A Beginner's Guide to Basic Statistics using R

Gregory S Gilbert

2021.03.25

Overview

This document provides model code for how to handle data and do basic statistical analyses in R. Here is an overview of topics and functions covered.

Handling data in R	Descriptive Stats	Relationships	Differences
Example data sets	mean()	cor() pearson	t.test() nonpaired
Importing / looking at data	sd() standard deviation	cor() spearman	t.test() paired
read.csv() import data	median()	lm() regression	plot() boxplot
head() tail() snapshot	quantile()	plot() scatterplot	aov() ANOVA
str() data structure	summary()	points() overlay	TukeyHSD() posthoc
Parts of objects [r,c]	sum()	abline() trendline	bartlett.test() variance
table() count table	length() dim()	–	shapiro.test() normality
aggregate() functions by groups	hist()	–	chisq.test() contingency

Handling data in R

Example data overview and access

Overview of data used in the data frames for this tutorial. For each, access the data using the Google sheets link, and download a .csv file to your project directory. You may need to slightly rename downloaded files to match that used in read.csv functions later in the the tutorial.

---

df1: two_sample_unpaired.csv
Above-ground biomass of California poppies grown in full sunlight or shade. First column represents a categorical variable with two states; the second column includes numerical measures of a continuous variable. n=12 for each.
https://docs.google.com/spreadsheets/d/1gNteUfXKCRam33d-NAOcCMzOzamEMnElSbEduvdoQsc

	A	B
1	treatment	mass_g
2	shade	16.8
3	shade	20.6
4	shade	22.9
5	shade	16.0
22	light	22.8
23	light	25.4
24	light	22.8
25	light	21.0

---

df2: two_sample_paired.csv
Volume of red or blue dyed 20% sucrose solution consumed by hummingbirds at each of 20 sites in two hour periods. First column represents the site number; the second and third columns show the mL of solution consumed as a continuous variable. Observations are paired because the feeders were hanging next to each other at each of the sites.
https://docs.google.com/spreadsheets/d/1vVJvDX7G3DxlWGjR2bl8viOdoy-noeonm5UAErskRuw

	A	B	C
1	site	red_mL	blue_mL
2	1	39.6	30.4
3	2	36.7	27.7
4	3	32.1	23.5
5	4	40.2	31.0
19	18	32.2	23.6
20	19	34.8	26.0
21	20	39.4	30.2

df3: four_variables.csv
Wing length (mm), wing width (mm), eye color, and wing pattern of a moth species. n=20. Used for summary statistics, regression, correlation, tables. The first two variables are continuous numerical values, and the second two variables are categorical. Well represents the observations on a single moth. https://docs.google.com/spreadsheets/d/1rRlU4XCm_nPiPV0bTzkP-juxmO5avfsGfYjbl3aP9Bc

	A	B	C	D
1	length	width	eye_color	wing_stripe
2	12.5	8.2	black	striped
3	13.2	8.1	black	plain
4	12.1	6.5	red	striped
5	14.8	9.0	black	striped
19	13.8	7.2	black	striped
20	13.5	7.1	red	plain
21	15.8	9.1	black	plain

---

df4: three_treatments.csv
Above-ground biomass of radish plants (oven-dry weight, in g) at 4 wk, from three treatments: control, irrigated (5mm water every 3 days), and fertigated (5mm human urine every 3 days).
https://docs.google.com/spreadsheets/d/1nvoOsgpNXa5iI-uBpzHAbfGOUcLl6P4oWxqKI8cEDvs

	A	B
1	treatment	biomass_g
2	control	5.4
3	control	3.27
4	control	5.38
5	control	4.93
35	fertigated	8.11
36	fertigated	8.87
37	fertigated	6.5

Importing and looking at your data

We will use the simple data set of **four_variables**, imported into data frame **df3**, to explore the basic functions used to generate summary statistics. The first step is to download the data as a .csv file to your project directory. I put mine inside the project directory folder, inside another folder called **data**.

	A	B	C	D
1	length	width	eye_color	wing_stripe
2	12.5	8.2	black	striped
3	13.2	8.1	black	plain
4	12.1	6.5	red	striped
5	14.8	9.0	black	striped
19	13.8	7.2	black	striped
20	13.5	7.1	red	plain
21	15.8	9.1	black	plain

Figure 1: Figure 1. Partial view of spreadsheet for summary statistics, regression, correlation: Wing length and width (mm), eye color, and wing pattern of a moth species. n=20. These data are used in the data frame df3

read.csv() Read in the data from a CSV file

The function read.csv() reads data from a comma-delimited file into a data frame. In this case we call the data frame df3. Your data should be arranged with the first row including variable names (avoid spaces and special characters except . or _). Each column is a variable of one type. Each row is a single observation, so that the values in each of the columns corresponds to a single observation. If there are missing data, leave that cell in the spreadsheet blank.

```
df3<-read.csv("data/four_variables.csv",as.is=FALSE)
```

This approach to the address for the CSV file to import assumes that it is inside your project directory and then inside a folder called **data**. There are a number of other ways for you to point to the CSV file that you want to import. The most flexible, is to use the function file.choose() to use your finder to browse for the file. Like this:

df3<-read.csv(file.choose(),as.is=FALSE)

You can also specify the full path to your file on your computer. Note that file that choose does not work within R markdown, but works very well just within the console. And easy way to get the path to a file on your computer is to use the file.choose() function, but instead of embedding it in the read.csv() function, just use it on its own in the console and it will show you the full path to your file. Like this:

file.choose()
[1] "/Users/greg/Dropbox/classes/ENVS104/Basic_Statistical_Tests/data/four_variables.csv"

Then you can embed that in the read.csv function like this:

df3<-read.csv("/Users/greg/Dropbox/classes/ENVS104/Basic_Statistical_Tests/data/four_variables.csv")

Here is a little script that will import each of the data frames used in this tutorial, after you have downloaded the .csv files from Google sheets and put them into your project directory in another folder called data.

```
df1<-read.csv("data/two_sample_unpaired.csv",as.is=FALSE)
df2<-read.csv("data/two_sample_paired.csv",as.is=FALSE)
df3<-read.csv("data/four_variables.csv",as.is=FALSE)
df4<-read.csv("data/three_treatments.csv",as.is=FALSE)
```

head() and tail() Look at the first six or last six lines of a data frame

It is always a good idea to look at your data after you have read it into R. You can use the head(x) function to look at the first six lines, and the tail(x) function to look at the last six lines. You can also specify the number of lines that you would like to look at with an additional numeric argument. Note 1: If you want to see all the data in your data frame, just type the name of the data frame. But if you have a lot of data, this can look pretty messy.

```
head(df3) #peek at first six lines of your data in data frame df3
```

```
##   length width eye_color wing_stripe
## 1   12.5   8.2     black     striped
## 2   13.2   8.1     black       plain
## 3   12.1   6.5       red     striped
## 4   14.8   9.0     black     striped
## 5   12.6   6.9     black       plain
## 6   16.2  10.0       red       plain
```

```
tail(df3, 4) #peek at the last four lines of your data
```

```
##    length width eye_color wing_stripe
## 17   16.6  10.1     black     striped
## 18   13.8   7.2     black     striped
## 19   13.5   7.1       red       plain
## 20   15.8   9.1     black       plain
```

str() Look at the structure (variable types and length) of your data

It is always a good idea to look at the structure of your data before you move forward with analysis. Many functions in R differ in behavior depending on the type of variable used as an argument. The str(x) function tells you what type of variable you have, as well as the dimensions of your data set.

```
str(df3) #look at the structure of your data
```

```
## 'data.frame':    20 obs. of  4 variables:
##  $ length     : num  12.5 13.2 12.1 14.8 12.6 16.2 10.6 13.4 14.1 11.1 ...
##  $ width      : num  8.2 8.1 6.5 9 6.9 10 4.7 8.5 8.7 5.8 ...
##  $ eye_color  : Factor w/ 2 levels "black","red": 1 1 2 1 1 2 2 2 1 1 ...
##  $ wing_stripe: Factor w/ 2 levels "plain","striped": 2 1 2 2 1 1 2 1 2 2 ...
```

Here we see that we have imported 20 observations from four variables. The first two observations are numeric, and the third and fourth are listed as factors. Factors are categorical values, And most often the way we want to read character values into data frames. Alternatively, those values could be represented as character strings, and then used as variables rather than categories.
Common data structure types in R include

num[#:##]: a vector of numbers
num[#:##, #:##]: a matrix of numbers
data.frame: A data frame where each column is a single variable type, and each row an observation
list: A complex structure in R that includes variables, descriptors, and formulas. Extractor functions allow you to access different components of a list.

Specifying or subsetting parts of data frames or vectors

R has a number of types of **objects**; all functions act on objects, together with other arguments that shape what to do with the object. Functions take the form of function_name(object, arguments).

vector: multiple values in one variable (e.g., vector a includes the first five even numbers)

```
a<-c(2,4,6,8,10)
a
```

```
## [1]  2  4  6  8 10
```

scalar: a variable with a single value (e.g., b<-6.2. b is a constant of value 6.2)

matrix: a 2-dimensional object of rows and columns, with every element of the same type. e.g.,a 3 x 4 (rows by columns) matrix m

```
m<-matrix(c(2,4,6,2,1,4,7,0,3,3,2,9),nrow=3, ncol=4, byrow=T)
m
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    2    4    6    2
## [2,]    1    4    7    0
## [3,]    3    3    2    9
```

data frame: 2-dimensional object where each column is a variable with values of one type, and each row is an observation. This is the most common way to arrange data. e.g., df1-4, above.

You can refer to variables (columns in a data frame using the structure `df$variable`) and they behave as vector objects and can be arguments for a function (e.g., `mean(df3$width)` gives the mean of variable width; `f<-a*5` creates a new vector f where each element of a is multiplied by five: 10, 20, 30, 40, 50)

You can use square bracket notation to refer to specific object elements [row,col] or create subsets of data.
`a[3]` will return the third element of a, which is 6
`m[2,3]` will return the element in the second row and third column of m, which is 7
`m[,4]` will return the vector of all elements in the fourth column: 2,0,9
`m[1,]` will return the vector of all elements in the first row: 2,4,6,2
`df3$length` will return the vector of all the elements of variable length
`df3$length[df3$wing_stripe=="plain"]` will return the subset of those values in `df3$length` where wing_stripe is plain.
`df3$wing_stripe[df3$width>=8]` will return a vector wing_stripe values where width is greater than or equal to 8.

Basic Statistics

Summary statistics of single variables: Histograms

mean() Mean of a set of values.

The mean(x) function calculates the mean of a set of values in a vector or a variable in a data frame. The function assumes that all observations in the vector are included (na.rm=FALSE). If there are any missing data (NA), add the argument na.rm=TRUE so that the means calculation ignores missing data.

```
mean(df3$length, na.rm=TRUE) #mean of variable length in data frame df3
```

```
## [1] 13.825
```

sd() Standard deviation of a set of values.

The sd(x) function calculates the standard deviation of a set of values in a vector or a variable in a data frame. The function assumes that all observations in the vector are included (na.rm=FALSE). If there are any missing data (NA), add the argument na.rm=TRUE so that the means calculation ignores missing data.

```
sd(df3$length, na.rm=TRUE) #st dev of variable length in data frame df3
```

```
## [1] 1.815105
```

median() Median of a set of values.

The median(x) function calculates the median of a set of values in a vector or a variable in a data frame. The function assumes that all observations in the vector are included (na.rm=FALSE). If there are any missing data (NA), add the argument na.rm=TRUE so that the means calculation ignores missing data. The median is the quantile for the 0.5 probability (half of the observations are larger, and half smaller than the median). See quantile() for additional details

```
median(df3$length, na.rm=TRUE) #median of variable length in data frame df3
```

```
## [1] 13.75
```

quantile() Quantiles of a set of values.

The quantile(x,probs) function calculates quantiles for of a set of values in a vector or a variable in a data frame. It assumes that all observations in the vector are included (na.rm=FALSE). If there are any missing data (NA), add the argument na.rm=TRUE so that the calculation ignores missing data.
The function quantile(x,probs) requires at least two arguments; first is the object that contains the values (a vector or variable in a data frame), and second is the probabilities for which to determine the quantiles. Probabilities can be represented in a couple ways.

As a vector: c(0, 0.025, 0.25,0.5,0.75,0.975,1)
As a function: seq(0,1,0.25) which gives the sequence of values from 0 to 1 in intervals of 0.25.
Note 1: quantile(x,0.5) returns the median of the values in x Note 2: There are a number of different ways to calculate quantiles. You can see the nine different options available through the *type* argument by looking at the full R documentation: search for *quantile* in the help function. The default is type=7.

```
quantile(df3$length, probs=seq(0,1,0.1), na.rm=TRUE) #Quantiles for each 10% probability value of varia
```

```
##    0%   10%   20%   30%   40%   50%   60%   70%   80%   90%  100%
## 10.60 11.37 12.42 13.02 13.46 13.75 14.18 14.59 15.24 16.24 17.20
```

summary() Summary statistic of a set of values.

The summary(x) function is a strange function that does different things depending on what kind of arguments you put in. When applied to a single variable it returns some useful summary statistics. It assumes that all observations in the vector are included (na.rm=FALSE). If there are any missing data (NA), add the argument na.rm=TRUE so that the calculation ignores missing data.

```
summary(df3$length, na.rm=TRUE) #summary statistics of variable length in data frame df3
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   10.60   12.57   13.75   13.82   14.88   17.20
```

The summary(x) function actually returns a vector of six values. You can access each of the different elements in the vector independently

```
summary(df3$length, na.rm=TRUE)[3] #the third element is the median
```

```
## Median
##  13.75
```

```
summary(df3$length, na.rm=TRUE)[6] #the 6th element is the max
```

```
## Max.
## 17.2
```

```
out1<-summary(df3$length, na.rm=TRUE) #put into object out1
out1[c(1,3,6)] #show elements 1, 3, 6
```

```
##   Min. Median   Max.
##  10.60  13.75  17.20
```

sum() If the sum of the values in a set

The sum(x) function calculates the sum of a set of values in a vector or a variable in a data frame. The function assumes that all observations in the vector are included (na.rm=FALSE). If there are any missing data (NA), add the argument na.rm=TRUE so that the calculation ignores missing data.

```
sum(df3$length, na.rm=TRUE) #sum of values in variable length in data frame df3
```

```
## [1] 276.5
```

length() and dim() Number of values in a vector or dimensions of a dataframe

The length(x) function gives you the number of elements in a vector, or the number of elements in a single column of a data frame. You can also use the dim(x) function to get the number of rows (values) and number of columns (variables) in a data frame.The output of dim() function is a vector of 2 elements [rows,columns], and you can reference them independently.

```
length(df3$length) #length of values in variable length in data frame df3
```

```
## [1] 20
```

```
dim(df3) # number of rows, columns in data frame df3
```

```
## [1] 20  4
```

```
dim(df3)[2] #Number of variables in data frame df3
```

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

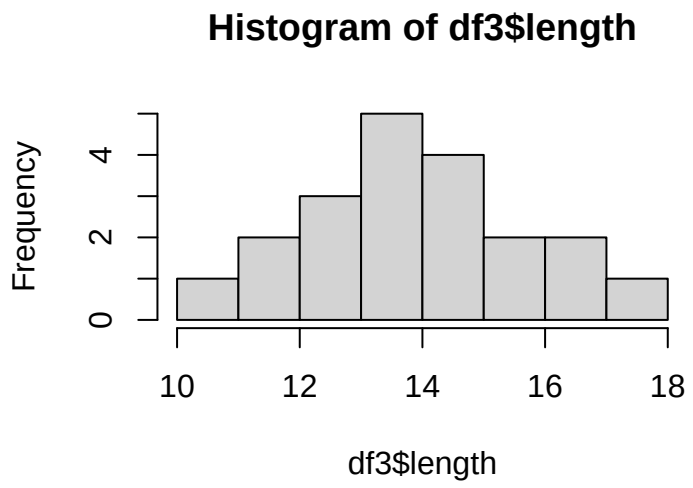
Importantly, both these functions do not differentiate between elements c ontaining values and those with missing data (NAs), and there is no na.rm argument available. If you want to count only the number of non-missing values in the length function, you need to specify not to include missing values. The ! signifies "not" in R-speak.

length(x[!is.na(x)]) #length of variable x where the value of x is not missing

hist() Histogram of values in a single variable

The most fundamental way to summarize the values of a continuous variable graphically is in a histogram. The histogram counts the frequency of values in arbitrarily divided bins along the continuous variable. Let's first look at the default histogram produced by R, and then we will look at making it a little prettier.

```
hist(df3$length) #default histogram of variable length in data frame df3
```



A few things you can do that help make the figure stronger:
1. Axes should have informative labels, and include units whenever possible. Use xlab and ylab arguments to set those levels.
2. Figures should generally not have a title above them. We move the title by setting the option main=NA.
3. The values on the axes should generally all read horizontally. You can set this with the option las=1.
4. You might want more or less resolution in the bin sizes and can set these with the brakes option. This can be done in two ways, either by setting the number of bins (breaks=6), or by specifying the start values for each bin [e.g., breaks=c(4,8,12,16,20,24,28) or breaks=seq(4,26,2)].
5. Can you change the color of the bars if you would like, but please choose colors that are friendly on the eyes.
6. In R markdown you can add caption to the figure in the header of the code like this.

```
hist(df3$length,
     main=NA,
     xlab='Wing length (mm)',
     ylab='Number of moths',
     breaks=seq(10,18,1),
     las=1,
     col='lightblue')
box() # puts boundary box around figure
```

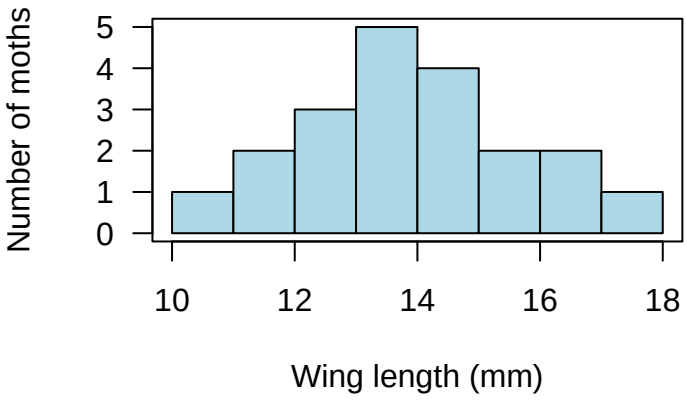


Figure 2: Figure 4. Length of wings of the Santa Cruz brown moth (n=20).

Making summary tables of multiple groups

table() tabulate counts within categories

Use the table() function to count the number of observations within categories (factors). Tables can be one dimensional (e.g., number of individuals in each category within a single variable) or multi-way, contingency tables, that count the number of observations in intersections of categories.

```r
# Count the number of moths with each eye color
table(df3$eye_color)
```

```
##
## black   red
##    13     7
```

```r
# Contingency count the number of moths with each eye color and wing color combination
table(df3$eye_color,df3$wing_stripe)
```

```
##
##         plain striped
##   black     4       9
##   red       4       3
```

aggregate() calculates summary statistics on subsets (groups) of data

Use the aggregate() function to divide the data into subsets (categories, often based on factors) and then calculate all kinds of summary statistics for each subset. Aggregate() takes three important arguments: aggregate (x, by, FUN)
x is the variable you want to summarize
by indicates the variable used to subset the data. Note that you must enclose the variable name in list(), because the function expects the categories to be presented in that form.
FUN is the function for the desired summary statistic. You can use almost any summary statistics function you can apply to a vector.
In some cases, there will be additional options available.

For instance, we can calculate summary statistics of wing length of moths with plain ro striped wing patterns in **df3**

```r
#calculate the mean wing length of moths in each wing pattern group
aggregate(df3$length,by=list(df3$wing_stripe),FUN=mean)
```

```
##   Group.1        x
## 1   plain 14.18750
## 2 striped 13.58333
```

```r
#or embed aggregate() in setNames() to add informative column names to the output
setNames(aggregate(df3$length,by=list(df3$wing_stripe),FUN=mean),c("Wing_stripe","mean_length"))
```

```
##   Wing_stripe mean_length
## 1       plain    14.18750
## 2     striped    13.58333
```

```r
#you some FUNctions requires additional arguments, which just follow a comma.
#Calculate the min, 1st quartile, median, 3rd quartile, and max for length
aggregate(df3$length,by=list(df3$wing_stripe),FUN=quantile, probs=c(0, .25, .5, .75, 1))
```

```
##   Group.1   x.0%  x.25%  x.50%  x.75% x.100%
## 1   plain 12.600 13.350 13.600 15.275 16.200
## 2 striped 10.600 11.925 13.950 14.575 17.200
```

```
#you can assign the output of aggregate() to an object (data frame) for further use.
#calculate the mean for each group
mn<-setNames(aggregate(df3$length,by=list(df3$wing_stripe),FUN=mean),c("Wing_stripe","mean"))
#calculate the standard deviation for each group
sd<-setNames(aggregate(df3$length,by=list(df3$wing_stripe),FUN=sd),c("Wing_stripe","sd"))
#calculate the number of individuals for each group
num<-setNames(aggregate(df3$length,by=list(df3$wing_stripe),FUN=length),c("Wing_stripe","n"))
msn<-merge(mn,sd); msn<-merge(msn,num) #merge the objects by Wing_stripe
msn #show the resulting summary
```

```
##   Wing_stripe     mean       sd  n
## 1       plain 14.18750 1.326044  8
## 2     striped 13.58333 2.100577 12
```

Relationships between variables

Correlations

cor() Correlation between two sets of data

The **Pearson correlation coefficient** r is a measure of the linear correlation between two variables. Pearson's r ranges from -1 to 1. r = -1 means a plot of the two variables x and y fall on a line where y decreases as x increases; $r = 1$ indicates they fall on a line where y increases as x increases. $r = 0$ indicates no correlation between them.
The **Spearman rank correlation coefficient** ρ (rho or r_s) is a non-parametric, rank-based correlation between two variables. It asks if the two variable change together in a monotonic way, but not if that relationship is linear. ρ is -1 or +1 if one variables is a perfect monotonic function of the other.
Both of these correlation coefficients can be testes using **cor()**.

```
cor(df3$length,df3$width,method="pearson") #Pearson correlation
```

```
## [1] 0.8493295
```

```
cor(df3$length,df3$width,method="spearman") #Spearman rank correlation
```

```
## [1] 0.8412342
```

Present statistics in the text:
The lengths and widths of wings of the Santa Cruz brown moth were strongly positively correlated (r = 0.85).

There are several possible, but approximated, statistical tests of significance (is r or ρ significantly different from zero) available using the test.cor() function. However, in most cases, we don't need to invoke a test of significance to make inferences; we are more interested in the correlations as descriptors of relationships than of tests of relationships. The closer to |1|, the stronger the correlation. If you are interested in statistical tests of relationships between variables, use regression.

lm() Linear regression

Calculate a linear regression $y = mx + b$ of two continuous variables. In this case y is the dependent and x is the independent variable. m is the slope,, and b is the intercept.
In the case of the relationship between the length and width of moth wings. In R, we express y as a function of x as $y \sim x$

```
# calculate a linear regression of lenght as a function of width
lm(df3$length~df3$width)
```

```
##
## Call:
## lm(formula = df3$length ~ df3$width)
##
## Coefficients:
## (Intercept)    df3$width
##       5.346        1.072
```

This gives you the intercept and slope values: $length = 1.072 * width + 5.346$

To ask whether the slope is significantly different from zero, and what the R^2 value is, we need to use an **extractor function**, either summary() or anova()

First, lets send the regression output into an object we'll call lmout1. Then we can do several things to that object. * look at the structure of the object str() * look at the summary of the object summary() * plot the points in a scatterplot, and overlay the regression line (abline())

```
#make an object lmout1 using the lm() function
lmout1 <- lm(df3$length~df3$width)
#lmout1 is an object of type list with many hidden components
#try typing in the console
#str(lmout1)
#to see what is inside the list object lmout1
#Fortunately, you can easily extract the statistics you need from lmout1 using the extractor function s
summary(lmout1)
```

```
##
## Call:
## lm(formula = df3$length ~ df3$width)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.6359 -0.6326 -0.0963  0.5823  1.7145
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    5.346      1.261   4.238 0.000495 ***
## df3$width      1.072      0.157   6.826 2.17e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9844 on 18 degrees of freedom
## Multiple R-squared:  0.7214, Adjusted R-squared:  0.7059
## F-statistic:  46.6 on 1 and 18 DF,  p-value: 2.17e-06
```

Look to see where to find the statistics you need to evaluate the line. This output shows the estimates for the Intercept (5.346) and slope (1.072). The Adjusted $R^2 = 0.7059$. The F statistic is 46.6, with degrees of freedom 1 and 18, and a corresponding p-value = 2.17e-06 (0.00000217). The p-value tell you if the slope is significantly different from zero (note that it is the same as the value for Pr(>|t|) for df3$width). Can you find all those numbers in the output?

Present statistics in the text, calling out to Figure 3 (see next section):
There was a strong linear relationship between the length and width of the wings of the Santa Cruz brown moth (Fig. 3; $length = 5.346 + 1.072 * width$, $F_{1,18}$=46.6, $P \leq 0.000002$, R^2_{adj}=0.706).

'There was a strong linear relationship between the length and width of the wings of the Santa Cruz brown moth (Fig. 3; $length = 5.346 + 1.072 * width$, $F_{1,18}$=46.6, $P \leq 0.000002$, R^2_{adj}=0.706).

plot() Plot to create a scatterplot

Let's then look at the same relationship between length and width of the moth wings graphically, as a scatterplot with the dependent variable on the vertical axis. We can then overlay the best-fit line from the linear regression above on top of the scatterplot.

The plot() function will create a scatterplot from either of two formulations plot(y~x) or plot(x,y).

```
plot(df3$length~df3$width,
     xlab='Wing width (mm)',ylab= 'Wing length (mm)',
     las=1, pch=19, col="orange")
abline(lmout1) #draws the regression line using slope and intercept from lmout1
```

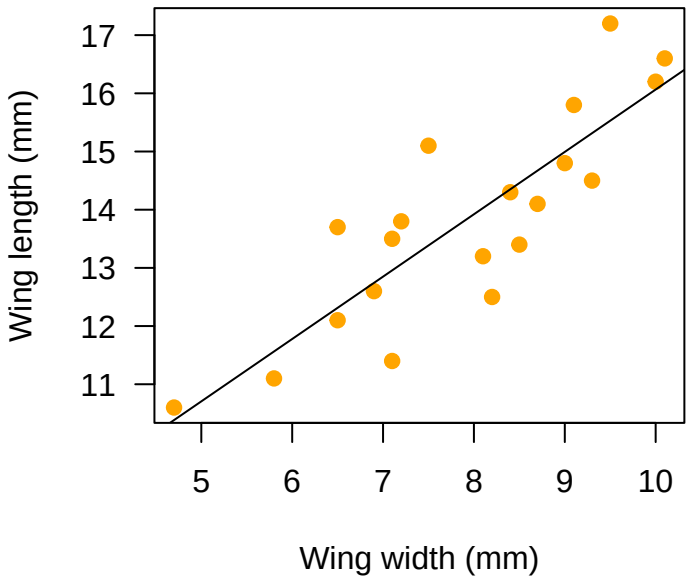


Figure 3: Figure 3. Relationship between the wing length and width of the Santa Cruz brown moth.

points(x,y) Create overlays on a plot

Let's imagine we want to analyze and plot the wing length~width independently for moths with striped or plain wings. Here is one way to do that: +create separate data frames for striped and plain morphs of moths using []
+do separate regressions of *length width* for each morph using lm()
+create a scatterplot with different symbols for each morph plot() and points()

```
striped<-df3[df3$wing_stripe=="striped",] #make new df with only striped
plain<-df3[df3$wing_stripe=="plain",] #make new df with only plain
#peak at each
head(striped); head(plain)
```

```
##    length width eye_color wing_stripe
## 1    12.5   8.2     black     striped
## 3    12.1   6.5       red     striped
## 4    14.8   9.0     black     striped
## 7    10.6   4.7       red     striped
## 9    14.1   8.7     black     striped
## 10   11.1   5.8     black     striped
```

```
##    length width eye_color wing_stripe
## 2    13.2   8.1     black       plain
## 5    12.6   6.9     black       plain
## 6    16.2  10.0       red       plain
## 8    13.4   8.5       red       plain
## 12   15.1   7.5       red       plain
## 15   13.7   6.5     black       plain
```

```
#make the two linear regressions
stripeout<-lm(striped$length~striped$width)
plainout<-lm(plain$length~plain$width)

#look at the statistics
summary(stripeout)
```

```
##
## Call:
## lm(formula = striped$length ~ striped$width)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.45986 -0.51291  0.01454  0.50294  1.73404
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     4.4598     1.4256   3.128   0.0107 *
## striped$width   1.1585     0.1776   6.524 6.69e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9609 on 10 degrees of freedom
```

```
## Multiple R-squared:  0.8098, Adjusted R-squared:  0.7907
## F-statistic: 42.57 on 1 and 10 DF,  p-value: 6.686e-05
```

```
summary(plainout)
```

```
##
## Call:
## lm(formula = plain$length ~ plain$width)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.2155 -0.8303  0.1947  0.6845  1.2808
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   7.8468     2.5413   3.088   0.0214 *
## plain$width   0.7963     0.3161   2.519   0.0453 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9984 on 6 degrees of freedom
## Multiple R-squared:  0.5141, Adjusted R-squared:  0.4331
## F-statistic: 6.348 on 1 and 6 DF,  p-value: 0.04532
```

From these statistics, we can see that both of the regression lines are statistically significant (slopes are different from zero, $P \leq 0.05$). We can go ahead with the scatterplot and superimposed trend lines in a figure.

```
#make a plot with striped and then overlay plain.
plot(striped$length~striped$width,
     xlab='Wing width (mm)',ylab= 'Wing length (mm)',
     las=1, pch=19, col="orange")
points(plain$width,plain$length, pch=1)
abline(stripeout,lty=1)
abline(plainout, lty=2)
```

#Differences between and among groups ## Two-sample means comparison: independent t-test and box plots t-tests are used to compare means of a continuous variables measured in two discrete groups. The groups may be experimental treatments or clear categories (species, biological sex). A t-test is considered independent (a.k.a., non-paired) if the individuals sampled are selected independently from the two groups, or as a paired t-test if pairs of individuals across groups have a structural reason that makes them non-independent (e.g., mates; from same location; paired experimental design). The organization of data differs between **two-sample unpaired (df1)** and **two-sample paired (df2)** approaches.

t.test() to compare means for unpaired data

The t.test() function tests for differences between the mean values of the dependent variable across two discrete groups (independent variable). If observations of individuals in the two groups are not structurally related to each other (i.e., they are not in natural pairs) a non-paired, independent t-test is appropriate. There are assumptions of normality of data. The Welch Two-Sample t-test does not assume equal variances across the groups. (see bartlett.test() below for test of homogeneity of variances) We will use **df1** as an example.

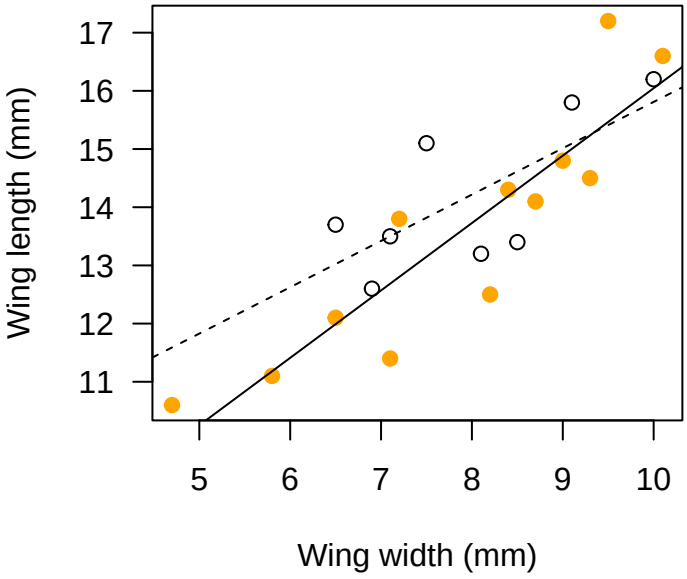


Figure 4: Figure 3. Relationship between the wing length and width for striped (orange points) and plain (open circles) morphs of the Santa Cruz brown moth. Striped: $length = 4.46 + 1.16 * width$, $F_{1,10}$=42.6, P 0.000066, R^2_{adj}=0.79; Plain: $length = 7.85 + 0.79 * width$, $F_{1,6}$=6.3, P 0.045, R^2_{adj}=0.43

```
aggregate(df1$mass_g, by=list(df1$treatment),FUN=mean) #group means
```

```
##   Group.1        x
## 1   light 21.70833
## 2   shade 18.66667
```

```
aggregate(df1$mass_g, by=list(df1$treatment),FUN=sd) #group stdevs
```

```
##   Group.1        x
## 1   light 2.287698
## 2   shade 2.433977
```

```
t.test(df1$mass_g~df1$treatment) #perform t-test
```

```
##
##  Welch Two Sample t-test
##
## data:  df1$mass_g by df1$treatment
## t = 3.1544, df = 21.916, p-value = 0.004615
## alternative hypothesis: true difference in means between group light and group shade is not equal to
## 95 percent confidence interval:
##  1.041448 5.041885
## sample estimates:
## mean in group light mean in group shade
##            21.70833            18.66667
```

Present statistics in the text:
California poppies grown in full sunlight were larger (mean 21.7 ± sd 2.8 g) than those grown in shade (18.7 ± 3.5 g) (t = 3.15, df = 21.9, p = 0.0046).

plot() to create boxplot

```
plot(df1$treatment, df1$mass_g, xlab='Treatment',ylab='Above-ground biomass (g)', las=1,col='lightblue'
points(c(1,2),c(21.7,18.7),pch=19) #optional superimpose mean
```

Call out to figure from the text:
California poppies grown in full sunlight were larger than those grown in shade (Figure 2; t = 3.15, df = 21.9, p = 0.0046).

t.test() to compare means for paired data

The t.test() function tests for differences between the mean values of the dependent variable across two discrete groups (independent variable). If observations of individuals in the two groups are structurally related to each other in natural pairs, a paired t-test is appropriate. There are assumptions of normality of data. The Welch Two-Sample t-test does not assume equal variances across the groups. We will use **df2** as an example.

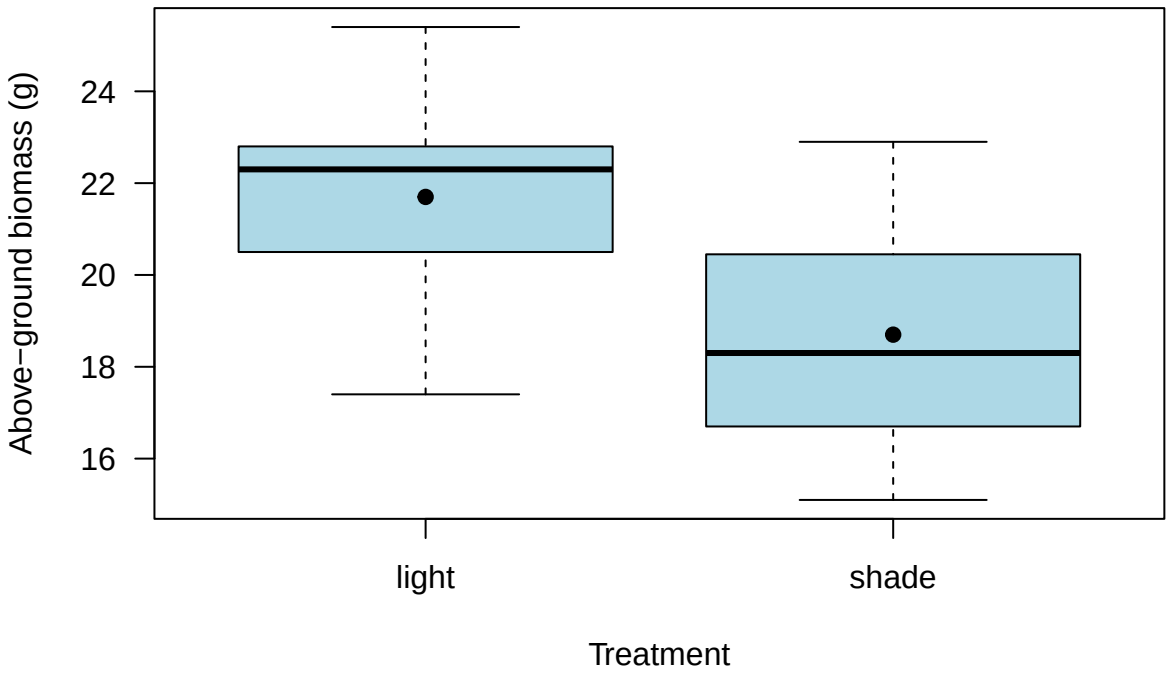


Figure 5: Figure 2. Above ground biomass of California poppies grown in full sunlight or shade. Solid circle indicates mean.

```
mean(df2$red_mL); mean(df2$blue_mL)  #group means
```

```
## [1] 37.04
```

```
## [1] 28.03
```

```
sd(df2$red_mL); sd(df2$blue_mL) #group stdevs
```

```
## [1] 3.929296
```

```
## [1] 3.599284
```

```
t.test(df2$red_mL,df2$blue_mL, paired=TRUE) #perform paired t-test
```

```
##
##  Paired t-test
##
## data:  df2$red_mL and df2$blue_mL
## t = 121.26, df = 19, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  8.85448 9.16552
## sample estimates:
## mean of the differences
##                    9.01
```

Present statistics in text:
Hummingbirds consumed much more sugar water when it was dyed red (mean 37.0 ± sd 3.9 mL h^{-2}) than blue (28.0 ± 3.6 mL h^{-2}) (t = 121.3, df = 19, p ≤ 0.00001).

plot() to create boxplot from unpaired data

```
boxplot(df2$red_mL, df2$blue_mL,
        xlab='Sucrose solution color',ylab='Liquid consumed (mL/h)',
        las=1,names=c('red','blue'),
        col=c('red','lightblue'))
points(c(1,2),c(37.04,28.03),pch=19) #optional superimpose mean
```

Call out to figure from the text:
Hummingbirds consumed more sucrose solution (20% w/v) when died red than blue (Figure 3) (paired t = 121.26, df = 19, p ≤ 0.00001).

Analysis of Variance: compare means of three or more groups

lm() and anova()

To compare the means of three or more groups, we use Analysis of Variance (ANOVA), which compares the differences in the means among groups to that expected given the variance within groups. The test statistic

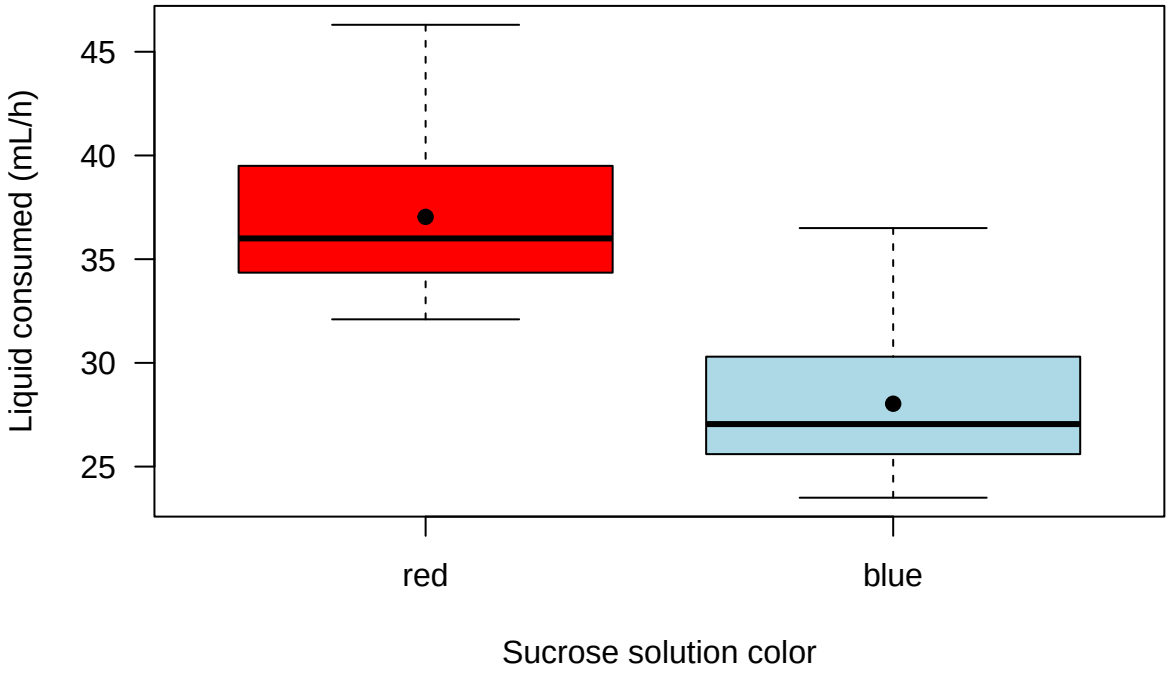


Figure 6: Figure 3. Consumption of red and blue sucrose solution (20%) by hummingbirds.

is an F-statistic, with two degrees of freedom, the first for the model (number of groups - 1) and the second related to the overall number of observations (# observations - # groups - 1). The significance of differences among any groups is evaluated with a P-value. If the P-value is below the alpha threshold ($P \leq 0.05$), we can compare each of the means to each other using the Tukey HSD (Honestly Significant Difference) post-hoc test.

There are several ways in R to do ANOVAs, including aov(), lm(), and glm(); each has slightly different approaches. The output of the models is best evaluated using extractor functions: summary(), anova(), and TukeyHSD().

Like for regression, the general model takes the form $y \sim x$; where y is the continuous dependent variable and x is a categorical value (like a treatment or characteristic).

Data frame **df4** includes plant biomass for each of three treatments: irrigation, fertigation, and control. We will use ANOVA to ask whether the treatments have an effect on plant biomass, and which of the treatments differ from the other. In particular, is there an effect of fertigation (applying human urine) that differs from the effect of irrigation with plain water?

```
# use aggregate to calculate basic summary statistics of the three treatments
mn4<-setNames(aggregate(df4$biomass_g,by=list(df4$treatment),FUN=mean),c("Biomass_g","mean"))
sd4<-setNames(aggregate(df4$biomass_g,by=list(df4$treatment),FUN=sd),c("Biomass_g","stdev"))
num4<-setNames(aggregate(df4$biomass_g,by=list(df4$treatment),FUN=length),c("Biomass_g","n"))
msn4<-merge(mn4,sd4); msn4<-merge(msn4,num4)
msn4 #show the resulting summary
```

```
##    Biomass_g     mean    stdev  n
## 1    control 5.251667 1.157331 12
## 2 fertigated 8.098333 1.293844 12
## 3  irrigated 7.916667 1.115024 12
```

```
#use aov() function to calculate the analysis of variance
aovout4<-aov(df4$biomass_g~df4$treatment) #analysis of variance
summary(aovout4) #extractor function
```

```
##               Df Sum Sq Mean Sq F value   Pr(>F)
## df4$treatment  2  60.95  30.477   21.48 1.06e-06 ***
## Residuals     33  46.82   1.419
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The ANOVA table provide the statistics needed to determine if *any* of the group means differ from each other.
F statistic = 21.48 Degrees of freedom = 2,33 P-value = 1.06e-06 = 0.00000106 Note the *** following the treatment row P value; according to the key that is statistically significant at the 0.001 level.

A shorthand way to present those statistics in text:
($F_{2,33} = 21.48$, $P \leq 0.000001$)

Because the overall model is statistically different (there are differences in the means, but we don't know *which* means differ) we can then do a Tukey's HSD post-hoc comparison.

```
#Post-hoc Tukey's Honestly Significant Difference comparison of means
TukeyHSD(aovout4) #post-hoc comparison of means
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = df4$biomass_g ~ df4$treatment)
##
## $`df4$treatment`
##                             diff       lwr      upr     p adj
## fertigated-control    2.8466667  1.653394 4.039940 0.0000044
## irrigated-control     2.6650000  1.471727 3.858273 0.0000131
## irrigated-fertigated -0.1816667 -1.374940 1.011606 0.9261321
```

This output looks a bit weird at first. It shows the difference in mean values for each *pair* of treatments (e.g., fertigated - control). If you look at the means from the aggregate table above, fertigated biomass mean = 8.098333 and control = 5.251667.
8.098333-5.251667 = 2.846666, the value of "diff" in the Tukey's table. The table also give lower and upper bounds for the HSD range: if the range includes zero, the difference between the two treatments is not considered significant. There is also a P-value (adjusted for multiple comparisons) given for the statistical difference between each pair. In this case, fertigated and irrigated are both highly statistically difference from control, but they are not different from each other.

We can present these results graphically using a box plot or with a table, and with accompanying text.

Here we can make a box plot and use blue points to overlay the means of each treatment (calculated above in mn4; we just need the values in the second column, so we use [,2]). We can use the text function to overlay letters that indicate which treatments are the same (have the same letter) and which differ. The text function takes the arguments text(x,y,labels). The center of the treatments are ordered as 1,2,3 on the x axis. The y placement is just above the 0.75 quantile, calculated using the quantiles function.

```
plot(df4$biomass_g~df4$treatment,
    xlab="Treatment", ylab="Biomass (g)", las=1)
points(c(1,2,3),mn4[,2],pch=19, col="blue") #treatment means
text(c(1.2,2.2,3.2),
     y=aggregate(df4$biomass_g,by=list(df4$treatment),FUN=quantile,prob=0.75)[,2],
     labels=c("A","B","B"),pos=3)
```

Describe that in text:
Radish growth appears to be limited by access to water, since both irrigation and fertigation (with the same amount of liquid applied) significantly increased growth compared to the control, which only received ambient water from rain and dew (Figure 4). The additional nutrients in fertigation (primarily nitrogen in the human urine) did not significant increase growth more than plain water, suggesting the plants are not nutrient limited (Figure 4).

Testing assumptions of normality and homogeneity of variance

bartlett.test() F test of equality of variance among groups

For t-tests and ANOVAS, there is an assumption that the variances in each of the groups are the same (assumption of homogeneity of variance). There are several possible tests for the (F-test of variance (for two groups), Bartlett's test (assumes normality), Fligner-Killeen test (non-parametric), and the Levene test). The Bartlett test can be used either with data as a formula or as separate groups in a list

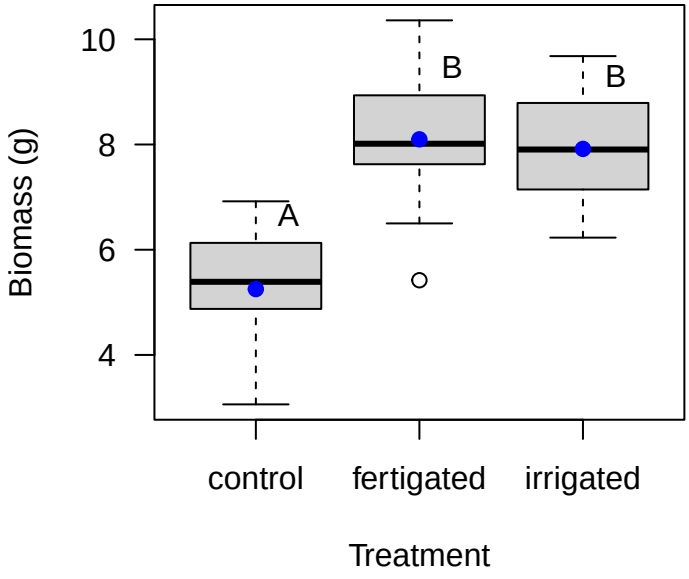


Figure 7: Figure 4. Above ground biomass of radishes growth with three treatments. Box plots indicate quartiles; blue circles are mean values. Treatments with the different letters are statistically different ($F_{2,33}$ = 21.48, P 0.000001, Tukey's HSD post-hoc comparison).

```
bartlett.test(df1$mass_g~df1$treatment) #formula version
```

```
##
##  Bartlett test of homogeneity of variances
##
## data:  df1$mass_g by df1$treatment
## Bartlett's K-squared = 0.040394, df = 1, p-value = 0.8407
```

```
bartlett.test(list(df2$red_mL,df2$blue_mL)) #separate groups version
```

```
##
##  Bartlett test of homogeneity of variances
##
## data:  list(df2$red_mL, df2$blue_mL)
## Bartlett's K-squared = 0.14229, df = 1, p-value = 0.706
```

For both comparisons, the p-value is » 0.05, so there is no evidence that the variances of the two groups are different (meets assumption of homogeneity of variance). ($P \leq 0.05$ would indicate the variances are unequal between groups)

shapiro.test() Shapiro-Wilk test of normality

Many parametric statistical tests assume the values in a single variable are drawn from a normal distribution. The Shapiro-Wilk test asks if the distribution of values in a variable differ from the normal distribution

```
shapiro.test(df3$length)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  df3$length
## W = 0.98365, p-value = 0.9722
```

From the output, the p-value > 0.05 implying that the distribution of the data are not significantly different from normal distribution. In other words, we can assume the normality. (p$ $0.05 would indicate the values are non-normal).

Contingency table analysis

chisq.test(observed_table) Chi-square (χ^2) test of independence

Let's use **df3** to ask if eye color (red or black) is independent of wing pattern (plain or striped) in Santa Cruz moths. First we use the table() function to calculate the two-way contingency table of observed states. Then the chisq.test() function takes that table as its argument.

```
eye_wings<-table(df3$eye_color,df3$wing_stripe) #make contingency table
eye_wings #show the table
```

```
##
##         plain striped
##   black     4       9
##   red       4       3
```

```
chisq.test(eye_wings) #calculate the chi sq
```

```
## Warning in chisq.test(eye_wings): Chi-squared approximation may be incorrect
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  eye_wings
## X-squared = 0.44872, df = 1, p-value = 0.5029
```

Describe the results in text:
Eye color and wing pattern are independent traits in Santa Cruz moths ($\chi^2 = 0.45$, df=1, P $\leq$ 0.50).
Note: small sample size influences the reliability of the statistic

#Still to come
#logistic?
#barplot()
#apply()
#for-loops