Basic Biostatistics in R for the NON-Computer Programmer

A teaching pamphlet by
Holly Harlin
Epidemiology MPH
Loma Linda University

All analyses performed in R Studio
Version 0.98.1062 – © 2009-2013 RStudio, Inc.
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_5) AppleWebKit/537.78.2 (KHTML, like Gecko)

Table of Contents

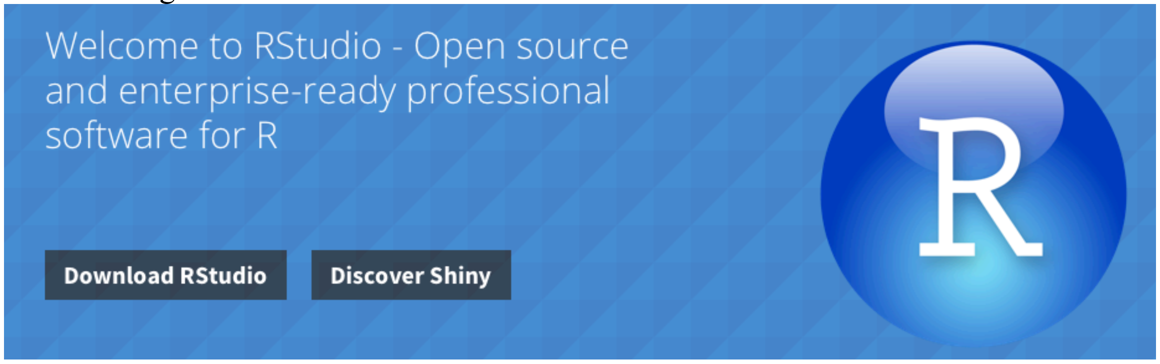
How to Download R and R-Studio

How to Download R

1. Go to: http://www.r-project.org
2. In the 'Getting Started' section, click **download R**
3. Scroll down to the USA options and click on the first option http://cran.cnr.Berkeley.edu
4. Click the **Download R for**… (your respective operating system)
    a. For Mac operating systems click on the R program for your respective computer version and installation will begin.
    b. For Windows operating systems click on the install **R for the first time** link. This will then take you to another page in which you click on **Download R 3.1.2 for Windows** and the download will begin.

How to Download R Studio

1. Go to: http://www.rstudio.com
2. Click on '**Download RStudio**' on the first picture screen, if you wait too long, you will have to scroll back to this image.


Welcome to RStudio - Open source and enterprise-ready professional software for R
Download RStudio
Discover Shiny
R

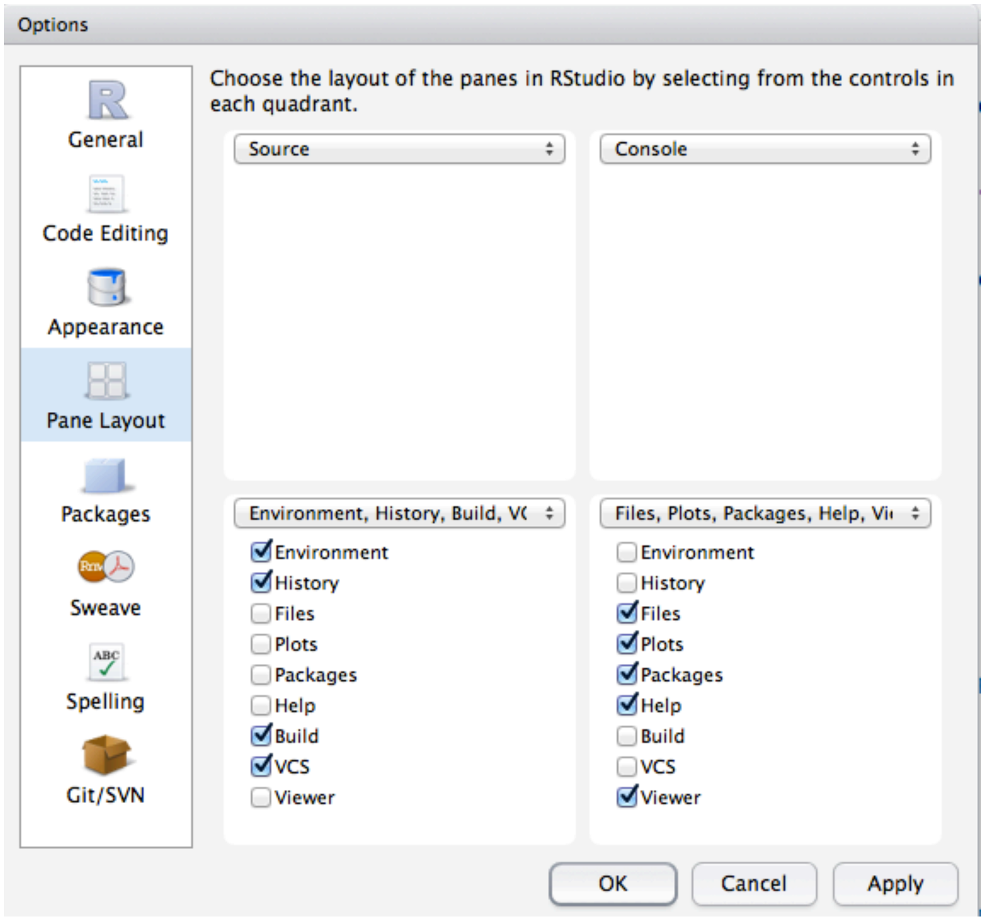
3. Click on '**R Studio Desktop >**'
4. Scroll down and click on **DOWNLOAD RSTUDIO DESKTOP**
    a. This should say the price is FREE above
5. Then select the R Studio for your respective operating system, and the download will start right away.

For Mac Computers:

- Once the download is complete a pop up box will appear containing an Applications folder logo and the new R Studio file logo.
- Click on and drag the R Studio file logo into your Applications folder logo.

Open R Studio:

1. First, set the R studio viewing preferences:
    a. In the upper-left corner of the screen click on the **R Studio** tab
    b. Select **Preferences**
    c. This window will appear (as seen on next page)



Choose the layout of the panes in RStudio by selecting from the controls in each quadrant.

Source	Console
Environment, History, Build, VC	Files, Plots, Packages, Help, Vi
[✓] Environment	[ ] Environment
[✓] History	[ ] History
[ ] Files	[✓] Files
[ ] Plots	[✓] Plots
[ ] Packages	[✓] Packages
[ ] Help	[✓] Help
[✓] Build	[ ] Build
[✓] VCS	[ ] VCS
[ ] Viewer	[✓] Viewer

i. Select **Pane Layout** on the left side
   1. Set the upper left to **Source**
   2. Set the upper right to **Console**
   3. Set the bottom left to **Environment, History, Build…**
   4. Set the bottom right to **Files, Plots, Packages, Help…**

ii. Click **Apply**

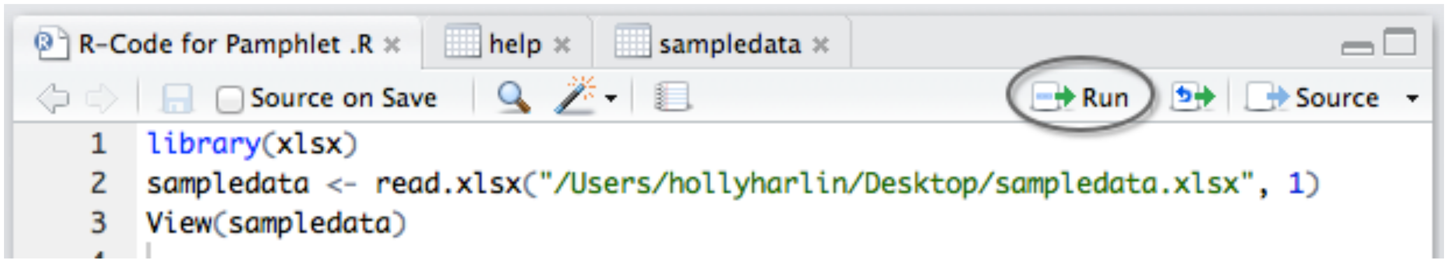
iii. Then click **OK** and the box will disappear.

While code in R and R-Studio are the same, R-Studio is a more user-friendly interface of R, and is what will be used in this tutorial. All examples in this pamphlet will also be based on the formatting scheme presented above and through R-Studio for Mac version 0.98.1062. While the majority of code stays the same between the updated versions of R, there can be changes in packages that are accepted between the versions. It is important to remember your particular version of R in the rare event that such a discrepancy arises.

Basic Commands in R

Where to write your code:

1. Code will be written the Source box in the top-left corner of your R Studio pane.
2. If there is not a new or existing source editor, or to add another, go to:
   a. **File**
   b. Then click on: **New File**
   c. From there click: **R Script**
   d. This will bring up an untitled Script page in your source window to write your code.
3. All coding lines in R will be numbered on the left side
4. An example of the source window with R code can be seen below:

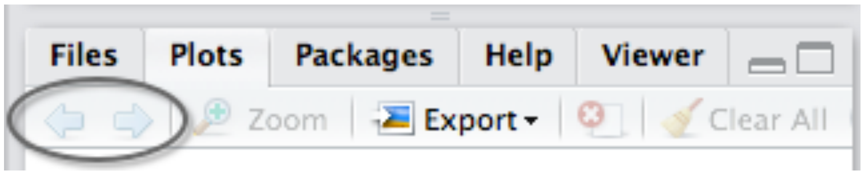


```
library(xlsx)
sampledata <- read.xlsx("/Users/hollyharlin/Desktop/sampledata.xlsx", 1)
View(sampledata)
```

How to 'Run' your data:

1. After writing a line or section of code, it must be '**RUN**' to produce the desired analytical results
2. If you are only running one line, you can simply click on that line of code with your cursor and click on the '**Run**' command, circled above.
3. If running multiple lines of data at once, highlight completely all lines desired to be run, and click the '**Run**' command.

Where to find the Results?

1. Most output will print in the '**Console**' section of R Studio, in the upper right hand corner.
2. Graphs and Plots will appear in the bottom right hand corner, under the '**Plots**' tab.
   a. If multiple plots have been outputted, you can scroll through the plots with the Right and Left arrows circled below:
   b.


3. Inputted data sets and created variables will be stored in the '**Environment**' section (bottom left). This section is not used often.

Important Note: Phrases preceded by '#' are comments and explanations and do not affect the code:

Example:

```
# Read in Data
>sampledata<-read.xlsx("/Users/anonymous/Desktop/sampledata.xlsx", 1)
# The only line actually read for analysis here is the '>sampledata…'above.
# These comments will be denoted in R with GREEN text
# Comments can be helpful for organizing your code and will be used throughout
this text to guide you through coding
```

Case Sensitivity: It is VERY IMPORTANT to recognize that R is case sensitive, CAPITAL and lowercase Letters Must Be Consistent ThroughOut. The variable 'ID' is a different variable from 'Id', which is also different from 'id'.

Inputting Data Into R Studio

How to Input data from an EXCEL file:

To input data from Excel, you must first download the .xlsx package in R.

To do this:

1. Click on '**Tools**' in your top menu bar
2. Click '**Install Packages…**'
3. The default 'Install From' should be "**Repository (CRAN)**" if it is not, switch it to this from the menu
4. Type "**xlsx**" into the '**Packages**' section, it should appear while you type
5. Click "Install"
6. Installation details should appear in the "**console**" window

Or a simpler method is to type and run in your source window:

```
install.packages("xlsx") # Respective package name goes between quotes
```

Once the package is installed:
Type in the 'Source' section, under the statement above:

```
library(xlsx)

Loading required package: rJava
Loading required package: xlsxjars
```

When you run this, these two lines in Red will appear in the output.

**** If a message about 'R Java SE 6' appears when you attempt to run '`library(xlsx)`' or any other downloaded package, install this update from the pop-up window. Once installed, click on the window that was behind this message that says to start your R session again. R will automatically refresh. Run the '`library(xlsx)`' command again. ****

Represents data location
Represents data file name

```
sampledata <- read.xlsx("/Users/anonymous/Desktop/sampledata.xlsx", 1)

View(sampledata)
```

The view command then outputs your dataset into another tab on the Source window and further information about your dataset will appear in the Environment tab.

The '<-' is the assignment operator in R, similar to an '=' sign. Here, we assigned the name 'sampledata' to the data we imported from Excel. We can now simply refer to it as 'sampledata' as was done in the view statement.

How to Input data from a TEXT file:

```
sampledata2 <- read.table("/Users/anonymous/Desktop/sampledata.txt", header=T,
sep=",")

View(sampledata2)
```

Denotes that we have comma separated data

Sample of what comma separated data looks like in the .txt file:

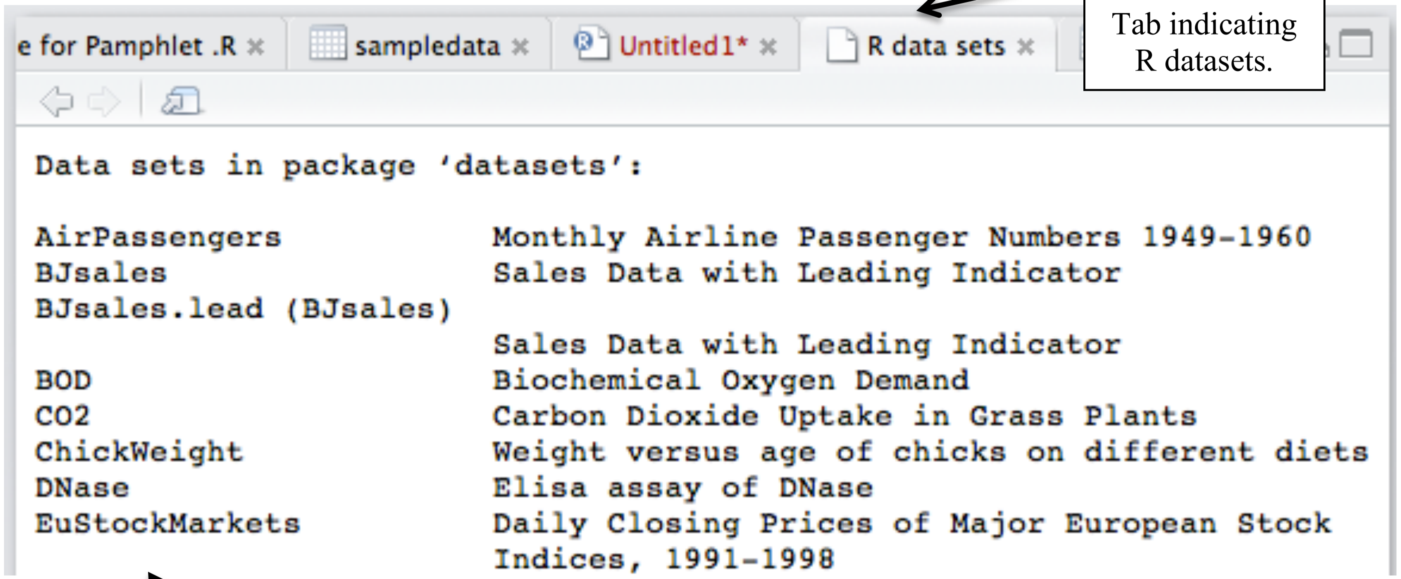
```
ID,Sex,Age
1,1,34
2,1,33
3,2,40
4,1,28
5,2,37
```

Using R Sample Datasets:

In this pamphlet we will be practicing on datasets supplied by the R program. The available R datasets can be viewed by typing:

```
data()
```

into your source code window. Run the line. The list of R datasets will appear in another source window (similar to the 'View(sampledata)' command from above).



Tab indicating R datasets.

```
Data sets in package 'datasets':

AirPassengers             Monthly Airline Passenger Numbers 1949-1960
BJsales                   Sales Data with Leading Indicator
BJsales.lead (BJsales)
                          Sales Data with Leading Indicator
BOD                       Biochemical Oxygen Demand
CO2                       Carbon Dioxide Uptake in Grass Plants
ChickWeight               Weight versus age of chicks on different diets
DNase                     Elisa assay of DNase
EuStockMarkets            Daily Closing Prices of Major European Stock
                          Indices, 1991-1998
```

List of dataset names

To view the content of this data, simply type the dataset name into your main code writing source window and run the line. No other commands are necessary.
Example:

```
ToothGrowth
```

The dataset 'ToothGrowth' will then appear like output in your console window. If you desire to see to see that dataset in its regular form and use that dataset for analyses, type and Run:

```
View(ToothGrowth)
```

in your source window and it will appear as a dataset in another tab in your source window, as well.

How to Organize Your Data

In R, after reading in a dataset each variable name will be:

```
DatasetName$VariableName
```

To rename these variables into a simpler and shorter form:

```
NewVariableName <- DatasetName$VariableName
```

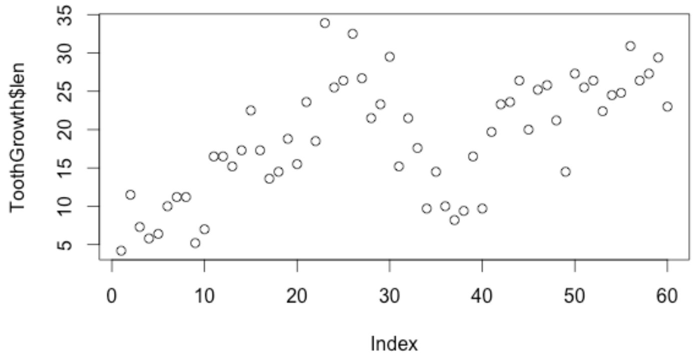
This new variable name can be the same as the original name minus the attached dataset name.

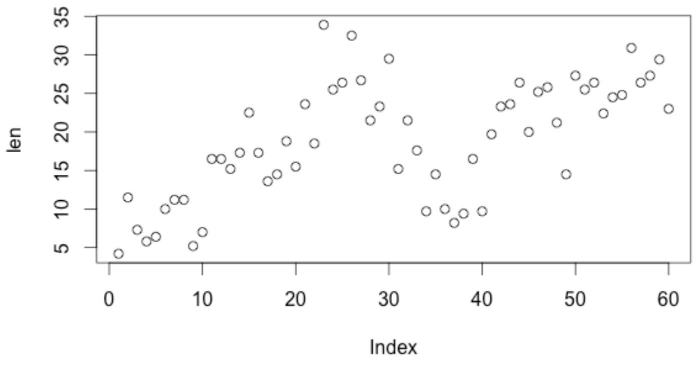
The '<-' command can be used to give variables different names or different values. This symbol is used throughout R coding.

For example:

```
#Plot variable len
plot(ToothGrowth$len)

# Or rename the variable into a simpler, single level name
len <- ToothGrowth$len
plot(len)
```

Representation of the two output graphs:



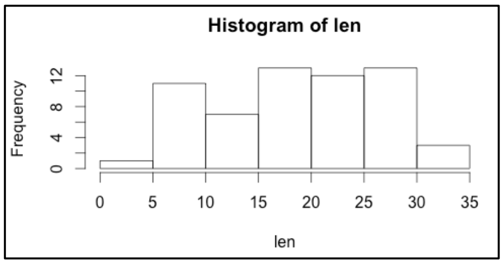


How to Perform Basic Statistical Tests in R

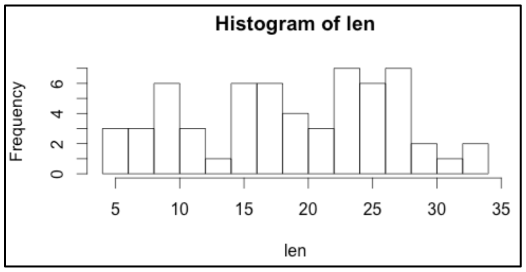
Graphical Representations

Prior to any data analysis, it is important look at a graphical representation of the data via histograms and scatterplots. Please Note: if your window for viewing plots in R Studio is too small you will get an error when you run the code that states your figure margins are too large. If this occurs, simply expand the plot window and re-run the command. The plot should then appear.

```
# simple histogram of tooth length
hist(len)
```



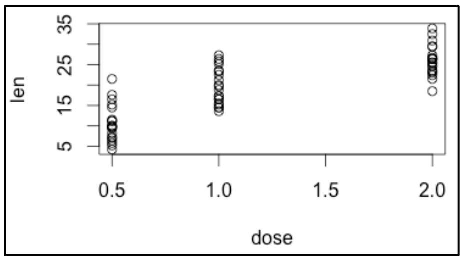
```
# simple histogram with specification of number of bars, 20
hist(len, breaks=20)
```



This helps reveal if the data is normally distributed, which is a requirement in many statistical analyses.

Prior to any data analysis, look at the graphical relationship between variables via scatter plots.

```
# The X-axis is listed first, followed by the Y-axis
# Basic Plot
plot(dose,len)
```



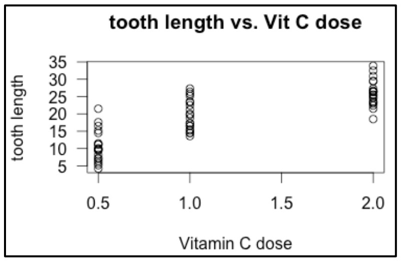
```
# To make the Plot more informative and visually appealing
plot(dose,len, xlab="Vitamin C dose", ylab="tooth length", main="tooth length vs.
Vit C dose", las=1)
```

Converts the Y-axis numbers to a horizontal rather than vertical view.

Assigns labels to X and Y

Main Graph Title

Output



Equality of Variance

```
#Equality of variance test
var.test(len~supp, data=ToothGrowth)
```

Output

```
      F test to compare two variances

data:  len by supp
F = 0.6386, num df = 29, denom df = 29, p-value = 0.2331
alternative hypothesis: true ratio of variances is not equal
to 1
95 percent confidence interval:
 0.3039488 1.3416857
sample estimates:
ratio of variances
         0.6385951
```

The p-value greater than 0.05 (p=0.2331) reveals that the variances between the two groups are equal.

Correlation Among Variables

Pearson's Correlation Coefficients

```
# simple correlation matrix, no missing variables
# Taken from the 'women' dataset
cor(women$weight, women$height)
```

Output

```
[1] 0.9954948
```

```
# simple correlation matrix, accounting for missing values
cor(women$weight, women$height, use="complete.obs")
```

Output

```
[1] 0.9954948  (Same due to no missing observations)
```

```
# correlation significance testing
cor.test(women$weight, women$height)
```

Output

```
      Pearson's product-moment correlation

data:  women$weight and women$height
t = 37.8553, df = 13, p-value = 1.088e-14
alternative hypothesis: true correlation is not
equal to 0
95 percent confidence interval:
 0.9860970 0.9985447
sample estimates:
      cor
0.9954948
```

Statistics Note: Pearson's correlations are to be used when the data is normally distributed and without influential outliers. However, if you are faced with these, the nonparametric, Spearman's correlation, is the preferred test, as it makes no assumption about the required shape of the data.

Spearman's Correlation Coefficients

```
cor.test(women$weight, women$height, method="spearman")
```

Output

```
Spearman's rank correlation rho

data:  women$weight and women$height
S = 0, p-value < 2.2e-16
alternative hypothesis: true rho is not equal to 0
sample estimates:
rho
  1
```

Multicolinearity and Centralization:
Multicolinearity is a term used for high correlations between variables. When correlations are above 0.7 you often want to try to figure out why these variables are so correlated. If you have a squared or product term in the model you are guaranteeing multicolinearity (such as X and X^2). In order to combat the high correlations in such a scenario we can use centralization, which is subtracting the mean of X from X.

Research Questions About One Group: Before/After Analysis

Comparing Before and After Means

```
# Paired t-test
# Groups 1 and 2 are the same subjects (indicated by having the same ID)
# Running a paired t-test takes into account the correlation that exists between
subjects with repeated measures.
View(sleep)
t.test(sleep$extra~sleep$group,paired=TRUE)
```

Output

```
     Paired t-test

data:  sleep$extra by sleep$group
t = -4.0621, df = 9, p-value = 0.002833
alternative hypothesis: true difference in means
is not equal to 0
95 percent confidence interval:
 -2.4598858 -0.7001142
sample estimates:
mean of the differences
                  -1.58
```

We can see here that the mean difference between groups is -1.58 hours of sleep, this difference is statistically significant (p-value = 0.002833).

Research Questions About 2 Groups

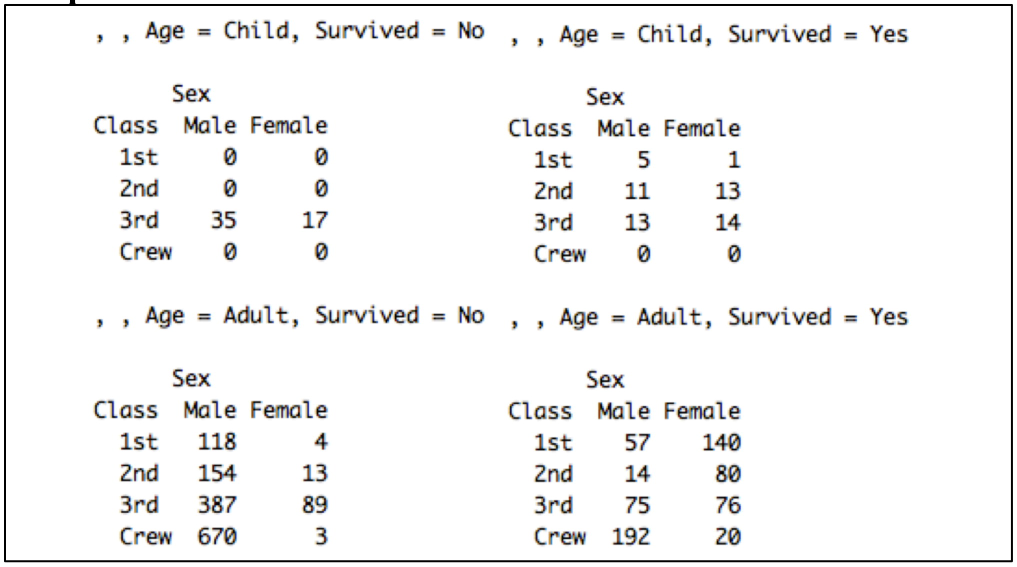
Chi-Square Test for Comparing Two Proportions

To perform a binary analysis using an R dataset we need to add an extra step due to the fact that none of the R datasets are in the correct form. When data is categorical with high overlapping symmetry it might be placed in an aggregate form to make it smaller, as R has done with the Titanic dataset. This is denoted with the Freq variable, which is a count for how many people in the dataset fit those specific criteria. Adding all of the counts in Freq will tell you the actual size of the dataset.

To initiate this analysis we use the xtabs command, which will create a 2x2 table of the variables we desire to compare, adjusting for the frequencies that these variables occur in the dataset. To do this, we need to install the prettyR package.

```
View(Titanic)
# Install the prettyR package, which has the xtabs command
install.packages(“prettyR")
library(prettyR)
xtabs(Freq~Class + Sex + Age + Survived, data=Titanic)
# The order the variables are listed is important
# Class=Rows, Sex=Columns, Age and Survived are broken up into different tables
# If you are familiar with SAS this order is reversed, in R the last variable(s)
listed are the stratum specific table variables; in SAS it is the first.
```

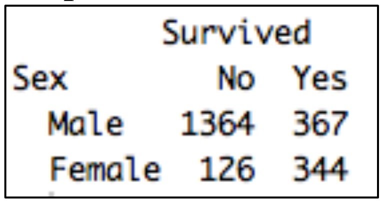
Output



```
, , Age = Child, Survived = No     , , Age = Child, Survived = Yes

      Sex                                Sex
Class  Male Female                 Class  Male Female
  1st     0      0                   1st     5      1
  2nd     0      0                   2nd    11     13
  3rd    35     17                   3rd    13     14
  Crew    0      0                   Crew    0      0

, , Age = Adult, Survived = No     , , Age = Adult, Survived = Yes

      Sex                                Sex
Class  Male Female                 Class  Male Female
  1st   118      4                   1st    57    140
  2nd   154     13                   2nd    14     80
  3rd   387     89                   3rd    75     76
  Crew  670      3                   Crew  192     20
```

```
# However, in our simple analysis we only want to look at survival vs. sex,
ignoring all other factors
xtabs(Freq~Sex + Survived, data=Titanic)
# This command put the data into a 2x2 table with counts of survival (columns) by
sex (rows)
```

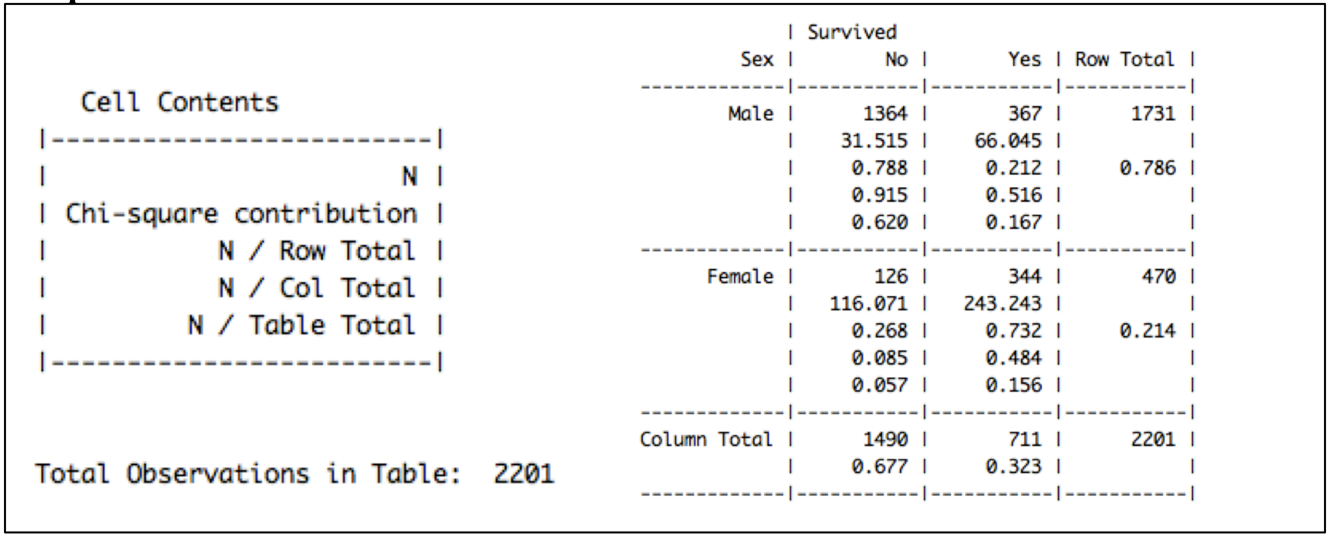
Output



```
        Survived
Sex        No  Yes
  Male   1364  367
  Female  126  344
```

```
# If the data were already in a normal form to create the 2x2 table we would use:
# Example
survivaltable <- table(Titanic$Sex, Titanic$Survived) #Sex=rows, Survived=columns
survivaltable  # to print the table

# The CrossTable command provides a more succinct, pintable table
# To use this command install the gmodels package
# You can enter the entire xtabs command directly into a CrossTable command
install.packages(“gmodels")
library(gmodels)
CrossTable(xtabs(Freq~Sex + Survived, data=Titanic))

# OR: name the xtabs command and insert the name into the CrossTable
titanictable <- xtabs(Freq~Sex + Survived, data=Titanic)
library(gmodels)
CrossTable(titanictable)
# Naming the table will make coding simpler when you begin your data analysis
```

Output



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


Total Observations in Table:  2201


             | Survived
         Sex |        No |       Yes | Row Total |
-------------|-----------|-----------|-----------|
        Male |      1364 |       367 |      1731 |
             |    31.515 |    66.045 |           |
             |     0.788 |     0.212 |     0.786 |
             |     0.915 |     0.516 |           |
             |     0.620 |     0.167 |           |
-------------|-----------|-----------|-----------|
      Female |       126 |       344 |       470 |
             |   116.071 |   243.243 |           |
             |     0.268 |     0.732 |     0.214 |
             |     0.085 |     0.484 |           |
             |     0.057 |     0.156 |           |
-------------|-----------|-----------|-----------|
Column Total |      1490 |       711 |      2201 |
             |     0.677 |     0.323 |           |
-------------|-----------|-----------|-----------|
```

Now we want to take this table and Run a Chi-square significance test.

```
# Chi Square Test
chisq.test(titanictable)
```

Output

```
      Pearson's Chi-squared test with Yates' continuity correction

data:  titanictable
X-squared = 454.4998, df = 1, p-value < 2.2e-16
```

```
# For Testing Significance of a 3-dimenstional array use mantel haenszel
withage <- xtabs(Freq~Sex + Survived + Age, data=Titanic)
withage
mantelhaen.test(withage)
```

Output

```
      Mantel-Haenszel chi-squared test with continuity correction

data:  withage
Mantel-Haenszel X-squared = 440.5677, df = 1, p-value < 2.2e-16
alternative hypothesis: true common odds ratio is not equal to 1
95 percent confidence interval:
  7.487267 11.992767
sample estimates:
common odds ratio
          9.47592
```

Comparing two group means

Many statistical analyses can be easily computed by either comparing two different column variables or by comparing a single column that is stratified by another column variable, such as weight values stratified by gender. The subjects in these groups are independent of each other. This first example shows how to compare the average weights of two groups that are their own unique column variables.

```
# Independent 2-group t-test
View(beaver1)
View(beaver2)
t.test(beaver1,beaver2, var.equal=TRUE)
```

Indicates that the variances between the two groups are equal. If unequal: var.equal=FALSE

Output

```
      Two Sample t-test

data:  beaver1 and beaver2
t = 1e-04, df = 212, p-value = 0.9999
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.09084928  0.09085525
sample estimates:
   mean of x    mean of y
2.982456e-06 1.563337e-15
```

Comparing Two Group Means by Stratifying on Another Variable

The second way of performing an independent t-test is by stratifying on a variable, as shown below. The method chosen is dependent upon the organization of data.

```
# Independent 2-group t-test
# Testing the difference between tooth lengths stratified by the two different
mediums of Vitamin C intake (supp)
t.test(len~supp)
```

Output

```
	Welch Two Sample t-test

data:  len by supp
t = 1.9153, df = 55.309, p-value = 0.06063
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.1710156  7.5710156
sample estimates:
mean in group OJ mean in group VC 
        20.66333         16.96333 
```

Comparing a Study Mean Against a Standard Mean

```
# one sample t-test
# Test if the average number of murder
arrests in this dataset differs from 10

View(USArrests)
t.test(USArrests$Murder, mu=10,
data=USArrests)

# Analysis Reveals the average number
of arrests for murder in the US (7.788)
is significantly different than 10.
```

Output

```
	One Sample t-test

data:  USArrests$Murder
t = -3.5911, df = 49, p-value =
0.0007609
alternative hypothesis: true mean is
not equal to 10
95 percent confidence interval:
 6.550178 9.025822
sample estimates:
mean of x 
    7.788 
```

Additional Specifications

```
# One Tailed - Less Than
t.test(len~supp, alternative='less')

# One Tailed - Greater Than
t.test(len~supp, alternative='greater')

# Confidence Interval other than 95%
# Specifying a 99% Confidence Interval
t.test(len~supp, conf.level=.99)
```

alternative='less' & alternative='greater' can be specified for a one tailed test. If not specified then R will perform a two-tailed test.

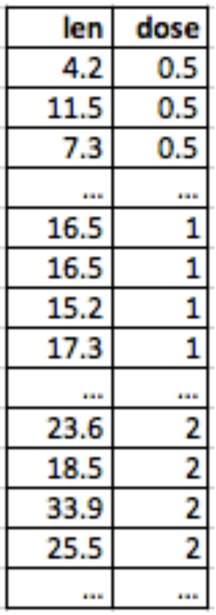
The default CI is 95%; this command gives the ability to test different alpha values.

Output

```
	Welch Two Sample t-test

data:  len by supp
t = 1.9153, df = 55.309, p-value = 0.06063
alternative hypothesis: true difference in means is not equal to 0
99 percent confidence interval:
 -1.453546  8.853546
sample estimates:
mean in group OJ mean in group VC 
        20.66333         16.96333 
```

One Way ANOVA

Comparison of 2 or More Group Means



len	dose
4.2	0.5
11.5	0.5
7.3	0.5
...	...
16.5	1
16.5	1
15.2	1
17.3	1
...	...
23.6	2
18.5	2
33.9	2
25.5	2
...	...

```
# Check data format: for ANOVA comparisons, all groups must be in
the same column, differentiated by another column variable (in
this case dose), which categorizes them.

# One Way Anova
# In this analysis we are testing to see if there is a difference
in the average tooth length between each of the three different
doses of vitamin C (0.5, 1, and 2).
toothgrowth.anova <- aov(len ~ factor(dose), data=ToothGrowth)
summary (toothgrowth.anova)
```

Output

```
             Df Sum Sq Mean Sq F value   Pr(>F)    
factor(dose)  2   2426    1213   67.42 9.53e-16 ***
Residuals    57   1026      18                     
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Findings

Based on this output, we can see that there is a significant difference between at least one of the three Vitamin C doses and tooth length. However, this overall F Test does not tell us which group means are different, only that there is something significant within your model. To test which group weights are actually different we must continue our analysis by performing a Tukey Post Hoc Analysis, which tests for significant differences between all group combinations.

Subgroup Analysis

```
# Tukey significance test for multiple comparisons
TukeyHSD(toothgrowth.anova)
```

Output

```
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = len ~ factor(dose), data = ToothGrowth)

$`factor(dose)`
        diff       lwr       upr    p adj
1-0.5  9.130  5.901805 12.358195 0.00e+00
2-0.5 15.495 12.266805 18.723195 0.00e+00
2-1    6.365  3.136805  9.593195 4.25e-05
```

One-Way ANOVA Conclusion

Based on the final Tukey comparison of the three group means, it is evident that all three group means are significantly different from each other. These group mean differences are listed under the ‘diff’ column in the output, followed by the 95% confidence limits and the adjusted p-values.

Regression Analysis

Regression analysis is a method of analyzing the best relationship between variables and using them in an equation to predict outcomes. In basic regression analysis you have one or more covariates (independent variables) and one outcome variable (dependent variable). In the example below we will use linear regression to synthesize the relationship between the two covariates: vitamin C dose and the type of vitamin C supplementation, to predict our outcome variable, tooth length.

```
# Multiple Linear Regression Using ToothGrowth data
toothlength <- lm(len ~ dose + supp , data=ToothGrowth)
summary(toothlength) # show results
```

Output

```
Call:
lm(formula = len ~ dose + supp, data = ToothGrowth)

Residuals:
   Min     1Q Median     3Q    Max
-6.600 -3.700  0.373  2.116  8.800

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)          9.2725     1.2824   7.231 1.31e-09 ***
dose                 9.7636     0.8768  11.135 6.31e-16 ***
suppVC              -3.7000     1.0936  -3.383   0.0013 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.236 on 57 degrees of freedom
Multiple R-squared:  0.7038,    Adjusted R-squared:  0.6934
F-statistic: 67.72 on 2 and 57 DF,  p-value: 8.716e-16
```

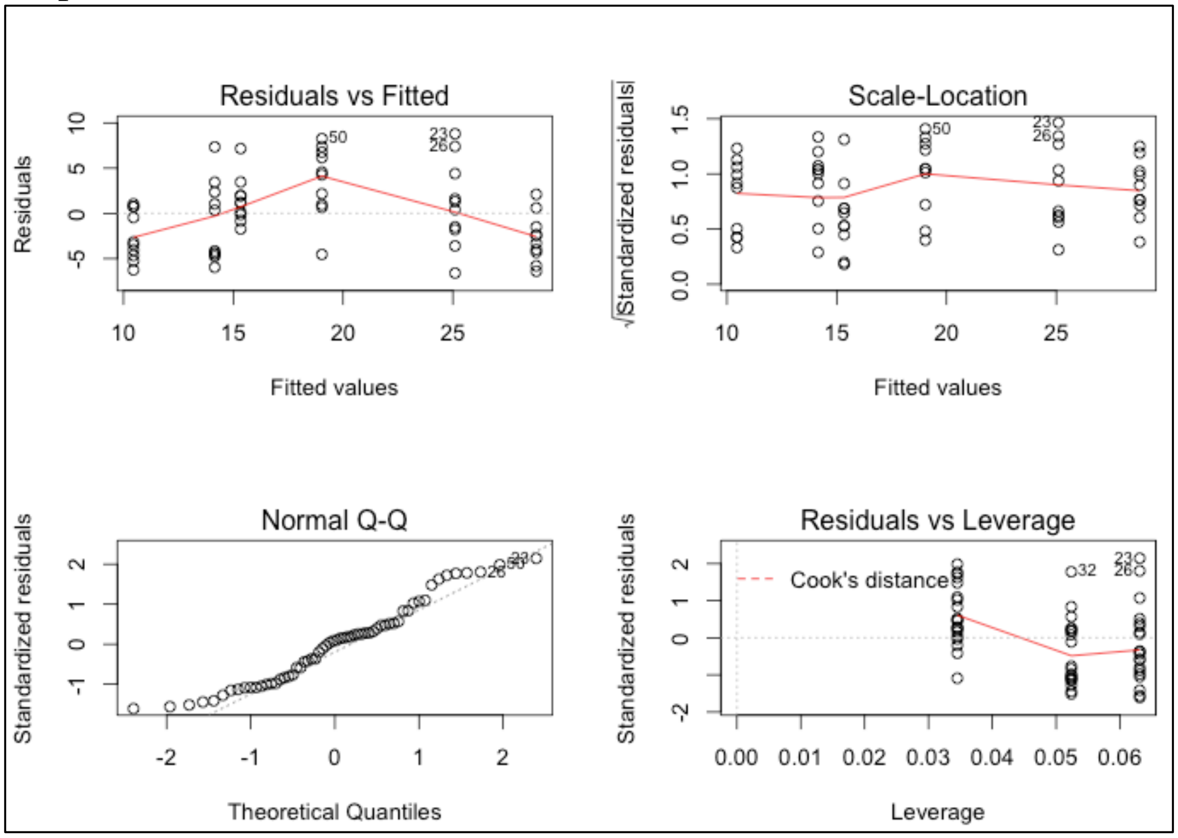
```
# Other useful functions
confint(toothlength, level=0.95) # CIs for model parameters
```

Output

```
                        2.5 %    97.5 %
(Intercept)          6.704608 11.840392
dose                 8.007741 11.519402
suppVC              -5.889905 -1.510095
```

```
fitted(toothlength) # predicted len values
residuals(toothlength) # residuals

# diagnostic plots to check regression assumptions
layout(matrix(c(1,2,3,4),2,2)) # This prints out 4 diagnostic graphs at once
plot(toothlength)
```

Output



Plot Interpretations

- **Residual vs. Fitted:** This plot should be randomly distributed around zero with no pattern or direction. In this plot we do not see the residuals evenly and randomly dispersed around zero, but running low on both ends and high in the center. This lack of randomness suggests that the linearity assumption is not being met. Large deviations in the width of the dispersion of the data across the x-axis are also indicative of unequal variance.
- **Scale-Location:** This chart is similar to the residual vs. fitted plot, showing the distribution of the residuals. This chart helps distinguish equality of variance and linearity. The standardized residuals should also be evenly and randomly distributed around zero.
- **Normal Q-Q:** This is a test for normality of data. You want the points on the graph to fit as closely to the straight diagonal line as possible. In this example we can see that our data is not perfectly normal, with a fair amount of deviation from the fitted line. Transforming this variable might improve normality and the fit of this graph.
- **Residuals vs. Leverage:** This plot shows how much influence each value has on the beta coefficients. High leverage values should be evaluated.

Logistic Regression

Logistic Regression is ideal for binary outcomes. When you exponentiate the beta coefficients in logistic regression you get the odds ratio.

```
# Logistic Regression
View(infert)
# Create the equation
baby <- glm(case~age+induced+parity+spontaneous, data=infert, family=binomial())
# Output equation summary statistics
summary(baby)
```

Output

```
Call:
glm(formula = case ~ age + induced + parity + spontaneous, family =
binomial(),
    data = infert)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.6281  -0.8055  -0.5299   0.8669   2.6141

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -2.85239    1.00428  -2.840  0.00451 **
age          0.05318    0.03014   1.764  0.07767 .
induced      1.18966    0.28987   4.104 4.06e-05 ***
parity      -0.70883    0.18091  -3.918 8.92e-05 ***
spontaneous  1.92534    0.29863   6.447 1.14e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 316.17  on 247  degrees of freedom
Residual deviance: 260.94  on 243  degrees of freedom
AIC: 270.94

Number of Fisher Scoring iterations: 4
```

```
# Results do not include confidence intervals, to view confidence intervals:
confint(baby)
```

Output

```
                  2.5 %     97.5 %
(Intercept) -4.87438890 -0.9222185
age         -0.00542269  0.1132530
induced      0.63613264  1.7774032
parity      -1.08149001 -0.3692163
spontaneous  1.36636751  2.5412467
```

```
# Exponentiate the beta coefficients to calculate the odds ratio
exp(coef(baby))
```

Output

```
(Intercept)         age     induced      parity spontaneous 
 0.05770622  1.05462050  3.28595133  0.49221973  6.85746772 
```

```
# Confidence intervals surrounding the odds ratios
exp(confint(baby))
```

Output

```
                  2.5 %     97.5 %
(Intercept) 0.007639761  0.3976359
age         0.994591986  1.1199153
induced     1.889160664  5.9144775
parity      0.339089903  0.6912759
spontaneous 3.921081497 12.6954881
```

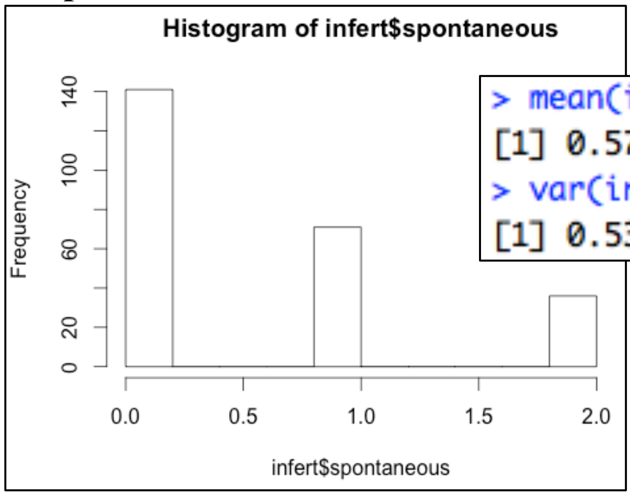
Poisson Regression

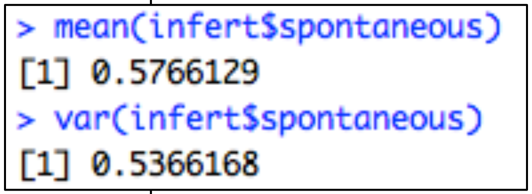
Poisson Regression is useful for count data, such as number of hospital visits. Poisson regression should always begin by creating a histogram of your outcome variable and running summary statistics such as the mean, median, and variance. These summary statistics must show a mode of 0, with a mean close to 1, preferably no more than 5, and a variance that is less than the mean. If the mean of the counting variable is 10 or greater the graph has a relatively normal curve and regular regression analysis is a better technique to analyze the relationship between the outcome and predictors.

We will use the same infertility (infert) dataset in this Poisson analysis. In logistic regression we analyzed the binary outcome of infertility (yes/no) against age, parity and the number of induced and spontaneous abortions. Here we will answer a different question. Our outcome will be number of spontaneous abortions experienced in relation to a woman's age, history of parity, and induced abortions. This allows us to be able to predict the rate of spontaneous abortion in relation to these three factors.

```
# Analysis of Poisson Assumptions
hist(infert$spontaneous)
mean(infert$spontaneous)
var(infert$spontaneous)
```

Output





```
> mean(infert$spontaneous)
[1] 0.5766129
> var(infert$spontaneous)
[1] 0.5366168
```

These statistics reveal that Poisson regression does appear to be the correct analysis. The mode is 0, the mean is below 1, and the variance is lower than the mean. If the variance had exceeded the mean a scaling factor would need to be added to the analysis to correct the problem.

```
# Poisson Regression
preg <- glm(spontaneous ~ age+parity+induced, data=infert, family=poisson())
summary(preg)
```

Output

```
Call:
glm(formula = spontaneous ~ age + parity + induced, family =
poisson(),
    data = infert)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.6200  -0.8591  -0.5795   0.5582   1.8141

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.50373    0.53168   0.947  0.34342
age         -0.05509    0.01761  -3.129  0.00176 **
parity       0.47356    0.06017   7.871 3.53e-15 ***
induced     -0.98425    0.15237  -6.460 1.05e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 257.28  on 247  degrees of freedom
Residual deviance: 183.40  on 244  degrees of freedom
AIC: 427.49

Number of Fisher Scoring iterations: 5
```

Like Logistic regression, you also exponentiate the beta coefficients in Poisson regression. These exponentiated beta coefficients then represent the incidence rate ratio.

```
# Rate interpretation of Poisson Regression
exp(coef(preg))
```

Output

```
(Intercept)         age      parity     induced
  1.6548869   0.9464009   1.6057038   0.3737176
```

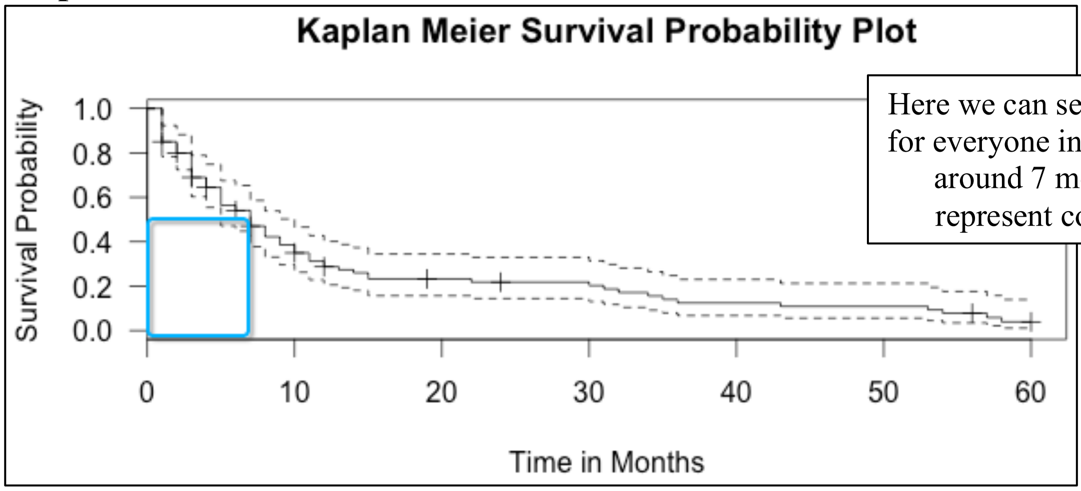
Interpretation:
Through this output we can say that age has a slight protective effect against having a spontaneous abortion (SA). As age increases, the rate of SA slightly decreases. Parity appears to be the greatest risk factor for SA. Every 1 increase in giving birth increases the rate of SA by 1.6 times. Having an induced abortion decreases the rate of having a SA by 63%.

Kaplan Meier & Survival Curves

Kaplan Meier curves do not provide a risk estimate (hazards ratio), however they are useful for explaining median survival time. Kaplan Meier estimates are crude (unadjusted) estimates that allow you to see the overall survival trend of a particular disease.

```
# Total population study survival time
km <- with(Surv(time,censor), data=hiv)
hiv.km <- survfit(km~1 , data=hiv)
summary(hiv.km)
plot(hiv.km, xlab="Time in Months", ylab="Survival Probability", main="Kaplan
Meier Survival Probability Plot", las=1)
```

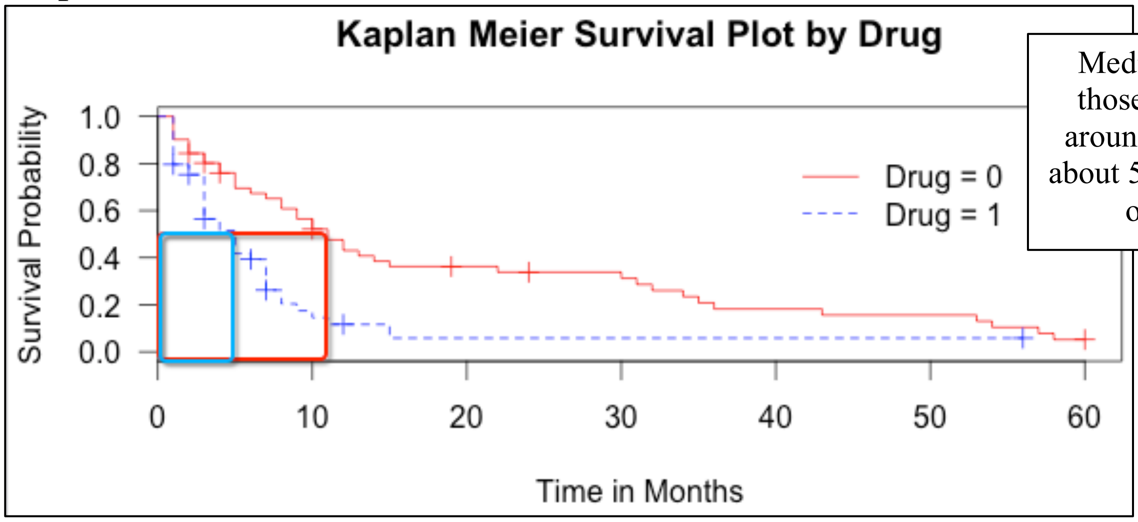
Output



Here we can see median survival time for everyone in this HIV population is around 7 months. Dashed lines represent confidence intervals.

```
# Now we can compare median survival time stratified by drug
hiv.drug <- survfit(km~drug, data=hiv)
summary(hiv.drug)
plot(hiv.drug, col=c(2,4), lty=c(1,2), xlab="Time in Months", ylab="Survival
Probability", main="Kaplan Meier Survival Plot by Drug", las=1)
legend(40, .9, lty=c(1,2), col=c(2,4), bty="n", legend=c("Drug = 0", "Drug = 1"))
```

Output



Median Survival for those on Drug = 0 is around 11 months and about 5 months for those on Drug = 1.

```
# logrank test for Kaplan Meier survival curves
survdiff(km~drug, data=hiv)
```

Output

```
Call:
survdiff(formula = km ~ drug, data = hiv)

        N Observed Expected (O-E)^2/E
drug=0 51       42     54.9      3.02
drug=1 49       38     25.1      6.60
       (O-E)^2/V
drug=0      11.9
drug=1      11.9

 Chisq= 11.9  on 1 degrees of freedom, p= 0.000575
```

The log-rank test for differences between the two survival curves is significant (p-value = 0.000575) and thus we reject the null hypothesis and conclude that based on the unadjusted Kaplan Meier estimates, there is a significant survival difference between drug groups.

Cox Regression & Survival Analysis

The goal of Cox Regression in survival analysis is to see if there is a difference in the survival times between two groups (example: those on a drug of interest and those taking a placebo) and the differing hazards (type of risk estimate) of experiencing an event between groups. The benefit of Cox regression is that we are also able to adjust for other confounding variables. In this example, taken from an online dataset on HIV survival, we adjust for age. This example analyzes the survival time based on the person's time in the study. Another common form of survival analysis is through basing the event time on the person's actual age. Chronic events are usually assessed through age, and acute events are often studied through time on study.

```
# Install the survival package
install.packages("survival")
library(survival)

# Import the online dataset
url <- "http://www.ats.ucla.edu/stat/r/examples/asa/hmohiv.csv"
hiv <- read.table(url, sep=",", header=TRUE)
summary(hiv)
# Sumary of hiv reveals variable names: ID, time, age, drug, censor, entdate,
enddate


# First check how many cases you have in your dataset (Frequency of dependent
variable)
table(hiv$censor)
```

Output

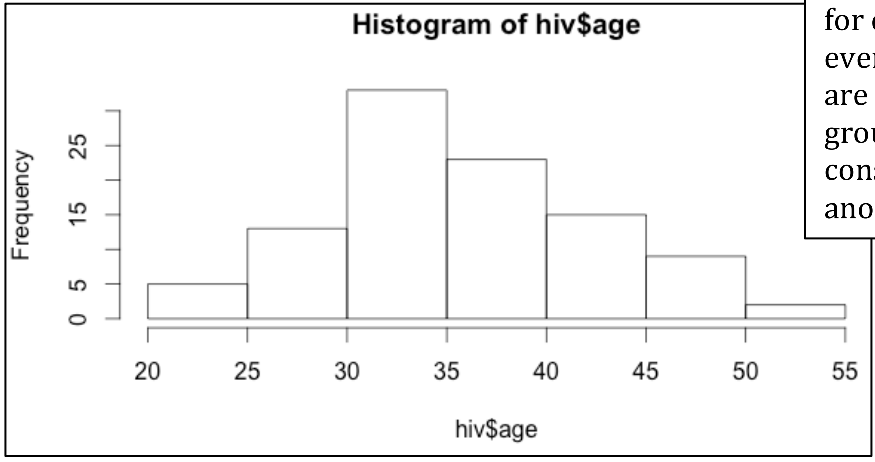
```
table(hiv$censor)

 0  1
20 80
```

This produces a very simple 2x2 table with the first row containing the coding for the censor (0/1) and the second row containing the counts for the frequency of each respective value that occurs in the dataset.

```
# Then check the frequencies of all categorical predictor variables and the
distributions of all continuous predictor variables that might be included in
your model.
hist(hiv$age)
table(hiv$drug)
```

This shows us that age is approximately normally distributed, which is what is desired for analysis, and the frequencies for drug tell us that there were about even numbers between groups. If there are very small numbers in a categorical group, you will probably want to consider collapsing the small group into another.

Output



```
> table(hiv$drug)

 0  1
51 49
```

```
# Create Cox equation
# You will want to compare an unadjusted model against the adjusted

# Base Model (unadjusted)
hiv.base <- coxph(formula = Surv(time, censor)~ drug, data=survival)
summary(hiv.base)
```

Output

```
Call:
coxph(formula = Surv(time, censor) ~ drug,
data = hiv)

  n= 100, number of events= 80

       coef exp(coef) se(coef)     z
Pr(>|z|)
drug 0.8309    2.2953   0.2418 3.436
0.00059 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’
0.05 ‘.’ 0.1 ‘ ’ 1

     exp(coef) exp(-coef) lower .95 upper
.95
drug     2.295     0.4357     1.429
3.687

Concordance= 0.611  (se = 0.036 )
Rsquare= 0.11   (max possible= 0.997 )
Likelihood ratio test= 11.6  on 1 df,
p=0.0006593
Wald test            = 11.81  on 1 df,
p=0.0005903
Score (logrank) test = 12.33  on 1 df,
p=0.0004464
```

This crude model produces a Beta coefficient for drug that is 0.83 and a hazard ratio (HR) for drug of 2.295. P-value is significant at 0.00059.

```
# Now compare these results with the Model adjusting for age
hiv.full <- coxph(formula = Surv(time, censor)~ drug + age, data=hiv)
summary(hiv.full)
```

Output

```
Call:
coxph(formula = Surv(time, censor) ~ drug + age, data = hiv)

  n= 100, number of events= 80

        coef exp(coef) se(coef)     z Pr(>|z|)
drug 1.01670   2.76405  0.25622 3.968 7.24e-05 ***
age  0.09714   1.10202  0.01864 5.211 1.87e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’

     exp(coef) exp(-coef) lower .95 upper .95
drug     2.764     0.3618     1.673     4.567
age      1.102     0.9074     1.062     1.143

Concordance= 0.711  (se = 0.042 )
Rsquare= 0.324   (max possible= 0.997 )
Likelihood ratio test= 39.13  on 2 df,   p=3.182e-09
Wald test            = 36.13  on 2 df,   p=1.431e-08
Score (logrank) test = 38.39  on 2 df,   p=4.602e-09
```

In this example, adjusting for age makes the HR for drug an even stronger risk (HR=2.76). This is a 20% increase in the HR, and thus we conclude that age is a confounder in this model that was depressing the true hazards associated with this drug. We conclude that age should be kept in our model. A percent change greater than 10% is often a better predictor of confounding than the p-value. But in this case, both p-value and percent change indicate confounding by age.

If you desire to interpret the hazards ratio for age it is important to note that this hazard is based on a one-unit increase in age, which is why it is so small of a risk. For a more clinically meaningful analysis you might want to convert this into an estimate that shows the increase in risk for every 5 or 10 years, instead.

```
# Change the units of age for increased hazards for every 5 years
hiv.age5 <- coxph(formula = Surv(time, censor)~ drug + I(age/5), data=hiv)
summary(hiv.age5)
```

Output

```
Call:
coxph(formula = Surv(time, censor) ~ drug +
I(age/5), data = hiv)

  n= 100, number of events= 80

            coef exp(coef) se(coef)     z Pr(>|z|)
drug      1.0167    2.7641   0.2562 3.968 7.24e-05
***
I(age/5) 0.4857    1.6253   0.0932 5.211 1.87e-07
***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’
0.1 ‘ ’ 1

         exp(coef) exp(-coef) lower .95 upper .95
drug         2.764     0.3618     1.673     4.567
I(age/5)     1.625     0.6153     1.354     1.951

Concordance= 0.711  (se = 0.042 )
Rsquare= 0.324   (max possible= 0.997 )
Likelihood ratio test= 39.13  on 2 df,   p=3.182e-09
Wald test            = 36.13  on 2 df,   p=1.431e-08
Score (logrank) test = 38.39  on 2 df,   p=4.602e-09
```

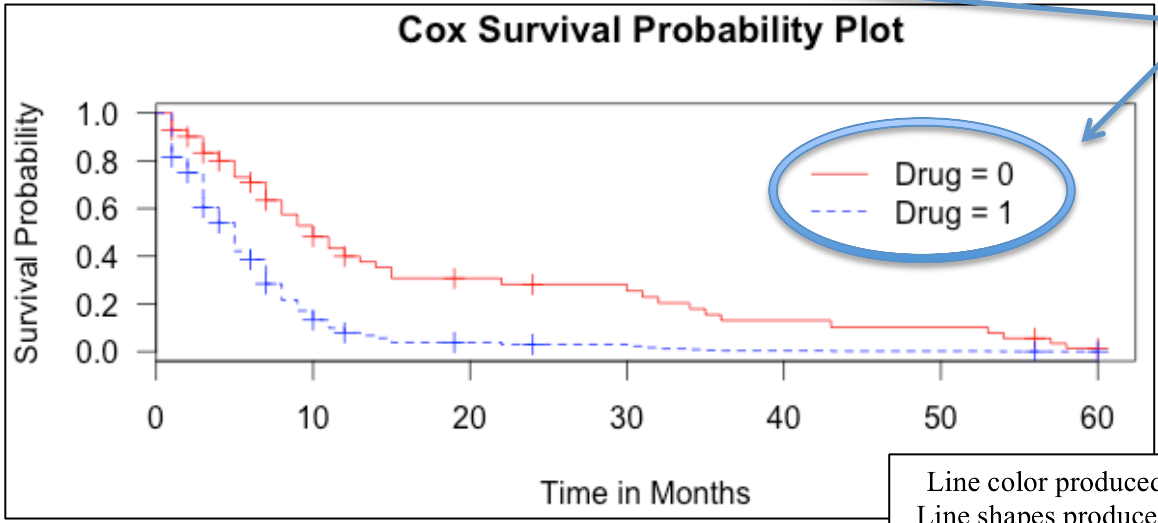
We can see here that by changing the units of age, it alters our beta coefficient and hazard ratio for age only; significance remains the same.

The HR for age is changed from a very small effect for one year (HR=1.1, or a 10% increase in risk of death each year) to a 60% increased risk of death for every 5 years (HR=1.6).

The HR for drug is unaffected.

```
# It is also important to get a visual representation of our survival curves
# Adjust by mean age and stratify by drug group
hiv.test <- data.frame(drug=0:1, age = mean(hiv$age))
plot(survfit(hiv.full, newdata=hiv.test), col=c(2,4), lty=c(1,2), las=1, main="
Cox Survival Probability Plot", ylab="Survival Probability", xlab="Time in
Months")
legend(40, .9, lty=c(1,2), col=c(2,4), bty="n", legend=c("Drug = 0", "Drug = 1"))
```

Output



Produced by the legend command.

Line color produced by: col=c(2,4)
Line shapes produced by: lty=c(1,2)

Checking the Assumptions for Cox Regression

Check Proportional Hazards Assumption via Cox Proportionality Assumption Test

```
# Cox PH assumption significance test
cox.zph(hiv.full)
```

Output

```
 rho      chisq      p
 drug   0.00188 0.000276 0.987
 age    0.01626 0.018958 0.890
 GLOBAL      NA 0.019077 0.991
```

You desire insignificant p-values for this test. Significant p-values indicate non-proportionality and insignificant p-values indicate that the proportionality assumption has been met. Here we can conclude that the proportionality assumption has been met and continue with the analysis.

What to do if the Proportionality Assumption is NOT met

```
# If the proportionality assumption is NOT met we need to look into the
discrepancy. One way to adjust for non-proportionality is through time
interaction terms.
# Example: Non-proportional hazards - Time interactions
interact <-coxph(formula = Surv(time, censor) ~ drug + drug:time + age, data =
hiv)
summary(interact)
```

Output

```
Call:
coxph(formula = Surv(time, censor) ~ drug + drug:time +
age, data = hiv)

  n= 100, number of events= 80

              coef exp(coef) se(coef)      z Pr(>|z|)
drug       3.62613  37.56733  0.59394  6.105 1.03e-09 ***
age        0.06890   1.07133  0.01977  3.486 0.000491 ***
drug:time -0.36541   0.69391  0.07933 -4.606 4.11e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘
’ 1

          exp(coef) exp(-coef) lower .95 upper .95
drug        37.5673    0.02662    11.729  120.3287
age          1.0713    0.93342     1.031    1.1137
drug:time    0.6939    1.44111     0.594    0.8107

Concordance= 0.793  (se = 0.042 )
Rsquare= 0.52   (max possible= 0.997 )
Likelihood ratio test= 73.42  on 3 df,   p=7.772e-16
Wald test            = 56.45  on 3 df,   p=3.374e-12
Score (logrank) test = 46.25  on 3 df,   p=5.027e-10
```

Time interaction, modeled through the addition of the `drug:time` variable (which is a variable of drug multiplied by time), help solve the problem of non-proportionality. However, interpretation becomes more complex when interaction terms are required, so they should only be used when your proportionality assumption is not met.

```
# Time interaction centered on the average time, or another time of clinical
significance, can further aid interaction term interpretations
mean(hiv$time)
# mean = 11.36
centered <-coxph(formula = Surv(time, censor) ~ drug + drug:I(time-11.36) + age,
data = hiv)
summary(centered)
```

Output

```
Call:
coxph(formula = Surv(time, censor) ~ drug + drug:I(time - 11.36) + age, data =
hiv)

  n= 100, number of events= 80

                         coef exp(coef) se(coef)      z Pr(>|z|)
drug                 -0.52494   0.59159  0.52381 -1.002 0.316268
age                   0.06890   1.07133  0.01977  3.486 0.000491 ***
drug:I(time - 11.36) -0.36541   0.69391  0.07933 -4.606 4.11e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

                     exp(coef) exp(-coef) lower .95 upper .95
drug                    0.5916     1.6904    0.2119    1.6515
age                     1.0713     0.9334    1.0306    1.1137
drug:I(time - 11.36)    0.6939     1.4411    0.5940    0.8107

Concordance= 0.793  (se = 0.042 )
Rsquare= 0.52   (max possible= 0.997 )
Likelihood ratio test= 73.42  on 3 df,   p=7.772e-16
Wald test            = 56.45  on 3 df,   p=3.374e-12
Score (logrank) test = 46.25  on 3 df,   p=5.027e-10
```

Centering on the average time `drug:I(time-11.36)` changes our interpretation of the average affect to be centered on the time 11.36 months, rather than on zero

Check Linearity Assumption via adding a square term.

```
# Check linearity of the continuous variables, in this model, age is the only
continuous variable to check, one way this can be done is by adding a square term
# First create the square term
hiv$age2 <- (hiv$age*hiv$age)
# Then add the square term to the model and re-run the regression
hiv.line <- coxph(Surv(time,censor)~drug + age + age2, data=hiv)
summary(hiv.line)
```

Output

```
Call:
coxph(formula = Surv(time, censor) ~ drug + age + age2,
data = hiv)
  n= 100, number of events= 80

               coef              exp(coef)     se(coef)
z       Pr(>|z|)
drug  1.0081168  2.7404353  0.2578233  3.910 9.23e-05 ***
age   0.1463216  1.1575685  0.1347883  1.086    0.278
age2 -0.0006735  0.9993268  0.0018265 -0.369    0.712
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1
‘ ’ 1

     exp(coef) exp(-coef) lower .95 upper .95
drug    2.7404     0.3649    1.6533     4.542
age     1.1576     0.8639    0.8888     1.508
age2    0.9993     1.0007    0.9958     1.003

Concordance= 0.709  (se = 0.042 )
Rsquare= 0.325   (max possible= 0.997 )
Likelihood ratio test= 39.27  on 3 df,   p=1.52e-08
Wald test            = 35.66  on 3 df,   p=8.848e-08
Score (logrank) test = 39.04  on 3 df,   p=1.7e-08
```

We desire the square term to be insignificant. Insignificant square terms mean that the linearity assumption is met. If the p-value is for age-squared were significant that would mean that age does not follow a linear pattern and we would need to keep the square term in the model. This complicates interpretation. In this example our p-value is 0.712 and thus we conclude that the linearity assumption is met and we do not need to keep the square term in our model.

Another method for dealing with non-linearity is to categorize the variable.

```
# If no standard clinical cut points exists, sort the variable to help determine
category cut points
sort(hiv$age) # orders age variable from smallest to largest

# The number of categories depends on what is biologically meaningful for that
variable as well on the number of cases in your study.

# Create the new categorized age variables
hiv$agecat[hiv$age >= 20 & hiv$age <=30] <- 1
hiv$agecat[hiv$age >= 31 & hiv$age <=40] <- 2
hiv$agecat[hiv$age >= 41] <- 3

# Re-run your cox regression with the new categorical age variables
# Categorical variables need to be dictated by the factor() command
# In R, the default reference category is the lowest group (group 1)
hiv3 <- coxph(Surv(time, censor)~ drug + factor(agecat), data=hiv)
summary(hiv3)
```

Output

```
Call:
coxph(formula = Surv(time, censor) ~ drug +
factor(agecat), data = hiv)

  n= 100, number of events= 80

                  coef exp(coef) se(coef)     z Pr(>|z|)
drug            0.8898    2.4346   0.2501 3.557 0.000375
***
factor(agecat)2 0.9694    2.6364   0.3575 2.712 0.006693
**
factor(agecat)3 1.6733    5.3295   0.4053 4.129 3.65e-05
***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1
' ' 1

                exp(coef) exp(-coef) lower .95 upper .95
drug                2.435     0.4107     1.491     3.975
factor(agecat)2     2.636     0.3793     1.308     5.313
factor(agecat)3     5.330     0.1876     2.408    11.794

Concordance= 0.669  (se = 0.041 )
Rsquare= 0.262   (max possible= 0.997 )
Likelihood ratio test= 30.41  on 3 df,   p=1.133e-06
Wald test            = 27.37  on 3 df,   p=4.927e-06
Score (logrank) test = 29.79  on 3 df,   p=1.527e-06
```

If you have a large enough dataset and the effect between the ages within each category is relatively consistent, categorizing variables can be an easier way to deal with non-linearity than including a square term. We can see through this output that the hazards for the age categories are quite different. The risk of an event for age group 2 against age group 1 is 2.6 times greater and the risk of an event for age group 3 compared to age group 1 is 5.3 times greater. The estimates are average risks between those included inside the range of each categorical variable.

Power Calculations & Sample Size Determination

Performing power and sample size calculations in R is dependent upon the statistical test you are performing. In order to do this, we must first download the 'pwr', power calculation package. Whatever value is left blank in the power calculation will be the outcome value calculated.

```
# Sample size and power calculations
install.packages("pwr")
library(pwr)
```

T-Test with Equal Sample Sizes:

```
# n = sample size (left blank because that is what we are testing for)
# d = effect size (difference between the groups)
# sig.level = alpha level (standard is .05)
# power = power to detect a difference (usually ranges between .8 and .9,
depending on analysis)
```

```
# Here we give sample size calculation code for a two sample, one sample, and
paired t-test
pwr.t.test(n = , d =.5 , sig.level =.05 , power =.9 , type = c("two.sample"))
pwr.t.test(n = , d =.5 , sig.level =.05 , power =.9 , type = c("one.sample"))
pwr.t.test(n = , d =.5 , sig.level =.05 , power =.9 , type = c("paired"))

# This can be converted into a power calculation when you insert the value for n
and leave the power value blank
```

Output

```
Two-sample t test power
calculation

              n = 85.03126
              d = 0.5
      sig.level = 0.05
          power = 0.9
    alternative = two.sided

NOTE: n is number in *each*
group
```

```
One-sample t test power
calculation

              n = 43.99551
              d = 0.5
      sig.level = 0.05
          power = 0.9
    alternative = two.sided
```

```
Paired t test power
calculation

              n = 43.99551
              d = 0.5
      sig.level = 0.05
          power = 0.9
    alternative = two.sided

NOTE: n is number of *pairs*
```

T-Test with Unequal Sample Sizes:

```
# To run a power calculation when you don't have equal sample sizes
pwr.t2n.test(n1 = 10 , n2=40 , d = .5 , sig.level = .05, power = )

# You cannot test for unequal sample size without knowing at least one of the
group sizes.
```

Output

```
t test power calculation

             n1 = 10
             n2 = 40
              d = 0.5
      sig.level = 0.05
          power = 0.2833987
    alternative = two.sided
```

Chi-Square Tests:

```
# w = effect size
# N = total sample size
# df=degrees of freedom
# A regular 2x2 table only has 1 degree of freedom
pwr.chisq.test(w = .3, N = , df = 1 , sig.level = .05, power = .9 )
```

Output

```
Chi squared power calculation

              w = 0.3
              N = 116.7491
             df = 1
      sig.level = 0.05
          power = 0.9

NOTE: N is the number of observations
```

ANOVA Tests:

```
# k = number of groups
# n = sample size in each group
# f = effect size
pwr.anova.test(k = 4, n = , f = .5, sig.level = .05, power = .9)
```

Output

```
Balanced one-way analysis of variance power calculation

              k = 4
              n = 15.18834
              f = 0.5
      sig.level = 0.05
          power = 0.9

NOTE: n is number in each group
```

Correlation Tests:

```
# n = total sample size
# r = effect size
pwr.r.test(n = , r = .7, sig.level = .05, power =.9 )
```

Output

```
approximate correlation power calculation (arctangh
transformation)

              n = 16.81483
              r = 0.7
      sig.level = 0.05
          power = 0.9
    alternative = two.sided
```

Test Between Two Proportions:

```
# n = number of observations per group
# h = effect size
pwr.2p.test(h = .3, n = , sig.level =.05, power =.9 )
```

Output

```
Difference of proportion power calculation for binomial distribution
(arcsine transformation)

              h = 0.3
              n = 233.4982
      sig.level = 0.05
          power = 0.9
    alternative = two.sided

NOTE: same sample sizes
```

One Sample Proportion Test:

```
# n = number of observations needed in the one group
# h = effect size
pwr.p.test(h = .3, n = , sig.level =.05, power =.9 )
```

Output

```
proportion power calculation for binomial distribution
(arcsine transformation)

              h = 0.3
              n = 116.7491
      sig.level = 0.05
          power = 0.9
    alternative = two.sided
```