Psychology 350: Advanced statistics and programming in R
Correlation and regression

William Revelle

Department of Psychology
Northwestern University
Evanston, Illinois USA

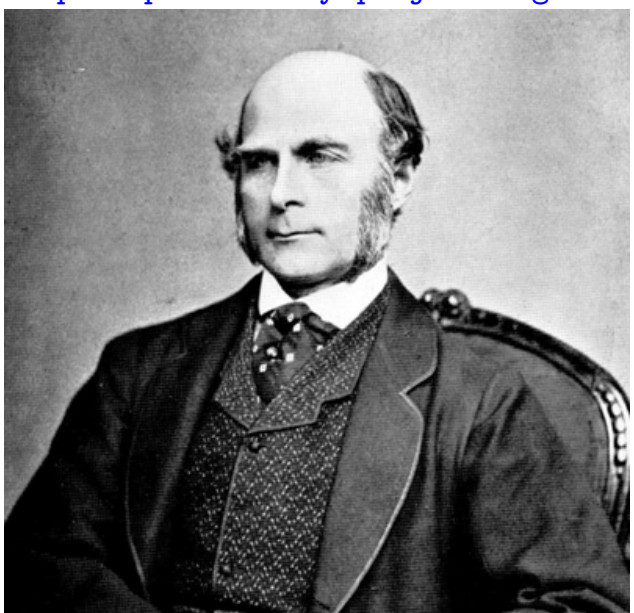
April, 2024

Outline

Francis Galton 1822-1911

Francis Galton (1822-1911) was among the most influential psychologists of the 19th century. He did pioneering work on the correlation coefficient, behavior genetics and the measurement of individual differences. He introspectively examined the question of free will and introduced the lexical hypothesis to the study of personality and character. In addition to psychology, he did pioneering work in meteorology and introduced the scientific use of fingerprints. Whenever he could, he counted.

https://personality-project.org/revelle/publications/galton.pdf (Revelle, 2015b)



Karl Pearson 1857-1936

Carl (Karl) Pearson was among the most influential statisticians of the early 20th century. Founder of the statistics department at University College London. He developed the Pearson Product Moment Correlation Coefficient, its special case the ϕ coefficient, and the tetrachoric correlation. Major behavior geneticist and eugenicist.

Charles Spearman 1863-1945

Charles Spearman (1863-1945) was the leading psychometrician of the early 20th century. His work on the classical test theory, factor analysis, and the g theory of intelligence continues to influence psychometrics, statistics, and the study of intelligence. More than 100 years after their publication, his most influential papers remain two of the most frequently cited articles in psychometrics and intelligence.

https://personality-project.org/revelle/publications/spearman.pdf (Revelle, 2015a)

Galton's height data

Table: The relationship between the average of both parents (mid parent) and the height of their children. The basic data table is from Galton (1886) who used these data to introduce reversion to the mean (and thus, linear regression). The data are available as part of the **UsingR** or **psychTools** packages.

```
> library(psych)
> data(galton)
> galton.tab <- table(galton)
> galton.tab[order(rank(rownames(galton.tab)),decreasing=TRUE),] #sort it by decreasing row va
```

	child													
parent	61.7	62.2	63.2	64.2	65.2	66.2	67.2	68.2	69.2	70.2	71.2	72.2	73.2	73.7
73	0	0	0	0	0	0	0	0	0	0	0	1	3	0
72.5	0	0	0	0	0	0	0	1	2	1	2	7	2	4
71.5	0	0	0	0	1	3	4	3	5	10	4	9	2	2
70.5	1	0	1	0	1	1	3	12	18	14	7	4	3	3
69.5	0	0	1	16	4	17	27	20	33	25	20	11	4	5
68.5	1	0	7	11	16	25	31	34	48	21	18	4	3	0
67.5	0	3	5	14	15	36	38	28	38	19	11	4	0	0
66.5	0	3	3	5	2	17	17	14	13	4	0	0	0	0
65.5	1	0	9	5	7	11	11	7	7	5	2	1	0	0
64.5	1	1	4	4	1	5	5	0	2	0	0	0	0	0
64	1	0	2	4	1	2	2	1	1	0	0	0	0	0

Galton's height data

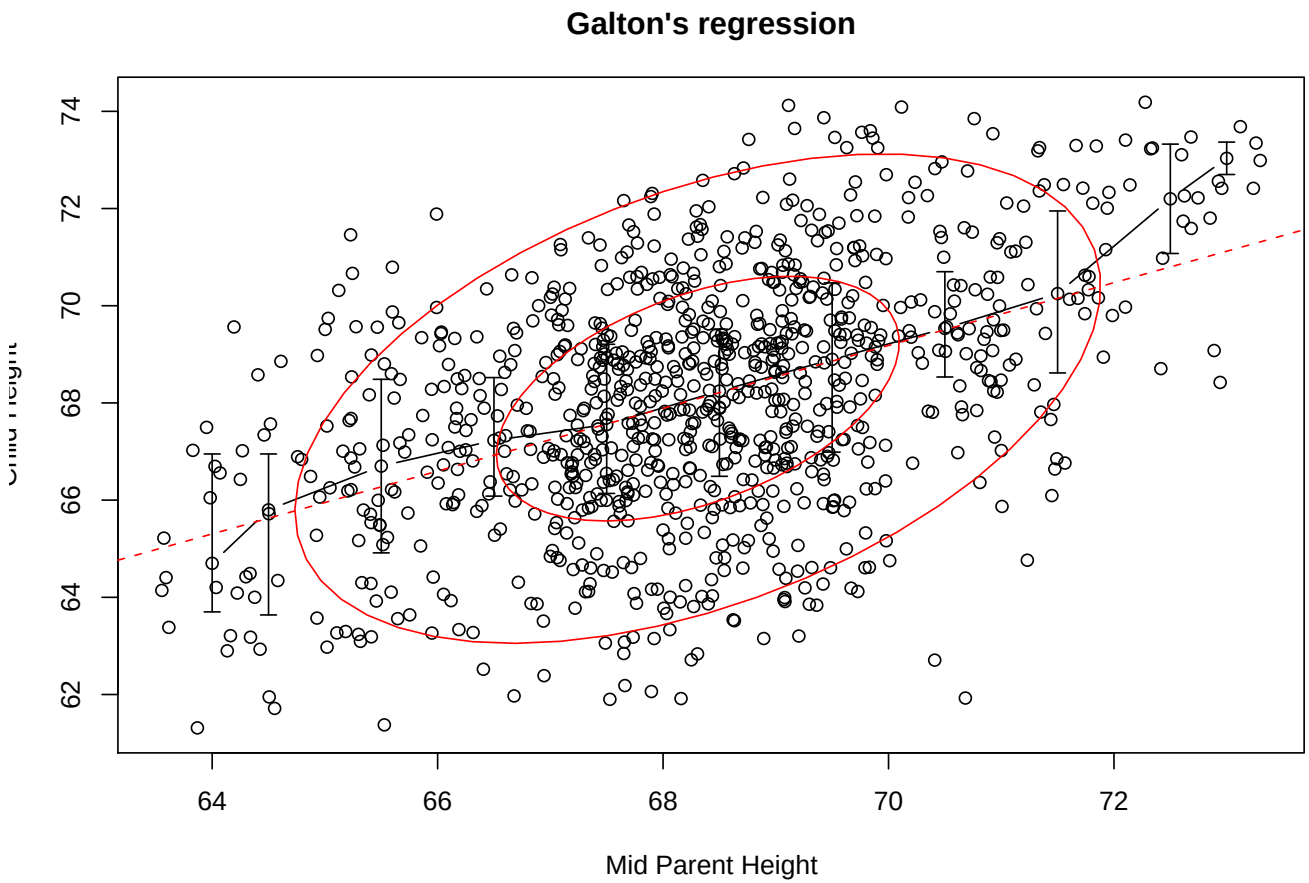
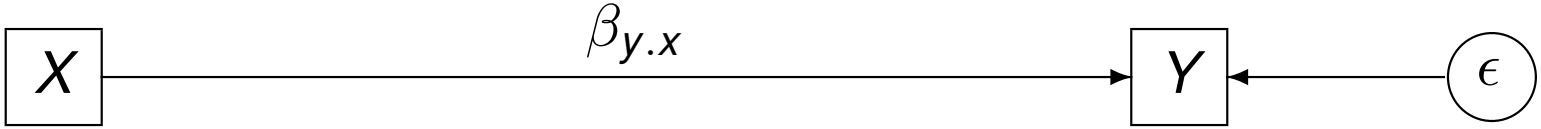


Figure: Galton's data can be plotted to show the relationships between mid parent and child heights. Because the original data are grouped, the data points have been *jittered* to emphasize the density of points along the median. The bars connect the first, 2nd (median) and third quartiles. The dashed line is the best fitting linear fit, the ellipses represent one and two standard deviations from the mean.

Bivariate Regression

X Y ϵ



$$y = \hat{y} + \epsilon = \beta_{y.x}x + \epsilon$$

$$\beta_{y.x} = \frac{\sigma_{xy}}{\sigma_x^2}$$

$$\epsilon = y - \hat{y}$$

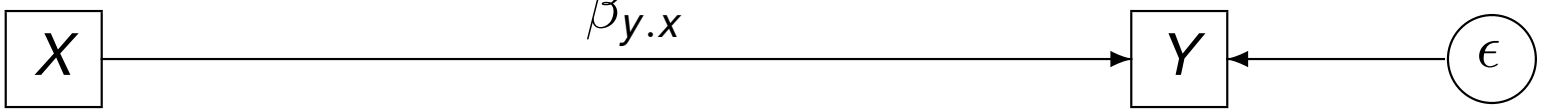
$$\sum(\epsilon^2) = \sum(y - \hat{y})^2 = \sum(y - \beta_{y.x}x)^2 = \sum(y^2 - 2y\beta_{y.x}x + (\beta_{y.x}x)^2)$$

$$\text{Minimize } \sum(\epsilon^2) w.r.t. \beta => \frac{d(\epsilon^2)}{d\beta} = 0 => -2\sigma_{xy} + 2\beta_{y.x}\sigma_x^2 = 0 =>$$

$$\beta_{y.x} = \frac{\sigma_{xy}}{\sigma_x^2}$$

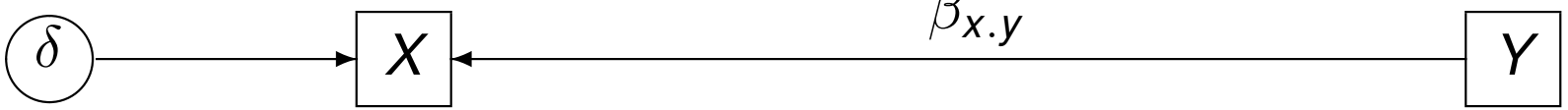
Bivariate Regression

δ X Y ϵ



$$y = \hat{y} + \epsilon = \beta_{y.x}x + \epsilon$$

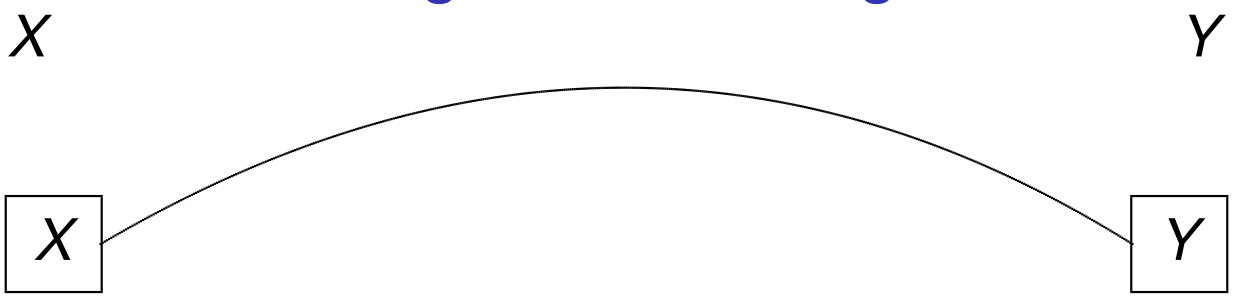
$$\beta_{y.x} = \frac{\sigma_{xy}}{\sigma_x^2}$$



$$x = \hat{x} + \delta = \beta_{x.y}y + \delta$$

$$\beta_{y.x} = \frac{\sigma_{xy}}{\sigma_y^2}$$

Bivariate Correlation is the geometric average of the two regressions



$$x = \hat{x} + \delta = \beta_{x.y} y + \delta \qquad y = \hat{y} + \epsilon = \beta_{y.x} x + \epsilon$$

$$\beta_{y.x} = \frac{\sigma_{xy}}{\sigma_y^2} \qquad \beta_{y.x} = \frac{\sigma_{xy}}{\sigma_x^2}$$

$$r_{xy} = \frac{\sigma_{xy}}{\sqrt{\sigma_x^2 \sigma_y^2}}$$

$r_{xy} = \sigma_{z_x z_y}$ (the covariance of standard scores)

Using R

1. The standard 3 steps
   - 1.1 Input the data
   - 1.2 Descriptive statistics
   - 1.3 Inferential statistics
2. Core R will find correlations and do some simple graphics
3. *psych* was developed for the particular needs of psychologists

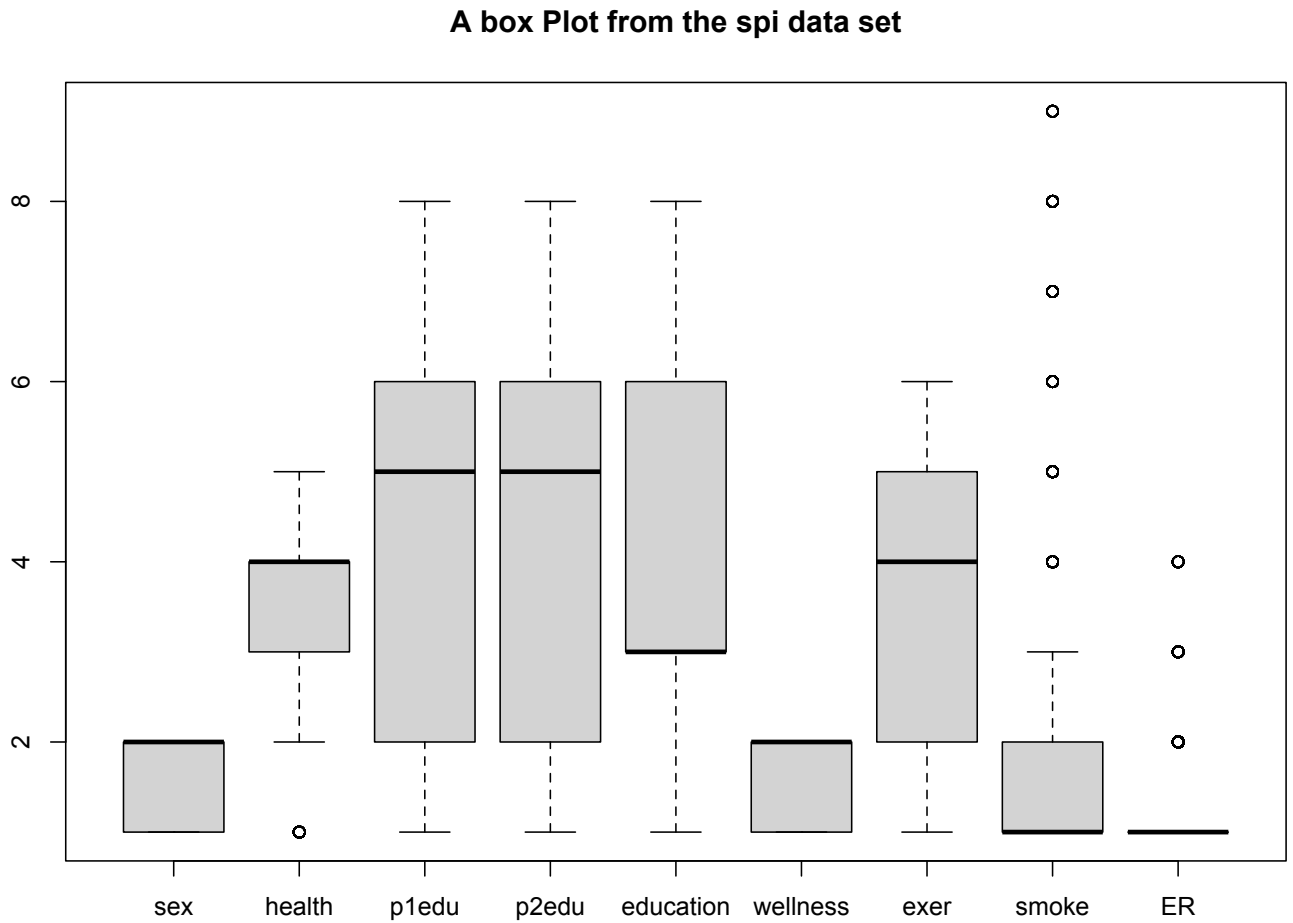
Use the SPI data set

R code

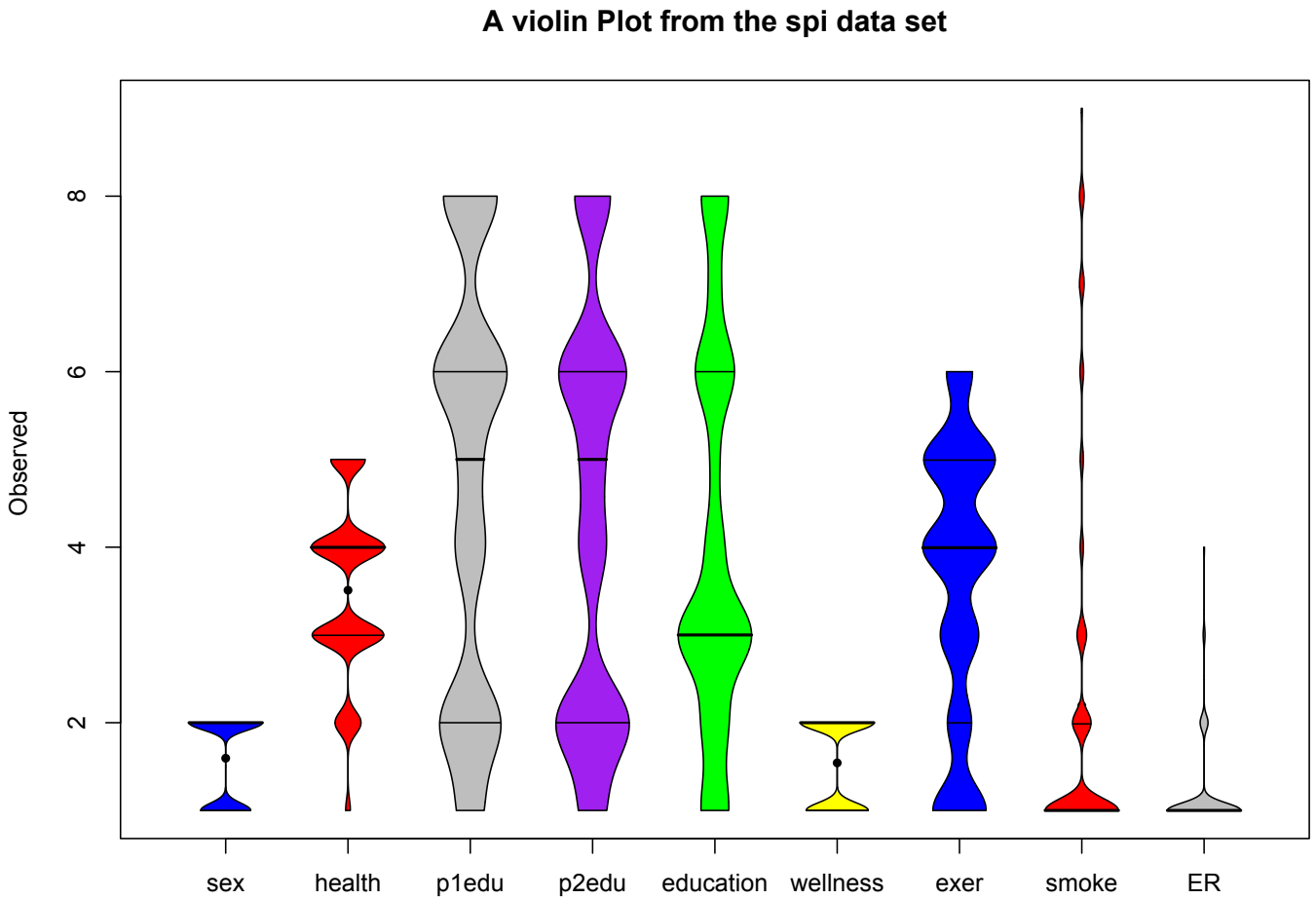
```
library(psych}
#filename <- spi # a built in data set
my.data <- data(spi)
dim(spi)
colnames(spi)
boxplot(spi[2:10],main="A box Plot from the spi data set")
violin(spi[2:10],main="A violin Plot from the spi data set")
```

```
dim(spi)
[1] 4000  145
> colnames(spi)
  [1] "age"       "sex"       "health"    "p1edu"     "p2edu"     "education" "wellness"  "ex
 [14] "q_578"     "q_1367"    "q_4252"    "q_4296"    "q_904"     "q_240"     "q_2745"    "q_
 [27] "q_1243"    "q_219"     "q_610"     "q_1389"    "q_530"     "q_56"      "q_152"     "q_
 [40] "q_312"     "q_811"     "q_1664"    "q_1989"    "q_1812"    "q_1744"    "q_1253"    "q_
 [53] "q_4289"    "q_1244"    "q_1081"    "q_348"     "q_1738"    "q_1915"    "q_736"     "q_
 [66] "q_1683"    "q_1923"    "q_2765"    "q_1781"    "q_4249"    "q_501"     "q_1444"    "q_
 [79] "q_1242"    "q_377"     "q_1248"    "q_803"     "q_607"     "q_755"     "q_571"     "q_
 [92] "q_851"     "q_1585"    "q_4243"    "q_820"     "q_598"     "q_1505"    "q_2853"    "q_
[105] "q_369"     "q_901"     "q_379"     "q_296"     "q_1635"    "q_612"     "q_1880"    "q_
[118] "q_1825"    "q_1832"    "q_176"     "q_684"     "q_1371"    "q_1662"    "q_808"     "q_
[131] "q_169"     "q_398"     "q_131"     "q_871"     "q_1685"    "q_1706"    "q_1132"    "q_
[144] "q_1840"    "q_1328"
```

A simple box plot to describe the data



A simple violin plot to show the density



Or reading a data set

R code

```
filename<-"https://personality-project.org/r/datasets/simulation.txt"
my.data <- read.file(filename)
dim(my.data)    #what are the number of Rows and Columns of the data
colnames(my.data)    # #what are the column names?
headTail(my.data)  #show the first and last 4 rows
describe(my.data)  #get descriptive statistics
```

```
> dim(my.data)
[1] 72  7
> colnames(my.data)
[1] "Time"        "Anxiety"     "Impulsivity" "sex"         "Arousal"     "Tension"     "Perf
> headTail(my.data)
    Time Anxiety Impulsivity sex Arousal Tension Performance
1      9       4           9   1      50      55          40
2     19       8           8   1      70      64          90
3      9       5          10   2      50      69          48
4      9       4           1   2      57      55          68
...  ...     ...         ... ...     ...     ...         ...
69    19       6           1   1      66      53          88
70     9       5          10   2      48      63          40
71    19       6           8   2      69      60          95
72    19      10           1   2      66      48          93
> describe(my.data)
            vars  n  mean   sd median trimmed  mad min max range  skew kurtosis   se
Time           1 72 14.28 5.03   19.0   14.34 0.00   9  19    10 -0.11    -2.02 0.59
Anxiety        2 72  5.24 2.18    5.0    5.24 2.97   0  10    10 -0.04    -0.65 0.26
Impulsivity    3 72  4.90 3.98    4.5    4.88 5.19   0  10    10  0.02    -1.83 0.47
sex            4 72  1.50 0.50    1.5    1.50 0.74   1   2     1  0.00    -2.03 0.06
Arousal        5 72 60.90 8.10   66.0   61.29 5.93  48  70    22 -0.27    -1.67 0.96
Tension        6 72 56.83 6.29   57.0   57.14 5.93  38  69    31 -0.53     0.42 0.74
Performance    7 72 72.21 17.41  78.0   73.19 18.53 38  98    60 -0.43    -1.10 2.05
```

The `summary` function versus `describe`

R code

```
summary(my.data)  #core R way of showing summary statistics
describe(my.data)  # the psych package approach
```

```
summary(my.data)
      Time          Anxiety        Impulsivity          sex          Arousal         Tension
 Min.   : 9.00   Min.   : 0.000   Min.   : 0.000   Min.   :1.0   Min.   :48.00   Min.   :38.0
 1st Qu.: 9.00   1st Qu.: 4.000   1st Qu.: 1.000   1st Qu.:1.0   1st Qu.:54.00   1st Qu.:53.7
 Median :19.00   Median : 5.000   Median : 4.500   Median :1.5   Median :66.00   Median :57.0
 Mean   :14.28   Mean   : 5.236   Mean   : 4.903   Mean   :1.5   Mean   :60.90   Mean   :56.8
 3rd Qu.:19.00   3rd Qu.: 7.000   3rd Qu.: 8.250   3rd Qu.:2.0   3rd Qu.:68.25   3rd Qu.:61.0
 Max.   :19.00   Max.   :10.000   Max.   :10.000   Max.   :2.0   Max.   :70.00   Max.   :69.0
```

```
describe(my.data)
            vars  n  mean    sd median trimmed   mad min max range  skew kurtosis   se
Time           1 72 14.28  5.03   19.0   14.34  0.00   9  19    10 -0.11    -2.02 0.59
Anxiety        2 72  5.24  2.18    5.0    5.24  2.97   0  10    10 -0.04    -0.65 0.26
Impulsivity    3 72  4.90  3.98    4.5    4.88  5.19   0  10    10  0.02    -1.83 0.47
sex            4 72  1.50  0.50    1.5    1.50  0.74   1   2     1  0.00    -2.03 0.06
Arousal        5 72 60.90  8.10   66.0   61.29  5.93  48  70    22 -0.27    -1.67 0.96
Tension        6 72 56.83  6.29   57.0   57.14  5.93  38  69    31 -0.53     0.42 0.74
Performance    7 72 72.21 17.41   78.0   73.19 18.53  38  98    60 -0.43    -1.10 2.05
```

Functions have various options

Examine the help menu,

R code

```
describe(my.data, quant=c(.1,.25,.75,.9), skew=FALSE)
describeData(my.data,head=4,tail=4)
```

```
describe(x, na.rm = TRUE, interp=FALSE,skew = TRUE, ranges = TRUE,trim=.1,
             type=3,check=TRUE,fast=NULL,quant=NULL,IQR=FALSE,omit=FALSE,data=NULL)
```

```
            vars  n  mean    sd min max range   se Q0.1 Q0.25 Q0.75 Q0.9
Time           1 72 14.28  5.03   9  19    10 0.59    9  9.00 19.00 19.0
Anxiety        2 72  5.24  2.18   0  10    10 0.26    2  4.00  7.00  8.0
Impulsivity    3 72  4.90  3.98   0  10    10 0.47    0  1.00  8.25 10.0
sex            4 72  1.50  0.50   1   2     1 0.06    1  1.00  2.00  2.0
Arousal        5 72 60.90  8.10  48  70    22 0.96   50 54.00 68.25 69.0
Tension        6 72 56.83  6.29  38  69    31 0.74   49 53.75 61.00 64.9
Performance    7 72 72.21 17.41  38  98    60 2.05   45 60.00 88.00 92.7

n.obs =  72 of which  72   are complete cases.   Number of variables =  7  of which all are n
            variable # n.obs type H1 H2 H3 H4 T1 T2 T3 T4
Time                1    72    1  9 19  9  9 19  9 19 19
Anxiety             2    72    1  4  8  5  4  6  5  6 10
Impulsivity         3    72    1  9  8 10  1  1 10  8  1
sex                 4    72    1  1  1  2  2  1  2  2  2
Arousal             5    72    1 50 70 50 57 66 48 69 66
Tension             6    72    1 55 64 69 55 53 63 60 48
Performance         7    72    1 40 90 48 68 88 40 95 93
```

Find and display correlations

1. Several ways to display the correlations
2. Using just base R `cor`
3. Rounding and options to base R: the `lowerCor` function
4. Graphically using `pairs.panels`
5. Heatmaps using `corPlot`

Correlation using `cor` function

R code

```
cor(my.data)
round(cor(my.data), digits=2)
```

```
                    Time      Anxiety Impulsivity         sex      Arousal    Tension Performa
Time          1.00000000  0.013216858  0.06121135 -0.05564149  0.956608901 0.05050668  0.87406
Anxiety       0.01321686  1.000000000  0.15052463 -0.12197796  0.002116381 0.34330479  0.04696
Impulsivity   0.06121135  0.150524635  1.00000000  0.06677940 -0.065812947 0.08773475 -0.15565
sex          -0.05564149 -0.121977956  0.06677940  1.00000000 -0.060408880 0.04004630 -0.02812
Arousal       0.95660890  0.002116381 -0.06581295 -0.06040888  1.000000000 0.05358998  0.91904
Tension       0.05050668  0.343304791  0.08773475  0.04004630  0.053589983 1.00000000  0.02194
Performance   0.87406181  0.046965844 -0.15565556 -0.02812176  0.919047382 0.02194427  1.00000
```

```
            Time Anxiety Impulsivity   sex Arousal Tension Performance
Time        1.00    0.01        0.06 -0.06    0.96    0.05        0.87
Anxiety     0.01    1.00        0.15 -0.12    0.00    0.34        0.05
Impulsivity 0.06    0.15        1.00  0.07   -0.07    0.09       -0.16
sex        -0.06   -0.12        0.07  1.00   -0.06    0.04       -0.03
Arousal     0.96    0.00       -0.07 -0.06    1.00    0.05        0.92
Tension     0.05    0.34        0.09  0.04    0.05    1.00        0.02
Performance 0.87    0.05       -0.16 -0.03    0.92    0.02        1.00
```

The problem of missing data

R code

```
cor(spi[1:10])       #this shows NA
 describe(spi[1:10])  #notice we have lots of missing values
```

```
cor(spi[1:10])
          age sex health p1edu p2edu education wellness exer smoke ER
age         1  NA     NA    NA    NA        NA       NA   NA    NA NA
sex        NA   1     NA    NA    NA        NA       NA   NA    NA NA
health     NA  NA      1    NA    NA        NA       NA   NA    NA NA
p1edu      NA  NA     NA     1    NA        NA       NA   NA    NA NA
p2edu      NA  NA     NA    NA     1        NA       NA   NA    NA NA
education  NA  NA     NA    NA    NA         1       NA   NA    NA NA
wellness   NA  NA     NA    NA    NA        NA        1   NA    NA NA
exer       NA  NA     NA    NA    NA        NA       NA    1    NA NA
smoke      NA  NA     NA    NA    NA        NA       NA   NA     1 NA
ER         NA  NA     NA    NA    NA        NA       NA   NA    NA  1
> describe(spi[1:10])
          vars    n  mean    sd median trimmed  mad min max range  skew kurtosis   se
age          1 4000 26.90 11.49     23   25.02 7.41  11  90    79  1.45     1.80 0.18
sex          2 3946  1.60  0.49      2    1.62 0.00   1   2     1 -0.39    -1.85 0.01
health       3 3536  3.51  0.98      4    3.54 1.48   1   5     4 -0.25    -0.42 0.02
p1edu        4 3051  4.72  2.39      5    4.77 4.45   1   8     7 -0.11    -1.33 0.04
p2edu        5 2896  4.33  2.32      5    4.28 4.45   1   8     7  0.09    -1.33 0.04
education    6 3330  4.10  2.21      3    4.00 1.48   1   8     7  0.41    -1.04 0.04
wellness     7 3311  1.54  0.50      2    1.55 0.00   1   2     1 -0.17    -1.97 0.01
exer         8 3310  3.57  1.60      4    3.60 1.48   1   6     5 -0.35    -1.06 0.03
smoke        9 3348  2.19  2.04      1    1.70 0.00   1   9     8  1.83     2.19 0.04
ER          10 3347  1.16  0.48      1    1.03 0.00   1   4     3  3.42    12.74 0.01
```

Do it again, but treating missing values correctly, and rounding the output

R code

```
round(cor(spi[1:10], use="pairwise"),digits = 2)
```

```
           age   sex health p1edu p2edu education wellness  exer smoke    ER
age       1.00 -0.05   0.00 -0.12 -0.14      0.57     0.08  0.04  0.11 -0.06
sex      -0.05  1.00  -0.05 -0.02 -0.02     -0.08     0.10 -0.07 -0.05  0.10
health    0.00 -0.05   1.00  0.12  0.12      0.09     0.09  0.35 -0.15 -0.14
p1edu    -0.12 -0.02   0.12  1.00  0.58      0.06     0.04  0.10 -0.08 -0.05
p2edu    -0.14 -0.02   0.12  0.58  1.00      0.06     0.06  0.09 -0.08 -0.06
education 0.57 -0.08   0.09  0.06  0.06      1.00     0.00  0.07  0.01 -0.10
wellness  0.08  0.10   0.09  0.04  0.06      0.00     1.00  0.15 -0.06  0.08
exer      0.04 -0.07   0.35  0.10  0.09      0.07     0.15  1.00 -0.14 -0.05
smoke     0.11 -0.05  -0.15 -0.08 -0.08      0.01    -0.06 -0.14  1.00  0.08
ER       -0.06  0.10  -0.14 -0.05 -0.06     -0.10     0.08 -0.05  0.08  1.00
```

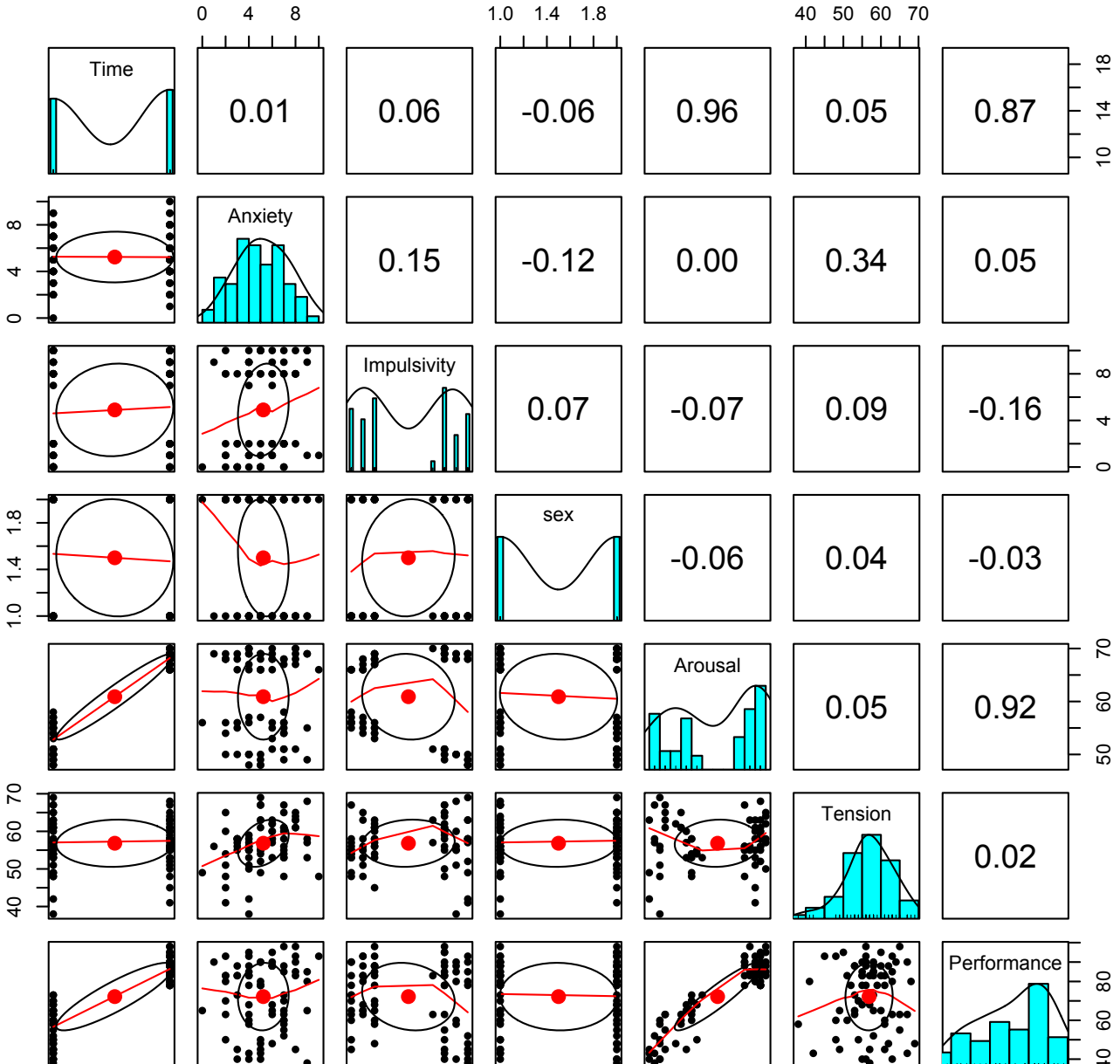
lowerCor shows pairwise correlations and rounds them

R code

```
lowerCor(spi[1:10]
```

```
          age   sex   helth p1edu p2edu edctn wllns exer  smoke ER
age        1.00
sex       -0.05  1.00
health     0.00 -0.05  1.00
p1edu     -0.12 -0.02  0.12  1.00
p2edu     -0.14 -0.02  0.12  0.58  1.00
education  0.57 -0.08  0.09  0.06  0.06  1.00
wellness   0.08  0.10  0.09  0.04  0.06  0.00  1.00
exer       0.04 -0.07  0.35  0.10  0.09  0.07  0.15  1.00
smoke      0.11 -0.05 -0.15 -0.08 -0.08  0.01 -0.06 -0.14  1.00
ER        -0.06  0.10 -0.14 -0.05 -0.06 -0.10  0.08 -0.05  0.08  1.00
```

A Scattter Plot Matrix (splom) plot shows frequencies and correlations



`lowerCor`: a simple function to call `cor`

To learn how to code, it is helpful to examine code of existing functions. Just type the name of the function without the () to see the function.

Each function has three parts: the definition (including the parameters); the execution; the results).

Calls `cor` with certain defaults, uses another function `lowerMat` to make for pretty output, then returns the full correlation matrix

```
lowerCor <-
function (x, digits = 2, use = "pairwise", method = "pearson",
    minlength = 5, show = TRUE)    # the input parameters
{           #this begins the function
    nvar <- NCOL(x)                 #not used
    x <- char2numeric(x)    #convert characters to numbers
    R <- cor(x, use = use, method = method)
    if (show)
        lowerMat(R, digits, minlength = minlength)
    invisible(R)     #return a value, but don't show it
}
```

the `lowerMat` function is a bit more complicated

```
lowerMat <- function (R, digits = 2)  {
    lowleft <- lower.tri(R, diag = TRUE)
    nvar <- ncol(R)
    nc <- digits + 3
    width <- getOption("width")
    k1 <- width/(nc + 2)
    if (is.null(colnames(R))) {
        colnames(R) <- paste("C", 1:nvar, sep = "")
    }
    if (is.null(rownames(R))) {
        rownames(R) <- paste("R", 1:nvar, sep = "")
    }
    colnames(R) <- abbreviate(colnames(R), minlength = digits +
        3)
    nvar <- ncol(R)
    nc <- digits + 3
    if (k1 * nvar < width) {
        k1 <- nvar
    }
    k1 <- floor(k1)
    fx <- format(round(R, digits = digits))
    if (nrow(R) == ncol(R)) {
        fx[!lowleft] <- ""
    }
    for (k in seq(0, nvar, k1)) {
        if (k < nvar) {
            print(fx[(k + 1):nvar, (k + 1):min((k1 + k), nvar)],
                quote = FALSE)
        }
    }
    invisible(R[lower.tri(R, diag = FALSE)])
}
```

Lets unpack the Galton height data graphic display

1. Get the data from `galton` data set in the *psychTools* package.
2. `describe` it
3. make a table
4. sort the table to be decreasing order'
5. show the table
6. plot the data (but the points do not show individual subjects)
7. Try `jitter`ing the points

See the R studio analysis

1. https://personality-project.org/courses/350/350.week2.html

Correlation and sampling variability

1. Correlations are sample based estimates
2. Can we examine how much the estimate varies across samples?
3. Use the subjects from the `galton` data set as an example
4. We use the `sample` function to create multiple samples from `galton`
5. We then do again, but this time sampling with replacement (i.e., a bootstrap sample).

Sample from `galton`

R code

```
set.seed(42) #for replicability
nsub <- nrow(galton)  #how many cases"
samp.size <- 20  #we set the sample size we want
samp <- sample(nsub,samp.size)     #samples without replacement
samp  #show the case numbers
cor(galton[samp,]) #find the correlation for this sample
```

```
> set.seed(42) #for replicability
> nsub <- nrow(galton)  #how many cases"
> samp.size <- 20  #we set the sample size we want
> samp <- sample(nsub,samp.size)
> samp  #show the case numbers
 [1] 849 869 265 769 593 480 680 125 605 648 421 660 857 234 423 859 893 108 433 510
> cor(galton[samp,]) #find the correlation for this sample
          parent     child
parent 1.0000000 0.4591206
child  0.4591206 1.0000000
```

Many samples from `galton`

R code

```
set.seed(42) #for replicability
nsub <- nrow(galton)  #how many cases"
samp.size <- 20  #we set the sample size we want
replications <- 100  #we want to repeat this 100 times
result <- rep(NA,replications) #create this vector
#use a for loop to repeat the code inside the { }
for(i in 1:replications) { #repeat some code
   samp <- sample(nsub,samp.size)
    result[i] <- cor(galton[samp,])[1,2] #find the correlation for th
}    #end of the loop
describe(result)
```

```
set.seed(42) #for replicability
> nsub <- nrow(galton)  #how many cases"
> samp.size <- 20  #we set the sample size we want
> replications <- 100  #we want to repeat this 100 times
> #
> result <- rep(NA,replications) #create this vector
> #use a for loop
>
> for(i in 1:replications) { #repeat some code
+ samp <- sample(nsub,samp.size)
+ result[i] <- cor(galton[samp,])[1,2] #find the correlation for this sample}
+ }    #end of the loop
> describe(result)
   vars   n mean   sd median trimmed  mad   min  max range skew kurtosis   se
X1    1 100 0.41 0.23   0.43    0.42 0.25 -0.16 0.79  0.95 -0.6    -0.32 0.02
```

Make a small function to do this

1. If you have a piece of code you want to use again and again, you can create a small function to do it.
2. All functions have a beginning (where name the function specify the input and any other options).
3. A body (where you actually do some work).
4. An ending (where you return the result).

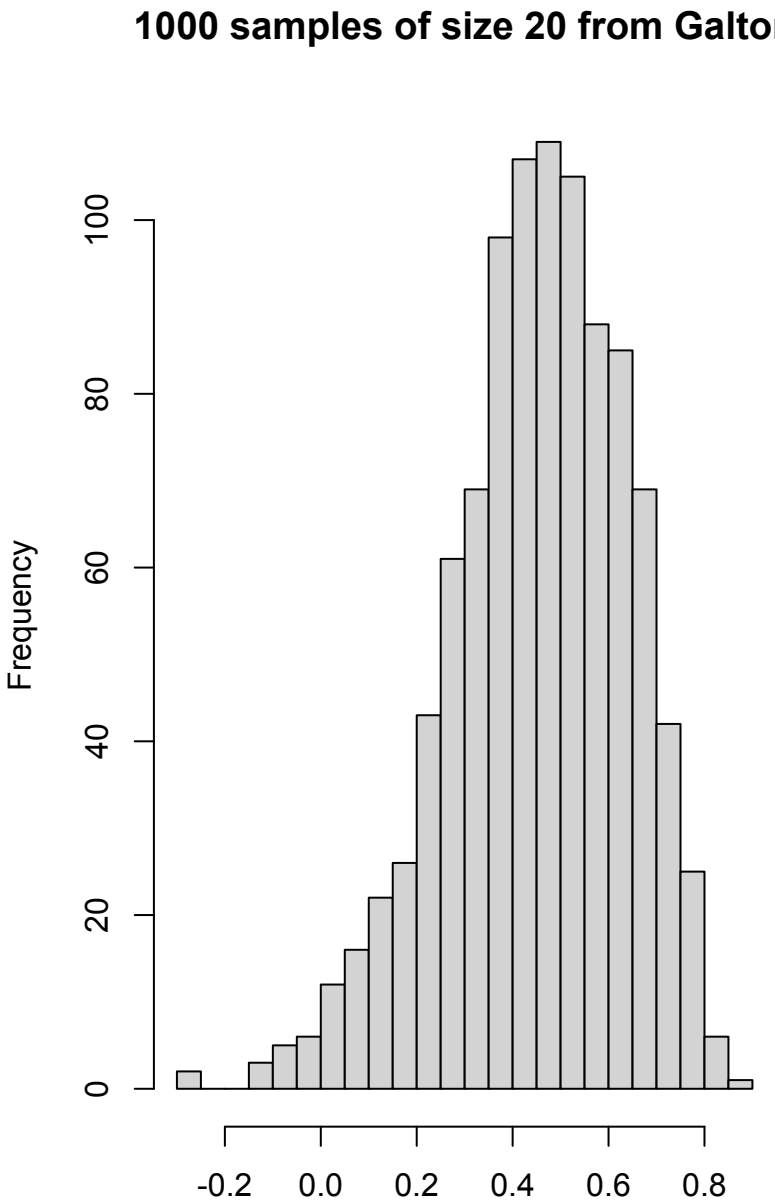
A sample function: to do samplIng

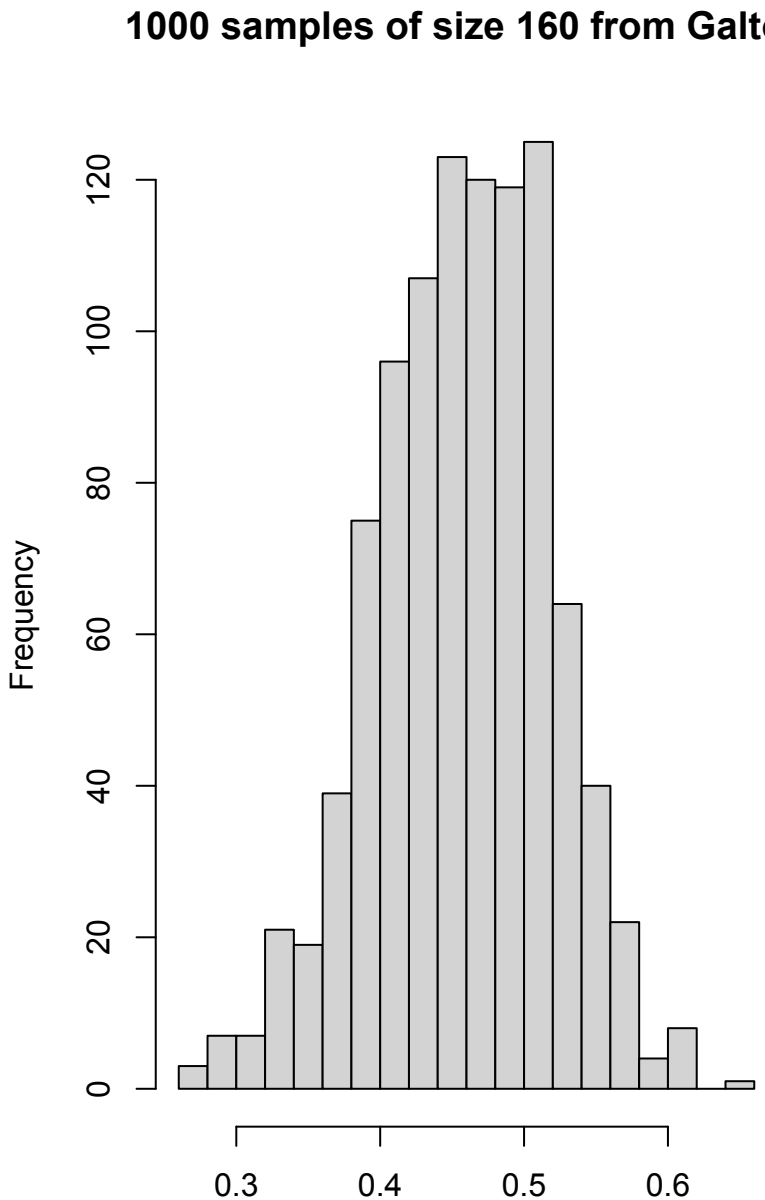
R code

```
                # default values may be specified
small <- function(data=NULL,sample.size=20, n.iter=1000)  {
nsub <- nrow(data)    #this figures out the sampe size dynamically
result <- rep(NA,n.iter) #create a vector store the results

#use a for loop to repeat the code inside the { }

for(i in 1:n.iter) { #repeat some code
   samp <- sample(nsub,sample.size)
    result[i] <- cor(data[samp,])[1,2]
         #find the correlation for this sample and save it
                        }    #end of the loop
return(result)   #return the value we find
}   #end of function
```

```
test <- small(galton) #this uses the default values
par(mfrow=c(1,2))  #two column output
hist(test,breaks=21,main="1000 samples of size 20 from Galton")
test <- small(galton,160) #this uses the default values
hist(test,breaks=21,main="1000 samples of size 160 from Galton")
```

HIstograms of correlations from Galton for samples of 20 and 160





Errors and sample size

1. What happens if we plot our results as a function of sample size?
2. We use our small function, but call it with different sample sizes
3. Store the results in a *data.frame*
4. We will then plot this resulting data.frame

R code

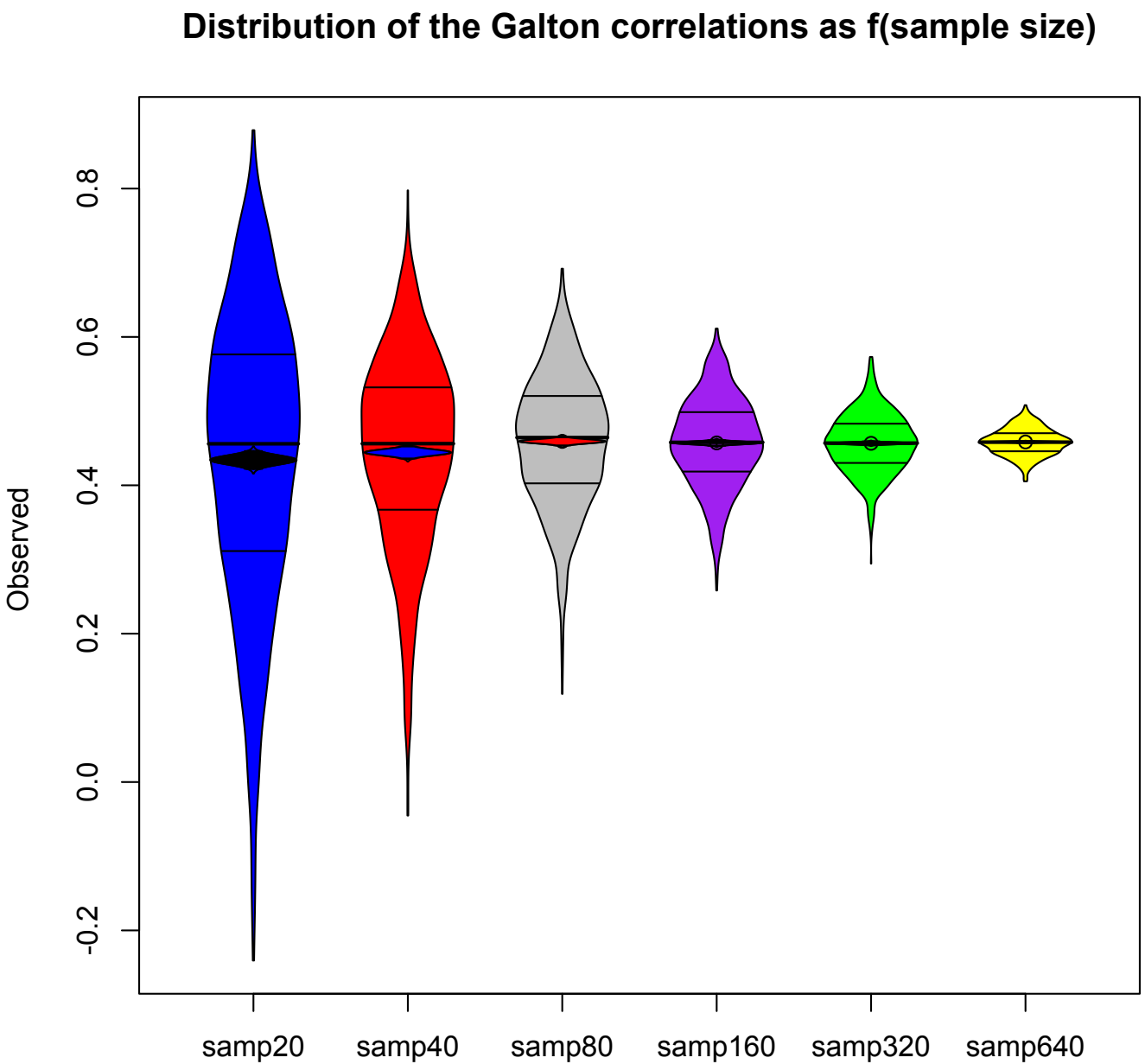
```
samp20 <- small(galton,20)
samp40 <- small(galton,40)
samp80 <- small(galton,80)
samp160 <- small(galton,160)
samp320 <- small(galton,320)
samp640<- small(galton,640)
sample.df <- data.frame(samp20 ,samp40, samp80, samp160,
            samp320,samp640)
describe(sample.df)
```

R code

```
samp20 <- small(galton,20)
samp40 <- small(galton,40)
samp80 <- small(galton,80)
samp160 <- small(galton,160)
samp320 <- small(galton,320)
samp640<- small(galton,640)
sample.df <- data.frame(samp20 ,samp40, samp80, samp160,
           samp320,samp640)
describe(sample.df)
```

```
describe(sample.df)
        vars    n mean   sd median trimmed  mad   min  max range  skew kurtosis   se
samp20     1 1000 0.43 0.19   0.46    0.44 0.20 -0.24 0.88  1.12 -0.52     0.05 0.01
samp40     2 1000 0.44 0.13   0.46    0.45 0.12 -0.05 0.80  0.84 -0.58     0.38 0.00
samp80     3 1000 0.46 0.09   0.46    0.46 0.09  0.12 0.69  0.57 -0.31     0.18 0.00
samp160    4 1000 0.46 0.06   0.46    0.46 0.06  0.26 0.61  0.35 -0.22    -0.08 0.00
samp320    5 1000 0.46 0.04   0.46    0.46 0.04  0.29 0.57  0.28 -0.09     0.27 0.00
samp640    6 1000 0.46 0.02   0.46    0.46 0.02  0.41 0.51  0.10 -0.05    -0.12 0.00
```

A violin plot of the results



The bootstrap

1. Prior analysis has a problem in that as the sample size increases, the variability of any estimate has to decrease as we use the same sample every time.
2. Consider what happens if the original sample is not very big (say 50)
3. Then any sample more than 50 will have identical values
4. The solution is to sample with replacement.
5. Treat the original sample as a population, and get sample estimates of the same size by sampling with replacement.
6. This is known as the Bootstrap (Efron, 1979; Efron & Gong, 1983).

Change our original small function to a `boot` function

R code

```
small <- function(data=NULL,sample.size=20, n.iter=1000)  {
   nsub <- nrow(data)    #this figures out the sampe size dynamically
    result <- rep(NA,n.iter) #create this vector
#use a for loop to repeat the code inside the { }
 for(i in 1:n.iter) { #repeat some code
     samp <- sample(nsub,sample.size)
    result[i] <- cor(data[samp,])[1,2]
         #find the correlation for this sample and save it
         }    #end of the loop
return(result)   #return the value we find
}   #end of function

#Our bootstrap function
boot <- function(data=NULL,sample.size=20, n.iter=1000)  {
  nsub <- nrow(data)    #this figures out the sampe size dynamically
  result <- rep(NA,n.iter) #create this vector
 #use a for loop to repeat the code inside the { }
  for(i in 1:n.iter) { #repeat some code
     samp <- sample(nsub,sample.size,replace=TRUE)   #sample with repl
    result[i] <- cor(data[samp,])[1,2]
    #find the correlation for this sample and save it
                       }    #end of the loop
return(result)   #return the value we find
}   #end of function
```

Compare the two

R code

```
samp20 <- small(galton,20)
samp40 <- small(galton,40)
samp80 <- small(galton,80)
samp160 <- small(galton,160)
samp320 <- small(galton,320)
samp640<- small(galton,640)
samp928 <- small(galton,928)
sample.df <- data.frame(samp20 ,samp40, samp80, samp160,
           samp320,samp640,samp928)
describe(sample.df)
```

```
describe(sample.df)
        vars    n mean   sd median trimmed  mad   min  max range     skew kurtosis   se
samp20     1 1000 0.44 0.19   0.47    0.45 0.19 -0.25 0.83  1.08    -0.63     0.33 0.01
samp40     2 1000 0.45 0.13   0.47    0.46 0.13  0.01 0.79  0.78    -0.42    -0.02 0.00
samp80     3 1000 0.46 0.09   0.46    0.46 0.09  0.09 0.70  0.61    -0.37     0.29 0.00
samp160    4 1000 0.45 0.06   0.45    0.45 0.06  0.19 0.61  0.42    -0.22     0.00 0.00
samp320    5 1000 0.46 0.04   0.46    0.46 0.04  0.32 0.56  0.24    -0.17    -0.07 0.00
samp640    6 1000 0.46 0.02   0.46    0.46 0.02  0.41 0.52  0.12    -0.09     0.12 0.00
samp928    7 1000 0.46 0.00   0.46    0.46 0.00  0.46 0.46  0.00 11711.20     0.09 0.00
```

Bootstrap of galton

R code

```
boot20 <- boot(galton,20)
boot40 <- boot(galton,40)
boot80 <- boot(galton,80)
boot160 <- boot(galton,160)
boot320 <- boot(galton,320)
boot640<- boot(galton,640)
boot928 <- boot(galton,928)
boot.df <- data.frame(boot20 ,boot40, boot80, boot160,
           boot320,boot640,boot928)
describe(boot.df)
```

```
describe(boot.df)
        vars    n mean   sd median trimmed  mad   min  max range  skew kurtosis   se
boot20     1 1000 0.44 0.19   0.46    0.46 0.19 -0.26 0.86  1.13 -0.63     0.28 0.01
boot40     2 1000 0.45 0.13   0.46    0.45 0.13 -0.13 0.79  0.91 -0.59     0.68 0.00
boot80     3 1000 0.45 0.09   0.46    0.46 0.09  0.14 0.70  0.56 -0.26    -0.09 0.00
boot160    4 1000 0.46 0.06   0.46    0.46 0.06  0.24 0.68  0.44 -0.19     0.06 0.00
boot320    5 1000 0.46 0.05   0.46    0.46 0.05  0.32 0.59  0.27 -0.10    -0.10 0.00
boot640    6 1000 0.46 0.03   0.46    0.46 0.03  0.35 0.54  0.19 -0.21     0.00 0.00
boot928    7 1000 0.46 0.03   0.46    0.46 0.03  0.38 0.53  0.15 -0.07    -0.23 0.00
```

Compare the two descriptions

R code

```
describe(sample.df)
 describe(boot.df)
```

```
describe(sample.df)
        vars    n mean   sd median trimmed  mad   min  max range     skew kurtosis   se
samp20     1 1000 0.44 0.19   0.47    0.45 0.19 -0.25 0.83  1.08    -0.63     0.33 0.01
samp40     2 1000 0.45 0.13   0.47    0.46 0.13  0.01 0.79  0.78    -0.42    -0.02 0.00
samp80     3 1000 0.46 0.09   0.46    0.46 0.09  0.09 0.70  0.61    -0.37     0.29 0.00
samp160    4 1000 0.45 0.06   0.45    0.45 0.06  0.19 0.61  0.42    -0.22     0.00 0.00
samp320    5 1000 0.46 0.04   0.46    0.46 0.04  0.32 0.56  0.24    -0.17    -0.07 0.00
samp640    6 1000 0.46 0.02   0.46    0.46 0.02  0.41 0.52  0.12    -0.09     0.12 0.00
samp928    7 1000 0.46 0.00   0.46    0.46 0.00  0.46 0.46  0.00 11711.20     0.09 0.00
```

```
describe(boot.df)
        vars    n mean   sd median trimmed  mad   min  max range  skew kurtosis   se
boot20     1 1000 0.44 0.19   0.46    0.46 0.19 -0.26 0.86  1.13 -0.63     0.28 0.01
boot40     2 1000 0.45 0.13   0.46    0.45 0.13 -0.13 0.79  0.91 -0.59     0.68 0.00
boot80     3 1000 0.45 0.09   0.46    0.46 0.09  0.14 0.70  0.56 -0.26    -0.09 0.00
boot160    4 1000 0.46 0.06   0.46    0.46 0.06  0.24 0.68  0.44 -0.19     0.06 0.00
boot320    5 1000 0.46 0.05   0.46    0.46 0.05  0.32 0.59  0.27 -0.10    -0.10 0.00
boot640    6 1000 0.46 0.03   0.46    0.46 0.03  0.35 0.54  0.19 -0.21     0.00 0.00
boot928    7 1000 0.46 0.03   0.46    0.46 0.03  0.38 0.53  0.15 -0.07    -0.23 0.00
```

sd varies by $\sqrt{(1/N)}$ compare N=20, 80, 320 and 40,160, 640

But without replacement, the larger sample variances are too small.

The effect of sampling with replacement

1. Some subjects are sampled more than once, some are never sampled.

```
set.seed(17)    #to get the same 'random' sequence
sample(20,20)
 [1]  4 19  9 14  7 18  3 17 10 13  5  1 16  6  8 12 11 15  2 20
> sample(20,20,replace=T)
 [1] 18 13 15 17 19  2 12 12 19  5 10  1 16  8 12  5 15 15 15 17
  s <- sample(20,20,replace=T);s  # do it again- another sequence
 [1] 16 13  5 12  5  6 20 18  6 12 18  3 11  5 13 15 13  4  5  2
 unique(s);table(s)  #show the unique values
 [1] 16 13  5 12  6 20 18  3 11 15  4  2
s
 2  3  4  5  6 11 12 13 15 16 18 20
 1  1  1  4  2  1  2  3  1  1  2  1
```

2. For N participants, probability of being sampled on any one trial is 1/N
3. Probability of being not sampled is $1 - 1/N$
4. For N trials, the probability of not being sampled is $(1 - 1/N)^N = 1/e \approx .368$
5. And thus, the probability of being in the sample is $1 - 1/e \approx .632$

Do that in R

R code

```
x <- 1:100
notsampled <- (1-1/x)^x
sampled <- 1-(1-1/x)^x
round(notsampled,2)
```

```
x <- 1:100
> notsampled <- (1-1/x)^x
> sampled <- 1-(1-1/x)^x
> round(notsampled,2)
[1] 0.00 0.25 0.30 0.32 0.33 0.33 0.34 0.34 0.35 0.35 0.35 0.35 0.35 0.35 0.36 0.36
 [17] 0.36 0.36 0.36 0.36 0.36 0.36 0.36 0.36 0.36 0.36 0.36 0.36 0.36 0.36 0.36 0.36
 [33] 0.36 0.36 0.36 0.36 0.36 0.36 0.36 0.36 0.36 0.36 0.36 0.36 0.36 0.36 0.36 0.36
 [49] 0.36 0.36 0.36 0.36 0.36 0.36 0.36 0.36 0.36 0.36 0.36 0.36 0.36 0.36 0.36 0.36
 [65] 0.37 0.37 0.37 0.37 0.37 0.37 0.37 0.37 0.37 0.37 0.37 0.37 0.37 0.37 0.37 0.37
 [81] 0.37 0.37 0.37 0.37 0.37 0.37 0.37 0.37 0.37 0.37 0.37 0.37 0.37 0.37 0.37 0.37
 [97] 0.37 0.37 0.37 0.37
> round(sampled,2)
  [1] 1.00 0.75 0.70 0.68 0.67 0.67 0.66 0.66 0.65 0.65 0.65 0.65 0.65 0.65 0.64 0.64
 [17] 0.64 0.64 0.64 0.64 0.64 0.64 0.64 0.64 0.64 0.64 0.64 0.64 0.64 0.64 0.64 0.64
 [33] 0.64 0.64 0.64 0.64 0.64 0.64 0.64 0.64 0.64 0.64 0.64 0.64 0.64 0.64 0.64 0.64
 [49] 0.64 0.64 0.64 0.64 0.64 0.64 0.64 0.64 0.64 0.64 0.64 0.64 0.64 0.64 0.64 0.64
 [65] 0.63 0.63 0.63 0.63 0.63 0.63 0.63 0.63 0.63 0.63 0.63 0.63 0.63 0.63 0.63 0.63
 [81] 0.63 0.63 0.63 0.63 0.63 0.63 0.63 0.63 0.63 0.63 0.63 0.63 0.63 0.63 0.63 0.63
 [97] 0.63 0.63 0.63 0.63
```

Testing significance

1. Estimates of correlations, like any other statistic, have error.
2. We compare the value of the statistic to its error.
3. The distribution of a correlation is not normal, but if we convert it to Fisher's z, it is.
4. t test of a correlation = $\frac{r}{se_r}$ and $se_r = \sqrt{\frac{1-r*2}{n-2}}$
5. Therefore, $t_r = r * \sqrt{\frac{n-2}{1-r^2}}$
6. A single correlation `cor.test` (core R)
7. Many correlations `corr.test` psych
8. Confidence intervals based upon the bootstrap `corCi`
   $r - z_{\alpha/2}se < r < r + z_{\alpha/2}se$

Fisher's z and the t.test

Two useful functions from *psych*

R code

```
fisherz    #convert r to z
r2t        #given an r, and n, convert to a t
```

```
fisherz
function (rho)
{
    0.5 * log((1 + rho)/(1 - rho))
}

r2t
function (rho, n)
{
    return(rho * sqrt((n - 2)/(1 - rho^2)))
}
```

?r2t

Testing correlations

R code

```
cor.test(galton$parent,galton$child)
r2t(.4587624,928)
corr.test(galton)
corCi(galton)
```

```
cor.test(galton$parent,galton$child)
        Pearson's product-moment correlation
data:  galton$parent and galton$child
t = 15.711, df = 926, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.4064067 0.5081153
sample estimates:
      cor
0.4587624
>r2t(.4587624,928)
[1] 15.71112
> corr.test(galton)
Call:corr.test(x = galton)
Correlation matrix
       parent child
parent   1.00  0.46
child    0.46  1.00
Sample Size
[1] 928
Probability values (Entries above the diagonal are adjusted for multiple tests.)
       parent child
parent      0     0
child       0     0
```

More output

R code

```
temp <- corr.test(galton)
print(temp, short=FALSE)
names(temp)
corCi(galton)
```

```
Call:corr.test(x = galton)

...

 Confidence intervals based upon normal theory.  To get bootstrapped values, try cor.ci
           raw.lower raw.r raw.upper raw.p lower.adj upper.adj
parnt-child      0.41  0.46      0.51     0      0.41      0.51
> names(temp)
 [1] "r"      "n"      "t"      "p"      "p.adj"  "se"     "sef"    "adjust" "sym"    "ci"
[14] "Call"
> corCi(galton)
Call:corCi(x = galton)

 Coefficients and bootstrapped confidence intervals
       parnt child
parent 1.00
child  0.46  1.00

 scale correlations and bootstrapped confidence intervals
           lower.emp lower.norm estimate upper.norm upper.emp p
parnt-child       0.4        0.4     0.46       0.51      0.52 0
```

Adjusting for multiple tests – Bonferroni and Holm corrections

1. “Significance tests" are typically based upon a single test
2. But if we perform multiple tests, we need to adjust the probabilities.
3. The Bonferroni (1936) test adjusts p values by dividing by the number of tests, but this is too conservative (Abdi, 2010).
4. Holm (1979) corrections sorts the p values and adjusts by the sorted order (Bonferroni for first one, Bonferroni with tests -1 for second one, etc.) Still too conservative, but better.
5. Adjustments are provided by `corr.testj` or `p.adjust`

P adjustments depend upon number of tests

R code

```
corr.test(epi.bfi[4:6])
```

```
Call:corr.test(x = epi.bfi[4:6])
Correlation matrix
        epilie epiNeur bfagree
epilie    1.00   -0.25    0.17
epiNeur  -0.25    1.00   -0.08
bfagree   0.17   -0.08    1.00
Sample Size
[1] 231
Probability values (Entries above the diagona
        l are adjusted for multiple tests.)
        epilie epiNeur bfagree
epilie    0.00    0.00    0.02
epiNeur   0.00    0.00    0.21
bfagree   0.01    0.21    0.00
```

R code

```
corr.test(epi.bfi[4:8])
```

```
Call:corr.test(x = epi.bfi[4:8])
Correlation matrix
        epilie epiNeur bfagree bfcon bfext
epilie    1.00   -0.25    0.17  0.23 -0.04
epiNeur  -0.25    1.00   -0.08 -0.13 -0.17
bfagree   0.17   -0.08    1.00  0.45  0.48
bfcon     0.23   -0.13    0.45  1.00  0.27
bfext    -0.04   -0.17    0.48  0.27  1.00
Sample Size
[1] 231
Probability values (Entries above the diagonal
     are adjusted for multiple tests.)
        epilie epiNeur bfagree bfcon bfext
epilie    0.00    0.00    0.04  0.00  0.50
epiNeur   0.00    0.00    0.43  0.13  0.04
bfagree   0.01    0.21    0.00  0.00  0.00
bfcon     0.00    0.04    0.00  0.00  0.00
bfext     0.50    0.01    0.00  0.00  0.00
```

P adjustments Bonferroni versus Holm

R code

```
corr.test(epi.bfi[4:8],
      adjust="bonferroni")
```

```
 corr.test(epi.bfi[4:8],adjust="bonferroni")
Call:corr.test(x = epi.bfi[4:8], adjust = "bonferroni")
Correlation matrix
        epilie epiNeur bfagree bfcon bfext
epilie    1.00   -0.25    0.17  0.23 -0.04
epiNeur  -0.25    1.00   -0.08 -0.13 -0.17
bfagree   0.17   -0.08    1.00  0.45  0.48
bfcon     0.23   -0.13    0.45  1.00  0.27
bfext    -0.04   -0.17    0.48  0.27  1.00
Sample Size
[1] 231
Probability values (Entries above the diagonal
      are adjusted for multiple tests.)
        epilie epiNeur bfagree bfcon bfext
epilie    0.00    0.00    0.09  0.01  1.00
epiNeur   0.00    0.00    1.00  0.43  0.09
bfagree   0.01    0.21    0.00  0.00  0.00
bfcon     0.00    0.04    0.00  0.00  0.00
bfext     0.50    0.01    0.00  0.00  0.00
```

R code

```
corr.test(epi.bfi[4:8])
```

```
Call:corr.test(x = epi.bfi[4:8])
Correlation matrix
        epilie epiNeur bfagree bfcon bfext
epilie    1.00   -0.25    0.17  0.23 -0.04
epiNeur  -0.25    1.00   -0.08 -0.13 -0.17
bfagree   0.17   -0.08    1.00  0.45  0.48
bfcon     0.23   -0.13    0.45  1.00  0.27
bfext    -0.04   -0.17    0.48  0.27  1.00
Sample Size
[1] 231
Probability values (Entries above the diagonal
      are adjusted for multiple tests.)
        epilie epiNeur bfagree bfcon bfext
epilie    0.00    0.00    0.04  0.00  0.50
epiNeur   0.00    0.00    0.43  0.13  0.04
bfagree   0.01    0.21    0.00  0.00  0.00
bfcon     0.00    0.04    0.00  0.00  0.00
bfext     0.50    0.01    0.00  0.00  0.00
```

If you insist on "magic astericks'

Some people think that correlations should be accompanied by "Magic Astericks" or *. These are available in the `stars` object from `corr.test`.

R code

```
ct <- corr.test(epi.bfi[4:8])
ct$stars
```

```
ct$stars
        epilie     epiNeur    bfagree   bfcon     bfext
epilie  "1***"     "-0.25***" "0.17*"   "0.23**"  "-0.04 "
epiNeur "-0.25***" "1***"     "-0.08 "  "-0.13 "  "-0.17*"
bfagree "0.17**"   "-0.08 "   "1***"    "0.45***" "0.48***"
bfcon   "0.23***"  "-0.13*"   "0.45***" "1***"    "0.27***"
bfext   "-0.04 "   "-0.17**"  "0.48***" "0.27***" "1***"
```

Raw probabilities below diagonal, Holm corrected above the diagonal.

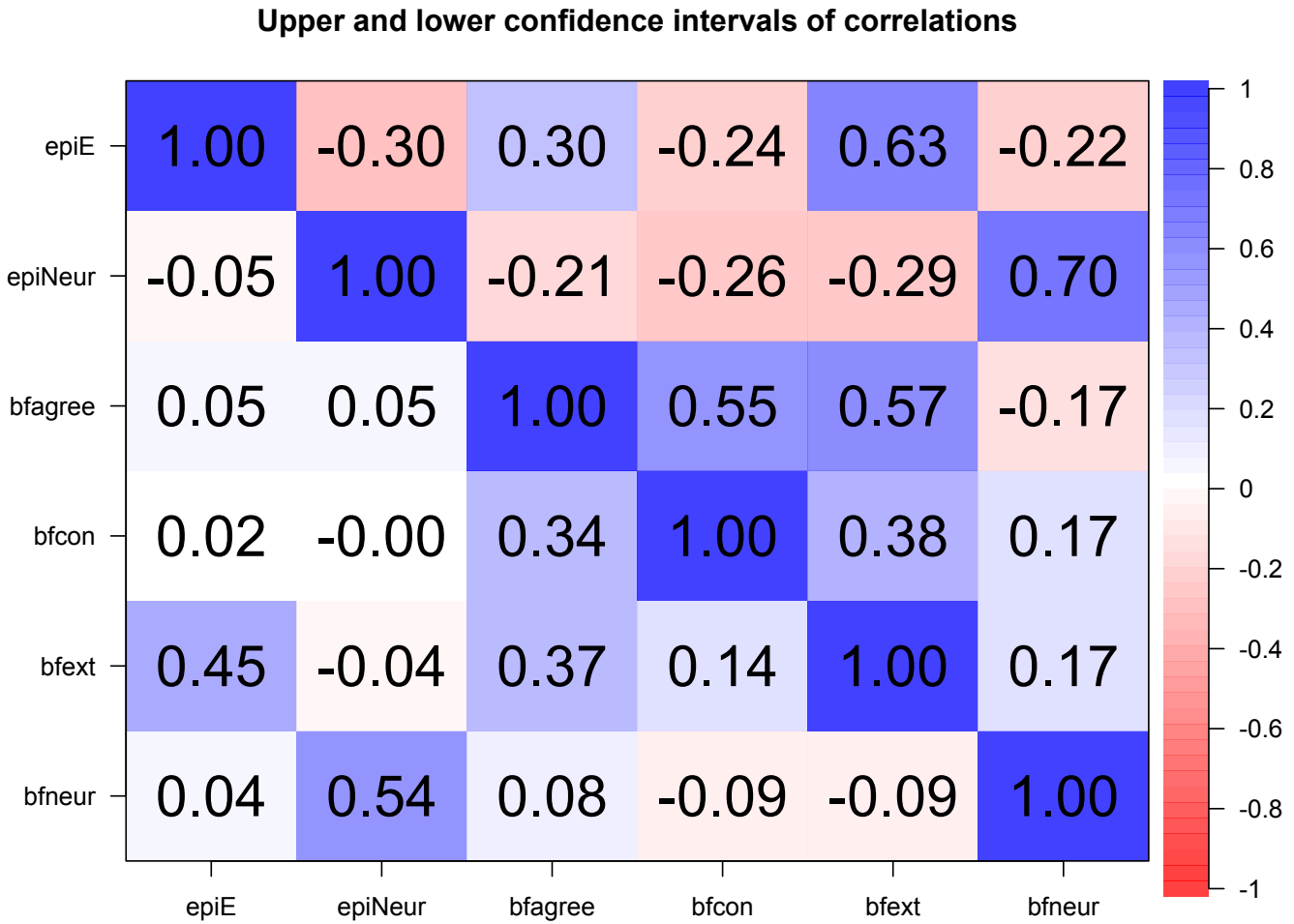
corCi of epi.bfi

R code

```
corCi(epi.bfi[c(1,5,6:9)])
corPlotUpperLowerCi(epi.bfi[c(1,5,6:9)])
```

```
Call:corCi(x = epi.bfi[c(1, 5, 6:9)])

 Coefficients and bootstrapped confidence intervals
        epiE  epiNr bfagr bfcon bfext bfner
epiE     1.00
epiNeur -0.18  1.00
bfagree  0.18 -0.08  1.00
bfcon   -0.11 -0.13  0.45  1.00
bfext    0.54 -0.17  0.48  0.27  1.00
bfneur  -0.09  0.63 -0.04  0.04  0.04  1.00

 scale correlations and bootstrapped confidence intervals
            lower.emp lower.norm estimate upper.norm upper.emp    p
epiE-epiNr      -0.28      -0.30    -0.18      -0.06     -0.05 0.00
epiE-bfagr       0.08       0.07     0.18       0.29      0.29 0.00
epiE-bfcon      -0.25      -0.26    -0.11       0.03      0.03 0.12
epiE-bfext       0.47       0.46     0.54       0.63      0.61 0.00
epiE-bfner      -0.22      -0.24    -0.09       0.06      0.07 0.24
epiNr-bfagr     -0.19      -0.21    -0.08       0.06      0.06 0.26
epiNr-bfcon     -0.24      -0.24    -0.13      -0.03     -0.04 0.02
epiNr-bfext     -0.28      -0.28    -0.17      -0.05     -0.05 0.01
epiNr-bfner      0.56       0.54     0.63       0.70      0.71 0.00
bfagr-bfcon      0.35       0.34     0.45       0.55      0.54 0.00
bfagr-bfext      0.37       0.37     0.48       0.57      0.56 0.00
bfagr-bfner     -0.19      -0.18    -0.04       0.11      0.11 0.63
bfcon-bfext      0.13       0.14     0.27       0.38      0.38 0.00
bfcon-bfner     -0.09      -0.10     0.04       0.18      0.19 0.58
bfext-bfner     -0.10      -0.09     0.04       0.19      0.18 0.51
```

corPlotUpperLowerCi



Abdi, H. (2010). Holm's sequential Bonferroni procedure. *Encyclopedia of research design*, *1*(8), 1–8.

Bonferroni, C. (1936). Teoria statistica delle classi e calcolo delle probabilita. *Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commericiali di Firenze*, *8*, 3–62.

Efron, B. (1979). Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, *7*(1-26).

Efron, B. & Gong, G. (1983). A leisurely look at the bootstrap, the jackknife, and cross-validation. *The American Statistician*, *37*(1), 36–48.

Galton, F. (1886). Regression towards mediocrity in hereditary stature. *Journal of the Anthropological Institute of Great Britain and Ireland*, *15*, 246–263.

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, *6*(2), pp. 65–70.

Revelle, W. (2015a). Charles Spearman. In R. L. Cautin & S. O. Lilienfeld (Eds.), *The Encyclopedia of Clinical Psychology*. John Wiley & Sons Inc.

Revelle, W. (2015b). Francis Galton. In R. L. Cautin & S. O. Lilienfeld (Eds.), *The Encyclopedia of Clinical Psychology*. John Wiley & Sons, Inc.