Introduction to R: Graphics and Data Manipulation Tutorial

Dr Jeromy Anglim

Initialise Project

```
library(ProjectTemplate); load.project()
library(MASS)
data(survey)
csurvey <- na.omit(survey)
```

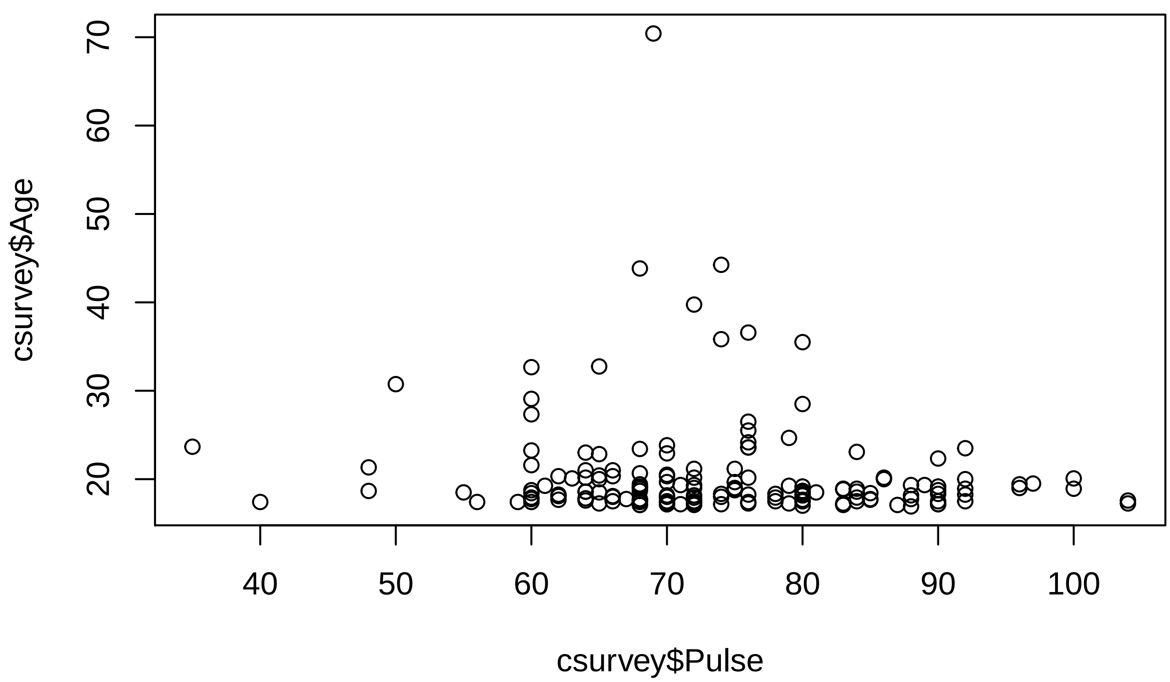
Graphics systesms

There are three main graphics packages in R:

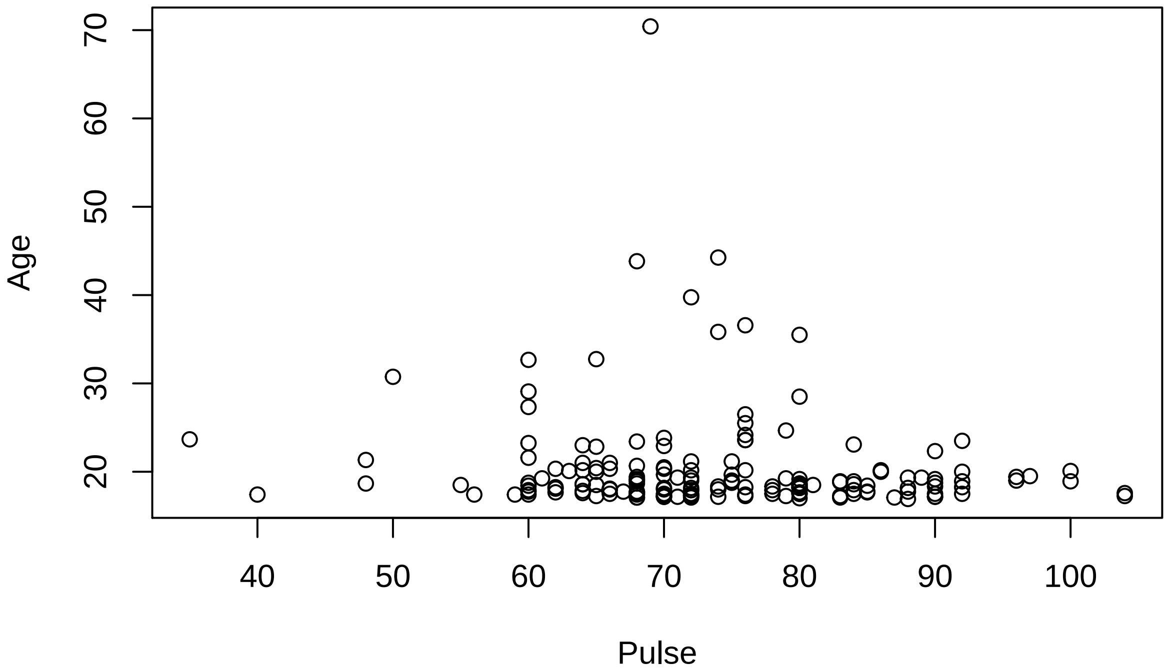
- Base graphics
- lattice
- ggplot2

Base graphics

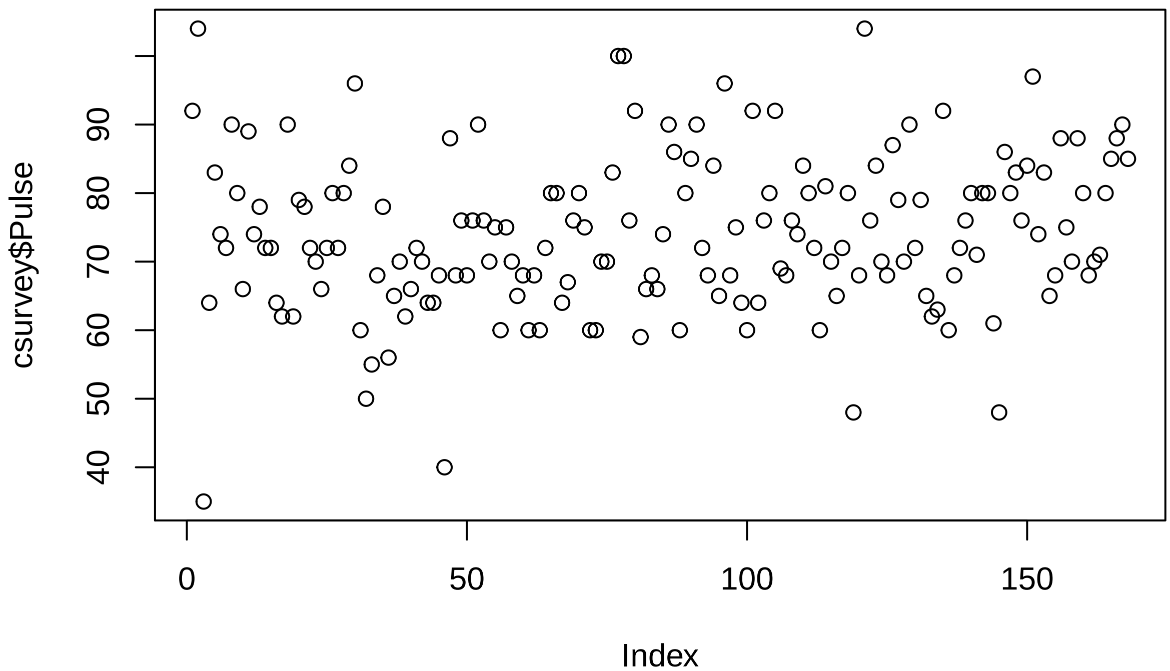
```
# scatterplot
plot(csurvey$Pulse, csurvey$Age)
```



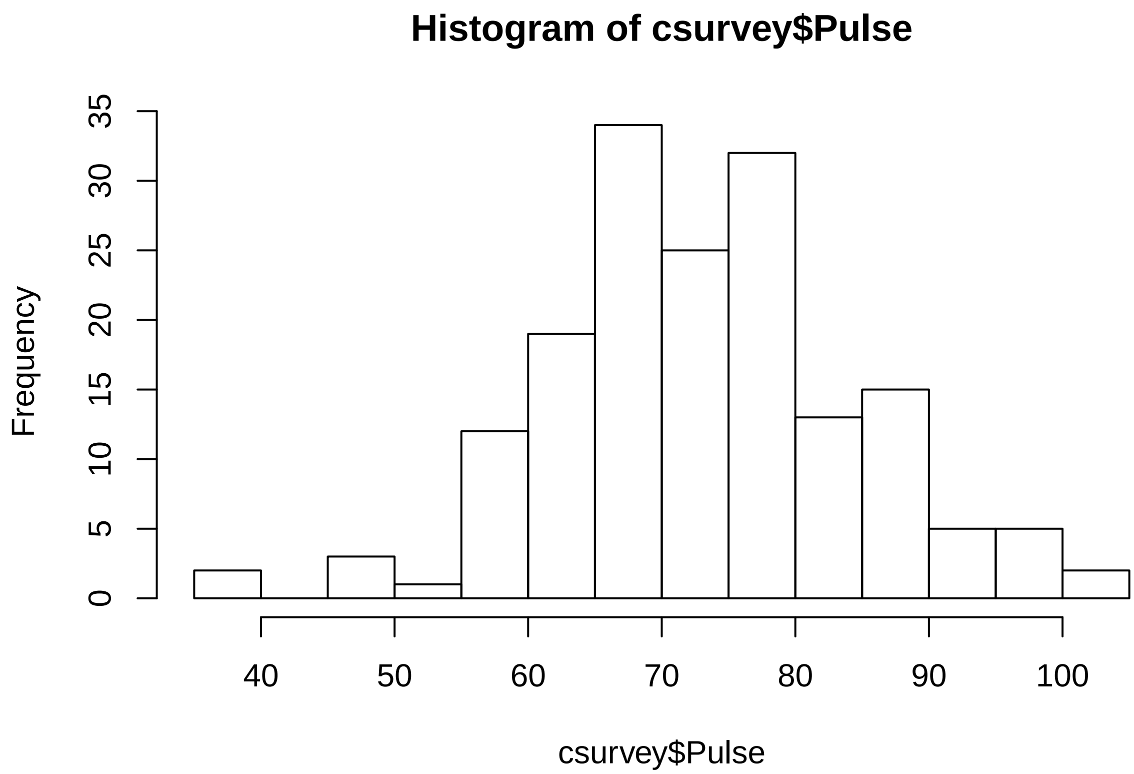
```r
plot(Age ~ Pulse, csurvey)
```




```r
plot(csurvey$Pulse)
```



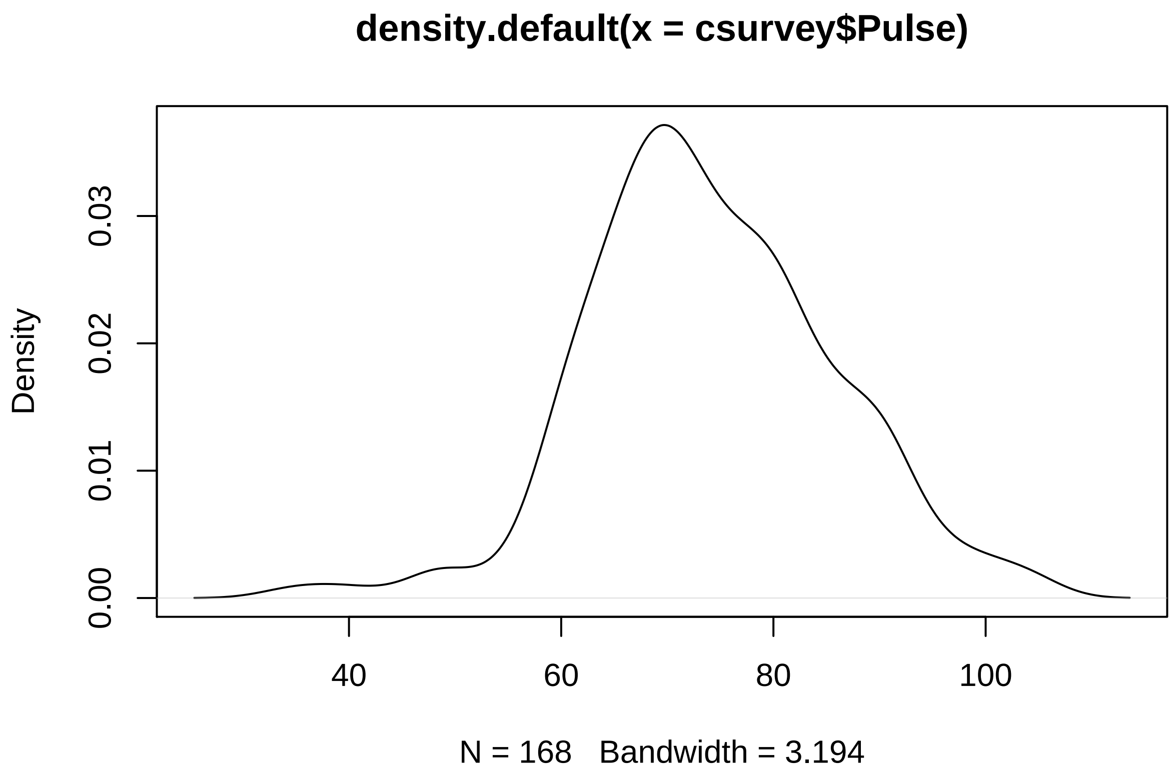

```r
# distribution
hist(csurvey$Pulse, 10) # histogram
```



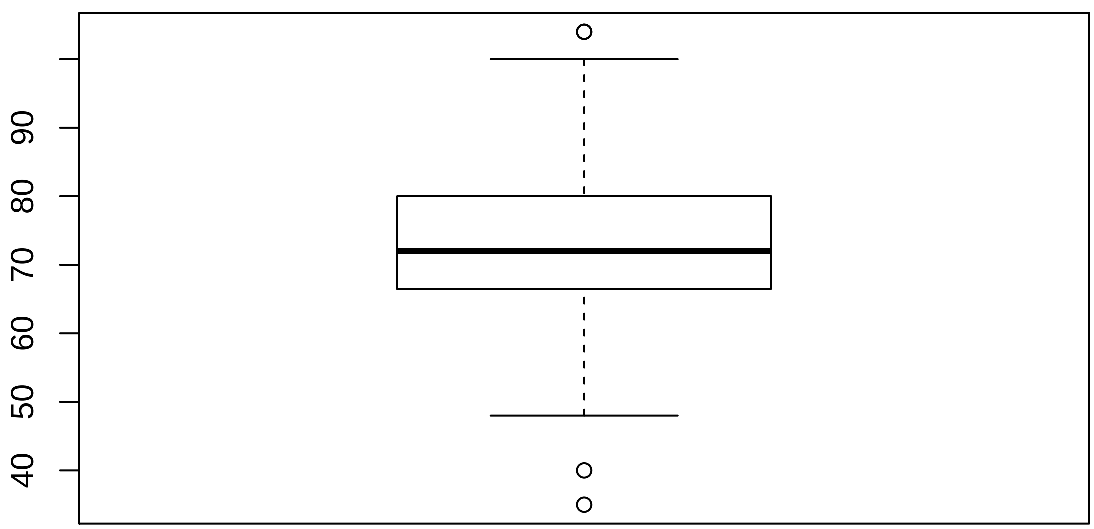

```r
stem(csurvey$Pulse) # Stem and leaf plot
```

```
##
##   The decimal point is 1 digit(s) to the right of the |
##
##    3 | 5
##    4 | 0
##    4 | 88
##    5 | 0
##    5 | 569
##    6 | 00000000001222234444444
##    6 | 55555566666788888888888888889
##    7 | 00000000000011222222222222244444
##    7 | 5555566666666666888999
##    8 | 00000000000000001333344444
##    8 | 55566788889
##    9 | 000000022222
##    9 | 667
##   10 | 0044
```

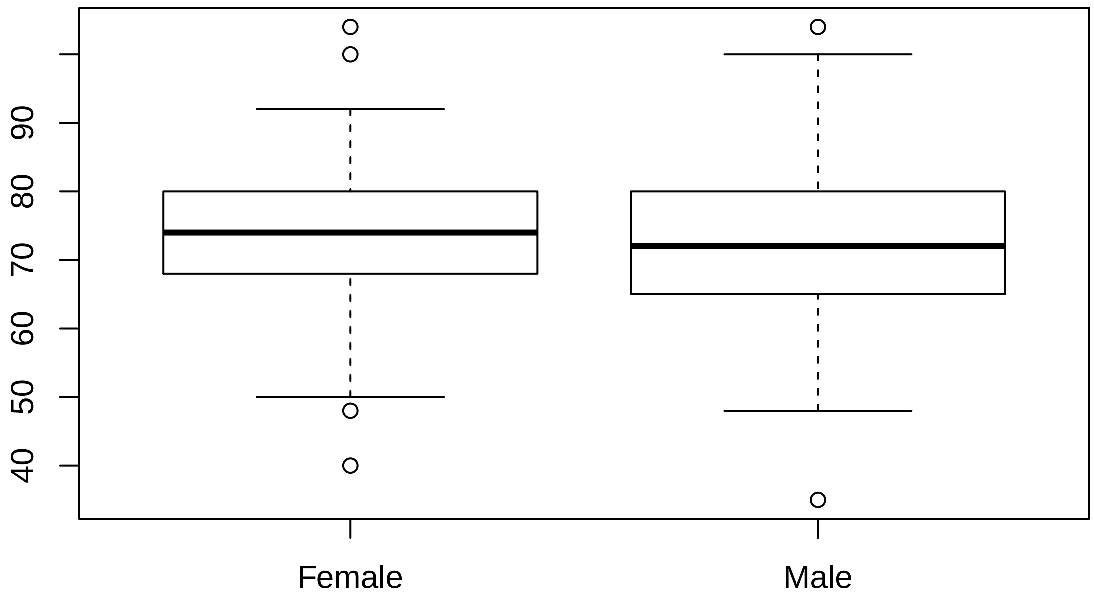
```r
plot(density(csurvey$Pulse)) # density plot
```



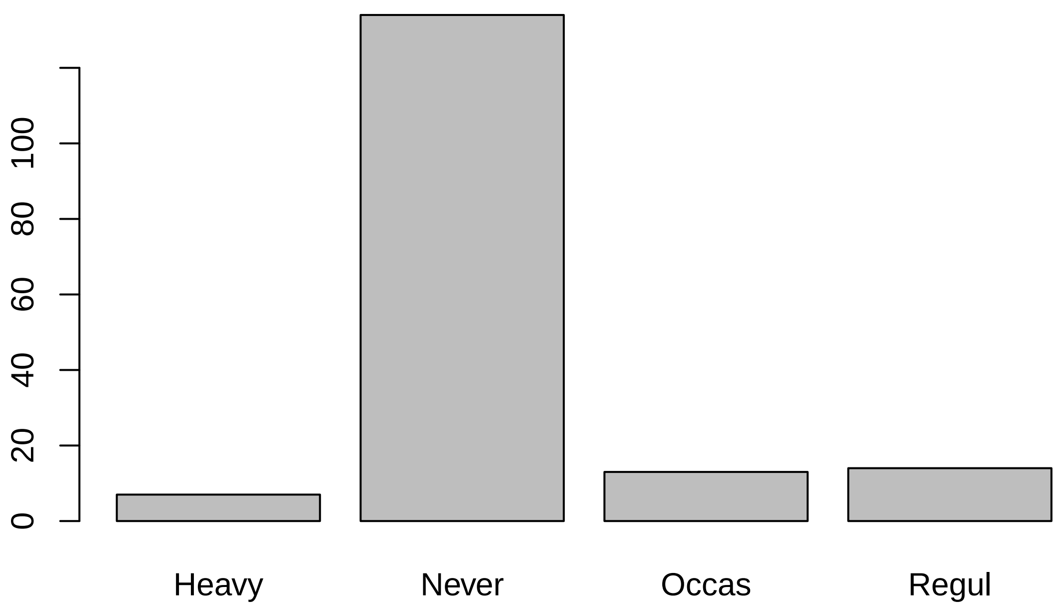

```r
boxplot(csurvey$Pulse) # box plot
```



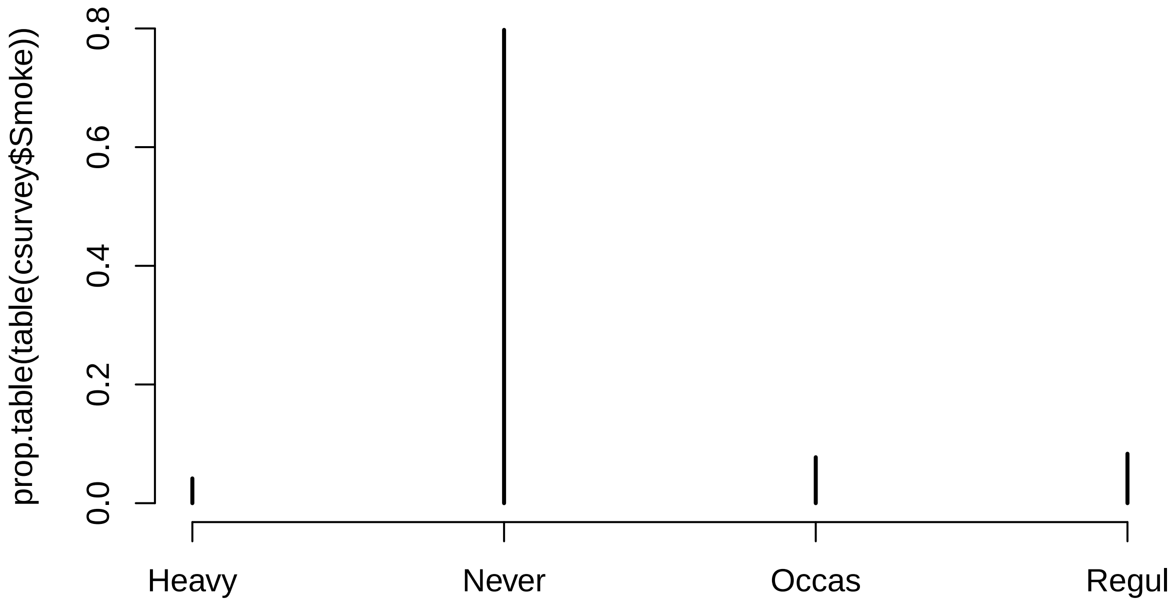

```r
boxplot(csurvey$Pulse ~ csurvey$Sex) # box plot by group
```



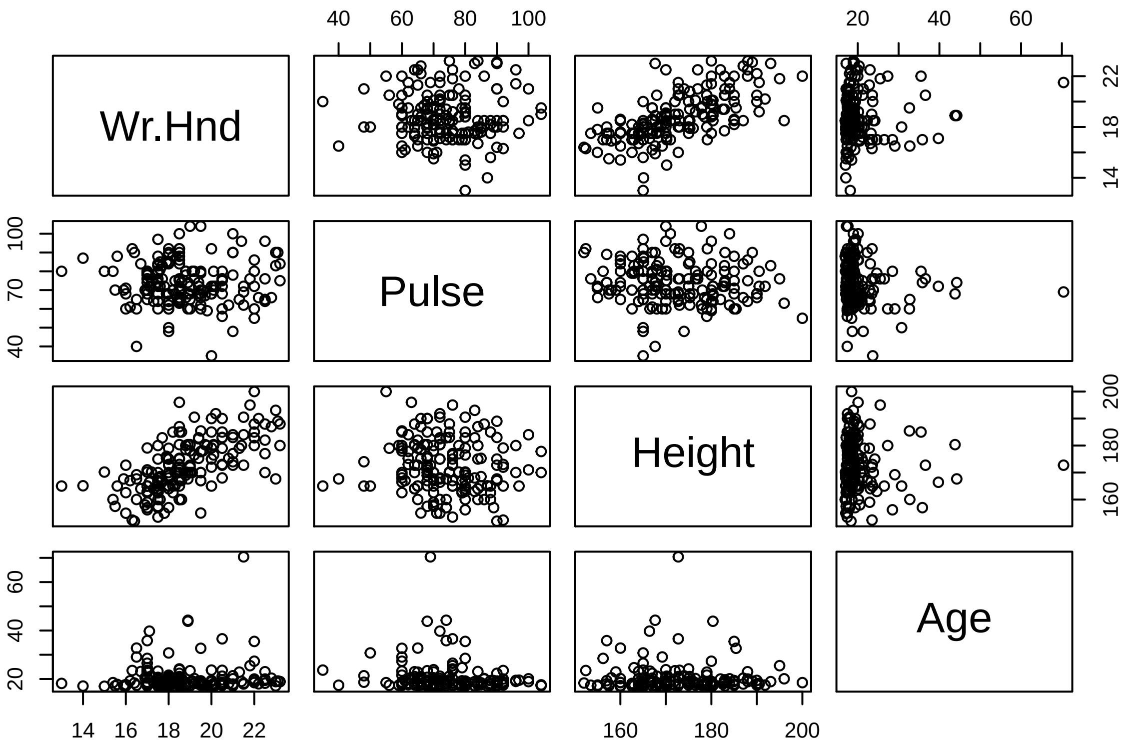

```r
# categorical variables
plot(csurvey$Smoke)
```



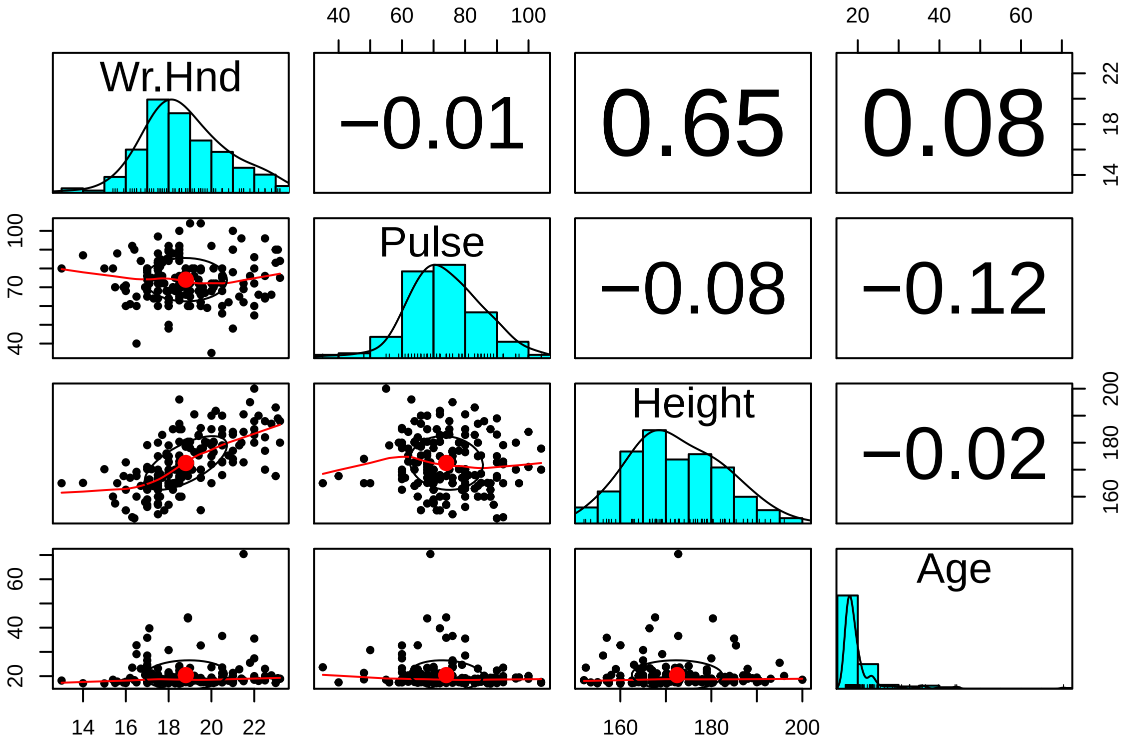

```r
plot(prop.table(table(csurvey$Smoke)))
```




```r
# plot covariation of multiple numeric variables
pairs( csurvey[, c("Wr.Hnd", "Pulse", "Height", "Age") ])
```

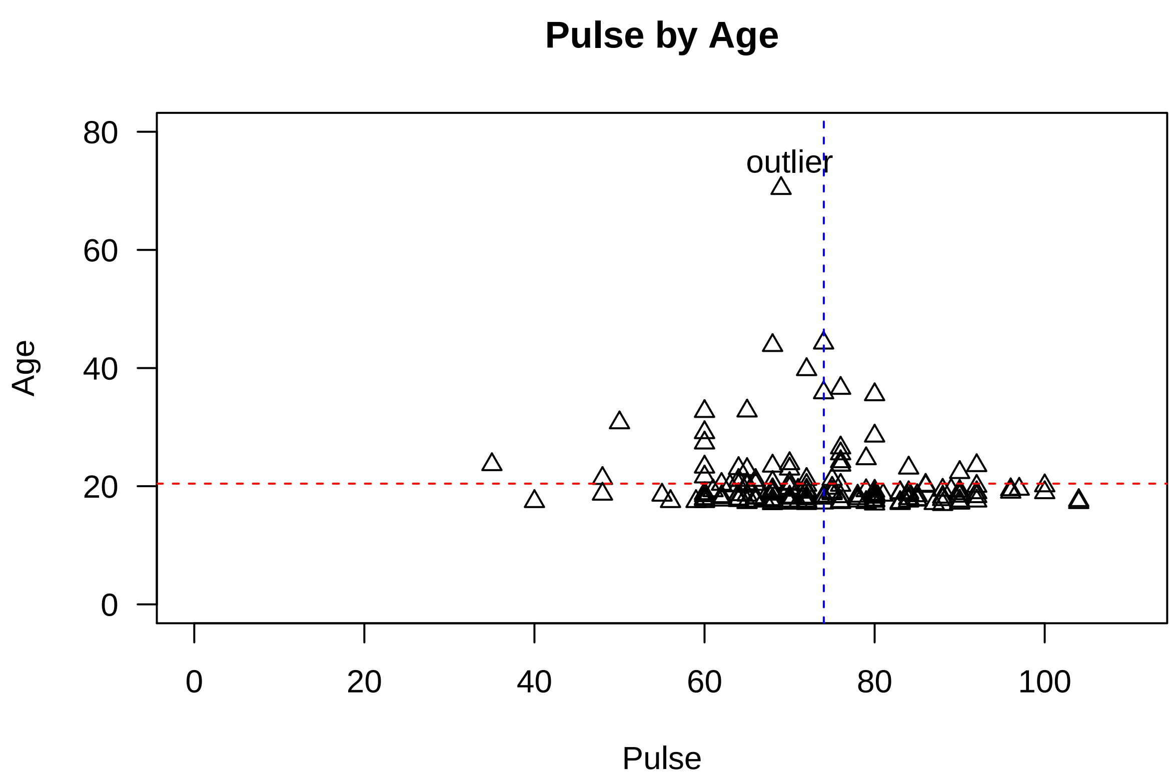



```r
psych::pairs.panels( csurvey[, c("Wr.Hnd", "Pulse", "Height", "Age") ])
```

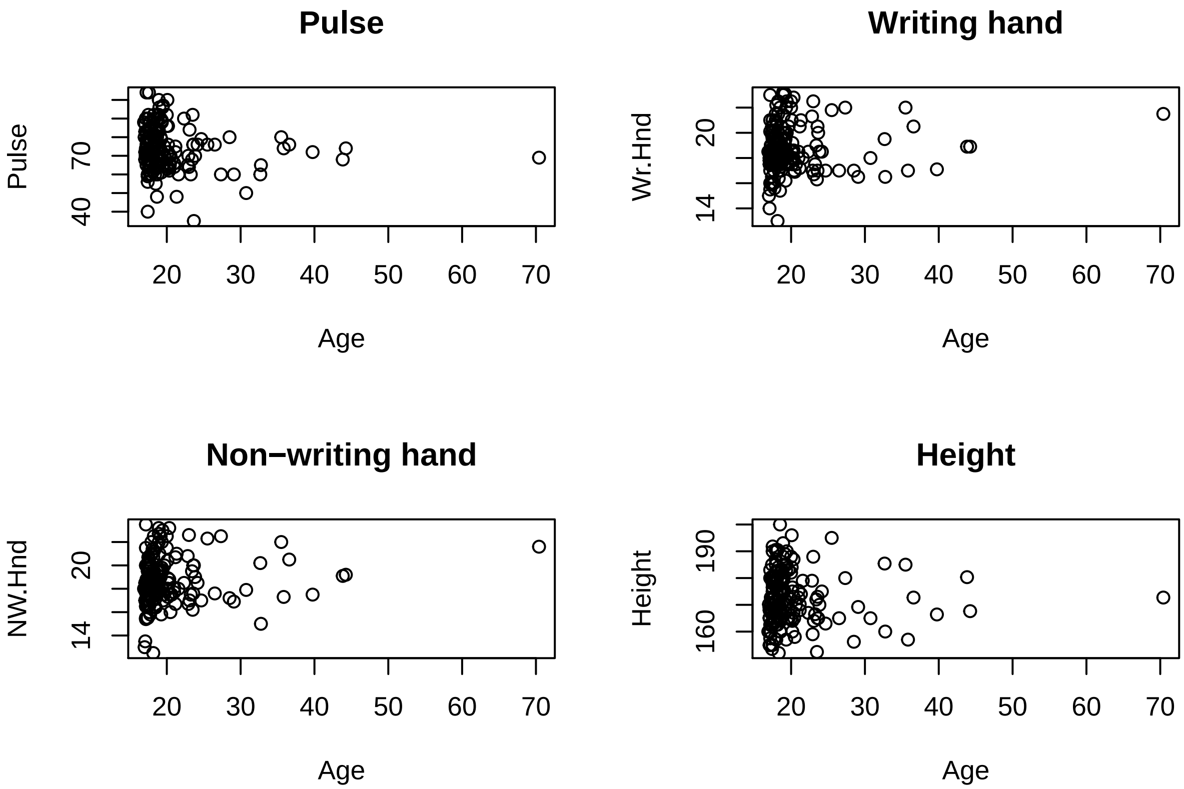


Graphics options

```
# Base graphics uses a "painting the page metaphor"

# add options to the main plotting functions
plot(csurvey$Pulse, csurvey$Age,
     xlab = "Pulse",  # x-axis label
     ylab = "Age",      # y-axis label
     pch = 2,          # plotting character
     las = 1,         # orientation of axis labels
     xlim = c(0, 110), # x axis limits
     ylim = c(0, 80)    # y axis limits
)

# overlay different elements
title("Pulse by Age") # add title to top of plot
abline(h = mean(csurvey$Age), lty = 2, col = "red") # add straight line
abline(v= mean(csurvey$Pulse), lty = 2, col = "blue")  # add straight line
text(70, 75, "outlier") # add text
```




```
# There are various graphic parameters like
# lty for line type
# col for colour
# For further information, see
# http://www.statmethods.net/advgraphs/parameters.html
?par # built-in help for graphics parameters
?plot.default # built-in help


# Arrange plots in grids
par(mfrow = c(2, 2)) # create grid of plots with 2 rows and 2 columns
plot(Pulse ~ Age, csurvey, main = "Pulse")
plot(Wr.Hnd ~ Age, csurvey,  main = "Writing hand")
plot(NW.Hnd ~ Age, csurvey, main = "Non-writing hand")
plot(Height ~ Age, csurvey, main = "Height")
```




```r
par(mfrow = c(1, 1)) # return to standard settings
```

Saving plot

```r
# Option 1. Click on export in RStudio

# Option 2. Use a graphics device
?Devices # see list of graphics devices

# Step 1. turn on graphics device
# In this case I am using pdf
pdf(file = "output/height-histogram.pdf")
# Step 2. Run plotting code
hist(csurvey$Height)

# Step 3. Turn of graphics device
dev.off()
```

```
## pdf
##   2
```

Exercise 1

```r
# For this exercise will use the GSS7402 dataset
library(AER)
```

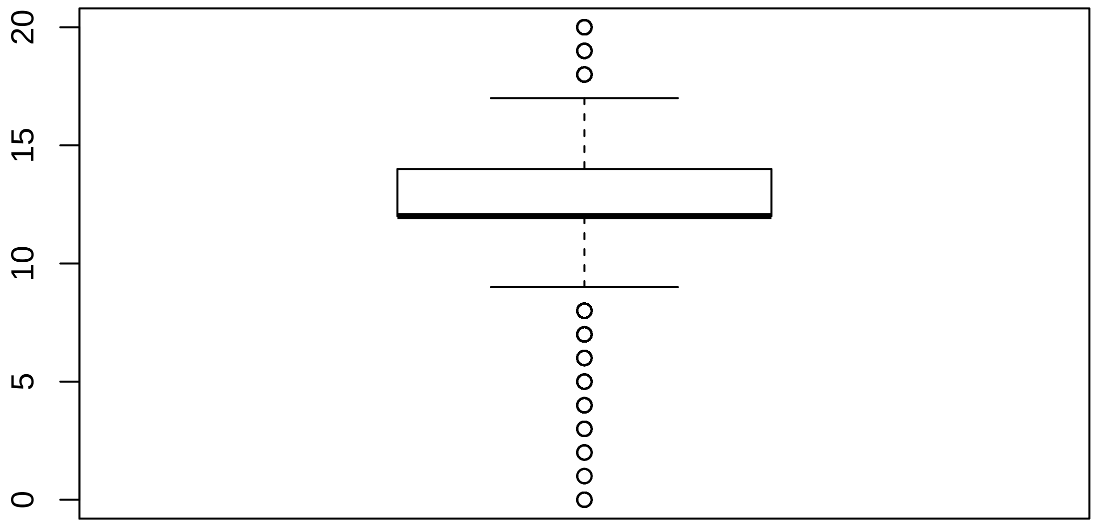
```
## Loading required package: car
##
## Attaching package: 'car'
##
## The following object is masked from 'package:psych':
##
##     logit
##
## Loading required package: lmtest
## Loading required package: zoo
##
## Attaching package: 'zoo'
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
##
## Loading required package: sandwich
```

```r
data("GSS7402")
?GSS7402 # to learn about the dataset
# It might be easier to work with a shorter variable name
gss <- GSS7402

# 1. Use base graphics to create a boxplot for education

# 2.  Create a boxplot for education split by year

# 3. Add some elements to the plot
#    (a) x and y labels,
#    (b) a red horizontal line at 12 years of education

# 4. Save the previous plot as a pdf in the output directory.
#    Paste the document into a word processor (e.g., MS Word)
```

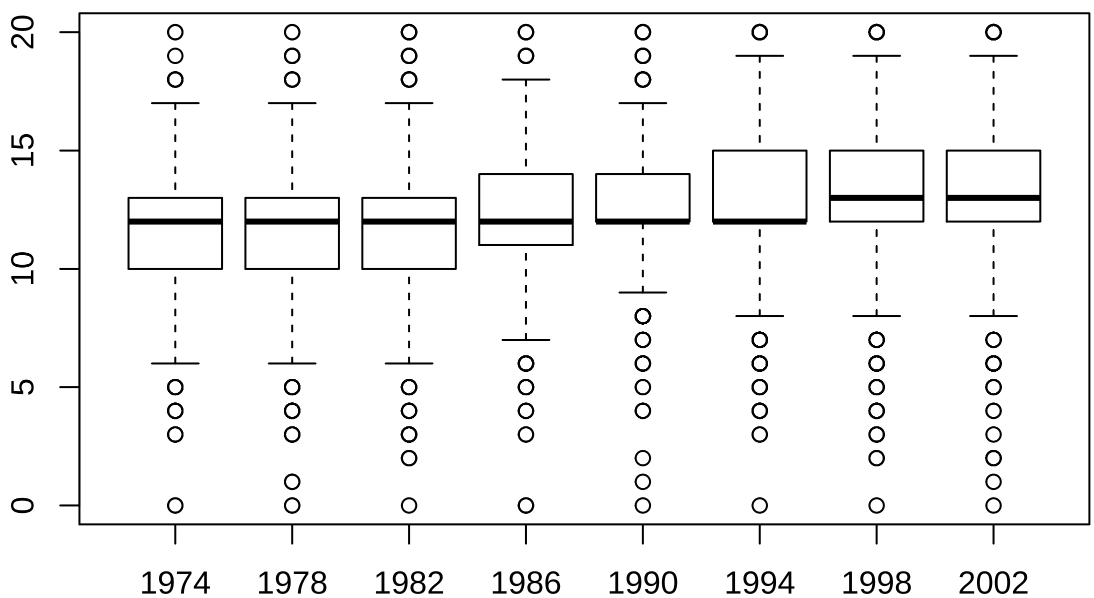
Answer 1

```r
# For this exercise will use the GSS7402 dataset
library(AER)
data("GSS7402")
?GSS7402 # to learn about the dataset
# It might be easier to work with a shorter variable name
gss <- GSS7402
```

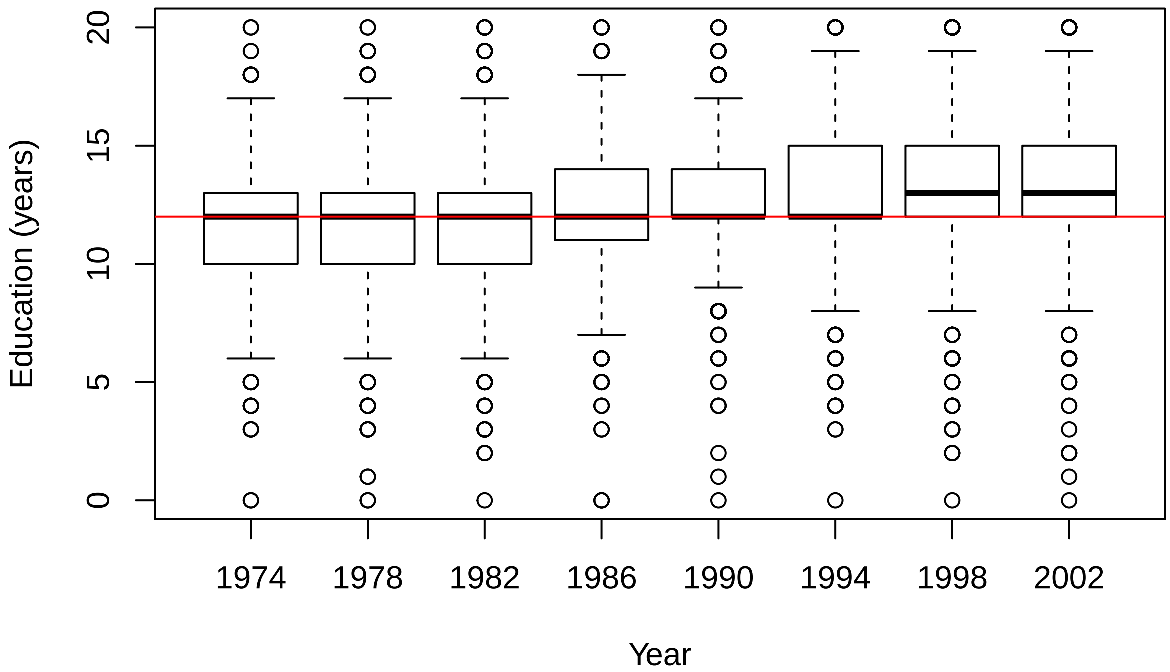
```r
# 1. Use base graphics to create a boxplot for education
boxplot(gss$education)
```




```r
# 2.  Create a boxplot for education split by year
boxplot(gss$education ~ gss$year)
```




```r
# 3. Add some elements to the plot
#    (a) x and y labels,
#    (b) a red horizontal line at 12 years of education
boxplot(gss$education ~ gss$year, xlab = "Year", ylab = "Education (years)")
abline(h = 12, col="red")
```




```r
# 4. Save the previous plot as a pdf in the output directory.
#    Paste the document into a word processor (e.g., MS Word)
pdf("output/graphics-boxplot.pdf")
boxplot(gss$education ~ gss$year, xlab = "Year", ylab = "Education (years)")
abline(h = 12, col="red")
dev.off()
```

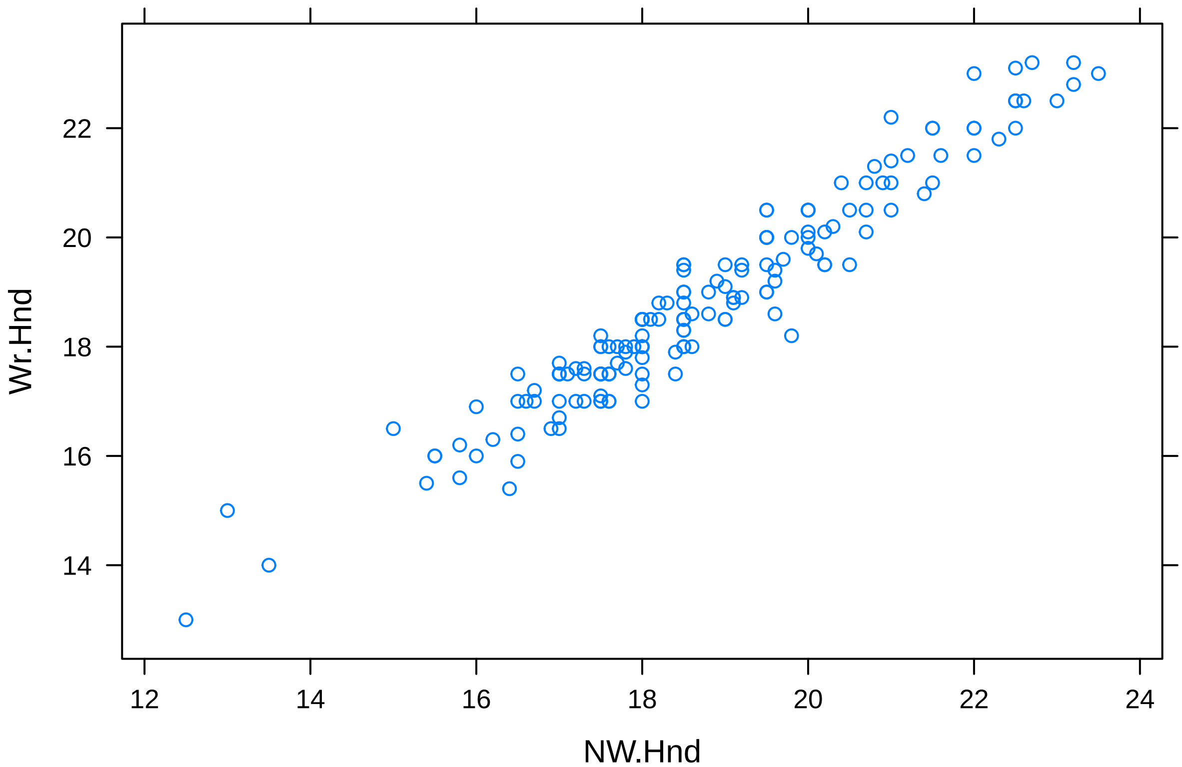
```
## pdf
##   2
```

Lattice Plots

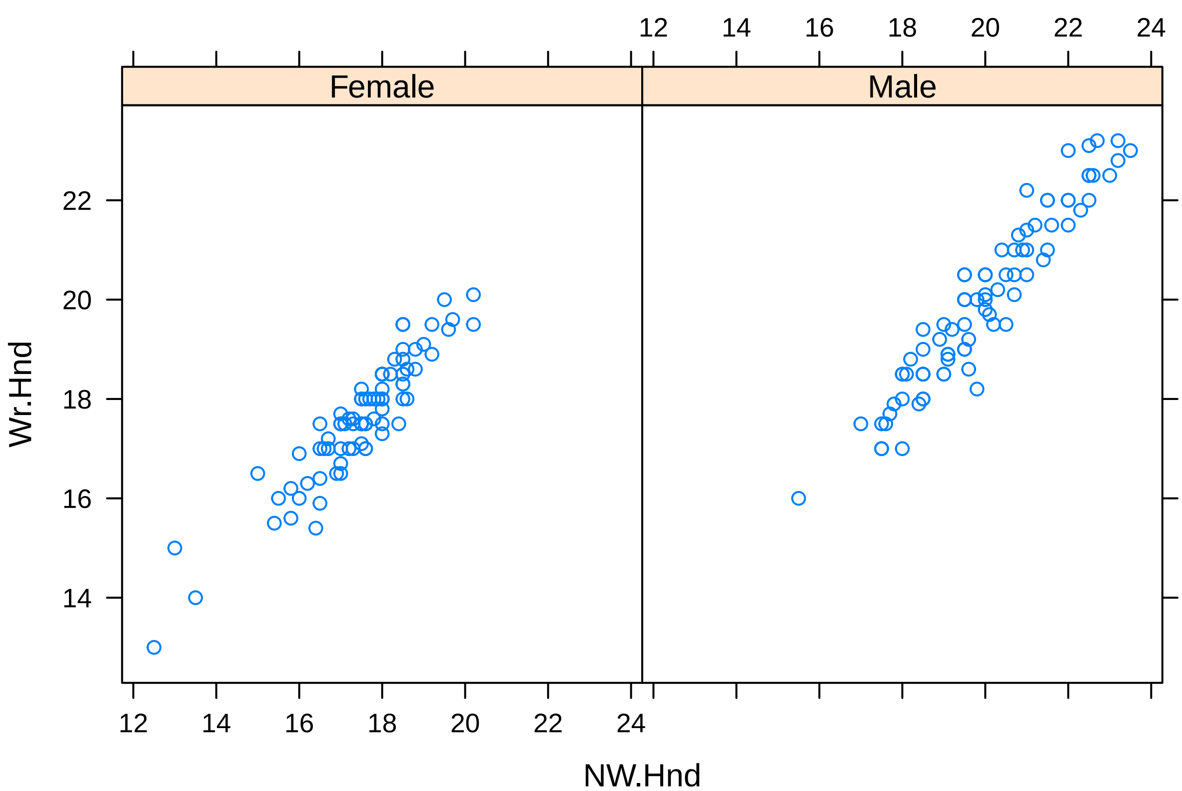
```r
library(lattice)
head(csurvey)
```

```
##      Sex Wr.Hnd NW.Hnd W.Hnd    Fold Pulse  Clap Exer Smoke Height
## 1 Female   18.5   18.0 Right  R on L    92  Left Some Never 173.00
## 2   Male   19.5   20.5  Left  R on L   104  Left None Regul 177.80
## 5   Male   20.0   20.0 Right Neither    35 Right Some Never 165.00
## 6 Female   18.0   17.7 Right  L on R    64 Right Some Never 172.72
## 7   Male   17.7   17.7 Right  L on R    83 Right Freq Never 182.88
## 8 Female   17.0   17.3 Right  R on L    74 Right Freq Never 157.00
##        M.I    Age
## 1   Metric 18.250
## 2 Imperial 17.583
## 5   Metric 23.667
## 6 Imperial 21.000
## 7 Imperial 18.833
## 8   Metric 35.833
```

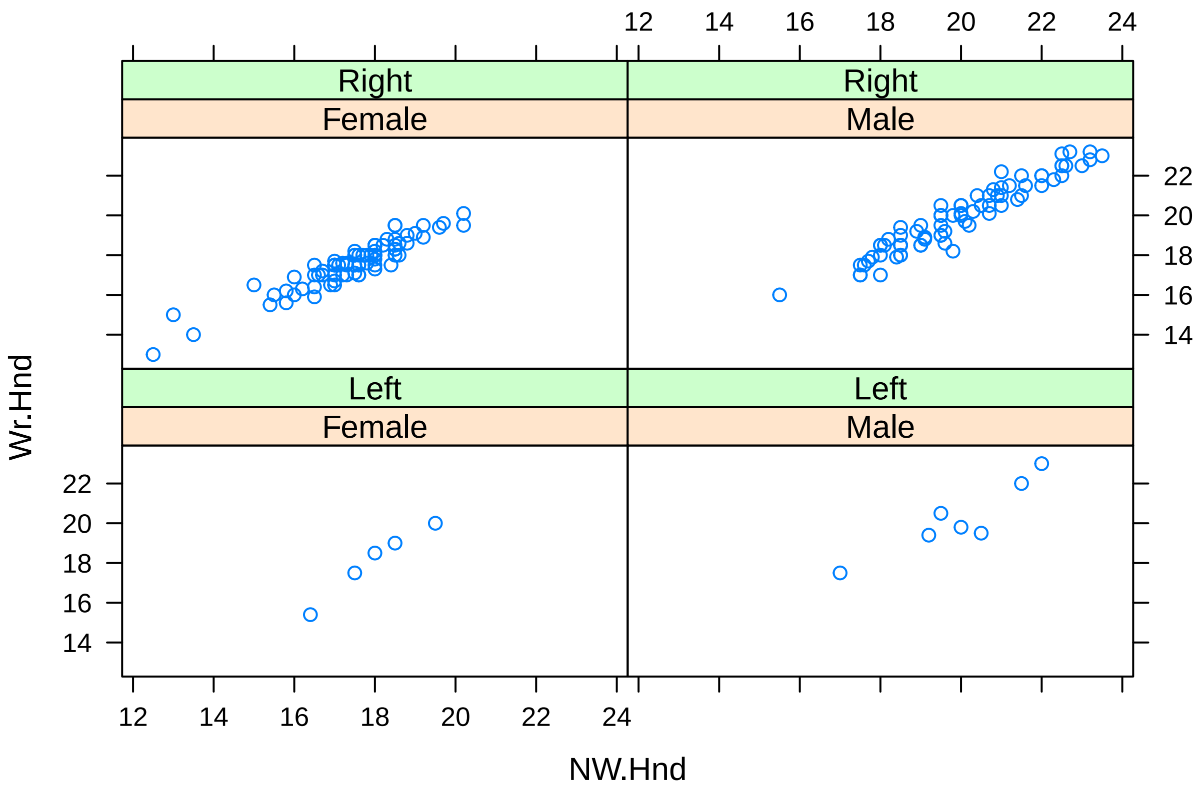
```r
xyplot(Wr.Hnd ~ NW.Hnd, csurvey)
```



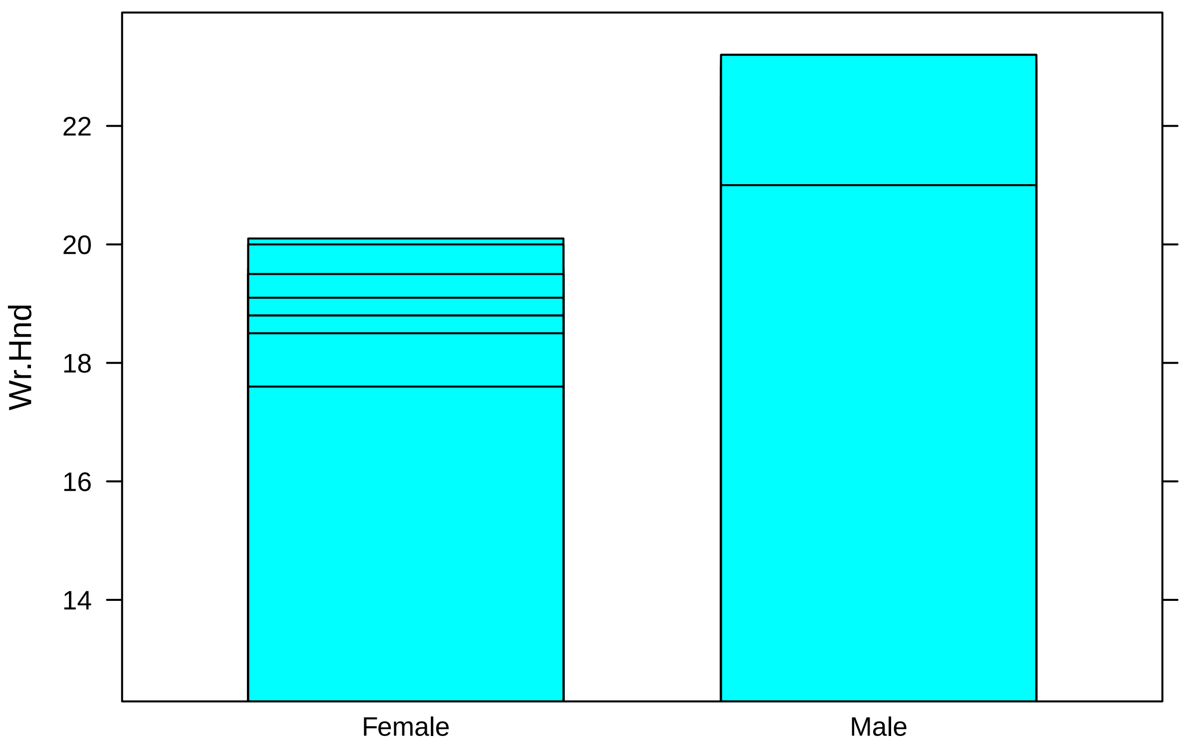

```r
xyplot(Wr.Hnd ~ NW.Hnd | Sex, csurvey)
```



```r
xyplot(Wr.Hnd ~ NW.Hnd | Sex + W.Hnd, csurvey)
```




```r
barchart(Wr.Hnd ~  Sex, csurvey)
```



```r
# saving lattice plots
# same as base but you need to print the plot
pdf(file = "output/lattice-plot.pdf")
# Step 2. Run plotting code with print
print(xyplot(Wr.Hnd ~ NW.Hnd, csurvey))

# Step 3. Turn of graphics device
dev.off()
```

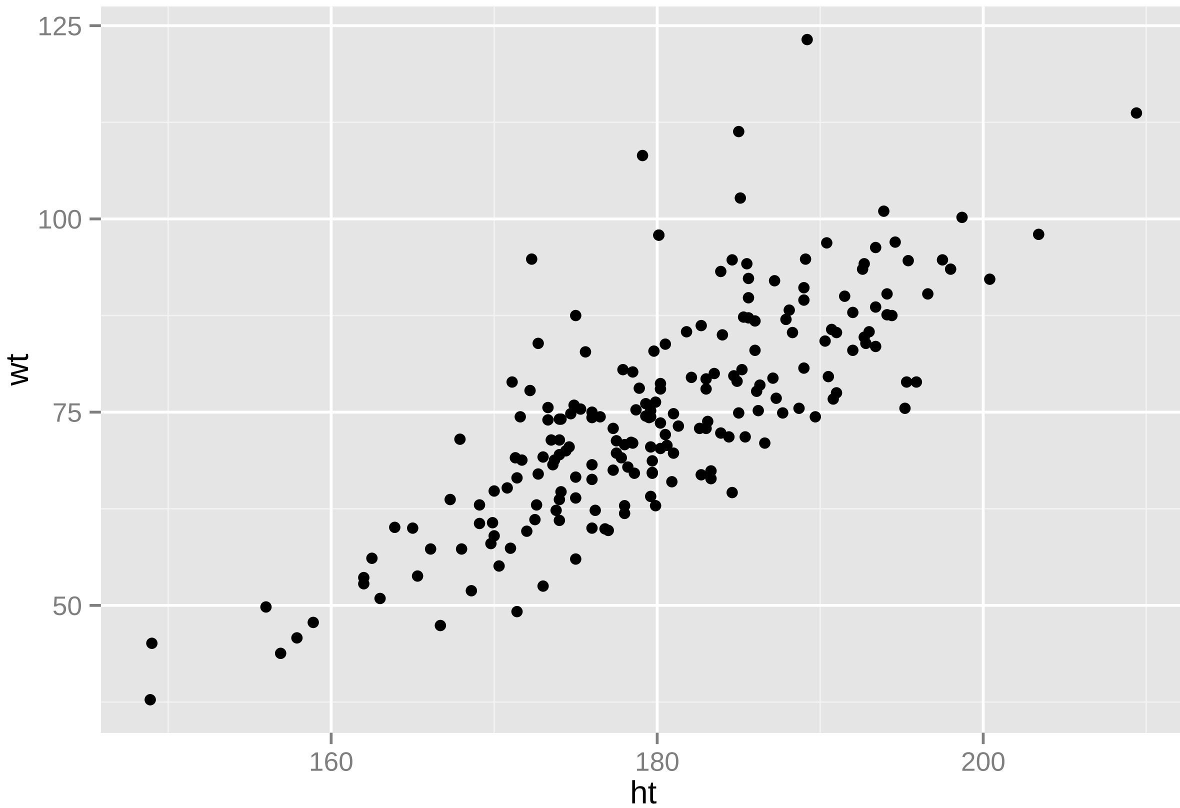
```
## pdf
##   2
```

ggplot2

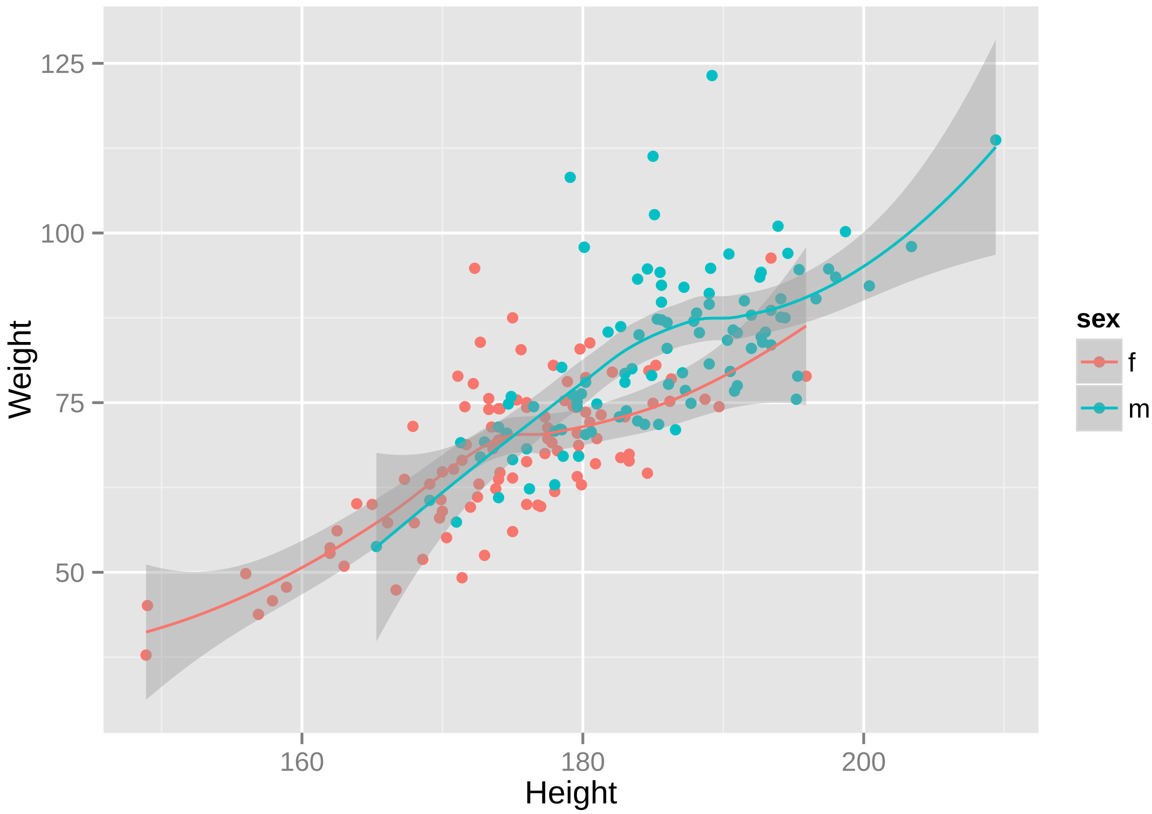
```r
library(ggplot2)

# Let's look at the ais dataset
library(DAAG)
```

```
##
## Attaching package: 'DAAG'
##
## The following object is masked from 'package:car':
##
##     vif
##
## The following object is masked from 'package:MASS':
##
##     hills
##
## The following object is masked from 'package:survival':
##
##     lung
##
## The following object is masked from 'package:psych':
##
##     cities
```

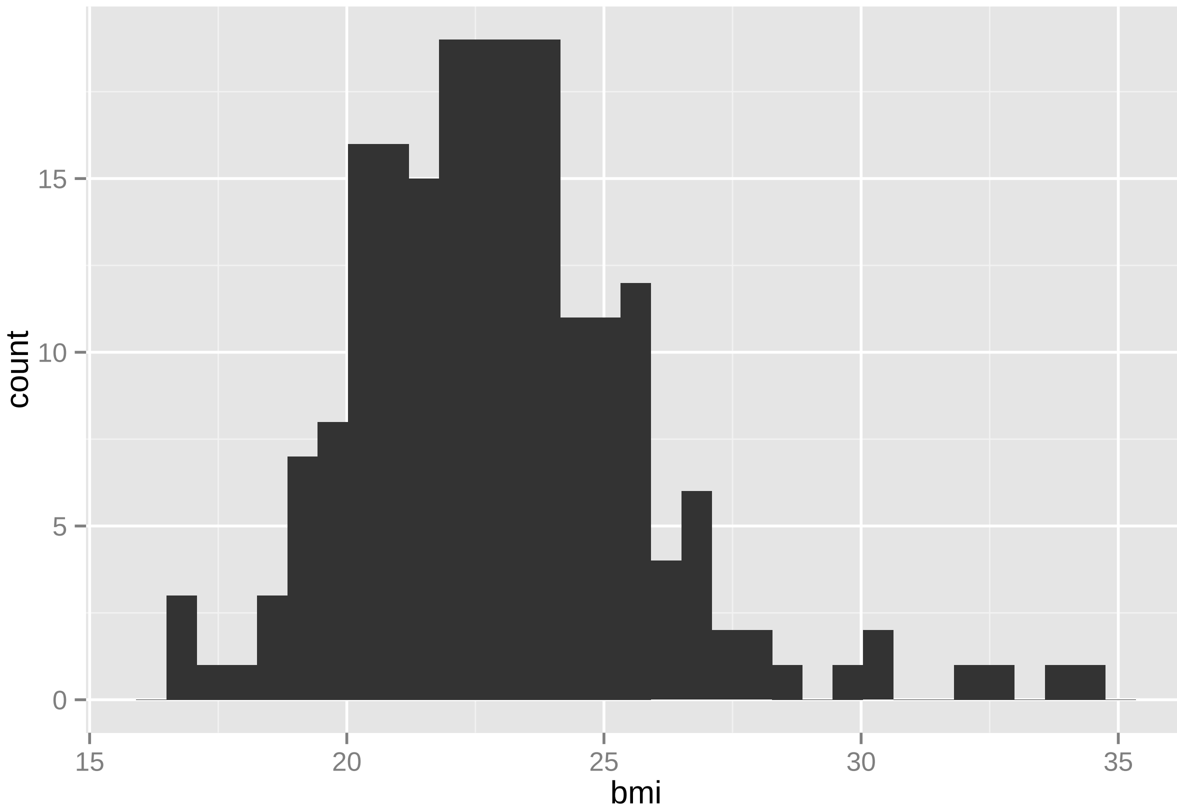
```r
data(ais)
?ais

# See the Rstudio ggplot2 cheatsheet
# and the ggplot2 documentation: http://docs.ggplot2.org/current/

# specify the data frame and the mapping of variables to plot attributes

# scatter plot
# 1. supply a data.frame
# 2. add aesthetic mapping between variables in data.frame
#    and
ggplot(ais, aes(x = ht, y = wt)) + geom_point()
```



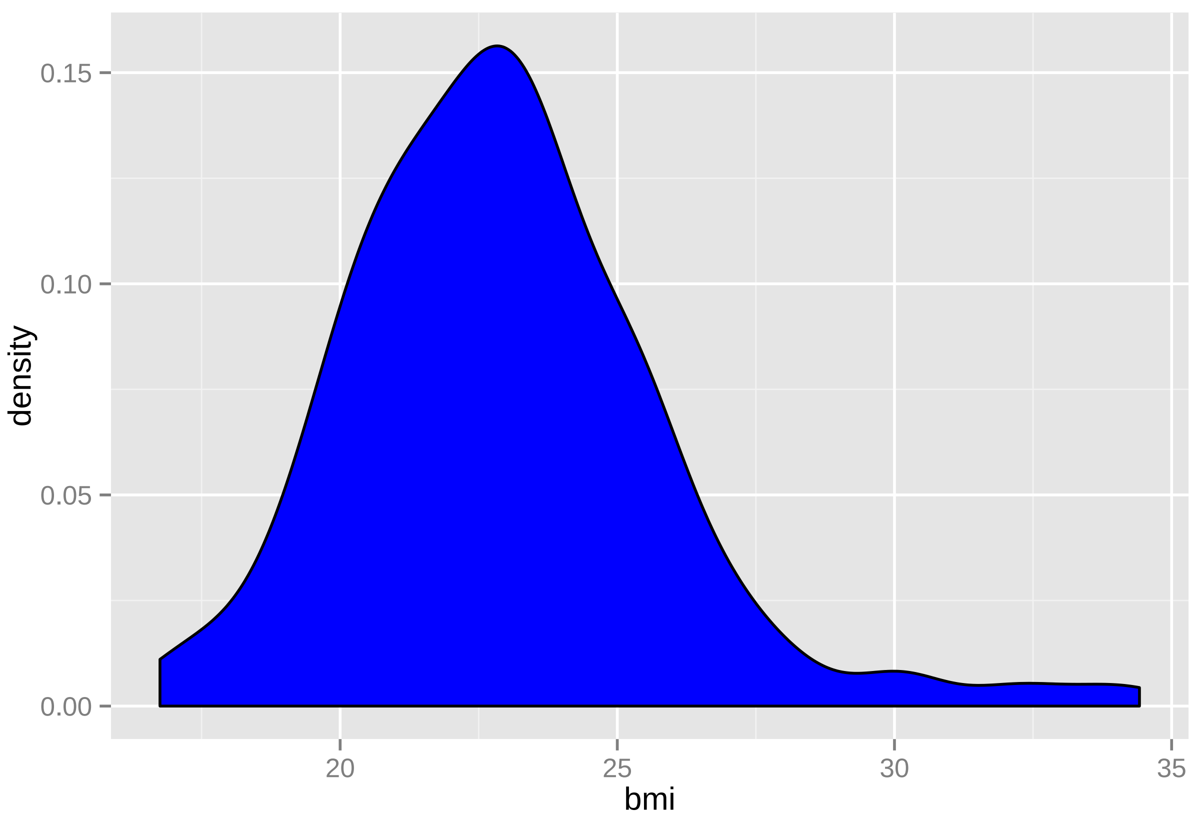

```r
ggplot(ais, aes(x = ht, y= wt, colour = sex)) +
    geom_point() +
    geom_smooth() +
    xlab("Height") +
    ylab("Weight")
```



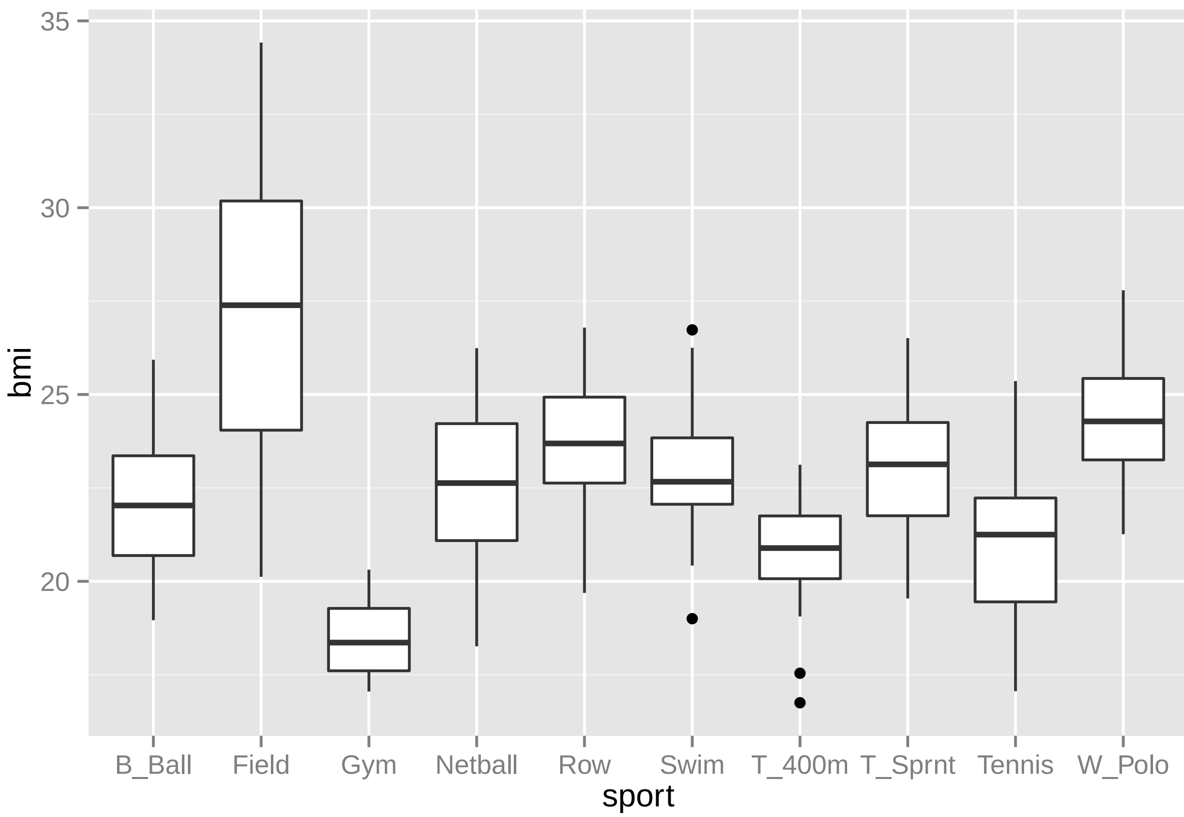

```
ggsave("output/height_weight.pdf", width = 5, height = 5) # save last plot

# distribution
p <- ggplot(ais, aes(x = bmi))
p + geom_histogram() #histogram
```



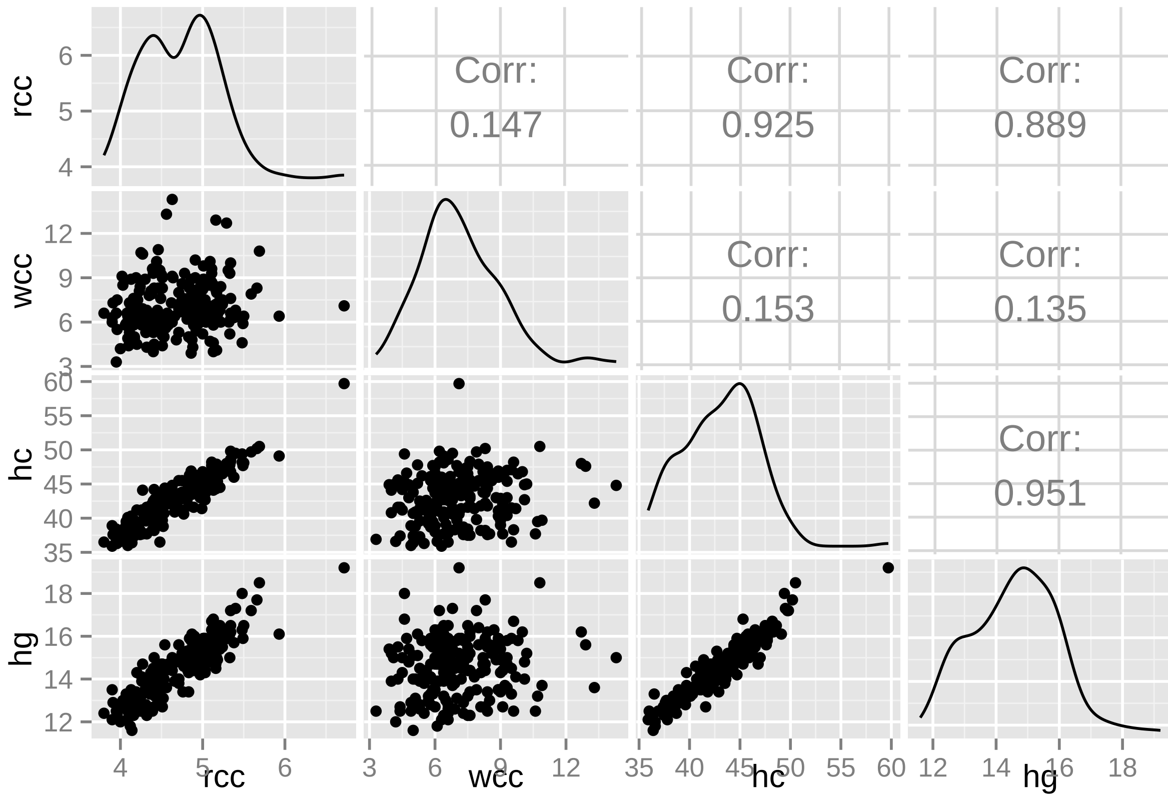

```
p + geom_density(fill = "blue") # density plot
```



```r
# Show group differences
p <- ggplot(ais, aes(x = sport, y = bmi))
p + geom_boxplot()
```




```r
# Scatterplot matrix
# Something like pairs and pairs.panels
library(GGally)
GGally::ggpairs(ais[1:4])
```



Data Manipulation

```r
# We'll work with the GSS dataset and the bfi dataset
library(AER)
data("GSOEP9402")
gss <- GSOEP9402

library(psych)
data(bfi)
cbfi <- na.omit(bfi)
dput(names(cbfi))
```

```
## c("A1", "A2", "A3", "A4", "A5", "C1", "C2", "C3", "C4", "C5",
## "E1", "E2", "E3", "E4", "E5", "N1", "N2", "N3", "N4", "N5", "O1",
## "O2", "O3", "O4", "O5", "gender", "education", "age")
```

```r
v <- list()
v$items <- c("A1", "A2", "A3", "A4", "A5", "C1", "C2", "C3", "C4", "C5",
    "E1", "E2", "E3", "E4", "E5", "N1", "N2", "N3", "N4", "N5", "O1",
    "O2", "O3", "O4", "O5")

# Aggegate statistic over grouping variable
aggregate(A1 ~ gender, cbfi, function(X) mean(X))
```

```
##   gender       A1
```

```
## 1      1 2.697959
## 2      2 2.202532
```

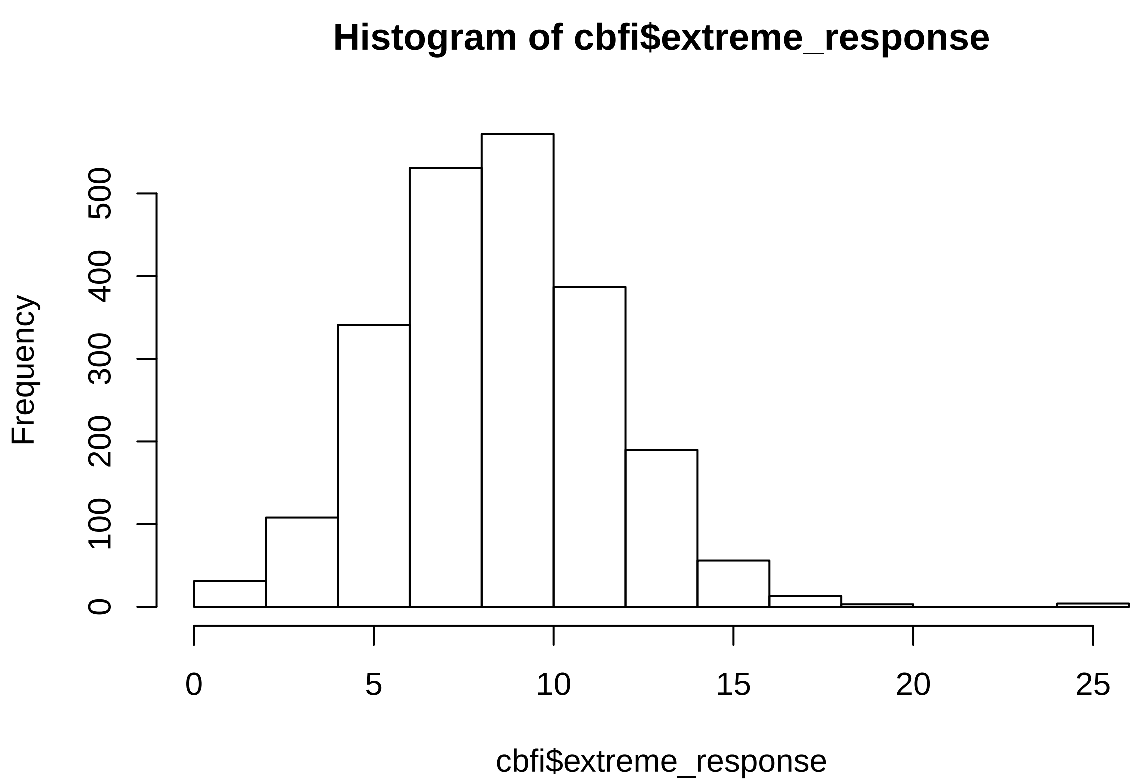
```r
aggregate(A1 ~ gender, cbfi, mean)
```

```
##   gender       A1
## 1      1 2.697959
## 2      2 2.202532
```

```r
aggregate(cbfi$A1, list(gender=cbfi$gender), function(X) mean(X))
```

```
##   gender        x
## 1      1 2.697959
## 2      2 2.202532
```

```r
# calculate statistic on each row of data
cbfi$average_response <- apply(cbfi[ v$items], 1, mean)
cbfi$extreme_response <- apply(cbfi[ v$items], 1, function(X) sum(X %in% c(1,5)))
hist(cbfi$extreme_response)
```




```r
# psych::scoreItems to score personality tests with a given key
# or see the final personality example in session 4

# calculate statistic for each element in a vector, list or column of
# a data.frame

# lapply returns a list
lapply(cbfi[, v$items], function(X) mean(X))
```

```
## $A1
## [1] 2.365385
##
## $A2
## [1] 4.834079
##
## $A3
## [1] 4.629249
##
## $A4
## [1] 4.749553
##
## $A5
## [1] 4.584973
##
## $C1
## [1] 4.569767
##
## $C2
## [1] 4.401163
##
## $C3
## [1] 4.322898
##
## $C4
## [1] 2.500894
##
## $C5
## [1] 3.255367
##
## $E1
## [1] 2.969589
##
## $E2
## [1] 3.121199
##
## $E3
## [1] 4.009839
##
## $E4
## [1] 4.43068
##
## $E5
## [1] 4.418605
##
## $N1
## [1] 2.908318
##
## $N2
## [1] 3.485689
##
## $N3
## [1] 3.198569
##
```

```
## $N4
## [1] 3.175313
##
## $N5
## [1] 2.952147
##
## $O1
## [1] 4.821556
##
## $O2
## [1] 2.689177
##
## $O3
## [1] 4.483005
##
## $O4
## [1] 4.948122
##
## $O5
## [1] 2.455277
```

```
# sapply attempt to simplifies the result (e.g., to a vector)
sapply(cbfi[, v$items], function(X) mean(X))
```

```
##       A1       A2       A3       A4       A5       C1       C2       C3
## 2.365385 4.834079 4.629249 4.749553 4.584973 4.569767 4.401163 4.322898
##       C4       C5       E1       E2       E3       E4       E5       N1
## 2.500894 3.255367 2.969589 3.121199 4.009839 4.430680 4.418605 2.908318
##       N2       N3       N4       N5       O1       O2       O3       O4
## 3.485689 3.198569 3.175313 2.952147 4.821556 2.689177 4.483005 4.948122
##       O5
## 2.455277
```

```
# Most operations are vectorised anyway
x <- 1:10
x * 2
```

```
##  [1]  2  4  6  8 10 12 14 16 18 20
```

```
# But this can be useful when they are not
sapply(x, function(X) X * 2)
```

```
##  [1]  2  4  6  8 10 12 14 16 18 20
```

```
# Works on lists
fits <- list()
fits$fit1 <- lm(income ~ gender, gss)
fits$fit2 <- lm(income ~ gender + size, gss)
fits$fit3 <- lm(income ~ gender + size + memployment, gss)
sfits <- lapply(fits, summary)
# sfits
```

```r
# Example, use it to extract same property from
# set of statistical models.
# See how to extract one element with code completion
sfits$fit1$adj.r.squared
```

```
## [1] -0.00147071
```

```r
# then apply elementwise
sapply(sfits, function(X) X$adj.r.squared)
```

```
##        fit1        fit2        fit3
## -0.00147071  0.07448579  0.11648601
```

```r
# re-order a data.fraame
# decreasing
x <- cbfi[ order(cbfi$extreme_response, decreasing = TRUE), ]
head(x)
```

```
##       A1 A2 A3 A4 A5 C1 C2 C3 C4 C5 E1 E2 E3 E4 E5 N1 N2 N3 N4 N5 O1 O2 O3
## 62783  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
## 64642  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
## 64953  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
## 65974  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
## 63838  1  5  5  5  4  5  5  5  1  1  1  1  5  6  5  1  1  1  1  1  6  1  5
## 65888  5  1  6  5  5  5  1  5  1  4  4  1  5  6  5  1  1  1  1  1  5  1  6
##       O4 O5 gender education age average_response extreme_response
## 62783  5  5      1         3  25             5.00               25
## 64642  1  1      1         3  23             1.00               25
## 64953  5  5      1         3  20             5.00               25
## 65974  1  1      1         3  19             1.00               25
## 63838  4  4      2         4  43             3.20               20
## 65888  1  1      2         3  40             3.12               20
```

```r
# or increasing
x <- cbfi[ order(cbfi$extreme_response), ]
head(x)
```

```
##       A1 A2 A3 A4 A5 C1 C2 C3 C4 C5 E1 E2 E3 E4 E5 N1 N2 N3 N4 N5 O1 O2 O3
## 62252  3  4  4  3  4  4  4  4  4  3  3  4  3  4  3  4  4  3  3  3  4  3  4
## 63271  3  3  4  2  3  3  3  3  3  4  3  4  4  4  2  3  3  2  3  3  3  4  3
## 62612  2  4  4  3  3  4  2  4  4  2  5  3  3  4  4  2  3  3  3  3  4  2  3
## 63909  3  3  4  4  3  3  3  4  2  4  2  2  3  3  2  2  2  2  4  2  3  3  2
## 64311  3  3  3  4  4  3  4  4  3  4  4  2  4  3  3  1  2  2  4  3  4  4  4
## 64950  3  3  3  4  4  4  3  3  4  5  2  3  4  4  4  3  4  2  4  2  4  3  6
##       O4 O5 gender education age average_response extreme_response
## 62252  4  3      2         2  19             3.56                0
## 63271  4  4      1         1  19             3.20                0
## 62612  3  3      2         2  30             3.20                1
## 63909  5  2      1         1  35             2.88                1
## 64311  4  3      1         3  18             3.28                1
## 64950  6  2      2         4  24             3.56                1
```

```
# Extract subsets of data based on condition
# Use logical vector in the rows
cbfi_cleaned <- cbfi[ cbfi$extreme_response < 25, ]

# Extract subset of variables
# subset of column names
cbfi_items <- cbfi[, v$items]

# or subset provides another option
x <- subset(cbfi, subset = extreme_response < 25, select = v$items)
head(x); nrow(x); nrow(cbfi)
```

```
##       A1 A2 A3 A4 A5 C1 C2 C3 C4 C5 E1 E2 E3 E4 E5 N1 N2 N3 N4 N5 O1 O2 O3
## 61623  6  6  5  6  5  6  6  6  1  3  2  1  6  5  6  3  5  2  2  3  4  3  5
## 61629  4  3  1  5  1  3  2  4  2  4  3  6  4  2  1  6  3  2  6  4  3  2  4
## 61634  4  4  5  6  5  4  3  5  3  2  1  3  2  5  4  3  3  4  2  3  5  3  5
## 61640  4  5  2  2  1  5  5  5  2  2  3  4  3  6  5  2  4  2  2  3  5  2  5
## 61661  1  5  6  5  6  4  3  2  4  5  2  1  2  5  2  2  2  2  2  2  6  1  5
## 61664  2  6  5  6  5  3  5  6  3  6  2  2  4  6  6  4  4  4  6  6  6  1  5
##       O4 O5
## 61623  6  1
## 61629  5  3
## 61634  6  3
## 61640  5  5
## 61661  5  2
## 61664  6  1
```

```
## [1] 2232
```

```
## [1] 2236
```

```
mat <- matrix(c(1,2,
                3,4), nrow= 2)
mat
```

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

```
# Add columns
mat <- cbind(mat, c(8,8))
mat
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    8
## [2,]    2    4    8
```

```
# add rows
mat <- rbind(mat, c(9,9,9))
mat
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    8
## [2,]    2    4    8
## [3,]    9    9    9
```

```
######################################
# Merge
# Merge on common variable
# Let's create an aggregate variable
# to merge into the lower level data
meankids <- aggregate(kids ~ birthyear,  gss, mean)
names(meankids) <- c("birthyear", "mean_kids")

temp <- merge(gss, meankids)
dim(temp)
```

```
## [1] 675  13
```

```
dim(gss)
```

```
## [1] 675  12
```

```
# it's good to check that the merge worked before
# overriding the original data.frame
gss <- merge(gss, meankids)
head(gss)
```

```
##   birthyear     school gender kids parity   income size
## 1      1980 Realschule female    4      1 63276.86    6
## 2      1980 Realschule female    2      1 58493.02    4
## 3      1980 Realschule   male    3      1 36848.08    3
## 4      1980  Gymnasium female    2      1 64421.03    4
## 5      1980  Gymnasium female    3      1 62880.01    5
## 6      1980 Realschule female    2      1 37095.14    4
##                      state marital meducation memployment year mean_kids
## 1      Nordrhein-Westfalen married       11.5        none 1994  2.478261
## 2            Niedersachsen married       10.5    parttime 1994  2.478261
## 3                  Hamburg  single       15.0    parttime 1994  2.478261
## 4 Rheinland-Pfalz/Saarland married       16.0        none 1994  2.478261
## 5       Schleswig-Holstein married       12.0        none 1994  2.478261
## 6 Rheinland-Pfalz/Saarland married       15.0        none 1994  2.478261
```

```
######################################
# Reshape
# http://www.ats.ucla.edu/stat/r/faq/reshape.htm
# With longitudinal data we sometimes want to
# reshape from wide to long and long to wide
longfile <- aggregate(income ~ birthyear + kids,  gss, mean)
head(longfile)
```

```
##   birthyear kids   income
```

```
## 1      1980    1 43337.97
## 2      1981    1 50626.72
## 3      1982    1 63120.73
## 4      1983    1 49881.25
## 5      1984    1 65770.93
## 6      1985    1 71038.43
```

```
widefile <- reshape(longfile, timevar = "kids",
        idvar = "birthyear", direction = "wide")
widefile
```

```
##   birthyear income.1 income.2 income.3  income.4  income.5  income.6
## 1      1980 43337.97 60470.82 68198.55  50306.38  50663.89 118268.49
## 2      1981 50626.72 61537.64 69980.48  51096.84  50922.99  59044.23
## 3      1982 63120.73 74592.18 67491.38  63522.31  39541.18  71163.02
## 4      1983 49881.25 68409.12 84071.01 106646.11  76622.87  78408.50
## 5      1984 65770.93 71815.98 96493.08 103467.57  31916.83 101191.96
## 6      1985 71038.43 66980.94 78872.38  71212.07  76045.87  72121.65
## 7      1986 81141.91 66165.33 68162.82  78032.69 108962.09        NA
## 8      1987 70722.32 68583.31 90831.65  84758.25  72547.12  59925.36
## 9      1988 68247.47 82409.43 83574.29  93106.50 104344.44        NA
```

```
back2long <- reshape(widefile,
                     times =  c("income.1", "income.2", "income.3",
                         "income.4",  "income.5", "income.6"),
                         direction = "long")
head(back2long)
```

```
##        birthyear kids income.1
## 1980.1      1980    1 43337.97
## 1981.1      1981    1 50626.72
## 1982.1      1982    1 63120.73
## 1983.1      1983    1 49881.25
## 1984.1      1984    1 65770.93
## 1985.1      1985    1 71038.43
```

```
#################################################
# The Hadleyverse
# http://had.co.nz/

# Hadley Wickham is a celebrity in the R world
# and has developed many new packages that attempt
# to make R more user friendly.
# Most prominently these include the graphics package
# ggplot2
# as well as several for data manipulation including
# dplyr, tidyr
# You may wish to examine the RStudio Data Wrangling cheat sheet.
# The above data manipulation methods are built into base R.
# Hadley's packages do similar things but you may find them more
# elegant and consistent.
```

```r
# examples
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
##
## The following object is masked from 'package:GGally':
##
##     nasa
##
## The following object is masked from 'package:MASS':
##
##     select
##
## The following objects are masked from 'package:Hmisc':
##
##     combine, src, summarize
##
## The following object is masked from 'package:stats':
##
##     filter
##
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```r
# A bit like aggregate
dplyr::summarise(cbfi[, v$items],
                 mean_A1 = mean(A1), sd_A1 = sd(A1))
```

```
##    mean_A1    sd_A1
## 1 2.365385 1.391968
```

```r
# Similar to sapply but returns a data.frame
dplyr::summarise_each(cbfi[, v$items], funs(sd))
```

```
##         A1       A2       A3       A4       A5       C1      C2       C3
## 1 1.391968 1.156915 1.289373 1.447941 1.255833 1.216611 1.31131 1.287153
##         C4      C5       E1      E2       E3     E4       E5       N1
## 1 1.362817 1.62959 1.618121 1.60566 1.342438 1.4591 1.330117 1.564455
##         N2       N3     N4      N5       O1       O2       O3       O4
## 1 1.534764 1.596394 1.5606 1.62198 1.120043 1.545865 1.193261 1.175435
##         O5
## 1 1.329501
```

Exercise 2

```r
library(psych)
data(bfi)
cbfi <- na.omit(bfi)
dput(names(cbfi))
```

```
## c("A1", "A2", "A3", "A4", "A5", "C1", "C2", "C3", "C4", "C5",
## "E1", "E2", "E3", "E4", "E5", "N1", "N2", "N3", "N4", "N5", "O1",
## "O2", "O3", "O4", "O5", "gender", "education", "age")
```

```r
v <- list()
v$items <- c("A1", "A2", "A3", "A4", "A5", "C1", "C2", "C3", "C4", "C5",
    "E1", "E2", "E3", "E4", "E5", "N1", "N2", "N3", "N4", "N5", "O1",
    "O2", "O3", "O4", "O5")

# 1. Get the median of all items in cbfi

# 2. Get the number of times each participant gave
#    the response of 3 and assign this to a new variable

# 3. Produce frequency counts for each each

# 4. Create a new dataset excluding those over 50
#    and those under 18

# 5. Get the mean of each item by age from this younger sample
```

Answers 2

```r
library(psych)
data(bfi)
cbfi <- na.omit(bfi)
dput(names(cbfi))
```

```
## c("A1", "A2", "A3", "A4", "A5", "C1", "C2", "C3", "C4", "C5",
## "E1", "E2", "E3", "E4", "E5", "N1", "N2", "N3", "N4", "N5", "O1",
## "O2", "O3", "O4", "O5", "gender", "education", "age")
```

```r
v <- list()
v$items <- c("A1", "A2", "A3", "A4", "A5", "C1", "C2", "C3", "C4", "C5",
    "E1", "E2", "E3", "E4", "E5", "N1", "N2", "N3", "N4", "N5", "O1",
    "O2", "O3", "O4", "O5")

# 1. Get the median of all items in cbfi
sapply(cbfi[ ,v$items], median)
```

```
## A1 A2 A3 A4 A5 C1 C2 C3 C4 C5 E1 E2 E3 E4 E5 N1 N2 N3 N4 N5 O1 O2 O3 O4 O5
##  2  5  5  5  5  5  5  5  2  3  3  3  4  5  5  3  4  3  3  3  5  2  5  5  2
```

```r
# 2. Get the number of times each participant gave
#    the response of 3 and assign this to a new variable
cbfi$response3 <- apply(cbfi[ ,v$items],1,  function(X) sum(X == 3))

# 3. Produce frequency counts for each each
table(cbfi$age)
```

```
##
##   3  11  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29
##   1   1   1   3  14  31 107 164 178 129 102 128  92 100  81  83  74  72
##  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47
##  55  62  57  44  46  45  41  33  47  40  46  28  23  32  24  24  20  15
##  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65
##  26  10  31  16  19  15  14  11  13   6   6   2   6   3   3   3   1   1
##  66  67  68  74  86
##   1   3   1   1   1
```

```r
# 4. Create a new dataset excluding those over 50
#    and those under 18
cbfi_younger <- cbfi[ cbfi$age <= 50 & cbfi$age >= 18, ]

# 5. Get the mean of each item by age from this younger sample
x <- aggregate(cbfi_younger[,v$items], list(age = cbfi_younger$age), mean)
```