Introduction to R

Using a scriptable language, like R, and software that makes life a little easier, like R studio, are great ways to create replicable and shareable sets of analyses.

So let's begin with a few basic ideas about how R works. The fundamental worker is the *function*, which is always followed by the open and close parantheses. Each function executes an operation as designated by the series of *arguments*, which are set by values in the parentheses. Arguments can tell the function which data to use, the methods (one-tail vs. two-tail), and many other options. Some functions are more complicated that others and require many arguments, while other functions are similar and may require only one or two arguments. Arguments can be set by using =, and some funtions will have default values for arguments and others require specific arguments to be referenced.

Data structures (not necessarily the dataset you are analyzing) in R can be in several forms, and we will begin with *vectors*. Vectors are sets of n elements, and you can generate them by using the function `c()` and listing individual numbers separated by commas. In this function, the arguments are the individual elements that need to be combined. If you need to generate a sequence of numbers, use the function `seq()` or if the numerals are separated by 1, you can use the colon; if you need to repeat something, the `rep()` function is the way to go. You can mix and match some of the functions, too.

```r
c(1,2,3,4)
```

```
## [1] 1 2 3 4
```

```r
seq(1,4, by=1)
```

```
## [1] 1 2 3 4
```

```r
c(1:4)
```

```
## [1] 1 2 3 4
```

```r
rep(seq(1,4, by=1), 2)
```

```
## [1] 1 2 3 4 1 2 3 4
```

Often, our datasets are not stored as vectors of data, but as matrices or dataframes. A matrix contains $(n \times m)$ numeric elements, where n is the number of rows and m is the number of columns. Functions like `rbind()` and `cbind()` can be used to glue vectors and matrices together, based on rows or columns. A dataframe also contains $(n \times m)$ elements, but these can be stored as numeric and/or other data types besides numeric (e.g. factor). Notice that the R displays matrices different than dataframes.

```r
matrix(rep(seq(1,4, by=1),2), nrow=4)
```

```
##      [,1] [,2]
## [1,]    1    1
## [2,]    2    2
## [3,]    3    3
## [4,]    4    4
```

```r
cbind(c(1,2,3,4), seq(1,4, by=1)) #This is still stored as a matrix
```

```
##      [,1] [,2]
## [1,]    1    1
## [2,]    2    2
## [3,]    3    3
## [4,]    4    4
```

```
rbind(c(1,2,3,4), seq(1,4, by=1)) #This is still a matrix
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    2    3    4
## [2,]    1    2    3    4
```

```
data.frame(cbind(c(1,2,3,4), seq(1,4, by=1)))
```

```
##   X1 X2
## 1  1  1
## 2  2  2
## 3  3  3
## 4  4  4
```

Another datastructure that might useful is an array, which is similar to a matrix, but has more than two dimensions.

```
array(cbind(c(1,2,3,4), seq(1,4, by=1)),dim=c(3,2,2))
```

```
## , , 1
##
##      [,1] [,2]
## [1,]    1    4
## [2,]    2    1
## [3,]    3    2
##
## , , 2
##
##      [,1] [,2]
## [1,]    3    2
## [2,]    4    3
## [3,]    1    4
```

```
array(cbind(c(1,2,3,4), seq(1,4, by=1)),dim=c(2,2,2,2))
```

```
## , , 1, 1
##
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
##
## , , 2, 1
##
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
##
## , , 1, 2
##
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
##
```

```
## , , 2, 2
##
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
```

Finally, another traditional datastructure that you might encounter is a 'list', which has can have many elements, data types and objects of different dimensions.

```
list(cbind(c(1,2,3,4), seq(1,4, by=1)),data.frame(rep(2,4)),c(1:5))
```

```
## [[1]]
##      [,1] [,2]
## [1,]    1    1
## [2,]    2    2
## [3,]    3    3
## [4,]    4    4
##
## [[2]]
##   rep.2..4.
## 1         2
## 2         2
## 3         2
## 4         2
##
## [[3]]
## [1] 1 2 3 4 5
```

OK, so we can generate these sequences, but mostly we need to save these as 'variables' or 'objects'. In R, I find the best practice is to use '<-' to assign a value to object. An object is stored in the computer memory while your R session is active, and can be named whatever is convient to you, but it cannot start with a numeral and cannot include spaces. Capitalization matters for object names, and it is a general rule to not name an object that is the same name as a function. For instance, don't name an object `mean` because there is a function `mean()`, so instead name the object `myMean` or something.

```
var1<-c(1:4)
var2<-matrix(rep(seq(1,4, by=1),2), nrow=4)
var3<-data.frame(cbind(c(1,2,3,4), seq(1,4, by=1)))
var4<-array(cbind(c(1,2,3,4), seq(1,4, by=1)),dim=c(3,2,2))
var5<-list(cbind(c(1,2,3,4), seq(1,4, by=1)),data.frame(rep(2,4)),c(1:5))
```

Now we can do things, like multiply or identify elements within an variable...

```
var1*.4 #multiply each element by .4
```

```
## [1] 0.4 0.8 1.2 1.6
```

```
var1*var2 #performs matrix multiplication
```

```
##      [,1] [,2]
## [1,]    1    1
## [2,]    4    4
## [3,]    9    9
## [4,]   16   16
```

```
var3[2,] #yields the second row
```

```
##   X1 X2
## 2  2  2
```

```
var3[,2] #yields the second column
```

```
## [1] 1 2 3 4
```

```
var4[,1,2] #yields the first column of the second matrix
```

```
## [1] 3 4 1
```

```
var5[[2]] #yields the second item of the list
```

```
##   rep.2..4.
## 1         2
## 2         2
## 3         2
## 4         2
```

```
var5[[1]][3,2] #yields the element in the third row, second column in the second item on the list
```

```
## [1] 3
```

Other fun and useful functions

```
runif(5) #draw 5 random numbers from a uniform distirbution between 0 and 1
```

```
## [1] 0.2182858 0.8424067 0.7120580 0.9674401 0.9247717
```

```
runif(5, 1,10) #draw 5 random numbers from a uniform distribution between 1 and 10
```

```
## [1] 9.511835 3.618496 9.818324 4.376753 8.510775
```

```
rnorm(5) #draw 5 random numbers from a normal distribution, mean of 0, sd of 1
```

```
## [1] -1.1801059 -0.5742422  1.1672223 -1.0522932 -1.3051982
```

```
rnorm(5,1,2) #draw 5 random numbers from a normal distribution, mean of 1, sd of 2
```

```
## [1] -0.2736560  4.9242422  0.1503116  0.1112048 -1.0558610
```

```
sample(var1) #sample without replacement from var1
```

```
## [1] 2 1 3 4
```

```
sample(var1, replace=TRUE) #sample with replacement from var1
```

```
## [1] 2 2 4 4
```

More often, we need to read in data from an external file. Common file types include comma separated files (.csv) and tab delimited files (.txt). Other formats, such as Excel files (.xls, .xlsx) or Minitab worksheets (.mtp) require other packages (we'll get to them in a bit) to import into R.

Note that I use forwardslashes ('/') rather than backslashes (") when identifying the folder to import the data.

```
#Climate data for Springfield
prism <- read.csv('c:/users/sean maher/dropbox/Teaching/Biostatistics/springfield_prismdata.csv')
head(prism)
```

```
##   Year Month tmin_deg_F tmax_deg_F ppt_inches
## 1 2002     1       22.9       46.2       3.04
## 2 2002     2       26.1       48.1       1.34
## 3 2002     3       29.3       52.2       4.50
## 4 2002     4       46.5       69.2       3.50
## 5 2002     5       51.1       71.2       9.90
## 6 2002     6       63.4       83.7       1.31
```

```
class(prism)
```

```
## [1] "data.frame"
```

```
#Mammal data
rmnp <- read.csv('c:/users/sean maher/dropbox/Teaching/Biostatistics/rmnp_data.csv')
head(rmnp)
```

```
##       Location      Date dumb.date AM.PM                Species Sex TTL TL
## 1 Mid-Upper BB 24-Jul-07 24-Jul-07    AM Peromyscus maniculatus   F 130 54
## 2 Mid-Upper BB 26-Jul-07 26-Jul-07    AM Peromyscus maniculatus   F  NA 51
## 3 Mid-Upper BB 26-Jul-07 26-Jul-07    AM Peromyscus maniculatus   F  NA 52
## 4 Mid-Upper BB 24-Jul-07 24-Jul-07    AM Peromyscus maniculatus   F 138 52
## 5 Mid-Upper BB 24-Jul-07 24-Jul-07    AM Peromyscus maniculatus   F 153 61
## 6 Mid-Upper BB 25-Jul-07 25-Jul-07    AM Peromyscus maniculatus   F 135 54
##   HF EL Mass ElevOrder Elevation
## 1 17 16 11.0         8  3043.733
## 2 18 16 11.0         8  3043.733
## 3 18 17 12.5         8  3043.733
## 4 16 17 16.5         8  3043.733
## 5 17 13 19.0         8  3043.733
## 6 18 19 19.5         8  3043.733
```

Let's get some summary statistics from our data…

```
#Provides general summaries for each column
summary(prism)
```

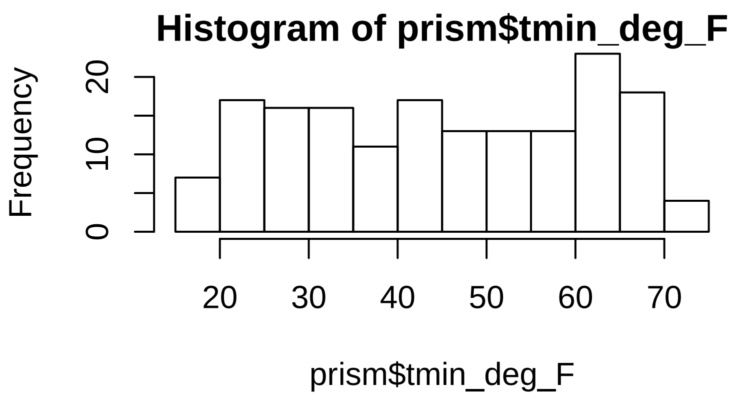
```
##       Year          Month         tmin_deg_F      tmax_deg_F
##  Min.   :2002   Min.   : 1.00   Min.   :15.20   Min.   :35.80
##  1st Qu.:2005   1st Qu.: 3.75   1st Qu.:31.20   1st Qu.:52.15
##  Median :2008   Median : 6.50   Median :45.05   Median :68.40
##  Mean   :2008   Mean   : 6.50   Mean   :45.36   Mean   :66.70
##  3rd Qu.:2012   3rd Qu.: 9.25   3rd Qu.:61.25   3rd Qu.:82.30
##  Max.   :2015   Max.   :12.00   Max.   :71.90   Max.   :97.30
##    ppt_inches
##  Min.   : 0.250
##  1st Qu.: 2.353
##  Median : 3.520
##  Mean   : 4.009
##  3rd Qu.: 5.388
##  Max.   :14.220
```

```
#Calculate mean for each month
mean.tmin.month<-aggregate(tmin_deg_F~Month,data=prism,mean)


#Calculate mean for each year
mean.precip.year<-aggregate(ppt_inches~Year,data=prism,mean)

#Tabulate counts of species by location
rmnp.sum<-table(rmnp$Species, rmnp$Location)
rmnp.sum
```

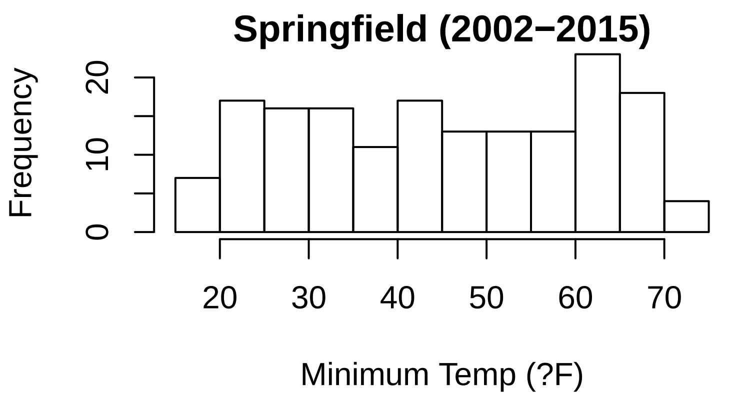
```
##
##                               Cow Creek Hollowell Park Lower BB
##   Callospermophilus lateralis         8             15        0
##   Myodes gapperi                      0              0        6
##   Neotoma cinerea                     0              0        8
##   Peromyscus maniculatus             23             64       29
##   Tamias minimus                     15             10        6
##   Tamias umbrinus                     4              0       13
##   Urocitellus elegans                 0             11        0
##
##                               Mid-Upper BB Mid BB RNF Upper BB WindRiver A
##   Callospermophilus lateralis            7      2   0        1           2
##   Myodes gapperi                         0     16   0       17           1
##   Neotoma cinerea                        0      1   0        0           4
##   Peromyscus maniculatus                21     25  32       20          32
##   Tamias minimus                        10      0   0       18           2
##   Tamias umbrinus                        1     12   0        0           6
##   Urocitellus elegans                    0      0   0        0           0
##
##                               WindRiver B
##   Callospermophilus lateralis           2
##   Myodes gapperi                        0
##   Neotoma cinerea                       0
##   Peromyscus maniculatus                5
##   Tamias minimus                        0
##   Tamias umbrinus                       3
##   Urocitellus elegans                   0
```

What is the first thing we should do with our data? Plot it! R has a base set of plotting functions that take a little time to sort out, but allow for flexibility. Let's start with histograms:

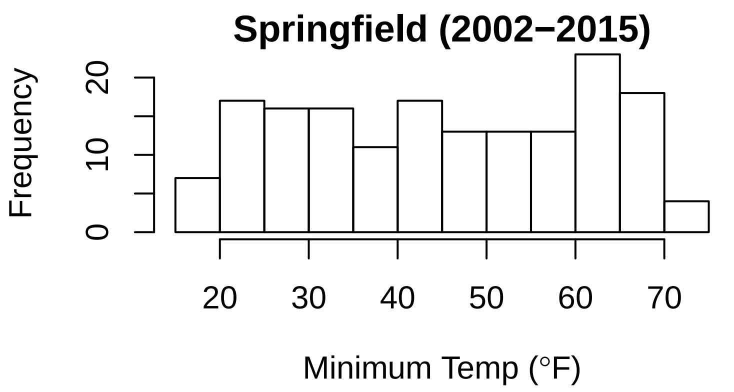
```r
par(mar=c(4,4,1,1)) #this changes the margins around plots
hist(prism$tmin_deg_F) #histogram of minimum temperatures
```



```r
hist(prism$tmin_deg_F, #histograms of log10(meadow area)
     xlab='Minimum Temp (?F)', #let's apply our own label
     main='Springfield (2002-2015)') #and adjust the title
```

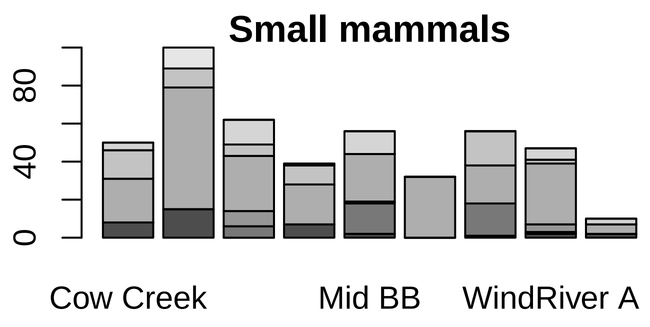


```r
hist(prism$tmin_deg_F, #histograms of log10(meadow area)
     xlab=expression(paste('Minimum Temp (',degree,'F)')), #alternative to above
     main='Springfield (2002-2015)')
```

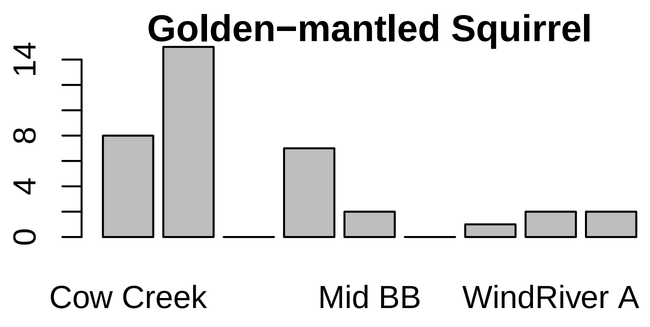


And barplots…

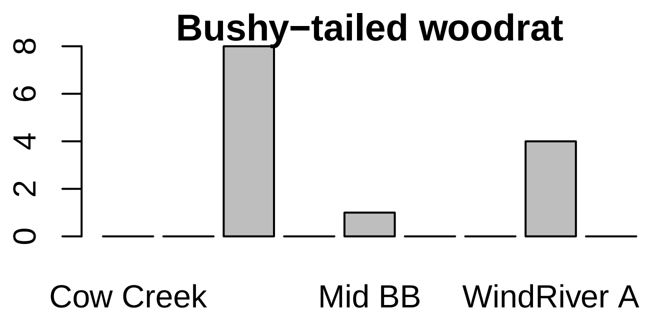
```r
par(mar=c(4,4,1,1)) #this changes the margins around plots
barplot(rmnp.sum, main="Small mammals")
```



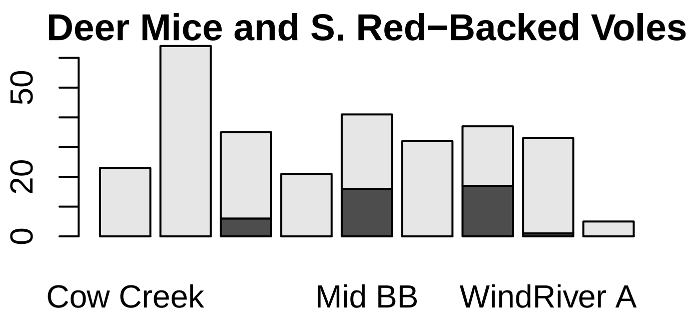
```r
barplot(rmnp.sum[1,], main="Golden-mantled Squirrel")
```



```r
barplot(rmnp.sum[3,], main="Bushy-tailed woodrat")
```

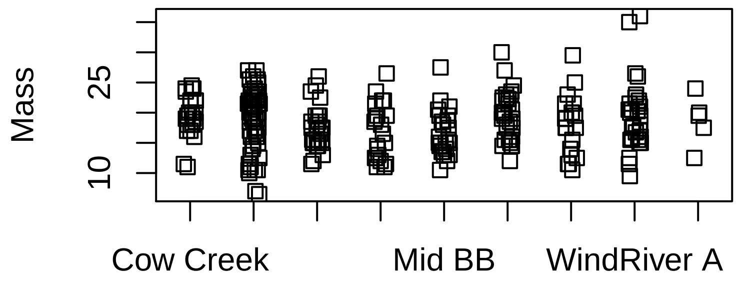


```r
barplot(rmnp.sum[c(2,4),], main="Deer Mice and S. Red-Backed Voles")
```

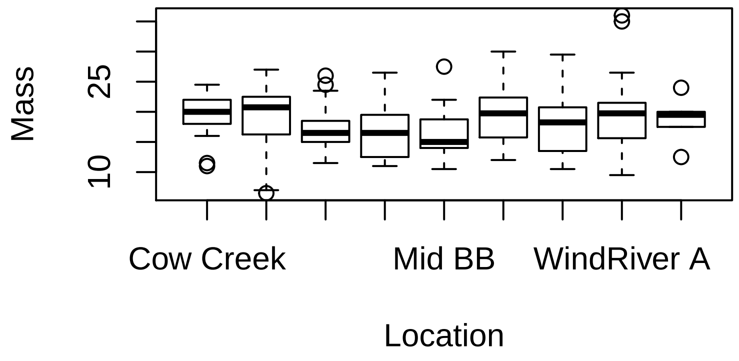


Now let's look at some additional plots:

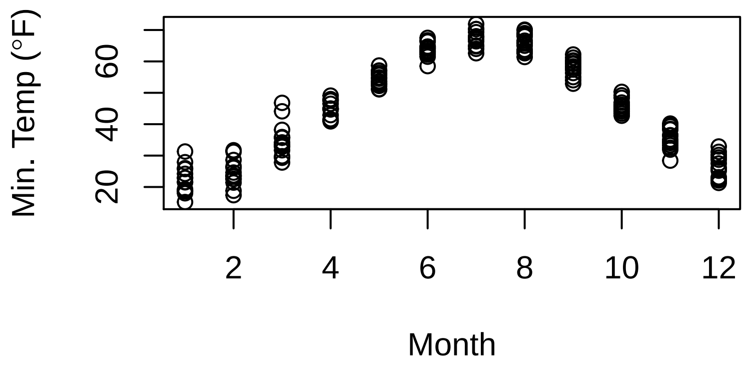
```
par(mar=c(4,4,1,1)) #this changes the margins around plots
stripchart(Mass~Location, data=subset(rmnp, Species=='Peromyscus maniculatus'), vertical=TRUE, method="
```



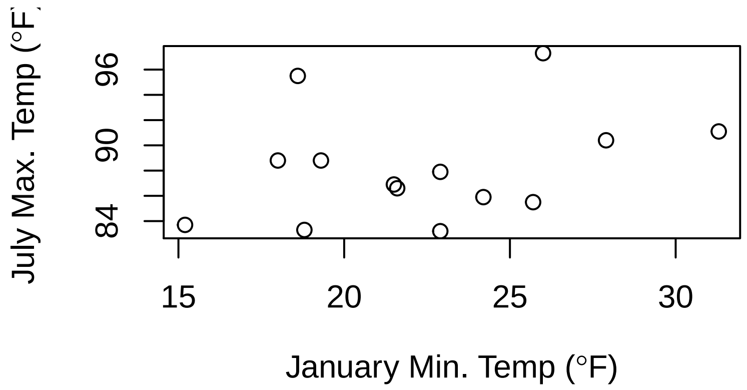
```
boxplot(Mass~Location, data=subset(rmnp, Species=='Peromyscus maniculatus'))
```



```
plot(prism$tmin_deg_F~prism$Month,
     xlab = "Month",
     ylab = expression(paste('Min. Temp (',degree,'F)')))
```



```r
plot(subset(prism$tmin_deg_F, prism$Month==1), subset(prism$tmax_deg_F, prism$Month==7),
     xlab = expression(paste('January Min. Temp (',degree,'F)')),
     ylab = expression(paste('July Max. Temp (',degree,'F)')))
```



One big reason to use R, is that it is statistics software. We'll see if minimum temp in January is correlated with maximum temperature in July.

```r
cor(subset(prism$tmin_deg_F, prism$Month==1), subset(prism$tmax_deg_F, prism$Month==7))
```

```
## [1] 0.3115645
```

```r
cor.test(subset(prism$tmin_deg_F, prism$Month==1), subset(prism$tmax_deg_F, prism$Month==7))
```

```
##
##  Pearson's product-moment correlation
##
## data:  subset(prism$tmin_deg_F, prism$Month == 1) and subset(prism$tmax_deg_F, prism$Month == 7)
## t = 1.1358, df = 12, p-value = 0.2782
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.2623907  0.7226783
## sample estimates:
##       cor
## 0.3115645
```

Packages are basically add-ons to R that provide additional functions and options. They can be for specific types of analysis (e.g. ape, adehabitat), different graphical options (e.g. lattice and ggplot2), or connecting R to other software (e.g R2Winbugs). R-Studio allows for a simple installation of packages to your hard drive, and the code to load one or more is straightforward:

```r
library(sp) #loads the library
```

```
## Warning: package 'sp' was built under R version 3.6.3
```

```r
library(maptools) #notice the error message!
```

```
## Warning: package 'maptools' was built under R version 3.6.3
```

```
## Checking rgeos availability: FALSE
##      Note: when rgeos is not available, polygon geometry     computations in maptools depend on gpcli
##      which has a restricted licence. It is disabled by default;
##      to enable gpclib, type gpclibPermit()
```

```r
require(raster) #also loads the library, but should be used in specific circumstances
```

```
## Loading required package: raster
```

```
## Warning: package 'raster' was built under R version 3.6.3
```

Just like R has different versions, packages can have different versions. What this means from a user stand-point, is that sometimes alterations to the underlying code may break your script or do something unexpected. Using R-studio can help with a bit of version control, but this could be problematic when using some of the campus machines where the drive is wiped after you log out.

Because we also want to give credit where credit is due, packages contain citation information:

```r
citation('base')
```

```
##
## To cite R in publications use:
##
##   R Core Team (2019). R: A language and environment for
##   statistical computing. R Foundation for Statistical Computing,
##   Vienna, Austria. URL https://www.R-project.org/.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {R: A Language and Environment for Statistical Computing},
##     author = {{R Core Team}},
##     organization = {R Foundation for Statistical Computing},
##     address = {Vienna, Austria},
##     year = {2019},
##     url = {https://www.R-project.org/},
##   }
##
## We have invested a lot of time and effort in creating R, please
## cite it when using it for data analysis. See also
## 'citation("pkgname")' for citing R packages.
```

```r
citation(package='raster')
```

```
##
## To cite package 'raster' in publications use:
##
```

```
##   Robert J. Hijmans (2020). raster: Geographic Data Analysis and
##   Modeling. R package version 3.0-12.
##   https://CRAN.R-project.org/package=raster
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {raster: Geographic Data Analysis and Modeling},
##     author = {Robert J. Hijmans},
##     year = {2020},
##     note = {R package version 3.0-12},
##     url = {https://CRAN.R-project.org/package=raster},
##   }
```

```r
citation('sp')
```

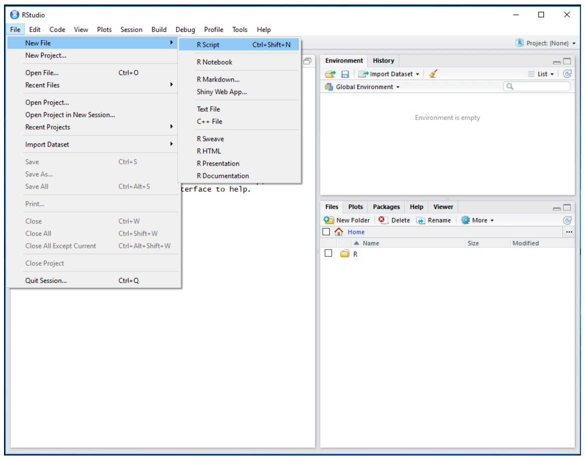
```
##
## To cite package sp in publications use:
##
##   Pebesma, E.J., R.S. Bivand, 2005. Classes and methods for
##   spatial data in R. R News 5 (2),
##   https://cran.r-project.org/doc/Rnews/.
##
##   Roger S. Bivand, Edzer Pebesma, Virgilio Gomez-Rubio, 2013.
##   Applied spatial data analysis with R, Second edition. Springer,
##   NY. https://asdar-book.org/
##
## To see these entries in BibTeX format, use 'print(<citation>,
## bibtex=TRUE)', 'toBibtex(.)', or set
## 'options(citation.bibtex.max=999)'.
```

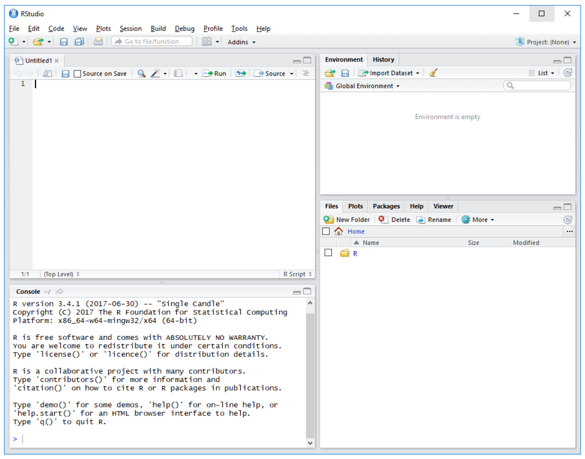
Using RStudio

RStudio works as shell to allow for a easier interaction with R. The RStudio window contains four panes that allow for viewing the script, console (or other function), environment (or history or connections), and plots (or packages, help, or files) simultaneously. This can make the user experience a bit more friendly because it limits the pop up windows and facilitates checking on available variables and figures. Further, RStudio includes features such as RMarkdown which allows you to generate readily available documents (inlcuding Word and PDF) that include data analysis and comments. For the record, I use RMarkdown to generate the lecture slides because I can create the formulas and plots much more easily than in other available formats.

If you want to work in RStudio rather than base R, that is fine with me. You can follow the base R instructions on the Blackboard site, but might have to tweak aspects of printing or saving documents.

To begin a Script you'll need to click File. . . New File. . . R Script

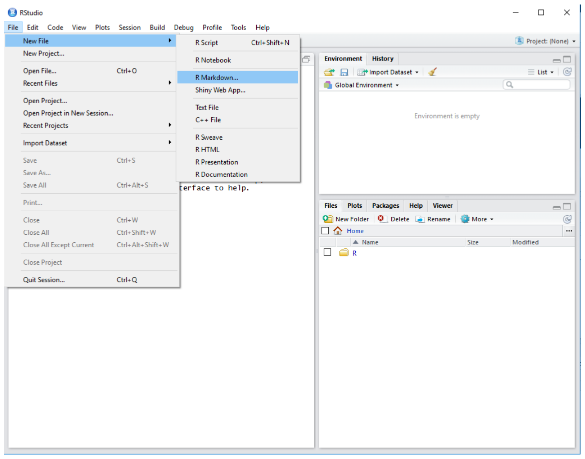


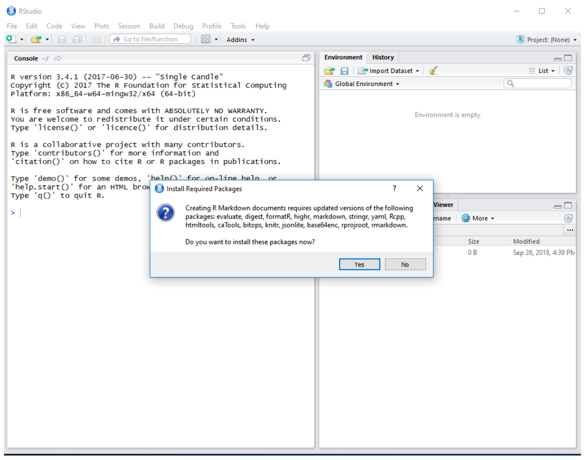


```
R version 3.4.1 (2017-06-30) -- "Single Candle"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

You can then type your commands in the script pane then send the commands to R using Ctrl+Enter or by highlighting the appropriate text and hitting the "Run" button on the top right of the pane. You would save your script by choosing File. . . Save (or Save As) and picking a place. Remember to add ".r" and the name to designate the file as an R script.

If you choose to explore RMarkdown, you would need to open a new R Markdown document, which on your first time will lead to a pop-up box to load several packages. Click "Yes" and then they will load. A new pop-up box will eventually appear to choose what type of output you want to create, and you will see this box immediately after the first time.



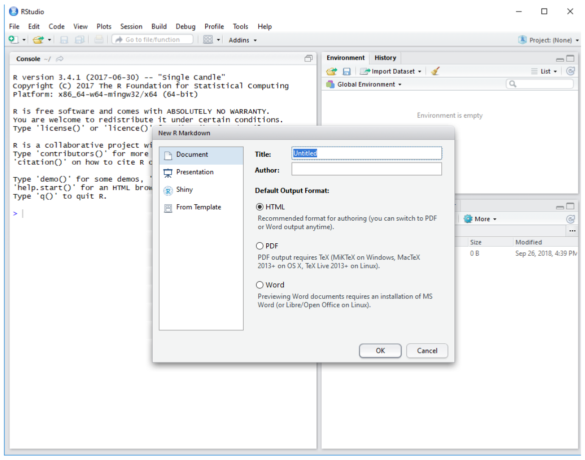


```
R version 3.4.1 (2017-06-30) -- "Single Candle"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML brow
Type 'q()' to quit R.

> |
```

Install Required Packages

Creating R Markdown documents requires updated versions of the following packages: evaluate, digest, formatR, highr, markdown, stringr, yaml, Rcpp, htmltools, caTools, bitops, knitr, jsonlite, base64enc, rprojroot, rmarkdown.

Do you want to install these packages now?



```
R version 3.4.1 (2017-06-30) -- "Single Candle"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()'

R is a collaborative project wi
Type 'contributors()' for more
'citation()' on how to cite R c

Type 'demo()' for some demos, '
'help.start()' for an HTML brow
Type 'q()' to quit R.

> |
```

New R Markdown

Title: Untitled

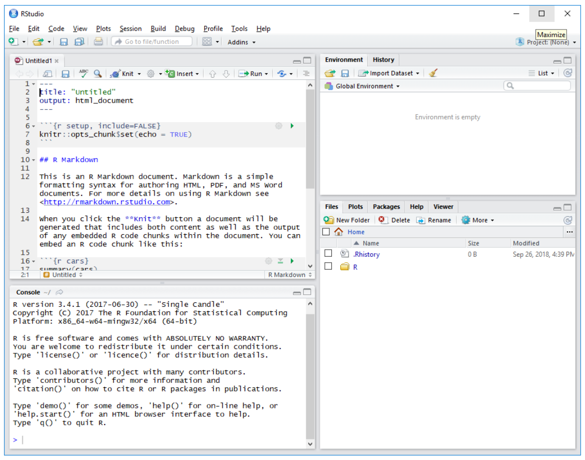
Author:

Default Output Format:

(●) HTML
Recommended format for authoring (you can switch to PDF or Word output anytime).

( ) PDF
PDF output requires TeX (MiKTeX on Windows, MacTeX 2013+ on OS X, TeX Live 2013+ on Linux).

( ) Word
Previewing Word documents requires an installation of MS Word (or Libre/Open Office on Linux).



~~~
---
title: "Untitled"
output: html_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
~~~

R Markdown

This is an R Markdown document. Markdown is a simple
formatting syntax for authoring HTML, PDF, and MS Word
documents. For more details on using R Markdown see
<http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be
generated that includes both content as well as the output
of any embedded R code chunks within the document. You can
embed an R code chunk like this:

```{r cars}
summary(cars)
```

```
R version 3.4.1 (2017-06-30) -- "Single Candle"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> |
```

If you want to generate PDFs or Beamer presentations, you will need to load another piece of software that can interpret LaTex code.

I recommend viewing the instructions at https://rmarkdown.rstudio.com/lesson-1.html if you really want to learn how to use RMarkdown effectively.

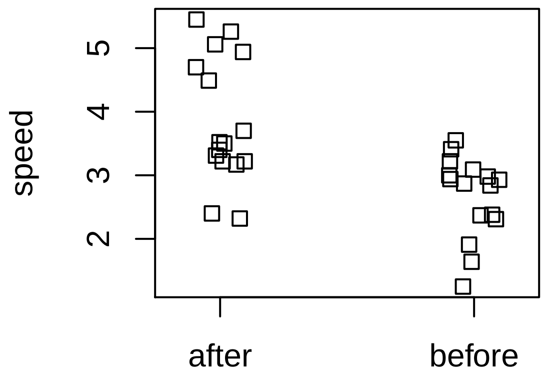
Homework Assigment 0

Generate an R script that accomplishes each of the following steps and email that script to [me at spmaher@missouristate.edu by the due date.

1. Create the following vector using both the functions seq() and rep().

   4, 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.7, 4.8, 4.9, 5, 5.1, 5.2, 2, 2, 2, 2

2. Read in the Spider Amputation data from chapter 3 (I posted it on Blackboard).
3. Use the stripchart() function to recreate a version of this graph. Set vertical=TRUE and method="jitter".



4. Use the function subset() twice to split the data into two new variables for 'before' and 'after', then determine the mean speed for each group using mean().

```r
before<-subset(spider, treatment=="before")
after<-subset(spider, treatment=="after")
```

```
## [1] 2.668125
```

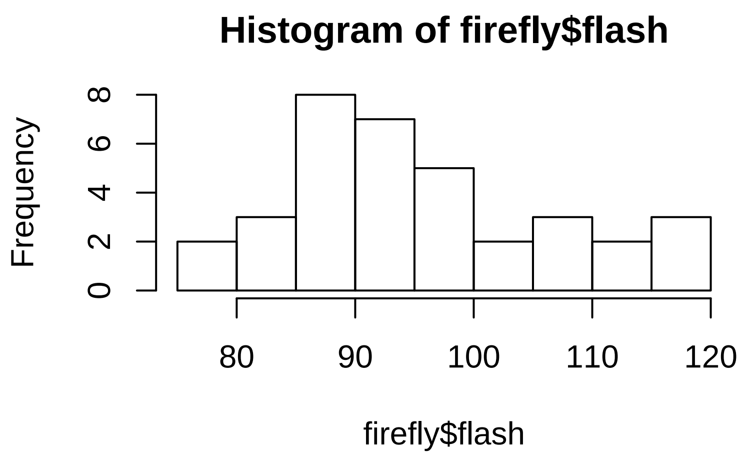
```
## [1] 3.85375
```

5. Find the standard error of each group, using the function sd() to find the standard deviation and sqrt() for the square root of the sample size.

```
## [1] 0.1603881
```

```
## [1] 0.248158
```

6. Read in the Firefly dataset, also posted on Blackboard.
7. Generate a histogram using the hist() function.



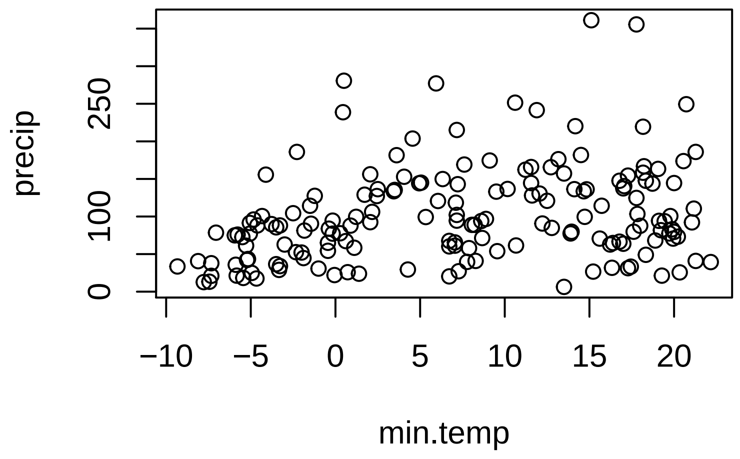
8. Use quantile(), with the arguments probs=c(0.25,0.75) and type=5, to find the first and third quartiles.

```
##    25%    75%
##  87.25 102.50
```

9. Take a random sample of 10 from the firefly data and calculate the mean of the sample. It should look like this…

```
mean(sample(firefly$flash,10))
```

10. Read in the PRISM dataset, also on Blackboard.
11. Convert the minimum temperature data to Celsius, and precipitation data to millimeters. You may want to store these as new variables.
12. Use plot() to create a scatterplot for minimum temperature in Celsius and precipitation in millimeters.


precip
min.temp