An Introduction to R

A Step Towards Statistical Analysis in R

Cygu Steve Bicko (cygu@aims.ac.za)

Contents

About the Author:

Cygu S. Bicko is currently a student at the University of Stellenbosch undertaking a Master's degree in Biomathematics. He holds a Masters degree in Mathematical Sciences (Statistical Epidemiology) from the University of Cape Town and Bachelors degree in Applied Statistics with Computing. He has worked in various Research firms in Kenya and is experienced in various Statistical and computing packages (including R, SPSS, Stata, Python, Sagemath).

1 Factors and Frequency Tables

Factors defines a partion into groups. It specifies discrete classification (grouping) of the vectors.

Suppose we have the following specifying the states of students in a school in Homabay County.

```
vilg <- c("tas", "qld", "sa", "sa", "sa", "vic", "nt", "act",
          "qld", "nsw", "wa", "nsw", "nsw", "vic", "vic", "vic",
          "nsw", "qld", "qld", "vic", "nt", "wa", "wa", "qld",
          "sa", "tas", "nsw", "nsw", "wa", "act")
```

Now we create *factor* using the function $factor()$:

```
vilgof <- factor(vilg)
vilgof
```

```
##  [1] tas qld sa  sa  sa  vic nt  act qld nsw wa  nsw nsw vic vic vic nsw
## [18] qld qld vic nt  wa  wa  qld sa  tas nsw nsw wa  act
## Levels: act nsw nt qld sa tas vic wa
```

The levels of the factor can be obtained using the function $levels()$.

```
levels(vilgof)
```

```
## [1] "act" "nsw" "nt"  "qld" "sa"  "tas" "vic" "wa"
```

Suppose we have marks scores from a test of these student and we wish to calculate the sample mean score for each village, we can use the special function **tapply()**.

```
marks <- c(60, 49, 40, 61, 64, 60, 59, 54, 62, 69, 70,
           42, 56, 61, 61, 61, 58, 51, 48, 65, 49, 49,
           41, 48, 52, 46,59, 46, 58, 43)

markmeans <- tapply(marks, vilgof, mean)
markmeans
```

```
##   act   nsw    nt   qld    sa   tas   vic    wa
## 48.50 55.00 54.00 51.60 54.25 53.00 61.60 54.50
```

Suppose we now want to calculate the standard error of the marks for each village. There is no built in function to calculate the standard error but we can write our own function.

```
stder <- function(x){
  sqrt(var(x)/length(x))
}
```

We can now call the function $stderr()$

```
markstder <- tapply(marks, vilgof, stder)
markstder
```

```
##       act       nsw        nt       qld        sa       tas       vic
## 5.5000000 3.9665266 5.0000000 2.6570661 5.3909647 7.0000000 0.8717798
##        wa
## 6.2249498
```

Frequency tables can be computed in r using the function *table*().

```
vilgfr <- table(vilgof)
vilgfr
```

```
## vilgof
## act nsw  nt qld  sa tas vic  wa
##   2   6   2   5   4   2   5   4
```

We can also use the function *tapply*() to do the same:

```
vilgfr <- tapply(vilgof, vilgof, length)
vilgfr
```

```
## act nsw  nt qld  sa tas vic  wa
##   2   6   2   5   4   2   5   4
```

Suppose we want to do a **cross tabulation** of "marks class" for each village, the function *cut*() (get help on these functions) can be used to create the classes.

```
markf <- factor(cut(marks, breaks = 35 + 10*(0:7)))
table(markf, vilgof)
```

```
##          vilgof
## markf     act nsw nt qld sa tas vic wa
##   (35,45]   1   1  0   0  1   0   0  1
##   (45,55]   1   1  1   4  1   1   0  1
##   (55,65]   0   3  1   1  2   1   5  1
##   (65,75]   0   1  0   0  0   0   0  1
```

2 Computing Lists and Data frames

2.1 Lists

A *list* is a collection of objects (components).

```
Lst <- list(name = "Dwale", wife = "Njoki", no.childreen = 10, child.ages = seq(1,20, 2))
length(Lst)
```

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

The components of a *list* are numbered and can alway be referenced using [[]] or $.

2.2 Data frames

A data frame is a combination of list(s). These can be matrices, arrays and other vectors.

2.3 Creating data frames

Objects forming the data subsets are placed on the columns (components) and the the function $data.frame()$ is used. A *list* can also be coerced into data frame using the function $as.data.frame()$.

```
students <- data.frame(Village = vilg, Marks = marks, grps = markf)
```

Access to an individual variable in a data frame uses the $ operator in the following syntax: dataframe$variable, with(dataframe, variable) or attach() and (detach()).

3 Functions and Expressions

3.1 Functions

A function takes arguments as input and returns an object as output. There are functions to do all sorts of things. R has inbuilt functions such as mean, sum, sd, plot,... etc. However, we can also define our own functions to do specific tasks. We show some examples below.

```
x <- c(74, 31, 95, 61, 76, 34, 23, 54, 96)
results <- list(Mean = mean(x), Sum = sum(x), Length = length(x), SD = sd(x), Max = max(x))
results
```

```
## $Mean
## [1] 60.44444
##
## $Sum
## [1] 544
##
## $Length
## [1] 9
##
```

```
## $SD
## [1] 27.14365
##
## $Max
## [1] 96
```

```
plot(x,x)
```

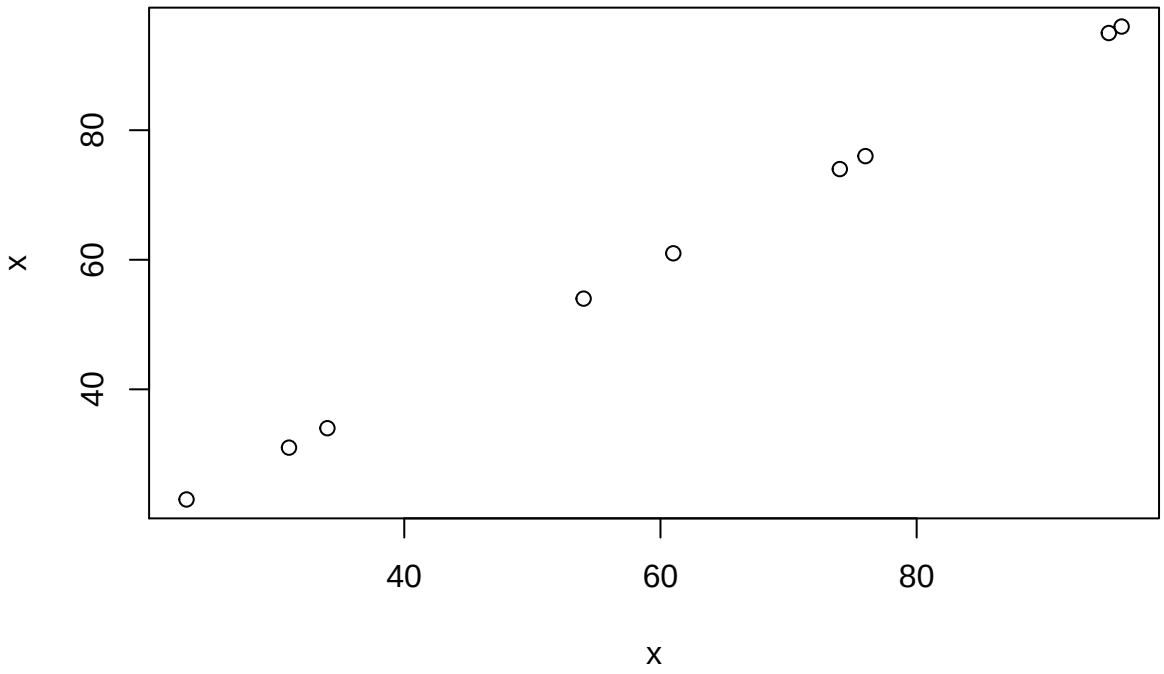

Figure 1:

The syntax for writing R functions is:

```
functionName <- function(args)
  {
  ... code ...
  }
```

Example of Function in R:

```
SD <- function(x)
{
  dev.sqd <- sum((x- mean(x))^2)
  std.dev <- sqrt(dev.sqd/(length(x)-1))
  cat("The Standard Deviation is:\n", std.dev)
}
x1 <- c(74, 31, 95, 61, 76, 34, 23, 54, 96)
SD(x1)
```

```
## The Standard Deviation is:
##  27.14365
```

3.2 Conditions Statements and Loop

The if condition and statement syntax

```
if (...condition....) {
...code 1...
}
else {
...code 2...
}
```

The while loop syntax

```
while (...condition....)
{...code...}
```

The for loop syntax

```
for(rank of indices)
{...code...}
```

Examples: If conditions

```
Iq <- 10
Thinking <- 4
if (Thinking > 3){
  Iq <- "Normal"
  Thinking <- "Straight"
} else{
    Iq <- "Dwale"
    Thinking <- "Kuria"
}
Iq
```

```
## [1] "Normal"
```

```
Thinking
```

```
## [1] "Straight"
```

Examples: for loop

```
cunt <- c(0,0,0,0)
n <- c(2,4,6,4)
for (i in 1:length(n)){
  cunt <- c(cunt, rep(i, n[i]))
}
cunt
```

```
##  [1] 0 0 0 0 1 1 2 2 2 2 3 3 3 3 3 3 4 4 4 4
```

4 Data Manupilations

4.1 Load and Read R data

Data analysis involves a large amount of manupilation and cleaning to facilitate downstream data analysis. This section covers basic data manupilation using R default functions. Many packages contain data sets. You can see a list of all data sets in all loaded packages using

```
data()
```

You can use data sets by simply typing their names. But if you have already used that name for something or need to refresh the data after making some changes you no longer want, you can explicitly load the data using the data() function with the name of the data set you want.

R comes with some data and ready for to be used. For Example, the iris data frame contains 5 variables measured for each of 150 iris plants (the obervational units). The iris data set is included with the default R installation and located in a package called datasets which is always available. There are several ways we can get some idea about what is in the iris data frame.

```
data("iris")
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

```
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
```

```
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```
names(iris)
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"
## [5] "Species"
```

We can access the variables using the methods we described above.

```
iris$Sepal.Length[1:10]
```

```
##  [1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9
```

```
with(iris, Sepal.Length)[1:10]
```

```
##  [1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9
```

The attach() function in R can be used to make objects within dataframes accessible in R with fewer keystrokes, but we strongly discourage its use, as it often leads to name conflicts.

4.2 Using Your Own Data

RStudio will help you import your own data. To do so use the "Import Dataset" button in the Workspace tab. You can load data from text files, from the web, or from google spreadsheets. The folowing commands can be used to import data in R:

1. From A Comma Delimited Text File (.csv)

   read.csv("path/Name.csv", header = TRUE)

   or

   mydata <- read.table("Path/mydata.csv", header=TRUE, sep=",")

2. From Excel

   ```
   # read in the first worksheet from the workbook myexcel.xlsx
   # first row contains variable names
   library(xlsx)
   mydata <- read.xlsx("Path/myexcel.xlsx", 1)

   # read in the worksheet named mysheet
   mydata <- read.xlsx("Path/myexcel.xlsx", sheetName = "mysheet")
   ```

3. From SPSS

   ```
   # save SPSS dataset in trasport format
   get file='Path\mydata.sav'.
   export outfile='Path\mydata.por'.

   # in R
   library(Hmisc)
   mydata <- spss.get("Path/mydata.por", use.value.labels=TRUE)
   # last option converts value labels to R factors
   ```

4. From Stata

```
# input Stata file
library(foreign)
mydata <- read.dta("Path/mydata.dta")
```

Something to know....

Value labels can created like this in R:

Norminal Data

```
x <- rep(c(1,2,4), 10)
x
```

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

```
x <- factor(x, levels = c(1,2,3,4), labels = c("Green", "Yellow", "Blue", "Black"))
table(x)
```

```
## x
##  Green Yellow   Blue  Black
##     10     10      0     10
```

Ordinal Data

```
x <- rep(c(1,2,4), 10)
x
```

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

```
x <- ordered(x, levels = c(1,2,3,4), labels = c("Low", "Medium", "High", "Very High"))
table(x)
```

```
## x
##       Low    Medium      High Very High
##        10        10         0        10
```

Variable labels can also be added in R:

This can be done using function names() and then refer to the variable by position index.

```
names(x) <- "This is the name of this variable"
```

Examples on Reading Data

Now let use, the dataset class data.csv to illustrate basic data manupilation in R.

```
dat <- read.csv("/media/cygu/Disk/Works/UON/class_data.csv", header = TRUE)
attach(dat)
names(dat)
```

```
##  [1] "years"      "birthmonth" "gender"     "siblings"   "height"
##  [6] "handspan"   "footlength" "breath"     "armcross"   "tongue"
## [11] "dice"       "beans"      "handed"
```

To remove rows from dat with missing data, the R function to check for this is complete.cases() or na.omit()

```
dat <- dat[complete.cases(dat), ] # or dat <- na.omit(dat)
plotcol <- c("red","green")[dat$gender]
plot(height, footlength, pch = "+", col = plotcol, cex = 1.5)
```

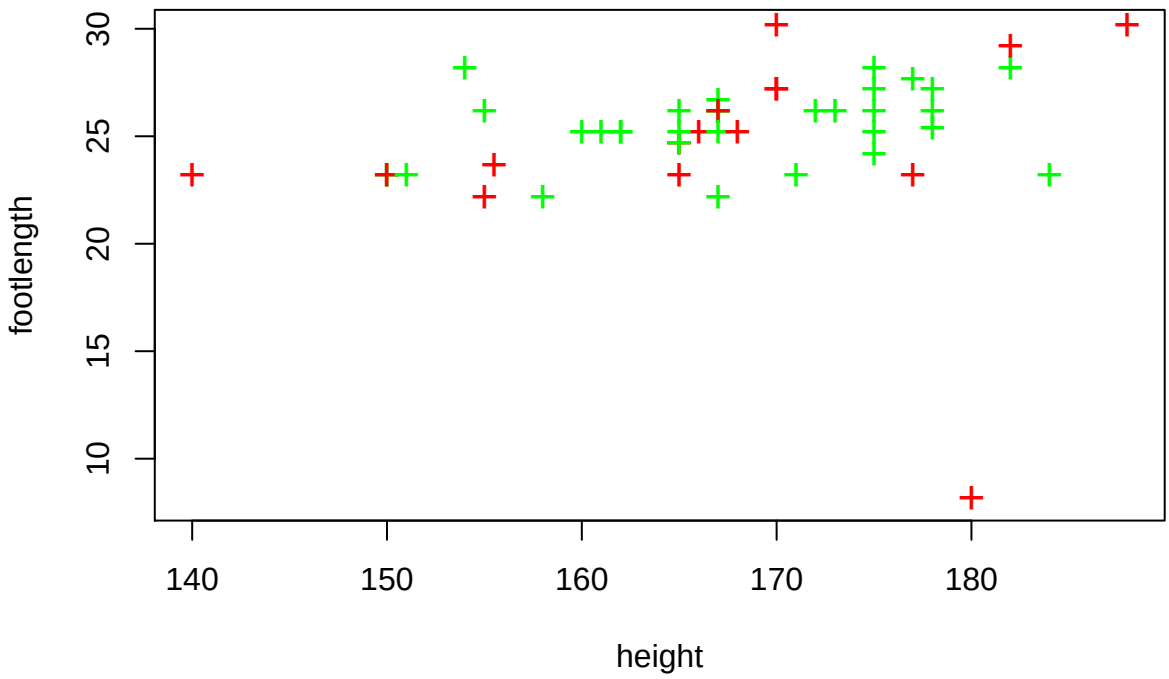

Figure 2:

5 Basic Statistics

In this section, we describe various (basic, to some) statistics. This will include:

a) Descriptive Statistics

b) Frequency Counts

c) Crosstabulations

d) Correlations

e) Multiple Linear Regression

f) Analysis of Variance

5.1 Descriptive Statistics

There are a wide range of packages and functions in R that can be used to obtain summary statistics. These include:

1. The sapply( ) function

Possible functions used in sapply include mean, sd, var, min, max, median, range, and quantile.

```
dat <- read.csv("/media/cygu/Disk/Works/UON/class_data.csv", header = TRUE)
sapply(dat, mean)
```

```
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

##      years birthmonth     gender   siblings     height   handspan
##    25.8600         NA         NA     3.7000         NA    20.3520
## footlength     breath   armcross     tongue       dice      beans
##    24.9440         NA         NA         NA     5.0800         NA
##     handed
##     0.6521
```

2. The summary( ) function

```
summary(dat)
```

```
##      years          birthmonth gender    siblings        height
##  Min.   :20.00   December: 7   F:19   Min.   : 0.0   Min.   :140.0
##  1st Qu.:24.00   August  : 6   M:31   1st Qu.: 2.0   1st Qu.:162.0
##  Median :26.00   May     : 5          Median : 3.0   Median :167.0
##  Mean   :25.86   April   : 4          Mean   : 3.7   Mean   :167.6
##  3rd Qu.:27.00   February: 4          3rd Qu.: 5.0   3rd Qu.:175.0
##  Max.   :32.00   January : 4          Max.   :11.0   Max.   :188.0
##                  (Other) :20                         NA's   :3
##     handspan       footlength        breath       armcross  tongue
##  Min.   :17.00   Min.   : 8.00   Min.   : 9.00   left :23      : 1
##  1st Qu.:19.00   1st Qu.:23.50   1st Qu.:18.00   right:27   no :13
##  Median :20.00   Median :25.00   Median :29.00              yes:36
##  Mean   :20.35   Mean   :24.94   Mean   :29.49
##  3rd Qu.:21.90   3rd Qu.:26.00   3rd Qu.:36.00
##  Max.   :25.50   Max.   :30.00   Max.   :95.00
##                                  NA's   :1
##       dice           beans            handed
##  Min.   : 1.00   Min.   : 300.0   Min.   :-0.8200
##  1st Qu.: 2.00   1st Qu.: 642.5   1st Qu.: 0.6000
##  Median : 3.50   Median : 925.0   Median : 0.7950
##  Mean   : 5.08   Mean   : 977.2   Mean   : 0.6521
##  3rd Qu.: 6.75   3rd Qu.:1200.0   3rd Qu.: 0.8975
##  Max.   :22.00   Max.   :2600.0   Max.   : 1.0000
##                  NA's   :2
```

2. The summary Statistics by Group

Package psych can be used to generate summary statistics per group.

```
library(psych)
```

```
##
## Attaching package: 'psych'
##
## The following object is masked _by_ '.GlobalEnv':
##
##     SD
```

```
des <- describe.by(dat, group = dat$gender)
```

Package doBy, pastecs and Hmisc can do more, find more on this.

5.2 Frequencies and Crosstabs

We will consider ways to create frequency and contigency tables, tests of independence, measures of association and graphical representation of the results of categorical variables.

5.2.1 Generating Frequency Tables

The following methods can be used to generate frequency tables:

The table function

Three functions i.e., table() (for generating frequency tables), prop.table() (for generating tables of proportions) and margin.table( ) (for generating marginal frequencies).

```
# 2-Way Frequency Table
mytable <- table(dat$gender, dat$tongue)
mytable
```

```
##
##        no yes
##   F  1  4  14
##   M  0  9  22
```

```
margin.table(mytable, 1) # A frequencies (summed over gender)
```

```
##
##  F  M
## 19 31
```

```
margin.table(mytable, 2) # B frequencies (summed over tongue)
```

```
##
##      no yes
##   1  13  36
```

```
prop.table(mytable) # cell percentages
```

```
##
##           no  yes
##   F 0.02 0.08 0.28
##   M 0.00 0.18 0.44
```

```
prop.table(mytable, 1) # row percentages
```

```
##
##                        no        yes
##   F 0.05263158 0.21052632 0.73684211
##   M 0.00000000 0.29032258 0.70967742
```

```
prop.table(mytable, 2) # column percentages
```

```
##
##                      no       yes
##   F 1.0000000 0.3076923 0.3888889
##   M 0.0000000 0.6923077 0.6111111
```

ftable( ) can be used to print multidimensional tables (3 or more categorical variables) nicely.

```r
# 3-Way Frequency Table
mytable <- table(dat$gender, dat$tongue, dat$armcross)
ftable(mytable)
```

```
##        left right
##
## F         0     1
##   no      1     3
##   yes     8     6
## M         0     0
##   no      3     6
##   yes    11    11
```

The xtabs function

The xtabs( ) function allows you to create crosstabulations using formula style input.

```r
# 3-Way Frequency Table
mytable <- xtabs(~dat$gender+dat$tongue+dat$armcross, data=dat)

ftable(mytable) # print table
```

```
##                       dat$armcross left right
## dat$gender dat$tongue
## F                                     0     1
##            no                         1     3
##            yes                        8     6
## M                                     0     0
##            no                         3     6
##            yes                       11    11
```

```r
summary(mytable) # chi-square test of indepedence
```

```
## Call: xtabs(formula = ~dat$gender + dat$tongue + dat$armcross, data = dat)
## Number of cases in table: 50
## Number of factors: 3
## Test for independence of all factors:
##  Chisq = 6.254, df = 7, p-value = 0.5104
##  Chi-squared approximation may be incorrect
```

```r
mytable <- table(dat$gender, dat$tongue)
```

5.2.2 Generting Crosstables

The CrossTable( ) function in the gmodels package produces crosstabulations modeled after PROC FREQ in SAS or CROSSTABS in SPSS. It has a wealth of options.

```r
#install.packages("gmodels")
library(gmodels)
myctab <- CrossTable(dat$gender, dat$tongue, format = "SPSS")
```

```
##
##    Cell Contents
## |-------------------------|
## |                   Count |
## | Chi-square contribution |
## |             Row Percent |
## |          Column Percent |
## |           Total Percent |
## |-------------------------|
##
## Total Observations in Table:  50
##
##              | dat$tongue
##   dat$gender |           |       no  |      yes  | Row Total |
## -------------|-----------|-----------|-----------|-----------|
##            F |         1 |         4 |        14 |        19 |
##              |     1.012 |     0.179 |     0.007 |           |
##              |     5.263% |    21.053% |    73.684% |    38.000% |
##              |   100.000% |    30.769% |    38.889% |           |
##              |     2.000% |     8.000% |    28.000% |           |
## -------------|-----------|-----------|-----------|-----------|
##            M |         0 |         9 |        22 |        31 |
##              |     0.620 |     0.110 |     0.005 |           |
##              |     0.000% |    29.032% |    70.968% |    62.000% |
##              |     0.000% |    69.231% |    61.111% |           |
##              |     0.000% |    18.000% |    44.000% |           |
## -------------|-----------|-----------|-----------|-----------|
## Column Total |         1 |        13 |        36 |        50 |
##              |     2.000% |    26.000% |    72.000% |           |
## -------------|-----------|-----------|-----------|-----------|
##
##
```

```
myctab
```

```
## NULL
```

There are options to report percentages (row, column, cell), specify decimal places, produce Chi-square, Fisher, and McNemar tests of independence, report expected and residual values (pearson, standardized, adjusted standardized), include missing values as valid, annotate with row and column titles, and format as **SAS** or **SPSS** style output! See help(CrossTable) for details.

5.2.3 Test of Independence

The test for independence asserts that the row and column variables are independent of each other. The variables under consideration should be categorical variables. The hypothesis for the independence test is:

H_0: In the population, the two categorical variables are independent.
H_a: In the population, two categorical variables are dependent. The following test can be used to conduct test of independence.

1. Chi-Square Test

The function chisq.test() can be used to conduct this test in R. For 2-way tables you can use chisq.test(mytable) to test independence of the row and column variable. For example

```
chisq.test(mytable)
```

```
##
##  Pearson's Chi-squared test
##
## data:  mytable
## X-squared = 1.9321, df = 2, p-value = 0.3806
```

2. Fisher Exact Test

It is used in two norminal variables with a data set which can be represented in a row by column form. It highly recommended when the sample size is small. The function fisher.test()

```
fisher.test(mytable)
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  mytable
## p-value = 0.4802
## alternative hypothesis: two.sided
```

3. Mantel-Haenszel Comparison

It is mostly used when controlling two dichotomous viables while adjusting for an addition one. It helps determine whether the additional variable affects the relationship between the two. The hypothesis for this test in defined as follows:

H_0: There is no relationship between the two variables of interest when controlling for a third variable.
H_a : There is a relationship between the two variables of interest when controlling for a third variable.

```
mytable <- table(dat$gender, dat$tongue, dat$armcross)
mantelhaen.test(mytable)
```

```
##
##  Cochran-Mantel-Haenszel test
##
## data:  mytable
## Cochran-Mantel-Haenszel M^2 = 1.9061, df = 2, p-value = 0.3856
```

5.3 Correlation

Correlation measures the association between two variables. It measures the strength of linear association between the two variables. The variables should be quantative. The correlattion coefficient is between -1 and 1 and its denoted by ρ or r. For example the income of people and the type of the car they drive, relationship between the sales and cost of advertisement. The hypothesis definition for correlation coefficient is as follows:

H_0: $\rho = 0$ (there is no linear relationship between the two variables).
H_a : $\rho \neq 0$ (there is a linear relationship between the two variables).

In R, correlation can be obtained using the function cor() and the corresponding covariances can be obtained using cov(), type help on this functions.

Unfortunately, cor() does not give significance test. However, rcorr() function from package can be used to produce pearson and spearman correlations. The input must be as.matrix.

```
#install.packages("xtable")
library(xtable)

data("mtcars")
cortest <- cor(mtcars)

USAdata <- read.csv("/media/cygu/Disk/Works/UON/Datasets/Correlation.csv",
                    header = T)
names(USAdata)
```

```
##  [1] "State"                        "state_code"
##  [3] "spss_GS"                      "regression_GS"
##  [5] "statistically_significant_GS" "business_intelligence_GS"
##  [7] "data_visualization_GS"        "totally_lost_GS"
##  [9] "facebook_GS"                  "american_idol_GS"
## [11] "modern_dance_GS"              "nba_GS"
## [13] "nfl_GS"                       "fifa_GS"
## [15] "has_nfl"                      "has_nba"
## [17] "has_mls"                      "degree"
## [19] "statistics_ed"                "age"
## [21] "region"                       "division"
## [23] "Northeast"                    "Midwest"
## [25] "South"
```

```
attach(USAdata)

NewUSAdata <- data.frame(facebook_GS, spss_GS, regression_GS,
                         statistically_significant_GS, modern_dance_GS,
                         american_idol_GS, modern_dance_GS, region)

USAcor <- cor(NewUSAdata)
tab1 <- xtable(USAcor)
print(tab1, file = "tab1.tex", table.placement = "h")
```

```
#install.packages("Hmisc")
#library(Hmisc)
#cortest <- rcorr(as.matrix(mtcars))
```

	facebook_GS	spss_GS	regression_GS	statistically_significant_GS	modern_dance_GS	american_idol_GS	modern_dance_GS.1	region
facebook_GS	1.00	-0.18	-0.30	-0.35	-0.30	0.52	-0.30	-0.20
spss_GS	-0.18	1.00	0.82	0.77	0.14	-0.07	0.14	-0.26
regression_GS	-0.30	0.82	1.00	0.87	0.24	-0.20	0.24	-0.32
statistically_significant_GS	-0.35	0.77	0.87	1.00	0.32	-0.24	0.32	-0.30
modern_dance_GS	-0.30	0.14	0.24	0.32	1.00	0.00	1.00	0.14
american_idol_GS	0.52	-0.07	-0.20	-0.24	0.00	1.00	0.00	-0.07
modern_dance_GS.1	-0.30	0.14	0.24	0.32	1.00	0.00	1.00	0.14
region	-0.20	-0.26	-0.32	-0.30	0.14	-0.07	0.14	1.00

5.4 t-Test: Comparing One or Two Means

This is also referred to as Student's t-test, named after a statistician William Gosset. Mostly, this test deals with comparing means.

5.4.1 One-sample t-test

It compares a single mean to a fixed one, sometimes considered as "gold standard". The fixed mean value is always known and one-sample t-test determines whether we have sufficient evidence to conclude that this value is different from the mean of the population from which the sample is drawn. The hypothesis definition for correlation coefficient is as follows:

H_0: The population mean is equal to the fixed value.
H_a : The population mean is not equal to the specified value.

This test would be appropriate in the following examples:

1. Does the average volume of liquid in filled soft drink bottles match the 12 ounces advertised on the label?
2. Based on a random sample of 200 students, can we conclude that the averageSAT score this year is lower than the national average from 3 years ago?

In *R*, one-sample t-test can be done as follows:

```
# Testing whether the mean age is different from 20 years
t.test(dat$years, mu=20)
```

```
##
##  One Sample t-test
##
## data:  dat$years
## t = 16.581, df = 49, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 20
## 95 percent confidence interval:
##  25.14977 26.57023
## sample estimates:
## mean of x
##     25.86
```

5.4.2 Two-sample t-test

Two-sample t-test is used to compare means of two populations based on independent samples. It tests whether the two means are different or not. Suppose we denote mean of the population from which the sample is drawn as μ and denote the hypothesized value for the population mean as μ_0, then the hypothesis for this kind of test is defined as follow:

H_0: $\mu = \mu_0$ (in words: the population mean is equal to the hypothesized value μ_0).
H_a : $\mu \neq \mu_0$ (the population mean is not equal to μ_0).

The hypothesis can also be One-Tailed.

Examples in R

A lecturer wishes to know whether students given open-ended questions performs better than those given choices questions. The students are taught by the same lecturer under the same conditions. He randomly selects 7 students and gives them open ended questions and also gives choices questions to 6 other students. The results are shown below.

```
OpenEnded <- c(51, 53.3, 55.6, 51, 55.5, 53, 52.1)
Choices <- c(54, 56.1, 52.1, 56.4, 52.9)

# independent 2-Sample t-test
t.test(OpenEnded, Choices) #
```

```
##
##  Welch Two Sample t-test
##
## data:  OpenEnded and Choices
## t = -1.0998, df = 8.78, p-value = 0.3006
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -3.765216  1.308073
## sample estimates:
## mean of x mean of y
##  53.07143  54.30000
```

Suppose in our data (dat), we want to determine whether Males or Females have longer foots, then we would proceed as follows:

```
#Selecting complete data

Newdat <- na.omit(dat)
#Selecting male foot length

MaleFootLen <- Newdat$footlength[Newdat$gender=="M"]

FemaleFootLen <- Newdat$footlength[Newdat$gender=="F"]

t.test(MaleFootLen, FemaleFootLen)
```

```
##
##  Welch Two Sample t-test
##
## data:  MaleFootLen and FemaleFootLen
## t = 1.6751, df = 41.357, p-value = 0.1015
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.2881852  3.0960013
## sample estimates:
## mean of x mean of y
##  25.51724  24.11333
```

5.4.3 Paired t-test

Paired t-test is used to compare two means based on samples that are paired in some way. This can be applied in cases where:

1. Measurements on the same subject taken before and after experiment.
2. Subjects in treatment groups compared to those in control group.
3. Two measurements taken on the same subject,

The hypothesis specification for Paired t-test is similar to that of Two-sample t-test.

Examples in R

Suppose we want to test whether a diet is effective in weight loss. The "before" and "after" weights for 15 randomly selected participants are recorded. Suppose we define the difference in weights "before" and "after" the excercise as d_i, then the hypotheses is defined as:

H_0: $\mu_d = 0$ (the mean of the differences is zero; i.e., the diet is ineffective).
H_a : $\mu_d > 0$ (the mean of the differences is positive; i.e., the diet is effective).

We would proceed as follows in R:

```
Wbefore <- c(210, 207, 183, 195, 187, 201, 158, 180, 173, 198,
             225, 243, 168, 177, 196)
Wafter <- c(204, 205, 182, 196, 177, 193, 152, 182, 165, 186,
            218, 237, 174, 178, 199)

#Paired t-test

t.test(Wbefore, Wafter, paired = TRUE)
```

```
##
##  Paired t-test
##
## data:  Wbefore and Wafter
## t = 2.5674, df = 14, p-value = 0.02235
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.5816453 6.4850214
## sample estimates:
## mean of the differences
##                3.533333
```

5.5 Regression Analysis

Regression Analysis examines the relationship between variables. There are two types of regression, namely: Simple linear regression which examines the relationship between a single predictor (independent) variable and a quantitative response (dependent) variable and Multiple linear regression which examines the relationship between two or more predictor variables with a response variable. The hypothesis for simple linear regression is defined as

H_0: $\beta_0 = 0$ (the slope is zero; there is no linear relationship between the variables).
H_a : $\beta_0 \neq 0$ (the slope is not zero; there is a linear relationship between the variables).

while for multiple linear regression, the hypothesis is defined as

H_0: $\beta_i = 0$ (At least one of the β_i is zero).
H_a : $\beta_i \neq 0$ (None of the β_i is zero).

5.5.1 Simple linear regression

The function lm() can be used to conduct regression anlysis in r. See help(lm)

Suppose we want to know whether there is a relationship between those who search for SPSS and those who search for regression in Google in the USA data above, we can conduct regression analysis as follows:

```
slr <- lm(spss_GS ~ regression_GS)
summary(slr)
```

```
##
## Call:
## lm(formula = spss_GS ~ regression_GS)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.4508 -0.3171 -0.1189  0.2337  1.7992
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -3.563e-05  8.151e-02   0.000        1
## regression_GS  8.173e-01  8.233e-02   9.928 2.56e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5821 on 49 degrees of freedom
## Multiple R-squared:  0.6679, Adjusted R-squared:  0.6612
## F-statistic: 98.57 on 1 and 49 DF,  p-value: 2.556e-13
```

```
names(slr)
```

```
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
```

We can view additional objects in lm() by typying names(slr) and the call them using "slr$coefficients" or use ( ), for example:

```
# Other useful functions
#coefficients(slr) # model coefficients
# confint(slr, level=0.95) # CIs for model parameters
# fitted(slr) # predicted values
# residuals(slr) # residuals
# anova(slr) # anova table
# vcov(slr) # covariance matrix for model parameters
# influence(slr) # regression diagnostics
```

5.5.2 Multiple (Linear) Regression

Multiple linear regression extends the simple linear regression. Multiple linear regression has one dependent variable, (say Y) and k independent variables (say $X_i, i = 1, \cdots, k$). The dependent variable is quantitative while independent variable may be binary/indicator $(0, 1)$.

Suppose we want to know whether there is a relationship between those who search for SPSS and those who search for regression, statistical significance, facebook, business intelligence, modern dance, has nfl and has mls in Google in the USA data above, we can conduct regression analysis as follows:

```
detach(USAdata)
USAdata2 <- read.csv("/media/cygu/Disk/Works/UON/Datasets/Multiple-reg.csv",
                     header = T)
names(USAdata2)
```

```
##  [1] "State"                          "state_code"
##  [3] "spss_GS"                        "regression_GS"
##  [5] "statistically_significant_GS" "business_intelligence_GS"
##  [7] "data_visualization_GS"          "totally_lost_GS"
##  [9] "facebook_GS"                    "american_idol_GS"
## [11] "modern_dance_GS"                "nba_GS"
## [13] "nfl_GS"                         "fifa_GS"
## [15] "has_nfl"                        "has_nba"
## [17] "has_mls"                        "degree"
## [19] "statistics_ed"                  "age"
## [21] "region"                         "division"
## [23] "Northeast"                      "Midwest"
## [25] "South"
```

```
attach(USAdata2)

mlr <- lm(spss_GS ~ regression_GS + statistically_significant_GS + facebook_GS +
            business_intelligence_GS + modern_dance_GS + has_nfl + has_mls)
summary(mlr)
```

```
##
## Call:
## lm(formula = spss_GS ~ regression_GS + statistically_significant_GS +
##     facebook_GS + business_intelligence_GS + modern_dance_GS +
##     has_nfl + has_mls)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.38847 -0.32919 -0.07651  0.24353  1.60422
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 -0.02325    0.13293  -0.175  0.86198
## regression_GS                0.56084    0.17948   3.125  0.00318 **
## statistically_significant_GS  0.32097    0.17608   1.823  0.07529 .
## facebook_GS                  0.08861    0.12216   0.725  0.47217
## business_intelligence_GS     0.01415    0.15102   0.094  0.92578
## modern_dance_GS             -0.06318    0.09704  -0.651  0.51848
```

```
## has_nfl                       0.08694    0.20173   0.431  0.66866
## has_mls                      -0.05604    0.24035  -0.233  0.81674
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5924 on 43 degrees of freedom
## Multiple R-squared:  0.6982, Adjusted R-squared:  0.649
## F-statistic: 14.21 on 7 and 43 DF,  p-value: 2.188e-09
```