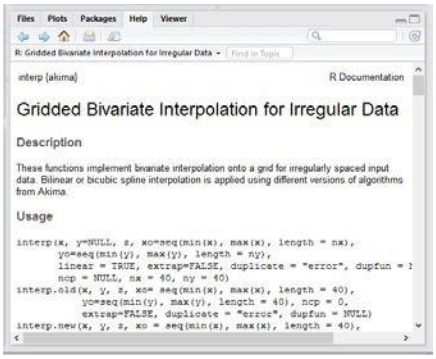
R for Assessors



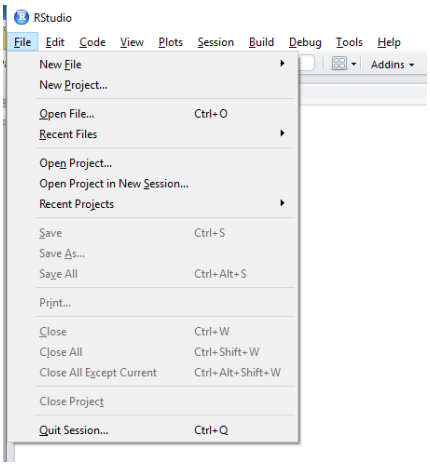
Unlike the basic R package, RStudio allows the user to save plots and move from one to the other using arrow keys.



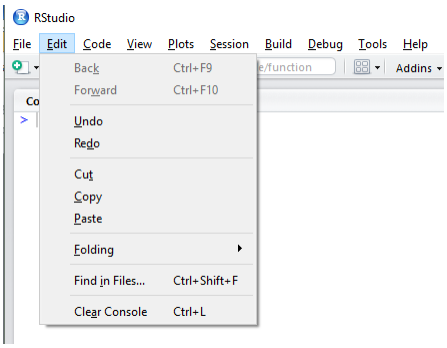
The Packages tab lists available packages the might be useful with R projects.



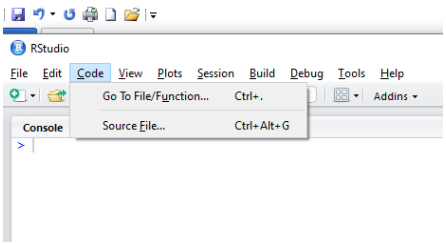
The Help tab allows the user to search for information on specific R topics such as the description of packages.



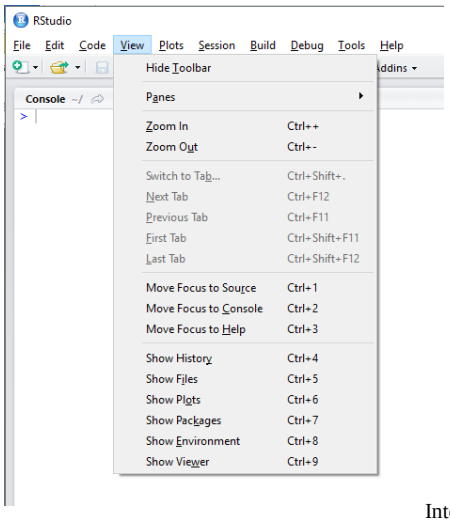
The File menu is very similar to that found in many commercial programs.



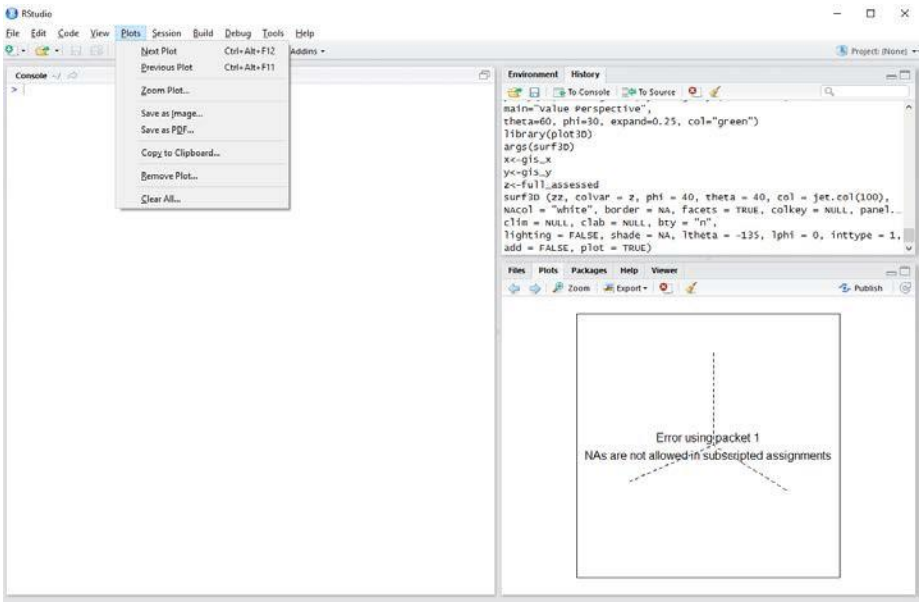
The Edit menu is also very similar to that found in many commercial programs. Notice the Clear Console function at the bottom. It may become useful as syntax is being built, tested and finalized.



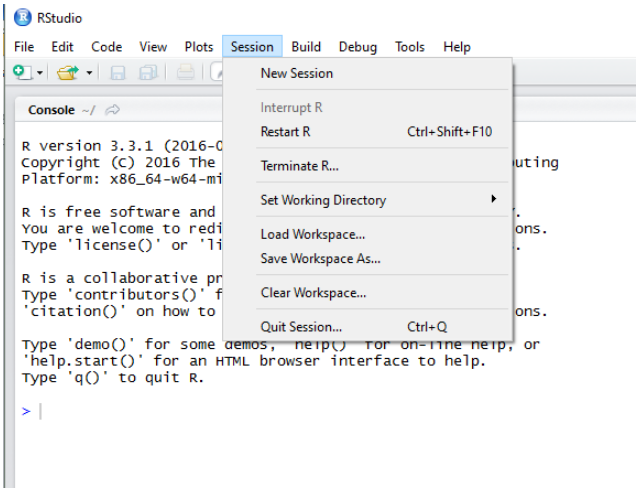
The Code menu provides options to help the user locate specific code and files.



The View menu provides control over what specific items are available for view by the user.



The Plots menu allows the user to control the plots created through execution of the syntax.



This menu allows the user to control aspects of the current RStudio session. One of those aspects is the working directory, which can be set and changed using this menu or through script.

The Working Directory

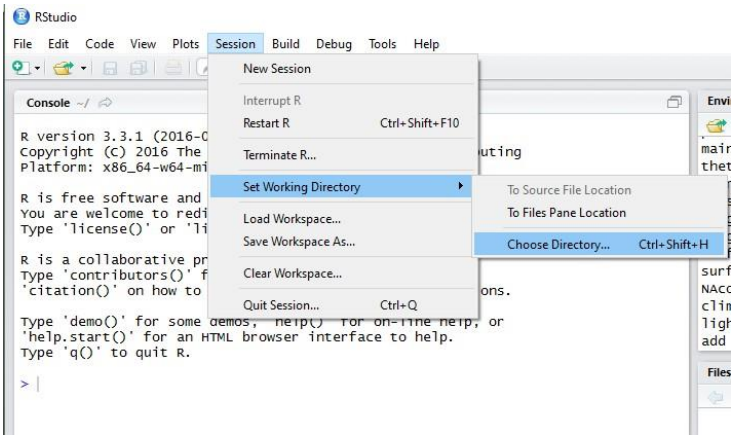
The working directory is where RStudio will store items such as script and the workspace. The current working directory can be accessed using the following script:

```
getwd()
```

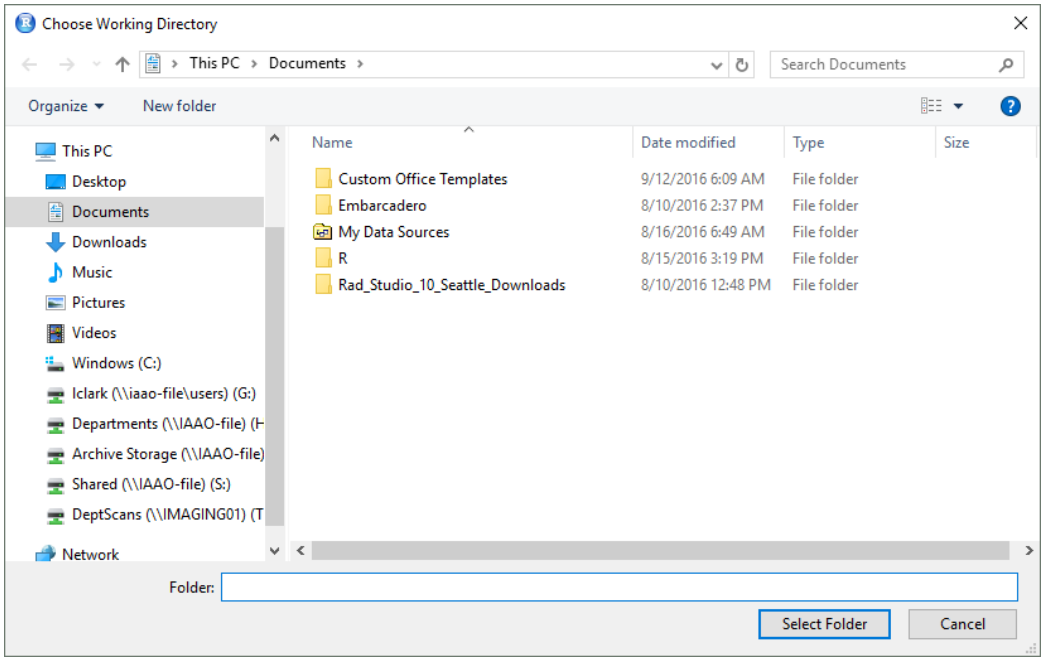
It can be changed using this script:

```
Setwd('C:\Data\RScript')
```

The user also has the option of going through the menu as shown below.



Selecting the Choose Directory option opens the Choose Working Directory dialogue window, allowing the user to graphically select a directory

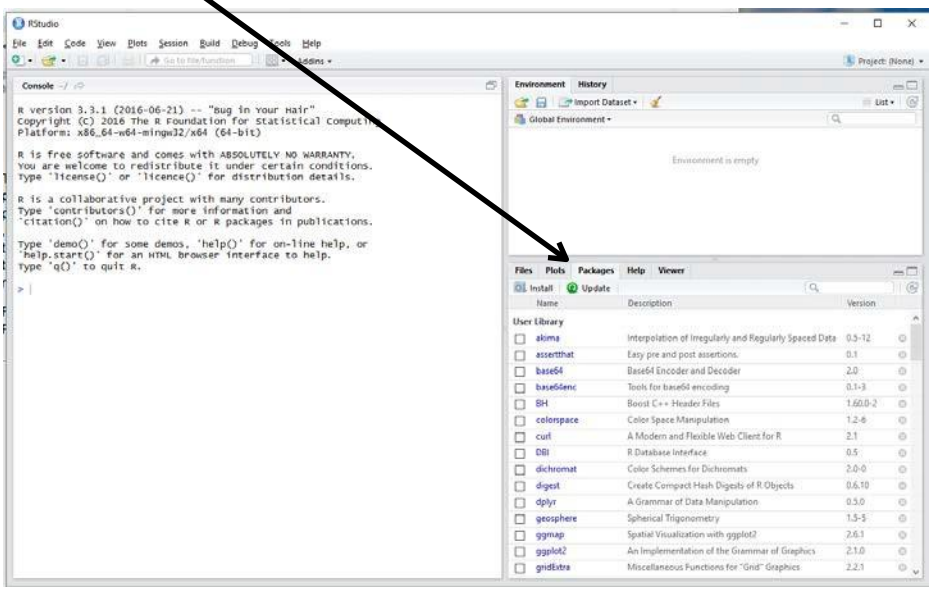


Packages

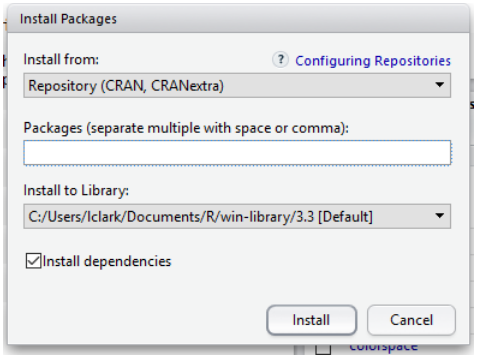
The twin benefits of using R are its open source base and the relatively large user group. New users can avoid the usual purchase costs associated with learning other statistical packages and they have a built-in support group of users to call on for help.

The down side, especially for new users, is the use of “packages” in R. A generic statistical package such as SPSS or SAS is purchased with certain built-in functionality and the option to purchase additional pieces. R as delivered has a great deal of functionality, but the new user, once exposed to the myriad of help sites, quickly learns of the availability of other packages that were developed to meet a specific need, one they may have. The challenge for the user is to select the appropriate package to satisfy a given need. We will deal with specific packages in more detail later in this tutorial.

RStudio is very helpful in working with packages. A list of packages will appear when the Packages menu item is selected.



This same dialogue window can be accessed from the Packages menu by selected the Install item or by selecting Install Packages under the Tools menu.



Entering a package name and selecting the Install option will cause that package to be installed in the library location under the title “Install to Library”.

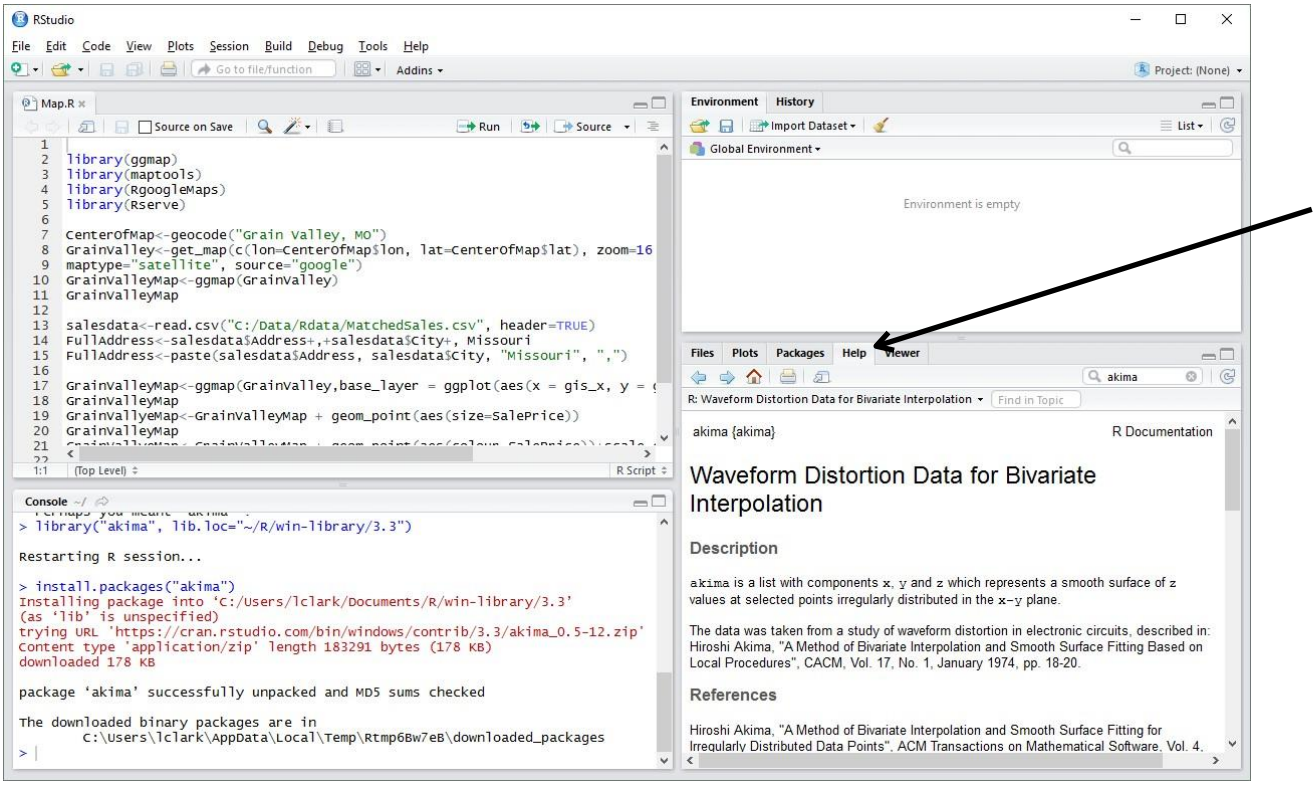
For example, entering the akima package will cause that package to be installed and the following script to appear in the console window:

> install.packages (“akima”)

This syntax can be used to install packages as well as the dialogue window.

It is necessary then to call each package that is going to be used in a syntax routine.

The syntax “library(akima)” is used to call the akima library. Once it is called any of the functions it contains can be used. Definitions of those functions can be found using the Help menu.



Loading Data

You might find that loading data into R can be quite frustrating. Almost every single type of file that you want to get into R seems to require its own function, and even then you might get lost in the functions' arguments. In short, it can be fairly easy to mix up things from time to time, whether you are a beginner or a more advanced R user... To cover these needs, DataCamp decided to publish a comprehensive, yet easy tutorial to quickly importing data into R, going from simple text files to the more advanced SPSS and SAS files. Keep on reading to find out how to easily import your files into R

Your Data

To import data into R, you first need to have data. This data can be saved in a file onto your computer (e.g. a local Excel, SPSS, or some other type of file), but can also live on the Internet or be obtained through other sources.

If you work with spreadsheets, the first row 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, insert a . in between to words instead of a space.

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.

Preparing Your R Workspace

Make sure to go into RStudio and see what needs to be done before you start your work there. You might have an environment that is still filled with data and values, which you can all delete using the following line of code:

rm(list=ls())

The rm() function allows you to “remove objects from a specified environment”. In this case, you specify that you want to consider a list for this function, which is the outcome of the ls() function. This last function returns you a vector of character strings that gives the names of the objects in the specified environment. Since this function has no argument, it is assumed that you mean the data sets and functions that you as a user have defined. Next, you might also find it handy to know where your working directory is set at the moment:

```
getwd()
```

And you might consider changing the path that you get as a result of this function, maybe to the folder in which you have stored your data set:

```
setwd("<location of your dataset>")
```

Getting Data From Common Sources into R

You will see that the following basic R functions focus on getting spreadsheets into R, rather than Excel or other type of files. If you are more interested in the latter, scroll a bit further to discover the ways of importing other files into R.

Using a CSV File

The most efficient means of bringing data into R for analysis is through one of several import functions. The simplest of these is through a comma delimited text file. If data is stored in an Excel spreadsheet or is downloaded from another computer program into an Excel format, the user can simply use the “Save As” functionality within Excel to save the file as Comma Delimited or “.csv”. Then the following command can be used to import that data into R:

```
Mydata<-read.table("C:/Data/Rdata/CRatios.csv", header=TRUE, sep=",")
```

“Mydata” is now the name of the data frame created within R using the command “read.table”. The name of the data frame created is up to the user (e.g. “2010Sales,” “Foreclosures,”).

“<-“is comparable to an equivalent or equals sign (Mydata=)

"read.table" tells the program to read the data from the table defined within the parentheses into Mydata

"C:/Data/Rdata/CRatios.csv" is the path and file name under which the data file is stored.

"header=TRUE" means variable names may be read from the first row in the imported table. This would represent the column headings in a spreadsheet. If data begins on line 1:row one, and there are no column names, "header=FALSE" would be used.

"sep=","" means each of the variables in a row is separated by a comma. If separated, for example, by spaces, "sep=" "" would be used.

The user should be aware R does not provide effective data management tools for "cleaning" data. Caution should be exercised in importing data that is consistent, accurate and complete. In this case consistency means that all entries within a given column represent the same data type, i.e., numeric, date/time, text. Accuracy means the data entered reflects the true state of affairs the data is meant to represent (a parcel that sold for $350,000 has a $350,000 sale price value and not $35,000 or $3,500,000). Finally, complete means there is no missing data or data elements.

It is common for a program such as Excel to change the nature of variables during the process of creating a comma delimited file. Dates may be expressed as numbers and vice versa. Variables with numerical values may be saved as text. Do not attempt to transform a worksheet that has calculations, merged cells, colored cells or graphs within it. If any of these exist, copy and paste the data into another clean worksheet and use the "Save As" command on the new worksheet.

R is not a forgiving program. It is up to the user to ensure the data made available to the functions in R is appropriate to the purpose.

Importing Files With Other Separator Characters

In case you have a file with a separator character that is different from a tab, a comma or a semicolon, you can always use the read.delim() and read.delim2() functions. These are variants of the read.table() function, just like the read.csv() function. Consequently, they have much in common with the read.table() function, except for the fact that they assume that the first line that is being read in is a header with the attribute names, while they use a tab as a separator instead of a whitespace, comma or semicolon. They also have the fill argument set to TRUE, which

means that blank field will be added to rows of unequal length. You can use the read.delim() and read.delim2() functions as follows:

```
df <- read.delim("<name and extension of your file>")
df <- read.delim2("<name and extension of your file>")
```

Importing Excel Files Into R

To load Excel files into R, you first need to do some further prepping of your workspace in the sense that you need to install packages. Simply run the following piece of code to accomplish this:

```
install.packages("<name of the package>")
```

When you have installed the package, you can just type in the following to activate it in your workspace:

```
library("<name of the package>")
```

To check if you already installed the package or not, type in

```
any(grepl("<name of your package>", installed.packages()))
```

Importing Excel Files With The Readxl Package

The readxl package allows R users to easily read in Excel files, just like this:

```
library(readxl)
df <- read_excel("<name and extension of your file>")
```

Note that the first argument specifies the path to your .xls or .xlsx file, which you can set by using the getwd() and setwd() functions. You can also add a sheet argument, as with the XLConnect package.

Importing XML Data Into R

If you want to get XML data into R, one of the easiest ways is through use of the XML package. First, make sure you install and load the XML package in your workspace, as demonstrated

above. Then, you can use the xmlTreeParse() function to parse the XML file directly from the web:

```
library(XML)
xmlfile <- xmlTreeParse("<Your URL to the XML data>")
```

Next, you can check whether R knows that xmlfile is in XML by entering:

```
class(xmlfile) #Result is usually similar to this: [1] "XMLDocument"
"XMLAbstractDocument"
```

Tip: you can use the xmlRoot() function to access the top node:

```
topxml <- xmlRoot(xmlfile)
```

You will see that the data is presented kind of weirdly when you try printing out the xmlfile vector. That is because the XML file is still a real XML document in R at this point. In order to put the data in a data frame, you first need to extract the XML values. You can use the xmlSApply() function to do this:

```
topxml <- xmlSApply(topxml, function(x) xmlSApply(x, xmlValue))
```

The first argument of this function will be topxml, since it is the top node on whose children you want to perform a certain function. Then, you list the function that you want to apply to each child node. In this case, you want to extract the contents of a leaf XML node. This, in combination with the first argument topxml, will make sure that you will do this for each leaf XML node. Lastly, you put the values in a dataframe You use the data.frame() function in combination with the matrix transposition function t() to do this. Additionally you also specify that no row names should be included:

```
xml_df <- data.frame(t(topxml), row.names=NULL)
```

You can also choose not to do all the previous steps, which are a bit more complicated, and to just do the following:

```
url <- "<a URL with XML data>"
data_df <- xmlToDataFrame(url)
```

Importing Data From HTML Tables Into R

Getting data From HTML tables into R is pretty straightforward:

```
url <- "<a URL>"
data_df <- readHTMLTable(url, which=3)
```

Note that the which argument allows you to specify which tables to return from within the document. If this gives you an error in the nature of “failed to load external entity”, don't be confused: this error has been signaled by many people and has been confirmed by the package's author here. You can work around this by using the RCurl package in combination with the XML package to read in your data:

```
library(XML)
library(RCurl)
url <- "YourURL"
urldata <- getURL(url)
data <- readHTMLTable(urldata, stringsAsFactors = FALSE)
```

Note that you don't want the strings to be registered as factors or categorical variables You can also use the httr package to accomplish exactly the same thing, except for the fact that you will want to convert the raw objects of the URL's content to characters by using the rawToChar argument:

```
library(httr)
urldata <- GET(url)
data <- readHTMLTable(rawToChar(urldata$content), stringsAsFactors =
FALSE)
```

Getting Data From Statistical Software Packages into R

For the following more advanced statistical software programs, there are corresponding packages that you first need to install in order to read your data files into R, just like you do with Excel or JSON.

Importing SPSS Files into R

If you are looking to import your SPSS files into R, install the foreign package and run the read.spss() function contained within it.

```
library(foreign)
mySPSSData <- read.spss("example.sav")
```

Tip: if you wish the result to be displayed in a data frame, make sure to set the to.data.frame argument of the read.spss() function to TRUE. Furthermore, if you do NOT want the variables with value labels to be converted into R factors with corresponding levels, you should set the use.value.labels argument to FALSE:

```
library(foreign)
mySPSSData <- read.spss("example.sav", to.data.frame=TRUE,
use.value.labels=FALSE)
```

Remember that factors are variables that can only contain a limited number of different values. As such, they are often called “categorical variables”. The different values of factors can be labeled and are therefore often called “value labels”

Importing Stata Files into R

Use the foreign package to import Stata files:

```
library(foreign)
mydata <- read.dta("<Path to file>")
```

Importing Systat Files into R

This package is also used to get Systat files into R as shown below:

```
library(foreign)
mydata <- read.systat("<Path to file>")
```

Importing SAS Files into R

Inporting SAS files into R requires installation of the sas7bdat package. Load it, and then invoke the read.sas7bdat() function contained within the package.

```
library(sas7bdat)
mySASData <- read.sas7bdat("example.sas7bdat")
```

Using RODBC

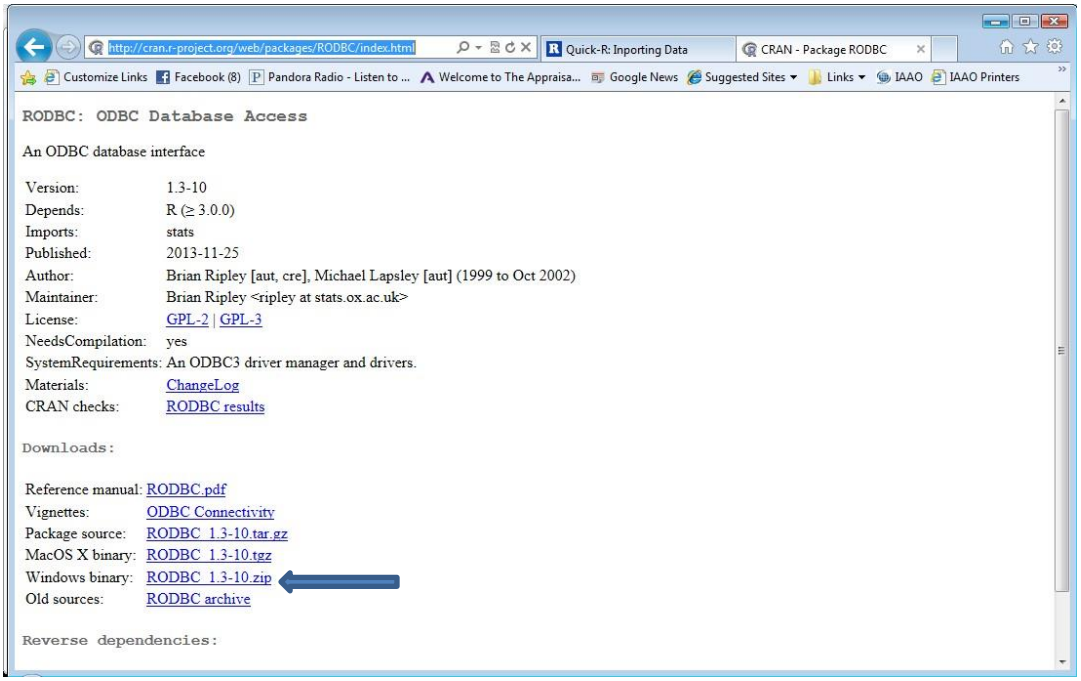
According to Wikipedia, “ODBC (Open Database Connectivity) is a standard programming language middleware API (Application Programming Interface) for accessing database management systems (DBMS)”. It is typically used to gain access to data stored in ODBC compliant database systems. RODBC represents the R language implementation of open database connectivity.

Although RODBC will help the user gain access to odbc compliant data bases, it will also allow access to Excel worksheets, which is the focus of this discussion. The first piece of advice concerning importing Excel spreadsheets from the Cran.r-project website is “avoid doing so if possible ” The functionality is there but the ability to do something says nothing about the advisability of doing it. With that said, I am certain there will be assessors who want to directly import their Excel spreadsheets into R and the following steps will provide one mechanism for accomplishing that.

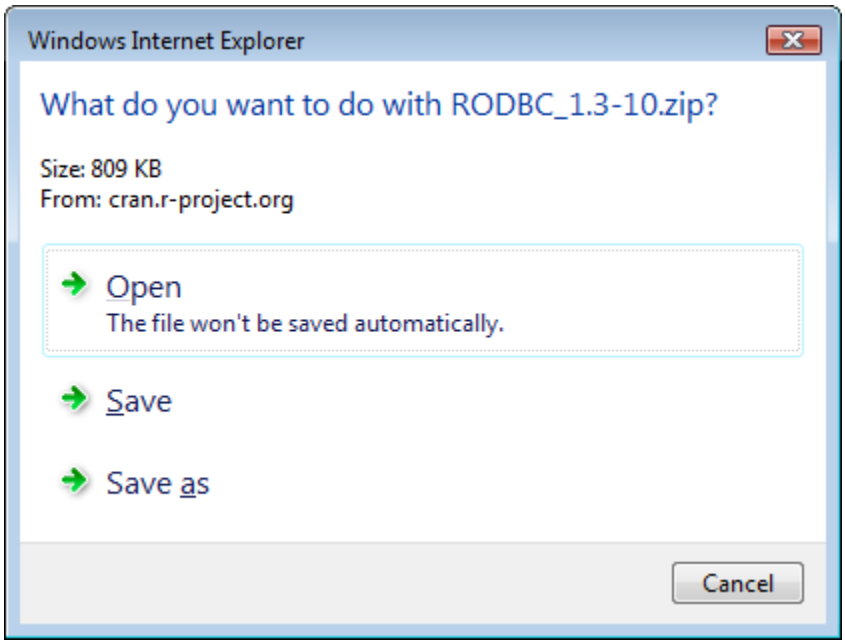
These reflect the steps I took to download and use RODBC. It is possible that some of them may change from the time of this writing. The Cran.r-project website is relatively user-friendly, making finding the needed package easier.

First the user must download the package. I found it using this URL:

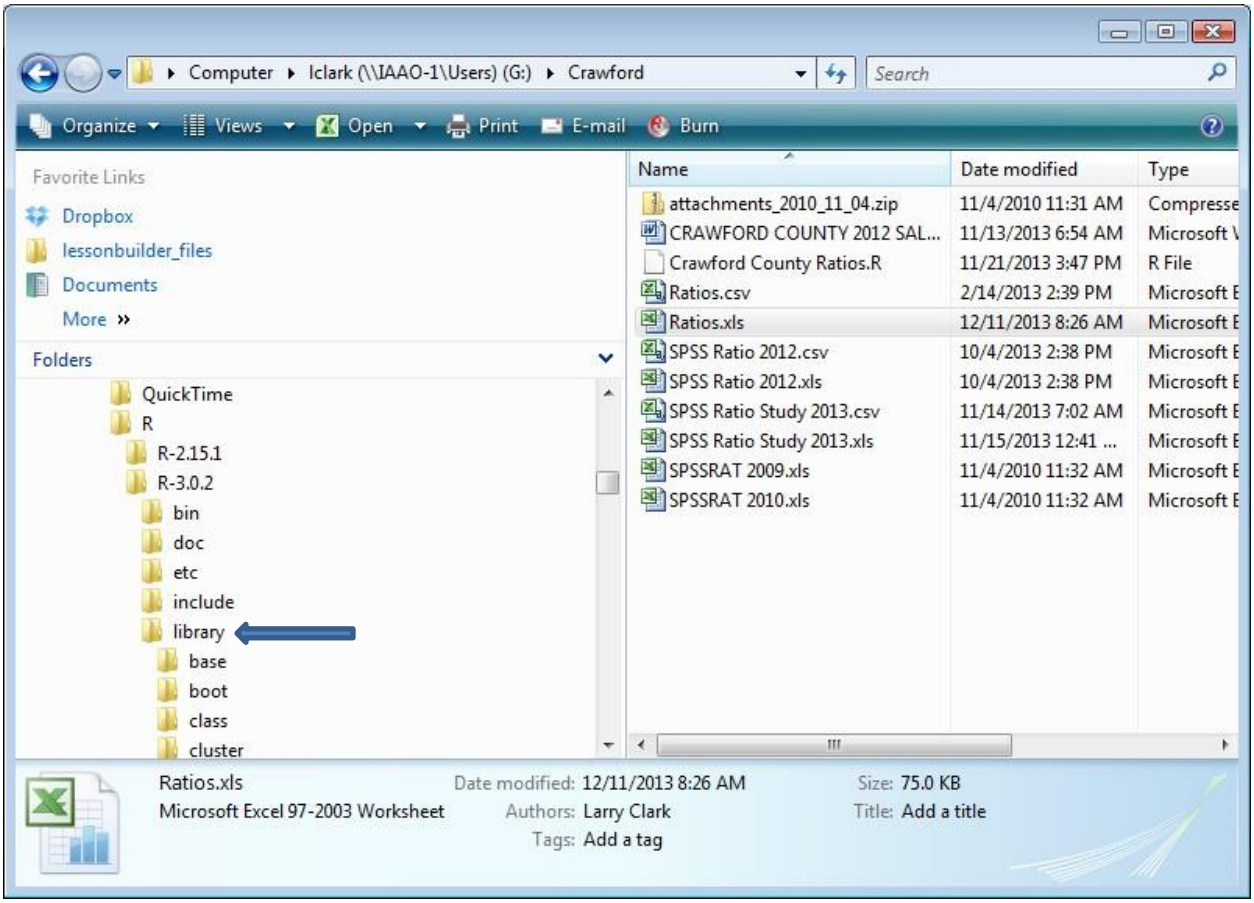
http://cran.r-project.org/web/packages/RODBC/index.html

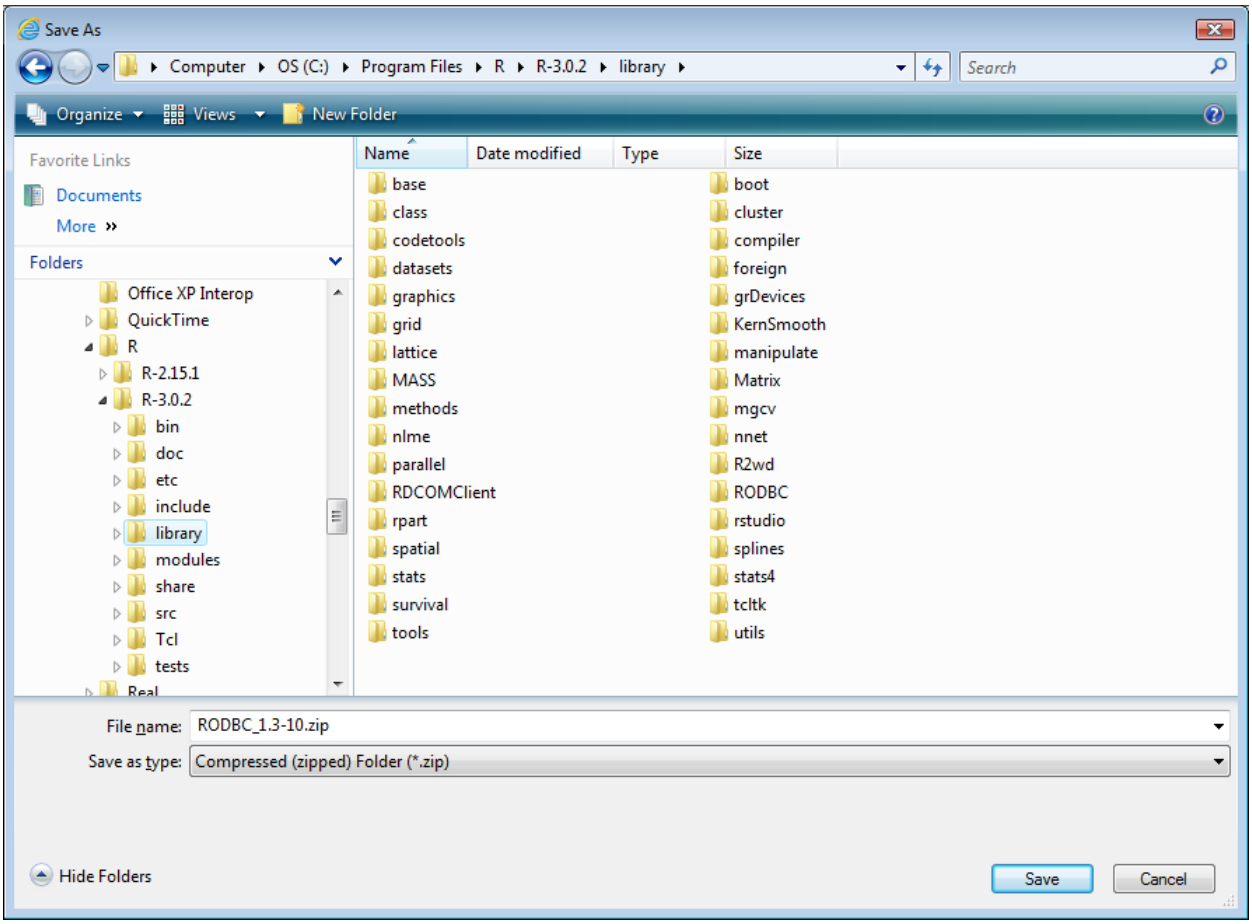


Being a Windows user, I selected the package labeled “Windows binary: EODBC_1.3-10.zip” Clicking on that link brought up the following dialog window:

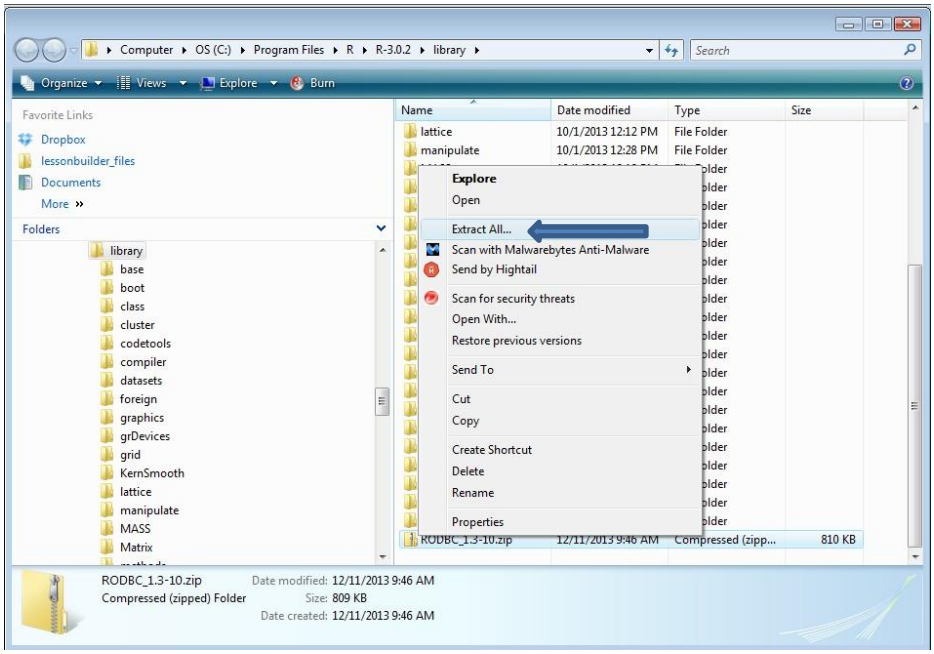


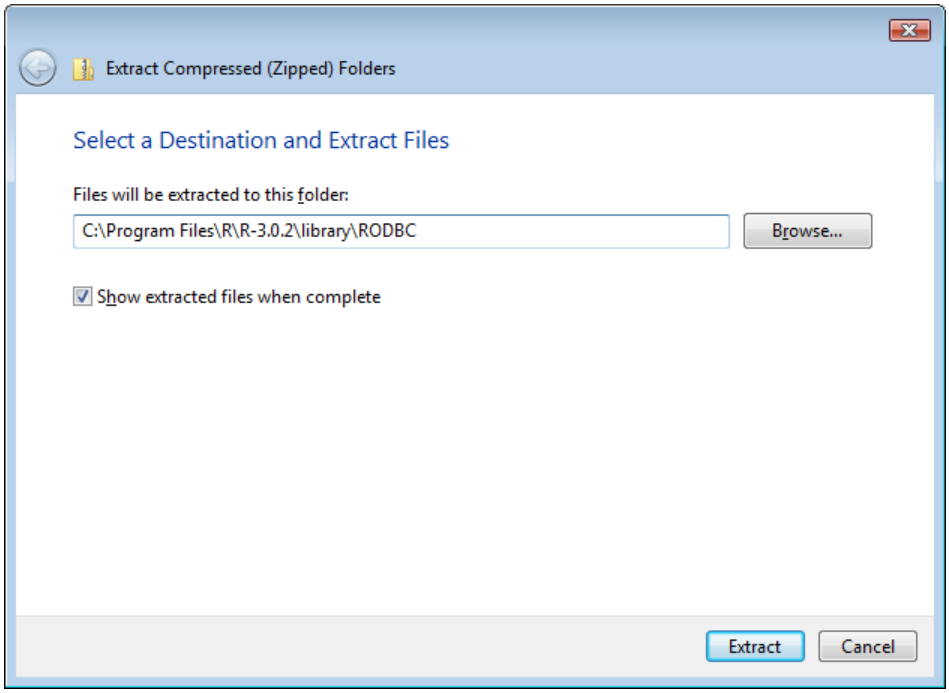
You the user can save some time by selecting the “Save as” function and save the downloaded file to “library” folder under the “R” folder within the “Program Files” on your C drive.





This action will create a folder under “library” called “RODBC_1.3-10.zip”. Use the Windows extract functionality to unzip all the necessary files and place them in a folder called “RODBC” under “library”. Once that is done the user can delete the zip folder.





Several commands are then used to import the worksheet data into R for analysis. First, the RODBC commands must be made available to the program using the library command.

```
library (RODBC)
```

Then the user specifies the connection to Excel and the location of the worksheet.

```
channel<-odbcConnectExcel("C:/Data/Ratios.xls")
```

The user assigns the data from a named worksheet to a data frame.

```
mydata<-sqlFetch(channel, "Ratios")
```

The "mydata" data frame is now available for use by R.

The final step in the process is for the user to close the odbc connection.

```
odbcClose(channel)
```

Basic Analysis

Summary

At its most basic level, analysis of data is looking at data in very specific ways. One of the easiest approaches in R uses the "summary" command:

```
summary(sample)
```

where:

summary is the specific R command

Sample is the data set being described

Unfortunately for the user, the output is not as elegant as that produced by some other software packages. The following is a portion of the output from one set of data:

```
      NBHD          LANDSQFT          BLDGS            QUAL
 Min.   :1.000   Min.   :     0   Min.   :1.000   Min.   :1.00
 1st Qu.:3.000   1st Qu.: 14286   1st Qu.:1.000   1st Qu.:3.00
 Median :5.000   Median : 34832   Median :1.000   Median :3.00
 Mean  :3.982  Mean   : 68231   Mean   :1.018   Mean   :3.43
 3rd Qu.:5.000  3rd Qu.:  62468   3rd Qu.:1.000   3rd Qu.:4.00
 Max.   :6.000   Max.   :962383   Max.   :2.000   Max.   :6.00
     YRBLT           EYB          LIVAREA        SALEYEAR
 Min.   :1900   Min.   :1928   Min.   : 432   Min.   :2015
 1st Qu.:1955   1st Qu.:1970   1st Qu.:1189   1st Qu.:2015
 Median :1974   Median :1982   Median :1520   Median :2015
 Mean  :1973  Mean   :1981   Mean   :1702   Mean   :2015
 3rd Qu.:1994   3rd Qu.:1995   3rd Qu.:2026   3rd Qu.:2015
 Max.   :2009   Max.   :2009   Max.   :5473   Max.   :2015
     MONTH            PRICE
 Min.   : 1.000   Min.   :  4000
 1st Qu.: 3.000   1st Qu.: 40000
 Median : 6.000   Median : 70000
 Mean   : 6.236   Mean   : 91036
 3rd  Qu.:10.000  3rd  Qu.:132000
 Max.   :12.000   Max.   :330000
```

R produces different statistics depending on the nature of the underlying variable. For the numeric (continuous) variable "LANDSQFT" it provides the minimum, median, mean and maximum values along with the values at each of the first and third quartiles. For a character (categorical) variable it provides the frequency of the occurrence of each category. In this sample the variable QUAL is actually a categorical variable but is represented as numbers and therefore,

treated like other continuous variables. An actual categorical variable such as CONDITION might look like the table below.

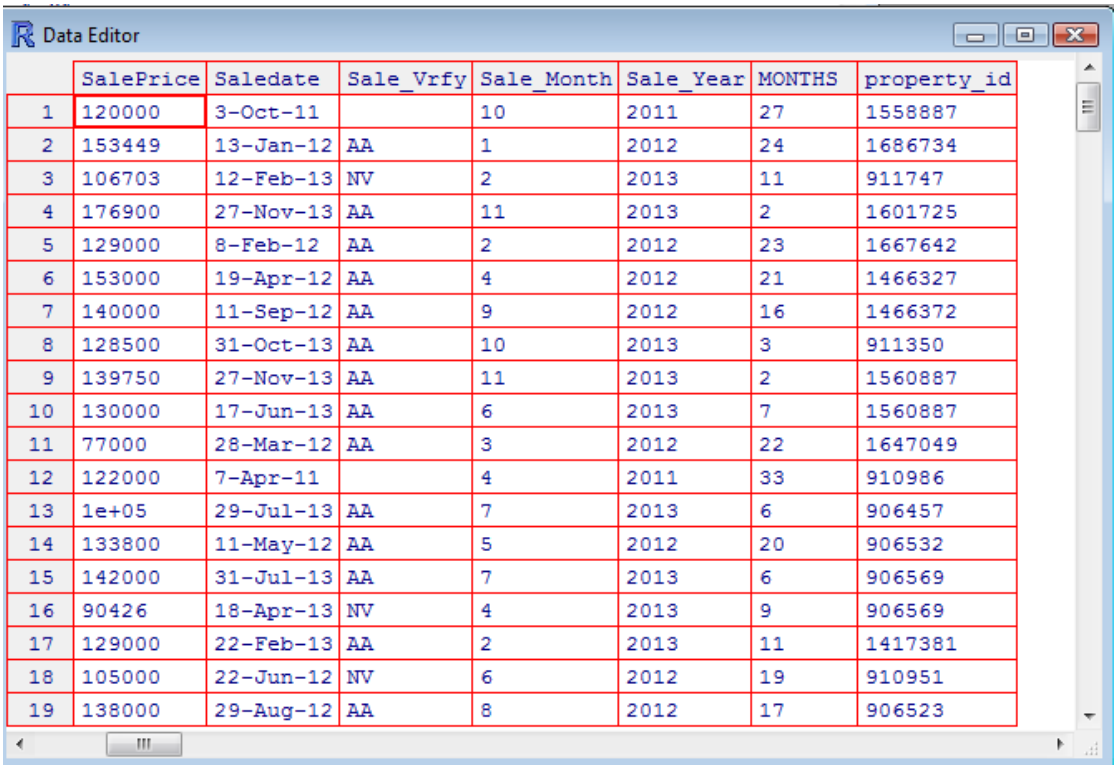
CONDITION	
AV	96
GD	54
EX	40
VG	30
FR	10
PR	2
(Other)	2

Summary statistics on individual variables may be obtained using the mean, median and sd commands.

R provides two approaches to examining the data in a spreadsheet format. One command is “edit”.

```
edit(sample)
```

This causes the data to be displayed in a spreadsheet format as follows:


Data Editor

	SalePrice	Saledate	Sale_Vrfy	Sale_Month	Sale_Year	MONTHS	property_id
1	120000	3-Oct-11		10	2011	27	1558887
2	153449	13-Jan-12	AA	1	2012	24	1686734
3	106703	12-Feb-13	NV	2	2013	11	911747
4	176900	27-Nov-13	AA	11	2013	2	1601725
5	129000	8-Feb-12	AA	2	2012	23	1667642
6	153000	19-Apr-12	AA	4	2012	21	1466327
7	140000	11-Sep-12	AA	9	2012	16	1466372
8	128500	31-Oct-13	AA	10	2013	3	911350
9	139750	27-Nov-13	AA	11	2013	2	1560887
10	130000	17-Jun-13	AA	6	2013	7	1560887
11	77000	28-Mar-12	AA	3	2012	22	1647049
12	122000	7-Apr-11		4	2011	33	910986
13	1e+05	29-Jul-13	AA	7	2013	6	906457
14	133800	11-May-12	AA	5	2012	20	906532
15	142000	31-Jul-13	AA	7	2013	6	906569
16	90426	18-Apr-13	NV	4	2013	9	906569
17	129000	22-Feb-13	AA	2	2013	11	1417381
18	105000	22-Jun-12	NV	6	2012	19	910951
19	138000	29-Aug-12	AA	8	2012	17	906523

The user can move around freely in this spreadsheet using the arrow keys or the horizontal and/or vertical scroll bars. The weaknesses in this approach are that not all the cases are displayed and the user cannot make changes to the underlying data, because what he or she sees is a copy of that data.

The “fix” command on the other hand, will allow the user to edit the underlying data, but it does not make all cases available to the user. Therefore, it does not have the same utility as a regular spreadsheet program such as Excel. It will allow the user to determine whether variables were translated in a numeric or character format. This is helpful given the fact that some translations into CSV format produce inconsistent and unexpected results.

Selecting Observations

Looking once again at the LANDSQFT variable notice the minimum value is 0, which means one of the cases in this sample does not have a land component. If the analyst wants to remove that case the easiest approach is to create a subset of data that excludes cases based on land size as follows:

```
newdata<-Sample[which(Sample$LANDSQFT>0),]
```

Running the summary command against this new data set produces the following results:

```
     NBHD          LANDSQFT          BLDGS            QUAL
 Min.   :1.00   Min.   :  3561   Min.   :1.000   Min.   :1.000
 1st Qu.:3.00   1st Qu.: 14448   1st Qu.:1.000   1st Qu.:3.000
 Median :5.00   Median : 35547   Median :1.000   Median :3.000
 Mean   :3.97   Mean   : 68647   Mean   :1.018   Mean   :3.427
 3rd Qu.:5.00   3rd Qu.: 62630   3rd Qu.:1.000   3rd Qu.:4.000
 Max.   :6.00   Max.   :962383   Max.   :2.000   Max.   :6.000
     YRBLT           EYB          LIVAREA        SALEYEAR
 Min.   :1900   Min.   :1928   Min.   : 432   Min.   :2015
 1st Qu.:1955   1st Qu.:1970   1st Qu.:1187   1st Qu.:2015
 Median :1974   Median :1982   Median :1523   Median :2015
 Mean   :1973   Mean   :1981   Mean   :1703   Mean   :2015
 3rd Qu.:1994   3rd Qu.:1995   3rd Qu.:2028   3rd Qu.:2015
 Max.   :2009   Max.   :2009   Max.   :5473   Max.   :2015
     MONTH            PRICE
 Min.   : 1.000   Min.   :  4000
 1st Qu.: 3.000   1st Qu.: 39975
 Median : 6.000   Median : 70000
 Mean   : 6.213   Mean   : 90775
 3rd Qu.:10.000   3rd Qu.:132000
 Max.   :12.000   Max.   :330000
```

Frequencies

It is helpful for the appraiser to learn how often certain categorical variables occur in the sample. For example, when the dominant grade in a sales sample is above average (A or B) rather than the average (C), it will skew the results of any model created with that data. The appraiser needs to know that before running a model and one of the ways of discovering it is through the use of Frequencies.

The syntax to produce a table of frequencies of a categorical variable is:

```
freq<-table(variable)
freq
```

The first line will create a table showing the number of occurrences of each category of the variable. The next line will print the result. For example, the number of occurrences of each of the categories of a quality variable may be found as follows:

```
freq<-table(QUAL)
freq
```

The result may appear as follows:

```
QUAL
 1  2  3  4  5  6
 7 12 84 29 29  3
```

This table shows there are 84 instances of quality level 3 in this sample. Of course, the appraiser must know how to interpret each of these levels for the output to be useful.

QUAL

1	2	3	4	5	6
7	12	84	29	29	3

Measures of Central Tendency

This and the following discussion of measures of dispersion closely follow chapter 2 of the student reference manual for Course 332 – Modeling Concepts.

In statistics, a central tendency is a central or typical value for a probability distribution, according to Wikipedia. Another definition describes it as a single value that attempts to describe a set of data by identifying the central position with that set of data.

In either case, appraisers use measures of central tendency in data analysis and that is why we need to learn how to calculate them in R software. The DescTools package has gathered together many of the most common statistical functions in one place. While at this writing it is in beta, I would recommend its use.

One of the most common measures of central tendency is the arithmetic mean or average. It is calculated by adding together every instance of a given variable within a sample and dividing that sum by the number of instances.

1+2+3+4+5 = 15 ÷ 5 = 3

Earlier in this tutorial the “summary” command was introduced. It can be used to display the mean of every numeric variable in a data set. When the average or mean of a single variable is desired the “mean” command is used.

```
mean(variable name)
```

Output will appear like the example below:

>mean(SPR)
[1] 358327.7

Simply by adding the trim command, the program will generate a trimmed mean.

```
mean(variable name, trim=decimal)
```

This results in the following output:

>mean(SPR, trim=0.05)
[1] 344041.2

The median is simply the middle number in a list of numbers that are arrayed in ascending or descending order of magnitude. In the example listing from above the number 3 is the middle number in that array.

1,2,**3**,4,5 or 5,4,**3**,2,1

The median is included in the summary command, but like the mean can be calculated separately.

```
median(variable name)
```

The results of this calculation will appear as follows:

>median(SPR)
[1] 321111

The geometric mean is seldom used in mass appraisal work except in some ratio studies. It is calculated as the nth root of the product of multiplying a set of n numbers together. This function is calculated as follows:

```
Geometric.mean(variable name)
```

>geometric.mean(SPR)
[1] 340982.8

The mode is seldom used because it simply represents the most frequently occurring value in a sample and is not used in any calculations.

```
Mode(variable name)
```

>Mode(SPR)

[1] 305000

Measures of Dispersion

Measures of dispersion summarize the degree of spread or variance in the data. The easiest measures to calculate are the minimum, maximum and range. The minimum is the smallest value in an array and the maximum is the largest value, with the range being the difference between the two.

R can be used to find these measures in different ways. First, the summary command calculates the minimum and maximum values for every numeric variable in the data frame.

```
      SPR
Min.    :  179000
1st Qu. :  276900
Median  :  321111
Mean    :  358328
3rd Qu. :  404850
Max.    : 1270000
```

The Range command within the DescTools package produces the amount of the range along with the bounds.

```
Range(variable name)
```

```
>Range(SPR)
[1] 1091000
attr(,"bounds")
[1] 179000 1270000
```

The average deviation is the average (sign-ignored) difference between each value and a chosen measure of central tendency, typically the mean or median. The DescTools package contains a command called MeanAD which calculates the mean absolute deviation from a chosen measure of central tendency.

```
MeanAD(x, FUN=mean, na.rm=FALSE)
X     the vector containing the observations
FUN   the name of the function to be used as center. Default is mean
na.rm      a logical value indicating whether or not missing values
           whould be removed.
```

>MeanAD(SPR, FUN=median)
[1] 84706.8

The coefficient of dispersion is the most common measure of dispersion used by assessors. There is no command built into R that calculates the COD. However, it is easily calculated using results from the previous calculations of the median and mean absolute deviation.

```
COD<-(MeanAD/Median)*100
```

>COD<-(84706.8/321111)*100
>COD
[1] 26.37929

The most common measure of dispersion in statistical practice is the standard deviation. Because of that it is natural to find a function for its calculation within R.

```
SD(variable name)
```

>SD(SPR)
[1] 128645.5

Percentiles are another way of analyzing the dispersion of data in a sample. A percentile is defined the student reference manual as dividing points between specific percentages of the

data. For example, the median represents the 50th percentile because 50% of the data lies below that amount and 50% lies above it. We have already found the median of our subject variable SPR (selling price) at 321111. The command used to find the percentile is:

```
quantile(variable name, c(percentile, percentile, . . .)
```

```
> quantile(SPR, c(.32, .57, .98))
    32%        57%        98%
288594.0   335000.0   749194.4
```

Any decimal figures can be inserted in place of those shown above.

Cross tabulations

Just as it is important for the appraiser to know how many times a particular category of a variable occurs within a dataset, it is valuable to compare one categorical variable to another. Cross tabulations show the relationship between two non-continuous variables (categorical variables or variables based on counts). The table below compares architectural style (Style) to a combination condition-desirability-utility (CDU) rating. This type of information will be helpful in specifying and calibrating the valuation model. This comparison may be accomplished using a cross tabulation of one categorical variable to another through cross tabulations. This is accomplished in R using the “gmodels” package and the syntax

```
library(gmodels)
CrossTable(mydata$rowvariable, mydata$columnvariable)
```

Where “mydata” represents the data frame;

“rowvariable” represents the categorical variable to be displayed on each row and

“columnvariable” represents the categorical variable to be displayed in each column.

There are several options described in the help file with this package and one to consider is the format option that displays the cross tabulation table just as it would appear using SPSS. That is especially helpful for those of us who are familiar with SPSS output. The image below is an example of the output.

The cross tabulations command can be used to show the number and percent of occurrences of each architectural style within each Grade.

```
   Cell Contents
|-------------------------|
|                       N |
| Chi-square contribution |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|

Total Observations in Table:  861
```

	GRADE							
STYL	A	A-	B	B-	B+	C	C+	Row Total
2	0	0	0	0	0	1	1	2
	0.007	0.023	0.279	0.402	0.070	14.618	0.022	
	0.000	0.000	0.000	0.000	0.000	0.500	0.500	0.002
	0.000	0.000	0.000	0.000	0.000	0.038	0.002	
	0.000	0.000	0.000	0.000	0.000	0.001	0.001	
3	0	0	0	0	0	0	4	4
	0.014	0.046	0.557	0.804	0.139	0.121	1.220	
	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.005
	0.000	0.000	0.000	0.000	0.000	0.000	0.008	
	0.000	0.000	0.000	0.000	0.000	0.000	0.005	
4	3	5	93	156	29	25	490	801
	0.016	1.990	3.112	0.152	0.043	0.027	1.431	
	0.004	0.006	0.116	0.195	0.036	0.031	0.612	0.930
	1.000	0.500	0.775	0.902	0.967	0.962	0.982	
	0.003	0.006	0.108	0.181	0.034	0.029	0.569	
16	0	5	27	17	1	0	4	54
	0.188	30.488	50.389	3.486	0.413	1.631	23.807	
	0.000	0.093	0.500	0.315	0.019	0.000	0.074	0.063
	0.000	0.500	0.225	0.098	0.033	0.000	0.008	
	0.000	0.006	0.031	0.020	0.001	0.000	0.005	
Column Total	3	10	120	173	30	26	499	861
	0.003	0.012	0.139	0.201	0.035	0.030	0.580	

This table reveals 156 of the 861 observations were graded as B- and were considered within the style rating of 4. That represented 19.5% of all the observations within that style and 90.2% of all observations within that grade or 18.1% of all observations.

Pivot Table

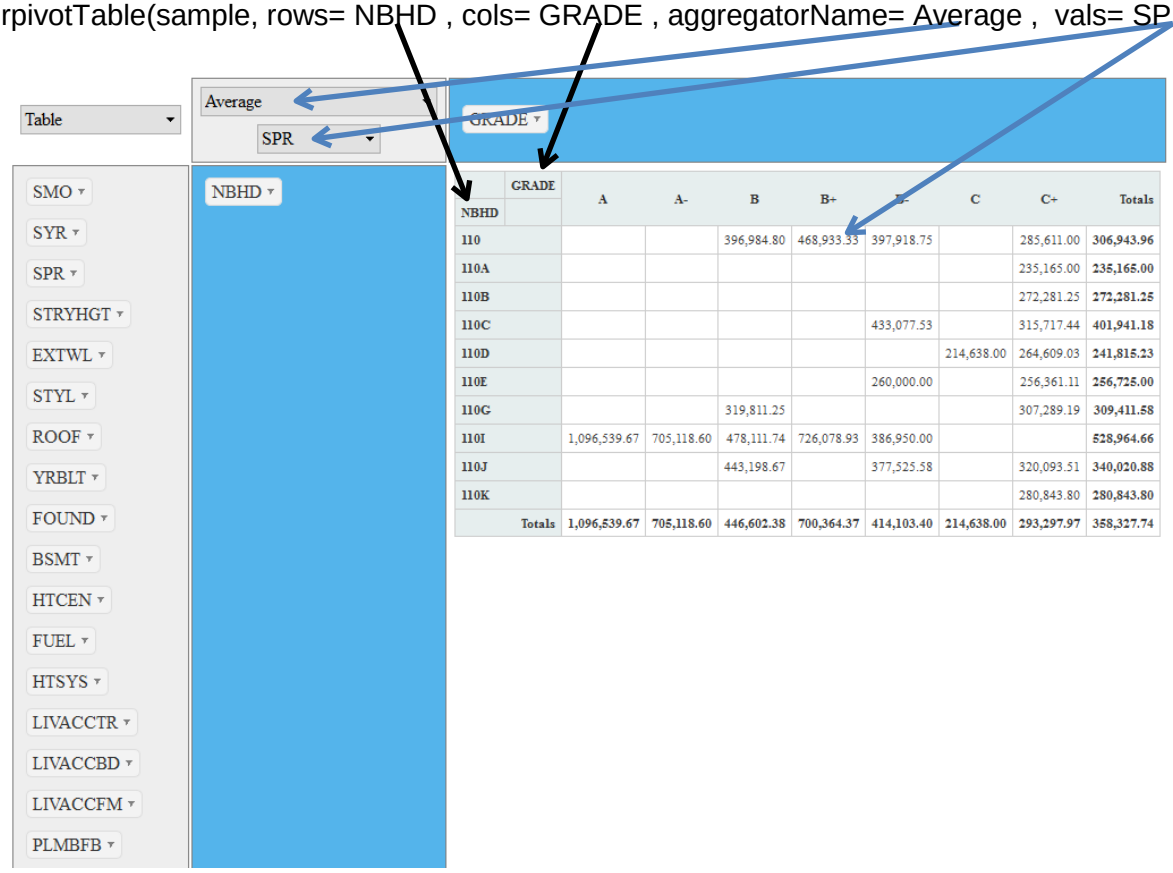
Another useful approach to analyzing data is through the use of a pivot table. Like the cross tabulation, this table utilizes variables as rows and columns. The difference is that a third variable is aggregated within the cells that represent the intersection of the row and column variables.

The package used is called rpivotTable and the command line looks like the following:

```
library(rpivotTable)
rpivotTable(data, rows=variable, cols=variable,
aggregatorName=function, vals=variable)
```

What follows is the syntax and its output as displayed in the Viewer of an RStudio session. The dataframe is called “sample”.

rpivotTable(sample, rows= NBHD , cols= GRADE , aggregatorName= Average , vals= SPR)



NBHD \ GRADE	A	A-	B	B+	B-	C	C+	Totals
110			396,984.80	468,933.33	397,918.75		285,611.00	**306,943.96**
110A							235,165.00	**235,165.00**
110B							272,281.25	**272,281.25**
110C					433,077.53		315,717.44	**401,941.18**
110D						214,638.00	264,609.03	**241,815.23**
110E					260,000.00		256,361.11	**256,725.00**
110G			319,811.25				307,289.19	**309,411.58**
110I	1,096,539.67	705,118.60	478,111.74	726,078.93	386,950.00			**528,964.66**
110J			443,198.67		377,525.58		320,093.51	**340,020.88**
110K							280,843.80	**280,843.80**
Totals	**1,096,539.67**	**705,118.60**	**446,602.38**	**700,364.37**	**414,103.40**	**214,638.00**	**293,297.97**	**358,327.74**

The resulting output is interactive, allowing the user to change the aggregator function as well as the variable selected for the vals. In this respect this functionality closely matches that found in Excel.

Correlation Coefficients

Correlation coefficients quantify the degree of linear relationship between two variables. Linear relationships can be approximated through a straight line. Correlation coefficients can range from -1 to +1. -1 indicates a perfect negative relationship with all points falling on a straight line. +1 indicates a perfect positive relationship with all points falling on a straight line. Correlation coefficients quantify the extent to which the points in a scatter graph lie near a straight line. Correlation coefficients near zero do not necessarily mean no relationship; the two variables may be related but not linearly related. When multiple variables are involved, correlation coefficients are displayed in a “correlation matrix.”

Creating a correlation matrix using R is a two step process. In the first step, a data frame is defined containing the variables of interest. The second step involves creating the matric using

the “cor” command. For ease of identification the variables involved, it is helpful to order the variables in the data frame with the dependent variable first. For the purposes of assessment, this variable will usually be selling price.

```
contin<-data.frame(SPR, SFLA, YRBLT, LIVACCTR, TOTFIX, GFLA, TLIVAR,
                   GRDFCTVL, TOTLSF, ATTGAR)
cor(contin)
```

The output from this particular command is shown below unformatted and then formatted into an easier to read table.

```
                SPR       SFLA         YRBLT      LIVACCTR      TOTFIX       GFLA
SPR        1.0000000 0.7191408  0.253596235  0.579629217  0.5333368 0.75675820
SFLA       0.7191408 1.0000000  0.100988928  0.570635863  0.6222539 0.52842299
YRBLT      0.2535962 0.1009889  1.000000000 -0.009082778 -0.0176067 0.07410478
LIVACCTR   0.5796292 0.5706359 -0.009082778  1.000000000  0.5833797 0.42011080
TOTFIX     0.5333368 0.6222539 -0.017606704  0.583379721  1.0000000 0.38667595
GFLA       0.7567582 0.5284230  0.074104784  0.420110803  0.3866760 1.00000000
TLIVAR     0.7596153 0.7868652  0.007308577  0.669527950  0.7390629 0.61415197
GRDFCTVL   0.8264674 0.4151086  0.256885227  0.400651069  0.2539892 0.66979877
TOTLSF     0.5284610 0.4895089 -0.086994277  0.397031629  0.3188506 0.46455429
ATTGAR     0.4741917 0.5328562  0.201383288  0.313765447  0.4556365 0.27811247
              TLIVAR   GRDFCTVL       TOTLSF     ATTGAR
SPR        0.759615305 0.8264674  0.52846098 0.4741917
SFLA       0.786865196 0.4151086  0.48950891 0.5328562
YRBLT      0.007308577 0.2568852 -0.08699428 0.2013833
LIVACCTR   0.669527950 0.4006511  0.39703163 0.3137654
TOTFIX     0.739062946 0.2539892  0.31885057 0.4556365
GFLA       0.614151973 0.6697988  0.46455429 0.2781125
TLIVAR     1.000000000 0.4848763  0.45529212 0.4290059
GRDFCTVL   0.484876284 1.0000000  0.41714217 0.2800198
TOTLSF     0.455292124 0.4171422  1.00000000 0.3358576
ATTGAR     0.429005886 0.2800198  0.33585761 1.0000000
```

	SPR	SFLA	YRBLT	LIVACCTR	TOTFIX	GFLA	TLIVAR	GRDFCTVL	TOTLSF	ATTGAR
SPR	1.0000	0.7191	0.2536	0.5796	0.5333	0.7568	0.7596	0.8265	0.5285	0.4742
SFLA	0.7191	1.0000	0.1010	0.5706	0.6223	0.5284	0.7487	0.4151	0.4895	0.5329
YRBLT	0.2536	0.1010	1.0000	-0.0091	-0.0176	0.0741	0.0073	0.2569	-0.0870	0.2014
LIVACCTR	0.5796	0.5706	-0.0091	1.0000	0.5834	0.4201	0.6695	0.4007	0.3970	0.3138
TOTFIX	0.5333	0.6223	-0.0176	0.5834	1.0000	0.3867	0.7391	0.2540	0.3189	0.4556
GFLA	0.7568	0.5284	0.0741	0.4201	0.3867	1.0000	0.6142	0.6698	0.4646	0.2781
TLIVAR	0.7596	0.7869	0.0073	0.6695	0.7391	0.6142	1.0000	0.4849	0.4553	0.4290
GRDFCTVL	0.8265	0.4151	0.2569	0.4007	0.2540	0.6698	0.4849	1.0000	0.4171	0.2800
TOTLSF	0.5285	0.4895	-0.0870	0.3970	0.3189	0.4646	0.4553	0.4171	1.0000	0.3359
ATTGAR	0.4742	0.5329	0.2014	0.3138	0.4556	0.2781	0.4290	0.2800	0.3359	1.0000

One of the characteristics of a correlation matrix is that the top is a mirror image of the bottom. Notice the amount in the cell at the intersection of row ATTGAR and column SPR is the same as the amount at the intersection of row SPR and column ATTGAR.

Graphs

Because R is a scripting language, graphs are created interactively within a window called the R Graphics Device. Once created, they can be modified and saved in one of a number of different formats for later use in reports (e.g. PDF, PNG. JPEG).

Pie Chart

"Bar charts and pie charts depict the distribution of a qualitative variable, such as construction grade, or a quantitative variable based on counts, such as number of bedrooms or bathrooms. In a bar chart, the heights of the bars represent the number of items in the class. In a pie chart, the size of the slices represents the percentage of the items in each class. The slices can be distinguished by colors, patterns, or shades. Both charts present graphically the information contained in a histogram." Fundamentals of Mass Appraisal (IAAO, 2011)

Pie charts are very useful for displaying the relationship between categorical or quality variables such as ratings for grade or physical condition. The process begins by determining the frequency of the occurrences of each of the ratings within the subject dataset. This is accomplished in R using the frequencies function. The following command calculates the number of occurrences for each GRADE level within our dataset.

```
Frequencies<-table(GRADE)
```

This table of frequencies will be used to determine the size of the slices in the pie chart. We can also assign labels by listing them in alphabetical order. If the user is unsure of the ratings present within the dataset, the summary function will provide a list such as the following:

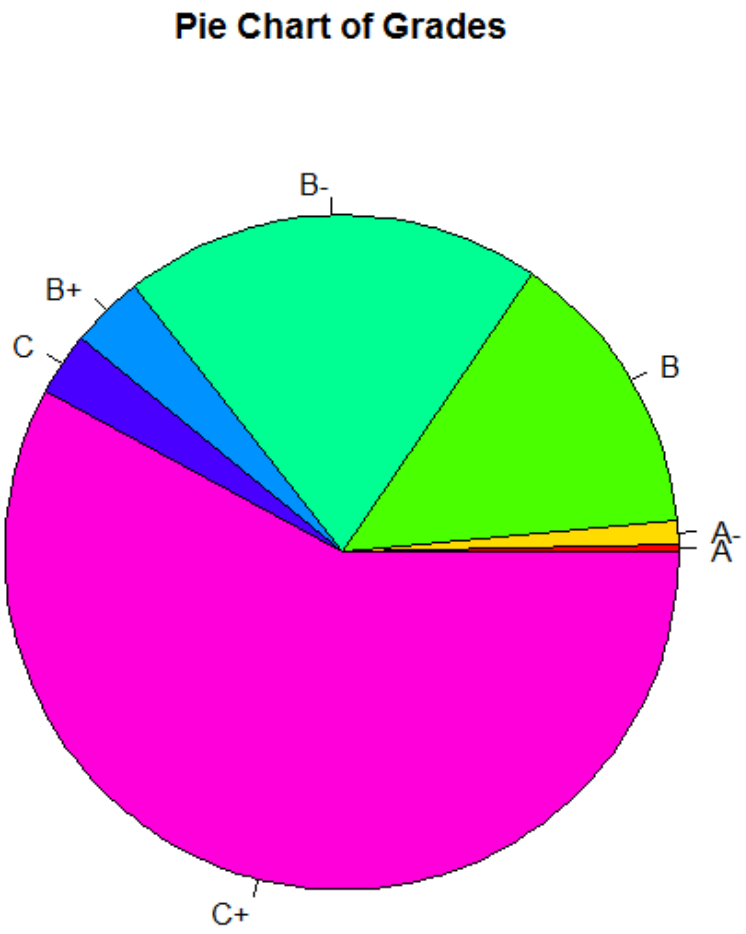
CDU			GRADE		
AV	:	454	A	:	3
EX	:	26	A-	:	10
FR	:	124	B	:	120
GD	:	177	B-	:	173

PR	:	3	B+	:	30
VG	:	77	C	:	26
			C+	:	499

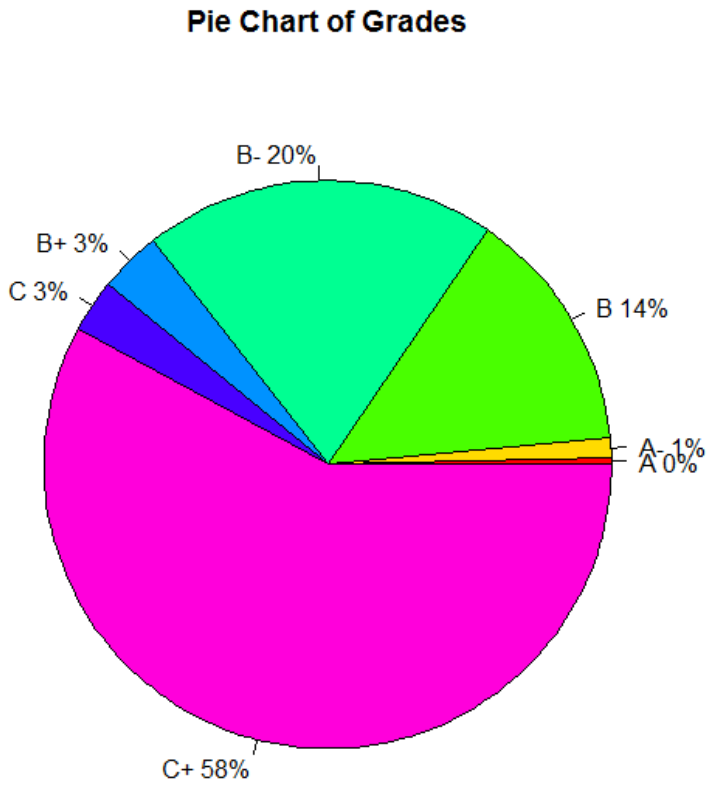
The final command set is shown below.

```
frequencies<-table(GRADE)
slices <- c(frequencies)
lbls <- c("A", "A-", "B", "B-", "B+", "C", "C+")
pie(slices, labels = lbls, col=rainbow(length(lbls)), main="Pie Chart
    of Grades")
```

This is the resulting chart:



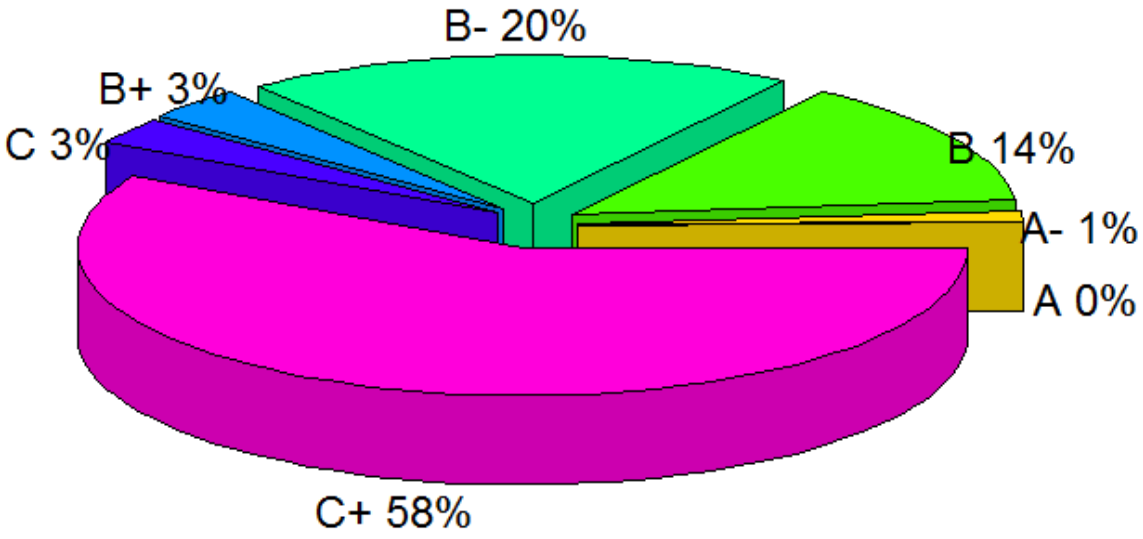
Like other R functions, the user can add more functionality to the pie chart. Two additional lines of code make our chart more understandable.




```
slices <- c(frequencies)
lbls <- c("A", "A-", "B", "B-", "B+", "C", "C+")
pct <- round(slices/sum(slices)*100)
lbls <- paste(lbls, pct) # add percents to labels
lbls <- paste(lbls,"%",sep="") # ad % to labels
pie(slices, labels = lbls, col=rainbow(length(lbls)), main="Pie Chart
    of Grades")
```

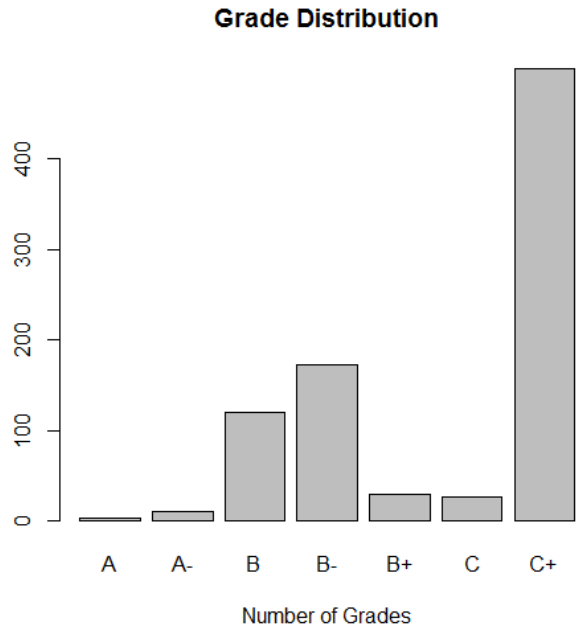
There are also other packages that add functionality, such as the plotrix package. This package allows the user to create a three dimensional chart with the pieces “exploded”

```
library(plotrix)
slices <- c(frequencies)
lbls <- c("A", "A-", "B", "B-", "B+", "C", "C+")
pct <- round(slices/sum(slices)*100)
lbls <- paste(lbls, pct) # add percents to labels
lbls <- paste(lbls,"%",sep="") # ad % to labels
pie3D(slices,labels=lbls,explode=0.1, main=”Pie Chart of Grades”)
```



Bar Charts

Another way of displaying the distribution of data is the bar chart. It is best used with categorical or discrete variables such as quality ratings where each rating is listed along the bar chart axis and the count of the rating is indicated by the height of the bar. A simple vertical bar chart of Grades might look like the following:



This chart, like the pie chart, provides the user with a picture of the relationship between the Grade ratings within the subject dataset.

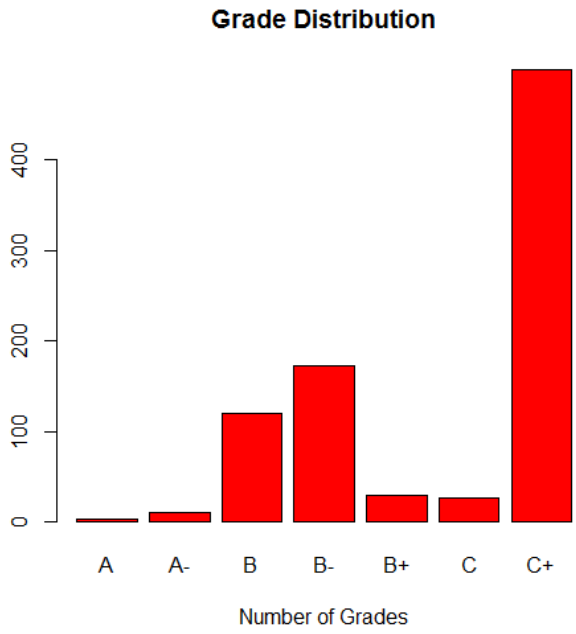
As with other plots in R, there are options to allow the user to change the look and presentation of the chart. This simple bar chart was created using the command:

```
frequencies<-table(GRADE)
barplot(frequencies, main="Grade Distribution", xlab="Number of
Grades")
```

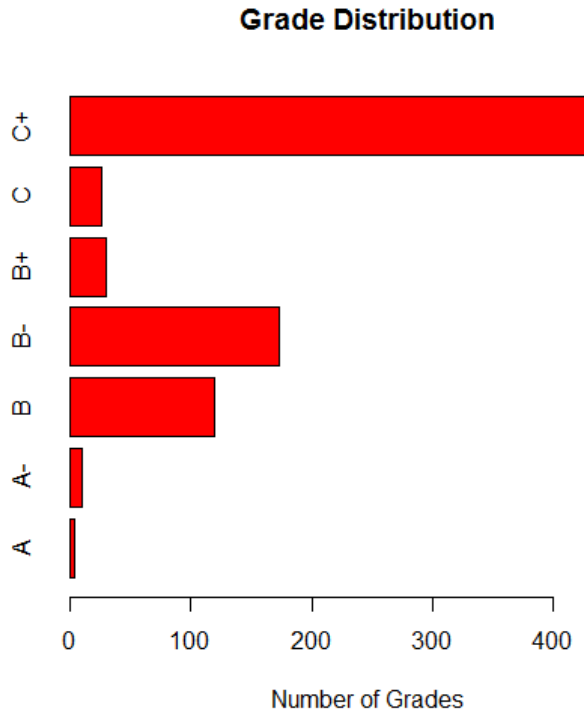
Adding one simple command can change the color.

```
barplot(frequencies, col=”Red”,
main="Grade Distribution",
xlab="Number of Grades")
```

Another command will change the orientation of the bars.



```
barplot(frequencies, col=”Red”, horiz=”True”, main="Grade
        Distribution", xlab="Number of Grades")
```

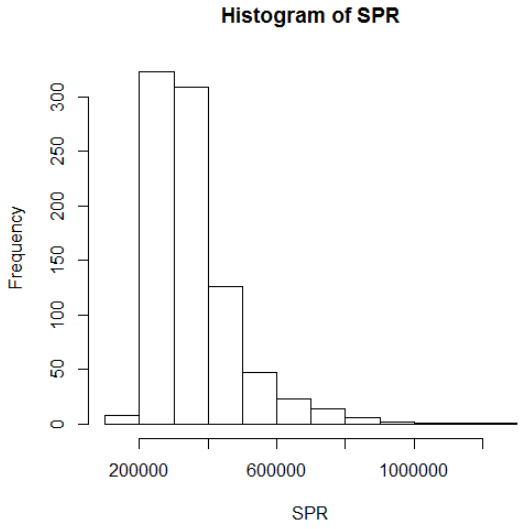


Histograms

A histogram is a bar chart of a continuous variable like lot size or living area. The x-axis of the chart represents values of the variable being analyzed (such as sales prices or sales ratios). The bars and vertical axis represent the frequency of each interval. (Chapter 2, Course 332 Modeling Concepts)

A simple histogram is created using the function “hist()”, where the variable of interest is placed inside the parentheses as follows:

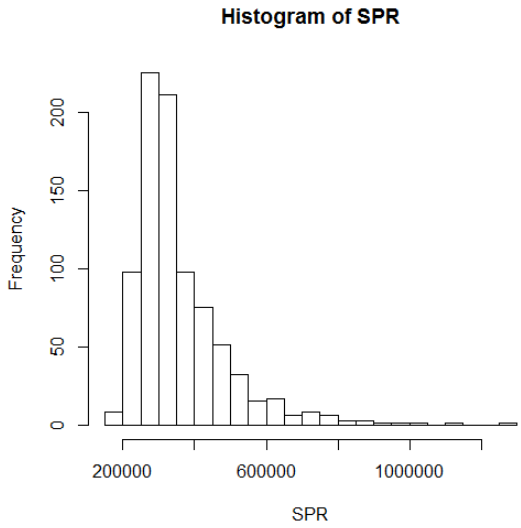
```
hist(SPR)
```



The variable of interest, SPR, is shown as a label below the horizontal axis. Intervals or bins are established along that axis to contain the bars that, in turn, reflect the frequency of occurrences within each bin. The intervals used in the histogram to the left reflect 100,000 intervals and the single largest number of occurrences appears in the interval from 200,000 to 300,000.

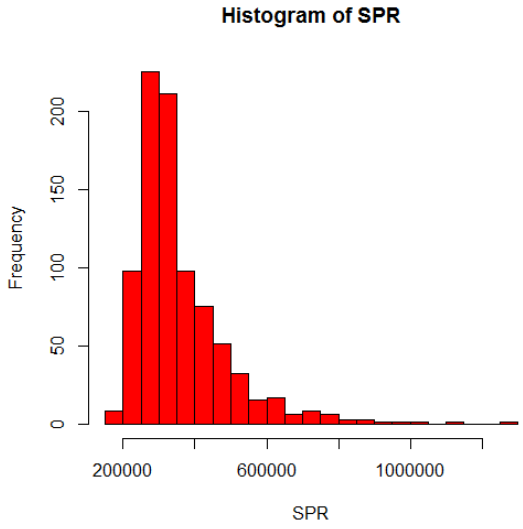
One of the first items subject to change is the number of intervals, which is controlled using the “breaks=” option. We can double the nine intervals shown in the default histogram by adding the breaks option and inserting the number 18.

```
hist(SPR, breaks=18)
```



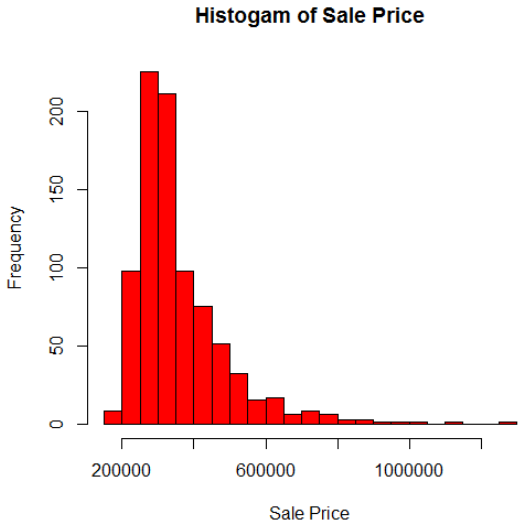
Colors can be added to the histogram using the option “col=”.

```
hist(SPR, breaks=18, col=”red”)
```



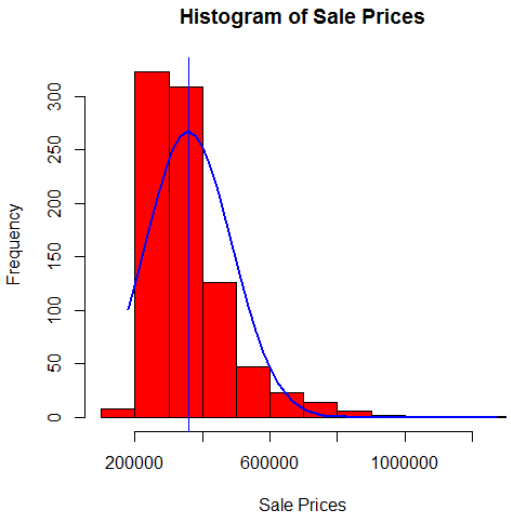
As with all graphs in R histograms can be modified with appropriate axis titles using the "main=", and "xlab=" and "ybal=" commands.

```
plot.new()
hist(SPR, breaks=18, col="red",
     border="black",
     main="Histogam of Sale
     Price", xlab="Sale Price",
     ylab="Frequency")
```



Since the histogram is designed to show the relative distribution of a numeric variable, it seems only natural to have a way of superimposing a normal curve on top of the histogram. The series of commands shown below will accomplish that task.

```
plot.new()
hist(SPR, breaks=15, col="Red",
xlab="Sale Prices", main="Histogam
of Sale Prices", ylab="Frequency")
x<-SPR
xfit<-seq(min(x),max(x),length=50)
yfit<-
dnorm(xfit,mean=mean(x),sd=sd(x))
yfit <-
yfit*diff(h$mids[1:2])*length(x)
lines(xfit, yfit, col="blue", lwd=2)
abline(v=mean(x),col="blue")
```



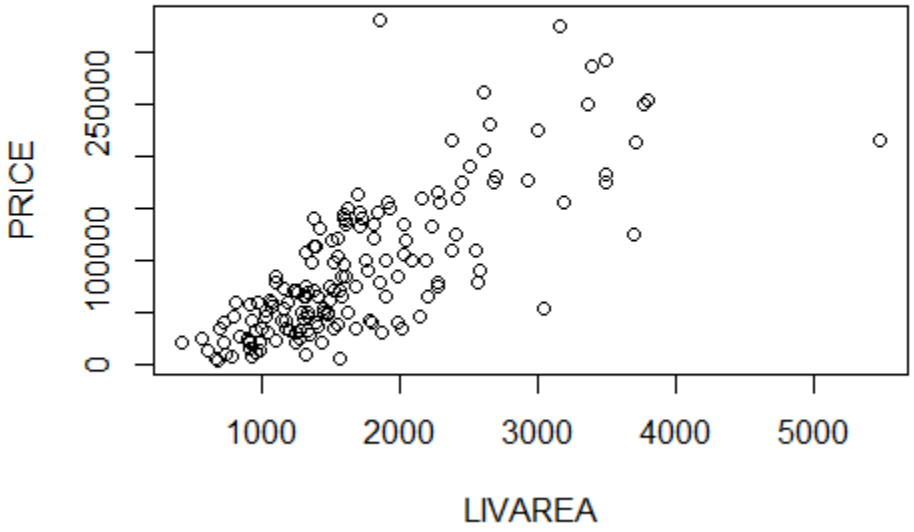
Scatterplot

Scatter graphs or scatter plots show the relationship between two variables graphically. They are particularly good for two quantitative variables. The independent variable goes on the horizontal axis, and the dependent variable goes on the vertical axis. An upward trend indicates a positive relationship and a downward trend indicates a negative relationship.

The commands to create a simple scatter plot of selling prices (PRICE) against square feet of living area (LIVAREA) are shown below:

```
plot.new()
plot(LIVAREA, PRICE)
```

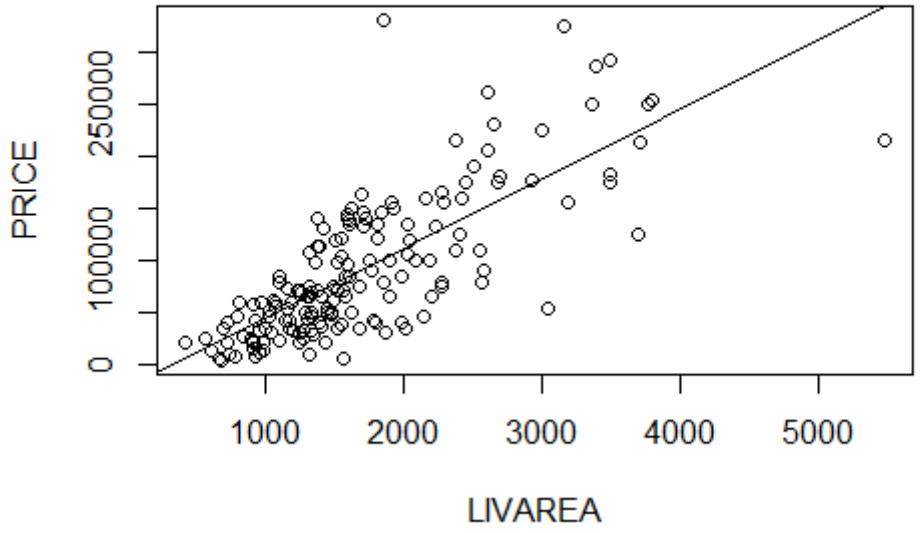
The command function is “plot” and the variable names “LIVAREA” and “PRICE” must appear in the command exactly as they appear in the dataframe (ie “livarea” or “Livarea” will not work). Plot.new() opens a graphics window, if it is not already open, and plot(LIVAREA, PRICE) produces a plot such as that shown below.



This plot can be enhanced by adding a regression line (called a trend line in Excel) through the following command:

```
abline(lm(PRICE~LIVAREA))
```

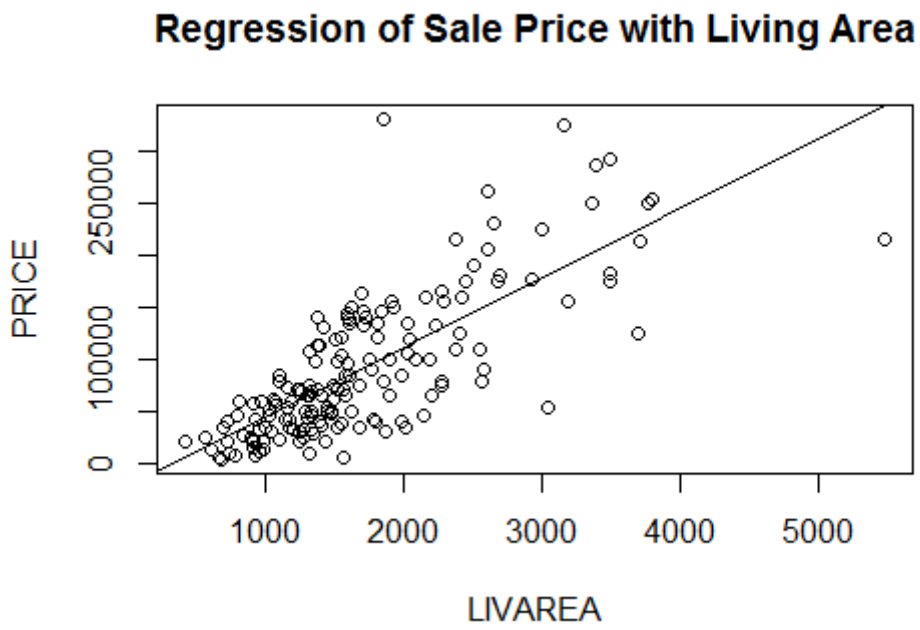
The “lm” command fits a linear model using the variables that follow in parentheses and adds a regression line to our plot.



With one more line we add a title to our plot so the set of commands looks like the following:

```
plot.new()
plot(LIVAREA, PRICE)
abline(lm(PRICE~LIVAREA))
title("Regression of Sale Price with Living Area")
```

and the result is the plot shown below.



The export command in the Plots window of RStudio will allow this graph to be saved in one of the following formats:

PNG

JPEG

TIFF

BMP

Metafile

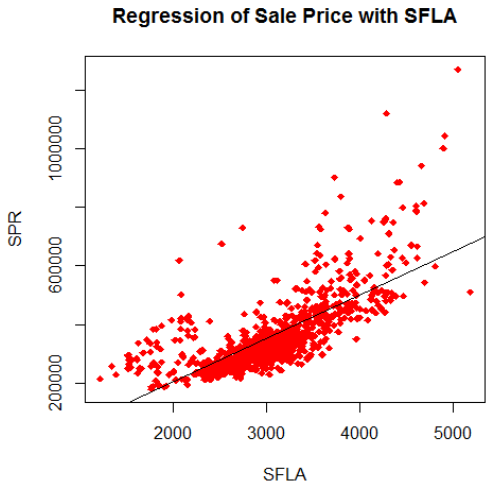
SVG

EPS

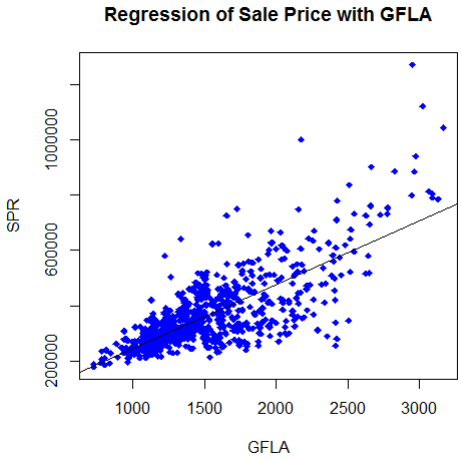
In addition, it can be saved as a pdf or copied to the clipboard to be pasted into another program such as this WORD document.

R Studio saves each plot and allows the user to retrieve and view any plot created during the active session.

Parameters can be set by graph, which may be helpful in differentiating graphs at a later date. For example, the user may change the color of the plot points in a scatter plot to distinguish between the variables SFLA and GFLA.

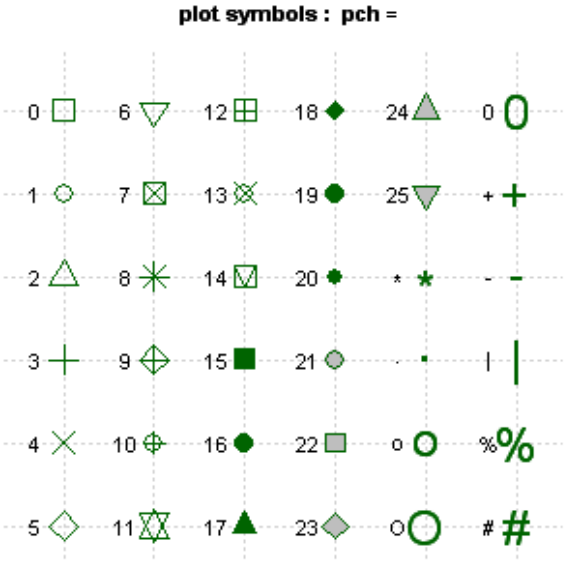



```
plot(SFLA, SPR, pch=18,
     col="red")
abline(lm(SPR~SFLA))
title("Regression of Sale Price
      with SFLA")
```



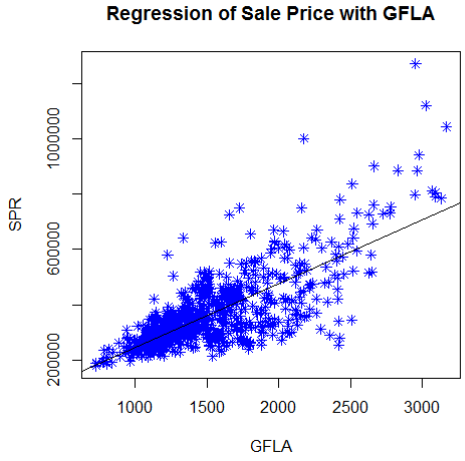

```
plot(GFLA, SPR, pch=18,
     col="blue")
abline(lm(SPR~GFLA))
title("Regression of Sale Price
      with GFLA")
```

Another alternative is to change the format of the plotting symbols. R provides a range of alternatives shown in this table/



Changing the plot symbol is as easy as entering a different number with the pch parameter.

```
plot(GFLA, SPR, pch=5,
     col="blue")
abline(lm(SPR~GFLA))
title("Regression of Sale Price
      with GFLA")
```



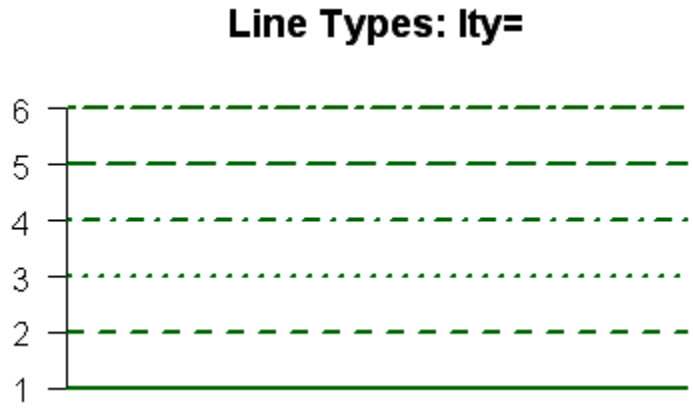

```
plot(GFLA, SPR, pch=8,
     col="blue")
abline(lm(SPR~GFLA))
title("Regression of Sale Price
      with GFLA")
```

The following can be used to control text and symbol size in graphs

option	description
cex	number indicating the amount by which plotting text and symbols should be scaled relative to the default. 1=default, 1.5 is 50% larger, 0.5 is 50% smaller, etc.
cex.axis	magnification of axis annotation relative to cex
cex.lab	magnification of x and y labels relative to cex
cex.main	magnification of titles relative to cex
cex.sub	magnification of subtitles relative to cex

You can change lines using the following options. This is particularly useful for reference lines, axes, and fit lines.

option	description
lty	line type. see the chart below.
lwd	line width relative to the default (default=1). 2 is twice as wide.



Options that specify colors include the following.

option	description
col	Default plotting color. Some functions (e.g. lines) accept a vector of values that are recycled.
col.axis	color for axis annotation
col.lab	color for x and y labels
col.main	color for titles
col.sub	color for subtitles

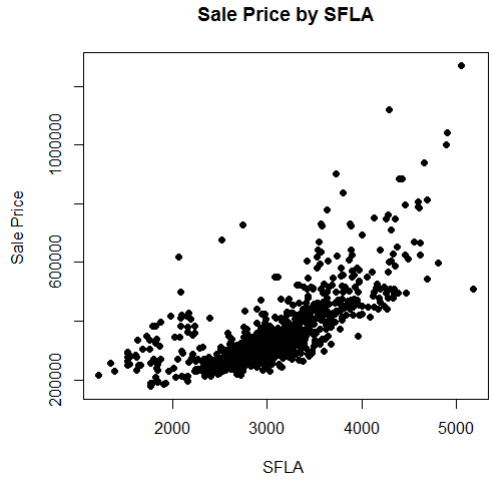
fg	plot foreground color (axes, boxes - also sets col= to same)
bg	plot background color

You can easily set font size and style, but font family is a bit more complicated.

option	**description**
font	Integer specifying font to use for text. 1=plain, 2=bold, 3=italic, 4=bold italic, 5=symbol
font.axis	font for axis annotation
font.lab	font for x and y labels
font.main	font for titles
font.sub	font for subtitles
ps	font point size (roughly 1/72 inch) text size=ps*cex
family	font family for drawing text. Standard values are serif , sans , mono , symbol . Mapping is device dependent.

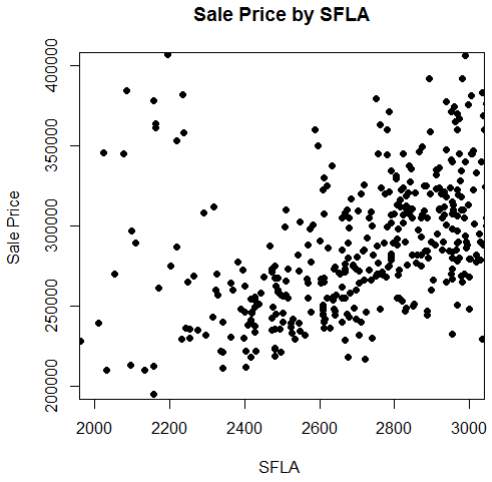
In windows, mono is mapped to TT Courier New , serif is mapped to TT Times New Roman , sans is mapped to TT Arial , mono is mapped to TT Courier New , and symbol is mapped to TT Symbol (TT=True Type). You can add your own mappings.

The user can also change the axis limits from the default. For example the initial scatterplot of Sale Price and Square Feet of Living Area may look like the following:



This graph may be suitable for most uses. However, if the assessor wants to view only a portion of the total extent of living area, the extent of the x (horizontal) axis can be changed using the command xlim=c(). Viewing just the parcels between 2000 and 3000 SFLA would result from the command xlim(2000, 3000).

The graph on the right produces the desired result with regard to SFLA, but the user may now desire to use a similar command to reduce the limits of the y axis.



The user can continue to make changes to the plot interactively until the desired graph picture is achieved.

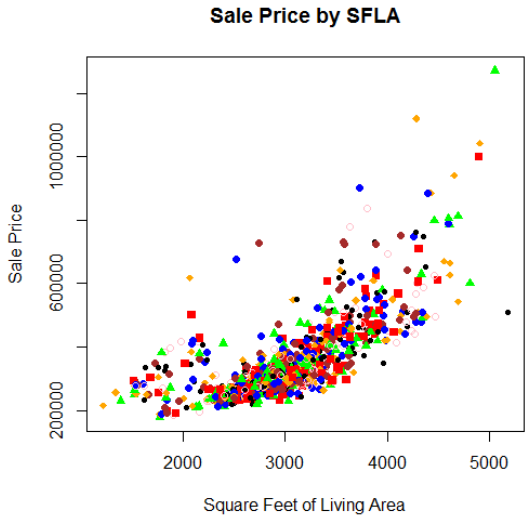
Some of these changes include changing the color and/or shape of the dots. This can be done for every occurrence on the plot or by an identifiable breakdown.

For example, we can use different colors or different shapes for each of the different grade levels in our dataframe.


```
plot.new{}
plot(SFLA, SPR, main="Sale Price
     by SFLA", xlab="Square Feet
     of Living Area",ylab="Sale
     Price", col=c("red", "blue",
     "green", "orange", "brown",
     "black", "pink"))
```

```
plot.new{}
plot(SFLA, SPR, main="Sale Price
     by SFLA", xlab="Square Feet
     of Living Area", ylab="Sale
     Price",
     pch=c(15,16,17,18,19,20,21))
```

Or both



Finally, we can add a legend by specifying what it should contain and where it is to be located on the plot canvas.

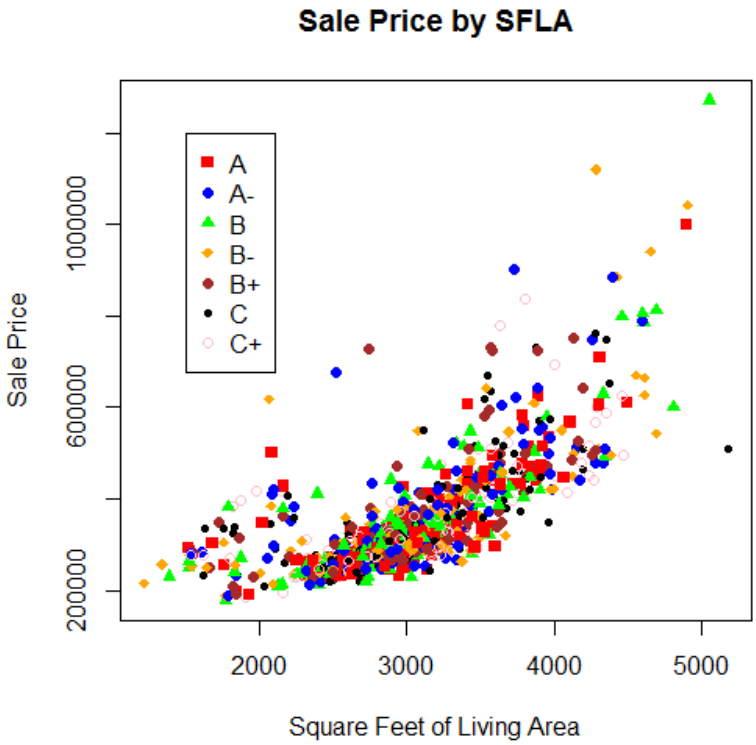
```
plot.new()
plot(SFLA, SPR, main="Sale Price by SFLA", xlab="Square Feet of
     Living Area", ylab="Sale Price", col=c("red", "blue", "green",
     "orange", "brown", "black", "pink"),
     pch=c(15,16,17,18,19,20,21))
legend(1500,1200000, c("A","A-", "B", "B-", "B+", "C", "C+"),
       pch=c(15,16,17,18,19,20,21), col=c("red", "blue", "green",
       "orange", "brown", "black", "pink"))
```

The “legend()” function tells R what is to be created. The numbers 1500 and 1200000 are the coordinates within the plot canvas to place the legend. The upper left corner of the legend box is located at position 1500 on the x or horizontal axis and position 1,200,000 on the y or vertical axis.

The next command [c(A , A- , B , B- , B+ , C , C+)] contains the legend text. The summary() command will provide information on, in this case, the number of occurrences of each Grade level as well as their order.

The “pch=c()” command establishes the shapes to be used and the “col=c()” command establishes the colors.

The result is a scatterplot that displays the relationship between sale price and living area according the grade.



Line Graphs

Line graphs provide summary plots of a quantitative variable versus a non-continuous variable such as number of baths or construction year range. The y-axis should contain the summary variable. The x-axis should contain the other variable. The most common continuous variable plotted by assessors is selling price and one of the most helpful noncontinuous variables to plot against selling price is time expressed as months or years. In order to plot a single line representing the relative change in selling prices over time, it is necessary to express selling price in a summary form per time period, such as average selling price per month.

We will begin this process by defining a variable called “months” that will actually represent the number of months back from the January 1, 2013 appraisal date represented by each sale date.

```
months<-(2013*12)-((SYR*12)+SMO)
```

Next, we aggregate selling price by that months variable using the mean function.

```
avgsl<-aggregate(x=SPR, by=list(months), FUN="mean")
```

The new variable is “avgsl” which represents the average selling price at each instance of the “months” variable previously defined.

Finally, we use the plot command in R to create a simple line graph with appropriate labels.


```
plot(avgsl, type="o",
     col="black",
     main="Average Sale Price
     by Month",
     xlab="Months",
     ylab="Average Sale
     Price")
```

By changing the aggregate function, the assessor can create multiple graphs using selling price and months.

```
maxsl<-aggregate(x=SPR, by=list(months), FUN="max")
minsl<-aggregate(x=SPR, by=list(months), FUN="min")
medsl<-aggregate(x=SPR, by=list(months), FUN="median")
```

In addition the average or mean, the assessor can display the maximum, minimum or median selling prices for each of the months in the time period. An alternative would be to display several of the aggregate variables on the same graph using a combination of the plot and lines commands as show below.

```
plot(maxsl,  ylim=range(SPR), type="o", col="blue", main="Sale
     Price by Month", xlab="Months", ylab="Average Sale")
lines(medsl, type="o", pch=22, col="red")
lines(minsl, type="o", pch=22, col="black")
legend(10,1300000, c("Maximum", "Median", "Minimum"),
     lty=c(1,1,1), lwd=c(2.5,2.5),col=c("blue", "red", "black"))
```

Each of the “lines” commands adds a line to the graph using the aggregated variable shown in the command. Using different colors for the lines and a legend helps the use quickly distinguish each function.

The “ylim” command sets the extent of the “y” or vertical axis as the range of selling prices. Running the “summary” command for the dataframe would provide the user with the dollar amounts for the minimum and maximum selling prices.

The “legend” command begins with the position of the legend using measures from the x and y axes. The command 10,1300000 describes the upper left corner of the legend box being set and the intersection of and x value of 10 and a y value of 1,300,000. Users should be careful to be consistent in assigning colors to match the colors of the lines being represented in the legend.

Box Plots

Another useful tool for displaying variable such as selling price is the box plot. “A box plot shows graphically the distribution of a continuous variable with a qualitative variable (or quantitative variable with limited data values).” Fundamentals of Mass Appraisal, page 127. The boxes represent the middle 50 percent of the data, termed the inter-quartile range (IQR). The horizontal lines toward the middle of the boxes represent median values. Values more than 1.5 IQRs above or below the boxes constitute outliers. Values more than 3 IQRs above or below the boxes constitute extreme outliers.

Using selling price and the months variable previously calculated, the following boxplot command produces a graph showing the extent of selling prices by month.

```
boxplot(SPR~months,data=sample, main="Sale Price by Month",
        xlab="Months", ylab="Selling Prices")
```

Each separate month back from the appraisal date is shown on the horizontal or X axis while selling prices are displayed on the vertical or Y axis.

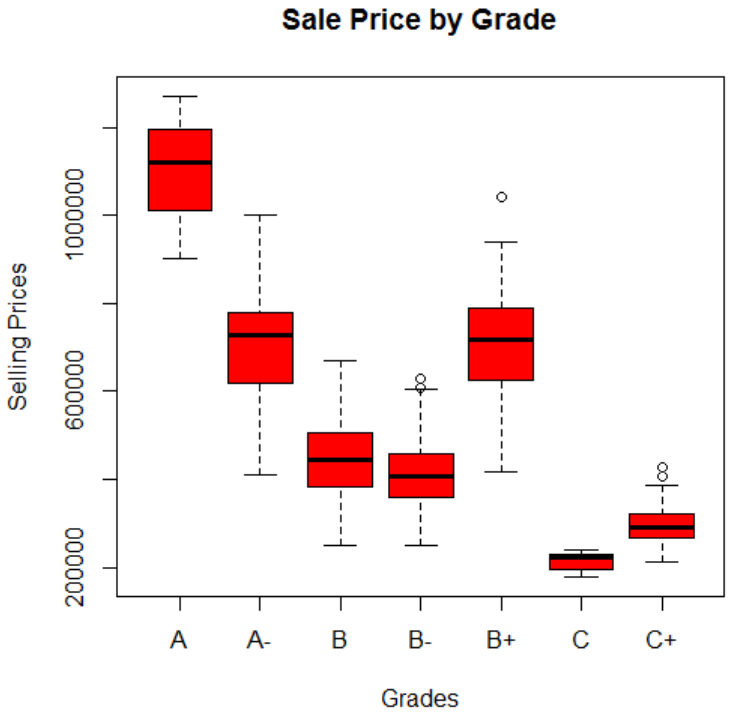
The horizontal bar in each box represents the median for the continuous variable, selling price, while the box represents the interquartile range. That is the range from the 25th percentile or first quartile to the 75th percentile or third quartile.

The vertical dashed lines that extend from the bottom and top of the box ending in horizontal lines are called whiskers. They represent data that is no more than 1.5 times the interquartile range from either the top or bottom of the box.

The circles that display beyond the whiskers are outliers. A boxplot provides the user with a visual representation of ranges of the continuous variable against some qualitative or categorical variable. In the chart above the continuous variable is selling price and the categorical variable is months.

A more useful comparison with selling price is either grade or condition. The graph below uses grade against selling price and adds some color for visual appeal.

R selects the order in which the grades are shown across the horizontal axis, so the user will have to adjust to that. However, consider the position of the median in the boxes for the A grade versus the A-. Then move to the B+, then the B, the C+ and the C. The progression downward in selling price appears to reflect a downward trend that would be anticipated with a lowering of the grade.



Another situation when a boxplot is very useful is in the evaluation of the results of an appraisal to sales price ratio study. A perfect appraisal situation would have the median ratios for every grade classification resting on a horizontal line that intersected the vertical axis at the 1.00 (100%) ratio level. A boxplot like the one below quickly shows the user the grade categories (here shown as grade factors) where there are problems. The syntax that produced the plot includes the command producing the horizontal reference line.

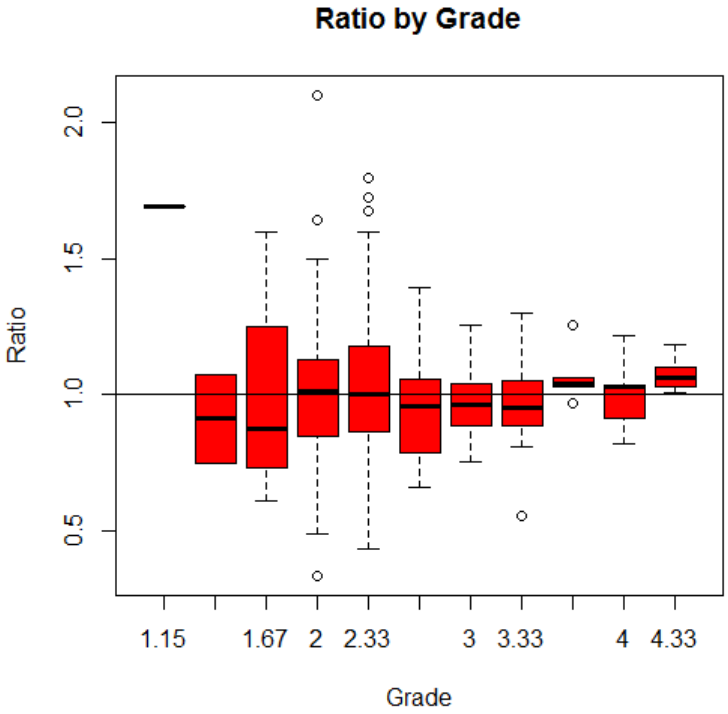
```
boxplot(Ratio~Grade,data=ratdata, col="red", main="Ratio by
        Grade", xlab="Grade", ylab="Ratio")
abline(h="1.00", col="black")
```

The continuous variable is “Ratio” and the categorical variable is “Grade”.

The dataframe used to produce the boxplot is called “ratdata”.

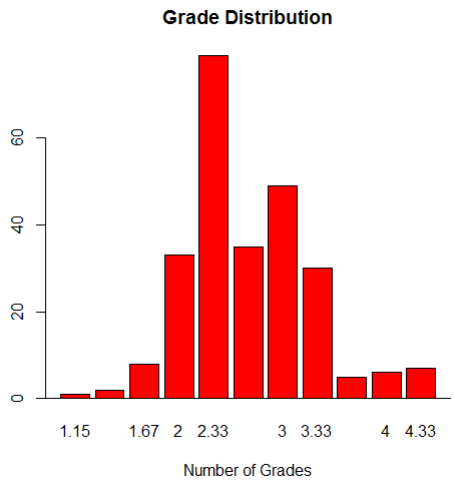
The horizontal reference line is created using the “abline” command.

The grade factors reflect the relative numeric value of each quality level. While many of the boxes overlap the horizontal reference line, few of the horizontal bars within those boxes fall on or very near that line. If they did, it would indicate the median ratio for that grade factor was equal to 100%. Assuming that a grade factor of 3.0 reflects average, the relative size of the boxes indicates there are more sales in this sample that reflect below average grades.



A combination of a frequencies and a barplot command will further verify the user's suspicion concerning the numbers of grades.

```
frequencies<-table(Grade)
barplot(frequencies, col="Red", main="Grade Distribution",
        xlab="Number of Grades")
abline(h="0.00", col="black")
```



Combining these two plots clearly shows how grades are occurring in this particular sample. The assessor can use that to help in further calibration of the valuation model with respect to the quality variable grade.

Plots are useful adjunct to any tabular representation of data.

Data Transformations

The variables used in model building are constructed from property data. A simple, or untransformed, variable is one that directly represents a data element, for example, lot size, square feet of living area, or number of bedrooms. Some data characteristics, particularly quantitative ones, can often be used directly in model building. More often, however, mass appraisal models can be improved by transforming property characteristics to better capture their contribution to value. In fact, since only numeric data can be used in modeling, alphanumeric data, sometimes also referred to as string data, must be transformed into numeric variables for modeling. (Fundamentals of Mass Appraisal, page 96).

Binary Transformations

Binary variables reflect the presence or absence of a feature, attribute or condition. If the feature is present, the variable is coded s 1. If the feature is not present, the variable is coded as 0. Binary variables let the model determine the contribution of the feature to value.

Grade is a good candidate for a binary transformation when the original code is alphanumeric (A+, A, A-. B+, B, B-, C+, C, C-, D+, D, D-). The existence of each of the grade levels can be verified and their contributory value determined by transforming them into binary variables.

```
sample$ADUM<-factor ( with ( sample, ifelse ( ( GRADE == 'A' ), 1 , 0 ) ) )
sample$AMDUM<-factor ( with ( sample, ifelse ( ( GRADE == 'A-' ), 1 , 0 ) )
) sample$BDUM<-factor ( with ( sample, ifelse ( ( GRADE == 'B' ), 1 , 0 ) )
) sample$BMDUM<-factor ( with ( sample, ifelse ( ( GRADE == 'B-' ), 1 , 0 )
) ) sample$BPDUM<-factor ( with ( sample, ifelse ( ( GRADE == 'B+' ), 1 , 0
) ) ) sample$CDUM<-factor ( with ( sample, ifelse ( ( GRADE == 'C' ), 1 , 0
) ) ) sample$CPDUM<-factor ( with ( sample, ifelse ( ( GRADE == 'C+' ), 1 ,
```

In the formulae above new variables are being assigned to the sample data frame using the command “sample$”. The “ifelse” command sets up a test of whether the GRADE variable is equal to any of the characters shown. When it is equal, the new variable is given a value of 1, otherwise the value is 0.

Running the summary command will yield the following information for the original variable and the newly transformed binary or dummy variables.

GRADE			ADUM			BDUM			CDUM			AMDUM			BMDUM			BPDUM			CPDUM		
A	:	3	0	:	858	0	:	741	0	:	835	0	:	851	0	:	688	0	:	831	0	:	362
A-	:	10	1	:	3	1	:	120	1	:	26	1	:	10	1	:	173	1	:	30	1	:	499
B	:	120																					
B-	:	173																					
B+	:	30																					
C	:	26																					
C+	:	499																					

Note the frequency indicated in the first box on the left is matched by the frequency of the value of 1 given to each of the binary or dummy variables. There are 173 occurrences of the B- Grade and the value of 1 for the BMDUM variable.

Exponential Transformations

Raising one property characteristic to a power in R is relatively simple. The formula is shown below.

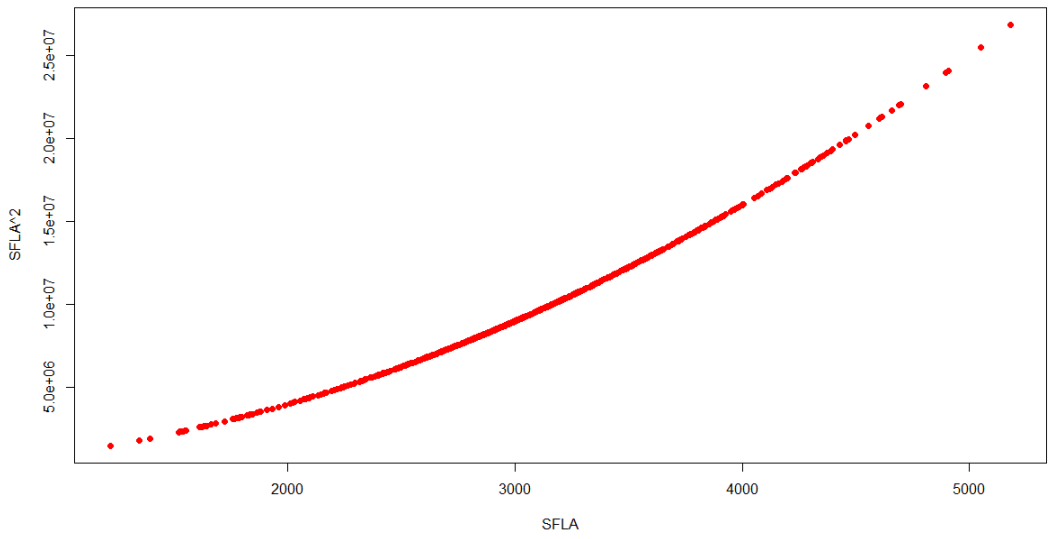
```
NewVariable<-exp(original variable)^some exponent
```

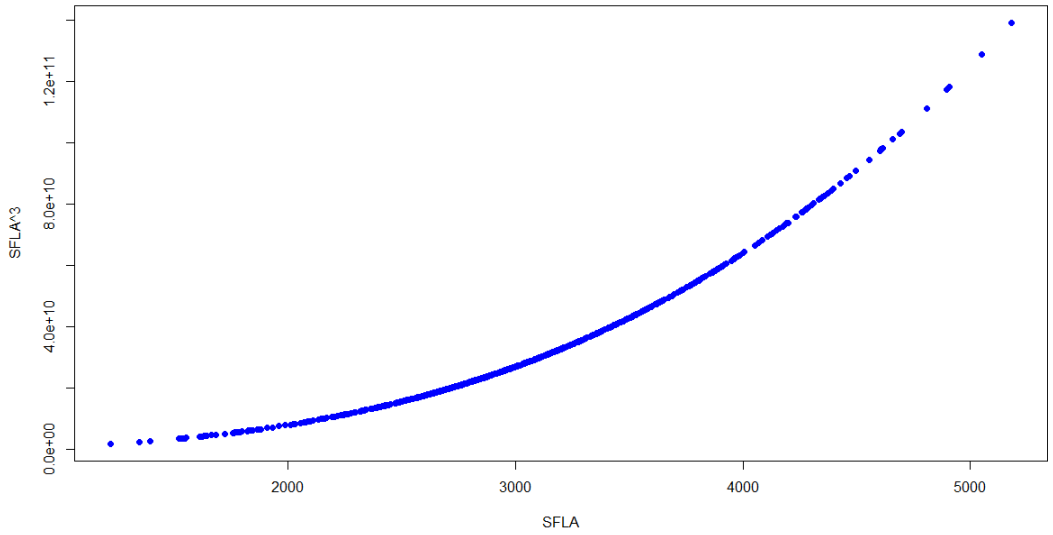
For example, raising square feet of living area to the power of 3 is accomplished using the following;

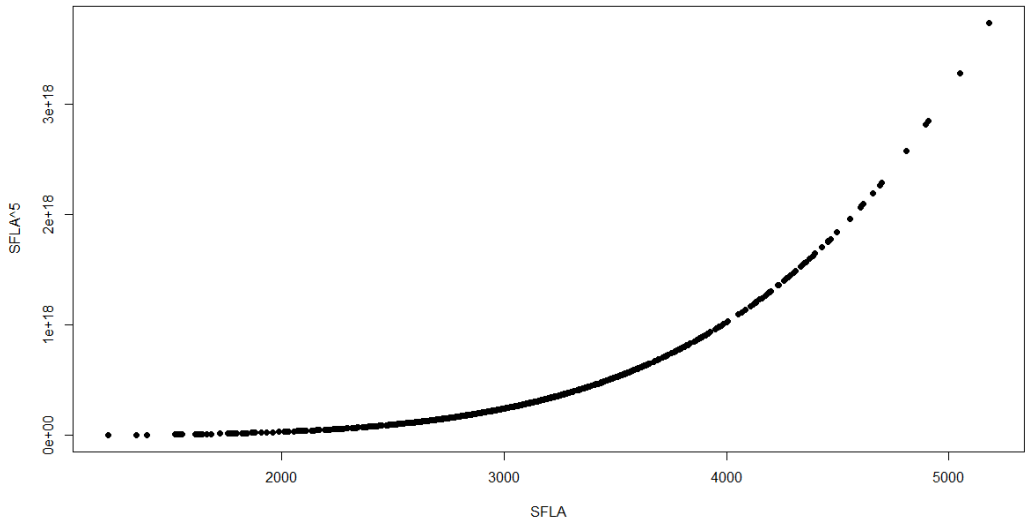
```
SFLA3<-(SFLA)^3
```

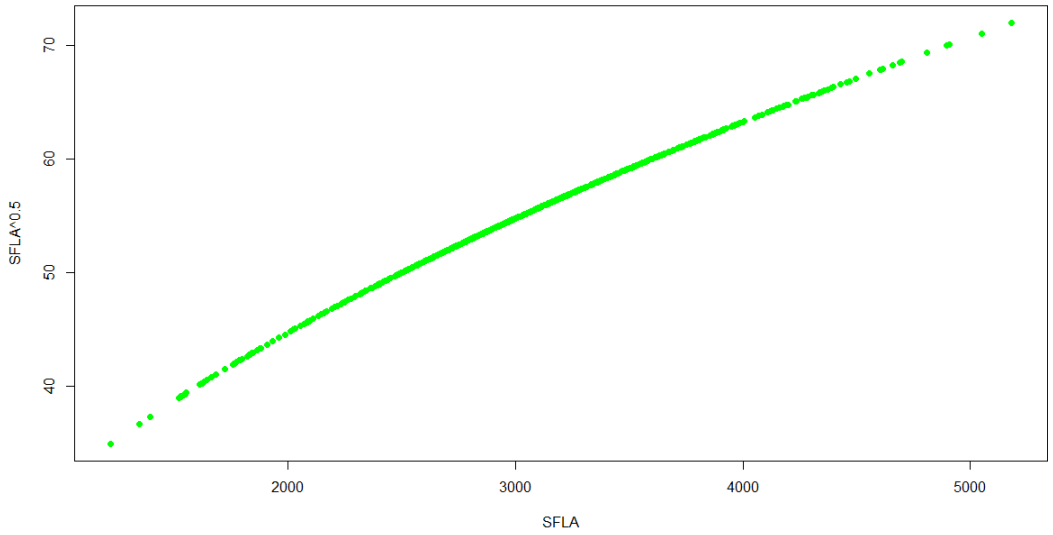
The purpose for any transformation is to replicate changes from one parcel to another that are occurring in the market. These are easier to visualize through the use of charts such as those shown below. Notice the change in the curvature of the lines.

```
Sample$SFLA3<-(SFLA)^3
Sample$SFLA2<-(SFLA)^2
Sample$SFLA5<-(SFLA)^5
plot(sample$SFLA, sample$SFLA2, xlab='SFLA', ylab='SFLA^2', col='red',
     pch=19)
plot(sample$SFLA, sample$SFLA3, xlab='SFLA', ylab='SFLA^3',col='blue',
     pch=19)
plot(sample$SFLA, sample$SFLA5, xlab='SFLA', ylab='SFLA^5',
     col='black', pch=19)
```










```
sample$SFLA.5<-(SFLA)^0.5
plot(sample$SFLA, sample$SFLA.5, xlab='SFLA', ylab='SFLA^0.5',
     col='Green', pch=19)
```

Logarithmic Transformations

A logarithm is an exponent, namely the exponent to which a base value must be raised to obtain a given number. Common logarithms have the base 10. Natural logarithms have the base 2.71828. Natural logarithms have the property that each doubling of value increases the corresponding logarithm by the same amount, namely 0.6934.

Transformations in R are accomplished using the log(x) or log10(x) commands.

```
sample$logsfla<-log(SFLA)
sample$log10sfla<-log10(SFLA)
```

Polynomial Transformations

According to the reference manual for IAAO Course 332 polynomial transformations involve raising a variable to multiple powers, one of which is 1.00. They are useful for capturing nonlinear trends that may change direction. A quadratic function has one ben and can accommodate one change in direction. A cubic function has two bends and can accommodate two changes in direction.

R uses this transformation in the process of describing a linear model.

```
fit2b <- lm(sample$SPR ~ poly(sample$SFLA, 2, raw=TRUE))
fit3b <- lm(sample$SPR ~ poly(sample$SFLA, 3, raw=TRUE))
```

The output from either or both of these operations may be viewed through the summary command.

```
> summary(fit2b)

Call:
lm(formula = sample$SPR ~ poly(sample$SFLA, 2, raw = TRUE))

Residuals:
    Min      1Q  Median      3Q     Max
-453446  -39624  -12505   23068  503814

Coefficients:
                                  Estimate Std. Error t value Pr(>|t|)
(Intercept)                      5.843e+05  4.194e+04   13.93   <2e-16 ***
poly(sample$SFLA, 2, raw = TRUE)1 -3.075e+02  2.709e+01  -11.35  <2e-16 ***
poly(sample$SFLA, 2, raw = TRUE)2  7.346e-02  4.313e-03   17.03   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 77370 on 858 degrees of freedom
Multiple R-squared:  0.6391,   Adjusted R-squared:  0.6383
F-statistic: 759.8 on 2 and 858 DF,  p-value: < 2.2e-16

> summary(fit3b)

Call:
lm(formula = sample$SPR ~ poly(sample$SFLA, 3, raw = TRUE))

Residuals:
    Min      1Q  Median      3Q     Max
-431737  -39626  -12203   23812  503753

Coefficients:
                                  Estimate Std. Error t value Pr(>|t|)
(Intercept)                      6.776e+05   1.208e+05   5.611 2.72e-08 ***
poly(sample$SFLA, 3, raw = TRUE)1 -4.070e+02   1.238e+02  -3.286  0.00106 **
poly(sample$SFLA, 3, raw = TRUE)2  1.070e-01  4.098e-02   2.611  0.00918 **
poly(sample$SFLA, 3, raw = TRUE)3 -3.595e-06  4.367e-06  -0.823  0.41060
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 77380 on 857 degrees of freedom
Multiple R-squared:  0.6394,   Adjusted R-squared:  0.6382
F-statistic: 506.6 on 3 and 857 DF,  p-value: < 2.2e-16
```

Multiplicative Transformations

Multiplicative transformations, also known as cross-product transformations, involve the multiplication of two or more variables. Multiplicative transformations are useful for capturing the interactive or synergistic effect of certain variables upon value.

One of the most common multiplicative transformations involves the combination of a quality rating with a size variable, such as grade factor and square feet of living area.

```
Newvariable<-QualityRating * SFLA
SFGRD<-GRDFCTVL*SFLA
```

Quotient Transformations

Quotient transformations are created by dividing one variable by another (such as assessed value by sale price). In mass appraisal, quotient transformations are used to convert sales prices, rents, and expenses to a per unit basis. Sale price divided by living area is a common implementation of this transformation.

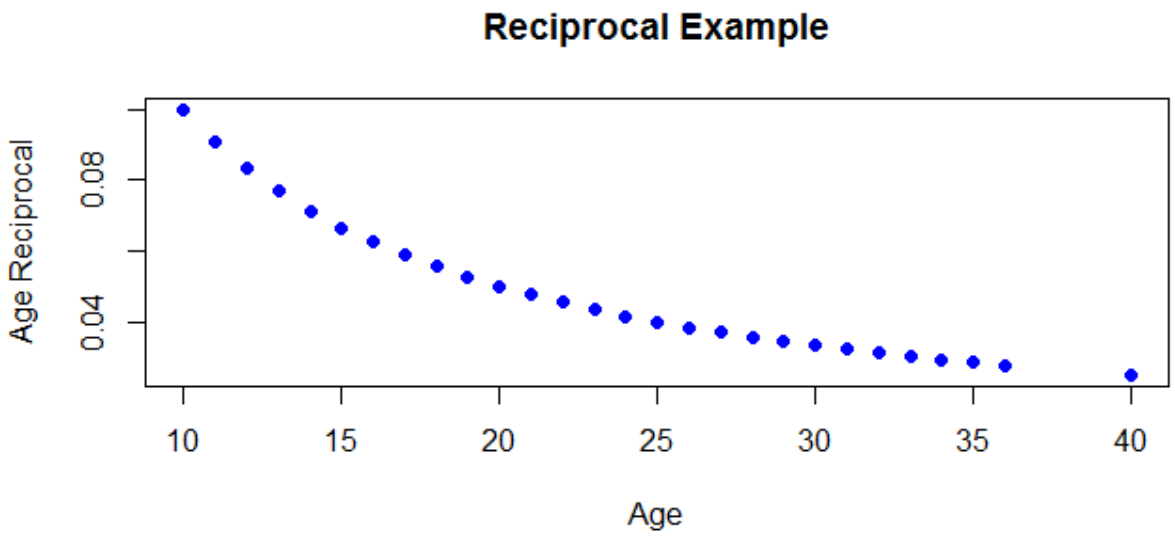
```
Newvariable<-SellingPrice / SFLA
PSF<-SPR/SFLA
```

Reciprocal Transformations

Reciprocal transformations are obtained by dividing 1 by a variable. As the value of the variable increases, its reciprocal decreases. In mass appraisal, reciprocal transformations can be used to capture prosimity influences, such as distance to the city center, beach, or light rail.

```
Newvariable<-1 / OriginalVariable
AGREC<-1/AGE
```

The scatter diagram below illustrates the results of this transformation by comparing the transformed variable to the original. More weight is given to the younger properties.



Multiple Regression Analysis

Generating descriptive statistics from the data set in R is very simple. The command is "summary(sample)" where summary is the r function and sample is the data set being used. The difficulty comes in reading the output. It is not formatted as nicely as SPSS and it combines summary statistics on the numeric variables with frequencies on the string variables.

NBHD		SMO		SYR		SPR	
110I	159	Min	1.000	Min	2011	Min	179,000
110C	147	1st Qu	3.000	1st Qu	2011	1st Qu	276,900
110K	146	Median	6.000	Median	2011	Median	321,111
110G	118	Mean	5.995	Mean	2011	Mean	358,328
110J	112	3rd Qu	8.000	3rd Qu	2011	3rd Qu	404,850
110	94	Max	12.000	Max	212	Max	1,270,000
(Other):	85						

STRYGHT		EXTWL		STYL		ROOF	
Min	1.000	Min	1.000	Min	2.000	Min	1.000
1st Qu	2.000	1st Qu	1.000	1st Qu	4.000	1st Qu	1.000
Median	2.000	Median	1.000	Median	4.000	Median	1.000
Mean	1.7941	Mean	1.508	Mean	4.743	Mean	2.383
3rd Qu	2.000	3rd Qu	1.000	3rd Qu	4.000	3rd Qu	4.000
Max	2.000	Max	7.000	Max	16.000	Max	6.000

YRBLT		FOUND		BSMT		HTCEN	
Min	1973	Min	2.000	Min	4.000	Min	4
1st Qu	1998	1st Qu	2.000	1st Qu	4.000	1st Qu	4
Median	2002	Median	2.000	Median	5.000	Median	4
Mean	2000	Mean	2.003	Mean	4.915	Mean	4
3rd Qu	2002	3rd Qu	2.000	3rd Qu	6.000	3rd Qu	4
Max	2003	Max	5.000	Max	6.000	Max	4

FUEL		HTSYS		LIVACCTR		LIVACCBD	
Min	1	Min	1	Min	4.000	Min	2.000
1st Qu	1	1st Qu	1	1st Qu	8.000	1st Qu	4.000
Median	1	Median	1	Median	9.000	Median	4.000
Mean	1	Mean	1	Mean	8.933	Mean	4.115
3rd Qu	1	3rd Qu	1	3rd Qu	10.000	3rd Qu	4.000
Max	1	Max	1	Max	15.000	Max	7.000

LIVACCFM		PLMBFB		PLMBHB		PLMBADD	
Min	0.000	Min	2.000	Min	0.000	Min	2.000
1st Qu	1.000	1st Qu	3.000	1st Qu	1.000	1st Qu	5.000
Median	1.000	Median	3.000	Median	1.000	Median	5.000
Mean	0.971	Mean	3.249	Mean	1.013	Mean	5.058
3rd Qu	1.000	3rd Qu	4.000	3rd Qu	1.000	3rd Qu	5.000
Max	2.000	Max	6.000	Max	3.000	Max	8.000

PLMBTFX		ATTIC		FLORCOV		INTWL	
Min	10.00	Min	1.00	Min	1	Min	1
1st Qu	16.00	1st Qu	1.00	1st Qu	1	1st Qu	1
Median	16.00	Median	1.00	Median	1	Median	1
Mean	16.83	Mean	1.41	Mean	1	Mean	1
3rd Qu	19.00	3rd Qu	1.00	3rd Qu	1	3rd Qu	1
Max	25.00	Max	5.00	Max	1	Max	1

PHYCOND		GFLA		CDU		BIGAR	
Min	2.000	Min	728	AV	454	Min	0
1st Qu	3.000	1st Qu	1192	EX	26	1st Qu	0
Median	3.000	Median	1384	FR	124	Median	0
Mean	2.995	Mean	1494	GD	177	Mean	0
3rd Qu	3.000	3rd Qu	1701	PR	3	3rd Qu	0
Max	3.000	Max	3167	VG	77	Max	0

MASFPST		MASFPOP		METFP		BSGAR	
Min	0.000	Min	0.0000	Min	0.000	Min	0.0000
1st Qu	0.000	1st Qu	0.0000	1st Qu	1.000	1st Qu	0.0000
Median	0.000	Median	0.0000	Median	1.000	Median	0.0000
Mean	0.0813	Mean	0.1022	Mean	1.467	Mean	0.00813
3rd Qu	0.0000	3rd Qu	0.0000	3rd Qu	2.000	3rd Qu	0.0000
Max	2.000	Max	4.0000	Max	4.000	Max	2.0000

TLIVAR		GRDFCTVL		TOTLSF		ACRES	
Min	1644	Min	1.000	Min	5558	Min	0.1276
1st Qu	2872	1st Qu	1.080	1st Qu	9619	1st Qu	0.2208
Median	3269	Median	1.080	Median	11584	Median	0.2659
Mean	3479	Mean	1.195	Mean	12548	Mean	0.8039
3rd Qu	3967	3rd Qu	1.240	3rd Qu	14572	3rd Qu	0.3351
Max	7159	Max	2.300	Max	43092	Max	444.4400

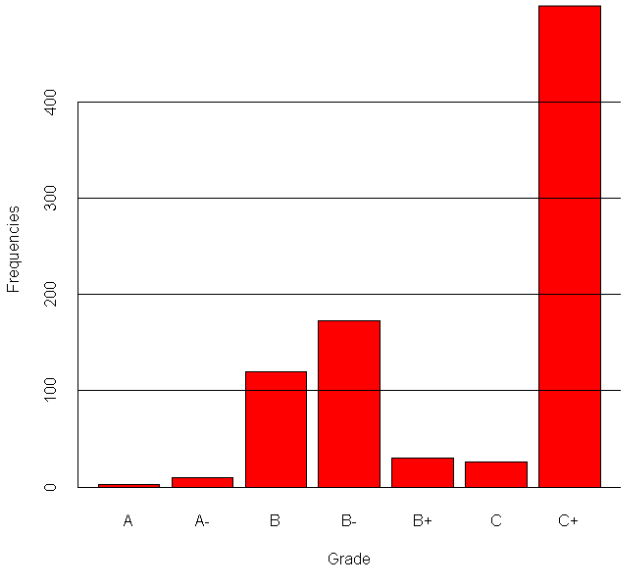
RECRM		UNFIN		FBLA		SFLA	
Min	0.000	Min	0	Min	0.0	Min	1220
1st Qu	0.000	1st Qu	0	1st Qu	0.0	1st Qu	2677
Median	0.000	Median	0	Median	0.0	Median	3010
Mean	3.149	Mean	0	Mean	429.3	Mean	3050
3rd Qu	0.000	3rd Qu	0	3rd Qu	850.0	3rd Qu	3405
Max	661.000	Max	0	Max	2500.0	Max	5182

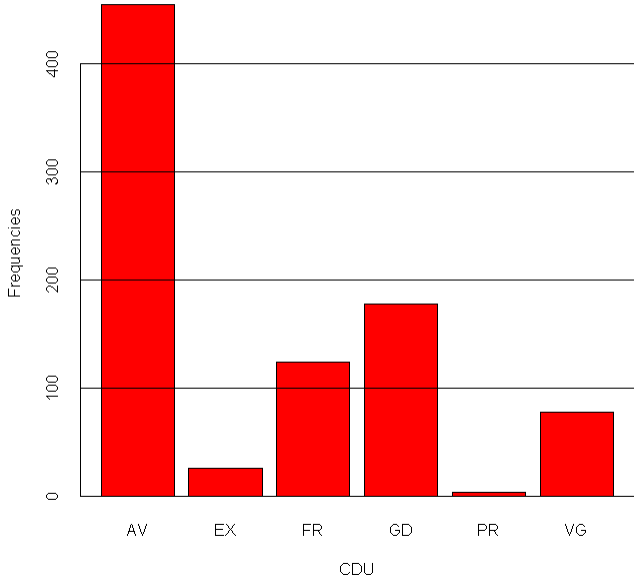
BSMTSF		FIRST		SECOND		UPPER	
Min	0.0	Min	783	Min	0	Min	0
1st Qu	0.0	1st Qu	1268	1st Qu	1328	1st Qu	0
Median	80.0	Median	1467	Median	1572	Median	0
Mean	590.4	Mean	1575	Mean	1475	Mean	0
3rd Qu	1254.0	3rd Qu	1773	3rd Qu	1757	3rd Qu	0
Max	2425.0	Max	3578	Max	2676	Max	0

LAND		BLDG		TOTAL		GRADE	
Min	0	Min	0	Min	0	A	3
1st Qu	0	1st Qu	0	1st Qu	272900	A-	10
Median	0	Median	0	Median	319035	B	120
Mean	222901	Mean	94762	Mean	3521113	B-	173
3rd Qu	52750	3rd Qu	212190	3rd Qu	393500	B+	30
Max	237370	Max	1118030	Max	1355400	C	26
						C+	499

Even with the formatting issues, this display allows the analyst to see some helpful patterns in the data. For example the range for the variable SPR (selling price) is from $179,000 to $1,270,000, which is a wide range. However, the inter-quartile range, containing fifty percent of the data, is from $276,900 to $404,850, which is significantly tighter. Likewise, the YRBLT (year built) range is 30 years, but the inter-quartile range is from 1998 to 2002.

The string variables of CDU and GRADE also show sufficient dispersion among several levels. This may allow the analyst to develop dummy variables based on these different levels.





This display also reveals some characteristics that are either missing or do not vary significantly over the range of sales. Keeping in mind that the purpose of this exercise is to explain the change in selling prices over a range of properties, any characteristic that does not change will not be helpful. The FUEL characteristic for example shows the same value for the minimum and the maximum, while the BIGAR (built in garage) characteristic displays zero entries for everything. These variables and others like them should be eliminated from further consideration.

The next step in the analysis is to consider the linear correlation of these variables with the selling price variable. The stronger the correlation between any variable and selling price, the greater the likelihood that variable will be helpful in a regression analysis. This is accomplished in R using the following syntax:

```
contin<-data.frame(SPR, EXTWL, LIVACCTR, PLMBFB,PLMBTFX, GFLA, TLIVAR,
                   GRDFCTVL,TOTLSF, SFLA, FIRST, TOTAL, ATTGAR)
cor(contin)
```

“contin” is a data.frame containing a select set of property characteristics to be shown in the correlation matrix. The correlation function is shown as “cor” and the output appears below.

```
                SPR      EXTWL   LIVACCTR     PLMBFB    PLMBTFX       GFLA     TLIVAR
SPR       1.0000000  0.6307238  0.5796292  0.5054668  0.5333368  0.7567582  0.7596153
EXTWL     0.6307238  1.0000000  0.4000229  0.2881604  0.2992583  0.5133889  0.4794060
LIVACCTR  0.5796292  0.4000229  1.0000000  0.5494698  0.5833797  0.4201108  0.6695279
PLMBFB    0.5054668  0.2881604  0.5494698  1.0000000  0.8693008  0.4034416  0.6642485
PLMBTFX   0.5333368  0.2992583  0.5833797  0.8693008  1.0000000  0.3866760  0.7390629
GFLA      0.7567582  0.5133889  0.4201108  0.4034416  0.3866760  1.0000000  0.6141520
TLIVAR    0.7596153  0.4794060  0.6695279  0.6642485  0.7390629  0.6141520  1.0000000
GRDFCTVL  0.8264674  0.6590116  0.4006511  0.2850769  0.2539892  0.6697988  0.4848763
TOTLSF    0.5284610  0.3901423  0.3970316  0.2908710  0.3188506  0.4645543  0.4552921
SFLA      0.7191408  0.4733677  0.5706359  0.5253764  0.6222539  0.5284230  0.7868652
FIRST     0.7857382  0.5368462  0.4768721  0.4328006  0.4206439  0.9705107  0.6602747
TOTAL     0.9663973  0.6102591  0.5515440  0.4733107  0.5009995  0.7250984  0.7251394
ATTGAR    0.4741917  0.2605217  0.3137654  0.3909664  0.4556365  0.2781125  0.4290059
            GRDFCTVL     TOTLSF       SFLA      FIRST      TOTAL     ATTGAR
SPR       0.8264674  0.5284610  0.7191408  0.7857382  0.9663973  0.4741917
EXTWL     0.6590116  0.3901423  0.4733677  0.5368462  0.6102591  0.2605217
LIVACCTR  0.4006511  0.3970316  0.5706359  0.4768721  0.5515440  0.3137654
PLMBFB    0.2850769  0.2908710  0.5253764  0.4328006  0.4733107  0.3909664
PLMBTFX   0.2539892  0.3188506  0.6222539  0.4206439  0.5009995  0.4556365
GFLA      0.6697988  0.4645543  0.5284230  0.9705107  0.7250984  0.2781125
TLIVAR    0.4848763  0.4552921  0.7868652  0.6602747  0.7251394  0.4290059
GRDFCTVL  1.0000000  0.4171422  0.4151086  0.6903963  0.8184978  0.2800198
TOTLSF    0.4171422  1.0000000  0.4895089  0.4915379  0.4912847  0.3358576
SFLA      0.4151086  0.4895089  1.0000000  0.5763400  0.6881975  0.5328562
FIRST     0.6903963  0.4915379  0.5763400  1.0000000  0.7545715  0.2883823
TOTAL     0.8184978  0.4912847  0.6881975  0.7545715  1.0000000  0.4562718
ATTGAR    0.2800198  0.3358576  0.5328562  0.2883823  0.4562718  1.0000000
```

The table on the next page shows the same output reformatted.

The raw output is somewhat difficult to read, which is why it has been placed into the following table.

	SPR	EXTWL	LIVACCTR	PLMBFB	PLMBTFX	GFLA	TLIVAR	GRDFCTVL	TOTLSF	SFLA	FIRST	TOTAL	ATTGAR
SPR	1.0000	0.6307	0.5796	0.5055	0.5333	0.7568	0.7596	0.8265	0.5285	0.7191	0.857	0.9663	0.4742
EXTWL	0.6307	1.0000	0.4000	0.2881	0.2992	0.5134	0.4794	0.6590	0.3901	0.4733	0.5368	0.6103	0.2605
LIVACCTR	0.5796	0.4000	1.0000	0.5494	0.5833	0.4201	0.6695	0.4006	0.3970	0.5706	0.4768	0.5515	0.3138
PLMBFB	0.5055	0.2881	0.5495	1.0000	0.8693	0.4034	0.6642	0.2851	0.2909	0.5254	0.4328	0.4733	0.3909
PLMBTFX	0.5333	0.2992	0.5834	0.8693	1.0000	0.3867	0.7391	0.2539	0.33189	0.6222	0.4206	0.5009	0.4556
GFLA	0.7567	0.5134	0.4201	0.4034	0.3867	1.0000	0.6141	0.6697	0.4645	0.5284	0.9705	0.7250	0.2781
TLIVAR	0.7459	0.4794	0.6695	0.6642	0.7390	0.6141	1.0000	0.4848	0.4553	0.7868	0.6603	0.7251	0.4290
GRDFCTVL	0.8264	0.6590	0.4006	0.2851	0.2539	0.6698	0.4848	1.0000	0.4171	0.4151	0.6904	0.8184	0.2800
TOTLSF	0.5284	0.3901	0.3970	0.2908	0.3188	0.4645	0.4552	0.4171	1.0000	0.4895	0.4915	0.4912	0.3359
SFLA	0.7191	0.4733	0.5706	0.5254	0.6222	0.5284	0.7868	0.4151	0.4895	1.0000	0.5763	0.6882	0.5328
FIRST	0.7858	0.5369	0.4768	0.4328	0.4206	0.9705	0.6602	0.6904	0.4915	0.5763	1.0000	0.7545	0.2883
TOTAL	0.9664	0.6103	0.5515	0.4733	0.5009	0.7251	0.7251	0.8185	0.4913	0.6882	0.7545	1.0000	0.4562
ATTGAR	0.4742	0.2605	0.3137	0.3906	0.4556	0.2781	0.4290	0.2800	0.3359	0.5329	0.2884	0.4563	1.0000

The appraiser can now use this data to begin specifying a regression model, or selecting those property characteristics that will be used in the regression model. This will require a careful analysis of the correlation matrix combined with the appraiser's knowledge of the factors that normally influence value.

Beginning at the top of the matrix, the exterior wall variable (EXTWL) reveals a strong correlation with selling price (.6307). The appraiser must consider whether that variable is also strongly correlated with another variable that he or she considers important in the prediction of market value. Looking down the column headed by EXTWL reveals a strong correlation with GRDFCTVL (grade factor value). In fact, that correlation is stronger than the correlation with selling price. Since grade is a very important factor in influencing value, it will be given precedence in the model and EXTWL will be left out.

There are several variables in the matrix relating to the size of the dwelling:

LIVACCTR – Living Accommodations Total Rooms

GFLA – Ground Floor Living Area

TLIVAR – Total Living Area

SFLA – Square Feet of Living Area

FIRST – First Floor Living Area

Appraisers should avoid placing two variables in a model that essentially describe the same characteristic. Evidence of that is found in the correlation matrix. Look at the correlation between LIVACCTR and TLIVAR, GFLA and FIRST or FIRST and TOTAL. A strong correlation between two variables may introduce instability into the model.

In selecting variables to include in the model, the appraiser should consider variables in the broad categories of size, age, location, quality and condition. The best variables to use in each of these categories will vary with the particular market and the data being collected. For example, location may be contained within the value of the underlying land when land models are built neighborhood by neighborhood. Condition may be captured through a physical condition rating or through an effective age variable. It is the responsibility of the appraiser to learn what influences value in the local market and what data is being collected to capture that information.

The correlation matrix will help identify the category in which variables are located. Total plumbing fixtures (PLMBTFX) accounts for the number of bathrooms in a dwelling, which also relates to size. That becomes evident when you look down the PLMBTFX column to total living area (TLIVAR) with a correlation of 0.7390 and square feet of living area (SFLA) with a correlation of 0.6222. If the appraiser wants to account for the presence of extra plumbing fixtures in a dwelling the difference in correlations would tend to indicate the desirability of SFLA over TLIVAR, since the correlation is smaller.

At this point, it appears the model will include PLMBTFX and SFLA to indicate size. GRDFCTVL (grade factor value) is the only quality variable in this list and TOTLSF (total land square footage) is the only land variable. That leaves age, location and condition unidentified in our variable set. The size of the sales data base may indicate the sales originated from a relatively small area and that location will not be an issue. Physical Condition fell out of consideration as not showing variation. There is a variable called CDU which attempts to identify condition, desirability and utility in a single rating. If there is sufficient variation in that rating, a transformation may be used to capture a type of condition rating. Age can also be captured through a transformation involving the appraisal year and the year built variable.

A total of seven transformations will be used in the regression model. An AGE variable will be created using the formula 2013 – Year Built (*Note: The latest sale year in the sample is 2012, so it is logical to assume a 1/1/2013 valuation date.*). A variable will be created that combines the size and quality variables called SFGF (SFLA * GRDFCTVL).

Five dummy variables will be created to reflect the CDU ratings of EX (Excellent), VG (Very Good), GD (Good), FR (Fair) and PR (Poor). The assumption in doing this is that the AV or average CDU rating will be reflected in the average selling price and the coefficients for the dummy variables will add to or subtract from that amount based on the relative CDU rating. In other words, the dummy variable CDUEX should have a positive coefficient, which, in an additive model will add to the constant term, while the variable CDUFR should have a negative coefficient that will subtract an amount from the constant term. The syntax to create these variables is R follows:

```
SFGF<-SFLA*GRDFCTVL
CDUEX<-ifelse(CDU=="EX",1,0)
CDUVG<-ifelse(CDU=="VG",1,0)
CDUGD<-ifelse(CDU=="GD",1,0)
CDUFR<-ifelse(CDU=="FR",1,0)
CDUPR<-ifelse(CDU=="PR",1,0)
```

The syntax to run a linear regression model in R is:

```
fit<- lm(SPR ~ SFGF + TOTLSF + PLMBTFX + AGE + CDUEX + CDUVG + CDUGD +
         CDUFR + CDUPR + Months, data=sample)
```

The function lm calculates a linear model using the contents of the following parentheses. In the instance shown above, the variable SPR (Selling Price) is the dependent variable and the independent variables follow the tilde.

There are several ways of showing the results of running that syntax. One approach is to look at the coefficients alone

fit$coefficients

(Intercept)	25288.921383
SFGF	80.605125
TOTLSF	1.209095
PLMBTFX	3941.121045
AGE	-4638.494049
CDUEX	142318.255083
CDUVG	78177.425272
CDUGD	39196.733312
CDUFR	-32703.562461
CDUPR	-72628.791005
MONTHS	60.323221

A second approach uses the “summary” command:

summary(fit)

This provides output similar to that provided in other regression packages.

Call:

lm(formula = SPR ~ SFGF + TOTLSF + PLMBTFX + AGE + CDUEX + CDUVG +

CDUGD + CDUFR + CDUPR + Months, data = sample)

Residuals:

Min	1Q	Median	3Q	Max
-122294	-21647	-2421	18327	156394

Coefficients

	Estimate	Std.Error	t-value	Pr(>|t|)	
(Intercept)	25288.921383	10220.00	2.475	0.01352	*
SFGF	80.605125	1.292	62.379	<2e-16	***
TOTLSF	1.209095	0.3354	3.605	0.00033	***
PLMBTFX	3941.121045	564.7	6.980	5.962-12	***
AGE	-4638.494049	280.9	-16.516	<2e-16	***
CDUEX	142318.255083	6822.0	20.8620	<2e-16	***
CDUVG	78177.425272	4202.0	18.606	<2e-16	***
CDUGD	39196.733312	2968.0	13.207	<2e-16	***
CDUFR	-32703.562461	3392.0	-9.642	<2e-16	***
CDUPR	-72628.791005	19150.0	-3.792	0.00016	***
MONTHS	60.323221	304.6	0.198	0.84308	

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error:	33030 on 850 degrees of freedom
Multiple R-squared:	0.9349
Adjusted R-squared:	0.9341
F-statistic:	1220 on 10 and 850 DF
p-value:	< 2.2e-16

This approach uses all variables in the regression equation. The t-value for the MONTHS variable points to the need to exclude it. When that is done, the coefficients change to the following:

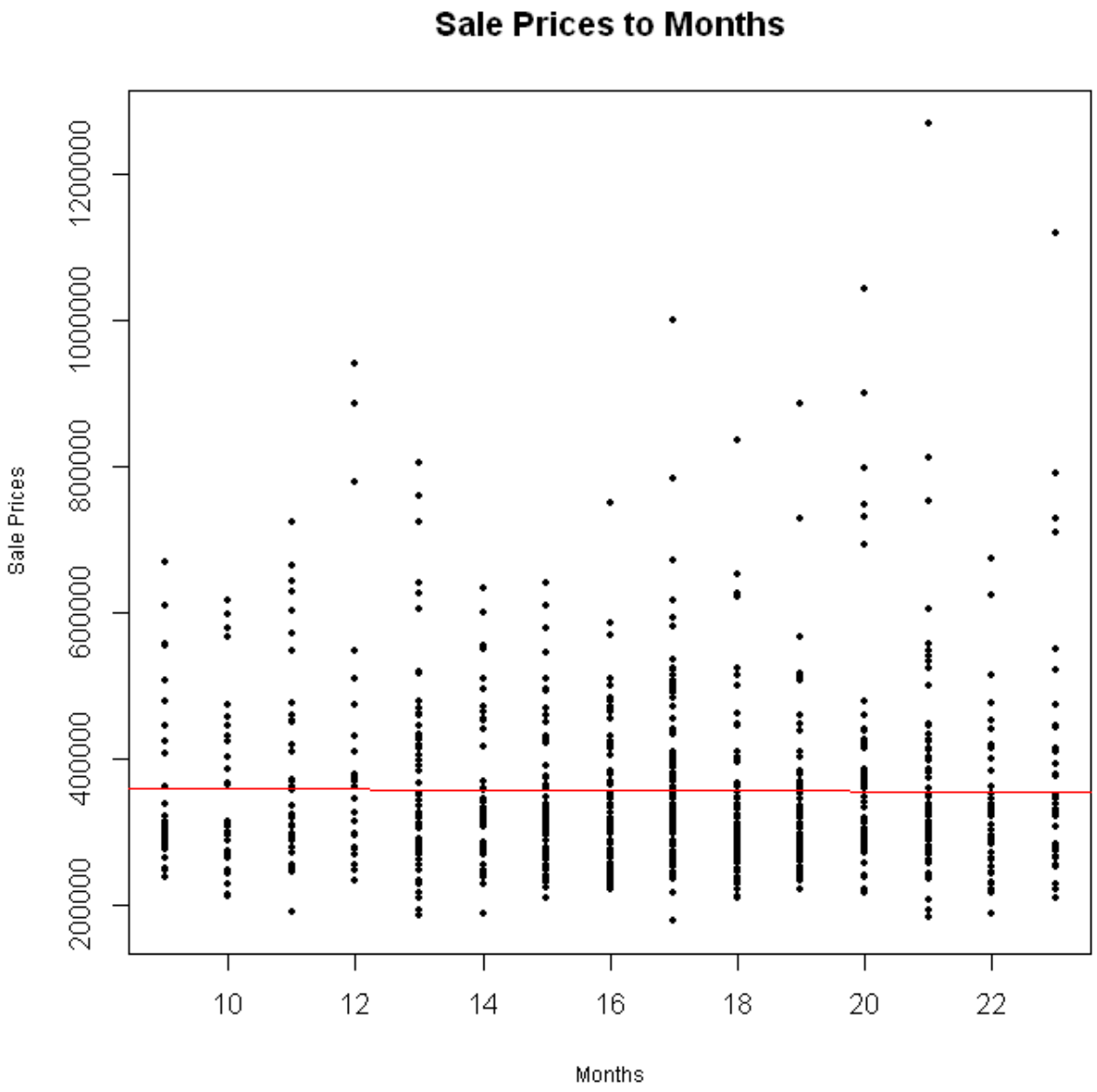
(Intercept)	26290.986609
SFGF	80.609489
TOTLSF	1.207407
PLMBTFX	3940.321511
AGE	-4638.864799
CDUEX	142320.749805
CDUVG	78192.828518
CDUGD	39229.616363
CDUFR	-32608.912056
CDUPR	-72664.515830

One of the methods of accounting for changes in market conditions over time is through the use of a variable that measures the months between the appraisal date and each sale in the data set. That variable did not work well in this model. The t-statistic is well below the significance level of 2.00, meaning this variable should not be introduced into the model.

If we run a correlation using the variables included in the model, we will quickly see that the MONTHS variable has virtually no correlation to selling price.

	SPR	*SFGF*	*TOTLSF*	*PLMBTFX*	*AGE*	*CDUEX*	*CDUVG*	*CDUGD*	*CDUFR*	*CDUPR*	*Months*
SPR	1.0000	0.9170	0.5284	0.5333	-0.2535	0.3054	0.2658	0.07484	-0.1691	-0.0121	-0.0098
SFGF	0.9170	1.0000	0.5288	0.5351	-0.1928	0.1627	0.1424	-0.0023	-0.0275	0.0314	0.0029
TOTLSF	0.5285	0.5288	1.0000	0.3188	0.0869	0.1489	0.1852	0.0301	-0.0988	0.0121	-0.0304
PLMBTFX	0.5333	0.5352	0.3188	1.0000	0.0176	0.0692	0.1327	-0.0297	-0.0591	-0.0040	-0.0134
AGE	-0.2535	-0.1928	0.0869	0.0176	1.0000	0.0411	0.1044	0.1089	-0657	-0.0224	-0.0154
CDUEX	0.3054	0.1627	0.1489	0.0692	0.0411	1.0000	-0.0553	-0.0897	-0.0723	-0.104	-0.0166
CDUVG	0.2658	0.1424	0.1852	0.1327	0.1044	-0.0553	1.0000	-0.1594	-0.1285	-0.0185	-0.0129
CDUGD	0.0784	-0.0024	0.0301	-0.0297	0.1089	-0.0897	-0.1594	1.0000	-0.2086	-0.0300	0.0236
CDUFR	-0.1691	-0.0275	-0.0988	-0.0591	-0.0656	-0.0723	-0.1285	-0.2086	1.0000	-0.0242	0.1360
CDUPR	-0.0121	0.0314	0.0121	-0.0040	-0.0224	-0.0104	0.0185	-0.0300	-0.0242	1.0000	-0.0145
MONTHS	-0.0098	0.0029	-0.0304	-0.0134	-0.0154	-0.0166	-0.0129	0.0237	0.1361	-0.0145	1.0000

Keep in mind that the value of the MONTHS variable will increase as the sale date moves farther back from the appraisal date. A significant change in selling prices over time would be reflected in a strong positive or negative correlation. A negative correlation would indicate that prices were generally increasing. A positive correlation would indicate prices were decreasing. The current lack of correlation indicates a relatively flat market. That can be viewed in a scatter diagram like the following that compares selling prices (SPR) to the MONTHS variable.



R allows the user to overlay a regression line, such as the red line above, over a scatter diagram of the variables MONTHS and SPR. The appraiser can readily see that sales prices reflect little change in the market over this period of time, which renders the MONTHS variable useless in predicting market value.

Ratio Study Analysis

The ratio study is the primary tool of the assessor for determining the accuracy of assessments. This tool uses various measures of the relationship between the assessor's estimated value of parcels and their actual selling prices. This portion of the booklet will explore the use of R in developing a jurisdiction wide ratio study report and, in the course of doing that, give the reader sufficient information to develop other reports on subsets of that data.

Most of the statistics used in a ratio study report have been previously covered and will be reviewed here in the context of a ratio study. The reader should keep in mind this is only one approach.

```
library(psych)
library(base)
library(rpivotTable)

ratdata <- read.table("G:/R Project/Rdata/CRatios.csv", header=TRUE,
sep=",")
attach(ratdata)

ratdata$Ratio<-ratdata$Appraised/ratdata$Sale.Amt

#Mean Ratio
Meanrat<-Mean(Ratio)
Meanrat

#Median Ratio
Medrat<-median(Ratio)
Medrat

#Coefficient of Dispersion
COD<-(MeanAD(Ratio, FUN=median)/Medrat)
COD

#Weighted Mean Ratio
WgtdMn<-sum(Appraised)/sum(Sale.Amt)
WgtdMn

#Price Related Differential
PRD<-Meanrat/WgtdMn
PRD
```

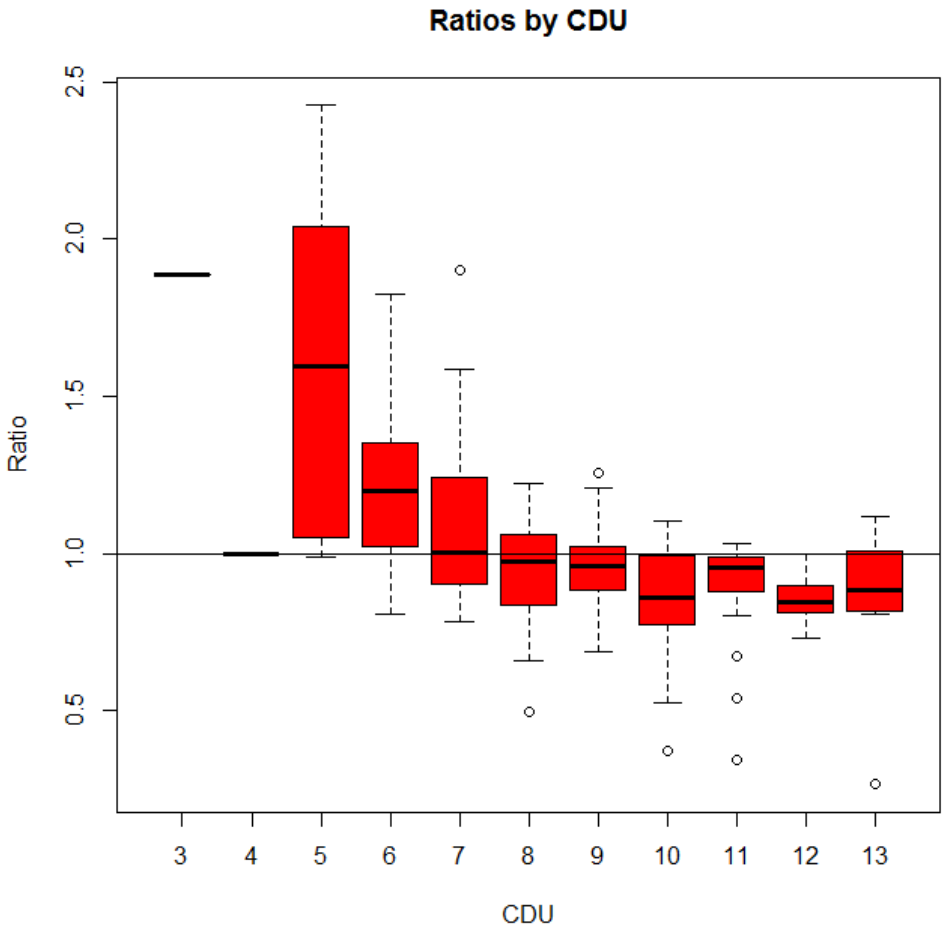
The results of these calculations appear below.

```
> Medrat<-median(Ratio)
> Medrat
[1] 0.97

> COD<-(MeanAD(Ratio, FUN=median)/Medrat)
> COD
[1] 0.1684589

> Meanrat<-Mean(Ratio)
> Meanrat
[1] 0.9965086

> WgtdMn<-sum(Appraised)/sum(Sale.Amt)
> WgtdMn
[1] 0.9195921

> PRD<-Meanrat/WgtdMn
> PRD
[1] 1.083642
```

A more complete report that groups parcels by year built, size, quality and other measures is shown in another document.

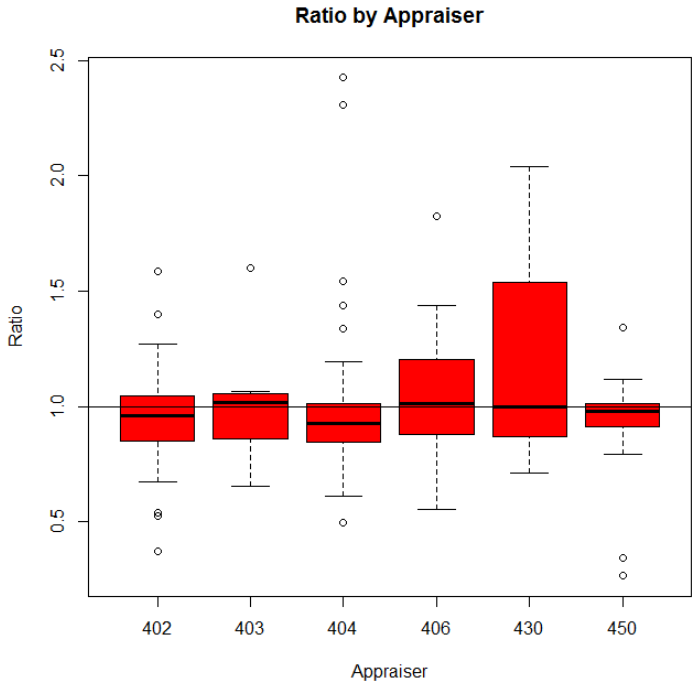
Charts and graphs present data in a way some people find easier to comprehend. Boxplots of ratio ranges by some categorical variable help the user make comparisons that can lead to adjustments. The following syntax yields a boxplot of ratios by Condition, Desirability and Utility (CDU) ratings. Included in the syntax is the ”abline” function, which draws a horizontal line from the vertical axis at the 100% ratio level. This provides the user with a benchmark to help with the comparisons.

```
# Boxplot of CDU by Ratio
boxplot(Ratio~CDU,data=ratdata, col="red", main="Ratios by CDU",
xlab="CDU", ylab="Ratio")
abline(h="1.00", col="black")
```



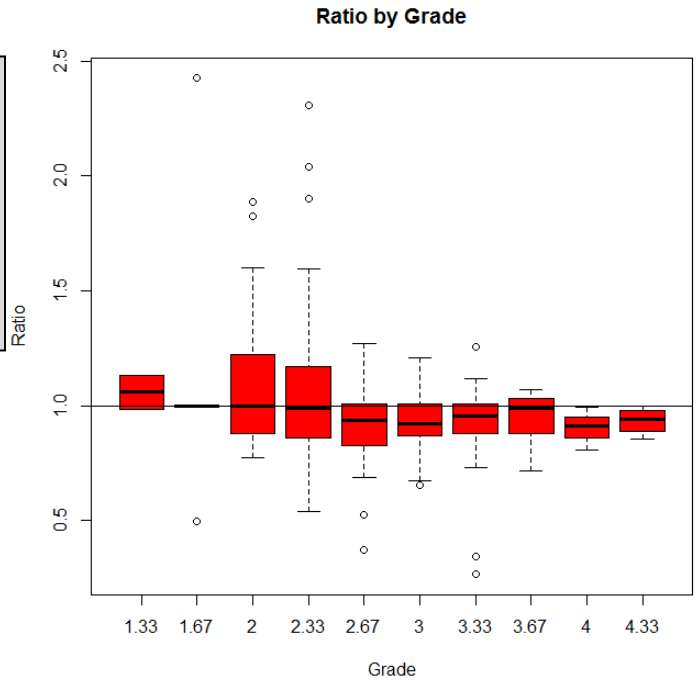
This plot appears to indicate properties receiving a lower CDU rating have relatively higher ratios than those with a higher CDU rating. How significant that is will be determined by a closer examination of the range of ratios within each group.

The following syntax produces boxplots of the relationship between ratios and specific appraisers and Grade or quality ratings.



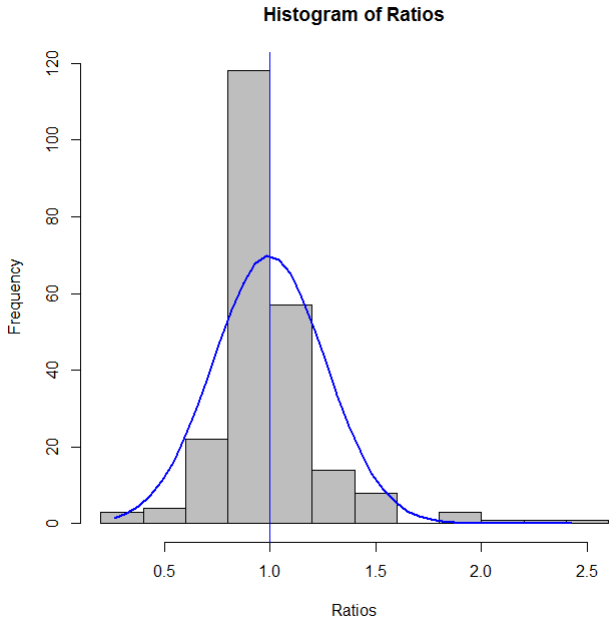

```
#Boxplot of Appraiser by
Ratio
boxplot(Ratio~Appraiser,dat
   a=ratdata, col="red",
   main="Ratio by
   Appraiser",
   xlab="Appraiser",
   ylab="Ratio")
abline(h="1.00",
   col="black")
```

```
#Boxplot of Grade by Ratio
boxplot(Ratio~Grade,data=ra
   tdata, col="red",
   main="Ratio by Grade",
   xlab="Grade",
   ylab="Ratio")
abline(h="1.00",
   col="black")
```

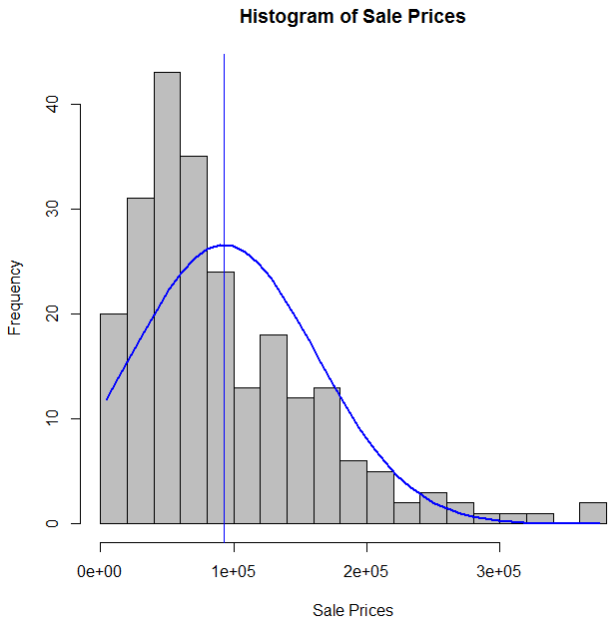


R for Assessors

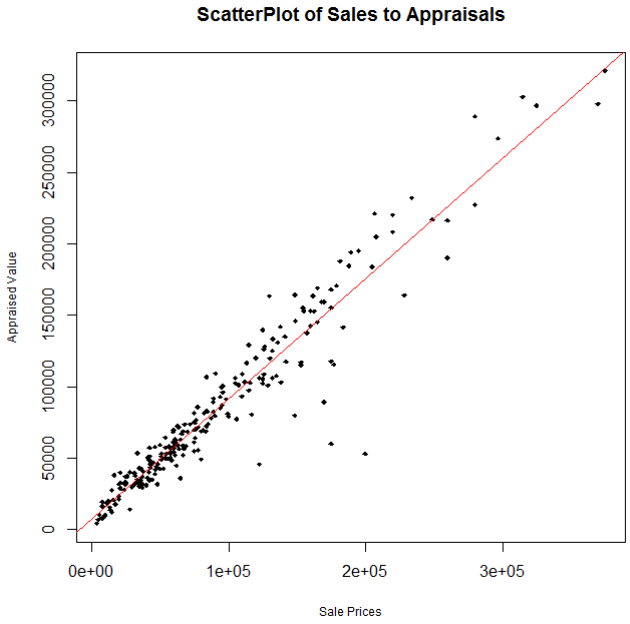
The syntax shown below provides various plots the user can modify to suit their individual needs and desires. The point of this description is not to prescribe a single approach but provide the tools for a multifaceted approach to ratio studies.




```
#Histogram of Ratios
h<-hist(ratdata$Ratio, col="Gray",
   xlab="Ratios", main="Histogram of
   Ratios")
x<-ratdata$Ratio
xfit<-seq(min(x),max(x),length=40)
yfit<-
dnorm(xfit,mean=mean(x),sd=sd(x))
yfit <-
yfit*diff(h$mids[1:2])*length(x)
lines(xfit, yfit, col="blue",
lwd=2)
abline(v=mean(x),col="blue")
```

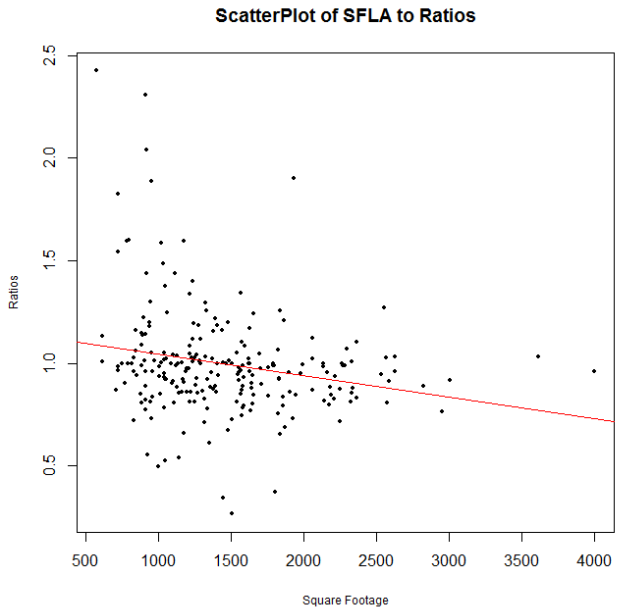



```
#Histogram of Sale Prices
h<-hist(ratdata$Sale.Amt,
   breaks=15,  col="Gray",
   xlab="Sale Prices",
   main="Histogram of Sale Prices")
x<-ratdata$Sale.Amt
xfit<-seq(min(x),max(x),length=40)
yfit<-
dnorm(xfit,mean=mean(x),sd=sd(x))
yfit <-
yfit*diff(h$mids[1:2])*length(x)
lines(xfit, yfit, col="blue",
lwd=2)
abline(v=mean(x),col="blue")
```




```
#Scatterplot of Sale Prices to
Appraised Values
plot(ratdata$Sale.Amt,
     ratdata$Appraised,
     main="ScatterPlot of Sales to
     Appraisals", xlab="Sale
     Prices", ylab="Appraised
     Value", pch=18, cex=.5,
     cex.lab=.75)
abline(lm(ratdata$Appraised~ratdata
$Sale.Amt), col="red")
```

```
#Scatterplot of SFLA to Ratios
plot(ratdata$SFLA, ratdata$Ratio,
     main="ScatterPlot of SFLA to
     Ratios", xlab="Square
     Footage",  ylab="Ratios",
     pch=16, cex=.5, cex.lab=.75)
abline(lm(ratdata$Ratio~ratdata$SFL
A), col="red")
```



Appendix

Glossary

API - An **application programming interface** (**API**) specifies how some software components should interact with each other.

Data Frame is the mechanism within "R" that is used to store data tables. It is a list of vectors of equal length.

Free Software means software that respects users' freedom and community. Roughly, the users have the freedom to run, copy, distribute, study, change and improve the software.

Thus, "free software" is a matter of liberty, not price. To understand the concept, you should think of "free" as in "free speech," not as in "free beer".

With these freedoms, the users (both individually and collectively) control the program and what it does for them. When users don't control the program, the program controls the users. The developer controls the program, and through it exercises power over the users. Therefore, a "nonfree" or "proprietary" program is an instrument of unjust power.

A program is free software if the program's users have the four essential freedoms:

- The freedom to run the program, for any purpose (freedom 0).
- The freedom to study how the program works, and change it so it does your computing as you wish (freedom 1). Access to the source code is a precondition for this.
- The freedom to redistribute copies so you can help your neighbor (freedom 2).
- The freedom to distribute copies of your modified versions to others (freedom 3). By doing this you can give the whole community a chance to benefit from your changes. Access to the source code is a precondition for this.

A program is free software if it gives users adequately all of these freedoms. Otherwise, it is nonfree. While we can distinguish various nonfree distribution schemes in terms of how far they fall short of being free, we consider them all equally unethical.

The rest of this page clarifies certain points about what makes specific freedoms adequate or not.

Freedom to distribute (freedoms 2 and 3) means you are free to redistribute copies, either with or without modifications, either gratis or charging a fee for distribution, to anyone anywhere. Being free to do these things means (among other things) that you do not have to ask or pay for permission to do so.

You should also have the freedom to make modifications and use them privately in your own work or play, without even mentioning that they exist. If you do publish your changes, you should not be required to notify anyone in particular, or in any particular way.

The freedom to run the program means the freedom for any kind of person or organization to use it on any kind of computer system, for any kind of overall job and purpose, without being required to communicate about it with the developer or any other specific entity. In this freedom, it is the *user's* purpose that matters, not the *developer's*purpose; you as a user are free to run the program for your purposes, and if you distribute it to someone else, she is then free to run it for her purposes, but you are not entitled to impose your purposes on her.

The freedom to redistribute copies must include binary or executable forms of the program, as well as source code, for both modified and unmodified versions. (Distributing programs in runnable form is necessary for conveniently installable free operating systems.) It is OK if there is no way to produce a binary or executable form for a certain program (since some languages don't support that feature), but you must have the freedom to redistribute such forms should you find or develop a way to make them.

In order for freedoms 1 and 3 (the freedom to make changes and the freedom to publish the changed versions) to be meaningful, you must have access to the source code of the program. Therefore, accessibility of source code is a necessary condition for free software. Obfuscated“source code” is not real source code and does not count as source code.

Freedom 1 includes the freedom to use your changed version in place of the original. If the program is delivered in a product designed to run someone else's modified versions but refuse to run yours — a practice known as “tivoization” or “lockdown”,or (in its practitioners' perverse terminology) as “secure boot” — freedom 1 becomes a theoretical fiction rather than a practical freedom. This is not sufficient. In other words, these binaries are not free software even if the source code they are compiled from is free.

One important way to modify a program is by merging in available free subroutines and modules. If the program's license says that you cannot merge in a suitably licensed existing module — for instance, if it requires you to be the copyright holder of any code you add — then the license is too restrictive to qualify as free.

Freedom 3 includes the freedom to release your modified versions as free software. A free license may also permit other ways of releasing them; in other words, it does not have to be a copyleft license. However, a license that requires modified versions to be nonfree does not qualify as a free license.

In order for these freedoms to be real, they must be permanent and irrevocable as long as you do nothing wrong; if the developer of the software has the power to revoke the license, or retroactively add restrictions to its terms, without your doing anything wrong to give cause, the software is not free.

However, certain kinds of rules about the manner of distributing free software are acceptable, when they don't conflict with the central freedoms. For example, copyleft (very simply stated) is the rule that when redistributing the program, you cannot add restrictions to deny other people the central freedoms. This rule does not conflict with the central freedoms; rather it protects them.

“Free software” does not mean “noncommercial”. A free program must be available for commercial use, commercial development, and commercial distribution. Commercial development of free software is no longer unusual; such free commercial software is very important. You may have paid money to get copies of free software, or you may have obtained copies at no charge. But regardless of how you got your copies, you always have the freedom to copy and change the software, even to sell copies.

Whether a change constitutes an improvement is a subjective matter. If your right to modify a program is limited, in substance, to changes that someone else considers an improvement, that program is not free.

However, rules about how to package a modified version are acceptable, if they don't substantively limit your freedom to release modified versions, or your freedom to make and use modified versions privately. Thus, it is acceptable for the license to require that you change the name of the modified version, remove a logo, or identify your modifications as yours. As long as these requirements are not so burdensome that they effectively hamper you from releasing your changes, they are acceptable; you're already making other changes to the program, so you won't have trouble making a few more.

Rules that “if you make your version available in this way, you must make it available in that way also” can be acceptable too, on the same condition. An example of such an acceptable rule is one saying that if you have distributed a modified version and a previous developer asks for a copy of it, you must send one. (Note that such a rule still leaves you the choice of whether to distribute your version at all.) Rules that require release of source code to the users for versions that you put into public use are also acceptable.

A special issue arises when a license requires changing the name by which the program will be invoked from other programs. That effectively hampers you from releasing your changed version so that it can replace the original when invoked by those other programs. This sort of requirement is acceptable only if there's a suitable aliasing facility that allows you to specify the original program's name as an alias for the modified version.

In the GNU project, we use copyleftto protect these freedoms legally for everyone. But noncopylefted free software also exists. We believe there are important reasons whyit is better to use copyleft, but if your program is noncopylefted free software, it is still basically ethical. (See Categories of Free Software for a description of how “free software,” “copylefted software” and other categories of software relate to each other.)

Sometimes government export control regulationsand trade sanctions can constrain your freedom to distribute copies of programs internationally. Software developers do not have the power to eliminate or

override these restrictions, but what they can and must do is refuse to impose them as conditions of use of the program. In this way, the restrictions will not affect activities and people outside the jurisdictions of these governments. Thus, free software licenses must not require obedience to any nontrivial export regulations as a condition of exercising any of the essential freedoms.

Merely mentioning the existence of export regulations, without making them a condition of the license itself, is acceptable since it does not restrict users. If an export regulation is actually trivial for free software, then requiring it as a condition is not an actual problem; however, it is a potential problem, since a later change in export law could make the requirement nontrivial and thus render the software nonfree.

Most free software licenses are based on copyright, and there are limits on what kinds of requirements can be imposed through copyright. If a copyright-based license respects freedom in the ways described above, it is unlikely to have some other sort of problem that we never anticipated (though this does happen occasionally). However, some free software licenses are based on contracts, and contracts can impose a much larger range of possible restrictions. That means there are many possible ways such a license could be unacceptably restrictive and nonfree.

We can't possibly list all the ways that might happen. If a contract-based license restricts the user in an unusual way that copyright-based licenses cannot, and which isn't mentioned here as legitimate, we will have to think about it, and we will probably conclude it is nonfree.

When talking about free software, it is best to avoid using terms like “give away” or “for free,” because those terms imply that the issue is about price, not freedom. Some common terms such as “piracy” embody opinions we hope you won't endorse. See Confusing Words and Phrases that are Worth Avoiding for a discussion of these terms. We also have a list of proper translations of“free software” into various languages.

Finally, note that criteria such as those stated in this free software definition require careful thought for their interpretation. To decide whether a specific software license qualifies as a free software license, we judge it based on these criteria to determine whether it fits their spirit as well as the precise words. If a license includes unconscionable restrictions, we reject it, even if we did not anticipate the issue in these criteria. Sometimes a license requirement raises an issue that calls for extensive thought, including discussions with a lawyer, before we can decide if the requirement is acceptable. When we reach a conclusion about a new issue, we often update these criteria to make it easier to see why certain licenses do or don't qualify.

If you are interested in whether a specific license qualifies as a free software license, see our list of licenses. If the license you are concerned with is not listed there, you can ask us about it by sending us email at <licensing@gnu.org>.

If you are contemplating writing a new license, please contact the Free Software Foundation first by writing to that address. The proliferation of different free software licenses means increased work for

users in understanding the licenses; we may be able to help you find an existing free software license that meets your needs.

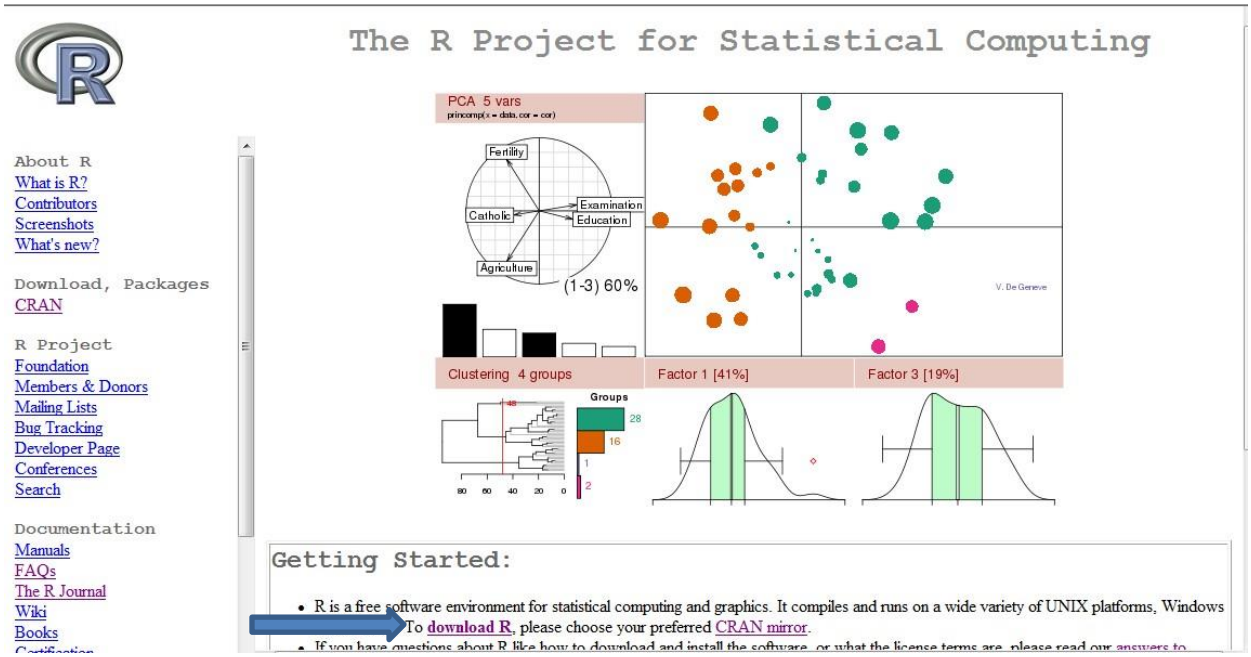
If that isn't possible, if you really need a new license, with our help you can ensure that the license really is a free software license and avoid various practical problems.

GNU is a Unix-like operating system that is free software—it respects your freedom. You can install versions of GNU (more precisely, GNU/Linux systems) which are entirely free software.

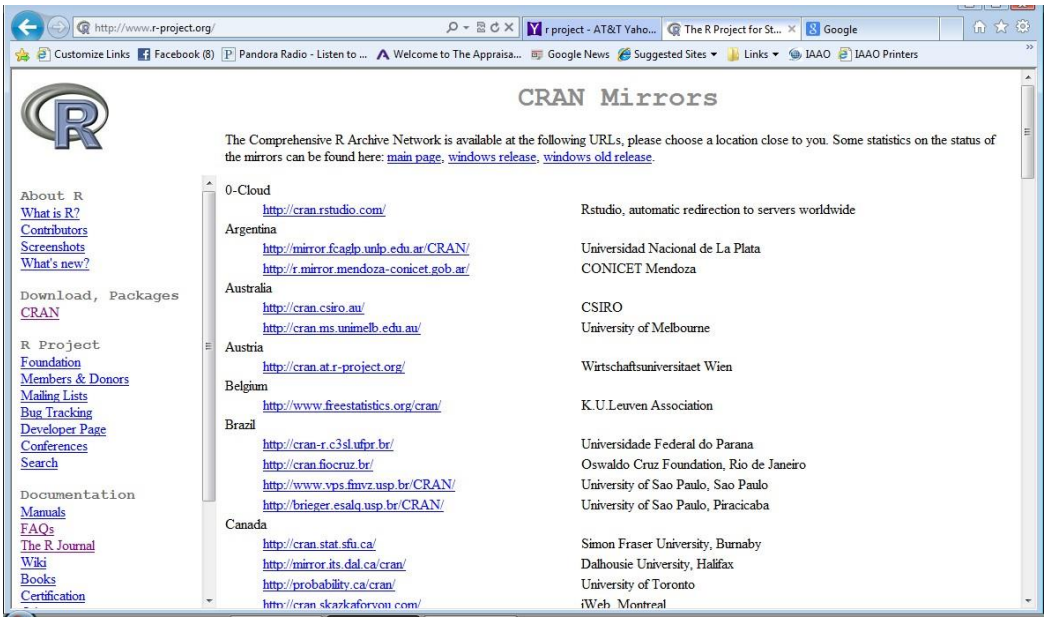
Vectors can be thought of as contiguous cells containing data.

Installation of R

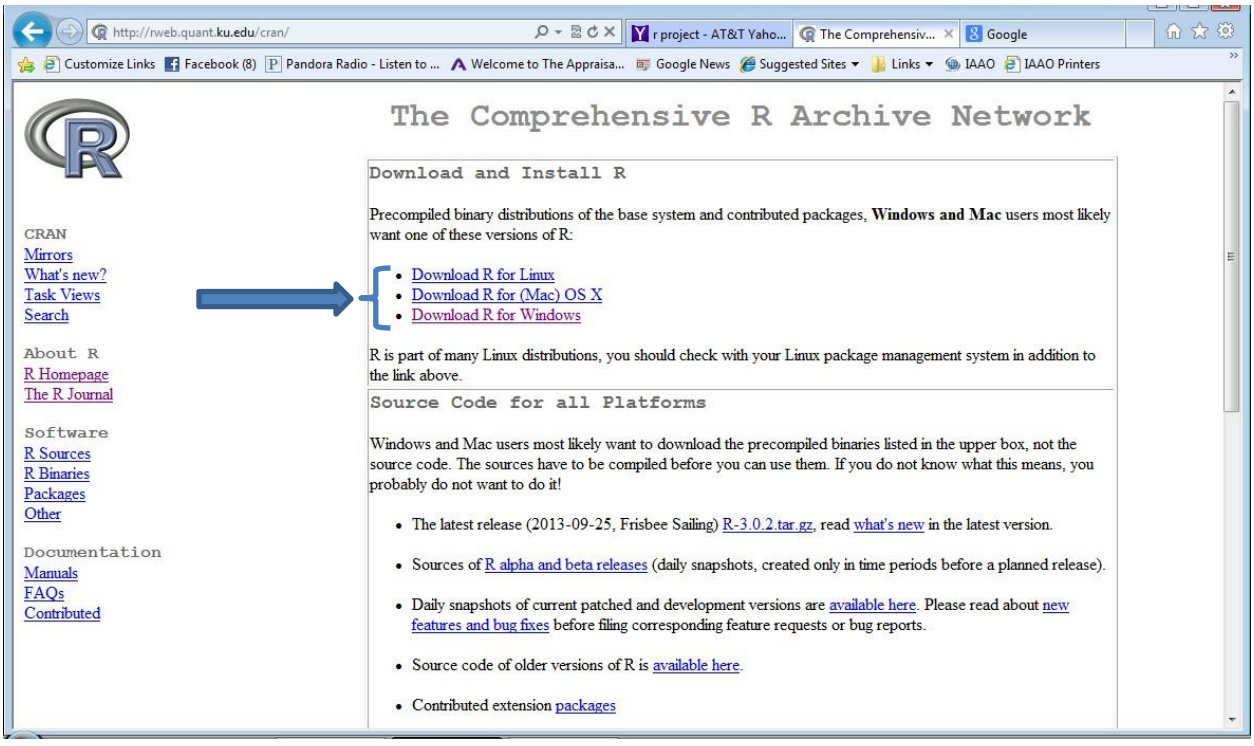
Begin by navigating to the R Project site at http://www.r-project.org/. Select download r on the home page.



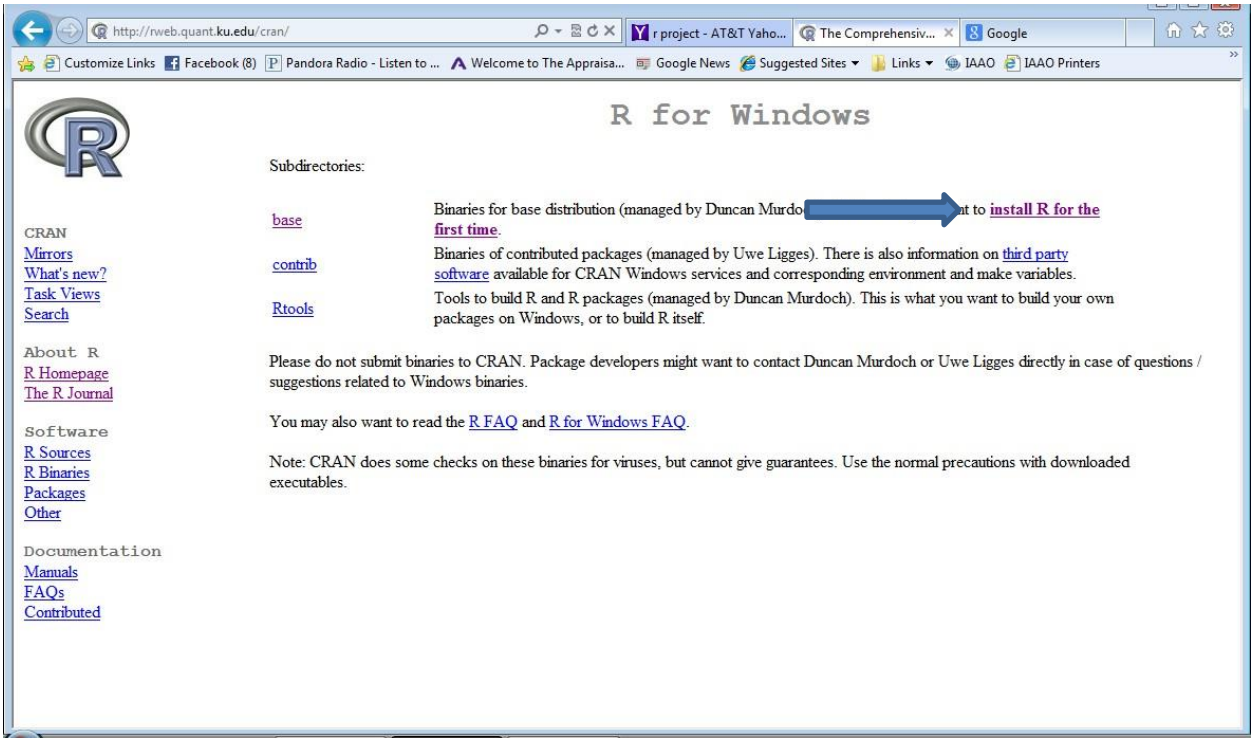
After selecting the download R option, you will be taken to a page containing the location of CRAN mirrors throughout the world. Select the mirror closest to your physical location.



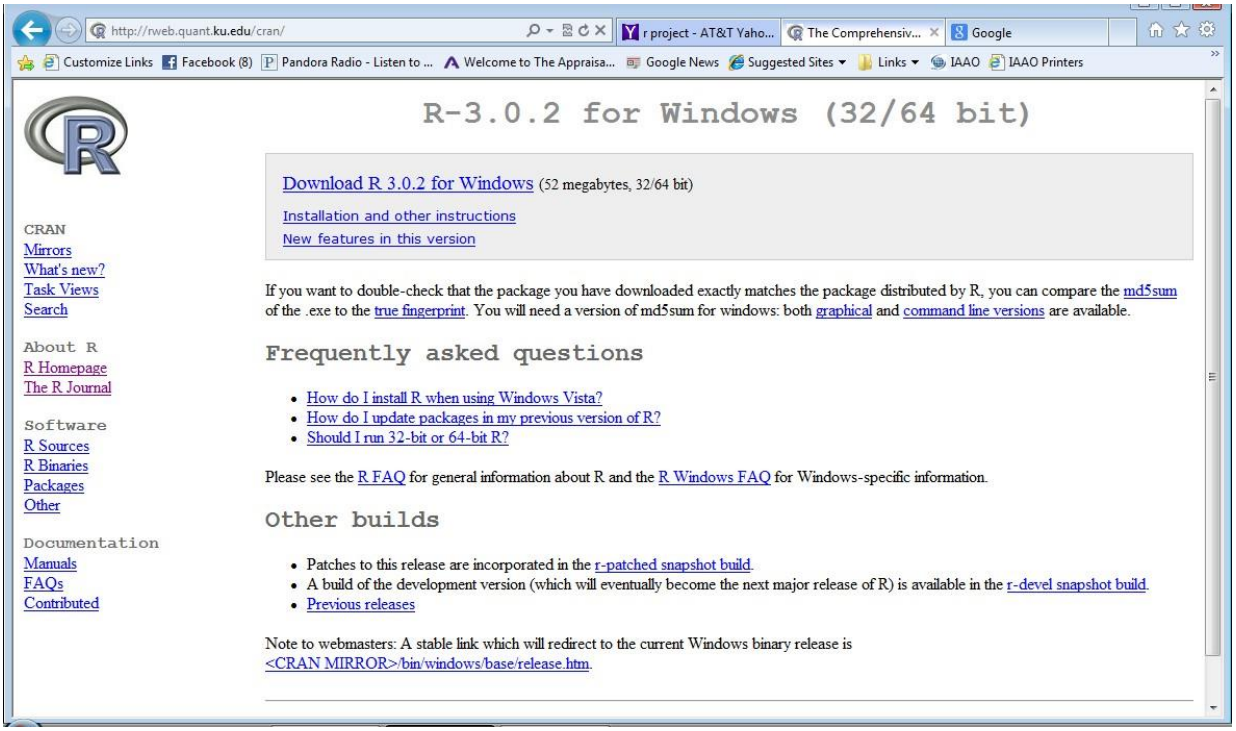
After selecting a location, the user will be taken to a screen containing options for the actual download. Most users will select the option of Download R for Windows.



Selecting Download R for Windows takes the user to the following screen where first-time users will select install R for the first time.



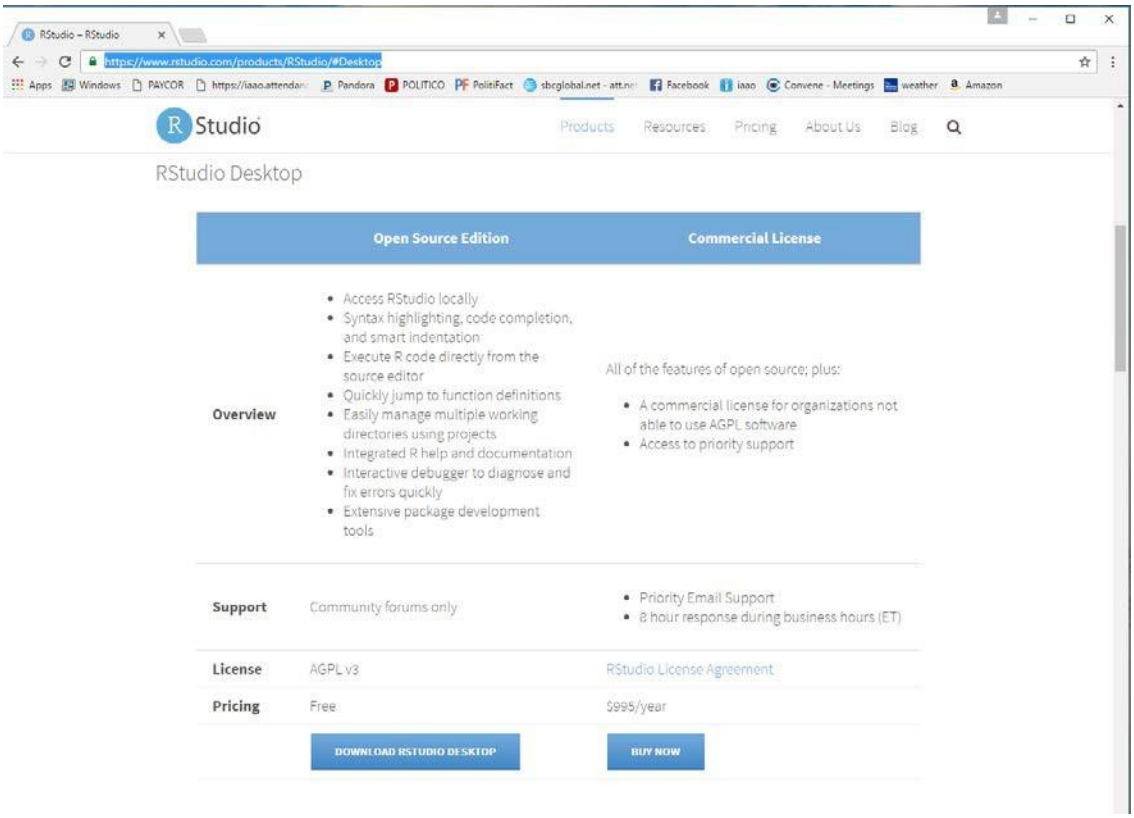
There are helpful instructions on the following screen, but the user who is ready to proceed with the installation can select Download R 3.0.2 for Windows. (Note the version number will change over time)



Once begun, the installation process will proceed much as any other software installation.

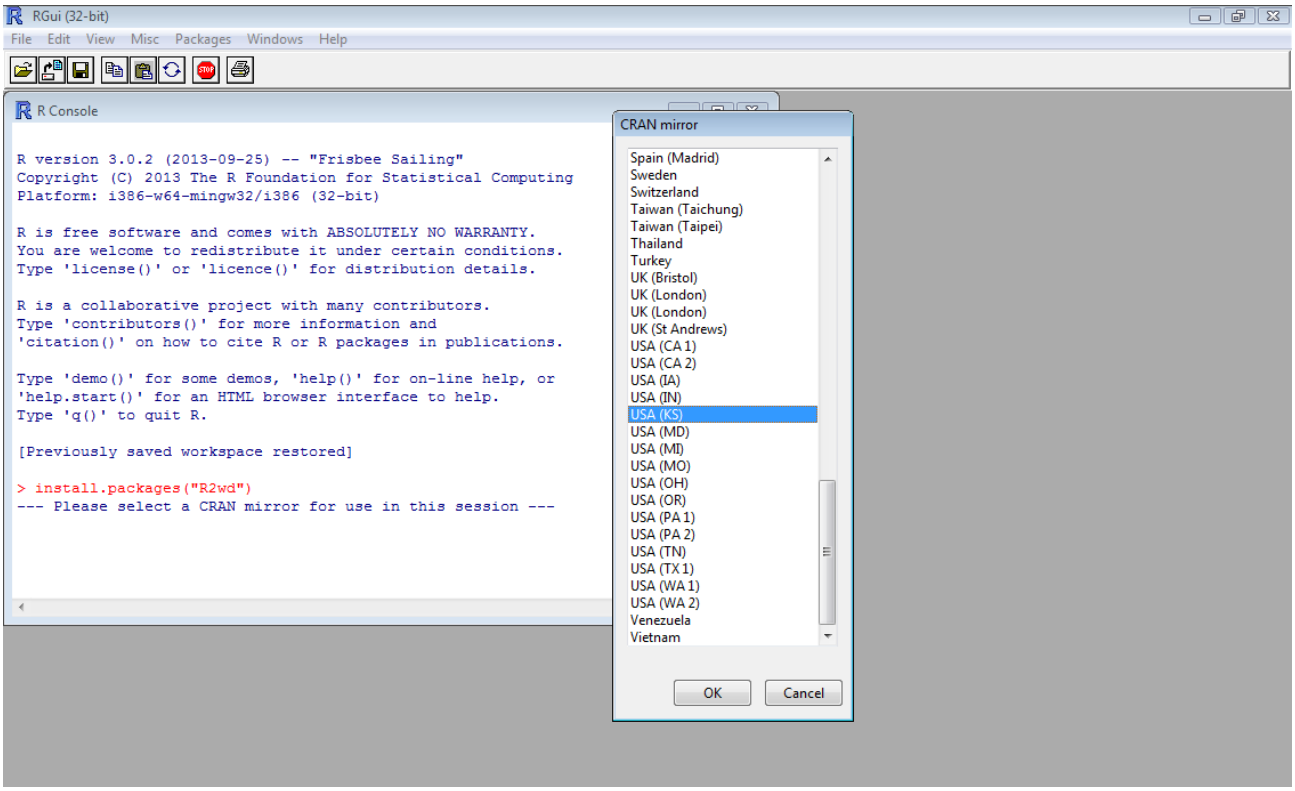
Installation of R Studio

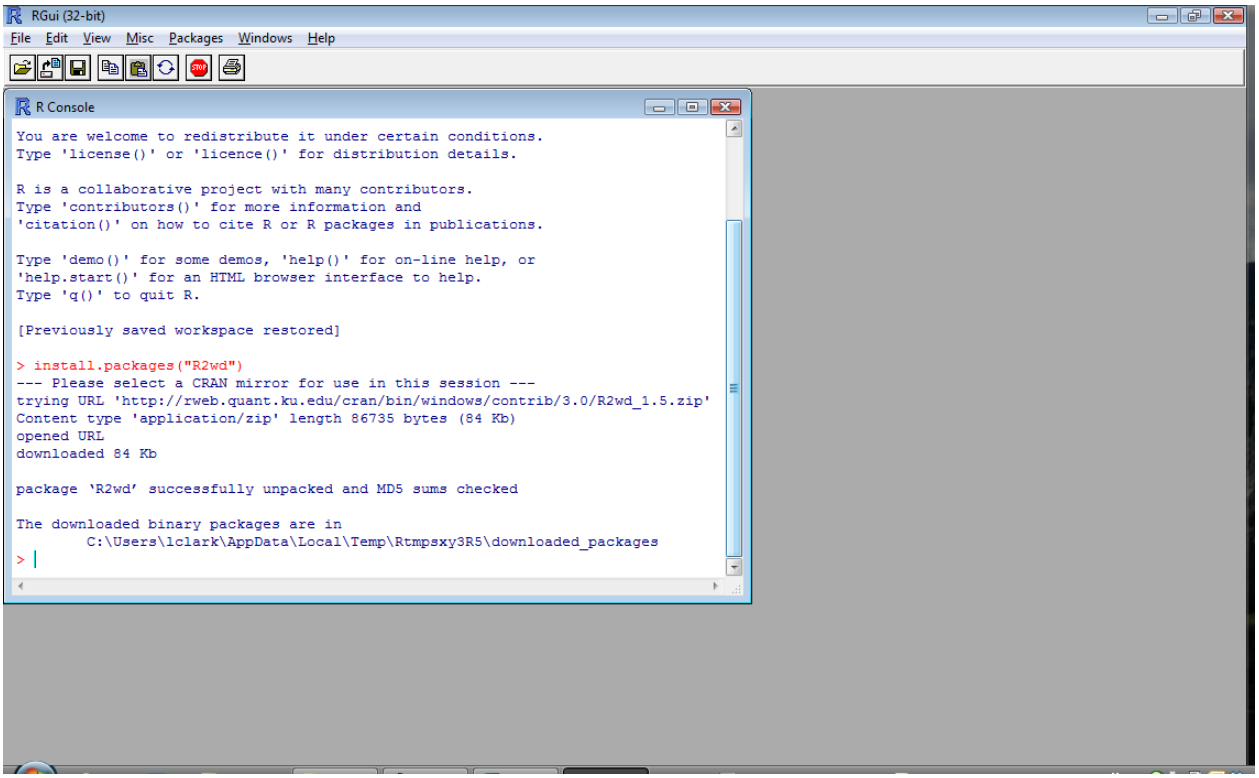
Navigate to the RStudio website at https://www.rstudio.com/products/RStudio/#Desktop. You will be presented with the options shown below.

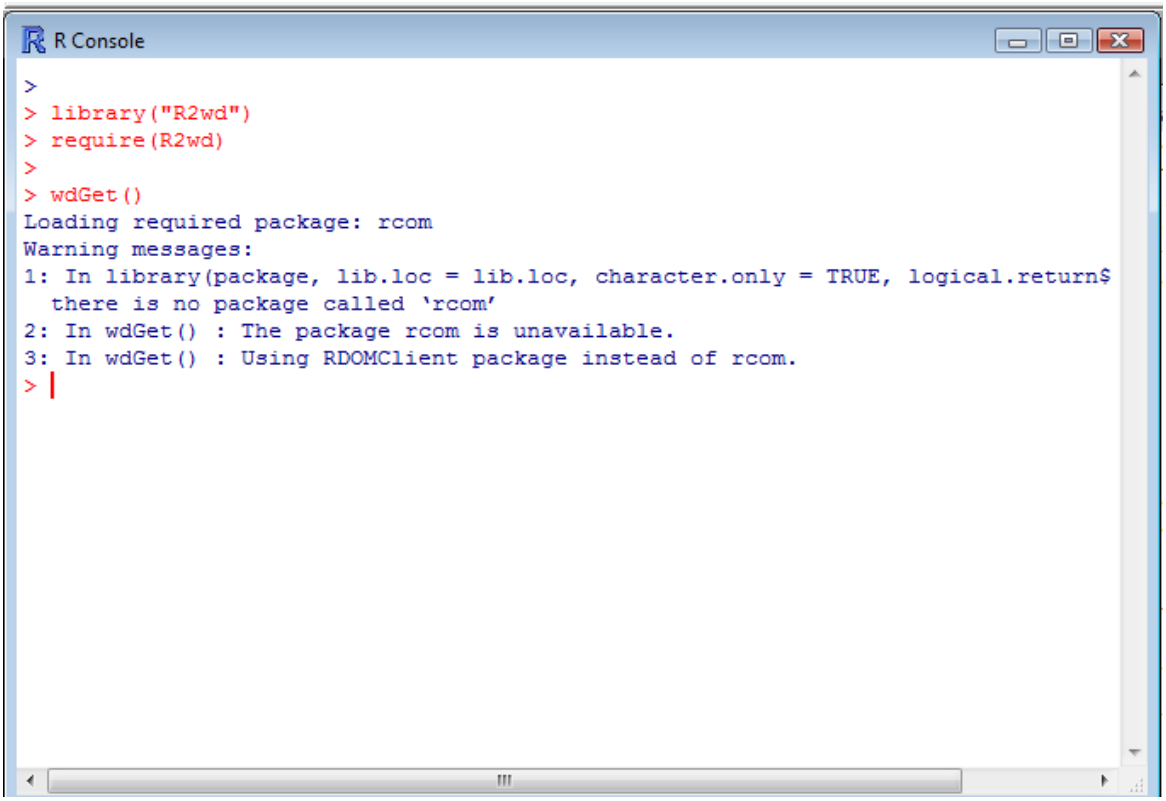


This tutorial is based on the implementation of the open source edition. The user should download and install that edition.

R2wd

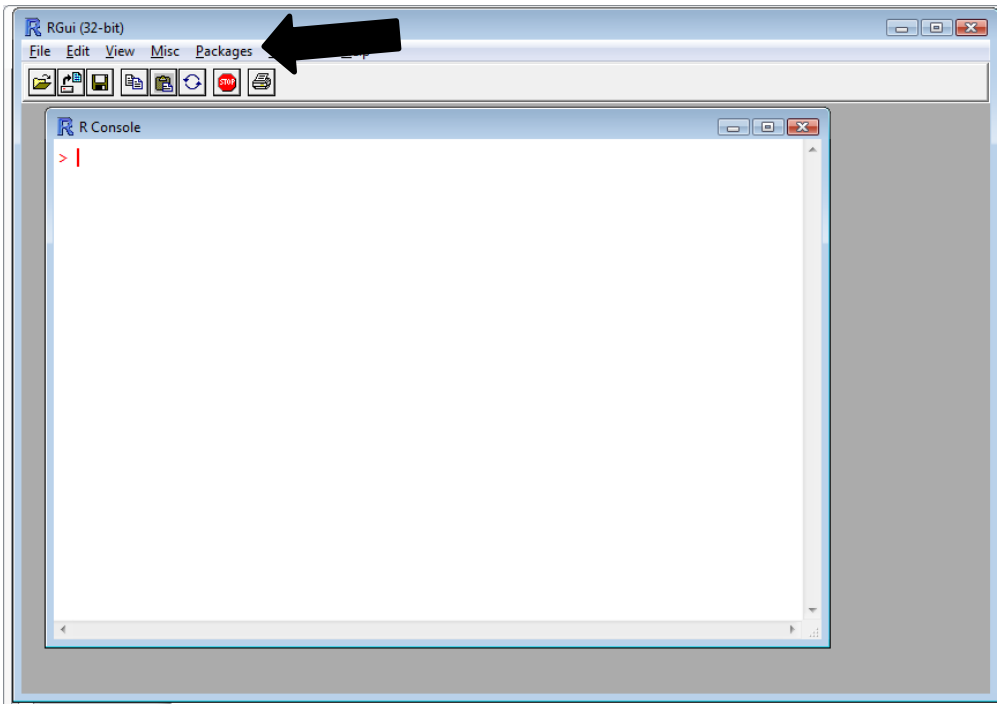




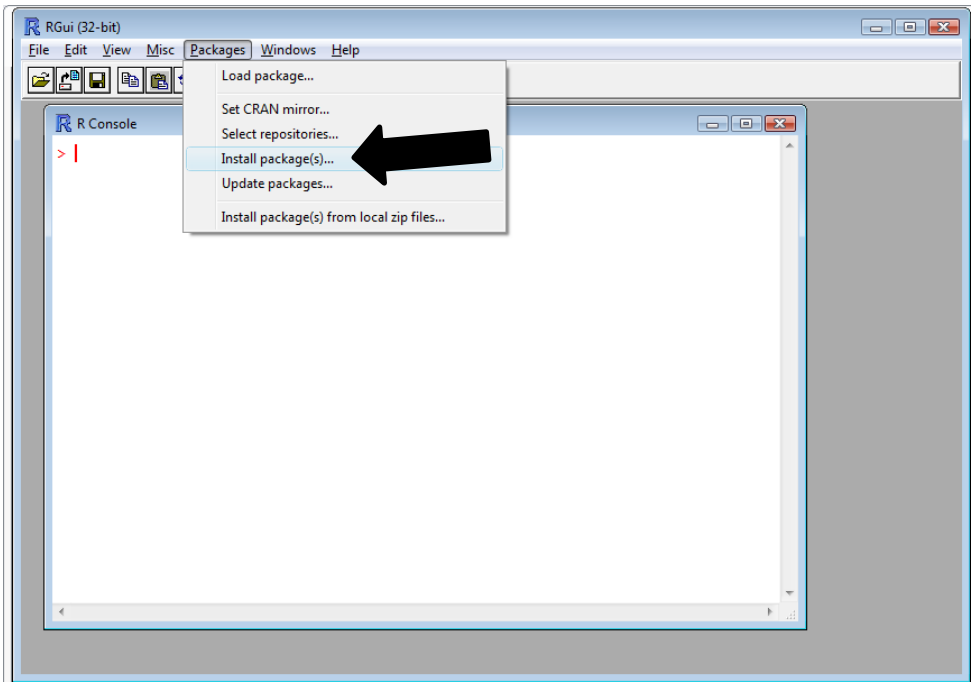


Installation of Packages

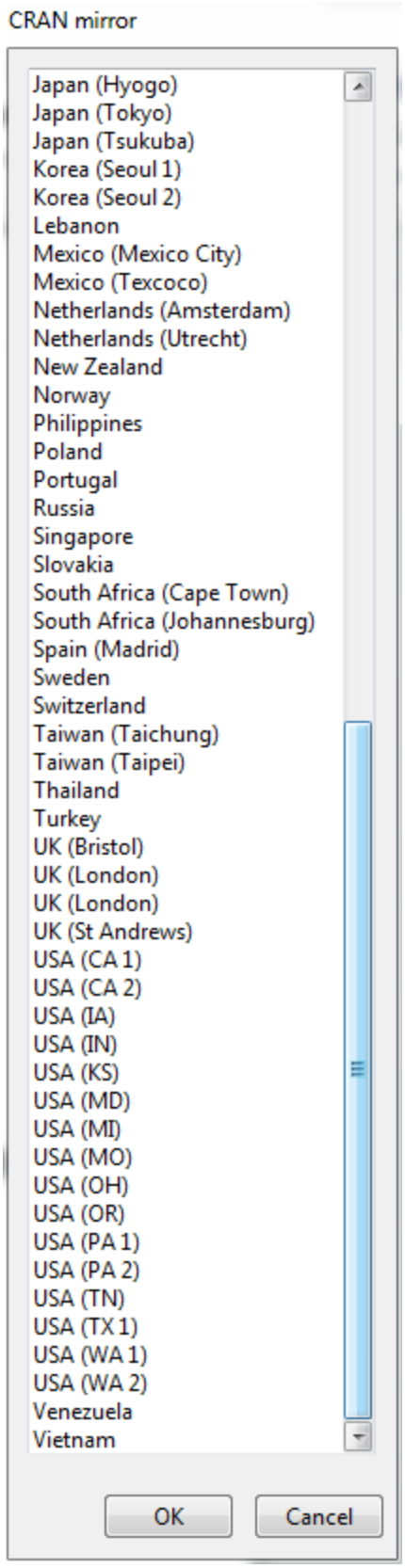
Select Packages from the top menus



From the dropdown, select Install packages

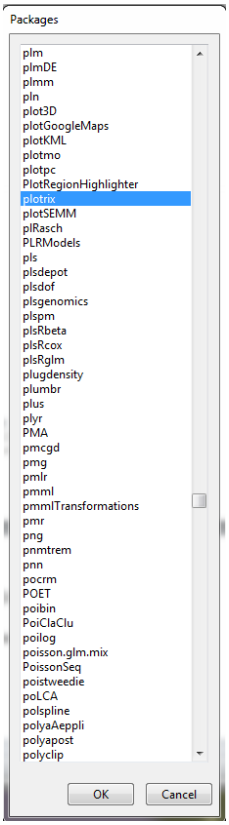


Select a CRAN mirror for use in the session



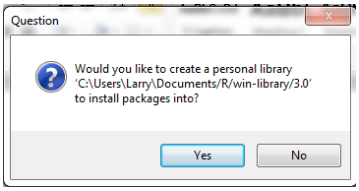
Select a mirror location close to your own and hit the OK button at the bottom of the selection window

This will cause another window to open containing a list of packages to be loaded.

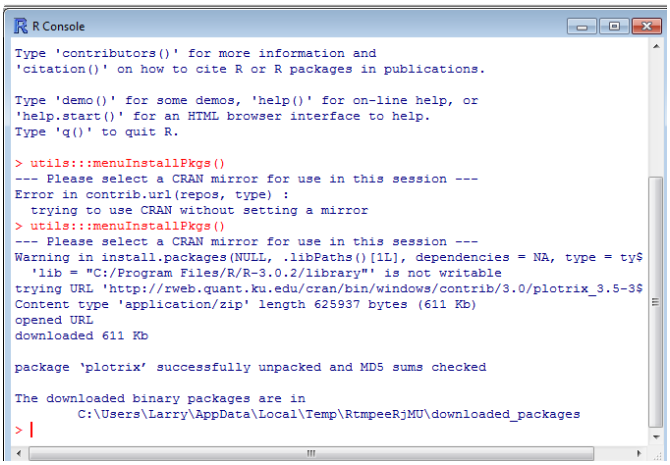


Select the package desired and hit OK.

If this is the first installation of packages on this computer, you may be asked to create a personal library. It is a good idea to do that.



The console will track this activity.



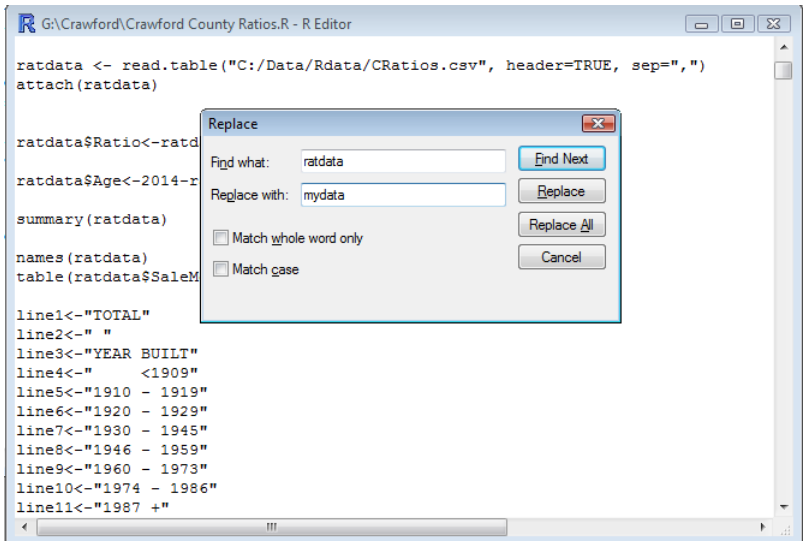
Now when it becomes necessary to use that package, the assessor calls it with the command

library()

The package name will be inserted between the parentheses.

Functions for Assessors

Find and Replace – R uses the Ctrl-H function also found in SPSS that allows the user to find specific words or phrases and replace them with another word or phrase.



In this example, the word to be located is “ratdata” to be replaced with “mydata”. By using the “Find Next” option the user can move from one occurrence of the target word to another and make individual decisions on replacement or can choose the “Replace All” option. The latter should be done carefully and only when the user is certain the target may not be found as part of another word or phrase other than those the user actually wants to change.

Websites of Interest

http://cran.us.r-project.org/ (Google CRAN)

http://cran.r-project.org/doc/manuals/R-intro.html

www.statmethods.net (Google Quick-R)

http://www.r-tutor.com/

http://www.ats.ucla.edu/stat/r/

http://ww2.coastal.edu/kingw/statistics/R-tutorials/

http://math.illinoisstate.edu/dhkim/rstuff/rtutor.html

http://heather.cs.ucdavis.edu/~matloff/r.html

http://math.ucdenver.edu/RTutorial/

http://rtutorialseries.blogspot.com/