PS792 Chapter 3 Basic Statistics

Jixiang Wu, Associate Professor of Quantitative Genetics/Biostatistics, South Dakota State University

January 25, 2018

In this R document, we will learn how to use R to run some basic statistics analysis. The data sets that will be used in this R document are mtcars, Arthritis, and UScrime. The first 6 rows of mtcars data are showed as follows.

```
data(mtcars)
head(mtcars)
```

```
##                    mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

Descriptive statistics

Before running some descriptive statistics, we will substract three columns of data from the data frame mtcars. The three variables are mpg, hp, and wt as demonstrated as follows. The three columns of data are assigned to newdata, which is also a data frame.

```
myvars=c("mpg","hp","wt")
newdata=mtcars[myvars]
```

Using summary function you can have some descriptive statistics for each of three variables in the data frame newdata.

```
summary(newdata)
```

```
##       mpg              hp              wt
##  Min.   :10.40   Min.   : 52.0   Min.   :1.513
##  1st Qu.:15.43   1st Qu.: 96.5   1st Qu.:2.581
##  Median :19.20   Median :123.0   Median :3.325
##  Mean   :20.09   Mean   :146.7   Mean   :3.217
##  3rd Qu.:22.80   3rd Qu.:180.0   3rd Qu.:3.610
##  Max.   :33.90   Max.   :335.0   Max.   :5.424
```

We may add more statistics for a variable such as standard deviation, skewness and kurtosis. The following R function mystats show the details that which parameters are added. Then you can use

```
mystats <- function(x, na.omit=FALSE){
  if (na.omit)
  x <- x[!is.na(x)]
  m <- mean(x)
  n <- length(x)
  s <- sd(x)
  skew <- sum((x-m)^3/s^3)/n
  kurt <- sum((x-m)^4/s^4)/n - 3
  return(c(n=n, mean=m, stdev=s, skew=skew, kurtosis=kurt))
}

#sapply(newdata,mystats)
```

With mystats function available, we can run one of variables pmg, hp, and wt in the data frame newdata as demonstrated as follows.

```
pmg.stat=with(newdata,mystats(mpg))

pmg.stat

##         n      mean     stdev      skew  kurtosis
## 32.000000 20.090625  6.026948  0.610655 -0.372766

hp.stat=with(newdata,mystats(hp))

hp.stat

##           n        mean       stdev        skew    kurtosis
##  32.0000000 146.6875000  68.5628685   0.7260237  -0.1355511

wt.stat=with(newdata,mystats(wt))
wt.stat

##           n        mean       stdev        skew    kurtosis
## 32.00000000  3.21725000  0.97845744  0.42314646 -0.02271075
```

We can combine the destriptive statistics for these three variables using dataframe funtion to make a data frame.

```
cars.stat=data.frame(pmg=pmg.stat,hp=hp.stat,wt=wt.stat)
cars.stat

##                 pmg          hp          wt
## n         32.000000  32.0000000 32.00000000
## mean      20.090625 146.6875000  3.21725000
## stdev      6.026948  68.5628685  0.97845744
## skew       0.610655   0.7260237  0.42314646
## kurtosis  -0.372766  -0.1355511 -0.02271075
```

If you have a large number of variables to be processed, then you may use loop to repeat the process.

```
wt.stat=with(newdata,mystats(wt))
nr=length(wt.stat)
nc=ncol(newdata)
STAT=matrix(0,nr,nc)
for(h in 1:nc){
  stat=mystats(newdata[,h])
  STAT[,h]=stat
}
colnames(STAT)=colnames(newdata)
rownames(STAT)=names(stat)

data.frame(STAT)

##                 mpg          hp          wt
## n         32.000000  32.0000000 32.00000000
## mean      20.090625 146.6875000  3.21725000
## stdev      6.026948  68.5628685  0.97845744
## skew       0.610655   0.7260237  0.42314646
## kurtosis -0.372766  -0.1355511 -0.02271075
```

You may also convert the above code as another R function MYSTAT.

```
MYSTAT=function(data){
  cn=ncol(data)
  stat=mystats(data[,1])
  rn=length(stat)
  STAT=matrix(0,rn,cn)
  for(h in 1:cn){
    stat=mystats(data[,h])
    STAT[,h]=stat
  }
  colnames(STAT)=colnames(data)
  rownames(STAT)=names(stat)
  STAT=data.frame(STAT)

  return(STAT)
}

stat=MYSTAT(newdata)
stat

##                 mpg          hp          wt
## n         32.000000  32.0000000 32.00000000
## mean      20.090625 146.6875000  3.21725000
## stdev      6.026948  68.5628685  0.97845744
## skew       0.610655   0.7260237  0.42314646
## kurtosis -0.372766  -0.1355511 -0.02271075
```

More methods

There are a couple of other functions packed in different R packages. For example, the describe function in the R package Hmisc, stat.desc in package pastecs, describe in package psych. It is very important to install these packages before using it. For detailed information, please refer to these functions in corresponding R packages by yourselves.

```
require(Hmisc)

## Loading required package: Hmisc

## Loading required package: lattice

## Loading required package: survival

## Loading required package: Formula

## Loading required package: ggplot2

##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:base':
##
##     format.pval, units

describe(newdata)

## newdata
##
##  3  Variables      32  Observations
## -------------------------------------------------------------------------
-
## mpg
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##       32        0       25    0.999    20.09    6.796    12.00    14.34
##      .25      .50      .75      .90      .95
##    15.43    19.20    22.80    30.09    31.30
##
## lowest : 10.4 13.3 14.3 14.7 15.0, highest: 26.0 27.3 30.4 32.4 33.9
## -------------------------------------------------------------------------
-
## hp
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##       32        0       22    0.997    146.7    77.04    63.65    66.00
##      .25      .50      .75      .90      .95
##    96.50   123.00   180.00   243.50   253.55
##
## lowest :  52  62  65  66  91, highest: 215 230 245 264 335
## -------------------------------------------------------------------------
-
## wt
```

```
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##       32        0       29    0.999    3.217    1.089    1.736    1.956
##      .25      .50      .75      .90      .95
##    2.581    3.325    3.610    4.048    5.293
##
## lowest : 1.513 1.615 1.835 1.935 2.140, highest: 3.845 4.070 5.250 5.345
5.424
## ----------------------------------------------------------------------------
-
```

```
library(pastecs)
```

```
## Loading required package: boot
```

```
##
## Attaching package: 'boot'
```

```
## The following object is masked from 'package:survival':
##
##     aml
```

```
## The following object is masked from 'package:lattice':
##
##     melanoma
```

```
stat.desc(newdata)
```

```
##                      mpg           hp          wt
## nbr.val       32.0000000   32.0000000  32.0000000
## nbr.null       0.0000000    0.0000000   0.0000000
## nbr.na         0.0000000    0.0000000   0.0000000
## min           10.4000000   52.0000000   1.5130000
## max           33.9000000  335.0000000   5.4240000
## range         23.5000000  283.0000000   3.9110000
## sum          642.9000000 4694.0000000 102.9520000
## median        19.2000000  123.0000000   3.3250000
## mean          20.0906250  146.6875000   3.2172500
## SE.mean        1.0654240   12.1203173   0.1729685
## CI.mean.0.95   2.1729465   24.7195501   0.3527715
## var           36.3241028 4700.8669355   0.9573790
## std.dev        6.0269481   68.5628685   0.9784574
## coef.var       0.2999881    0.4674077   0.3041285
```

```
library(psych)
```

```
##
## Attaching package: 'psych'
```

```
## The following object is masked from 'package:boot':
##
##     logit
```

```
## The following object is masked from 'package:Hmisc':
##
##     describe

## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
describe(newdata)
```

```
##     vars  n   mean    sd median trimmed   mad   min    max  range skew
## mpg    1 32  20.09  6.03  19.20   19.70  5.41 10.40  33.90  23.50 0.61
## hp     2 32 146.69 68.56 123.00  141.19 77.10 52.00 335.00 283.00 0.73
## wt     3 32   3.22  0.98   3.33    3.15  0.77  1.51   5.42   3.91 0.42
##     kurtosis    se
## mpg    -0.37  1.07
## hp     -0.14 12.12
## wt     -0.02  0.17
```

Descriptive statistics by group

```
#aggregate(newdata, by=list(am=mtcars$am), mean)

dstat=function(x)sapply(x,mystats)
by(newdata,mtcars$am,dstat)
```

```
## mtcars$am: 0
##                  mpg          hp        wt
## n        19.00000000  19.00000000 19.0000000
## mean     17.14736842 160.26315789  3.7688947
## stdev     3.83396639  53.90819573  0.7774001
## skew      0.01395038  -0.01422519  0.9759294
## kurtosis -0.80317826  -1.20969733  0.1415676
## --------------------------------------------------------
## mtcars$am: 1
##                  mpg         hp        wt
## n        13.00000000  13.0000000 13.0000000
## mean     24.39230769 126.8461538  2.4110000
## stdev     6.16650381  84.0623243  0.6169816
## skew      0.05256118   1.3598859  0.2103128
## kurtosis -1.45535200   0.5634635 -1.1737358
```

Additional methods by group

```
require(doBy)
```

```
## Loading required package: doBy
```

```
summaryBy(mpg+hp+wt~am,data=mtcars,FUN=mystats)
```

```
##   am mpg.n mpg.mean mpg.stdev   mpg.skew mpg.kurtosis hp.n  hp.mean
## 1  0    19 17.14737  3.833966 0.01395038   -0.8031783   19 160.2632
## 2  1    13 24.39231  6.166504 0.05256118   -1.4553520   13 126.8462
```

```
##   hp.stdev     hp.skew hp.kurtosis wt.n  wt.mean  wt.stdev   wt.skew
## 1 53.90820 -0.01422519  -1.2096973   19 3.768895 0.7774001 0.9759294
## 2 84.06232  1.35988586   0.5634635   13 2.411000 0.6169816 0.2103128
##   wt.kurtosis
## 1   0.1415676
## 2  -1.1737358
```

```
require(psych)
describeBy(newdata,list(am=mtcars$am))
```

```
## 
##  Descriptive statistics by group 
## am: 0
##     vars  n   mean    sd median trimmed   mad   min    max  range  skew
## mpg    1 19  17.15  3.83  17.30   17.12  3.11 10.40  24.40  14.00  0.01
## hp     2 19 160.26 53.91 175.00  161.06 77.10 62.00 245.00 183.00 -0.01
## wt     3 19   3.77  0.78   3.52    3.75  0.45  2.46   5.42   2.96  0.98
##     kurtosis    se
## mpg    -0.80  0.88
## hp     -1.21 12.37
## wt      0.14  0.18
## -------------------------------------------------------- 
## am: 1
##     vars  n   mean    sd median trimmed   mad   min    max  range skew
## mpg    1 13  24.39  6.17  22.80   24.38  6.67 15.00  33.90  18.90 0.05
## hp     2 13 126.85 84.06 109.00  114.73 63.75 52.00 335.00 283.00 1.36
## wt     3 13   2.41  0.62   2.32    2.39  0.68  1.51   3.57   2.06 0.21
##     kurtosis    se
## mpg    -1.46  1.71
## hp      0.56 23.31
## wt     -1.17  0.17
```

Frequency and contingency tables

```
require(vcd)
```

```
## Loading required package: vcd
```

```
## Loading required package: grid
```

```
head(Arthritis)
```

```
##   ID Treatment  Sex Age Improved
## 1 57   Treated Male  27     Some
## 2 46   Treated Male  29     None
## 3 77   Treated Male  30     None
## 4 17   Treated Male  32   Marked
## 5 36   Treated Male  46   Marked
## 6 23   Treated Male  58   Marked
```

```
names(Arthritis)
```

```
## [1] "ID"        "Treatment" "Sex"       "Age"       "Improved"
```

```
dim(Arthritis)
```

```
## [1] 84  5
```

```
str(Arthritis)
```

```
## 'data.frame':    84 obs. of  5 variables:
##  $ ID       : int  57 46 77 17 36 23 75 39 33 55 ...
##  $ Treatment: Factor w/ 2 levels "Placebo","Treated": 2 2 2 2 2 2 2 2 2 2
...
##  $ Sex      : Factor w/ 2 levels "Female","Male": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Age      : int  27 29 30 32 46 58 59 59 63 63 ...
##  $ Improved : Ord.factor w/ 3 levels "None"<"Some"<..: 2 1 1 3 3 3 1 3 1 1
...
```

One-way tables

Use table function can make an one-way table easily.

```
mytable = with(Arthritis,table(Improved))
mytable
```

```
## Improved
##   None   Some Marked 
##     42     14     28
```

```
mytable=table(Arthritis$Improved)
mytable
```

```
## 
##   None   Some Marked 
##     42     14     28
```

We can use prop.table to convert the count table generated from table function.

```
ptable=prop.table(mytable)*100
ptable
```

```
## 
##     None     Some   Marked 
## 50.00000 16.66667 33.33333
```

Two-way tables

Table function can be used to generate a two-way table and to generate a corresponding two-way proportional table. For example

```
mytable=with(Arthritis,table(Treatment,Improved))
mytable
```

```
##           Improved
## Treatment None Some Marked
##   Placebo   29    7      7
##   Treated   13    7     21
```

```r
prop.table(mytable)
```

```
##           Improved
## Treatment       None       Some     Marked
##   Placebo 0.34523810 0.08333333 0.08333333
##   Treated 0.15476190 0.08333333 0.25000000
```

We can use margin.table function to generate a margin table for each row or column.

```r
margin.table(mytable,1)
```

```
## Treatment
## Placebo Treated
##      43      41
```

```r
margin.table(mytable,2)
```

```
## Improved
##   None   Some Marked
##     42     14     28
```

```r
prop.table(mytable,1)
```

```
##           Improved
## Treatment      None      Some    Marked
##   Placebo 0.6744186 0.1627907 0.1627907
##   Treated 0.3170732 0.1707317 0.5121951
```

```r
prop.table(mytable,2)
```

```
##           Improved
## Treatment      None      Some    Marked
##   Placebo 0.6904762 0.5000000 0.2500000
##   Treated 0.3095238 0.5000000 0.7500000
```

```r
addmargins(prop.table(mytable,2),1)
```

```
##           Improved
## Treatment      None      Some    Marked
##   Placebo 0.6904762 0.5000000 0.2500000
##   Treated 0.3095238 0.5000000 0.7500000
##   Sum     1.0000000 1.0000000 1.0000000
```

Another method for creating two-way tables is the CrossTable() function on the gmodels package. The CrossTable() function produces two-ways tables modeled after PROC FREQ in SAS or CROSSTABS in SPSS. The following listing showed an example.

```r
require(gmodels)
```

```
## Loading required package: gmodels
```

```
mytable=with(Arthritis, CrossTable(Treatment,Improved))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## | Chi-square contribution |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  84
##
##
##              | Improved
##    Treatment |       None |       Some |     Marked |  Row Total |
## -------------|------------|------------|------------|------------|
##      Placebo |         29 |          7 |          7 |         43 |
##              |      2.616 |      0.004 |      3.752 |            |
##              |      0.674 |      0.163 |      0.163 |      0.512 |
##              |      0.690 |      0.500 |      0.250 |            |
##              |      0.345 |      0.083 |      0.083 |            |
## -------------|------------|------------|------------|------------|
##      Treated |         13 |          7 |         21 |         41 |
##              |      2.744 |      0.004 |      3.935 |            |
##              |      0.317 |      0.171 |      0.512 |      0.488 |
##              |      0.310 |      0.500 |      0.750 |            |
##              |      0.155 |      0.083 |      0.250 |            |
## -------------|------------|------------|------------|------------|
## Column Total |         42 |         14 |         28 |         84 |
##              |      0.500 |      0.167 |      0.333 |            |
## -------------|------------|------------|------------|------------|
##
##
```

```
mytable
```

```
## $t
##          y
## x          None Some Marked
##   Placebo   29    7      7
##   Treated   13    7     21
##
## $prop.row
##          y
## x               None      Some    Marked
```

```
##   Placebo 0.6744186 0.1627907 0.1627907
##   Treated 0.3170732 0.1707317 0.5121951
##
## $prop.col
##          y
## x              None      Some    Marked
##   Placebo 0.6904762 0.5000000 0.2500000
##   Treated 0.3095238 0.5000000 0.7500000
##
## $prop.tbl
##          y
## x               None       Some     Marked
##   Placebo 0.34523810 0.08333333 0.08333333
##   Treated 0.15476190 0.08333333 0.25000000
```

Tests of Independence

R provides several methods of testing the independence of categorical variables. The three tests described in this section are the chi-square test of independence, the Fisher exact test, and the Cochran-Mantel-Haenszel test.

Chi-square test

```
mytable=with(Arthritis, table(Treatment,Improved))
chisq.test(mytable)
```

```
##
##  Pearson's Chi-squared test
##
## data:  mytable
## X-squared = 13.055, df = 2, p-value = 0.001463
```

```
mytable=with(Arthritis, table(Treatment,Sex))
chisq.test(mytable)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  mytable
## X-squared = 0.38378, df = 1, p-value = 0.5356
```

Fisher’s exact test

```
mytable=with(Arthritis, table(Treatment,Improved))
fisher.test(mytable)
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  mytable
## p-value = 0.001393
## alternative hypothesis: two.sided
```

```r
mytable=with(Arthritis, table(Treatment,Sex))
fisher.test(mytable)

##
##  Fisher's Exact Test for Count Data
##
## data:  mytable
## p-value = 0.4763
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##  0.5320442 4.3286798
## sample estimates:
## odds ratio
##   1.500984
```

COCHRAN-MANTEL-HAENSZEL test

```r
mytable=with(Arthritis, table(Treatment,Improved,Sex))
mantelhaen.test(mytable)

##
##  Cochran-Mantel-Haenszel test
##
## data:  mytable
## Cochran-Mantel-Haenszel M^2 = 14.632, df = 2, p-value = 0.0006647
```

Measures of association

```r
require(vcd)
mytable=with(Arthritis, table(Treatment,Improved))
assocstats(mytable)

##                     X^2 df  P(> X^2)
## Likelihood Ratio 13.530  2 0.0011536
## Pearson          13.055  2 0.0014626
##
## Phi-Coefficient   : NA
## Contingency Coeff.: 0.367
## Cramer's V        : 0.394
```

Covariance and Correlation

```r
states=state.x77[,1:6]

cov(states)

##               Population      Income   Illiteracy     Life Exp      Murder
## Population 19931683.7588 571229.7796  292.8679592 -407.8424612 5663.523714
## Income       571229.7796 377573.3061 -163.7020408  280.6631837 -521.894286
## Illiteracy      292.8680   -163.7020    0.3715306   -0.4815122    1.581776
## Life Exp       -407.8425    280.6632   -0.4815122    1.8020204   -3.869480
## Murder         5663.5237   -521.8943    1.5817755   -3.8694804   13.627465
## HS Grad       -3551.5096   3076.7690   -3.2354694    6.3126849  -14.549616
```

```
##                  HS Grad
## Population -3551.509551
## Income      3076.768980
## Illiteracy    -3.235469
## Life Exp       6.312685
## Murder       -14.549616
## HS Grad       65.237894
```

```r
cor(states)
```

```
##             Population     Income Illiteracy    Life Exp     Murder
## Population  1.00000000  0.2082276  0.1076224 -0.06805195  0.3436428
## Income      0.20822756  1.0000000 -0.4370752  0.34025534 -0.2300776
## Illiteracy  0.10762237 -0.4370752  1.0000000 -0.58847793  0.7029752
## Life Exp   -0.06805195  0.3402553 -0.5884779  1.00000000 -0.7808458
## Murder      0.34364275 -0.2300776  0.7029752 -0.78084575  1.0000000
## HS Grad    -0.09848975  0.6199323 -0.6571886  0.58221620 -0.4879710
##                HS Grad
## Population -0.09848975
## Income      0.61993232
## Illiteracy -0.65718861
## Life Exp    0.58221620
## Murder     -0.48797102
## HS Grad     1.00000000
```

```r
cor(states, method="spearman")
```

```
##            Population     Income Illiteracy   Life Exp     Murder
## Population  1.0000000  0.1246098  0.3130496 -0.1040171  0.3457401
## Income      0.1246098  1.0000000 -0.3145948  0.3241050 -0.2174623
## Illiteracy  0.3130496 -0.3145948  1.0000000 -0.5553735  0.6723592
## Life Exp   -0.1040171  0.3241050 -0.5553735  1.0000000 -0.7802406
## Murder      0.3457401 -0.2174623  0.6723592 -0.7802406  1.0000000
## HS Grad    -0.3833649  0.5104809 -0.6545396  0.5239410 -0.4367330
##               HS Grad
## Population -0.3833649
## Income      0.5104809
## Illiteracy -0.6545396
## Life Exp    0.5239410
## Murder     -0.4367330
## HS Grad     1.0000000
```

```r
x = states[,c("Population", "Income", "Illiteracy", "HS Grad")]
y = states[,c("Life Exp", "Murder")]

cor(x,y)
```

```
##               Life Exp     Murder
## Population -0.06805195  0.3436428
## Income      0.34025534 -0.2300776
```

```
## Illiteracy -0.58847793  0.7029752
## HS Grad     0.58221620 -0.4879710
```

```
cor.test(states[,1],states[,3])
```

```
##
##  Pearson's product-moment correlation
##
## data:  states[, 1] and states[, 3]
## t = 0.74999, df = 48, p-value = 0.4569
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.1759976  0.3747441
## sample estimates:
##       cor
## 0.1076224
```

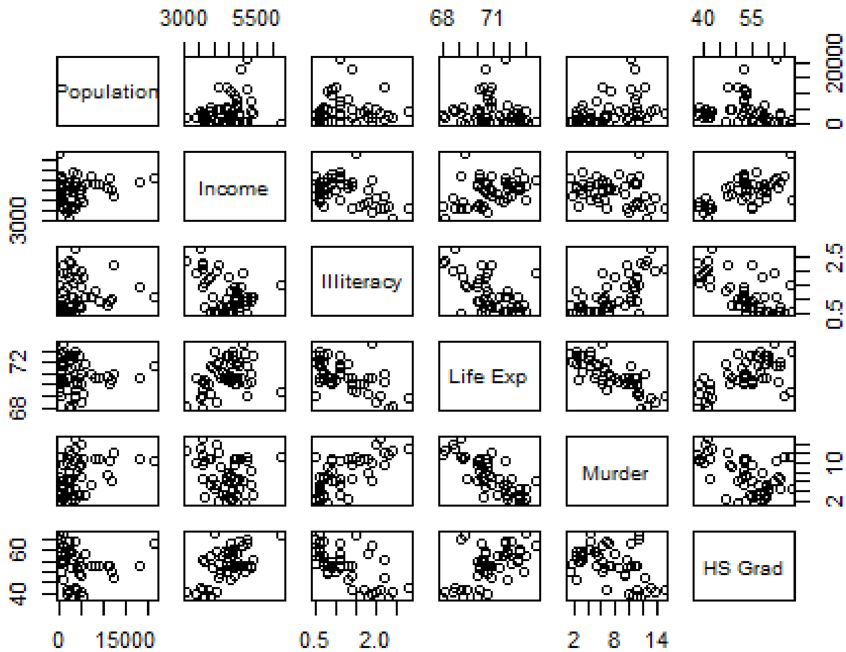
```
require(psych)
corr.test(states,use="complete")
```

```
## Call:corr.test(x = states, use = "complete")
## Correlation matrix
##             Population Income Illiteracy Life Exp Murder HS Grad
## Population       1.00   0.21       0.11    -0.07   0.34   -0.10
## Income           0.21   1.00      -0.44     0.34  -0.23    0.62
## Illiteracy       0.11  -0.44       1.00    -0.59   0.70   -0.66
## Life Exp        -0.07   0.34      -0.59     1.00  -0.78    0.58
## Murder           0.34  -0.23       0.70    -0.78   1.00   -0.49
## HS Grad         -0.10   0.62      -0.66     0.58  -0.49    1.00
## Sample Size
## [1] 50
## Probability values (Entries above the diagonal are adjusted for multiple
tests.)
##             Population Income Illiteracy Life Exp Murder HS Grad
## Population       0.00   0.59       1.00      1.0   0.10       1
## Income           0.15   0.00       0.01      0.1   0.54       0
## Illiteracy       0.46   0.00       0.00      0.0   0.00       0
## Life Exp         0.64   0.02       0.00      0.0   0.00       0
## Murder           0.01   0.11       0.00      0.0   0.00       0
## HS Grad          0.50   0.00       0.00      0.0   0.00       0
##
##  To see confidence intervals of the correlations, print with the
short=FALSE option
```

Visualizing correlations

We may use the pairs function to visualize the correlations among different examples

```
pairs(states)
```



t-tests

The most common activity in research is the comparison of two groups. There are two types of t-tests. The 1st one is independent t-test and the second one is paired t-test.

```
require(MASS)

## Loading required package: MASS

t.test(Prob~So, data=UScrime)

##
##  Welch Two Sample t-test
##
## data:  Prob by So
## t = -3.8954, df = 24.925, p-value = 0.0006506
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.03852569 -0.01187439
## sample estimates:
## mean in group 0 mean in group 1
##      0.03851265      0.06371269

Input = ("
Time     Student  Score
          Before  a          65
          Before  b          75
```

```
        Before  c         86
        Before  d         69
        Before  e         60
        Before  f         81
        Before  g         88
        Before  h         53
        Before  i         75
        Before  j         73
        After   a         77
        After   b         98
        After   c         92
        After   d         77
        After   e         65
        After   f         77
        After   g        100
        After   h         73
        After   i         93
        After   j         75
        ")

dat1 = read.table(textConnection(Input),header=TRUE)
dat1=with(dat1,dat1[order(Time,Student),])

library(psych)

headTail(dat1)

##       Time Student Score
## 11   After       a    77
## 12   After       b    98
## 13   After       c    92
## 14   After       d    77
## ...   <NA>    <NA>   ...
## 7   Before       g    88
## 8   Before       h    53
## 9   Before       i    75
## 10  Before       j    73

str(dat1)

## 'data.frame':    20 obs. of  3 variables:
##  $ Time   : Factor w/ 2 levels "After","Before": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Student: Factor w/ 10 levels "a","b","c","d",..: 1 2 3 4 5 6 7 8 9 10
...
##  $ Score  : int  77 98 92 77 65 77 100 73 93 75 ...

summary(dat1)

##      Time        Student      Score
##  After :10   a      :2   Min.   : 53.0
##  Before:10   b      :2   1st Qu.: 72.0
```

```
##               c      :2   Median : 76.0
##               d      :2   Mean   : 77.6
##               e      :2   3rd Qu.: 86.5
##               f      :2   Max.   :100.0
##               (Other):8
```

```r
t.test(Score ~ Time,
       data=dat1,
       paired = TRUE,
       conf.level = 0.95)
```

```
##
##  Paired t-test
##
## data:  Score by Time
## t = 3.8084, df = 9, p-value = 0.004163
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##   4.141247 16.258753
## sample estimates:
## mean of the differences
##                    10.2
```