Introduction to R

Sönke Ehret, NYU Data Services

Last Updated: December 2016

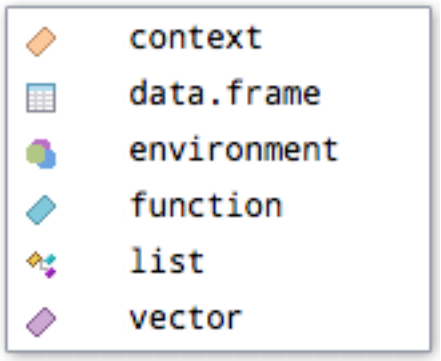
I. Overview of R's Interface

R's interactive command line will act as a calculator in basic form.

```
  1 + 1
```

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

RStudio Icons



Source: Kevin Ushey, RStudio _____ ## II. R Basics

Comments

Comments are annotations in code to make code easier to understand. Comments are writtine with a '#' in front. To comment out an entire section of code use Ctrl Shift 'C'.

Assignments

We can assign values to names using the assignment statement: '<-', which is just a less than sign '<' followed by a minus sign '-'.

```
  add <- 1 + 2
  add
```

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

A shortcut to create the assignment statement is 'Alt' '-'.

Functions

A function in R is the name of the function, followed by the parentheses. All of the arguments of the function are contained within the parentheses and can be set by the user.

```
  sqrt(2^3)
## [1] 2.828427
```

```
  sqrt(add)
## [1] 1.732051
```

Data Types

1. Vectors

Tie more than one value to a name by using the *'c'* function, which stands for 'combine.'

```
  vec1 <- c(4.6, 4.8, 5.2, 6.3, 6.8, 7.1, 7.2, 7.4, 7.5, 8.6)
  vec2 <- c('UT', 'IA', 'MN', 'AL', 'WI', 'MN', 'OH', 'IA', 'NY', 'IA')
```

Index vectors by using the square braces [ ].

```
vec2[1:3]
```

```
## [1] "UT" "IA" "MN"
```

```
vec2[c(1,3)]
```

```
## [1] "UT" "MN"
```

There are some functions which can be used on all types of vectors whether it is character or numeric.

```
  length(vec1)
  length(vec2)
  class(vec1)
  class(vec2)
```

There are other functions which only make sense to run on a numeric vector.

```
  mean(vec1)
  var(vec1)
  sum(vec1)
  max(vec1)
```

Transform the type of a vector

```
as.character(vec1)
as.factor(vec1)
```

Vectors can undergo all sorts of math and manipulations.

```
vec1+1
vec1*5
rep(1, times = 10)
rep(vec1, times = 5)
round(vec1/3, 2)
floor(vec1)
```

2. Matrices

A matrix brings two or more vectors together. There cannot be mixed data types within a matrix, i.e. a matrix is either all numeric or all character.

```
  m1 <- cbind(vec1, vec2)
  m1
```

```
##       vec1  vec2
##  [1,] "4.6" "UT"
```

```
##  [2,] "4.8" "IA"
##  [3,] "5.2" "MN"
##  [4,] "6.3" "AL"
##  [5,] "6.8" "WI"
##  [6,] "7.1" "MN"
##  [7,] "7.2" "OH"
##  [8,] "7.4" "IA"
##  [9,] "7.5" "NY"
## [10,] "8.6" "IA"
```

Index a matrix with braces: [rows, columns].

```
  m1[, 1]
  m1[2, ]
```

3. Data Frames

A data frame can hold all different classes of vectors.

```
  df1 <- data.frame(vec1, vec2)
```

Reference via brackets : dataframe[rows, columns]

```
  df1[, 1]
  df1[, 2]
  df1[, 'vec1']
```

Reference columns (variables) using the '$'.

```
  df1$vec1
  df1$vec2
```

4. Lists

A list is a container that can hold almost anything in one of its components.

```
  l1 <- list(vec1, df1, m1)
```

Reference a component by using double braces '[[ ]]' and an element within the component with following single braces '[ ].'

```
  l1[[1]]
  l1[[2]][2]
```

Missing Data

A missing value is represented by an 'NA', which is neither character or numeric.

```
  vec3 <- c(4.6, 4.8, 5.2, 6.3, 6.8, 7.1, 7.2, 7.4, 7.5, 8.6, NA)
```

Math functions, such as the *'sum'* function, will be unable to compute if there is a missing value in the data. To calculate the sum, we can set the argument na.rm = TRUE, which will remove the missing value from the calculation without removing the missing value from the vector.

```
  sum(vec3)
## [1] NA
  sum(vec3, na.rm = TRUE)
## [1] 65.5
```

III. Getting Started with R

A. Working Directory

A working directory is the path to where your data files are and/or where you want to save your files to.

```
getwd()
setwd('C:/Users/NYU User/Desktop')
dir()
```

B. Importing Datasets

Read .csv format (comma separated values)

```
health <- read.csv('Dataset.csv')
health <- read.table('Dataset.csv', sep = ',', header = TRUE)
```

Read .txt format

```
read.table('Dataset.txt', sep = ' ')
```

C. Packages

Packages are collections of user written functions.

```
install.packages('name of package')
library(name of package)
```

Use the package *'readr'*, to reading in tabular text data faster than the above functions. Use *'read_table'* when files your data is delimited by spaces, *'read_delim'* when you have other types of delimiters or *'read_csv'* when you have .csv files.

```
read_table('Dataset.txt')
read_delim('Dataset.txt', delim = '\t')
read_csv('Dataset.csv')
```

Use the package *'haven'*, to read in SPSS, Stata and SAS files

```
install.packages('haven')
library(haven)
read_spss('Dataset.sav')
read_dta('Dataset.dta')
read_sas('Dataset.sas7bdat')
```

Use the package *'readxl'* to read in .xls and .xlsx files.

```
install.packages('readxl')
library(readxl)
read_excel('Dataset.xls', sheet = 1, col_names = TRUE)
```

D. Export data file

```
  write.csv(health, 'health_csv.csv', row.names = FALSE)
```

Use *'readr'* to export csv files. About twice as fast as write.csv and never exports row names.

```
  write_csv(health, 'health_csv.csv')
```

Use *'haven'*, to export both SPSS and Stata files.

```
  write_spss(health, "health_spss.sav")
  write_dta(health, "health_stata.dta")
```

IV. Dataset Manipulation

A. Rename Columns (Variables)

```
  names(health)
##  [1] "id"      "gender"  "state"   "age"     "health1" "health2" "health3"
##  [8] "health4" "health5" "health6"

  names(health)[5:10] <- c('food', 'smoke', 'exercise', 'happy', 'alcohol', 'doctor')
  names(health)
##  [1] "id"       "gender"   "state"    "age"      "food"     "smoke"
##  [7] "exercise" "happy"    "alcohol"  "doctor"
```

B. Working with Missing Data

First we need to recode the -1 values to missing, NAs.

```
health$age
##  [1] 51 35 29 21 56 72 46 33 36 42 41 57 30 48 -1

which(health$age == -1)
## [1] 15

health$age[which(health$age == -1 )] <- NA
health$age
##  [1] 51 35 29 21 56 72 46 33 36 42 41 57 30 48 NA

is.na(health$age)
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [12] FALSE FALSE FALSE  TRUE
table(is.na(health$age))
##
## FALSE  TRUE
##    14     1
```

Then to run analysis, like the mean function, we must remember the na.rm = TRUE argument.

```
mean(health$age)
## [1] NA
```

```
mean(health$age, na.rm = TRUE)
## [1] 42.64286
```

C. Compute New Variables

To create a new variable in a data frame:

```
  health$health_sum <- health$food + health$smoke +
      health$exercise + health$happy + health$alcohol + health$doctor

health[1:5, 5:11]
```

```
##   food smoke exercise happy alcohol doctor health_sum
## 1    1     4        2     1       4      5         17
## 2    2     3        3     2       3      4         17
## 3    5     2        4     2       1      3         17
## 4    5     1        5     4       2      1         18
## 5    2     4        2     4       3      3         18
```

There are alternative and cleaner methods for summing up variables.

```
  rowSums(health[5:10])
  apply(health[, 5:10], 1, sum)
```

D. Recode Variables

To recode a continuous variable into a categorical variable, first create a new variable and use logical expressions to determine the new values of the new variable.

```
  summary(health$age)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   21.00   33.50   41.50   42.64   50.25   72.00       1

  health$age_cat[health$age <= 32.5] <-'Group 1'
  health$age_cat[health$age > 32.5 & health$age <= 50] <- 'Group 2'
  health$age_cat[health$age > 50] <-'Group 3'

  health[1:5, c('age', 'age_cat')]
##   age age_cat
## 1  51 Group 3
## 2  35 Group 2
## 3  29 Group 1
## 4  21 Group 1
## 5  56 Group 3
```

E. Recode Values

To recode values within the same variable, there is recode function in the *'car'* package.

```
  install.packages('car')
  library(car)
  health$health22 <- recode(health$smoke, '1=5;2=4;3=3;4=2;5=1')
```

F. Subsets of a Data Frame

There are several ways to subset the data frame. You can use the square brackets to specify certain rows and columns. The which command can be used to select rows that meet certain criteria. These two methods can be combined, or you can use the *'subset'* function.

```
  health[, c('id', 'gender', 'smoke')]
##    id gender smoke
## 1   1      M     4
## 2   2      F     3
## 3   3      F     2
## 4   4      M     1
## 5   5      M     4
## 6   6      M     5
## 7   7      F     5
## 8   8      M     2
## 9   9      F     3
## 10 10      M     3
## 11 11      F     4
## 12 12      F     4
## 13 13      M     2
## 14 14      F     3
## 15 15      M     2

  health[1:3, c('id', 'gender', 'smoke')]
##   id gender smoke
## 1  1      M     4
## 2  2      F     3
## 3  3      F     2

  which(health$age > 40)
## [1]  1  5  6  7 10 11 12 14
  which(health$age > 40 & health$age < 50)
## [1]  7 10 11 14
  which(health$age < 25 | health$age > 50)
## [1]  1  4  5  6 12

  sub1 <- health[which(health$age > 40), c('age','smoke')]
  sub2 <- subset(health, age > 40, select=c('age','smoke'))

  sub1
##    age smoke
## 1   51     4
## 5   56     4
## 6   72     5
## 7   46     5
## 10  42     3
## 11  41     4
## 12  57     4
## 14  48     3
```

V. Descriptive Statistics

A. Basic Statistics

The *'summary function'* is useful in many situations, it can be used to obtain a six number summary of your variables.

```
  summary(health)
##        id       gender     state            age             food
##  Min.   : 1.0   F:7    Min.   :1.000   Min.   :21.00   Min.   :1.000
##  1st Qu.: 4.5   M:8    1st Qu.:1.000   1st Qu.:33.50   1st Qu.:2.000
##  Median : 8.0          Median :2.000   Median :41.50   Median :3.000
##  Mean   : 8.0          Mean   :1.867   Mean   :42.64   Mean   :2.933
##  3rd Qu.:11.5          3rd Qu.:2.500   3rd Qu.:50.25   3rd Qu.:4.500
##  Max.   :15.0          Max.   :3.000   Max.   :72.00   Max.   :5.000
##                                        NA's   :1
##      smoke          exercise         happy      alcohol        doctor
##  Min.   :1.000   Min.   :2.000   Min.   :1   Min.   :1.0   Min.   :1.000
##  1st Qu.:2.000   1st Qu.:3.000   1st Qu.:2   1st Qu.:2.0   1st Qu.:2.000
##  Median :3.000   Median :3.000   Median :3   Median :2.0   Median :3.000
##  Mean   :3.133   Mean   :3.267   Mean   :3   Mean   :2.6   Mean   :3.133
##  3rd Qu.:4.000   3rd Qu.:4.000   3rd Qu.:4   3rd Qu.:3.0   3rd Qu.:4.000
##  Max.   :5.000   Max.   :5.000   Max.   :5   Max.   :5.0   Max.   :5.000
##
##    health_sum      age_cat
##  Min.   :16.00   Length:15
##  1st Qu.:17.00   Class :character
##  Median :18.00   Mode  :character
##  Mean   :18.07
##  3rd Qu.:19.00
##  Max.   :21.00
##

  summary(health$age)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   21.00   33.50   41.50   42.64   50.25   72.00       1
```

Or you can use individual functions to obtain one value at a time.

```
  mean(health$age, na.rm = TRUE)
## [1] 42.64286
  median(health$age, na.rm = TRUE)
## [1] 41.5
  sd(health$age, na.rm = TRUE)
## [1] 13.4999
  quantile(health$age, na.rm = TRUE)
##    0%   25%   50%   75%  100%
## 21.00 33.50 41.50 50.25 72.00
```

B. Frequencies and Cross Tabulations

```
  table(health$gender)
##
## F M
```

```
## 7 8
```

```
  prop.table(table(health$gender))
##
##         F         M
## 0.4666667 0.5333333
```

```
  table(health$gender, health$age_cat)
##
##     Group 1 Group 2 Group 3
##   F       1       5       1
##   M       2       2       3
```

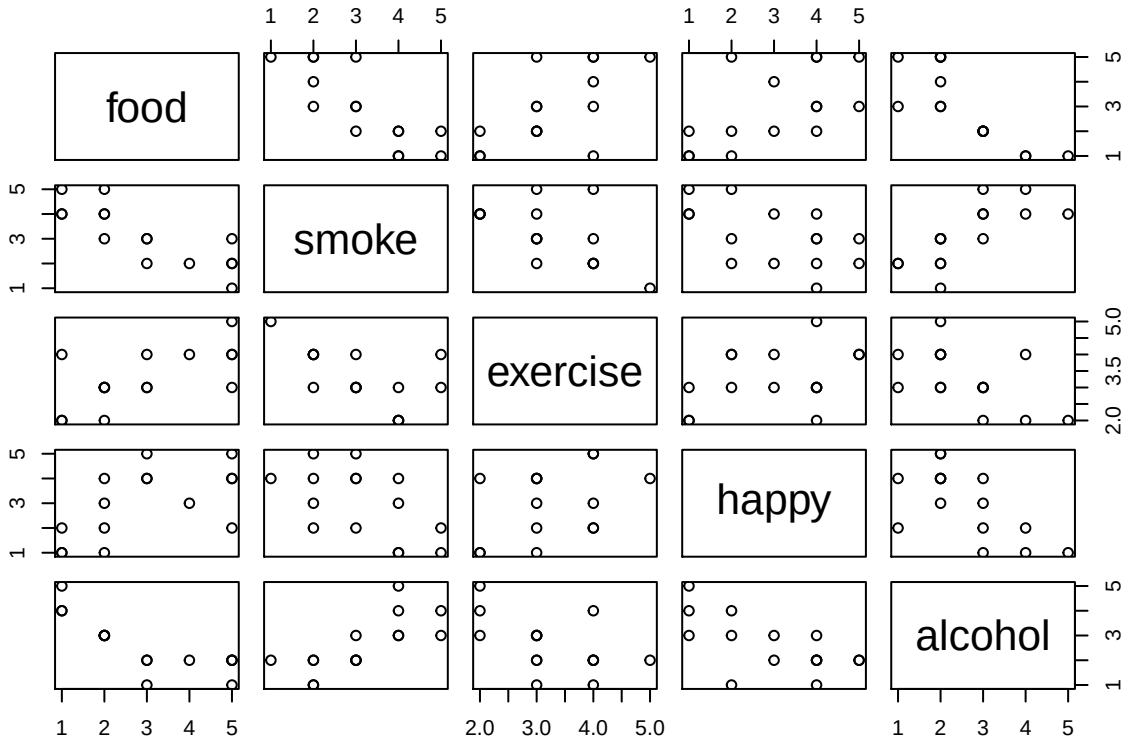
```
prop.table(margin.table(table(health$gender, health$age_cat), 1))
##
##   F   M
## 0.5 0.5
prop.table(margin.table(table(health$gender, health$age_cat), 2))
##
##   Group 1   Group 2   Group 3
## 0.2142857 0.5000000 0.2857143
```

C. Correlation

```
  cor(health[5:9])
```

```
##                food      smoke   exercise      happy    alcohol
## food      1.0000000 -0.8185308  0.6464463  0.5927556 -0.8058005
## smoke    -0.8185308  1.0000000 -0.5809546 -0.5530496  0.7404957
## exercise  0.6464463 -0.5809546  1.0000000  0.4000762 -0.5334566
## happy     0.5927556 -0.5530496  0.4000762  1.0000000 -0.6306562
## alcohol  -0.8058005  0.7404957 -0.5334566 -0.6306562  1.0000000
```

```
  plot(health[5:9])
```

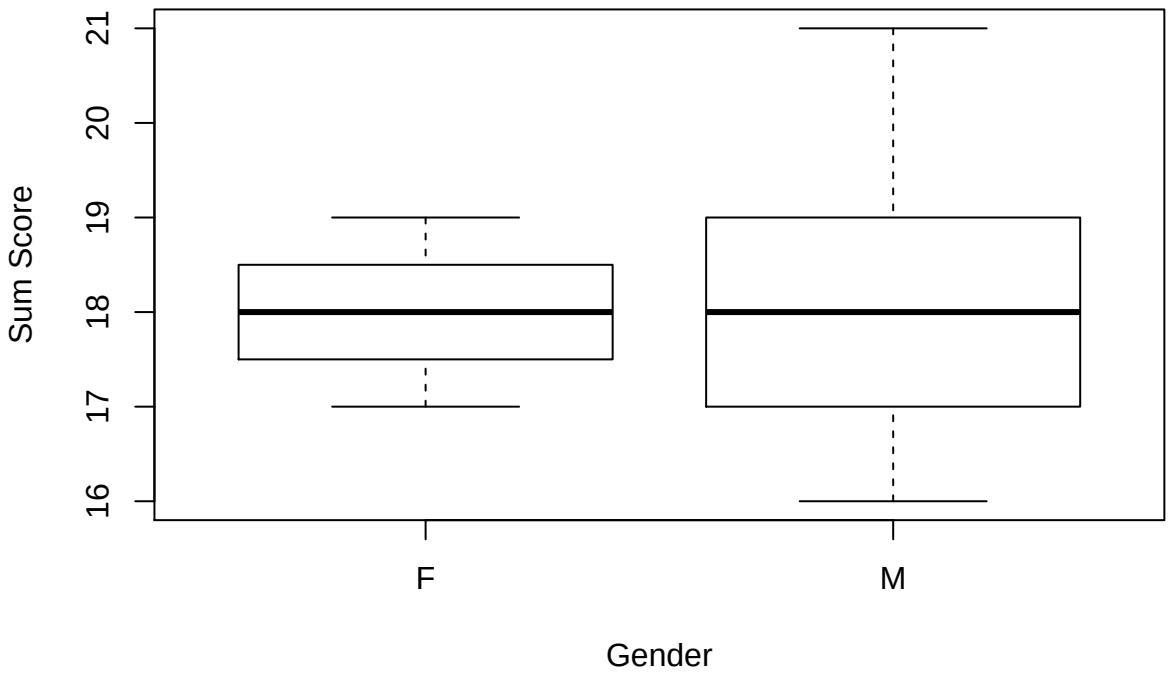


VI. Graphics

Base Graphics

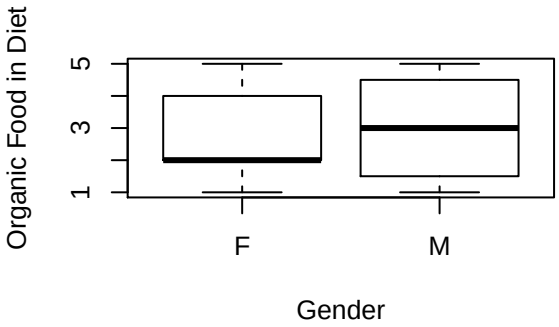
Formula notation is introduced below where you have your dependent variable on the left of the '~' and you have your independent variable(s) on the right side.

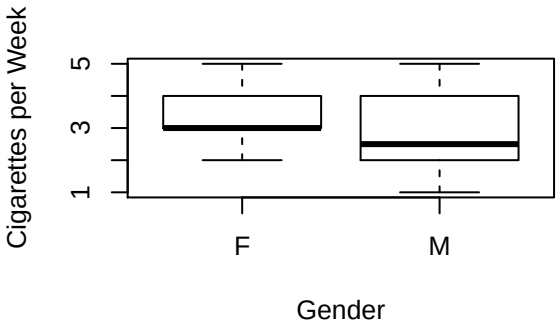
```
boxplot(health_sum ~ gender, ylab = 'Sum Score', xlab = 'Gender', data = health)
```

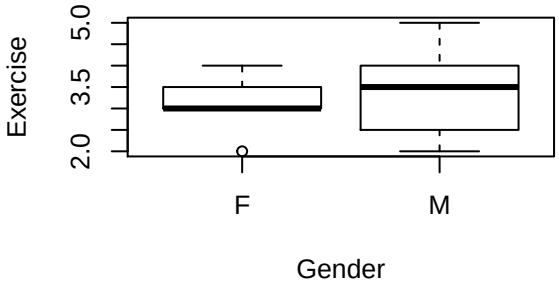


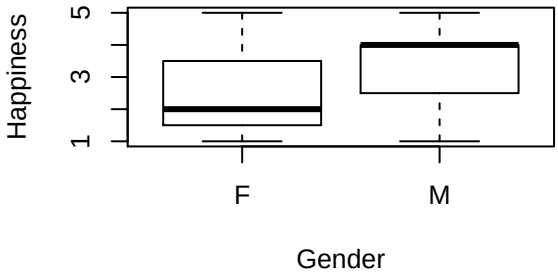
Use the *'par'* function to include more than one graphic in a single window.

```
par(mfrow = c(2, 2))
boxplot(food ~ gender, ylab = 'Organic Food in Diet', xlab = 'Gender', data = health)
boxplot(smoke ~ gender, ylab = 'Cigarettes per Week', xlab = 'Gender', data = health)
boxplot(exercise ~ gender, ylab = 'Exercise', xlab = 'Gender', data = health)
boxplot(happy ~ gender, ylab = 'Happiness', xlab = 'Gender', data = health)
```

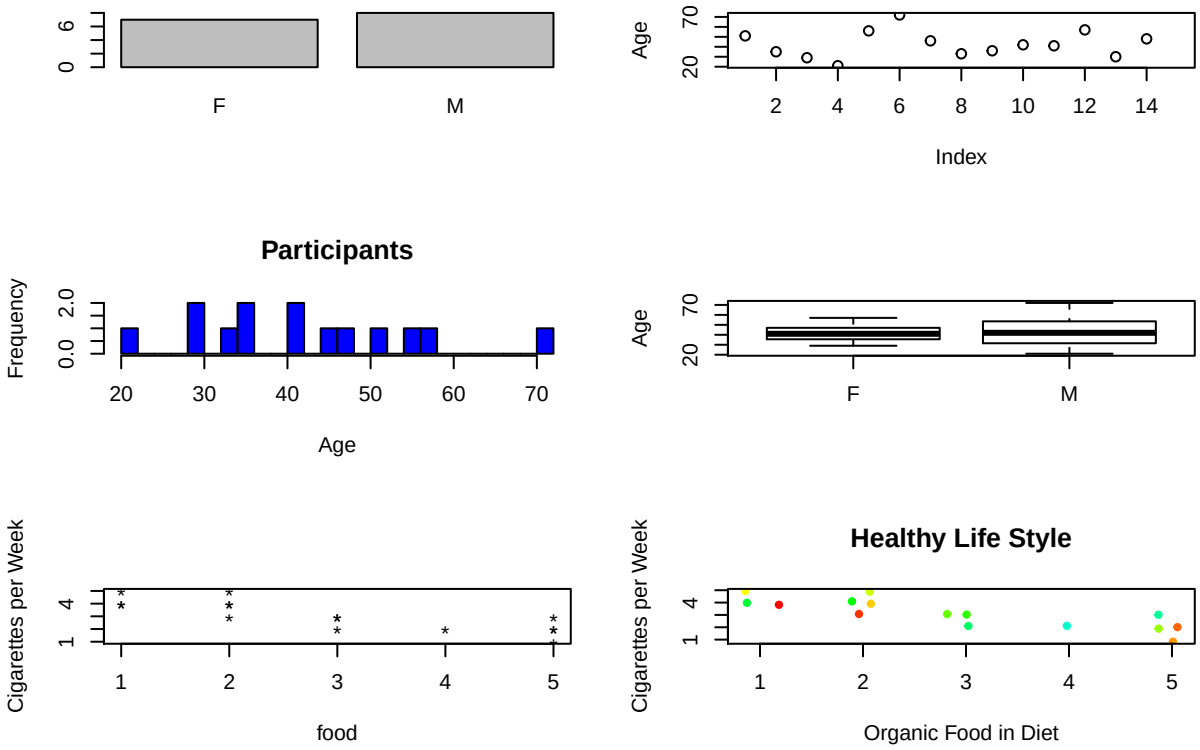








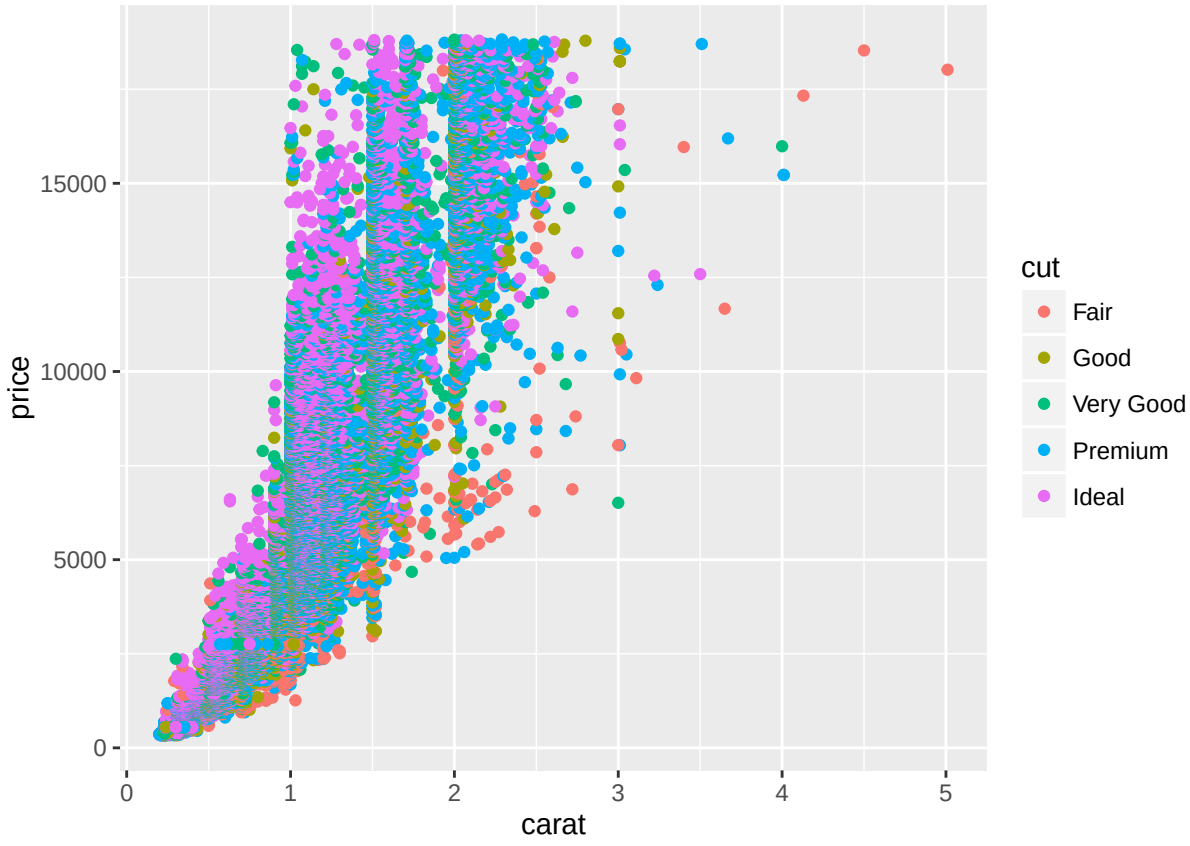

```r
par(mfrow = c(3, 2))
barplot(table(health$gender))
plot(health$age, ylab = 'Age')
hist(health$age, col = 'blue', breaks = 20, xlab = 'Age', main = 'Participants')
boxplot(age ~ gender, data = health, ylab = 'Age')
plot(smoke ~ food, pch = '*', data = health, ylab = 'Cigarettes per Week')
plot(jitter(smoke) ~  jitter(food), pch = 20, col = rainbow(30), xlab = 'Organic Food in Diet', ylab =
```




```
par(mfrow = c(1, 1))
```

Advanced Graphics with ggplot2

```
library('ggplot2')
qplot(x = carat, y = price, color=cut, data = diamonds)
```



VII. Hypothesis Testing

Chi-square Test

```
  chisq.test(health$gender, health$age_cat)
## Warning in chisq.test(health$gender, health$age_cat): Chi-squared
## approximation may be incorrect
##
##  Pearson's Chi-squared test
##
## data:  health$gender and health$age_cat
## X-squared = 2.619, df = 2, p-value = 0.2699

  summary(table(health$gender, health$age_cat))
## Number of cases in table: 14
## Number of factors: 2
## Test for independence of all factors:
##  Chisq = 2.619, df = 2, p-value = 0.2699
##  Chi-squared approximation may be incorrect
```

T-tests

Just by changing the arguments within the *'t.test'* function, we can run a one sample t-test, an independent t-test and a paired t-test.

```
  t.test(health$health_sum, mu=3)
##
##  One Sample t-test
##
## data:  health$health_sum
## t = 47.721, df = 14, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 3
## 95 percent confidence interval:
##  17.38950 18.74383
## sample estimates:
## mean of x
##  18.06667

  t.test(health_sum ~ gender, data = health)
##
##  Welch Two Sample t-test
##
## data:  health_sum by gender
## t = -0.19849, df = 10.859, p-value = 0.8463
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.513264  1.263264
## sample estimates:
## mean in group F mean in group M
##          18.000          18.125

  t.test(health$food, health$smoke, paired = TRUE)
##
##  Paired t-test
##
## data:  health$food and health$smoke
## t = -0.2983, df = 14, p-value = 0.7699
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.638005  1.238005
## sample estimates:
## mean of the differences
##                    -0.2
```

You can access different parts of the output that are stored, such as the t-statistic or p-value.

```
  my_ttest <- t.test(health$food, health$smoke, paired = TRUE)
  summary(my_ttest)
##             Length Class  Mode
## statistic   1      -none- numeric
## parameter   1      -none- numeric
## p.value     1      -none- numeric
## conf.int    2      -none- numeric
## estimate    1      -none- numeric
## null.value  1      -none- numeric
```

```
## alternative 1      -none- character
## method      1      -none- character
## data.name   1      -none- character

  my_ttest$statistic
##          t
## -0.2983003
```

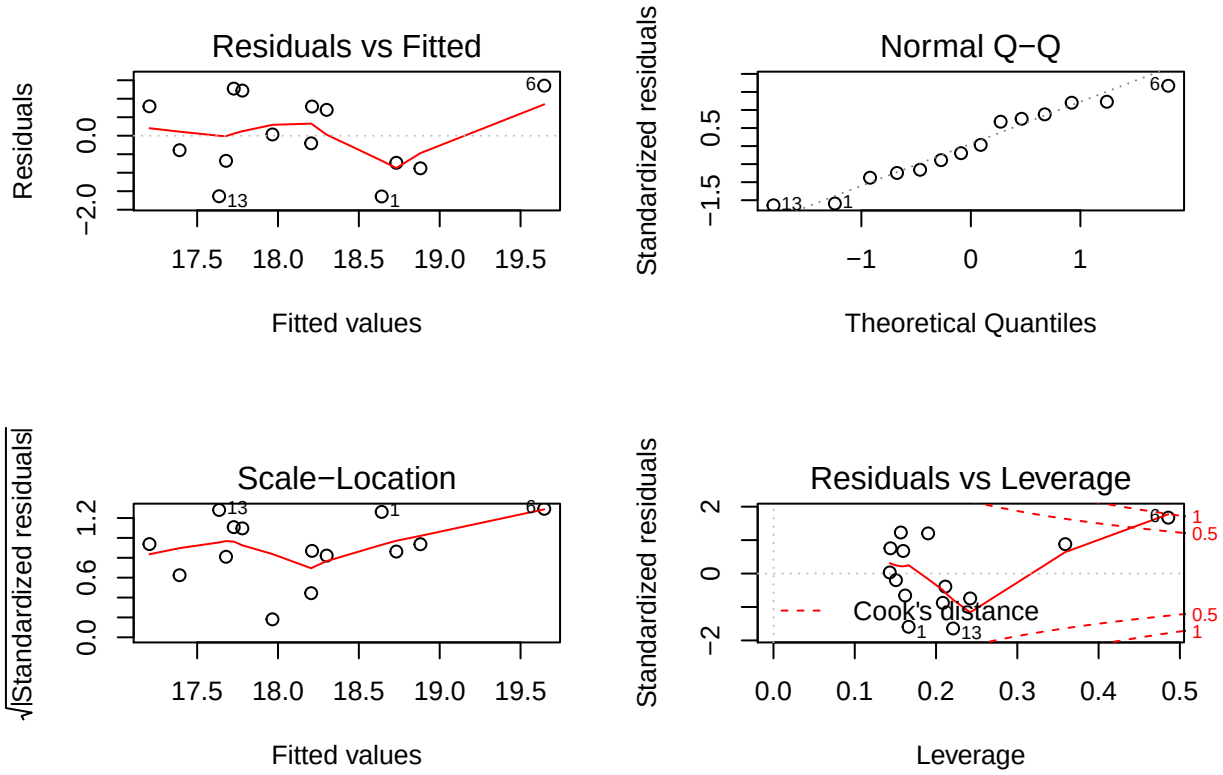
Linear Regression Model

When you run a regression model, you should save the results to an lm object through use of the assignment statement. This way you will be able to access different parts of the results later on.

```
  lm_health <- lm(health_sum ~ age + gender, data = health)
  summary(lm_health)
##
## Call:
## lm(formula = health_sum ~ age + gender, data = health)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.6414 -0.7185 -0.0855  0.7936  1.3532
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 16.00277    1.05990  15.098 1.06e-08 ***
## age          0.04788    0.02326   2.059    0.064 .
## genderM      0.19680    0.60511   0.325    0.751
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.129 on 11 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.2886, Adjusted R-squared:  0.1592
## F-statistic: 2.231 on 2 and 11 DF,  p-value: 0.1537

  confint(lm_health)
##                    2.5 %      97.5 %
## (Intercept) 13.669957304 18.33558382
## age         -0.003310717  0.09906829
## genderM     -1.135034341  1.52862741
```

```
  par(mfrow = c(2, 2))
  plot(lm_health)
```



ANOVA

```
aov_health <- aov(health_sum ~ state + gender, data = health)
summary(aov_health)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## state        1  0.132  0.1320   0.076  0.787
## gender       1  0.056  0.0564   0.033  0.860
## Residuals   12 20.745  1.7287
```