Lab 2 - Reading and Exploring Data

P. Johnson, L. Shaw, B. Rogers

Wednesday, February 4, 2015

Office hours

- Homework-a-palooza on Mondays 11:30a - 1:30p in Fraser 458
- Tuesday 9:30a - 11:00a Dr. Johnson at the CRMDA
- Tuesday 1:30p - 3:00p Ben's office Blake 311
- Wednesday 10:00a - 11:00a Leslie's office Fraser 10
- Thursday 3:00p - 5:00p Dr. Johnson at the CRMDA
- Friday 1:30p - 3:00p Repeat lab Blake 114

First homework due Feb. 11 at 5pm

- You can get half credit by attempting to answer each question, whether correct or not.
  - if asked for code, provide code
  - if asked for a table, provide table
  - if asked for plot, provide a plot
  - etc.
- If you minimally answer each question correctly, you can get a B
- You need to demonstrate a deeper understanding in your answer to get full points

To do

- Accessing internal data sets
- Creating data frames
- Reading and writing data
- Accessing specific entries in data
- Getting to know your data
- Plots

Internal data sets

```
install.packages("UsingR") # install UsingR package
```

```
library(UsingR) # load the package to R session
```

```
## Warning: package 'UsingR' was built under R version 3.0.3
```

```
## Loading required package: MASS
## Loading required package: HistData
```

```
## Warning: package 'HistData' was built under R version 3.0.3
```

```
## Loading required package: Hmisc
```

```
## Warning: package 'Hmisc' was built under R version 3.0.3
```

```
## Loading required package: grid
## Loading required package: lattice
## Loading required package: survival
## Loading required package: splines
## Loading required package: Formula
```

Look at objects in workspace

```
help(package="UsingR") # take a look at what's in the package
ls()   # look at the list of objects in the workspace
```

```
## [1] "h1"     "mydata"
```

- Do you see more than just the babies data set?
- Maybe you need to clear your workspace

```
rm(list = ls())
```

Looking at whole data set

```
babies  # simply type the name of the data frame
```

```
##        id pluralty outcome date gestation sex  wt parity race age ed ht
## 1      15        5       1 1411       284   1 120      1    8  27  5 62
## 2      20        5       1 1499       282   1 113      2    0  33  5 64
## 3      58        5       1 1576       279   1 128      1    0  28  2 64
## 4      61        5       1 1504       999   1 123      2    0  36  5 69
## 5      72        5       1 1425       282   1 108      1    0  23  5 67
## 6     100        5       1 1673       286   1 136      4    0  25  2 62
## 7     102        5       1 1449       244   1 138      4    7  33  2 62
## 8     129        5       1 1562       245   1 132      2    7  23  1 65
## 9     142        5       1 1408       289   1 120      3    0  25  4 62
## 10    148        5       1 1568       299   1 143      3    0  30  5 66
## 11    164        5       1 1554       351   1 140      2    0  27  5 68
## 12    171        5       1 1593       282   1 144      4    0  32  2 64
## 13    175        5       1 1491       279   1 141      3    0  23  1 63
## 14    183        5       1 1446       281   1 110      5    8  36  5 61
## 15    194        5       1 1524       273   1 114      3    7  30  2 63
## 16    195        5       1 1501       285   1 115      4    7  38  2 63
## 17    207        5       1 1481       255   1  92      3    4  25  7 65
## 18    217        5       1 1605       261   1 115      3    3  33  2 60
## 19    239        5       1 1431       261   1 144      2    0  33  2 68
```

Look at first 6 rows

- Use ?head to find out how to see more or less rows

```
head(babies) # show the first 6 rows
```

```
##    id pluralty outcome date gestation sex  wt parity race age ed ht wt1
## 1  15        5       1 1411       284   1 120      1    8  27  5 62 100
## 2  20        5       1 1499       282   1 113      2    0  33  5 64 135
## 3  58        5       1 1576       279   1 128      1    0  28  2 64 115
## 4  61        5       1 1504       999   1 123      2    0  36  5 69 190
## 5  72        5       1 1425       282   1 108      1    0  23  5 67 125
## 6 100        5       1 1673       286   1 136      4    0  25  2 62  93
##   drace dage ded dht dwt marital inc smoke time number
## 1     8   31   5  65 110       1   1     0    0      0
## 2     0   38   5  70 148       1   4     0    0      0
## 3     5   32   1  99 999       1   2     1    1      1
## 4     3   43   4  68 197       1   8     3    5      5
## 5     0   24   5  99 999       1   1     1    1      5
## 6     3   28   2  64 130       1   4     2    2      2
```

Look at first 6 rows

- Use colnames or rownames function so you know how to refer to your data

```
colnames(babies) # returns column names. Also see rownames()
```

```
##  [1] "id"        "pluralty"  "outcome"   "date"      "gestation"
##  [6] "sex"       "wt"        "parity"    "race"      "age"
## [11] "ed"        "ht"        "wt1"       "drace"     "dage"
## [16] "ded"       "dht"       "dwt"       "marital"   "inc"
## [21] "smoke"     "time"      "number"
```

Creating dataframes

- In lab 1 we saw how to create vectors
- Sometimes we want to put vectors together into a dataframe, a two dimensional object

```
x <- c(100, 95, 93, 97) # define a data vector (variable) x
y <- c(95, 98, 86, 91) # define y
dat <- data.frame(x, y) # create a data frame, cols are variables
## You can see dat in your Workspace now
dat
```

```
##     x  y
## 1 100 95
## 2  95 98
## 3  93 86
## 4  97 91
```

Working directory

- If you launch your R editor with the R file you are going to use and the data that you will use is in that same directory, then R will treat that folder as your working directory.
- Check your working directory and change if needed

```
getwd()
```

```
## [1] "D:/Users/l076s857/Dropbox/GTA706"
```

```
# see lab 1 for examples on setwd()
setwd()
```

Reading data into R

- There are many ways to read data into R
- We will use read.table() or variations on that function in labs
- All code will assume data is in the same folder as our R file

```
# Check out ?read.table
mydata1 <- read.table(file = "job1.txt")
```

Reading a comma-delimited file

- Instead of read.table(), we will use read.csv()
- Note the column names that we get by setting header = TRUE

```
mydata <- read.csv(file = "job.csv", header = TRUE)
head(mydata)
```

```
##   ID AGE TENURE FEMALE WBEING SATIS JOBPERF TURNOVER  IQ
## 1  1  40     10      1      8     8       6        0 106
## 2  2  53     14      1      6     5       5        0  93
## 3  3  46     10      1      7     7       7        0 107
## 4  4  37      8      1      7     5       5        0  94
## 5  5  44      9      1      8     5       5        0 107
## 6  6  39     10      1      7     6       7        0 118
```

Referencing specific entries or rows

```
mydata[1, 2]  # the entry at row 1 and column 2
```

```
## [1] 40
```

```
mydata[1:10, ] # the first ten rows (all columns)
```

```
##    ID AGE TENURE FEMALE WBEING SATIS JOBPERF TURNOVER  IQ
## 1   1  40     10      1      8     8       6         0 106
## 2   2  53     14      1      6     5       5         0  93
## 3   3  46     10      1      7     7       7         0 107
## 4   4  37      8      1      7     5       5         0  94
## 5   5  44      9      1      8     5       5         0 107
## 6   6  39     10      1      7     6       7         0 118
## 7   7  33      7      1      7     5       7         0 103
## 8   8  43      9      1      7     7       7         0 106
## 9   9  35      9      1      7     7       7         1 108
## 10 10  37     10      1      5     6       6         0  97
```

See two columns

```
mydata[ , 4:5] # the 4th and 5th columns
```

```
##     FEMALE WBEING
## 1        1      8
## 2        1      6
## 3        1      7
## 4        1      7
## 5        1      8
## 6        1      7
## 7        1      7
## 8        1      7
## 9        1      7
## 10       1      5
## 11       1      5
## 12       1      7
## 13       1      6
## 14       1      6
## 15       1      5
## 16       1      5
## 17       1      6
## 18       1      6
## 19       1      6
```

See specific set of rows and columns

```
mydata[c(2, 3, 5), 1:3] # rows 2, 3, and 5 of columns 1-3
```

```
##   ID AGE TENURE
## 2  2  53     14
## 3  3  46     10
## 5  5  44      9
```

See all data except for some rows

```
mydata[-1:-400, ] # all data except for the first 400 rows
```

```
##       ID AGE TENURE FEMALE WBEING SATIS JOBPERF TURNOVER  IQ
## 401 401  41      6      0      6     5       7         0 107
## 402 402  43     11      0      6     6       6         0 103
## 403 403  42     13      0      7     4       7         0  98
## 404 404  36      8      0      5     8       5         0 108
## 405 405  37      9      0      7     6       7         0  99
## 406 406  28      9      0      6     5       5         1  89
## 407 407  40     12      0      6     6       5         0  94
## 408 408  36      7      0      6     4       5         0  90
## 409 409  41     19      0      7     5       5         1  96
## 410 410  40      9      0     10     8       9         0 113
## 411 411  45      8      0      6     4       6         1  85
## 412 412  32      8      0      6     3       8         0  94
## 413 413  44     15      0      8     7       4         1 102
## 414 414  35     11      0      7     6       7         0  93
## 415 415  45     13      0      6     5       7         0  90
## 416 416  40      9      0      5     5       5         0  93
## 417 417  36     11      0      5     4       7         1 104
## 418 418  36     11      0      4     8       4         0  96
## 419 419  49     11      0      7     8       6         0 103
```

Refer to columns by name

```
mydata["1" , "AGE"]
```

```
## [1] 40
```

```
mydata[1:20, c("IQ", "AGE", "WBEING", "SATIS")]
```

```
##     IQ AGE WBEING SATIS
## 1  106  40      8     8
## 2   93  53      6     5
## 3  107  46      7     7
## 4   94  37      7     5
## 5  107  44      8     5
## 6  118  39      7     6
## 7  103  33      7     5
## 8  106  43      7     7
## 9  108  35      7     7
## 10  97  37      5     6
## 11 100  35      5     8
## 12 112  45      7     6
## 13 102  38      6     7
## 14  97  38      6     6
## 15  80  25      5     5
```

Look at a subset based on a factor variable

```
mydata[mydata$FEMALE == 0, ]  # select data for males
```

```
##      ID AGE TENURE FEMALE WBEING SATIS JOBPERF TURNOVER  IQ
## 261 261  44     12      0      6     7       8         0  96
## 262 262  31      7      0      6     5       5         1  93
## 263 263  42      8      0      8     6       7         0 109
## 264 264  43     11      0      5     5       4         0  94
## 265 265  46     10      0      4     5       4         0  99
## 266 266  36      8      0      4     7       8         0  97
## 267 267  36      7      0      6     5       6         1  95
## 268 268  43     10      0      7     8       6         0 116
## 269 269  36      7      0      7     6       5         0 100
## 270 270  30      7      0      4     4       7         1 104
## 271 271  50     14      0      7     6       6         0  93
## 272 272  42     10      0      9     6       6         0 109
## 273 273  42     10      0      8     7       6         0 107
## 274 274  40      7      0      5     5       3         1  90
## 275 275  40      8      0      4     5       6         1 105
## 276 276  33     14      0      6     6       6         1  97
## 277 277  45     10      0      6     6       7         0 100
## 278 278  40      7      0      4     4       6         1  99
## 279 279  28      5      0      7     6       8         0 106
```

Look at a subset based on a continuous value

```
mydata[mydata$AGE > 50, ]
```

```
##       ID AGE TENURE FEMALE WBEING SATIS JOBPERF TURNOVER  IQ
## 2      2  53     14      1      6     5       5        0  93
## 47    47  52     16      1      7     5       5        1  93
## 251 251  51     11      1      7     6       7        1 104
```

Reference the IQ column

```
mydata$IQ
```

```
##   [1] 106  93 107  94 107 118 103 106 108  97 100 112 102  97  89 104  90
##  [18]  96  97 102 102 110 113 102 108  93 104  96 100 105  91 103  93  98
##  [35] 111  91  94 102  94  79 110  89  91 104  95 101  93 113  92 112 106
##  [52] 117 102 112 103 103 103 110 112  96 108 102  95 104 107  97  84  97
##  [69] 105 102 110 109  99  96  98  93 101  92  95 111 103  98  98  98  93
##  [86] 101 101  98 101 100  98 124 107  95  90 105 101 105 112  95 107  98
## [103]  95  98 111  99 103  86  95  99 102 112 108 113 111  94  95 104  99
## [120]  87 106 101 110  93  87  94 106  93  97 108  99  97  71  86 108 116
## [137] 125  80 103 110 100 116 100  99 110  96 104 109 103  93  96  94 108
## [154]  89 101 105  94 102 117 101  96  96  95  95 103  99  99 114 105  95
## [171] 109  99 101 101  93  73 119 105 112  84 105  98  79 108 113 105  94
## [188] 109  90  87 104 103 104  85 107  98 107  89 114  89  90  97 109 101
## [205] 112 103  99  92  92 101 102  96 116  93  96 114 106 118  86  99 101
## [222]  98  93 101  97 102  99 104 104  90  95  93  98  94 109  96  99  88
## [239] 109 105  94 115 103 102  96 105 101  96  98  96 104  98  99  94  99
## [256] 103 112 100 118 100  96  93 109  94  99  97  95 116 100 104  93 109
## [273] 107  90 105  97 100  99 106  99 120  97 100  88  93 101 111 104  82
## [290]  98  92 109  78 107  89 105 101 104  98  91 105  88  96 102  96  97
## [307]  99  91 110 102  99  96  87  93  98  92  98 102  96 102  99  89 120
## [324]  98  94  99  99  91  94 101 116  92 104  98 116 106 104 107  96 100
```

Size of data set and basic statistics

```
dim(mydata) # returns dimensions of a data frame
```

```
## [1] 480   9
```

```
summary(mydata) # a summary of each variable
```

```
##        ID              AGE            TENURE          FEMALE
##  Min.   :  1.0   Min.   :18.00   Min.   : 1.00   Min.   :0.0000
##  1st Qu.:120.8   1st Qu.:34.00   1st Qu.: 8.00   1st Qu.:0.0000
##  Median :240.5   Median :38.00   Median :10.00   Median :1.0000
##  Mean   :240.5   Mean   :37.95   Mean   :10.05   Mean   :0.5417
##  3rd Qu.:360.2   3rd Qu.:42.00   3rd Qu.:12.00   3rd Qu.:1.0000
##  Max.   :480.0   Max.   :53.00   Max.   :21.00   Max.   :1.0000
##      WBEING           SATIS         JOBPERF          TURNOVER
##  Min.   : 3.000   Min.   :3.00   Min.   : 3.000   Min.   :0.0000
##  1st Qu.: 5.000   1st Qu.:5.00   1st Qu.: 5.000   1st Qu.:0.0000
##  Median : 6.000   Median :6.00   Median : 6.000   Median :0.0000
##  Mean   : 6.271   Mean   :5.99   Mean   : 6.021   Mean   :0.3208
##  3rd Qu.: 7.000   3rd Qu.:7.00   3rd Qu.: 7.000   3rd Qu.:1.0000
##  Max.   :10.000   Max.   :9.00   Max.   :10.000   Max.   :1.0000
##        IQ
##  Min.   : 71.00
```

Look at a summary of 4 columns

```
dim(mydata[ , c("IQ", "AGE", "WBEING", "SATIS")])
```

```
## [1] 480   4
```

```
summary(mydata[ , c("IQ", "AGE", "WBEING", "SATIS")])
```

```
##        IQ              AGE            WBEING          SATIS      
##  Min.   : 71.00   Min.   :18.00   Min.   : 3.000   Min.   :3.00  
##  1st Qu.: 94.75   1st Qu.:34.00   1st Qu.: 5.000   1st Qu.:5.00  
##  Median :100.00   Median :38.00   Median : 6.000   Median :6.00  
##  Mean   :100.10   Mean   :37.95   Mean   : 6.271   Mean   :5.99  
##  3rd Qu.:105.25   3rd Qu.:42.00   3rd Qu.: 7.000   3rd Qu.:7.00  
##  Max.   :125.00   Max.   :53.00   Max.   :10.000   Max.   :9.00
```

Frequency tables

```
table(mydata$FEMALE, mydata$JOBPERF)
```

```
##
##      3  4  5  6  7  8  9 10
##   0  4 16 48 80 48 19  5  0
##   1  9 22 55 84 58 28  3  1
```

Covariance and correlation

```
cov(mydata$AGE, mydata$JOBPERF)
```

```
## [1] -0.3308542
```

```
cor(mydata$AGE, mydata$JOBPERF)
```

```
## [1] -0.0490026
```

Functions for vectors applied to a dataframe

- All the one-dimensional descriptive functions apply here if we reference one column

```
mean(mydata$IQ)
```

```
## [1] 100.1021
```

```
sd(mydata$IQ)
```

```
## [1] 8.428503
```

```
range(mydata$IQ)
```

```
## [1]  71 125
```

Create variable and add to dataframe

- We can create a new variable in the data frame by directly assigning values to it.

```
mydata$IQ.centered <- mydata$IQ - mean(mydata$IQ)
summary(mydata)
```

```
##        ID              AGE            TENURE          FEMALE
##  Min.   :  1.0   Min.   :18.00   Min.   : 1.00   Min.   :0.0000
##  1st Qu.:120.8   1st Qu.:34.00   1st Qu.: 8.00   1st Qu.:0.0000
##  Median :240.5   Median :38.00   Median :10.00   Median :1.0000
##  Mean   :240.5   Mean   :37.95   Mean   :10.05   Mean   :0.5417
##  3rd Qu.:360.2   3rd Qu.:42.00   3rd Qu.:12.00   3rd Qu.:1.0000
##  Max.   :480.0   Max.   :53.00   Max.   :21.00   Max.   :1.0000
##      WBEING           SATIS         JOBPERF          TURNOVER
##  Min.   : 3.000   Min.   :3.00   Min.   : 3.000   Min.   :0.0000
##  1st Qu.: 5.000   1st Qu.:5.00   1st Qu.: 5.000   1st Qu.:0.0000
##  Median : 6.000   Median :6.00   Median : 6.000   Median :0.0000
##  Mean   : 6.271   Mean   :5.99   Mean   : 6.021   Mean   :0.3208
##  3rd Qu.: 7.000   3rd Qu.:7.00   3rd Qu.: 7.000   3rd Qu.:1.0000
##  Max.   :10.000   Max.   :9.00   Max.   :10.000   Max.   :1.0000
##        IQ          IQ.centered
##  Min.   : 71.00   Min.   :-29.1021
```

Factor variables in R

When reading in data using data.frame() or read.table()/ read.csv(), R guesses the types of variables. By default, numeric variables will still be numeric, while character data and logical data will be the translated into factors (categorical variables).

But sometimes we need to tell R specifically that some of the numeric variables are actually factors.

The factor() function converts a numeric variable into a factor. The labels= argument specifies labels for the factor levels.

Telling R about a factor

- Create a new variable
- Look at the levels

```
mydata$FEMALE.f <- factor(mydata$FEMALE)
levels(mydata$FEMALE.f) # look at the catagories (aka levels)
```

```
## [1] "0" "1"
```

Add and change level information

- We can associate labels to numbers in our factor
- Labels need to be assigned in order that R sees the levels (alpha-numeric)
- Note that in the data male = 0 and female = 1

```
mydata$FEMALE.f <- factor(mydata$FEMALE, labels=c("Male", "Female"))
levels(mydata$FEMALE.f)
```

```
## [1] "Male"   "Female"
```

```
mydata$TURNOVER.f <- factor(mydata$TURNOVER, labels=c("No", "Yes"))
levels(mydata$TURNOVER.f)
```

```
## [1] "No"  "Yes"
```

Look at the new summary

```
summary(mydata[ , c("FEMALE", "FEMALE.f", "TURNOVER", "TURNOVER.f")])
```

```
##      FEMALE          FEMALE.f      TURNOVER      TURNOVER.f
##  Min.   :0.0000   Male  :220   Min.   :0.0000   No :326
##  1st Qu.:0.0000   Female:260   1st Qu.:0.0000   Yes:154
##  Median :1.0000                Median :0.0000
##  Mean   :0.5417                Mean   :0.3208
##  3rd Qu.:1.0000                3rd Qu.:1.0000
##  Max.   :1.0000                Max.   :1.0000
```

Tired of typing mydata$ in front of your variable?

- For functions that don't have 'data =' as an option, use with() instead

```
# Equal to table(mydata$FEMALE, mydata$JOBPERF)
with(mydata, table(FEMALE, JOBPERF))
```

```
##       JOBPERF
## FEMALE  3  4  5  6  7  8  9 10
##      0  4 16 48 80 48 19  5  0
##      1  9 22 55 84 58 28  3  1
```

```
# Another example
with(mydata, cov(IQ, JOBPERF))
```

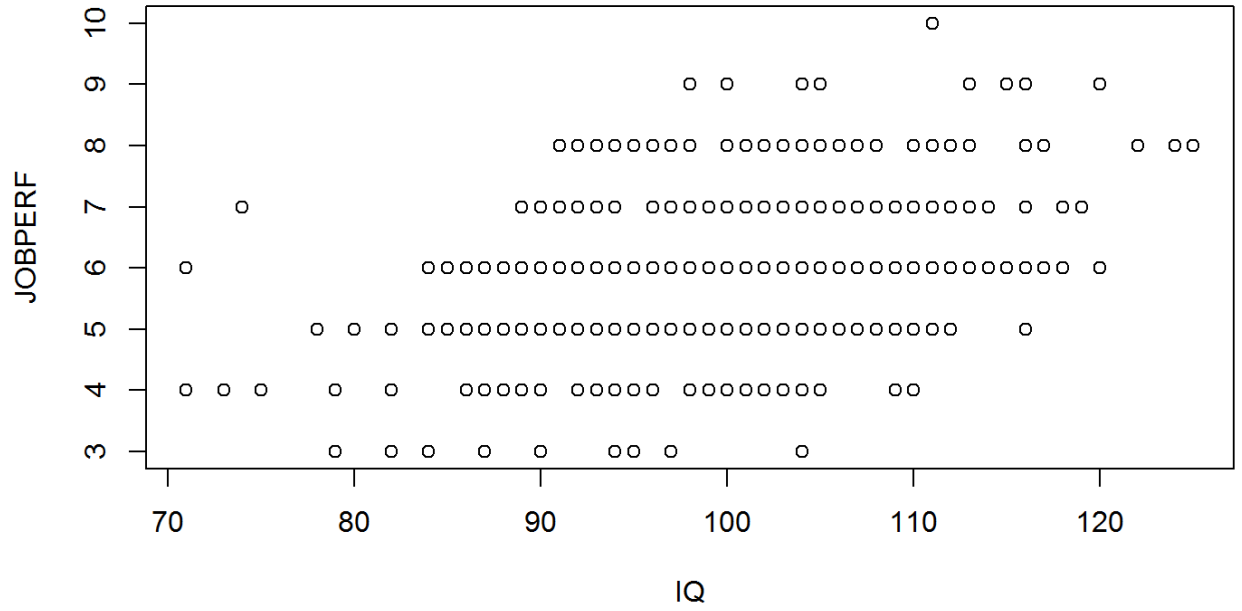
```
## [1] 4.505176
```

Plots

- Scatterplot
- Boxplot
- Crosstabs
- Bar plots

Basic scatterplot

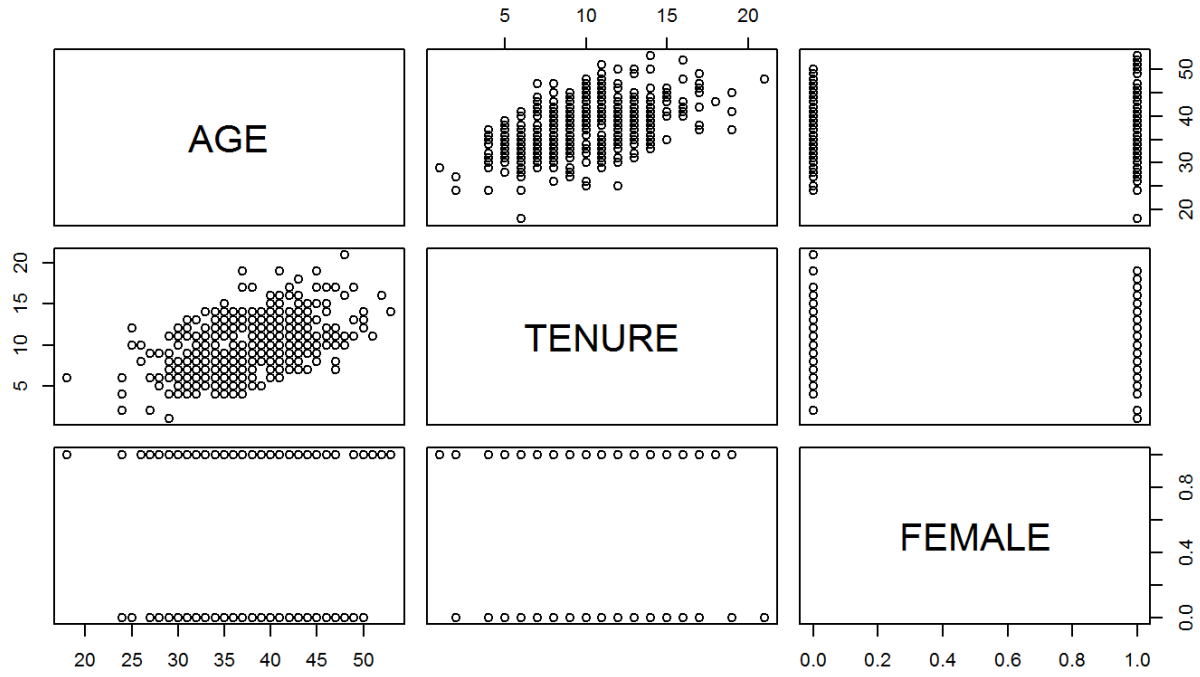
```
plot(JOBPERF ~ IQ, data = mydata)
```



Matrix of scatterplots

- Use all rows of cols 2, 3, 4

```
pairs(mydata[, c(2, 3, 4)])
```



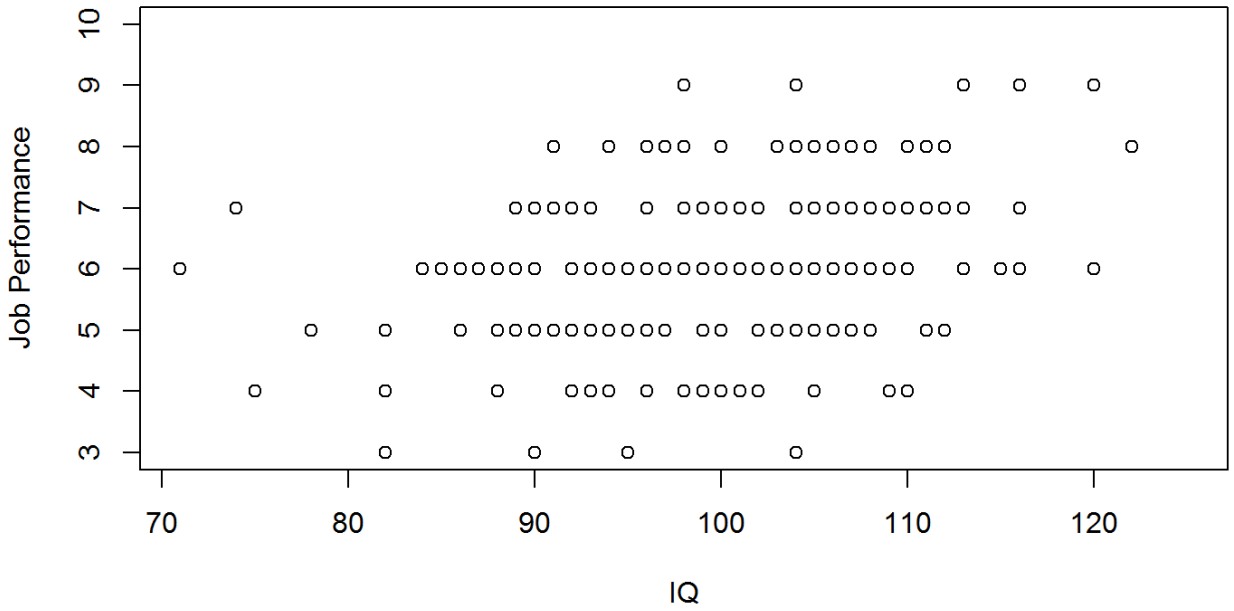
35/49

Create new dataframes separated by gender

```
mydata_m <- mydata[mydata$FEMALE==0, ]  # copy rows where FEMALE==0
mydata_f <- mydata[mydata$FEMALE==1, ]  # copy rows where FEMALE==1
```

Redraw scatterplot frame and then add points for males

```
plot(JOBPERF ~ IQ, data = mydata, xlab="IQ", ylab="Job Performance",
     type="n")
points(JOBPERF ~ IQ, data = mydata_m, col="black",
       pch=1) #  col= sets the color
```

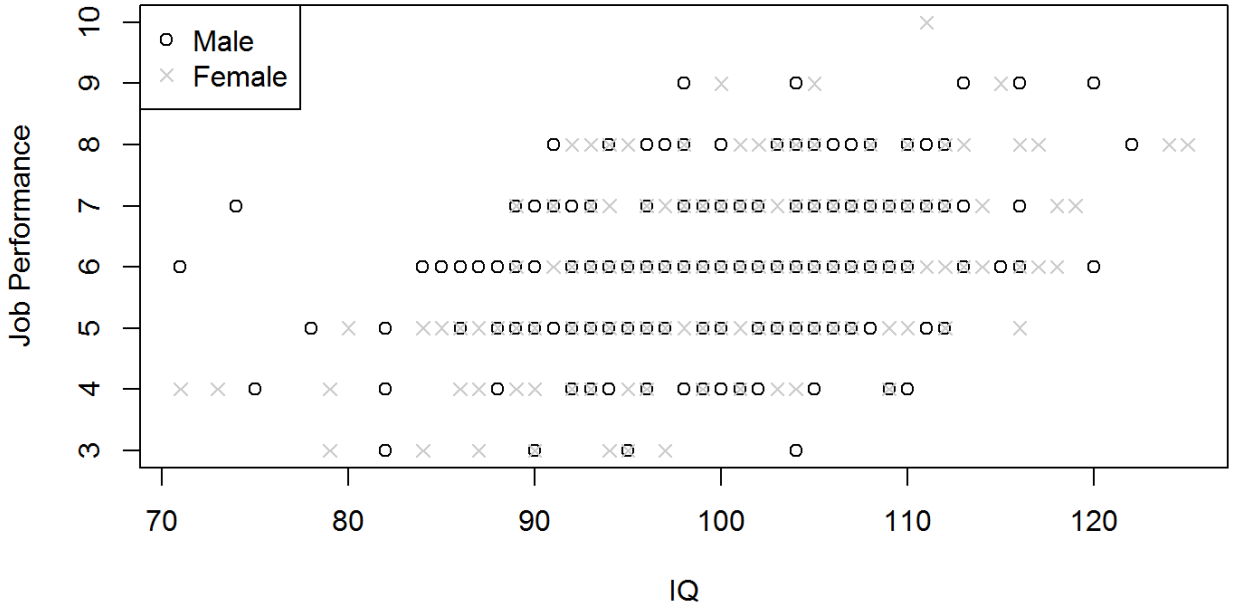


37/49

Syntax to add points for female and legend

```
plot(JOBPERF ~ IQ, data = mydata, xlab="IQ", ylab="Job Performance",
     type="n")
points(JOBPERF ~ IQ, data = mydata_m, col="black",
       pch=1) #  col= sets the color
points(JOBPERF ~ IQ, data = mydata_f, col="grey80",
       pch=4)  # pch= sets the shape
legend("topleft", legend=c("Male","Female"), col=c("black","grey80"),
       pch=c(1,4))
```

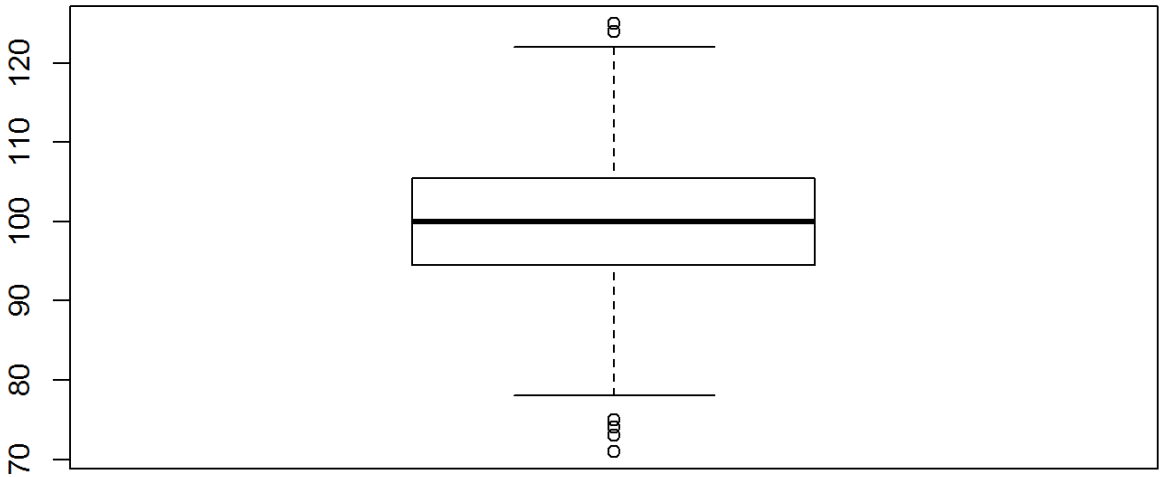
Syntax to add points for female and legend



39/49

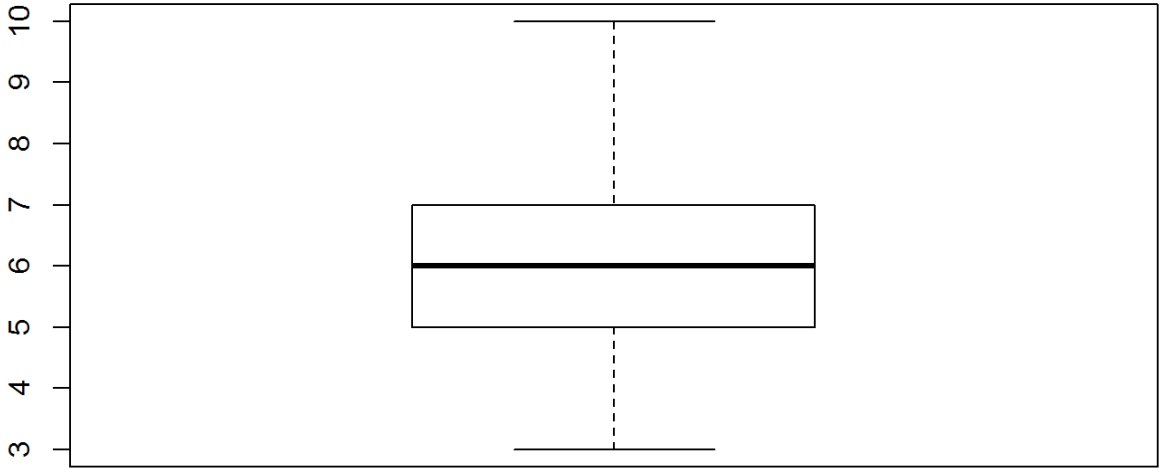
Boxplot of data with outliers

```
boxplot(mydata$IQ)
```



And another boxplot

```
boxplot(mydata$JOBPERF)
```



Boxplot of a subgroup with labels

```
boxplot(mydata$JOBPERF ~ mydata$FEMALE.f)  # accepts an equation
boxplot(mydata$JOBPERF ~ mydata$FEMALE.f, ylab="Job Performance",
        xlab="Gender", names=c("Male","Female"))  #  Add labels

# Equivalent to first plot above but cleaner to read
boxplot(JOBPERF ~ FEMALE.f, data=mydata)

# Commented out in lab code but try running it!
```

Cross-Tabulation

```
# Run this line if you need to install the package
install.packages("descr")
```

```
library(descr)
```

```
## Warning: package 'descr' was built under R version 3.0.3
```

```
myTable1 <- with(mydata, CrossTable(FEMALE.f, TURNOVER.f))
```

Chi-Square output

```
myTable1
```

```
##    Cell Contents
## |-------------------------|
## |                       N |
## | Chi-square contribution |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
## ================================
##              TURNOVER.f
## FEMALE.f        No     Yes   Total
## --------------------------------
## Male           150      70     220
##              0.002   0.005
##              0.682   0.318   0.458
##              0.460   0.455
##              0.312   0.146
## --------------------------------
## Female         176      84     260
```

Contingency table function

```
# Run this line if you need to install the package
install.packages("memisc")
```

```
library(memisc)
```

```
## Warning: package 'memisc' was built under R version 3.0.3
```

```
## Loading required namespace: car
##
## Attaching package: 'memisc'
##
## The following object is masked from 'package:Hmisc':
##
##     %nin%
##
## The following objects are masked from 'package:stats':
##
##     contr.sum, contr.treatment, contrasts
##
## The following object is masked from 'package:base':
##
##     as.array
```

Contingency table with counts and percents

```
myTable2 <- with(mydata, genTable(TURNOVER.f ~ FEMALE.f))
myTable2
```

```
##      FEMALE.f
##       Male Female
##   No   150    176
##   Yes   70     84
```

```
myTable3 <- with(mydata, genTable(percent(TURNOVER.f) ~ FEMALE.f))
myTable3
```

```
##      FEMALE.f
##            Male    Female
##   No   68.18182  67.69231
##   Yes  31.81818  32.30769
##   N   220.00000 260.00000
```

Barplot

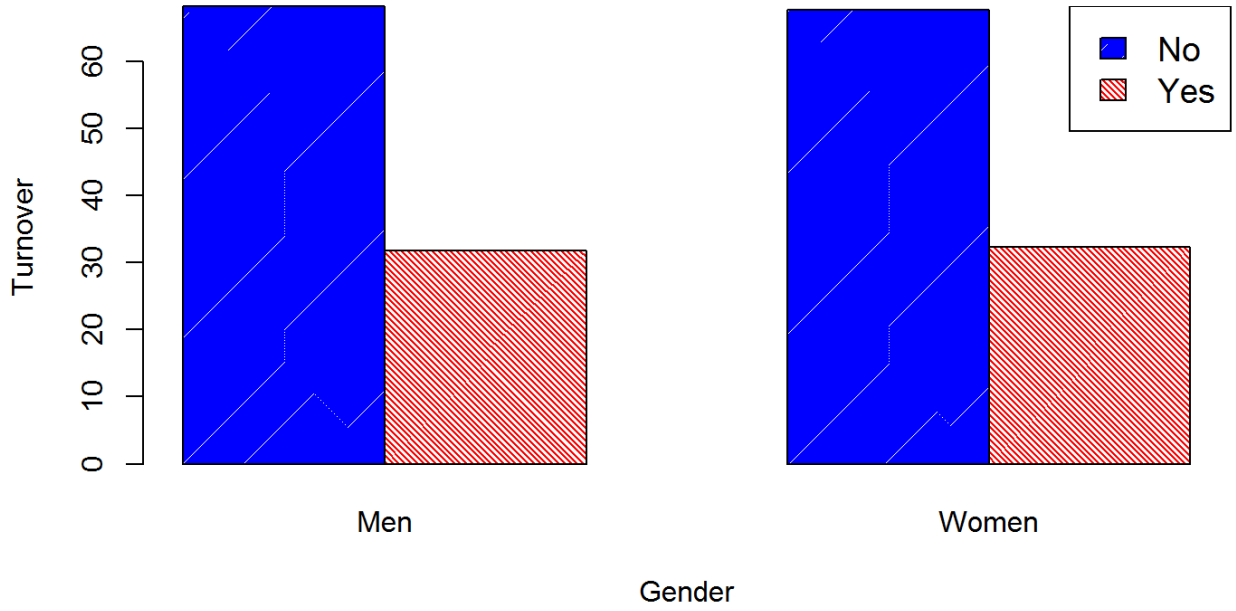
- A bar plot is a graphic presentation of a cross tabulation table.

```
barplot(myTable3[1:2,], beside=T, names=c("Men","Women"),
        col=c("blue","red"), density=c(90,40), angle=c(45,-45),
        ylab="Turnover", xlab="Gender")
legend("topright", legend = levels(mydata$TURNOVER.f),
       fill=c("blue","red"), density=c(90,40), angle = c(45,-45),
       cex=1.2)

# Note that barplot() uses "col=" but legend() uses "fill="
```

Barplot

- A bar plot is a graphic presentation of a cross tabulation table.



Exporting data

- Like read.table or read.csv there are corresponding write functions
- Double check your working directory using these functions

```
write.table(mydata, file="out.txt", sep="\t", row.names=FALSE,
            quote = FALSE)
write.csv(mydata, file="out.csv", row.names=FALSE, quote = FALSE)
```

- 'sep =' is used to denote how columns should be separated
- tab is used above; comma ',' and space ' ' are also common