Blue text should not be altered.
Red text must be changed to fit each scenario. Replac with the names of data frames, variables, conditions, time points, etc. Examples:
Data = the name of your data frame, as shown in R Studio
Data.L = the name you give to a data frame in long format that you create by reshaping the data
Identifier = the name of a subject identifier (name, birthdate, social security, student id, etc)
Green text identifies a choice you must make. Choosing one option by deleting the others.

General Notes about R

R is the most type A friend you'll ever have. Spelling, punctuation and syntax must be 100% perfect. Be vigilant code writers and read error messages when they occur. They *might* help, and so might Google! Keep your code organized and annotated in a script or Markdown file. R is frustrating at first for many people. Be patient and persistent and you will someday love it and never again do statistics with Excel.

Descriptive Statistics

```
mean(Data$Variable)          median(Data$Variable)          sd(Data$Variable)          min(Data$Variable)          max(Data$Variable)
summary(Data$Variable)       table(Data$FactorVariable)
tapply(Data$QuantitativeVariable, Data$FactorVariable, mean/sd/min/max/summary)   # returns statistics for measured variable at each level of the factor variable
```

Data Wrangling

Revealing Characteristics of a Dataframe

```
str(Data)   # reveals the structure of dataframe.          View(Data)   # creates or refreshes a spreadsheet view of the data
```

Converting Numeric Variables to Factor Variables

```
Data$Variable <- factor(Data$Variable, levels=c(1, 2, 3, etc), labels=c('Level1name', 'Level2name', 'Level3name', 'etc'))
```

Ordering the levels of a Factor Variables

```
Data$Variable <- ordered(Data$Variable, levels=c('Level1name', 'Level2name', 'Level3name', 'etc'))
```

Subsetting Data

by Factor Levels: `DataSubset <- subset(Data, Variable=='Level')`

by Values of a Numeric Variable: `DataSubset <- subset(Data, Variable == #)`

or use: `!=` is not equal to `>=` greater than or equal to `<=` less than or equal to `>` greater than `<` less than

by Columns/Variables: `DataSubset <- subset(Data, select =c(#, #, #:#))` Within select = c(), include column numbers separated by commas or use the colon for a continuous run of columns. E.g. `select =c(1, 4, 6:9))` will create a subset that includes columns 1, 4, 6, 7,8, 9.

by Rows/Observations: `DataSubset <- Data[row#:row#]` Note that we use brackets instead of parentheses when subsetting by rows/observsations. Example: `Data1to5 <- Data[1:5]` This creates a data frame, Data1to5, with just the first 5 rows of Data .

Reshaping data in 3 Steps: wide → long (stacked):

Step 1: Installing and/or load tidyr package

```
>install.packages("tidyr")        # Install packages once per computer
>library(tidyr)                   # Load package once per session of RStudio.
```

Step 2: Reshaping the variable of interest using the gather() command

```
>Data.L <- gather(Data, CategoricalVariable, MeasuredVariable, columns)
```

Note: CategoricalVariable and MeasuredVariable are new columns that must be named.

Step 3: Renaming the levels of the new Categorical Variable

```
Data$CategoricalVariable <- factor(Data$CategoricalVariable, levels=c('Var.t0', 'Var.t1', etc), labels=c('Time 0', 'Time 1', etc))
```

Rename variables using the names() command.

```
names(Data) <-c("NewVar1Name","NewVar2Name ","NewVar3Name ", "etc")
```

Graphing

Installing & Loading ggplot2

```
install.packages("ggplot2")     # ggplot2 must be installed on personal computers only once. ggplot2 is installed on all Claire 111 computers.
library("ggplot2")              # load ggplot2 using the library() command every time you start R and at the beginning of .Rmd files.
```

Building up a ggplot graph, 2 Approaches:

1) Single Assignment: ggplots begin with `plot <- ggplot(data, aes(x=…)` command, which simply tells R what data and what variables to graph, but not *how* to graph them. Additional layers are needed, and are added with a + sign at the end of a line and hitting return to advance to the next line. R automatically indents the next line, and understands this syntax to mean the ggplot command continues. Continue adding one layer per line, ending each line with +, until the last layer is added. This format is succinct and easy to read.

2) Assignment & Reassignment: The ggplot() command containing the dataframe and variable names to be graphed is initially created as an object using the assignment operator "<-". The graph is incomplete though and cannot yet be displayed. In subsequent lines, BGraph is reassigning to the existing BGraph plus any layers that follow, again separated by a + sign. See **Bar Graph for Quantitative Data** below for an example.

IMPORTANT NOTE: Regardless of the approach, display the completed graph by simply writing the graphs name and executing it.

Optional Layers for ggplot Graphs

```
+ ggtitle("Graph Title Here")                    # Add title to graph. Insert one or more "\n" when title is too long to fit in graph
+ labs(x="X-axis Label Here", y= " Y-axis Label Here")   # Add axis labels to graph
+ facet_wrap(~CategoricalVariable)               # Plots the data in separate panels, one for each level of a categorical variable
+ theme_economist() or theme_wsj() or other      # Applies journal formatting. Must install and load ggthemes package
```

Bar graphs for count data (C distributions and C → C Relationships)

1 categorical variable:

```
BGraph <- ggplot(Data, aes(x=CategoricalVariable, fill=CategoricalVariable )) + geom_bar()
```

2 categorical variables:

```
BGraph <- ggplot(Data, aes(x=CategoricalVariable1, fill=CategoricalVariable2)) + geom_bar(position=position_dodge())
```

Using frequency values:

```
BGraph <- ggplot(Data, aes(ExplanatoryVariable, Freq_values))+ geom_bar(stat='identity', position=position_dodge())
```

Bar graphs for Quantitative Data (means with error bars of C → Q relationships and Long Formatted Data)

1 categorical variable

```
BGraph <- ggplot(Data, aes(x=ExplanatoryVariable1, y=ResponseVar))
BGraph <- BGraph + stat_summary(geom ='bar', fun.y = mean, fill='rcolor') + stat_summary(fun.data = mean_cl_normal/mean_sdl/mean_se, geom = "errorbar")
```

2 categorical variables

```
BGraph <- ggplot(Data, aes(x=ExpVar1, y=ResVar, fill = ExpVar2))
BGraph <- BGraph + stat_summary(geom ='bar', fun.y = mean, position=position_dodge(.95))
BGraph <- BGraph + stat_summary(fun.data = mean_cl_normal/mean_sdl/mean_se, geom = "errorbar", position=position_dodge(.95), width=0.2)
```

Histograms

```
HGraph <- ggplot(Data, aes(x=QuantitativeVariable)) + geom_histogram(color='black', fill = 'R Color')
```
#Optional to geom_histogram: `binwidth = #`

Boxplots

All boxes within a boxplot filled with a single R Color

```
BxPlot <- ggplot(Data, aes(x=ExplanatoryVariable, y=ResponseVariable)) + geom_boxplot(fill = 'R Color')
```

Each box within a boxplot filled with a unique color

```
BxPlot <- ggplot(Data, aes(x=ExplanatoryVariable1, y=ResponseVariable, fill=ExplanatoryVariable2)) + geom_boxplot()
```

Scatterplots & Linear Regression

A single linear relationship

```
SPlot <- ggplot(Data, aes(x=ExplanatorylVariable, y=ResponseVariable)) + geom_point() # creates a scatterplot object called SPlot
SPlot <- SPlot + stat_smooth(method="lm", se = FALSE)          # add a single regression line
```

Multiple linear relationships split by a categorical variable

```
SPlot <- SPlot + geom_point(aes(color= CategoricalVariable))    # creates a scatterplot with color coded levels of a categorical variable (E.g. gender).
SPlot <- SPlot + stat_smooth(aes(color= CategoricalVariable), method="lm", se = FALSE)          #color coded regression lines
```

Statistical Tests

χ^2 test of Independence or Goodness of Fit

```
tbl <- table(Data$Handedness)
tbl                                  #note the order of the levels
chisq.test(tbl)                      #Goodness of Fit use chisq.test(tbl, p=c(frequency1, frequency2, frequency3, etc...)
```

t-Tests

1-Sample t-test: `t.test(Data$variable, mu=#)`

2-Sample Independent t-test: `t.test(ResponseVariable ~ ExplanatoryVariable, Data)`

2-Sample Paired t-test

Long Data Format: `t.test(ResponseVariable ~ ExplanatoryVariable, Data, paired= TRUE)`

Wide Data Format: `t.test(Data$ResponseVariable2, Data$ResponseVariable1, paired= TRUE)`

Optional Arguments:

```
alternative='two.sided/less/greater'      #Choose one form by deleting the other two.  The default is two.sided
conf.level=.95                            #The default is .95
```

Correlation

Between two variables: `cor(Data$ExplanatoryVar, data$ResponseVar)` `#requires the RcmdrMisc package`

Correlation matrix: `rcorr.adjust(Data[,c('measuredVar1', 'measuredVar2', 'measuredVar3', etc...)]`

Linear Regression & Multiple Regression

```
summary(lm(ResponseVariable ~ ExplanatoryVariable, Data))
summary(lm(ResponseVariable ~ ExplanatoryVariable1 + Explanatory Variable2 + ExplanatoryVariable3, Data))
```

ANOVA

1-Way ANOVA: `summary(aov(ResponseVariable ~ ExplanatoryVariable, Data))`

Bonferroni's post-hoc analysis for 1-Way ANOVA

```
pairwise.t.test(Data$ResponseVar, Data$ExplanatoryVar, p.adjust.method = "bonferroni")
```

Repeated Measures ANOVA (long format data)

```
install.packages('ez')      # Install packages once per computer
library(ez)                 # Load package once per RStudio session
ezANOVA(Data.L, dv=(ResponseVariable), wid =(Subjects), within =(ExplanatoryVariable), detailed=TRUE)
```

Bonferroni's post-hoc analysis for 1-Way Repeated Measures ANOVA

```
pairwise.t.test(Data.L$ResponseVariable, Data.L$ExplanatoryVariable, paired=TRUE, p.adjust.method='bonferroni')
```