Introduction to R

Solutions to Exercises

Willi Mutschler

Contents

Introduction

1. Start ***R-Studio*** and have a look at all menu items.
2. Under `Tools - Options` choose your preferred setting.
3. Install the packages `dplyr`, `ggplot2`, `tidyr`, `stringr`, `readr`, `foreign`, `readxl`, `haven`, `sandwich`, `prettyR`, `Rcmdr`, `xtable`, `texreg`, and `lmtest`

```
#you will need to install packages only once
install.packages("dplyr")
install.packages("ggplot2")
install.packages("tidyr")
install.packages("stringr")
install.packages("readr")
install.packages("foreign")
install.packages("readxl")
install.packages("haven")
install.packages("sandwich")
install.packages("prettyR")
install.packages("Rcmdr")
install.packages("xtable")
install.packages("texreg")
install.packages("lmtest")
```

More simply: Use the `Packages` Tab of RStudio to install and activate the libraries.

4. Load the ggplot2 package, list the packages in your memory and detach the ggplot2 package from the memory

```
library("ggplot2")
search()
```

```
##  [1] ".GlobalEnv"        "package:ggplot2"   "package:stats"
##  [4] "package:graphics"  "package:grDevices" "package:utils"
##  [7] "package:datasets"  "package:methods"   "Autoloads"
## [10] "package:base"
```

```
detach("package:ggplot2")
search()
```

```
## [1] ".GlobalEnv"        "package:stats"     "package:graphics"
## [4] "package:grDevices" "package:utils"     "package:datasets"
## [7] "package:methods"   "Autoloads"         "package:base"
```

5. The current working directory (where R reads and writes files) can be found by the command *getwd()*. Find your current working directory.

```
getwd()
```

6. Use the command *setwd("c:/path")* or *setwd(choose.dir())* (only available on windows) to change the working directory to drive *c:* and path */path*. Note that the path name is structured by slashes (/), **not** backslashes (\). Change the working directory to *c:/temp* and check if the change has been successful.

```
setwd("C:/temp")
#setwd(choose.dir()) # on windows
getwd()
```

Hint: The working directory can also be changed via the menu: *Session – Set Working Directory*.

7. Open a new script file. Type the commands to perform the following assignments:

$$
\begin{aligned}
a &= \frac{3 \cdot (4+9)}{8-12.5} \\
b &= (1, 4, 1999, 2011) \\
d &= 2\pi \\
e &= a + d
\end{aligned}
$$

Save the script and quit R.

```
a <- 3 * (4 + 9) / (8 - 12.5)
b <- c(1, 4, 1999, 2011)
d <- 2 * pi
e <- a + d
```

8. Start R and re-open the script. Mark all lines (*Ctrl-A*) and execute them. Print a, b, d, e.

```
## [1] -8.666667
```

```
## [1]    1    4 1999 2011
```

```
## [1] 6.283185
```

```
## [1] -2.383481
```

9. Why is c not used as a variable?

```
# Because c() is the built-in concatenation command! Do not use c for variable or function declaration.
```

Logical Operators

1. Use the command `c()` to define the vectors

$$
\begin{aligned}
x &= (-1, 0, 1, 4, 9, 2, 1, 4.5, 1.1, -0.9) \\
y &= (1, 1, 2, 2, 3, 3, 4, 4, 5, NA)\,.
\end{aligned}
$$

```
x <- c(-1, 0, 1, 4, 9, 2, 1, 4.5, 1.1, -0.9)
y <- c(1, 1, 2, 2, 3, 3, 4, 4, 5, NA)
```

2. Determine the lengths of the vectors using `length()` and check if `length(x)==length(y)`.

```r
length(x); length(y); length(x) == length(y)
```

```
## [1] 10
## [1] 10
## [1] TRUE
```

3. Perform the following logical operations:

$$
\begin{aligned}
x &< y \\
x &< 0 \\
x + 3 &\geq 0 \\
y &< 0 \\
x &< 0 \text{ or } y < 0
\end{aligned}
$$

```r
x < y
```

```
##  [1]  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE FALSE  TRUE    NA
```

```r
x < 0
```

```
##  [1]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
```

```r
x + 3 >= 0
```

```
##  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

```r
y < 0
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE    NA
```

```r
x < 0 | y < 0
```

```
##  [1]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
```

4. Use `all` to check if all elements of $x + 3 \geq 0$.

```r
all(x + 3 >= 0)
```

```
## [1] TRUE
```

5. Use `all` to check if all elements of $y > 0$. Use `any` to check if at least one element of $y > 0$.

```r
all(y > 0)
```

```
## [1] NA
```

```r
all(y > 0, na.rm = TRUE)
```

```
## [1] TRUE
```

```r
any(y > 0)
```

```
## [1] TRUE
```

Arithmetic operators and mathematical functions

1. Define the vectors

$$
\begin{aligned}
x &= (-1, 0, 1, 4, 9, 2, 1, 4.5, 1.1, -0.9) \\
y &= (1, 1, 2, 2, 3, 3, 4, 4, 5, NA)\,.
\end{aligned}
$$

and compute $x + y$ and xy and y/x.

```
x <- c(-1, 0, 1, 4, 9, 2, 1, 4.5, 1.1, -0.9)
y <- c(1, 1, 2, 2, 3, 3, 4, 4, 5, NA)
```

```
x + y
```

```
##  [1]  0.0  1.0  3.0  6.0 12.0  5.0  5.0  8.5  6.1   NA
```

```
x * y
```

```
##  [1] -1.0  0.0  2.0  8.0 27.0  6.0  4.0 18.0  5.5   NA
```

```
y / x
```

```
##  [1] -1.0000000        Inf  2.0000000  0.5000000  0.3333333  1.5000000
##  [7]  4.0000000  0.8888889  4.5454545         NA
```

2. Compute $\ln(x)$. Determine the length of the result vector.

```
log(x)
```

```
## Warning in log(x): NaNs wurden erzeugt
```

```
##  [1]        NaN       -Inf 0.00000000 1.38629436 2.19722458 0.69314718
##  [7] 0.00000000 1.50407740 0.09531018        NaN
```

```
length(log(x))
```

```
## Warning in log(x): NaNs wurden erzeugt
```

```
## [1] 10
```

3. Use `any` to check if the vector x contains elements satisfying $\sqrt{x} \geq 2$.

```
any(sqrt(x) >= 2)
```

```
## Warning in sqrt(x): NaNs wurden erzeugt
```

```
## [1] TRUE
```

4. Compute

$$
\begin{aligned}
a &= \sum_{i=1}^{10} x_i \\
b &= \sum_{i=1}^{10} y_i^2.
\end{aligned}
$$

Use the `na.rm=TRUE` option (**na-rem**ove) of the `sum` command to drop the `NA` in y.

```
a <- sum(x)
print(a)
```

```
## [1] 20.7
```

```r
b <- sum(y^2, na.rm = TRUE)
print(b)
```

```
## [1] 85
```

5. Compute

$$\sum_{i=1}^{10} x_i y_i^2.$$

```r
sum(x * (y^2), na.rm = TRUE)
```

```
## [1] 233.5
```

6. The `sum` command is a convenient way to count the number of elements satisfying a certain condition. Count the number of elements of $x > 0$.

```r
sum(x > 0)
```

```
## [1] 7
```

```r
sum(!is.na(y))
```

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

7. Predict what the following commands will return:

```r
x^y
```

```
##  [1]  -1.00000   0.00000   1.00000  16.00000 729.00000   8.00000   1.00000
##  [8] 410.06250   1.61051        NA
```

```r
x^(1/y)
```

```
##  [1] -1.000000  0.000000  1.000000  2.000000  2.080084  1.259921  1.000000
##  [8]  1.456475  1.019245        NA
```

```r
log(exp(y))
```

```
##  [1]  1  1  2  2  3  3  4  4  5 NA
```

```r
y*c(-1,1)
```

```
##  [1] -1  1 -2  2 -3  3 -4  4 -5 NA
```

```r
x+c(-1,0,1)
```

```
## Warning in x + c(-1, 0, 1): Länge des längeren Objektes
##       ist kein Vielfaches der Länge des kürzeren Objektes
```

```
##  [1] -2.0  0.0  2.0  3.0  9.0  3.0  0.0  4.5  2.1 -1.9
```

```r
sum(y*c(-1,1),na.rm=TRUE)
```

```
## [1] -5
```

Matrix functions

1. Define the matrix

$$X = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix},$$

print its transpose, its dimensions and its determinant.

```
X <- matrix(c(1,2,3,4,5,6,7,8,9), nrow = 3, ncol = 3)
t(X)
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
```

```
dim(X)
```

```
## [1] 3 3
```

```
det(X)
```

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

2. Compute the trace of X (i.e. the sum of its diagonal elements).

```
sum(diag(X))
```

```
## [1] 15
```

3. Type `diag(X) <- c(7,8,9)` to change the diagonal elements. Compute the eigenvalues of (the new) X.

```
diag(X) <- c(7, 8, 9)
eigen(X)
```

```
## eigen() decomposition
## $values
## [1] 18.000000  4.732051  1.267949
##
## $vectors
##            [,1]        [,2]       [,3]
## [1,] -0.5773503 -0.90894503 -0.3239853
## [2,] -0.5773503  0.41277422 -0.6824097
## [3,] -0.5773503  0.05862061  0.6552484
```

Is X positive definite?

```
# Yes, since all eigenvalues are positive.
```

4. Invert X and compute the eigenvalues of X^{-1}.

```
solve(X)  # inverts X
```

```
##             [,1]        [,2]       [,3]
## [1,]  0.22222222  0.05555556 -0.2222222
## [2,]  0.05555556  0.38888889 -0.3888889
## [3,] -0.11111111 -0.27777778  0.4444444
```

```
eigen(solve(X))
```

```
## eigen() decomposition
## $values
```

```
## [1] 0.78867513 0.21132487 0.05555556
##
## $vectors
##            [,1]        [,2]      [,3]
## [1,] -0.3239853 -0.90894503 0.5773503
## [2,] -0.6824097  0.41277422 0.5773503
## [3,]  0.6552484  0.05862061 0.5773503
```

5. Define the vector $a = (1, 3, 2)$ and compute `a*X`, ‘`a%*%X`, and `X%*%a`.

```
a <- c(1, 3, 2)
a * X
```

```
##      [,1] [,2] [,3]
## [1,]    7    4    7
## [2,]    6   24   24
## [3,]    6   12   18
```

```
a %*% X
```

```
##      [,1] [,2] [,3]
## [1,]   19   40   49
```

```
X %*% a
```

```
##      [,1]
## [1,]   33
## [2,]   42
## [3,]   39
```

6. Compute the quadratic form $a'Xa$.

```
t(a) %*% X %*% a
```

```
##      [,1]
## [1,]  237
```

7. Define the matrices $I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$, $Y = \begin{bmatrix} 1 & 4 & 7 & 1 & 0 & 0 \\ 2 & 5 & 8 & 0 & 1 & 0 \\ 3 & 6 & 9 & 0 & 0 & 1 \end{bmatrix}$ and $Z = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$.

```
I <- diag(3)
Y <- cbind(matrix(1:9,3,3),I)
Z <- rbind(matrix(1:9,3,3),I)
```

8. Predict what the following commands will return:

```
cbind(1,X)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    7    4    7
## [2,]    1    2    8    8
## [3,]    1    3    6    9
```

```
rbind(Y,c(1,2,3))
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1    4    7    1    0    0
## [2,]    2    5    8    0    1    0
```

```
## [3,]    3    6    9    0    0    1
## [4,]    1    2    3    1    2    3
```

```
X%*%I
```

```
##      [,1] [,2] [,3]
## [1,]    7    4    7
## [2,]    2    8    8
## [3,]    3    6    9
```

```
dim(X%*%Y)
```

```
## [1] 3 6
```

```
t(Y)+Z
```

```
##      [,1] [,2] [,3]
## [1,]    2    6   10
## [2,]    6   10   14
## [3,]   10   14   18
## [4,]    2    0    0
## [5,]    0    2    0
## [6,]    0    0    2
```

```
solve(t(Z)%*%Z)%*%(t(Z)%*%Z)
```

```
##               [,1]          [,2]          [,3]
## [1,]  1.000000e+00 -1.265481e-14 -2.042897e-14
## [2,]  2.331468e-15  1.000000e+00 -2.275957e-15
## [3,] -1.221245e-15 -7.771561e-16  1.000000e+00
```

Set operations and special functions

1. Define the vectors

$$x = (-1, 0, 1, 4, 9, 2, 1, 4.5, 1.1, -0.9)$$
$$y = (1, 1, 2, 2, 3, 3, 4, 4, 5, NA)\,.$$

   and compute $x \cup y$. Determine the lengths of x, y and $x \cup y$.

```
x <- c(-1, 0, 1, 4, 9, 2, 1, 4.5, 1.1, -0.9)
y <- c(1, 1, 2, 2, 3, 3, 4, 4, 5, NA)
union(x, y)
```

```
##  [1] -1.0  0.0  1.0  4.0  9.0  2.0  4.5  1.1 -0.9  3.0  5.0   NA
```

```
length(x)
```

```
## [1] 10
```

```
length(y)
```

```
## [1] 10
```

```
length(union(x, y))
```

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

2. Count the number of elements of y that are element of x.

```
sum(unique(y) %in% x)
```

```
## [1] 3
```

```
length(intersect(y,x))
```

```
## [1] 3
```

3. Determine the length of the vector of unique elements of y.

```
length(unique(y))
```

```
## [1] 6
```

4. Compute the vector z with elements

$$z_i = \sum_{j=1}^{i} x_j$$

for $i = 1, \ldots, 10$.

```
z <- cumsum(x)
print(z)
```

```
##  [1] -1.0 -1.0  0.0  4.0 13.0 15.0 16.0 20.5 21.6 20.7
```

5. Find the position of the largest element of x.

```
which.max(x)
```

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

Sequences and replications

1. Generate the vectors

$$\begin{aligned}
x_1 &= (1, 2, 3, \ldots, 9) \\
x_2 &= (0, 1, 0, 1, 0, 1, 0, 1) \\
x_3 &= (1, 1, 1, 1, 1, 1, 1, 1) \\
x_4 &= (-1, 1, -1, 1, -1, 1) \\
x_5 &= (1980, 1985, 1990, \ldots, 2010) \\
x_6 &= (0, 0.01, 0.02, \ldots, 0.99, 1)
\end{aligned}$$

```
x1 <- 1:9  # or c(1:9)
x2 <- rep(c(0, 1), times = 4)
x3 <- rep(1, times = 8)
x4 <- rep(c(-1, 1), times = 3)
x5 <- seq(from = 1980, to = 2010, by = 5)
x6 <- seq(0, 1, by = 0.01)
```

2. Replications can also be generated for vectors of strings (characters). Type

```r
a <- c("a", "b", "c")
rep(a, 3)
```

```
## [1] "a" "b" "c" "a" "b" "c" "a" "b" "c"
```

```r
rep(a, times = 3)
```

```
## [1] "a" "b" "c" "a" "b" "c" "a" "b" "c"
```

```r
rep(a, each = 3)
```

```
## [1] "a" "a" "a" "b" "b" "b" "c" "c" "c"
```

3. Generate a grid of $n = 500$ equidistant points on the interval $[-\pi, \pi]$.

```r
seq(-pi, pi, length = 500)
```

4. Compare `1:10+1` and `1:(10+1)`.

```r
1:10 + 1     # sequence from 2 to 11
```

```
##  [1]  2  3  4  5  6  7  8  9 10 11
```

```r
1:(10 + 1)  # sequence from 1 to 11
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11
```

5. Predict what the following commands will return:

```r
rep("bla",10)
```

```
##  [1] "bla" "bla" "bla" "bla" "bla" "bla" "bla" "bla" "bla" "bla"
```

```r
rep(rep(1:3,2),each=4)
```

```
##  [1] 1 1 1 1 2 2 2 2 3 3 3 3 1 1 1 1 2 2 2 2 3 3 3 3
```

```r
rep(c(1,6,NA,2),times=c(2,2,5,3))
```

```
##  [1]  1  1  6  6 NA NA NA NA NA  2  2  2
```

Reading and writing text files

Consider the files **bsp1.txt**, **bsp2.txt** and **bsp3.txt**. The three files contain computer generated random numbers.

1. Read the file **bsp1.txt** into a data frame `bsp1`. Have a look at the file and the data format *before* you decide which reading command you use (`read.csv`, `read.csv2` or `read.table`). Print the data frame. If the data frame is too large for your screen, you can use the commands `head` and `tail` to print just parts of it.

```r
bsp1 <- read.csv2("data/bsp1.txt")  # German format (sep=';', dec=','), therefore read.csv2
print(bsp1)
```

```
##     Variable1 Variable2 Variable3 Variable4
## 1          15        12         4      1.12
## 2           0         0        14      0.23
## 3          11         3         3      0.12
## 4           0         1        12      1.94
```

```
## 5            0         1        10       0.66
## 6            2         6        11       0.54
## 7            6        10         6       0.07
## 8           14         2         2       0.75
## 9            0         0         2       1.41
## 10           3         6         3       2.11
## 11           0        11         1       0.83
## 12           1         5        10       0.11
## 13          10         0         8       0.74
## 14           4         2         1       0.65
## 15           1         1         2       0.13
## 16           1         1         3       1.03
## 17           0         5         3       2.35
## 18           3        17         6       0.27
## 19           2        14        10       0.02
## 20           9         8         4       0.02
## 21           7         2        21       0.22
## 22          15         0         4       0.21
## 23           1         0         7       0.73
## 24           0         8        13       0.08
## 25           0         1         0       1.15
## 26           0         0         0       0.34
## 27           0         9         4       1.66
## 28          10        14         1       0.92
## 29           3         3         3       1.69
## 30           0        24         3       0.06
## 31           0         1         1       0.40
## 32           5         4        16       0.10
## 33           0         0         0       0.82
## 34          12         0         1       1.49
## 35           7         8        18       0.74
## 36           7        10         6       0.44
## 37          19         3        26       0.55
## 38           1        13         4       0.74
## 39           1         1         0       0.49
## 40           5         5         1       0.71
## 41           1         1         0       0.21
## 42           3         1         2       1.51
## 43          12         9         1       0.42
## 44           7         1         0       0.67
## 45           1         5         5       2.19
## 46           7         5         0       0.93
## 47           3         3         2       0.35
## 48           1         1        11       1.28
## 49           6         4         2       3.01
## 50           1         6        21       0.47
## 51           4         2         2       0.86
## 52          15         2         2       0.03
## 53          15        13         2       0.38
## 54           3         1         0       0.28
## 55           2         1         0       1.24
## 56           8        17         0       2.26
## 57           3         7         3       1.11
## 58           2         7         2       0.48
```

```
## 59          0          1          8       0.48
## 60          7          7          0       0.36
## 61          5         51          0       1.06
## 62         11          0         14       0.20
## 63          5          0         12       2.03
## 64          5          1         23       0.32
## 65          1          4          2       0.51
## 66          3          4          1       0.32
## 67          2          1          0       0.78
## 68          5          2         13       1.48
## 69          1          1          1       0.67
## 70          2          0          0       0.06
## 71          2          1          6       0.51
## 72          2          3         15       0.64
## 73          3          1          7       0.55
## 74          8          4          7       0.03
## 75          3         10          3       1.65
## 76          0         11          6       0.80
## 77          4          1          7       1.18
## 78         11          0          7       0.61
## 79          3          1          7       0.97
## 80          6          2          1       1.79
## 81         11         10          1       0.81
## 82          3          1          1       0.11
## 83          0         27         10       1.07
## 84          2         10          2       0.85
## 85          0         15          4       1.77
## 86         36          0          2       0.98
## 87          1          4          1       1.12
## 88          0         12          8       0.79
## 89          4          2         11       1.01
## 90          8         11          0       0.43
## 91          9          0          1       0.24
## 92         23         17          2       2.58
## 93          0          4          6       0.41
## 94          5         11          1       0.46
## 95          9         14          4       2.08
## 96          4          3          0       0.41
## 97          6          8          0       1.20
## 98          4          5         24       0.56
## 99         12         27          6       1.77
## 100         6         15          4       1.26
```

```r
head(bsp1)
```

```
##   Variable1 Variable2 Variable3 Variable4
## 1        15        12         4      1.12
## 2         0         0        14      0.23
## 3        11         3         3      0.12
## 4         0         1        12      1.94
## 5         0         1        10      0.66
## 6         2         6        11      0.54
```

```r
tail(bsp1)
```

```
##     Variable1 Variable2 Variable3 Variable4
## 95          9        14         4      2.08
## 96          4         3         0      0.41
## 97          6         8         0      1.20
## 98          4         5        24      0.56
## 99         12        27         6      1.77
## 100         6        15         4      1.26
```

2. Read the files **bsp2.txt** and **bsp3.txt** into data frames `bsp2` and `bsp3`. Note that **bsp2.txt** contains both numeric and character entries. It is usually advisable to set the option `as.is=TRUE` when reading characters (strings).

```r
# international format (sep=',', dec='.'), therefore read.csv:
bsp2 <- read.csv("data/bsp2.txt")
# See help file for default settings
bsp3 <- read.table("data/bsp3.txt", dec = ".", sep = ",")
```

3. Print the class of `bsp2`, its dimension, and its variable names (use `names`).

```r
class(bsp2)
```

```
## [1] "data.frame"
```

```r
dim(bsp2)
```

```
## [1] 40  5
```

```r
names(bsp2)
```

```
## [1] "X" "Y" "Z" "U" "V"
```

4. Print a summary of `bsp3`.

```r
summary(bsp3)
```

```
##        V1                 V2                 V3                V4        
##  Min.   :-5.3800   Min.   :-8.0800   Min.   :-5.814   Min.   :-5.527  
##  1st Qu.:-0.2288   1st Qu.:-2.2403   1st Qu.:-1.028   1st Qu.:-0.476  
##  Median : 2.8390   Median : 0.2575   Median : 1.657   Median : 2.219  
##  Mean   : 2.3281   Mean   : 0.1810   Mean   : 1.520   Mean   : 1.911  
##  3rd Qu.: 4.6227   3rd Qu.: 2.7015   3rd Qu.: 3.517   3rd Qu.: 3.901  
##  Max.   : 9.2180   Max.   : 8.4800   Max.   : 8.773   Max.   : 9.045
```

5. Compute the mean and the standard deviation of each column of `bsp3`.

```r
apply(bsp3, 2, sd)
```

```
##       V1       V2       V3       V4 
## 3.230965 3.775867 3.422948 3.345753
```

```r
apply(bsp3, 2, mean)
```

```
##      V1      V2      V3      V4 
## 2.32806 0.18104 1.51952 1.91102
```

6. Create a small data frame `a` with two variables

x	y
1	4

x	y
2	5
3	6

and write it to a file **smalldataframe.csv** in your working directory.

```
x <- 1:3
y <- 4:6
smalldataframe <- data.frame(x, y)
write.csv2(smalldataframe, file = "data/smalldataframe.csv")  # write.csv2 for German Excel
```

7. Read the (large) file **lest2001.csv** into a data frame `x`. The file is the campus file of the German income tax records 2001 (the data are provided by the Research Data Centre of the Federal Statistical Office, they are described in **lest2001.pdf**). Take care to set the options of the `read.csv` or `read.table` command correctly. The data format is as follows:

- All data entries are separated by semi-colons.
- The first row contains the variables names.
- Missing values are denoted by a dot.
- Apart from the last column all data are integer values.
- The decimal sign in the last column is a dot.

Execute `y <- x\$zve`. The variable `y` now contains the taxable income (**zu** **v**ersteuerndes **E**inkommen). Compute its range, its median, its mean, its variance, and the 0.01- and 0.99-quantiles. Remember to include the option `na.rm=TRUE` in the functions.

```
lest2001 <- read.table("data/lest2001.csv", header = TRUE, na.strings= ".", dec = ".", sep = ";")
head(lest2001)
```

```
##   ef0 ef8 ef11 ef12 ef48 ab merker tabelle alter_a alter_b kinder region
## 1   1   0    2    2    4  1      1       2       4       4      1      1
## 2   2   1   NA    2    1  1      1       1       0       5      0      2
## 3   3   0    2   NA    3  1      1       2       4       4      1      1
## 4   4   0    6   NA    0  1      1       1       5       0      0      1
## 5   5   0    2   NA    1  1      1       1       3       0      0      1
## 6   6   0    2    2    4  1      1       2       3       3      1      2
##     sde   gde einkommen   zve est_tarif est_fest c65101 c65102 c65121
## 1 58258 58258     45239 45239      8044     7532     NA     NA     NA
## 2 17536 17536     15479 15479      1985     1985     NA     NA     NA
## 3 29024 29024     23573 23573      2087     2087     NA     NA     NA
## 4 10162 10162      5398  5398         0        0     NA     NA      1
## 5 68216 68216     66009 66009     22143    22143     NA     NA     NA
## 6 54512 54512     44106 44106      7691     7180     NA     NA     NA
##   c65122 c65141 c65142 c65161 c65162 c65221 c65222 c65241 c65242 c65261
## 1     NA     NA     NA      1      1     NA     NA      1     NA     NA
## 2     NA     NA     NA     NA      1     NA     NA     NA     NA     NA
## 3     NA     NA     NA      1     NA     NA     NA     NA     NA     NA
## 4     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA
## 5     NA     NA     NA      1     NA     NA     NA     NA     NA     NA
## 6     NA     NA     NA      1      1     NA     NA      1     NA     NA
##   c65262 c65413 c65427 c65479 c65490 c65727 samplingweight
## 1     NA      1     NA      1     NA     NA       111.1101
## 2     NA     NA      1     NA     NA     NA       111.0959
## 3     NA      1      1     NA     NA     NA       111.1093
## 4     NA      1      1     NA     NA     NA       111.0878
```

```
## 5     NA      1      1      NA      NA      NA       111.1004
## 6     NA      1      1       1      NA      NA       111.0951
```

```
y <- lest2001$zve
range(y, na.rm = TRUE)
```

```
## [1]  -509791 44887210
```

```
median(y, na.rm = TRUE)
```

```
## [1] 22457
```

```
var(y, na.rm = TRUE)
```

```
## [1] 296245129137
```

```
quantile(y, p = c(0.01, 0.99), na.rm = TRUE)
```

```
##         1%        99%
##   -5247.16 1282557.64
```

8. Import the dataset **swimming_pools.csv**. It contains data on swimming pools in Brisbane, Australia (Source: data.gov.au). The file contains the column names in the first row and uses a comma to separate values within rows.

```
pools <- read.csv("data/swimming_pools.csv", stringsAsFactors = FALSE)
str(pools)# Check the structure of pools
```

```
## 'data.frame':    20 obs. of  4 variables:
##  $ Name     : chr  "Acacia Ridge Leisure Centre" "Bellbowrie Pool" "Carole Park" "Centenary Pool (in
##  $ Address  : chr  "1391 Beaudesert Road, Acacia Ridge" "Sugarwood Street, Bellbowrie" "Cnr Boundary
##  $ Latitude : num  -27.6 -27.6 -27.6 -27.5 -27.4 ...
##  $ Longitude: num  153 153 153 153 153 ...
```

9. Import hotdogs.txt, containing information on sodium and calorie levels in different hotdogs (Source: UCLA). The dataset has 3 variables (type, calories and sodium), but the variable names are not available in the first line of the file. The file uses tabs as field separators.

```
hotdogs <- read.delim("data/hotdogs.txt", header = FALSE)
summary(hotdogs) # Summarize hotdogs
```

```
##        V1           V2              V3       
##  Beef   :20   Min.   : 86.0   Min.   :144.0  
##  Meat   :17   1st Qu.:132.0   1st Qu.:362.5  
##  Poultry:17   Median :145.0   Median :405.0  
##               Mean   :145.4   Mean   :424.8  
##               3rd Qu.:172.8   3rd Qu.:503.5  
##               Max.   :195.0   Max.   :645.0
```

```
hotdogs <- read.delim("data/hotdogs.txt", header = FALSE, col.names = c("type", "calories", "sodium"))
str(hotdogs)
```

```
## 'data.frame':    54 obs. of  3 variables:
##  $ type    : Factor w/ 3 levels "Beef","Meat",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ calories: int  186 181 176 149 184 190 158 139 175 148 ...
##  $ sodium  : int  495 477 425 322 482 587 370 322 479 375 ...
```

```
# Edit the colClasses argument to import the data correctly: hotdogs2
hotdogs2 <- read.delim("data/hotdogs.txt", header = FALSE,
                       col.names = c("type", "calories", "sodium"),
```

```r
                        colClasses = c("factor", "NULL", "numeric"))
str(hotdogs2)
```

```
## 'data.frame':    54 obs. of  2 variables:
##  $ type  : Factor w/ 3 levels "Beef","Meat",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ sodium: num  495 477 425 322 482 587 370 322 479 375 ...
```

Import data using readr

Use the package `readr` to import the datasets used in the previous exercise, i.e. **bsp1.txt**, **bsp2.txt**, **bsp3.txt**, **lest2001.csv**, **swimming_pools.csv** and **hotdogs.txt**.

```r
library(readr)
```

Import Excel data

1. Load the `readxl` package.

```r
library(readxl)
```

2. Print the names of all worksheets in the excel file **urbanpop.xlsx**. This dataset is a subset of the gapminder dataset.

```r
excel_sheets("data/urbanpop.xlsx")
```

```
## [1] "1960-1966" "1967-1974" "1975-2011"
```

3. Now read the sheets, one by one, using `read_excel` and put these into a list.

```r
pop_1 <- read_excel("data/urbanpop.xlsx", sheet = 1)
pop_2 <- read_excel("data/urbanpop.xlsx", sheet = 2)
pop_3 <- read_excel("data/urbanpop.xlsx", sheet = 3)
pop_list <- list(pop_1, pop_2, pop_3)
pop_list
```

```
## [[1]]
## # A tibble: 209 x 8
##    country          `1960`   `1961`   `1962`  `1963`  `1964`  `1965`  `1966`
##    <chr>             <dbl>    <dbl>    <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
##  1 Afghanistan      769308   8.15e5   8.59e5  9.04e5  9.51e5  1.00e6  1.06e6
##  2 Albania          494443   5.12e5   5.29e5  5.47e5  5.66e5  5.84e5  6.03e5
##  3 Algeria         3293999   3.52e6   3.74e6  3.97e6  4.22e6  4.49e6  4.65e6
##  4 American Sam~        NA   1.37e4   1.42e4  1.48e4  1.54e4  1.60e4  1.67e4
##  5 Andorra              NA   8.72e3   9.70e3  1.07e4  1.19e4  1.31e4  1.42e4
##  6 Angola           521205   5.48e5   5.80e5  6.12e5  6.45e5  6.79e5  7.18e5
##  7 Antigua and ~     21699   2.16e4   2.17e4  2.17e4  2.18e4  2.19e4  2.20e4
##  8 Argentina      15224096   1.55e7   1.59e7  1.63e7  1.67e7  1.70e7  1.74e7
##  9 Armenia          957974   1.01e6   1.06e6  1.12e6  1.17e6  1.23e6  1.28e6
## 10 Aruba             24996   2.81e4   2.85e4  2.88e4  2.89e4  2.91e4  2.93e4
## # ... with 199 more rows
```

```
##
## [[2]]
## # A tibble: 209 x 9
##    country     `1967`  `1968`  `1969`  `1970`  `1971`  `1972` `1973` `1974`
##    <chr>        <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>  <dbl>  <dbl>
##  1 Afghanist~  1.12e6  1.18e6  1.25e6  1.32e6  1.41e6  1.50e6 1.60e6 1.70e6
##  2 Albania     6.21e5  6.40e5  6.59e5  6.78e5  6.99e5  7.20e5 7.42e5 7.63e5
##  3 Algeria     4.83e6  5.02e6  5.22e6  5.43e6  5.62e6  5.82e6 6.02e6 6.24e6
##  4 American ~  1.73e4  1.80e4  1.86e4  1.92e4  1.98e4  2.03e4 2.07e4 2.12e4
##  5 Andorra     1.54e4  1.67e4  1.81e4  1.95e4  2.09e4  2.24e4 2.39e4 2.55e4
##  6 Angola      7.57e5  7.98e5  8.41e5  8.86e5  9.55e5  1.03e6 1.10e6 1.18e6
##  7 Antigua a~  2.21e4  2.21e4  2.22e4  2.22e4  2.26e4  2.29e4 2.32e4 2.35e4
##  8 Argentina   1.78e7  1.81e7  1.85e7  1.89e7  1.93e7  1.98e7 2.02e7 2.07e7
##  9 Armenia     1.34e6  1.39e6  1.45e6  1.51e6  1.56e6  1.62e6 1.68e6 1.74e6
## 10 Aruba       2.94e4  2.96e4  2.97e4  2.99e4  3.01e4  3.03e4 3.05e4 3.06e4
## # ... with 199 more rows
##
## [[3]]
## # A tibble: 209 x 38
##    country `1975` `1976` `1977` `1978` `1979` `1980` `1981` `1982` `1983`
##    <chr>    <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
##  1 Afghan~ 1.79e6 1.91e6 2.02e6 2.14e6 2.27e6 2.40e6 2.49e6 2.59e6 2.69e6
##  2 Albania 7.85e5 8.08e5 8.31e5 8.54e5 8.78e5 9.02e5 9.27e5 9.52e5 9.78e5
##  3 Algeria 6.46e6 6.77e6 7.10e6 7.45e6 7.81e6 8.19e6 8.64e6 9.11e6 9.59e6
##  4 Americ~ 2.16e4 2.20e4 2.25e4 2.29e4 2.35e4 2.42e4 2.52e4 2.63e4 2.77e4
##  5 Andorra 2.70e4 2.84e4 2.97e4 3.10e4 3.26e4 3.44e4 3.64e4 3.86e4 4.10e4
##  6 Angola  1.27e6 1.37e6 1.48e6 1.60e6 1.72e6 1.86e6 2.02e6 2.19e6 2.37e6
##  7 Antigu~ 2.38e4 2.40e4 2.42e4 2.43e4 2.44e4 2.43e4 2.42e4 2.39e4 2.36e4
##  8 Argent~ 2.11e7 2.16e7 2.20e7 2.24e7 2.29e7 2.33e7 2.38e7 2.43e7 2.48e7
##  9 Armenia 1.80e6 1.85e6 1.90e6 1.95e6 2.00e6 2.05e6 2.08e6 2.12e6 2.16e6
## 10 Aruba   3.07e4 3.06e4 3.05e4 3.04e4 3.03e4 3.03e4 3.06e4 3.09e4 3.14e4
## # ... with 199 more rows, and 28 more variables: `1984` <dbl>,
## #   `1985` <dbl>, `1986` <dbl>, `1987` <dbl>, `1988` <dbl>, `1989` <dbl>,
## #   `1990` <dbl>, `1991` <dbl>, `1992` <dbl>, `1993` <dbl>, `1994` <dbl>,
## #   `1995` <dbl>, `1996` <dbl>, `1997` <dbl>, `1998` <dbl>, `1999` <dbl>,
## #   `2000` <dbl>, `2001` <dbl>, `2002` <dbl>, `2003` <dbl>, `2004` <dbl>,
## #   `2005` <dbl>, `2006` <dbl>, `2007` <dbl>, `2008` <dbl>, `2009` <dbl>,
## #   `2010` <dbl>, `2011` <dbl>
```

Import data using haven

1. Load the `haven` package.

```
library("haven")
```

2. In this exercise, you will work with data on yearly import and export numbers of sugar, both in USD and in weight. The data is given in **trade.dta**. Load the data using `read_dta` and have a look at the structure. Convert the values in Date column to dates.

```
sugar <- read_dta("data/trade.dta")
sugar
```

```
## # A tibble: 10 x 5
##    Date         Import Weight_I    Export  Weight_E
##    <dbl+lbl>     <dbl>    <dbl>     <dbl>     <dbl>
##  1 10         37664782 54029106  54505513  93350013
##  2  9         16316512 21584365 102700010 158000010
##  3  8         11082246 14526089  37935000  88000000
##  4  7         35677943 55034932  48515008 112000005
##  5  6          9879878 14806865  71486545 131800000
##  6  5          1539992  1749318  12311696  18500014
##  7  4            28021    54567  16489813  39599944
##  8  3             2652     3821  29273920 102072480
##  9  2          7067402 23722957  46497438 147583380
## 10  1          1033672  1964980  27131638  78268792
```

```r
str(sugar) # Structure of sugar
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    10 obs. of  5 variables:
##  $ Date    : 'labelled' num  10 9 8 7 6 5 4 3 2 1
##   ..- attr(*, "label")= chr "Date"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##   ..- attr(*, "labels")= Named num  1 2 3 4 5 6 7 8 9 10
##   .. ..- attr(*, "names")= chr  "2004-12-31" "2005-12-31" "2006-12-31" "2007-12-31" ...
##  $ Import  : num  37664782 16316512 11082246 35677943 9879878 ...
##   ..- attr(*, "label")= chr "Import"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ Weight_I: num  54029106 21584365 14526089 55034932 14806865 ...
##   ..- attr(*, "label")= chr "Weight_I"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ Export  : num  5.45e+07 1.03e+08 3.79e+07 4.85e+07 7.15e+07 ...
##   ..- attr(*, "label")= chr "Export"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ Weight_E: num  9.34e+07 1.58e+08 8.80e+07 1.12e+08 1.32e+08 ...
##   ..- attr(*, "label")= chr "Weight_E"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  - attr(*, "label")= chr "Written by R."
```

```r
sugar$Date <- as.Date(as_factor(sugar$Date))
str(sugar)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    10 obs. of  5 variables:
##  $ Date    : Date, format: "2013-12-31" "2012-12-31" ...
##  $ Import  : num  37664782 16316512 11082246 35677943 9879878 ...
##   ..- attr(*, "label")= chr "Import"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ Weight_I: num  54029106 21584365 14526089 55034932 14806865 ...
##   ..- attr(*, "label")= chr "Weight_I"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ Export  : num  5.45e+07 1.03e+08 3.79e+07 4.85e+07 7.15e+07 ...
##   ..- attr(*, "label")= chr "Export"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ Weight_E: num  9.34e+07 1.58e+08 8.80e+07 1.12e+08 1.32e+08 ...
##   ..- attr(*, "label")= chr "Weight_E"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  - attr(*, "label")= chr "Written by R."
```

Import data using `foreign`

1. Load the `foreign` package.

```
library(foreign)
```

1. In this exercise, you will import data on the US presidential elections in the year 2000. The data in **florida.dta** contains the total numbers of votes for each of the four candidates as well as the total number of votes per election area in the state of Florida. Import **florida.dta** and name the resulting data frame florida.

```
florida <- read.dta("data/florida.dta")
tail(florida)
```

```
##     gore   bush buchanan nader  total
## 62  2647   4051       27    59   6784
## 63  1399   2326       26    29   3780
## 64 97063  82214      396  2436 182109
## 65  3835   4511       46   149   8541
## 66  5637  12176      120   265  18198
## 67  2796   4983       88    93   7960
```

2. The arguments you will use most often are `convert.dates`, `convert.factors`, `missing.type` and `convert.underscore`. Consider the dataset **edequality** which contains socio-economic measures and access to education for different individuals (source: Worldbank).

```
edu_equal_1 <- read.dta("data/edequality.dta")
str(edu_equal_1)
```

```
## 'data.frame':    12214 obs. of  27 variables:
##  $ hhid             : num  1 1 1 2 2 3 4 4 5 6 ...
##  $ hhweight         : num  627 627 627 627 627 ...
##  $ location         : Factor w/ 2 levels "urban location",..: 1 1 1 1 1 2 2 2 1 1 ...
##  $ region           : Factor w/ 9 levels "Sofia city","Bourgass",..: 8 8 8 9 9 4 4 4 8 8 ...
##  $ ethnicity_head   : Factor w/ 4 levels "Bulgaria","Turks",..: 2 2 2 1 1 1 1 1 1 1 ...
##  $ age              : num  37 11 8 73 70 75 79 80 82 83 ...
##  $ gender           : Factor w/ 2 levels "male","female": 2 2 1 1 2 1 1 2 2 2 ...
##  $ relation         : Factor w/ 9 levels "head                    ",..: 1 3 3 1 2 1 1 2 1 1 ...
##  $ literate         : Factor w/ 2 levels "no","yes": 1 2 2 2 2 2 2 2 2 2 ...
##  $ income_mnt       : num  13.3 13.3 13.3 142.5 142.5 ...
##  $ income           : num  160 160 160 1710 1710 ...
##  $ aggregate        : num  1042 1042 1042 3271 3271 ...
##  $ aggr_ind_annual  : num  347 347 347 1635 1635 ...
##  $ educ_completed   : int  2 4 4 4 3 3 3 3 4 4 ...
##  $ grade_complete   : num  4 3 0 3 4 4 4 4 5 5 ...
##  $ grade_all        : num  4 11 8 11 8 8 8 8 13 13 ...
##  $ unemployed       : int  2 1 1 1 1 1 1 1 1 1 ...
##  $ reason_OLF       : int  NA NA NA 3 3 3 9 9 3 3 ...
##  $ sector           : int  NA NA NA NA NA NA 1 1 NA NA ...
##  $ occupation       : int  NA NA NA NA NA NA 5 5 NA NA ...
##  $ earn_mont        : num  0 0 0 0 0 0 20 20 0 0 ...
##  $ earn_ann         : num  0 0 0 0 0 0 240 240 0 0 ...
##  $ hours_week       : num  NA NA NA NA NA NA 30 35 NA NA ...
##  $ hours_mnt        : num  NA NA NA NA NA ...
##  $ fulltime         : int  NA NA NA NA NA NA 1 1 NA NA ...
##  $ hhexp            : num  100 100 100 343 343 ...
```

```
##  $ legacy_pension_amt: num  NA NA NA NA NA NA NA NA NA NA ...
##  - attr(*, "datalabel")= chr ""
##  - attr(*, "time.stamp")= chr ""
##  - attr(*, "formats")= chr  "%9.0g" "%9.0g" "%9.0g" "%9.0g" ...
##  - attr(*, "types")= int  100 100 108 108 108 100 108 108 108 100 ...
##  - attr(*, "val.labels")= chr  "" "" "location" "region" ...
##  - attr(*, "var.labels")= chr  "hhid" "hhweight" "location" "region" ...
##  - attr(*, "expansion.fields")=List of 12
##   ..$ : chr  "_dta" "_svy_su1" "cluster"
##   ..$ : chr  "_dta" "_svy_strata1" "strata"
##   ..$ : chr  "_dta" "_svy_stages" "1"
##   ..$ : chr  "_dta" "_svy_version" "2"
##   ..$ : chr  "_dta" "__XijVarLabcons" "(sum) cons"
##   ..$ : chr  "_dta" "ReS_Xij" "cons"
##   ..$ : chr  "_dta" "ReS_str" "0"
##   ..$ : chr  "_dta" "ReS_j" "group"
##   ..$ : chr  "_dta" "ReS_ver" "v.2"
##   ..$ : chr  "_dta" "ReS_i" "hhid dur"
##   ..$ : chr  "_dta" "note1" "variables g1pc, g2pc, g3pc, g4pc, g5pc, g7pc, g8pc, g9pc, g10pc, g11pc,
##   ..$ : chr  "_dta" "note0" "1"
##  - attr(*, "version")= int 7
##  - attr(*, "label.table")=List of 12
##   ..$ location: Named int  1 2
##   .. ..- attr(*, "names")= chr  "urban location" "rural location"
##   ..$ region  : Named int  1 2 3 4 5 6 7 8 9
##   .. ..- attr(*, "names")= chr  "Sofia city" "Bourgass" "Varna" "Lovetch" ...
##   ..$ ethnic  : Named int  1 2 3 4
##   .. ..- attr(*, "names")= chr  "Bulgaria" "Turks" "Roma" "Other"
##   ..$ s2_q2   : Named int  1 2
##   .. ..- attr(*, "names")= chr  "male" "female"
##   ..$ s2_q3   : Named int  1 2 3 4 5 6 7 8 9
##   .. ..- attr(*, "names")= chr  "head                         " "spouse/partner             " "child
##   ..$ lit     : Named int  1 2
##   .. ..- attr(*, "names")= chr  "no" "yes"
##   ..$         : Named int  1 2 3 4
##   .. ..- attr(*, "names")= chr  "never attanded" "primary" "secondary" "postsecondary"
##   ..$         : Named int  1 2
##   .. ..- attr(*, "names")= chr  "Not unemployed" "Unemployed"
##   ..$         : Named int  1 2 3 4 5 6 7 8 9 10
##   .. ..- attr(*, "names")= chr  "student" "housewife/childcare" "in retirement" "illness, disability
##   ..$         : Named int  1 2 3 4 5 6 7 8 9 10
##   .. ..- attr(*, "names")= chr  "agriculture" "mining" "manufacturing" "utilities" ...
##   ..$         : Named int  1 2 3 4 5
##   .. ..- attr(*, "names")= chr  "private company" "public works program" "government,public sector, a
##   ..$         : Named int  1 2
##   .. ..- attr(*, "names")= chr  "no" "yes"
```

```r
edu_equal_2 <- read.dta("data/edequality.dta", convert.factors = FALSE)
str(edu_equal_2)
```

```
## 'data.frame':    12214 obs. of  27 variables:
##  $ hhid              : num  1 1 1 2 2 3 4 4 5 6 ...
##  $ hhweight          : num  627 627 627 627 627 ...
##  $ location          : int  1 1 1 1 1 2 2 2 1 1 ...
##  $ region            : int  8 8 8 9 9 4 4 4 8 8 ...
```

```
##  $ ethnicity_head    : int  2 2 2 1 1 1 1 1 1 1 ...
##  $ age               : num  37 11 8 73 70 75 79 80 82 83 ...
##  $ gender            : int  2 2 1 1 2 1 1 2 2 2 ...
##  $ relation          : int  1 3 3 1 2 1 1 2 1 1 ...
##  $ literate          : int  1 2 2 2 2 2 2 2 2 2 ...
##  $ income_mnt        : num  13.3 13.3 13.3 142.5 142.5 ...
##  $ income            : num  160 160 160 1710 1710 ...
##  $ aggregate         : num  1042 1042 1042 3271 3271 ...
##  $ aggr_ind_annual   : num  347 347 347 1635 1635 ...
##  $ educ_completed    : int  2 4 4 4 3 3 3 3 4 4 ...
##  $ grade_complete    : num  4 3 0 3 4 4 4 4 5 5 ...
##  $ grade_all         : num  4 11 8 11 8 8 8 8 13 13 ...
##  $ unemployed        : int  2 1 1 1 1 1 1 1 1 1 ...
##  $ reason_OLF        : int  NA NA NA 3 3 3 9 9 3 3 ...
##  $ sector            : int  NA NA NA NA NA NA 1 1 NA NA ...
##  $ occupation        : int  NA NA NA NA NA NA 5 5 NA NA ...
##  $ earn_mont         : num  0 0 0 0 0 0 20 20 0 0 ...
##  $ earn_ann          : num  0 0 0 0 0 0 240 240 0 0 ...
##  $ hours_week        : num  NA NA NA NA NA NA 30 35 NA NA ...
##  $ hours_mnt         : num  NA NA NA NA NA ...
##  $ fulltime          : int  NA NA NA NA NA NA 1 1 NA NA ...
##  $ hhexp             : num  100 100 100 343 343 ...
##  $ legacy_pension_amt: num  NA NA NA NA NA NA NA NA NA NA ...
##  - attr(*, "datalabel")= chr ""
##  - attr(*, "time.stamp")= chr ""
##  - attr(*, "formats")= chr  "%9.0g" "%9.0g" "%9.0g" "%9.0g" ...
##  - attr(*, "types")= int  100 100 108 108 108 100 108 108 108 100 ...
##  - attr(*, "val.labels")= chr  "" "" "location" "region" ...
##  - attr(*, "var.labels")= chr  "hhid" "hhweight" "location" "region" ...
##  - attr(*, "expansion.fields")=List of 12
##   ..$ : chr  "_dta" "_svy_su1" "cluster"
##   ..$ : chr  "_dta" "_svy_strata1" "strata"
##   ..$ : chr  "_dta" "_svy_stages" "1"
##   ..$ : chr  "_dta" "_svy_version" "2"
##   ..$ : chr  "_dta" "__XijVarLabcons" "(sum) cons"
##   ..$ : chr  "_dta" "ReS_Xij" "cons"
##   ..$ : chr  "_dta" "ReS_str" "0"
##   ..$ : chr  "_dta" "ReS_j" "group"
##   ..$ : chr  "_dta" "ReS_ver" "v.2"
##   ..$ : chr  "_dta" "ReS_i" "hhid dur"
##   ..$ : chr  "_dta" "note1" "variables g1pc, g2pc, g3pc, g4pc, g5pc, g7pc, g8pc, g9pc, g10pc, g11pc,
##   ..$ : chr  "_dta" "note0" "1"
##  - attr(*, "version")= int 7
##  - attr(*, "label.table")=List of 12
##   ..$ location: Named int  1 2
##   .. ..- attr(*, "names")= chr  "urban location" "rural location"
##   ..$ region  : Named int  1 2 3 4 5 6 7 8 9
##   .. ..- attr(*, "names")= chr  "Sofia city" "Bourgass" "Varna" "Lovetch" ...
##   ..$ ethnic  : Named int  1 2 3 4
##   .. ..- attr(*, "names")= chr  "Bulgaria" "Turks" "Roma" "Other"
##   ..$ s2_q2   : Named int  1 2
##   .. ..- attr(*, "names")= chr  "male" "female"
##   ..$ s2_q3   : Named int  1 2 3 4 5 6 7 8 9
##   .. ..- attr(*, "names")= chr  "head                      " "spouse/partner            " "child
```

```
##   ..$ lit     : Named int  1 2
##   .. ..- attr(*, "names")= chr  "no" "yes"
##   ..$         : Named int  1 2 3 4
##   .. ..- attr(*, "names")= chr  "never attanded" "primary" "secondary" "postsecondary"
##   ..$         : Named int  1 2
##   .. ..- attr(*, "names")= chr  "Not unemployed" "Unemployed"
##   ..$         : Named int  1 2 3 4 5 6 7 8 9 10
##   .. ..- attr(*, "names")= chr  "student" "housewife/childcare" "in retirement" "illness, disability
##   ..$         : Named int  1 2 3 4 5 6 7 8 9 10
##   .. ..- attr(*, "names")= chr  "agriculture" "mining" "manufacturing" "utilities" ...
##   ..$         : Named int  1 2 3 4 5
##   .. ..- attr(*, "names")= chr  "private company" "public works program" "government,public sector, a
##   ..$         : Named int  1 2
##   .. ..- attr(*, "names")= chr  "no" "yes"
```

```
edu_equal_3 <- read.dta("data/edequality.dta", convert.underscore = TRUE)
str(edu_equal_3)
```

```
## 'data.frame':    12214 obs. of  27 variables:
##  $ hhid              : num  1 1 1 2 2 3 4 4 5 6 ...
##  $ hhweight          : num  627 627 627 627 627 ...
##  $ location          : Factor w/ 2 levels "urban location",..: 1 1 1 1 1 2 2 2 1 1 ...
##  $ region            : Factor w/ 9 levels "Sofia city","Bourgass",..: 8 8 8 9 9 4 4 4 8 8 ...
##  $ ethnicity.head    : Factor w/ 4 levels "Bulgaria","Turks",..: 2 2 2 1 1 1 1 1 1 1 ...
##  $ age               : num  37 11 8 73 70 75 79 80 82 83 ...
##  $ gender            : Factor w/ 2 levels "male","female": 2 2 1 1 2 1 1 2 2 2 ...
##  $ relation          : Factor w/ 9 levels "head                      ",..: 1 3 3 1 2 1 1 2 1 1 ...
##  $ literate          : Factor w/ 2 levels "no","yes": 1 2 2 2 2 2 2 2 2 2 ...
##  $ income.mnt        : num  13.3 13.3 13.3 142.5 142.5 ...
##  $ income            : num  160 160 160 1710 1710 ...
##  $ aggregate         : num  1042 1042 1042 3271 3271 ...
##  $ aggr.ind.annual   : num  347 347 347 1635 1635 ...
##  $ educ.completed    : int  2 4 4 4 3 3 3 3 4 4 ...
##  $ grade.complete    : num  4 3 0 3 4 4 4 4 5 5 ...
##  $ grade.all         : num  4 11 8 11 8 8 8 8 13 13 ...
##  $ unemployed        : int  2 1 1 1 1 1 1 1 1 1 ...
##  $ reason.OLF        : int  NA NA NA 3 3 3 9 9 3 3 ...
##  $ sector            : int  NA NA NA NA NA NA 1 1 NA NA ...
##  $ occupation        : int  NA NA NA NA NA NA 5 5 NA NA ...
##  $ earn.mont         : num  0 0 0 0 0 0 20 20 0 0 ...
##  $ earn.ann          : num  0 0 0 0 0 0 240 240 0 0 ...
##  $ hours.week        : num  NA NA NA NA NA NA 30 35 NA NA ...
##  $ hours.mnt         : num  NA NA NA NA NA ...
##  $ fulltime          : int  NA NA NA NA NA NA 1 1 NA NA ...
##  $ hhexp             : num  100 100 100 343 343 ...
##  $ legacy.pension.amt: num  NA NA NA NA NA NA NA NA NA NA ...
##  - attr(*, "datalabel")= chr ""
##  - attr(*, "time.stamp")= chr ""
##  - attr(*, "formats")= chr  "%9.0g" "%9.0g" "%9.0g" "%9.0g" ...
##  - attr(*, "types")= int  100 100 108 108 108 100 108 108 108 100 ...
##  - attr(*, "val.labels")= chr  "" "" "location" "region" ...
##  - attr(*, "var.labels")= chr  "hhid" "hhweight" "location" "region" ...
##  - attr(*, "expansion.fields")=List of 12
##   ..$ : chr  "_dta" "_svy_su1" "cluster"
##   ..$ : chr  "_dta" "_svy_strata1" "strata"
```

```
##   ..$ : chr  "_dta" "_svy_stages" "1"
##   ..$ : chr  "_dta" "_svy_version" "2"
##   ..$ : chr  "_dta" "__XijVarLabcons" "(sum) cons"
##   ..$ : chr  "_dta" "ReS_Xij" "cons"
##   ..$ : chr  "_dta" "ReS_str" "0"
##   ..$ : chr  "_dta" "ReS_j" "group"
##   ..$ : chr  "_dta" "ReS_ver" "v.2"
##   ..$ : chr  "_dta" "ReS_i" "hhid dur"
##   ..$ : chr  "_dta" "note1" "variables g1pc, g2pc, g3pc, g4pc, g5pc, g7pc, g8pc, g9pc, g10pc, g11pc,
##   ..$ : chr  "_dta" "note0" "1"
##  - attr(*, "version")= int 7
##  - attr(*, "label.table")=List of 12
##   ..$ location: Named int  1 2
##   .. ..- attr(*, "names")= chr  "urban location" "rural location"
##   ..$ region  : Named int  1 2 3 4 5 6 7 8 9
##   .. ..- attr(*, "names")= chr  "Sofia city" "Bourgass" "Varna" "Lovetch" ...
##   ..$ ethnic  : Named int  1 2 3 4
##   .. ..- attr(*, "names")= chr  "Bulgaria" "Turks" "Roma" "Other"
##   ..$ s2_q2   : Named int  1 2
##   .. ..- attr(*, "names")= chr  "male" "female"
##   ..$ s2_q3   : Named int  1 2 3 4 5 6 7 8 9
##   .. ..- attr(*, "names")= chr  "head                      " "spouse/partner            " "child
##   ..$ lit     : Named int  1 2
##   .. ..- attr(*, "names")= chr  "no" "yes"
##   ..$         : Named int  1 2 3 4
##   .. ..- attr(*, "names")= chr  "never attanded" "primary" "secondary" "postsecondary"
##   ..$         : Named int  1 2
##   .. ..- attr(*, "names")= chr  "Not unemployed" "Unemployed"
##   ..$         : Named int  1 2 3 4 5 6 7 8 9 10
##   .. ..- attr(*, "names")= chr  "student" "housewife/childcare" "in retirement" "illness, disability
##   ..$         : Named int  1 2 3 4 5 6 7 8 9 10
##   .. ..- attr(*, "names")= chr  "agriculture" "mining" "manufacturing" "utilities" ...
##   ..$         : Named int  1 2 3 4 5
##   .. ..- attr(*, "names")= chr  "private company" "public works program" "government,public sector, a
##   ..$         : Named int  1 2
##   .. ..- attr(*, "names")= chr  "no" "yes"
```

Indexing vectors

Define the following vectors

$$x = \begin{pmatrix} 1 \\ 1.1 \\ 9 \\ 8 \\ 1 \\ 4 \\ 4 \\ 1 \end{pmatrix}, \quad y = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 4 \\ 3 \\ 2 \\ NA \end{pmatrix}, \quad z = \begin{pmatrix} TRUE \\ TRUE \\ FALSE \\ FALSE \\ TRUE \\ FALSE \\ FALSE \\ FALSE \end{pmatrix}$$

```r
x <- c(1, 1.1, 9, 8, 1, 4, 4, 1)
y <- c(1, 2, 3, 4, 4, 3, 2, NA)
z <- c(TRUE, TRUE, FALSE, FALSE, TRUE, FALSE, FALSE, FALSE)
```

1. Predict what the following commands will return (and then check if you are right):

```r
x[-2]
```

```
## [1] 1 9 8 1 4 4 1
```

```r
x[2:5]
```

```
## [1] 1.1 9.0 8.0 1.0
```

```r
x[c(1,5,8)]
```

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

```r
x[-c(1,5,8)]
```

```
## [1] 1.1 9.0 8.0 4.0 4.0
```

```r
x[y]
```

```
## [1] 1.0 1.1 9.0 8.0 8.0 9.0 1.1  NA
```

```r
x[seq(2,8,by=2)]
```

```
## [1] 1.1 8.0 4.0 1.0
```

```r
x[rep(1:3,4)]
```

```
##  [1] 1.0 1.1 9.0 1.0 1.1 9.0 1.0 1.1 9.0 1.0 1.1 9.0
```

2. Predict what the following commands will return (and then check if you are right):

```r
y[z]
```

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

```r
y[!z]
```

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

```r
y[x>2]
```

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

```r
y[x==1]
```

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

```r
y
```

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

```r
x[!is.na(y)]
```

```
## [1] 1.0 1.1 9.0 8.0 1.0 4.0 4.0
```

```r
y[!is.na(y)]
```

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

3. Indexing is not only used to read certain elements of a vector but also to change them. Execute `x2 <- x` to make a copy of `x`. Change all elements of `x2` that have the value 4 to the value -4. Print `x2`.

```r
x2 <- x
x2[x2 == 4] <- -4
print(x2)
```

```
## [1]  1.0  1.1  9.0  8.0  1.0 -4.0 -4.0  1.0
```

4. Change all elements of `x2` that have the value 1 to a missing value (`NA`). Print `x2`.

```r
x2[x2 == 1] <- NA
print(x2)
```

```
## [1]   NA  1.1  9.0  8.0   NA -4.0 -4.0   NA
```

5. Execute `x2[z] <- 0`. Print `x2`.

```r
x2[z] <- 0
print(x2)
```

```
## [1]  0  0  9  8  0 -4 -4 NA
```

Indexing matrices

Define the matrix `x <- matrix(c(1:12,12:1),4,6)`.

```r
x <- matrix(c(1:12, 12:1), 4, 6)
```

1. Predict what the following commands will return (and then check if you are right):

```r
x[1,3]
```

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

```r
x[,5]
```

```
## [1] 8 7 6 5
```

```r
x[2,]
```

```
## [1]  2  6 10 11  7  3
```

```r
x[,-3]
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    5   12    8    4
## [2,]    2    6   11    7    3
## [3,]    3    7   10    6    2
## [4,]    4    8    9    5    1
```

```r
x[-4,]
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1    5    9   12    8    4
## [2,]    2    6   10   11    7    3
## [3,]    3    7   11   10    6    2
```

```r
x[2:3,3:4]
```

```
##      [,1] [,2]
## [1,]   10   11
```

```
## [2,]   11   10
```

```r
x[2:4,4]
```

```
## [1] 11 10  9
```

2. Predict what the following commands will return (and then check if you are right):

```r
x[x>5]
```

```
##  [1]  6  7  8  9 10 11 12 12 11 10  9  8  7  6
```

```r
x[,x[1,]<=5]
```

```
##      [,1] [,2] [,3]
## [1,]    1    5    4
## [2,]    2    6    3
## [3,]    3    7    2
## [4,]    4    8    1
```

```r
x[x[,2]>6,]
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    3    7   11   10    6    2
## [2,]    4    8   12    9    5    1
```

```r
x[x[,2]>6,4:6]
```

```
##      [,1] [,2] [,3]
## [1,]   10    6    2
## [2,]    9    5    1
```

```r
x[x[,1]<3 & x[,2]<6,]
```

```
## [1]  1  5  9 12  8  4
```

3. Print all rows where column 5 is at least three times larger than column 6.

```r
x[x[, 5]>= (3 * x[, 6]) , ]
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    3    7   11   10    6    2
## [2,]    4    8   12    9    5    1
```

4. Count the number of elements of `x` that are larger than 7.

```r
length(x[x > 7])  # or: sum(x>7)
```

```
## [1] 10
```

5. Count the number of elements in row 2 that are smaller than their neighbors in row 1.

```r
sum(x[2, ] < x[1, ])
```

```
## [1] 3
```

6. Count the number of elements of `x` that are larger than their left neighbor.

```r
sum(x[, 2:6] > x[, 1:5])  # alternativ: sum(x[,-1]>x[,-6])
```

```
## [1] 10
```

Indexing dataframes

Load the data set **bsp2.txt** as data frame `bsp2` and print it.

```
bsp2 <- read.csv("data/bsp2.txt",as.is=TRUE)
print(bsp2)
```

```
##        X     Y     Z U V
## 1  2.411 2.317 5.209 B M
## 2  2.469 2.116 2.566 D M
## 3  1.066 5.471 4.856 D M
## 4  2.264 2.107 2.647 C M
## 5  2.775 3.136 3.221 C M
## 6  2.542 3.072 2.999 A M
## 7  2.200 3.272 3.174 D F
## 8  8.947 3.058 3.178 B M
## 9  5.784 2.092 3.364 B F
## 10 2.180 2.745 3.731 B M
## 11 1.580 2.008 4.463 B M
## 12 3.590 2.214 1.556 C F
## 13 4.316 1.868 0.731 D M
## 14 2.509 6.982 1.265 B M
## 15 1.436 4.650 2.002 C F
## 16 2.275 3.761 1.376 D F
## 17 3.648 1.688 1.837 C F
## 18 1.724 1.857 4.980 B M
## 19 1.785 4.424 5.611 C F
## 20 2.133 3.958 3.530 A M
## 21 5.512 1.658 3.292 B M
## 22 3.140 6.972 4.354 B M
## 23 4.610 2.740 1.497 C M
## 24 2.575 2.967 5.191 A M
## 25 1.951 1.996 1.916 A M
## 26 3.282 6.299 3.116 D M
## 27 1.022 1.764 2.853 A M
## 28 1.344 4.229 2.308 A F
## 29 1.855 4.866 3.305 A F
## 30 0.886 2.795 4.485 A F
## 31 1.541 1.507 1.996 C F
## 32 5.492 1.202 3.887 B F
## 33 0.884 3.165 3.752 D F
## 34 4.371 1.234 2.591 C M
## 35 2.638 2.195 0.804 D M
## 36 2.606 3.889 2.388 A F
## 37 0.821 1.927 3.168 D M
## 38 1.889 4.294 2.137 C M
## 39 5.683 2.467 2.356 C F
## 40 1.624 2.650 3.247 A F
```

1. Use different ways to print the second column of the data frame `bsp2` (as a vector or a data frame).

```
bsp2[, 2]
bsp2$Y
bsp2[[2]]
bsp2["Y"]
```

2. Use different ways to print columns U and V.

```
bsp2[, 4]
bsp2$U
bsp2[[4]]
bsp2["U"]
bsp2[, 5]
bsp2$V
bsp2[[5]]
bsp2["V"]
```

3. Use the `attach` command to make the variables directly accessible. Print `X`. Now `detach` the data frame again.

```
attach(bsp2)
print(X)  # For safety reasons apply rm(list = ls()) upfront
detach(bsp2)
```

4. Print all rows of `bsp2` where the variable U has value A or B.

```
bsp2[bsp2$U == "A" | bsp2$U == "B", ]
```

```
##        X     Y     Z U V
## 1  2.411 2.317 5.209 B M
## 6  2.542 3.072 2.999 A M
## 8  8.947 3.058 3.178 B M
## 9  5.784 2.092 3.364 B F
## 10 2.180 2.745 3.731 B M
## 11 1.580 2.008 4.463 B M
## 14 2.509 6.982 1.265 B M
## 18 1.724 1.857 4.980 B M
## 20 2.133 3.958 3.530 A M
## 21 5.512 1.658 3.292 B M
## 22 3.140 6.972 4.354 B M
## 24 2.575 2.967 5.191 A M
## 25 1.951 1.996 1.916 A M
## 27 1.022 1.764 2.853 A M
## 28 1.344 4.229 2.308 A F
## 29 1.855 4.866 3.305 A F
## 30 0.886 2.795 4.485 A F
## 32 5.492 1.202 3.887 B F
## 36 2.606 3.889 2.388 A F
## 40 1.624 2.650 3.247 A F
```

5. Print all rows of `bsp2` where the variable X is smaller than its median and the variable Y is larger than its median.

```
bsp2[bsp2$X < median(bsp2$X) & bsp2$Y > median(bsp2$Y), ]
```

```
##        X     Y     Z U V
## 3  1.066 5.471 4.856 D M
## 7  2.200 3.272 3.174 D F
## 10 2.180 2.745 3.731 B M
## 15 1.436 4.650 2.002 C F
## 16 2.275 3.761 1.376 D F
## 19 1.785 4.424 5.611 C F
## 20 2.133 3.958 3.530 A M
```

```
## 28 1.344 4.229 2.308 A F
## 29 1.855 4.866 3.305 A F
## 30 0.886 2.795 4.485 A F
## 33 0.884 3.165 3.752 D F
## 38 1.889 4.294 2.137 C M
```

6. One can add row names to a data frame. Execute the following command and print the data frame to have a look at the new row names:

```r
row.names(bsp2) <- paste(rep(LETTERS[1:20], each = 2), rep(1:2, 20), sep = "")
print(bsp2)
```

```
##        X     Y     Z U V
## A1 2.411 2.317 5.209 B M
## A2 2.469 2.116 2.566 D M
## B1 1.066 5.471 4.856 D M
## B2 2.264 2.107 2.647 C M
## C1 2.775 3.136 3.221 C M
## C2 2.542 3.072 2.999 A M
## D1 2.200 3.272 3.174 D F
## D2 8.947 3.058 3.178 B M
## E1 5.784 2.092 3.364 B F
## E2 2.180 2.745 3.731 B M
## F1 1.580 2.008 4.463 B M
## F2 3.590 2.214 1.556 C F
## G1 4.316 1.868 0.731 D M
## G2 2.509 6.982 1.265 B M
## H1 1.436 4.650 2.002 C F
## H2 2.275 3.761 1.376 D F
## I1 3.648 1.688 1.837 C F
## I2 1.724 1.857 4.980 B M
## J1 1.785 4.424 5.611 C F
## J2 2.133 3.958 3.530 A M
## K1 5.512 1.658 3.292 B M
## K2 3.140 6.972 4.354 B M
## L1 4.610 2.740 1.497 C M
## L2 2.575 2.967 5.191 A M
## M1 1.951 1.996 1.916 A M
## M2 3.282 6.299 3.116 D M
## N1 1.022 1.764 2.853 A M
## N2 1.344 4.229 2.308 A F
## O1 1.855 4.866 3.305 A F
## O2 0.886 2.795 4.485 A F
## P1 1.541 1.507 1.996 C F
## P2 5.492 1.202 3.887 B F
## Q1 0.884 3.165 3.752 D F
## Q2 4.371 1.234 2.591 C M
## R1 2.638 2.195 0.804 D M
## R2 2.606 3.889 2.388 A F
## S1 0.821 1.927 3.168 D M
## S2 1.889 4.294 2.137 C M
## T1 5.683 2.467 2.356 C F
## T2 1.624 2.650 3.247 A F
```

7. Use the row name and the variable name to print the value of variable `Z` at observation `T1`.

```r
bsp2["T1", "Z"]
```

```
## [1] 2.356
```

8. Print the rows for observations `G1` and `G2`.

```r
bsp2[c("G1", "G2"), ]
```

```
##        X     Y     Z U V
## G1 4.316 1.868 0.731 D M
## G2 2.509 6.982 1.265 B M
```

Selection and transformation with dplyr

1. Load `dplyr` and `gapminder` package which will provide you with the **gapminder** dataset. How many observations and variables are in the dataset?

```r
library("dplyr")
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library("gapminder")
gapminder
```

```
## # A tibble: 1,704 x 6
##    country     continent  year lifeExp      pop gdpPercap
##    <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
##  1 Afghanistan Asia       1952    28.8  8425333      779.
##  2 Afghanistan Asia       1957    30.3  9240934      821.
##  3 Afghanistan Asia       1962    32.0 10267083      853.
##  4 Afghanistan Asia       1967    34.0 11537966      836.
##  5 Afghanistan Asia       1972    36.1 13079460      740.
##  6 Afghanistan Asia       1977    38.4 14880372      786.
##  7 Afghanistan Asia       1982    39.9 12881816      978.
##  8 Afghanistan Asia       1987    40.8 13867957      852.
##  9 Afghanistan Asia       1992    41.7 16317921      649.
## 10 Afghanistan Asia       1997    41.8 22227415      635.
## # ... with 1,694 more rows
```

2. The `filter` verb extracts particular observations based on a condition. Use pipes (`%>%`) to select all information of the year 2007. For how many countries is there data available?

```r
gapminder %>%
  filter(year == 2007)
```

```
## # A tibble: 142 x 6
##    country     continent  year lifeExp       pop gdpPercap
```

```
##    <fct>       <fct>     <int>   <dbl>     <int>     <dbl>
##  1 Afghanistan Asia       2007    43.8  31889923      975.
##  2 Albania     Europe     2007    76.4   3600523     5937.
##  3 Algeria     Africa     2007    72.3  33333216     6223.
##  4 Angola      Africa     2007    42.7  12420476     4797.
##  5 Argentina   Americas   2007    75.3  40301927    12779.
##  6 Australia   Oceania    2007    81.2  20434176    34435.
##  7 Austria     Europe     2007    79.8   8199783    36126.
##  8 Bahrain     Asia       2007    75.6    708573    29796.
##  9 Bangladesh  Asia       2007    64.1 150448339     1391.
## 10 Belgium     Europe     2007    79.4  10392226    33693.
## # ... with 132 more rows
```

3. Now choose all the data for the United States in the year 2007 using the `filter` command. How many observations do you get?

```r
gapminder %>%
  filter(year == 2007, country == "United States")
```

```
## # A tibble: 1 x 6
##   country       continent  year lifeExp       pop gdpPercap
##   <fct>         <fct>     <int>   <dbl>     <int>     <dbl>
## 1 United States Americas   2007    78.2 301139947    42952.
```

4. `arrange` is a useful command for sorting data frames. First, sort the data frame by gdp per capita in ascending order. Second, sort the data set by gdp per capita in descending order for the year 2007 only.

```r
gapminder %>%
  arrange(gdpPercap)
```

```
## # A tibble: 1,704 x 6
##    country          continent  year lifeExp      pop gdpPercap
##    <fct>            <fct>     <int>   <dbl>    <int>     <dbl>
##  1 Congo, Dem. Rep. Africa     2002    45.0 55379852      241.
##  2 Congo, Dem. Rep. Africa     2007    46.5 64606759      278.
##  3 Lesotho          Africa     1952    42.1   748747      299.
##  4 Guinea-Bissau    Africa     1952    32.5   580653      300.
##  5 Congo, Dem. Rep. Africa     1997    42.6 47798986      312.
##  6 Eritrea          Africa     1952    35.9  1438760      329.
##  7 Myanmar          Asia       1952    36.3 20092996      331
##  8 Lesotho          Africa     1957    45.0   813338      336.
##  9 Burundi          Africa     1952    39.0  2445618      339.
## 10 Eritrea          Africa     1957    38.0  1542611      344.
## # ... with 1,694 more rows
```

```r
gapminder %>%
  filter(year == 2007) %>%
  arrange(desc(gdpPercap))
```

```
## # A tibble: 142 x 6
##    country          continent  year lifeExp       pop gdpPercap
##    <fct>            <fct>     <int>   <dbl>     <int>     <dbl>
##  1 Norway           Europe     2007    80.2   4627926    49357.
##  2 Kuwait           Asia       2007    77.6   2505559    47307.
##  3 Singapore        Asia       2007    80.0   4553009    47143.
##  4 United States    Americas   2007    78.2 301139947    42952.
##  5 Ireland          Europe     2007    78.9   4109086    40676.
```

```
##  6 Hong Kong, China Asia       2007    82.2   6980412    39725.
##  7 Switzerland      Europe     2007    81.7   7554661    37506.
##  8 Netherlands      Europe     2007    79.8  16570613    36798.
##  9 Canada           Americas   2007    80.7  33390141    36319.
## 10 Iceland          Europe     2007    81.8    301931    36181.
## # ... with 132 more rows
```

5. `mutate` is useful whenever you want to change or add variables to your dataset. First, replace the variable `pop` (population) by dividing it by 1000000. Second, add a new variable **gdp** for total gross domestic product.

```
gapminder %>%
  mutate(pop = pop/1000000)    #change variable
```

```
## # A tibble: 1,704 x 6
##    country     continent  year lifeExp   pop gdpPercap
##    <fct>       <fct>     <int>   <dbl> <dbl>     <dbl>
##  1 Afghanistan Asia       1952    28.8  8.43      779.
##  2 Afghanistan Asia       1957    30.3  9.24      821.
##  3 Afghanistan Asia       1962    32.0 10.3       853.
##  4 Afghanistan Asia       1967    34.0 11.5       836.
##  5 Afghanistan Asia       1972    36.1 13.1       740.
##  6 Afghanistan Asia       1977    38.4 14.9       786.
##  7 Afghanistan Asia       1982    39.9 12.9       978.
##  8 Afghanistan Asia       1987    40.8 13.9       852.
##  9 Afghanistan Asia       1992    41.7 16.3       649.
## 10 Afghanistan Asia       1997    41.8 22.2       635.
## # ... with 1,694 more rows
```

```
gapminder %>%
  mutate(gdp = gdpPercap*pop) #add new variable total gdp
```

```
## # A tibble: 1,704 x 7
##    country     continent  year lifeExp      pop gdpPercap          gdp
##    <fct>       <fct>     <int>   <dbl>    <int>     <dbl>        <dbl>
##  1 Afghanistan Asia       1952    28.8  8425333      779.  6567086330.
##  2 Afghanistan Asia       1957    30.3  9240934      821.  7585448670.
##  3 Afghanistan Asia       1962    32.0 10267083      853.  8758855797.
##  4 Afghanistan Asia       1967    34.0 11537966      836.  9648014150.
##  5 Afghanistan Asia       1972    36.1 13079460      740.  9678553274.
##  6 Afghanistan Asia       1977    38.4 14880372      786. 11697659231.
##  7 Afghanistan Asia       1982    39.9 12881816      978. 12598563401.
##  8 Afghanistan Asia       1987    40.8 13867957      852. 11820990309.
##  9 Afghanistan Asia       1992    41.7 16317921      649. 10595901589.
## 10 Afghanistan Asia       1997    41.8 22227415      635. 14121995875.
## # ... with 1,694 more rows
```

6. Which countries have the highest gdp in 2007?

```
gapminder %>%
  mutate(gdp = gdpPercap*pop) %>%
  filter(year == 2007) %>%
  arrange(desc(gdp))
```

```
## # A tibble: 142 x 7
##    country        continent  year lifeExp       pop gdpPercap     gdp
##    <fct>          <fct>     <int>   <dbl>     <int>     <dbl>   <dbl>
```

```
##   1 United States  Americas   2007    78.2  301139947    42952. 1.29e13
##   2 China          Asia       2007    73.0 1318683096     4959. 6.54e12
##   3 Japan          Asia       2007    82.6  127467972    31656. 4.04e12
##   4 India          Asia       2007    64.7 1110396331     2452. 2.72e12
##   5 Germany        Europe     2007    79.4   82400996    32170. 2.65e12
##   6 United Kingdom Europe     2007    79.4   60776238    33203. 2.02e12
##   7 France         Europe     2007    80.7   61083916    30470. 1.86e12
##   8 Brazil         Americas   2007    72.4  190010647     9066. 1.72e12
##   9 Italy          Europe     2007    80.5   58147733    28570. 1.66e12
##  10 Mexico         Americas   2007    76.2  108700891    11978. 1.30e12
## # ... with 132 more rows
```

7. The basic use of the `summarize` verb is to turn many rows into one. Use it to output the mean and median of `lifeExp` as well as the total population in 2007 into a new data frame using pipes.

```
gapminder %>%
  filter(year==2007) %>%
  summarize(meanLifeExp = mean(lifeExp), medianLifeExp = median(lifeExp), totalPop = sum(as.numeric(pop
```

```
## # A tibble: 1 x 3
##   meanLifeExp medianLifeExp   totalPop
##         <dbl>         <dbl>      <dbl>
## 1        67.0          71.9 6251013179
```

8. Now do the same, but for all years, using `group_by(year)`.

```
gapminder %>%
  group_by(year) %>%
  summarize(meanLifeExp = mean(lifeExp), medianLifeExp = median(lifeExp), totalPop = sum(as.numeric(pop
```

```
## # A tibble: 12 x 4
##     year meanLifeExp medianLifeExp   totalPop
##    <int>       <dbl>         <dbl>      <dbl>
##  1  1952        49.1          45.1 2406957150
##  2  1957        51.5          48.4 2664404580
##  3  1962        53.6          50.9 2899782974
##  4  1967        55.7          53.8 3217478384
##  5  1972        57.6          56.5 3576977158
##  6  1977        59.6          59.7 3930045807
##  7  1982        61.5          62.4 4289436840
##  8  1987        63.2          65.8 4691477418
##  9  1992        64.2          67.7 5110710260
## 10  1997        65.0          69.4 5515204472
## 11  2002        65.7          70.8 5886977579
## 12  2007        67.0          71.9 6251013179
```

9. Again get the same statistics, but this time by continent for the year 2007.

```
gapminder %>%
  filter(year==2007) %>%
  group_by(continent) %>%
  summarize(meanLifeExp = mean(lifeExp), medianLifeExp = median(lifeExp), totalPop = sum(as.numeric(pop
```

```
## # A tibble: 5 x 4
##   continent meanLifeExp medianLifeExp   totalPop
##   <fct>           <dbl>         <dbl>      <dbl>
## 1 Africa           54.8          52.9  929539692
## 2 Americas         73.6          72.9  898871184
```

```
## 3 Asia             70.7          72.4 3811953827
## 4 Europe           77.6          78.6  586098529
## 5 Oceania          80.7          80.7   24549947
```

10. Lastly, get the same statistics by continent and year

```
gapminder %>%
  group_by(year,continent) %>%
  summarize(meanLifeExp = mean(lifeExp), medianLifeExp = median(lifeExp), totalPop = sum(as.numeric(pop
```

```
## # A tibble: 60 x 5
## # Groups:   year [12]
##     year continent meanLifeExp medianLifeExp   totalPop
##    <int> <fct>           <dbl>         <dbl>      <dbl>
##  1  1952 Africa           39.1          38.8  237640501
##  2  1952 Americas         53.3          54.7  345152446
##  3  1952 Asia             46.3          44.9 1395357351
##  4  1952 Europe           64.4          65.9  418120846
##  5  1952 Oceania          69.3          69.3   10686006
##  6  1957 Africa           41.3          40.6  264837738
##  7  1957 Americas         56.0          56.1  386953916
##  8  1957 Asia             49.3          48.3 1562780599
##  9  1957 Europe           66.7          67.6  437890351
## 10  1957 Oceania          70.3          70.3   11941976
## # ... with 50 more rows
```

Graphics with ggplot

1. Load `dplyr`, `ggplot2` and `gapminder` package which will provide you with the **gapminder** dataset. How many observations and variables are in the dataset?

```
library("dplyr")
library("ggplot2")
library("gapminder")
```

2. Save all data from 2007 into the data frame **gapminder_2007**.

```
gapminder_2007 <- gapminder %>%
  filter(year == 2007)
```

3. Visualize countries wealth (`gdpPercap` on x axis) against life expectancy (`lifeExp` on y axis) using the good old `plot` command. Compare this to the way ggplot draws a scatterplot when using `geom_point`.

```
plot(gapminder_2007$gdpPercap,gapminder_2007$lifeExp)
```

IntroRExercisesWithSolutions_files/figure-latex/unnamed-chunk-108-1.pdf

```
ggplot(gapminder_2007, aes(x = gdpPercap, y = lifeExp)) +
  geom_point()
```

IntroRExercisesWithSolutions_files/figure-latex/unnamed-chunk-108-2.pdf

4. Now add a log scale `scale_x_log10` for gdpPercap to your ggplot scatterplot.

```
ggplot(gapminder_2007, aes(x = gdpPercap, y = lifeExp)) +
  geom_point() +
  scale_x_log10() # # each unit on the x-axis represents a change of 10 times the gdp
```

IntroRExercisesWithSolutions_files/figure-latex/unnamed-chunk-109-1.pdf

5. Create a scatter plot comparing `pop` and `gdpPercap` for the year 2007, with both axes on a log scale.

```
ggplot(gapminder_2007, aes(x = pop, y = gdpPercap)) +
  geom_point() +
  scale_x_log10() +
  scale_y_log10()
```

IntroRExercisesWithSolutions_files/figure-latex/unnamed-chunk-110-1.pdf

6. To add more variables to a 2-dimensional plot, we can use two more asthetics: color for a categorial variable and size for numerical variables. Add the continent and pop to the scatterplot of `gdpPercap` and `lifeExp` for the year 2007

```
ggplot(gapminder_2007, aes(x = gdpPercap, y = lifeExp, color = continent, size = pop)) +
  geom_point() +
  scale_x_log10()
```

IntroRExercisesWithSolutions_files/figure-latex/unnamed-chunk-111-1.pdf

7. Now we want to compare the dynamic relationship between `gdpPercap` and `lifeExp` for all years. Use `facet_wrap(~ year)` on the original dataset **gapminder** to add a facet to your scatterplot.

```
ggplot(gapminder, aes(x = gdpPercap, y = lifeExp, color = continent, size = pop)) +
  geom_point() +
  scale_x_log10() +
  facet_wrap(~ year)
```

IntroRExercisesWithSolutions_files/figure-latex/unnamed-chunk-112-1.pdf

8. Create a data frame from the **gapminder** dataset with summarized data with the mean of `lifeExp` and total population, both grouped by year. Visualize this summarized data using `ggplot` and let your y axis begin at 0.

```
by_year <- gapminder %>%
  group_by(year) %>%
  summarize(meanLifeExp = mean(lifeExp), totalPop = sum(as.numeric(pop)))
ggplot(by_year, aes(x = year, y = totalPop)) +
  geom_point() +
  expand_limits(y=0) # start y axis at 0
```

IntroRExercisesWithSolutions_files/figure-latex/unnamed-chunk-113-1.pdf

9. Create a data frame from the **gapminder** dataset with summarized data with the mean of `lifeExp` and total population, both grouped by year and continent. Visualize this summarized data using `ggplot` and let your y axis begin at 0.

```
by_year_continent <- gapminder %>%
  group_by(year,continent) %>%
  summarize(meanLifeExp = mean(lifeExp), totalPop = sum(as.numeric(pop)))
ggplot(by_year_continent, aes(x = year, y = meanLifeExp, color = continent)) +
  geom_point() +
  expand_limits(y=0) # start y axis at 0
```

IntroRExercisesWithSolutions_files/figure-latex/unnamed-chunk-114-1.pdf

10. Create a line plot (`geom_line`) with `ggplot` for the just created subset of data grouped by year and continent. Put year on the x axis, meanLifeExp on the y axis and let the color indicate the continents.

```
ggplot(by_year_continent, aes(x = year, y = meanLifeExp, color = continent)) +
  geom_line() +
  expand_limits(y=0) # start y axis at 0
```

IntroRExercisesWithSolutions_files/figure-latex/unnamed-chunk-115-1.pdf

11. Filter the original **gapminder** data for the year 2007, group by continent and summarize the mean life

expectancy for this data frame. Create a bar plot (`geom_col`) with ggplot.

```
by_continent <- gapminder %>%
  filter(year == 2007) %>%
  group_by(continent) %>%
  summarize(meanLifeExp = mean(lifeExp))
#x is categorial variable
ggplot(by_continent, aes(x = continent, y = meanLifeExp, color = continent)) +
  geom_col()
```

IntroRExercisesWithSolutions_files/figure-latex/unnamed-chunk-116-1.pdf

12. Create a histogram (`geom_histogram`) of population (pop) in 2007 using a log scale.

```
gapminder_2007 <- gapminder %>%
  filter(year == 2007)

ggplot(gapminder_2007, aes(x = pop)) +
  geom_histogram() +
  scale_x_log10()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

IntroRExercisesWithSolutions_files/figure-latex/unnamed-chunk-117-1.pdf

13. Create a boxplot (`geom_bloxplot`) comparing gdpPercap among continents for the year 2007 and add a title to the graph using `ggtitle`.

```
gapminder_2007 <- gapminder %>%
  filter(year == 2007)
ggplot(gapminder_2007, aes(x = continent, y = gdpPercap)) +
  geom_boxplot() +
  scale_y_log10() +
  ggtitle("Comparing GDP per capita across continents (log-scale)")
```

IntroRExercisesWithSolutions_files/figure-latex/unnamed-chunk-118-1.pdf

---

Histograms

In this section, please always use the command `truehist` (which is included in the `MASS` package) to generate histograms.

```
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

1. Load the file **gemeinden2006.csv** into a data frame. Delete all observations where the number of inhabitants (`Einwohner`) is smaller than 5. Plot the histogram of the logarithm of the variable ‘Einwohner.

```
gemeinden2006 <- read.csv2("data/gemeinden2006.csv")
gemeinden2006new <- gemeinden2006[gemeinden2006$Einwohner >= 5, ] #or x[!x$Einwohner < 5, ]
truehist(log(gemeinden2006new$Einwohner), col = "lightblue")
```

IntroRExercisesWithSolutions_files/figure-latex/unnamed-chunk-121-1.pdf

2.Add the density function of a fitted normal distribution to the histogram.

```
truehist(log(gemeinden2006new$Einwohner), col = "lightblue")
m <- mean(log(gemeinden2006new$Einwohner))
s <- sd(log(gemeinden2006new$Einwohner))
x <- seq(1, 15, length = 500)
lines(x, dnorm(x, mean = m, sd = s), lwd = 2)
```

IntroRExercisesWithSolutions_files/figure-latex/unnamed-chunk-122-1.pdf

3. Load the Stata file **mikrozensus2002cf.dta** into a data frame. Consider the variable `ef462` (rent in April 2002). Drop all observations where the rent exceeds 2000 Euro. Plot the histogram.

```
library(foreign)
mikrozensus2002cf <- read.dta("data/mikrozensus2002cf.dta")
y <- mikrozensus2002cf$ef462[!is.na(mikrozensus2002cf$ef462)]
yy <- y[y <= 2000]
truehist(yy, col = "pink", xlab = "rent", ylab = "density", main = "histogram of rents")
```

IntroRExercisesWithSolutions_files/figure-latex/unnamed-chunk-123-1.pdf

4. Load the Stata file **mikrozensus2002cf.dta** into a data frame.

- Plot the histogram of the variable `ef453` (size of flat in square meters).
- Drop all observations with more than 300 m^2 and plot the histogram again.
- Set the number of bins in the histogram to 15.

```
truehist(mikrozensus2002cf$ef453, col = "lightblue", xlab = "size of flat (in qm)", ylab = "density")
```

IntroRExercisesWithSolutions_files/figure-latex/unnamed-chunk-124-1.pdf

```
truehist(mikrozensus2002cf$ef453[mikrozensus2002cf$ef453 <= 300], col = "lightgreen", xlab = "size of fl
```

IntroRExercisesWithSolutions_files/figure-latex/unnamed-chunk-124-2.pdf

```
truehist(mikrozensus2002cf$ef453[mikrozensus2002cf$ef453 <= 300], col = "steelblue", nbins = 15, xlab =
```

IntroRExercisesWithSolutions_files/figure-latex/unnamed-chunk-124-3.pdf

---

Correlation and covariance

1. Execute `data(Titanic)` to load the object `Titanic` of class `table`. Print it as an ordinary table and as a flat table. Plot it as well. Compute the univariate marginal distributions using the `apply` command. Compute the bivariate marginal distribution of survival and social class (again using `apply`).

```
data(Titanic)
Titanic
```

```
## , , Age = Child, Survived = No
##
##       Sex
## Class  Male Female
```

```
##   1st     0      0
##   2nd     0      0
##   3rd    35     17
##   Crew    0      0
##
## , , Age = Adult, Survived = No
##
##       Sex
## Class  Male Female
##   1st   118      4
##   2nd   154     13
##   3rd   387     89
##   Crew  670      3
##
## , , Age = Child, Survived = Yes
##
##       Sex
## Class  Male Female
##   1st     5      1
##   2nd    11     13
##   3rd    13     14
##   Crew    0      0
##
## , , Age = Adult, Survived = Yes
##
##       Sex
## Class  Male Female
##   1st    57    140
##   2nd    14     80
##   3rd    75     76
##   Crew  192     20
```

```
table(Titanic)
```

```
## Titanic
##   0   1   3   4   5  11  13  14  17  20  35  57  75  76  80  89 118 140
##   8   1   1   1   1   1   3   2   1   1   1   1   1   1   1   1   1   1
## 154 192 387 670
##   1   1   1   1
```

```
ftable(Titanic)
```

```
##                    Survived  No Yes
## Class Sex    Age
## 1st   Male   Child            0   5
##              Adult          118  57
##       Female Child            0   1
##              Adult            4 140
## 2nd   Male   Child            0  11
##              Adult          154  14
##       Female Child            0  13
##              Adult           13  80
## 3rd   Male   Child           35  13
##              Adult          387  75
##       Female Child           17  14
##              Adult           89  76
```

```
## Crew  Male   Child            0   0
##             Adult          670 192
##      Female Child            0   0
##             Adult            3  20
```

```r
ftable(Titanic, row.vars = c("Survived", "Age"))  # Write 'Survived' and 'Age' into rows
```

```
##                Class  1st         2nd         3rd         Crew
##                Sex    Male Female Male Female Male Female Male Female
## Survived Age
## No       Child           0      0    0      0   35     17    0      0
##          Adult         118      4  154     13  387     89  670      3
## Yes      Child           5      1   11     13   13     14    0      0
##          Adult          57    140   14     80   75     76  192     20
```

```r
plot(Titanic)
```

IntroRExercisesWithSolutions_files/figure-latex/unnamed-chunk-126-1.pdf

```r
apply(Titanic, 1, sum)
```

```
##  1st  2nd  3rd Crew
##  325  285  706  885
```

```r
apply(Titanic, 2, sum)
```

```
##   Male Female
##   1731    470
```

```r
apply(Titanic, 3, sum)
```

```
## Child Adult
##   109  2092
```

```r
apply(Titanic, 4, sum)
```

```
##   No  Yes
## 1490  711
```

```r
apply(Titanic, c(1, 4), sum)
```

```
##       Survived
## Class   No Yes
##   1st  122 203
##   2nd  167 118
##   3rd  528 178
##   Crew 673 212
```

```r
# or margin.table(Titanic,c(1,4))
```

2. Load the file **covmat.csv** into a data frame.

```r
covmat <- read.csv("data/covmat.csv")
```

- Compute the covariance matrix using the option `use="complete"`. Check if the covariance matrix is positive definite.

```r
cov(covmat)
```

```
##          V1 V2 V3
## V1 1.170296 NA NA
## V2       NA NA NA
## V3       NA NA NA
```

```r
cov(covmat,use="complete")
```

```
##             V1          V2         V3
## V1  0.35373543 -0.09918577 -0.1629030
## V2 -0.09918577  0.34900069  0.4293324
## V3 -0.16290303  0.42933237  0.7174221
```

```r
if (sum(eigen(cov(covmat,use="complete"))$value>0) == dim(covmat)[2]){
  print("Matrix is positive definite")
} else {
  print("Matrix is not positive definite")
}
```

```
## [1] "Matrix is positive definite"
```

- Now compute the covariance using the option `pairwise` and check again, if the covariance matrix is positive definite.

```r
cov(covmat,use="pairwise")
```

```
##            V1         V2        V3
## V1  1.1702962 -0.1119788 0.1337978
## V2 -0.1119788  0.3051700 0.4293324
## V3  0.1337978  0.4293324 0.5038195
```

```r
if (sum(eigen(cov(covmat,use="pairwise"))$value>0) == dim(covmat)[2]){
  print("Matrix is positive definite")
} else {
  print("Matrix is not positive definite")
}
```

```
## [1] "Matrix is not positive definite"
```

Cleaning data

We will consider historical weather data for Boston, USA for 12 months beginning in December 2014.

1. Load the packages `tidyr`, `dplyr`, `lubridate`, and `stringr`. Import the data using `weather <- readRDS("weather.rds")` and have a look how **dirty** it is.

```r
library("tidyr"); library("dplyr"); library("lubridate"); library("stringr")
```

```
##
## Attaching package: 'lubridate'

## The following object is masked from 'package:base':
##
##     date
```

```r
detach("package:MASS")
weather <- readRDS("data/weather.rds")
```

2. The first step is to understand the structure of your data with `class`, `dim`, `names`, `str`, `glimpse`, and `summary`. Also preview the first and last 15 observations with `head` and `tail`.

```r
class(weather) # Verify that weather is a data.frame
```

```
## [1] "data.frame"
```

```r
dim(weather) # Check the dimensions
```

```
## [1] 286  35
```

```r
names(weather) # View the column names
```

```
##  [1] "X"       "year"    "month"   "measure" "X1"      "X2"      "X3"
##  [8] "X4"      "X5"      "X6"      "X7"      "X8"      "X9"      "X10"
## [15] "X11"     "X12"     "X13"     "X14"     "X15"     "X16"     "X17"
## [22] "X18"     "X19"     "X20"     "X21"     "X22"     "X23"     "X24"
## [29] "X25"     "X26"     "X27"     "X28"     "X29"     "X30"     "X31"
```

```r
str(weather) # View the structure of the data
```

```
## 'data.frame':    286 obs. of  35 variables:
##  $ X      : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ year   : int  2014 2014 2014 2014 2014 2014 2014 2014 2014 2014 ...
##  $ month  : int  12 12 12 12 12 12 12 12 12 12 ...
##  $ measure: chr  "Max.TemperatureF" "Mean.TemperatureF" "Min.TemperatureF" "Max.Dew.PointF" ...
##  $ X1     : chr  "64" "52" "39" "46" ...
##  $ X2     : chr  "42" "38" "33" "40" ...
##  $ X3     : chr  "51" "44" "37" "49" ...
##  $ X4     : chr  "43" "37" "30" "24" ...
##  $ X5     : chr  "42" "34" "26" "37" ...
##  $ X6     : chr  "45" "42" "38" "45" ...
##  $ X7     : chr  "38" "30" "21" "36" ...
##  $ X8     : chr  "29" "24" "18" "28" ...
##  $ X9     : chr  "49" "39" "29" "49" ...
##  $ X10    : chr  "48" "43" "38" "45" ...
##  $ X11    : chr  "39" "36" "32" "37" ...
##  $ X12    : chr  "39" "35" "31" "28" ...
##  $ X13    : chr  "42" "37" "32" "28" ...
##  $ X14    : chr  "45" "39" "33" "29" ...
##  $ X15    : chr  "42" "37" "32" "33" ...
##  $ X16    : chr  "44" "40" "35" "42" ...
##  $ X17    : chr  "49" "45" "41" "46" ...
##  $ X18    : chr  "44" "40" "36" "34" ...
##  $ X19    : chr  "37" "33" "29" "25" ...
##  $ X20    : chr  "36" "32" "27" "30" ...
##  $ X21    : chr  "36" "33" "30" "30" ...
##  $ X22    : chr  "44" "39" "33" "39" ...
##  $ X23    : chr  "47" "45" "42" "45" ...
##  $ X24    : chr  "46" "44" "41" "46" ...
##  $ X25    : chr  "59" "52" "44" "58" ...
##  $ X26    : chr  "50" "44" "37" "31" ...
##  $ X27    : chr  "52" "45" "38" "34" ...
##  $ X28    : chr  "52" "46" "40" "42" ...
```

```
##  $ X29    : chr  "41" "36" "30" "26" ...
##  $ X30    : chr  "30" "26" "22" "10" ...
##  $ X31    : chr  "30" "25" "20" "8" ...
```

```r
glimpse(weather) # Look at the structure using dplyr's glimpse()
```

```
## Observations: 286
## Variables: 35
## $ X       <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,...
## $ year    <int> 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, ...
## $ month   <int> 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12...
## $ measure <chr> "Max.TemperatureF", "Mean.TemperatureF", "Min.Temperat...
## $ X1      <chr> "64", "52", "39", "46", "40", "26", "74", "63", "52", ...
## $ X2      <chr> "42", "38", "33", "40", "27", "17", "92", "72", "51", ...
## $ X3      <chr> "51", "44", "37", "49", "42", "24", "100", "79", "57",...
## $ X4      <chr> "43", "37", "30", "24", "21", "13", "69", "54", "39", ...
## $ X5      <chr> "42", "34", "26", "37", "25", "12", "85", "66", "47", ...
## $ X6      <chr> "45", "42", "38", "45", "40", "36", "100", "93", "85",...
## $ X7      <chr> "38", "30", "21", "36", "20", "-3", "92", "61", "29", ...
## $ X8      <chr> "29", "24", "18", "28", "16", "3", "92", "70", "47", "...
## $ X9      <chr> "49", "39", "29", "49", "41", "28", "100", "93", "86",...
## $ X10     <chr> "48", "43", "38", "45", "39", "37", "100", "95", "89",...
## $ X11     <chr> "39", "36", "32", "37", "31", "27", "92", "87", "82", ...
## $ X12     <chr> "39", "35", "31", "28", "27", "25", "85", "75", "64", ...
## $ X13     <chr> "42", "37", "32", "28", "26", "24", "75", "65", "55", ...
## $ X14     <chr> "45", "39", "33", "29", "27", "25", "82", "68", "53", ...
## $ X15     <chr> "42", "37", "32", "33", "29", "27", "89", "75", "60", ...
## $ X16     <chr> "44", "40", "35", "42", "36", "30", "96", "85", "73", ...
## $ X17     <chr> "49", "45", "41", "46", "41", "32", "100", "85", "70",...
## $ X18     <chr> "44", "40", "36", "34", "30", "26", "89", "73", "57", ...
## $ X19     <chr> "37", "33", "29", "25", "22", "20", "69", "63", "56", ...
## $ X20     <chr> "36", "32", "27", "30", "24", "20", "89", "79", "69", ...
## $ X21     <chr> "36", "33", "30", "30", "27", "25", "85", "77", "69", ...
## $ X22     <chr> "44", "39", "33", "39", "34", "25", "89", "79", "69", ...
## $ X23     <chr> "47", "45", "42", "45", "42", "37", "100", "91", "82",...
## $ X24     <chr> "46", "44", "41", "46", "44", "41", "100", "98", "96",...
## $ X25     <chr> "59", "52", "44", "58", "43", "29", "100", "75", "49",...
## $ X26     <chr> "50", "44", "37", "31", "29", "28", "70", "60", "49", ...
## $ X27     <chr> "52", "45", "38", "34", "31", "29", "70", "60", "50", ...
## $ X28     <chr> "52", "46", "40", "42", "35", "27", "76", "65", "53", ...
## $ X29     <chr> "41", "36", "30", "26", "20", "10", "64", "51", "37", ...
## $ X30     <chr> "30", "26", "22", "10", "4", "-6", "50", "38", "26", "...
## $ X31     <chr> "30", "25", "20", "8", "5", "1", "57", "44", "31", "30...
```

```r
summary(weather) # View a summary of the data
```

```
##        X               year          month          measure         
##  Min.   :  1.00   Min.   :2014   Min.   : 1.000   Length:286        
##  1st Qu.: 72.25   1st Qu.:2015   1st Qu.: 4.000   Class :character  
##  Median :143.50   Median :2015   Median : 7.000   Mode  :character  
##  Mean   :143.50   Mean   :2015   Mean   : 6.923                     
##  3rd Qu.:214.75   3rd Qu.:2015   3rd Qu.:10.000                     
##  Max.   :286.00   Max.   :2015   Max.   :12.000                     
##       X1                 X2                 X3           
##  Length:286         Length:286         Length:286        
```

```
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
##       X4                 X5                 X6
##  Length:286         Length:286         Length:286
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
##       X7                 X8                 X9
##  Length:286         Length:286         Length:286
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
##      X10                X11                X12
##  Length:286         Length:286         Length:286
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
##      X13                X14                X15
##  Length:286         Length:286         Length:286
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
##      X16                X17                X18
##  Length:286         Length:286         Length:286
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
##      X19                X20                X21
##  Length:286         Length:286         Length:286
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
##      X22                X23                X24
##  Length:286         Length:286         Length:286
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
```

```
##      X25                 X26                 X27
##  Length:286          Length:286          Length:286
##  Class :character    Class :character    Class :character
##  Mode  :character    Mode  :character    Mode  :character
##
##
##
##      X28                 X29                 X30
##  Length:286          Length:286          Length:286
##  Class :character    Class :character    Class :character
##  Mode  :character    Mode  :character    Mode  :character
##
##
##
##      X31
##  Length:286
##  Class :character
##  Mode  :character
##
##
##
```

```
head(weather,15)
```

```
##     X year month                  measure    X1    X2    X3    X4    X5
## 1   1 2014    12           Max.TemperatureF    64    42    51    43    42
## 2   2 2014    12          Mean.TemperatureF    52    38    44    37    34
## 3   3 2014    12           Min.TemperatureF    39    33    37    30    26
## 4   4 2014    12             Max.Dew.PointF    46    40    49    24    37
## 5   5 2014    12             MeanDew.PointF    40    27    42    21    25
## 6   6 2014    12              Min.DewpointF    26    17    24    13    12
## 7   7 2014    12               Max.Humidity    74    92   100    69    85
## 8   8 2014    12              Mean.Humidity    63    72    79    54    66
## 9   9 2014    12               Min.Humidity    52    51    57    39    47
## 10 10 2014    12  Max.Sea.Level.PressureIn 30.45 30.71  30.4 30.56 30.68
## 11 11 2014    12 Mean.Sea.Level.PressureIn 30.13 30.59 30.07 30.33 30.59
## 12 12 2014    12  Min.Sea.Level.PressureIn 30.01  30.4 29.87 30.09 30.45
## 13 13 2014    12        Max.VisibilityMiles    10    10    10    10    10
## 14 14 2014    12       Mean.VisibilityMiles    10     8     5    10    10
## 15 15 2014    12        Min.VisibilityMiles    10     2     1    10     5
##        X6    X7    X8    X9   X10   X11   X12   X13   X14   X15   X16   X17
## 1      45    38    29    49    48    39    39    42    45    42    44    49
## 2      42    30    24    39    43    36    35    37    39    37    40    45
## 3      38    21    18    29    38    32    31    32    33    32    35    41
## 4      45    36    28    49    45    37    28    28    29    33    42    46
## 5      40    20    16    41    39    31    27    26    27    29    36    41
## 6      36    -3     3    28    37    27    25    24    25    27    30    32
## 7     100    92    92   100   100    92    85    75    82    89    96   100
## 8      93    61    70    93    95    87    75    65    68    75    85    85
## 9      85    29    47    86    89    82    64    55    53    60    73    70
## 10 30.42 30.69 30.77 30.51 29.58 29.81 29.88 29.86 29.91 30.15 30.17 29.91
## 11 30.24 30.46 30.67 30.04  29.5 29.61 29.85 29.82 29.83 30.05 30.09 29.75
## 12 30.16 30.24 30.51 29.49 29.43 29.44 29.81 29.78 29.78 29.91 29.92 29.69
## 13     10    10    10    10    10    10    10    10    10    10    10    10
## 14      4    10     8     2     3     7    10    10    10    10     9     6
```

```
## 15     0     5     2     1     1     1     7    10    10    10     5     1
##      X18   X19   X20   X21   X22   X23   X24   X25   X26   X27   X28   X29
## 1     44    37    36    36    44    47    46    59    50    52    52    41
## 2     40    33    32    33    39    45    44    52    44    45    46    36
## 3     36    29    27    30    33    42    41    44    37    38    40    30
## 4     34    25    30    30    39    45    46    58    31    34    42    26
## 5     30    22    24    27    34    42    44    43    29    31    35    20
## 6     26    20    20    25    25    37    41    29    28    29    27    10
## 7     89    69    89    85    89   100   100   100    70    70    76    64
## 8     73    63    79    77    79    91    98    75    60    60    65    51
## 9     57    56    69    69    69    82    96    49    49    50    53    37
## 10 29.87 30.15 30.31 30.37  30.4 30.31 30.13 29.96 30.16 30.22 29.99 30.22
## 11 29.78 29.98 30.26 30.32 30.35 30.23  29.9 29.63 30.11 30.14 29.87 30.12
## 12 29.71 29.86 30.17 30.28  30.3 30.16 29.55 29.47 29.99 30.03 29.77    30
## 13    10    10    10    10    10    10     2    10    10    10    10    10
## 14    10    10    10     9    10     5     1     8    10    10    10    10
## 15    10    10     7     6     4     1     0     1    10    10    10    10
##      X30   X31
## 1     30    30
## 2     26    25
## 3     22    20
## 4     10     8
## 5      4     5
## 6     -6     1
## 7     50    57
## 8     38    44
## 9     26    31
## 10 30.36 30.32
## 11 30.32 30.25
## 12 30.23 30.13
## 13    10    10
## 14    10    10
## 15    10    10
```

```
tail(weather,15)
```

```
##       X year month                   measure    X1   X2   X3   X4   X5
## 272 272 2015    12             Mean.Humidity    83 <NA> <NA> <NA> <NA>
## 273 273 2015    12              Min.Humidity    69 <NA> <NA> <NA> <NA>
## 274 274 2015    12  Max.Sea.Level.PressureIn  30.4 <NA> <NA> <NA> <NA>
## 275 275 2015    12 Mean.Sea.Level.PressureIn 30.24 <NA> <NA> <NA> <NA>
## 276 276 2015    12  Min.Sea.Level.PressureIn 30.01 <NA> <NA> <NA> <NA>
## 277 277 2015    12       Max.VisibilityMiles    10 <NA> <NA> <NA> <NA>
## 278 278 2015    12      Mean.VisibilityMiles     8 <NA> <NA> <NA> <NA>
## 279 279 2015    12       Min.VisibilityMiles     1 <NA> <NA> <NA> <NA>
## 280 280 2015    12         Max.Wind.SpeedMPH    15 <NA> <NA> <NA> <NA>
## 281 281 2015    12        Mean.Wind.SpeedMPH     6 <NA> <NA> <NA> <NA>
## 282 282 2015    12         Max.Gust.SpeedMPH    17 <NA> <NA> <NA> <NA>
## 283 283 2015    12           PrecipitationIn  0.14 <NA> <NA> <NA> <NA>
## 284 284 2015    12                CloudCover     7 <NA> <NA> <NA> <NA>
## 285 285 2015    12                    Events  Rain <NA> <NA> <NA> <NA>
## 286 286 2015    12            WindDirDegrees   109 <NA> <NA> <NA> <NA>
##       X6   X7   X8   X9  X10  X11  X12  X13  X14  X15  X16  X17  X18  X19
## 272 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 273 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
```

```
## 274 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 275 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 276 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 277 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 278 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 279 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 280 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 281 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 282 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 283 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 284 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 285 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 286 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
##      X20  X21  X22  X23  X24  X25  X26  X27  X28  X29  X30  X31
## 272 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 273 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 274 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 275 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 276 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 277 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 278 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 279 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 280 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 281 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 282 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 283 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 284 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 285 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 286 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
```

3. The **weather** dataset suffers from one of the five most common symptoms of messy data: column names are values. In particular, the column names `X1-X31` represent days of the month, which should really be values of a new variable called `day`. The `tidyr` package provides the `gather()` function for exactly this scenario. Notice that `gather()` allows you to select multiple columns to be gathered by using the `:` operator. Call `gather()` on the weather data to gather columns `X1-X31`. The two columns created as a result should be called `day` and `value`. Save the result as weather2 and view it with `head`.

```
weather2 <- gather(weather, day, value, X1:X31, na.rm = TRUE)
head(weather2)
```

```
##   X year month           measure day value
## 1 1 2014    12  Max.TemperatureF  X1    64
## 2 2 2014    12 Mean.TemperatureF  X1    52
## 3 3 2014    12  Min.TemperatureF  X1    39
## 4 4 2014    12    Max.Dew.PointF  X1    46
## 5 5 2014    12    MeanDew.PointF  X1    40
## 6 6 2014    12     Min.DewpointF  X1    26
```

4. Our data suffer from a second common symptom of messy data: values are variable names. Specifically, values in the measure column should be variables (i.e. column names) in our dataset. The `spread()` function from `tidyr` is designed to help with this. Remove the first column of weather2, assigning to `without_x`. Spread the measure column of `without_x` and save the result to weather3. View the result with `head()`.

```
without_x <- weather2[, -1] # First remove column of row names
weather3 <- spread(without_x, measure, value) # Spread the data
```

```r
head(weather3)
```

```
##   year month day CloudCover    Events Max.Dew.PointF Max.Gust.SpeedMPH
## 1 2014    12  X1          6      Rain             46                29
## 2 2014    12 X10          8      Rain             45                29
## 3 2014    12 X11          8 Rain-Snow             37                28
## 4 2014    12 X12          7      Snow             28                21
## 5 2014    12 X13          5                       28                23
## 6 2014    12 X14          4                       29                20
##   Max.Humidity Max.Sea.Level.PressureIn Max.TemperatureF
## 1           74                    30.45               64
## 2          100                    29.58               48
## 3           92                    29.81               39
## 4           85                    29.88               39
## 5           75                    29.86               42
## 6           82                    29.91               45
##   Max.VisibilityMiles Max.Wind.SpeedMPH Mean.Humidity
## 1                  10                22            63
## 2                  10                23            95
## 3                  10                21            87
## 4                  10                16            75
## 5                  10                17            65
## 6                  10                15            68
##   Mean.Sea.Level.PressureIn Mean.TemperatureF Mean.VisibilityMiles
## 1                     30.13                52                   10
## 2                      29.5                43                    3
## 3                     29.61                36                    7
## 4                     29.85                35                   10
## 5                     29.82                37                   10
## 6                     29.83                39                   10
##   Mean.Wind.SpeedMPH MeanDew.PointF Min.DewpointF Min.Humidity
## 1                 13             40            26           52
## 2                 13             39            37           89
## 3                 13             31            27           82
## 4                 11             27            25           64
## 5                 12             26            24           55
## 6                 10             27            25           53
##   Min.Sea.Level.PressureIn Min.TemperatureF Min.VisibilityMiles
## 1                    30.01               39                  10
## 2                    29.43               38                   1
## 3                    29.44               32                   1
## 4                    29.81               31                   7
## 5                    29.78               32                  10
## 6                    29.78               33                  10
##   PrecipitationIn WindDirDegrees
## 1            0.01            268
## 2            0.28            357
## 3            0.02            230
## 4               T            286
## 5               T            298
## 6            0.00            306
```

5. A good package and function to tidy up dates into the same format is `lubridate`, e.g. try out this code #Dates with lubridate for most common combinations

```r
ymd("2015-08-25")
```

```
## [1] "2015-08-25"
```

```r
ymd("2015 August 25")
```

```
## [1] "2015-08-25"
```

```r
mdy("August 25, 2015")
```

```
## [1] "2015-08-25"
```

```r
hms("13:33:09")
```

```
## [1] "13H 33M 9S"
```

```r
ymd_hms("2015/08/25 13.33.09")
```

```
## [1] "2015-08-25 13:33:09 UTC"
```

We'll start by combining the year, month, and day columns and recoding the resulting character column as a date. We can use a combination of `stringr`, and `lubridate` to accomplish this task.

- Use `stringr`'s `str_replace()` to remove the `X`s from the `day` column of `weather3`.
- Create a new column called `date`. Use the `unite()` function from `tidyr` to paste together the `year`, `month`, and `day` columns in order, using `-` as a separator.
- Coerce the `date` column using the appropriate function from `lubridate`.
- Use select() to reorder columns, saving the result to `weather5`.
- View the head of `weather5`.

```r
weather3$day <- str_replace(weather3$day, "X", "") # Remove X's from day column
weather4 <- unite(weather3, date, year, month, day, sep = "-") # Unite the year, month, and day columns
weather4$date <- ymd(weather4$date) # Convert date column to proper date format using lubridates's ymd(
weather5 <- select(weather4, date, Events, CloudCover:WindDirDegrees) # Rearrange columns using dplyr's
head(weather5) # View the head of weather5
```

```
##         date    Events CloudCover Max.Dew.PointF Max.Gust.SpeedMPH
## 1 2014-12-01      Rain          6             46                29
## 2 2014-12-10      Rain          8             45                29
## 3 2014-12-11 Rain-Snow          8             37                28
## 4 2014-12-12      Snow          7             28                21
## 5 2014-12-13                    5             28                23
## 6 2014-12-14                    4             29                20
##   Max.Humidity Max.Sea.Level.PressureIn Max.TemperatureF
## 1           74                    30.45               64
## 2          100                    29.58               48
## 3           92                    29.81               39
## 4           85                    29.88               39
## 5           75                    29.86               42
## 6           82                    29.91               45
##   Max.VisibilityMiles Max.Wind.SpeedMPH Mean.Humidity
## 1                  10                22            63
## 2                  10                23            95
## 3                  10                21            87
## 4                  10                16            75
## 5                  10                17            65
```

```
## 6                  10                 15            68
##   Mean.Sea.Level.PressureIn Mean.TemperatureF Mean.VisibilityMiles
## 1                     30.13                52                   10
## 2                      29.5                43                    3
## 3                     29.61                36                    7
## 4                     29.85                35                   10
## 5                     29.82                37                   10
## 6                     29.83                39                   10
##   Mean.Wind.SpeedMPH MeanDew.PointF Min.DewpointF Min.Humidity
## 1                 13             40            26           52
## 2                 13             39            37           89
## 3                 13             31            27           82
## 4                 11             27            25           64
## 5                 12             26            24           55
## 6                 10             27            25           53
##   Min.Sea.Level.PressureIn Min.TemperatureF Min.VisibilityMiles
## 1                    30.01               39                  10
## 2                    29.43               38                   1
## 3                    29.44               32                   1
## 4                    29.81               31                   7
## 5                    29.78               32                  10
## 6                    29.78               33                  10
##   PrecipitationIn WindDirDegrees
## 1            0.01            268
## 2            0.28            357
## 3            0.02            230
## 4               T            286
## 5               T            298
## 6            0.00            306
```

6. Let's look closer at the column types as it is important that variables are coded appropriately for further statistical analysis. This is not yet the case with our weather data. Recall that functions such as `as.numeric()` and `as.character()` can be used to coerce variables into different types.

- Use `str()` to see how variables are stored in `weather5`.
- View the first 20 rows of weather5. Keep an eye out for strange values!
- Try coercing the `PrecipitationIn` column of `weather5` to numeric without saving the result.

```r
str(weather5) # View the structure of weather5
```

```
## 'data.frame':    366 obs. of  23 variables:
##  $ date                     : Date, format: "2014-12-01" "2014-12-10" ...
##  $ Events                   : chr  "Rain" "Rain" "Rain-Snow" "Snow" ...
##  $ CloudCover               : chr  "6" "8" "8" "7" ...
##  $ Max.Dew.PointF           : chr  "46" "45" "37" "28" ...
##  $ Max.Gust.SpeedMPH        : chr  "29" "29" "28" "21" ...
##  $ Max.Humidity             : chr  "74" "100" "92" "85" ...
##  $ Max.Sea.Level.PressureIn : chr  "30.45" "29.58" "29.81" "29.88" ...
##  $ Max.TemperatureF         : chr  "64" "48" "39" "39" ...
##  $ Max.VisibilityMiles      : chr  "10" "10" "10" "10" ...
##  $ Max.Wind.SpeedMPH        : chr  "22" "23" "21" "16" ...
##  $ Mean.Humidity            : chr  "63" "95" "87" "75" ...
##  $ Mean.Sea.Level.PressureIn: chr  "30.13" "29.5" "29.61" "29.85" ...
##  $ Mean.TemperatureF        : chr  "52" "43" "36" "35" ...
##  $ Mean.VisibilityMiles     : chr  "10" "3" "7" "10" ...
```

```
##  $ Mean.Wind.SpeedMPH       : chr  "13" "13" "13" "11" ...
##  $ MeanDew.PointF           : chr  "40" "39" "31" "27" ...
##  $ Min.DewpointF            : chr  "26" "37" "27" "25" ...
##  $ Min.Humidity             : chr  "52" "89" "82" "64" ...
##  $ Min.Sea.Level.PressureIn : chr  "30.01" "29.43" "29.44" "29.81" ...
##  $ Min.TemperatureF         : chr  "39" "38" "32" "31" ...
##  $ Min.VisibilityMiles      : chr  "10" "1" "1" "7" ...
##  $ PrecipitationIn          : chr  "0.01" "0.28" "0.02" "T" ...
##  $ WindDirDegrees           : chr  "268" "357" "230" "286" ...
```

```
head(weather5, 20) # Examine the first 20 rows of weather5. Are most of the characters numeric?
```

```
##          date    Events CloudCover Max.Dew.PointF Max.Gust.SpeedMPH
## 1  2014-12-01      Rain          6             46                29
## 2  2014-12-10      Rain          8             45                29
## 3  2014-12-11 Rain-Snow          8             37                28
## 4  2014-12-12      Snow          7             28                21
## 5  2014-12-13                    5             28                23
## 6  2014-12-14                    4             29                20
## 7  2014-12-15                    2             33                21
## 8  2014-12-16      Rain          8             42                10
## 9  2014-12-17      Rain          8             46                26
## 10 2014-12-18      Rain          7             34                30
## 11 2014-12-19                    4             25                23
## 12 2014-12-02 Rain-Snow          7             40                29
## 13 2014-12-20      Snow          6             30                26
## 14 2014-12-21      Snow          8             30                20
## 15 2014-12-22      Rain          7             39                22
## 16 2014-12-23      Rain          8             45                25
## 17 2014-12-24  Fog-Rain          8             46                15
## 18 2014-12-25      Rain          6             58                40
## 19 2014-12-26                    1             31                25
## 20 2014-12-27                    3             34                21
##    Max.Humidity Max.Sea.Level.PressureIn Max.TemperatureF
## 1            74                    30.45               64
## 2           100                    29.58               48
## 3            92                    29.81               39
## 4            85                    29.88               39
## 5            75                    29.86               42
## 6            82                    29.91               45
## 7            89                    30.15               42
## 8            96                    30.17               44
## 9           100                    29.91               49
## 10           89                    29.87               44
## 11           69                    30.15               37
## 12           92                    30.71               42
## 13           89                    30.31               36
## 14           85                    30.37               36
## 15           89                     30.4               44
## 16          100                    30.31               47
## 17          100                    30.13               46
## 18          100                    29.96               59
## 19           70                    30.16               50
## 20           70                    30.22               52
##    Max.VisibilityMiles Max.Wind.SpeedMPH Mean.Humidity
```

```
## 1                       10                 22             63
## 2                       10                 23             95
## 3                       10                 21             87
## 4                       10                 16             75
## 5                       10                 17             65
## 6                       10                 15             68
## 7                       10                 15             75
## 8                       10                  8             85
## 9                       10                 20             85
## 10                      10                 23             73
## 11                      10                 17             63
## 12                      10                 24             72
## 13                      10                 21             79
## 14                      10                 16             77
## 15                      10                 18             79
## 16                      10                 20             91
## 17                       2                 13             98
## 18                      10                 28             75
## 19                      10                 18             60
## 20                      10                 17             60
##    Mean.Sea.Level.PressureIn Mean.TemperatureF Mean.VisibilityMiles
## 1                      30.13                52                   10
## 2                       29.5                43                    3
## 3                      29.61                36                    7
## 4                      29.85                35                   10
## 5                      29.82                37                   10
## 6                      29.83                39                   10
## 7                      30.05                37                   10
## 8                      30.09                40                    9
## 9                      29.75                45                    6
## 10                     29.78                40                   10
## 11                     29.98                33                   10
## 12                     30.59                38                    8
## 13                     30.26                32                   10
## 14                     30.32                33                    9
## 15                     30.35                39                   10
## 16                     30.23                45                    5
## 17                      29.9                44                    1
## 18                     29.63                52                    8
## 19                     30.11                44                   10
## 20                     30.14                45                   10
##    Mean.Wind.SpeedMPH MeanDew.PointF Min.DewpointF Min.Humidity
## 1                  13             40            26           52
## 2                  13             39            37           89
## 3                  13             31            27           82
## 4                  11             27            25           64
## 5                  12             26            24           55
## 6                  10             27            25           53
## 7                   6             29            27           60
## 8                   4             36            30           73
## 9                  11             41            32           70
## 10                 14             30            26           57
## 11                 11             22            20           56
## 12                 15             27            17           51
```

```
## 13                 10              24             20             69
## 14                  9              27             25             69
## 15                  8              34             25             69
## 16                 13              42             37             82
## 17                  6              44             41             96
## 18                 14              43             29             49
## 19                 11              29             28             49
## 20                  9              31             29             50
##    Min.Sea.Level.PressureIn Min.TemperatureF Min.VisibilityMiles
## 1                     30.01               39                  10
## 2                     29.43               38                   1
## 3                     29.44               32                   1
## 4                     29.81               31                   7
## 5                     29.78               32                  10
## 6                     29.78               33                  10
## 7                     29.91               32                  10
## 8                     29.92               35                   5
## 9                     29.69               41                   1
## 10                    29.71               36                  10
## 11                    29.86               29                  10
## 12                     30.4               33                   2
## 13                    30.17               27                   7
## 14                    30.28               30                   6
## 15                     30.3               33                   4
## 16                    30.16               42                   1
## 17                    29.55               41                   0
## 18                    29.47               44                   1
## 19                    29.99               37                  10
## 20                    30.03               38                  10
##    PrecipitationIn WindDirDegrees
## 1             0.01            268
## 2             0.28            357
## 3             0.02            230
## 4                T            286
## 5                T            298
## 6             0.00            306
## 7             0.00            324
## 8                T             79
## 9             0.43            311
## 10            0.01            281
## 11            0.00            305
## 12            0.10             62
## 13               T            350
## 14               T              2
## 15            0.05             24
## 16            0.25             63
## 17            0.56             12
## 18            0.14            250
## 19            0.00            255
## 20            0.00            251
```

```r
as.numeric(weather5$PrecipitationIn) # See what happens if we try to convert PrecipitationIn to numeric
```

```
## Warning: NAs durch Umwandlung erzeugt
```

```
##   [1] 0.01 0.28 0.02   NA   NA 0.00 0.00   NA 0.43 0.01 0.00 0.10   NA   NA
##  [15] 0.05 0.25 0.56 0.14 0.00 0.00 0.01 0.00 0.44 0.00 0.00 0.00 0.11 1.09
##  [29] 0.13 0.03 2.90 0.00 0.00 0.00 0.20 0.00   NA 0.12 0.00 0.00 0.15 0.00
##  [43] 0.00 0.00 0.00   NA 0.00 0.71 0.00 0.10 0.95 0.01   NA 0.62 0.06 0.05
##  [57] 0.57 0.00 0.02   NA 0.00 0.01 0.00 0.05 0.01 0.03 0.00 0.23 0.39 0.00
##  [71] 0.02 0.01 0.06 0.78 0.00 0.17 0.11 0.00   NA 0.07 0.02 0.00 0.00 0.00
##  [85] 0.00 0.09   NA 0.07 0.37 0.88 0.17 0.06 0.01 0.00 0.00 0.80 0.27 0.00
##  [99] 0.14 0.00 0.00 0.01 0.05 0.09 0.00 0.00 0.00 0.04 0.80 0.21 0.12 0.00
## [113] 0.26   NA 0.00 0.02   NA 0.00 0.00   NA 0.00 0.00 0.09 0.00 0.00 0.00
## [127] 0.01 0.00 0.00 0.06 0.00 0.00 0.00 0.61 0.54   NA 0.00   NA 0.00 0.00
## [141] 0.10 0.07 0.00 0.03 0.00 0.39 0.00 0.00 0.03 0.26 0.09 0.00 0.00 0.00
## [155] 0.02 0.00 0.00 0.00   NA 0.00 0.00 0.27 0.00 0.00 0.00   NA 0.00 0.00
## [169]   NA 0.00 0.00   NA 0.00 0.00 0.00 0.91 0.00 0.02 0.00 0.00 0.00 0.00
## [183] 0.38 0.00 0.00 0.00   NA 0.00 0.40   NA 0.00 0.00 0.00 0.74 0.04 1.72
## [197] 0.00 0.01 0.00 0.00   NA 0.20 1.43   NA 0.00 0.00 0.00   NA 0.09 0.00
## [211]   NA   NA 0.50 1.12 0.00 0.00 0.00 0.03   NA 0.00   NA 0.14   NA 0.00
## [225]   NA   NA 0.00 0.00 0.01 0.00   NA 0.06 0.00 0.00 0.00 0.02 0.00   NA
## [239] 0.00 0.00 0.02   NA 0.15   NA 0.00 0.83 0.00 0.00 0.00 0.08 0.00 0.00
## [253] 0.14 0.00 0.00 0.00 0.63   NA 0.02   NA 0.00   NA 0.00 0.00 0.00 0.00
## [267] 0.00 0.00 0.49 0.00 0.00 0.00 0.00 0.00 0.00 0.17 0.66 0.01 0.38 0.00
## [281] 0.00 0.00 0.00 0.00 0.00 0.00   NA 0.00 0.00 0.00 0.00 0.00 0.00 0.00
## [295] 0.00 0.04 0.01 2.46   NA 0.00 0.00 0.00 0.20 0.00   NA 0.00 0.00 0.00
## [309] 0.12 0.00 0.00   NA   NA   NA 0.00 0.08   NA 0.07   NA 0.00 0.00 0.03
## [323] 0.00 0.00 0.36 0.73 0.01 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.34   NA
## [337] 0.07 0.54 0.04 0.01 0.00 0.00 0.00 0.00 0.00   NA 0.00 0.86 0.00 0.30
## [351] 0.04 0.00 0.00 0.00 0.00 0.21 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
## [365] 0.00 0.14
```

Scroll the output, notice the warning message. Go back to the results of the head command if need be. What values in PrecipitationIn would become NA if coerced to numbers? Why would they be in the dataset to begin with?

7. As you saw in the last exercise, `T` was used to denote a trace amount (i.e. too small to be accurately measured) of precipitation in the `PrecipitationIn` column. In order to coerce this column to numeric, you'll need to deal with this somehow. To keep things simple, we will just replace `T` with zero, as a string ("0").

- Use `str_replace()` from `stringr` to make the proper replacements in the `PrecipitationIn` column of `weather5`.
- Run the call to mutate_at to conveniently apply `as.numeric()` to all columns from `CloudCover` through `WindDirDegrees` (reading left to right in the data), saving the result to `weather6`.
- View the structure of weather6 to confirm the coercions were successful.

```r
weather5$PrecipitationIn <- str_replace(weather5$PrecipitationIn, "T", "0") # Replace "T" with "0" (T =
weather6 <- mutate_at(weather5, vars(CloudCover:WindDirDegrees), funs(as.numeric)) # Convert characters
```

```
## Warning: funs() is soft deprecated as of dplyr 0.8.0
## please use list() instead
##
## # Before:
## funs(name = f(.)
##
## # After:
## list(name = ~f(.))
## This warning is displayed once per session.
```

```r
str(weather6) # Look at result
```

```
## 'data.frame':    366 obs. of  23 variables:
##  $ date                     : Date, format: "2014-12-01" "2014-12-10" ...
##  $ Events                   : chr  "Rain" "Rain" "Rain-Snow" "Snow" ...
##  $ CloudCover               : num  6 8 8 7 5 4 2 8 8 7 ...
##  $ Max.Dew.PointF           : num  46 45 37 28 28 29 33 42 46 34 ...
##  $ Max.Gust.SpeedMPH        : num  29 29 28 21 23 20 21 10 26 30 ...
##  $ Max.Humidity             : num  74 100 92 85 75 82 89 96 100 89 ...
##  $ Max.Sea.Level.PressureIn : num  30.4 29.6 29.8 29.9 29.9 ...
##  $ Max.TemperatureF         : num  64 48 39 39 42 45 42 44 49 44 ...
##  $ Max.VisibilityMiles      : num  10 10 10 10 10 10 10 10 10 10 ...
##  $ Max.Wind.SpeedMPH        : num  22 23 21 16 17 15 15 8 20 23 ...
##  $ Mean.Humidity            : num  63 95 87 75 65 68 75 85 85 73 ...
##  $ Mean.Sea.Level.PressureIn: num  30.1 29.5 29.6 29.9 29.8 ...
##  $ Mean.TemperatureF        : num  52 43 36 35 37 39 37 40 45 40 ...
##  $ Mean.VisibilityMiles     : num  10 3 7 10 10 10 10 9 6 10 ...
##  $ Mean.Wind.SpeedMPH       : num  13 13 13 11 12 10 6 4 11 14 ...
##  $ MeanDew.PointF           : num  40 39 31 27 26 27 29 36 41 30 ...
##  $ Min.DewpointF            : num  26 37 27 25 24 25 27 30 32 26 ...
##  $ Min.Humidity             : num  52 89 82 64 55 53 60 73 70 57 ...
##  $ Min.Sea.Level.PressureIn : num  30 29.4 29.4 29.8 29.8 ...
##  $ Min.TemperatureF         : num  39 38 32 31 32 33 32 35 41 36 ...
##  $ Min.VisibilityMiles      : num  10 1 1 7 10 10 10 5 1 10 ...
##  $ PrecipitationIn          : num  0.01 0.28 0.02 0 0 0 0 0 0.43 0.01 ...
##  $ WindDirDegrees           : num  268 357 230 286 298 306 324 79 311 281 ...
```

8. Before dealing with missing values in the data, it's important to find them and figure out why they exist in the first place. If your dataset is too big to look at all at once, like it is here, remember you can use `sum()` and `is.na()` to quickly size up the situation by counting the number of NA values. The `summary()` function may also come in handy for identifying which variables contain the missing values. Finally, the `which()` function is useful for locating the missing values within a particular column.

- Use `sum()` and `is.na()` to count the number of `NA` values in `weather6`.
- Look at a `summary()` of `weather6` to figure out how the missings are distributed among the different variables.
- Use `which()` to identify the indices (i.e. row numbers) where `Max.Gust.SpeedMPH` is `NA` and save the result to `ind`.
- Use `ind` to look at the full rows of `weather6` for which `Max.Gust.SpeedMPH` is missing.

```r
sum(is.na(weather6)) # Count missing values
```

```
## [1] 6
```

```r
summary(weather6) # Find missing values
```

```
##       date               Events            CloudCover    Max.Dew.PointF
##  Min.   :2014-12-01   Length:366         Min.   :0.000   Min.   :-6.00
##  1st Qu.:2015-03-02   Class :character   1st Qu.:3.000   1st Qu.:32.00
##  Median :2015-06-01   Mode  :character   Median :5.000   Median :47.50
##  Mean   :2015-06-01                      Mean   :4.708   Mean   :45.48
##  3rd Qu.:2015-08-31                      3rd Qu.:7.000   3rd Qu.:61.00
##  Max.   :2015-12-01                      Max.   :8.000   Max.   :75.00
##
##  Max.Gust.SpeedMPH  Max.Humidity     Max.Sea.Level.PressureIn
```

```
##  Min.   : 0.00     Min.   :  39.00   Min.   :29.58
##  1st Qu.:21.00     1st Qu.:  73.25   1st Qu.:30.00
##  Median :25.50     Median :  86.00   Median :30.14
##  Mean   :26.99     Mean   :  85.69   Mean   :30.16
##  3rd Qu.:31.25     3rd Qu.:  93.00   3rd Qu.:30.31
##  Max.   :94.00     Max.   :1000.00   Max.   :30.88
##  NA's   :6
##  Max.TemperatureF Max.VisibilityMiles Max.Wind.SpeedMPH Mean.Humidity
##  Min.   :18.00    Min.   : 2.000      Min.   : 8.00     Min.   :28.00
##  1st Qu.:42.00    1st Qu.:10.000      1st Qu.:16.00     1st Qu.:56.00
##  Median :60.00    Median :10.000      Median :20.00     Median :66.00
##  Mean   :58.93    Mean   : 9.907      Mean   :20.62     Mean   :66.02
##  3rd Qu.:76.00    3rd Qu.:10.000      3rd Qu.:24.00     3rd Qu.:76.75
##  Max.   :96.00    Max.   :10.000      Max.   :38.00     Max.   :98.00
##
##  Mean.Sea.Level.PressureIn Mean.TemperatureF Mean.VisibilityMiles
##  Min.   :29.49             Min.   : 8.00     Min.   :-1.000
##  1st Qu.:29.87             1st Qu.:36.25     1st Qu.: 8.000
##  Median :30.03             Median :53.50     Median :10.000
##  Mean   :30.04             Mean   :51.40     Mean   : 8.861
##  3rd Qu.:30.19             3rd Qu.:68.00     3rd Qu.:10.000
##  Max.   :30.77             Max.   :84.00     Max.   :10.000
##
##  Mean.Wind.SpeedMPH MeanDew.PointF   Min.DewpointF     Min.Humidity
##  Min.   : 4.00      Min.   :-11.00   Min.   :-18.00   Min.   :16.00
##  1st Qu.: 8.00      1st Qu.: 24.00   1st Qu.: 16.25   1st Qu.:35.00
##  Median :10.00      Median : 41.00   Median : 35.00   Median :46.00
##  Mean   :10.68      Mean   : 38.96   Mean   : 32.25   Mean   :48.31
##  3rd Qu.:13.00      3rd Qu.: 56.00   3rd Qu.: 51.00   3rd Qu.:60.00
##  Max.   :22.00      Max.   : 71.00   Max.   : 68.00   Max.   :96.00
##
##  Min.Sea.Level.PressureIn Min.TemperatureF Min.VisibilityMiles
##  Min.   :29.16            Min.   :-3.00    Min.   : 0.000
##  1st Qu.:29.76            1st Qu.:30.00    1st Qu.: 2.000
##  Median :29.94            Median :46.00    Median :10.000
##  Mean   :29.93            Mean   :43.33    Mean   : 6.716
##  3rd Qu.:30.09            3rd Qu.:60.00    3rd Qu.:10.000
##  Max.   :30.64            Max.   :74.00    Max.   :10.000
##
##  PrecipitationIn  WindDirDegrees
##  Min.   :0.0000   Min.   :  1.0
##  1st Qu.:0.0000   1st Qu.:113.0
##  Median :0.0000   Median :222.0
##  Mean   :0.1016   Mean   :200.1
##  3rd Qu.:0.0400   3rd Qu.:275.0
##  Max.   :2.9000   Max.   :360.0
##
```

```r
ind <- which(is.na(weather6$Max.Gust.SpeedMPH)) # Find indices of NAs in Max.Gust.SpeedMPH
weather6[ind, ] # Look at the full rows for records missing Max.Gust.SpeedMPH
```

```
##           date Events CloudCover Max.Dew.PointF Max.Gust.SpeedMPH
## 161 2015-05-18    Fog          6             52                NA
## 205 2015-06-03                 7             48                NA
## 273 2015-08-08                 4             61                NA
```

```
## 275 2015-09-01                 1            63                NA
## 308 2015-10-12                 0            56                NA
## 358 2015-11-03                 1            44                NA
##     Max.Humidity Max.Sea.Level.PressureIn Max.TemperatureF
## 161          100                    30.30               58
## 205           93                    30.31               56
## 273           87                    30.02               76
## 275           78                    30.06               79
## 308           89                    29.86               76
## 358           82                    30.25               73
##     Max.VisibilityMiles Max.Wind.SpeedMPH Mean.Humidity
## 161                  10                16            79
## 205                  10                14            82
## 273                  10                14            68
## 275                  10                15            65
## 308                  10                15            65
## 358                  10                16            57
##     Mean.Sea.Level.PressureIn Mean.TemperatureF Mean.VisibilityMiles
## 161                     30.23                54                    8
## 205                     30.24                52                   10
## 273                     29.99                69                   10
## 275                     30.02                74                   10
## 308                     29.80                64                   10
## 358                     30.13                60                   10
##     Mean.Wind.SpeedMPH MeanDew.PointF Min.DewpointF Min.Humidity
## 161                 10             48            43           57
## 205                  7             45            43           71
## 273                  6             57            54           49
## 275                  9             62            59           52
## 308                  8             51            48           41
## 358                  8             42            40           31
##     Min.Sea.Level.PressureIn Min.TemperatureF Min.VisibilityMiles
## 161                    30.12               49                   0
## 205                    30.19               47                  10
## 273                    29.95               61                  10
## 275                    29.96               69                  10
## 308                    29.74               51                  10
## 358                    30.06               47                  10
##     PrecipitationIn WindDirDegrees
## 161               0             72
## 205               0             90
## 273               0             45
## 275               0             54
## 308               0            199
## 358               0            281
```

9. Besides missing values, we want to know if there are values in the data that are too extreme or bizarre to be plausible. A great way to start the search for these values is with `summary()`. Once implausible values are identified, they must be dealt with in an intelligent and informed way. Sometimes the best way forward is obvious and other times it may require some research and/or discussions with the original collectors of the data.

- View a `summary()` of `weather6`.
- Use `which()` to find the index of the erroneous element of `weather6$Max.Humidity`, saving the result

to ind.

- Use ind to look at the full row of weather6 for that day. You discover an extra zero was accidentally added to this value. Correct it in the data.

```
summary(weather6) # Review distributions for all variables
```

```
##       date               Events            CloudCover    Max.Dew.PointF
##  Min.   :2014-12-01   Length:366         Min.   :0.000   Min.   :-6.00
##  1st Qu.:2015-03-02   Class :character   1st Qu.:3.000   1st Qu.:32.00
##  Median :2015-06-01   Mode  :character   Median :5.000   Median :47.50
##  Mean   :2015-06-01                      Mean   :4.708   Mean   :45.48
##  3rd Qu.:2015-08-31                      3rd Qu.:7.000   3rd Qu.:61.00
##  Max.   :2015-12-01                      Max.   :8.000   Max.   :75.00
##
##  Max.Gust.SpeedMPH  Max.Humidity     Max.Sea.Level.PressureIn
##  Min.   : 0.00     Min.   :  39.00   Min.   :29.58
##  1st Qu.:21.00     1st Qu.:  73.25   1st Qu.:30.00
##  Median :25.50     Median :  86.00   Median :30.14
##  Mean   :26.99     Mean   :  85.69   Mean   :30.16
##  3rd Qu.:31.25     3rd Qu.:  93.00   3rd Qu.:30.31
##  Max.   :94.00     Max.   :1000.00   Max.   :30.88
##  NA's   :6
##  Max.TemperatureF Max.VisibilityMiles Max.Wind.SpeedMPH Mean.Humidity
##  Min.   :18.00    Min.   : 2.000      Min.   : 8.00     Min.   :28.00
##  1st Qu.:42.00    1st Qu.:10.000      1st Qu.:16.00     1st Qu.:56.00
##  Median :60.00    Median :10.000      Median :20.00     Median :66.00
##  Mean   :58.93    Mean   : 9.907      Mean   :20.62     Mean   :66.02
##  3rd Qu.:76.00    3rd Qu.:10.000      3rd Qu.:24.00     3rd Qu.:76.75
##  Max.   :96.00    Max.   :10.000      Max.   :38.00     Max.   :98.00
##
##  Mean.Sea.Level.PressureIn Mean.TemperatureF Mean.VisibilityMiles
##  Min.   :29.49             Min.   : 8.00     Min.   :-1.000
##  1st Qu.:29.87             1st Qu.:36.25     1st Qu.: 8.000
##  Median :30.03             Median :53.50     Median :10.000
##  Mean   :30.04             Mean   :51.40     Mean   : 8.861
##  3rd Qu.:30.19             3rd Qu.:68.00     3rd Qu.:10.000
##  Max.   :30.77             Max.   :84.00     Max.   :10.000
##
##  Mean.Wind.SpeedMPH MeanDew.PointF   Min.DewpointF     Min.Humidity
##  Min.   : 4.00      Min.   :-11.00   Min.   :-18.00   Min.   :16.00
##  1st Qu.: 8.00      1st Qu.: 24.00   1st Qu.: 16.25   1st Qu.:35.00
##  Median :10.00      Median : 41.00   Median : 35.00   Median :46.00
##  Mean   :10.68      Mean   : 38.96   Mean   : 32.25   Mean   :48.31
##  3rd Qu.:13.00      3rd Qu.: 56.00   3rd Qu.: 51.00   3rd Qu.:60.00
##  Max.   :22.00      Max.   : 71.00   Max.   : 68.00   Max.   :96.00
##
##  Min.Sea.Level.PressureIn Min.TemperatureF Min.VisibilityMiles
##  Min.   :29.16            Min.   :-3.00    Min.   : 0.000
##  1st Qu.:29.76            1st Qu.:30.00    1st Qu.: 2.000
##  Median :29.94            Median :46.00    Median :10.000
##  Mean   :29.93            Mean   :43.33    Mean   : 6.716
##  3rd Qu.:30.09            3rd Qu.:60.00    3rd Qu.:10.000
##  Max.   :30.64            Max.   :74.00    Max.   :10.000
##
```

```
##  PrecipitationIn  WindDirDegrees
##  Min.   :0.0000   Min.   :  1.0
##  1st Qu.:0.0000   1st Qu.:113.0
##  Median :0.0000   Median :222.0
##  Mean   :0.1016   Mean   :200.1
##  3rd Qu.:0.0400   3rd Qu.:275.0
##  Max.   :2.9000   Max.   :360.0
##
```

```r
ind <- which(weather6$Max.Humidity == 1000) # Find row with Max.Humidity of 1000
weather6[ind, ] # Look at the data for that day
```

```
##           date                 Events CloudCover Max.Dew.PointF
## 135 2015-04-21 Fog-Rain-Thunderstorm          6             57
##     Max.Gust.SpeedMPH Max.Humidity Max.Sea.Level.PressureIn
## 135                94         1000                    29.75
##     Max.TemperatureF Max.VisibilityMiles Max.Wind.SpeedMPH Mean.Humidity
## 135               65                  10                20            71
##     Mean.Sea.Level.PressureIn Mean.TemperatureF Mean.VisibilityMiles
## 135                      29.6                56                    5
##     Mean.Wind.SpeedMPH MeanDew.PointF Min.DewpointF Min.Humidity
## 135                 10             49            36           42
##     Min.Sea.Level.PressureIn Min.TemperatureF Min.VisibilityMiles
## 135                    29.53               46                   0
##     PrecipitationIn WindDirDegrees
## 135            0.54            184
```

```r
weather6$Max.Humidity[ind] <- 100 # Change 1000 to 100
```

10. You've discovered and repaired one obvious error in the data, but it appears that there's another. Sometimes you get lucky and can infer the correct or intended value from the other data. For example, if you know the minimum and maximum values of a particular metric on a given day.

- Use `summary()` to look at the value of only the `Mean.VisibilityMiles` variable of `weather6`.
- Determine the element of the value that is clearly erroneous in this column, saving the result to `ind`.
- Use `ind` to look at the full row of `weather6` for this day.
- Inspect the values of other variables for this day to determine the correct value of `Mean.VisibilityMiles`, then make the appropriate fix.

```r
summary(weather6$Mean.VisibilityMiles) # Look at summary of Mean.VisibilityMiles
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -1.000   8.000  10.000   8.861  10.000  10.000
```

```r
ind <- which(weather6$Mean.VisibilityMiles == -1) # Get index of row with -1 value
weather6[ind, ] # Look at full row
```

```
##           date Events CloudCover Max.Dew.PointF Max.Gust.SpeedMPH
## 192 2015-06-18                 5             54                23
##     Max.Humidity Max.Sea.Level.PressureIn Max.TemperatureF
## 192           72                    30.14               76
##     Max.VisibilityMiles Max.Wind.SpeedMPH Mean.Humidity
## 192                  10                17            59
##     Mean.Sea.Level.PressureIn Mean.TemperatureF Mean.VisibilityMiles
## 192                     30.04                67                   -1
##     Mean.Wind.SpeedMPH MeanDew.PointF Min.DewpointF Min.Humidity
```

```
## 192                  10              49            45            46
##     Min.Sea.Level.PressureIn Min.TemperatureF Min.VisibilityMiles
## 192                     29.93                57                  10
##     PrecipitationIn WindDirDegrees
## 192               0            189
```

```r
weather6$Mean.VisibilityMiles[ind] <- 10 # Set Mean.VisibilityMiles to the appropriate value
```

11. In addition to dealing with obvious errors in the data, we want to see if there are other extreme values. In addition to the trusty `summary()` function, `hist()` is useful for quickly getting a feel for how different variables are distributed.

- Check a `summary()` of `weather6` one more time for extreme or unexpected values.
- View a histogram for `MeanDew.PointF`, `Min.TemperatureF` and `Mean.TemperatureF` to compare distributions.

```r
summary(weather6) # Review summary of full data once more
```

```
##       date               Events            CloudCover     Max.Dew.PointF
##  Min.   :2014-12-01   Length:366         Min.   :0.000   Min.   :-6.00
##  1st Qu.:2015-03-02   Class :character   1st Qu.:3.000   1st Qu.:32.00
##  Median :2015-06-01   Mode  :character   Median :5.000   Median :47.50
##  Mean   :2015-06-01                      Mean   :4.708   Mean   :45.48
##  3rd Qu.:2015-08-31                      3rd Qu.:7.000   3rd Qu.:61.00
##  Max.   :2015-12-01                      Max.   :8.000   Max.   :75.00
##
##  Max.Gust.SpeedMPH  Max.Humidity    Max.Sea.Level.PressureIn
##  Min.   : 0.00     Min.   : 39.00   Min.   :29.58
##  1st Qu.:21.00     1st Qu.: 73.25   1st Qu.:30.00
##  Median :25.50     Median : 86.00   Median :30.14
##  Mean   :26.99     Mean   : 83.23   Mean   :30.16
##  3rd Qu.:31.25     3rd Qu.: 93.00   3rd Qu.:30.31
##  Max.   :94.00     Max.   :100.00   Max.   :30.88
##  NA's   :6
##  Max.TemperatureF Max.VisibilityMiles Max.Wind.SpeedMPH Mean.Humidity
##  Min.   :18.00    Min.   : 2.000      Min.   : 8.00     Min.   :28.00
##  1st Qu.:42.00    1st Qu.:10.000      1st Qu.:16.00     1st Qu.:56.00
##  Median :60.00    Median :10.000      Median :20.00     Median :66.00
##  Mean   :58.93    Mean   : 9.907      Mean   :20.62     Mean   :66.02
##  3rd Qu.:76.00    3rd Qu.:10.000      3rd Qu.:24.00     3rd Qu.:76.75
##  Max.   :96.00    Max.   :10.000      Max.   :38.00     Max.   :98.00
##
##  Mean.Sea.Level.PressureIn Mean.TemperatureF Mean.VisibilityMiles
##  Min.   :29.49             Min.   : 8.00     Min.   : 1.000
##  1st Qu.:29.87             1st Qu.:36.25     1st Qu.: 8.000
##  Median :30.03             Median :53.50     Median :10.000
##  Mean   :30.04             Mean   :51.40     Mean   : 8.891
##  3rd Qu.:30.19             3rd Qu.:68.00     3rd Qu.:10.000
##  Max.   :30.77             Max.   :84.00     Max.   :10.000
##
##  Mean.Wind.SpeedMPH MeanDew.PointF   Min.DewpointF     Min.Humidity
##  Min.   : 4.00      Min.   :-11.00   Min.   :-18.00   Min.   :16.00
##  1st Qu.: 8.00      1st Qu.: 24.00   1st Qu.: 16.25   1st Qu.:35.00
##  Median :10.00      Median : 41.00   Median : 35.00   Median :46.00
##  Mean   :10.68      Mean   : 38.96   Mean   : 32.25   Mean   :48.31
```

```
##   3rd Qu.:13.00      3rd Qu.: 56.00   3rd Qu.: 51.00   3rd Qu.:60.00
##   Max.   :22.00      Max.   : 71.00   Max.   : 68.00   Max.   :96.00
##
##   Min.Sea.Level.PressureIn Min.TemperatureF Min.VisibilityMiles
##   Min.   :29.16            Min.   :-3.00    Min.   : 0.000
##   1st Qu.:29.76            1st Qu.:30.00    1st Qu.: 2.000
##   Median :29.94            Median :46.00    Median :10.000
##   Mean   :29.93            Mean   :43.33    Mean   : 6.716
##   3rd Qu.:30.09            3rd Qu.:60.00    3rd Qu.:10.000
##   Max.   :30.64            Max.   :74.00    Max.   :10.000
##
##   PrecipitationIn  WindDirDegrees
##   Min.   :0.0000   Min.   :  1.0
##   1st Qu.:0.0000   1st Qu.:113.0
##   Median :0.0000   Median :222.0
##   Mean   :0.1016   Mean   :200.1
##   3rd Qu.:0.0400   3rd Qu.:275.0
##   Max.   :2.9000   Max.   :360.0
##
```

```r
hist(weather6$MeanDew.PointF) # Look at histogram for MeanDew.PointF
```

IntroRExercisesWithSolutions_files/figure-latex/unnamed-chunk-142-1.pdf

```r
hist(weather6$Min.TemperatureF) # Look at histogram for Min.TemperatureF
```

IntroRExercisesWithSolutions_files/figure-latex/unnamed-chunk-142-2.pdf

```r
hist(weather6$Mean.TemperatureF) # Compare to histogram for Mean.TemperatureF
```

IntroRExercisesWithSolutions_files/figure-latex/unnamed-chunk-142-3.pdf

12. Finally, the `Events` column contains an empty string ("") for any day on which there was no significant weather event such as rain, fog, a thunderstorm, etc. However, if it's the first time you're seeing these data, it may not be obvious that this is the case, so it's best for us to be explicit and replace the empty strings with something more meaningful. Replace all empty strings in the events column of weather6 with "None". One last time, print out the first 6 rows of the weather6 data frame to see the changes.

```r
weather6$Events[weather6$Events == ""] <- "None" # Replace empty cells in events column
head(weather6) # Print the first 6 rows of weather6
```

```
##         date    Events CloudCover Max.Dew.PointF Max.Gust.SpeedMPH
```

```
## 1 2014-12-01      Rain          6            46               29
## 2 2014-12-10      Rain          8            45               29
## 3 2014-12-11 Rain-Snow          8            37               28
## 4 2014-12-12      Snow          7            28               21
## 5 2014-12-13      None          5            28               23
## 6 2014-12-14      None          4            29               20
##   Max.Humidity Max.Sea.Level.PressureIn Max.TemperatureF
## 1           74                    30.45               64
## 2          100                    29.58               48
## 3           92                    29.81               39
## 4           85                    29.88               39
## 5           75                    29.86               42
## 6           82                    29.91               45
##   Max.VisibilityMiles Max.Wind.SpeedMPH Mean.Humidity
## 1                  10                22            63
## 2                  10                23            95
## 3                  10                21            87
## 4                  10                16            75
## 5                  10                17            65
## 6                  10                15            68
##   Mean.Sea.Level.PressureIn Mean.TemperatureF Mean.VisibilityMiles
## 1                     30.13                52                   10
## 2                     29.50                43                    3
## 3                     29.61                36                    7
## 4                     29.85                35                   10
## 5                     29.82                37                   10
## 6                     29.83                39                   10
##   Mean.Wind.SpeedMPH MeanDew.PointF Min.DewpointF Min.Humidity
## 1                 13             40            26           52
## 2                 13             39            37           89
## 3                 13             31            27           82
## 4                 11             27            25           64
## 5                 12             26            24           55
## 6                 10             27            25           53
##   Min.Sea.Level.PressureIn Min.TemperatureF Min.VisibilityMiles
## 1                    30.01               39                  10
## 2                    29.43               38                   1
## 3                    29.44               32                   1
## 4                    29.81               31                   7
## 5                    29.78               32                  10
## 6                    29.78               33                  10
##   PrecipitationIn WindDirDegrees
## 1            0.01            268
## 2            0.28            357
## 3            0.02            230
## 4            0.00            286
## 5            0.00            298
## 6            0.00            306
```

User-defined functions

1. Define a Cobb-Douglas production function with two inputs vectors,

$$x = \begin{pmatrix} L \\ K \end{pmatrix}$$
$$\theta = \begin{pmatrix} A \\ \alpha \\ \beta \end{pmatrix}$$

   and scalar output

$$y = AL^\alpha K^\beta.$$

   Evaluate the function at

$$x = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$$
$$\theta = \begin{pmatrix} 1 \\ 0.3 \\ 0.8 \end{pmatrix}.$$

```
cobb_douglas <- function(x, theta) {
  y <- theta[1] * x[1]^theta[2] * x[2]^theta[3]
  return(y)
}
cobb_douglas(x = c(2, 3), theta = c(1, 0.3, 0.8))
```

```
## [1] 2.964872
```

2. Define a function `lowdecile` with one input vector $(x_1, \ldots, x_n)$ of arbitrary length. The function should compute and return the mean of all observations in the lowest decile. Define the vector

$$x = (0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2, \ldots, 9, 9, 9, 9)$$

   and apply `lowdecile` to x.

```
lowdecile <- function(x) {
  quantil <- x[x <= quantile(x, p = 0.1)]
  return(mean(quantil))
}
lowdecile(x = rep(0:9, each = 4))
```

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

Programming

1. This exercise illustrates that loops are often not very efficient.

- Create the vector $x = (1, 2, \ldots, 1\,000\,000)$ and convert it from *integer* to *numeric* using the conversion command `as.numeric`.

```
x <- 1:1e+06
class(x)
```

```
## [1] "integer"
```

```
x <- as.numeric(x)
class(x)
```

```
## [1] "numeric"
```

- Write a `for`-loop to compute the sum of all vector elements without using the `sum` command. Put the command `p0 <- proc.time()[3]` in front of the loop and the command `print(proc.time()[3]-p0)` at the end. These commands allow to measure the execution time of the loop.

```
S <- 0  # initialize
p0 <- proc.time()[3]  #Startzeitpunkt festlegen
for (i in x) {
    S <- S + i
}
print(S)
```

```
## [1] 500000500000
```

```
print(proc.time()[3] - p0)  #time used
```

```
## elapsed
##   0.071
```

*Compare your result with the execution time of the `sum` command.

```
p0 <- proc.time()[3]
sum(x)
```

```
## [1] 500000500000
```

```
print(proc.time()[3] - p0)
```

```
## elapsed
##   0.001
```

2. Create a grid vector x of 60 equidistant points $x_{1},...,x_{60} $ on the interval $[-10, 10]$, and another grid vector y of 70 points $y_1, \ldots, y_{70}$ on $[-10, 10]$. Create an empty matrix Z of dimension 60×70.

Write a double loop to compute the matrix elements

$$Z_{ij} = \frac{10}{r_{ij}} \cdot \sin(r_{ij})$$

where $r_{ij} = \sqrt{x_i^2 + y_j^2}$. Execute `persp(x,y,Z)`.

```
x <- seq(-10, 10, length = 60)
y <- seq(-10, 10, length = 70)
Z <- matrix(NA, 60, 70)
for (i in 1:length(x)) {
    for (j in 1:length(y)) {
        r <- sqrt(x[i]^2 + y[j]^2) # note that r is overwritten in each run of the loop
        Z[i, j] <- 10/r * sin(r)
    }
```

```r
}
persp(x, y, Z, ticktype = "detailed", col = "lightblue")
```

IntroRExercisesWithSolutions_files/figure-latex/unnamed-chunk-151-1.pdf

3. Load the data set **fussballdaten.csv**. It contains all *1. Bundesliga* results between the seasons 1996/1997 and 2008/2009.

```r
fussballdaten <- read.csv2("data/fussballdaten.csv", as.is = TRUE)
```

- Create an alphabetically ordered vector of all clubs in the data set.

```r
home <- fussballdaten$Heim
away <- fussballdaten$Auswaerts
clubs <- sort(unique(home))
```

- Write a loop over all clubs. For each club compute the proportion of games won.

```r
ngames <- dim(fussballdaten)[1] # number of games in dataset
GoalsH <- fussballdaten$ToreH
GoalsA <- fussballdaten$ToreA
winner <- rep(NA, ngames)
propwin <- rep(NA, length(clubs))
# Get winning teams
for (i in 1:ngames) {
    if (GoalsH[i] > GoalsA[i]) {
        winner[i] <- home[i]
    }
    if (GoalsA[i] > GoalsH[i]) {
        winner[i] <- away[i]
    }
    if (GoalsH[i] == GoalsA[i]) {
        winner[i] <- "Remis"
    }
}
# Get proportions
for (i in 1:length(clubs)) {
    win <- sum(winner == clubs[i])  # Games won by club i
    tot <- sum(home == clubs[i] | away == clubs[i])  # Number of mathes of club i
    propwin[i] <- win/tot
}
names(propwin) <- clubs
print(propwin)
```

```
## 1860muenchen       aachen    bielefeld    bmuenchen       bochum
##    0.3496732    0.2647059    0.2720588    0.6153846    0.3202614
##       bremen      cottbus     dortmund  duesseldorf     duisburg
##    0.4751131    0.2823529    0.4298643    0.2500000    0.2598039
##      fckoeln    frankfurt     freiburg      hamburg     hannover
##    0.2689076    0.2830882    0.2720588    0.3891403    0.3235294
##    herthabsc    karlsruhe     klautern   leverkusen        mainz
```

```
##    0.4144385    0.3308824    0.3823529    0.4728507    0.2843137
##    mgladbach    nuernberg      rostock      schalke      stpauli
##    0.2764706    0.2731092    0.3048128    0.4343891    0.1960784
##    uerdingen      uhaching          ulm vfbstuttgart    wolfsburg
##    0.1470588    0.2941176    0.2647059    0.4185520    0.3582888
```

- Order the clubs descendingly according to the proportion of games won and plot a `barplot` of the proportion.

```
sort(propwin, decreasing = TRUE)
```

```
##    bmuenchen       bremen   leverkusen      schalke     dortmund
##    0.6153846    0.4751131    0.4728507    0.4343891    0.4298643
## vfbstuttgart    herthabsc      hamburg     klautern    wolfsburg
##    0.4185520    0.4144385    0.3891403    0.3823529    0.3582888
## 1860muenchen    karlsruhe     hannover       bochum      rostock
##    0.3496732    0.3308824    0.3235294    0.3202614    0.3048128
##     uhaching        mainz    frankfurt      cottbus    mgladbach
##    0.2941176    0.2843137    0.2830882    0.2823529    0.2764706
##    nuernberg    bielefeld     freiburg      fckoeln       aachen
##    0.2731092    0.2720588    0.2720588    0.2689076    0.2647059
##          ulm     duisburg  duesseldorf      stpauli    uerdingen
##    0.2647059    0.2598039    0.2500000    0.1960784    0.1470588
```

```
barplot(sort(propwin, decreasing = TRUE), col = "steelblue", las = 3)  # label is printed vertically us
```

IntroRExercisesWithSolutions_files/figure-latex/unnamed-chunk-155-1.pdf

---

Random numbers

This section is not only about random number generation but also includes exercises about the R-functions for standard distributions in statistics.

1. Let's consider a simple count data example.

- Let \$X∼ $N(0,1)$\$.*ComputetheprobabilityP*$(|X| > 3.5)$.

```
2 * (1 - pnorm(3.5))
```

```
## [1] 0.0004652582
```

- Generate $n = 10000$ random draws $X_1, \ldots, X_n$ from X and count the number of observations $|X_i| > 3.5$.

```
n <- 10000
X <- rnorm(n)
sum(abs(X) > 3.5)
```

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

```
sum(abs(X) > 3.5)/n
```

```
## [1] 4e-04
```

```
# the larger n the closer it is to the theoretical value of 0.0004652582
```

- Repeat drawing random samples $R = 5000$ times and write the counts into a vector $Z_1, \ldots, Z_{5000}$ of length 5000.

```
R <- 5000
Z <- rep(NA, R)
n <- 10000
for (i in 1:R) {
    X <- rnorm(n)
    Z[i] <- sum(abs(X) > 3.5)
}
```

- Tabulate Z and compare the frequencies with the probability function of a suitably fitted Poisson distribution.

```
table(Z)
```

```
## Z
##   0   1   2   3   4   5   6   7   8   9  10  11  12  13
##  57 245 466 787 943 867 661 448 280 125  77  31  10   3
```

```
t(data.frame(observ = 0:16 , prob = dpois(0:16, lambda=mean(Z))*R))
```

```
##             [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]
## observ  0.00000   1.0000   2.0000   3.0000   4.0000   5.0000   6.0000
## prob   46.56239 217.7444 509.1299 793.6317 927.8348 867.7853 676.3519
##             [,8]     [,9]    [,10]    [,11]    [,12]    [,13]    [,14]
## observ   7.0000   8.0000   9.0000 10.00000 11.00000 12.00000 13.00000
## prob   451.8417 264.1241 137.2389 64.17838 27.28398 10.63257  3.82478
##             [,15]      [,16]      [,17]
## observ 14.000000 15.0000000 16.0000000
## prob    1.277586  0.3983001  0.1164132
```

```
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```
truehist(Z, prob = F)
x <- seq(0, 16)
lines(x, dpois(x, lambda = mean(Z)) * R, lwd = 2)
```

IntroRExercisesWithSolutions_files/figure-latex/unnamed-chunk-160-1.pdf

2. Generate $n = 10000$ draws from a log normal distribution $X \sim e^Y$ where $Y \sim N(1, 0.5^2)$ (the parameters in the R function are `meanlog=1` and `sdlog=0.5`). Split the screen into two plotting areas using the command `par(mfrow=c(2,1))`. Plot the histograms of X and $\ln X$.

```r
n <- 10000
x <- rlnorm(n, meanlog = 1, sdlog = 0.5)
par(mfrow = c(2, 1))
truehist(x)
truehist(log(x))
```

IntroRExercisesWithSolutions_files/figure-latex/unnamed-chunk-161-1.pdf

3. Generate $n = 10000$ draws from $X \sim N(0, 1)$. Compute the cumulated means, i.e.

$$\bar{X}_j = \frac{1}{j}\sum_{i=1}^{j} X_i$$

for $j = 1, \ldots, n$ and plot them. Hint: Use the command `cumsum`.

```r
par(mfrow = c(1, 1))
n <- 10000
X <- rnorm(n)
m <- rep(NA, n)
for (i in 1:n) {
    m[i] <- cumsum(X)[i]/i
}
plot(m)
```

IntroRExercisesWithSolutions_files/figure-latex/unnamed-chunk-162-1.pdf

---

Simulations

1. This exercise illustrates the one-sample t-test.

- Generate $n = 10$ observations from $X \sim N(10, 3^2)$. Compute the mean and the standard deviation of $X_1, \ldots, X_{10}$.

```r
n <- 10
X <- rnorm(n, mean = 10, sd = 3)
m <- mean(X)
s <- sd(X)
print(m)
```

```
## [1] 10.08052
```

```r
print(s)
```

```
## [1] 3.344536
```

- The t-statistics of the hypothesis test $H_0 : \mu = 10$ against $H_1 : \mu \neq 10$ is

$$t = \sqrt{10}\frac{\bar{X} - 10}{sd}$$

where sd is the standard deviation (as computed by `sd`). Compute the t-statistic.

```r
t <- sqrt(n) * (m - 10)/s
print(t)
```

```
## [1] 0.07613527
```

- Create an empty vector Z of length $R = 5000$. Write a loop over $r = 1, \ldots, R$ and repeat the above steps for each r. Save the t-statistic at Z_r.

```r
R <- 5000
Z <- rep(NA, R)
for (r in 1:R) {
    X <- rnorm(n, mean = 10, sd = 3)
    m <- mean(X)
    s <- sd(X)
    Z[r] <- sqrt(n) * (m - 10)/s
}
```

- Plot the histogram of $Z_1, \ldots, Z_R$ and add the density function of the t_9-distribution.

```r
library(MASS)
truehist(Z, col = "lightblue")
x <- seq(-4, 4, by = 0.1)
lines(x, dt(x, df = 9), lwd = 2)
```

IntroRExercisesWithSolutions_files/figure-latex/unnamed-chunk-167-1.pdf

2. The classical central limit theorem states that the standardized sum of i.i.d. random variables with finite variance converges in distribution to the standard normal distribution $N(0, 1)$. This exercise illustrates the central limit theorem.

- Write a simulation that performs the following steps:
- Generate a random sample $X_1, \ldots, X_5$ of size $n = 5$ from the standard exponential distribution $Exp(1)$.
- Compute the sample sum.
- Repeat the steps $R = 10\,000$ times. For each replication, store the sum, e.g. into a vector Z.
- Plot the histogram of the sum and add the density function of $N(m, s^2)$ where m is the mean of Z and s is the standard deviation of Z.

```r
clt_exp <- function(n) {
  R <- 10000
  Z <- rep(NA, R)
```

```
  for (r in 1:R) {
      X <- rexp(n, rate = 1)
      Z[r] <- sum(X)
  }
  truehist(Z, col = "lightblue", main=paste("n =",n,sep=" "))
  coord <- par("usr")
  # par("usr") gives you a vector of the form c(x1, x2, y1, y2)
  # giving the extremes of the coordinates of the plotting region
  x <- seq(coord[1], coord[2], by = 0.1)
  lines(x, dnorm(x, mean = mean(Z), sd = sd(Z)), lwd = 2)
}
clt_exp(5)
```

IntroRExercisesWithSolutions_files/figure-latex/unnamed-chunk-168-1.pdf

- Increase the sample size n to $n = 50, 500, 5000$ and redo the exercise.

```
clt_exp(50)
```

IntroRExercisesWithSolutions_files/figure-latex/unnamed-chunk-169-1.pdf

```
clt_exp(500)
```

IntroRExercisesWithSolutions_files/figure-latex/unnamed-chunk-169-2.pdf

```
clt_exp(5000)
```

IntroRExercisesWithSolutions_files/figure-latex/unnamed-chunk-169-3.pdf

- Redo the exercise with other distributions than the exponential. Use the uniform distribution, the t-distribution with 3 degrees of freedom, the Bernoulli distribution (i.e. binomial with parameter size=1), and the Poisson distribution.

```
clt <- function(n, distrib, df=3, lambda=5, prob=0.6) {
  R <- 10000
  Z <- rep(NA, R)
  for (r in 1:R) {
```

```r
      if (distrib == 1){
        X <- runif(n)
        strdist <- "Uniform"
        }
      if (distrib == 2){
        X <- rt(n, df = df)
        strdist <- "Student''s t"
        }
      if (distrib == 3){
        X <- rbinom(n, size=1, prob=prob)
        strdist <- "Bernoulli"
        }
      if (distrib == 4){
        X <- rpois(n, lambda = lambda)
        strdist <- "Poisson"
        }
      Z[r] <- sum(X)
  }
  truehist(Z, col = "lightblue", xlab = strdist, main = paste("n =", n, sep = " "))
  coord <- par("usr")
  # par("usr") gives you a vector of the form c(x1, x2, y1, y2)
  # giving the extremes of the coordinates of the plotting region
  x <- seq(coord[1], coord[2], by = 0.1)
  lines(x, dnorm(x, mean = mean(Z), sd = sd(Z)), lwd = 2)
}
```

```r
par(mfrow = c(2,2))
for (n in c(5,50,500,5000)) {
  for (i in 1:4) {
    clt(n,i)
  }
}
```

IntroRExercisesWithSolutions_files/figure-latex/unnamed-chunk-171-1.pdf IntroRExercisesWithSolutions_files/figure-latex/unnamed-chunk-171-2.pdf IntroRExercisesWithSolutions_files/figure-latex/unnamed-chunk-171-3.pdf IntroRExercisesWithSolutions_files/figure-latex/unnamed-chunk-171-4.pdf

- The central limit theorem breaks down if the variance of the summands is infinite. Redo the exercise using a t-distribution with only 1.5 degrees of freedom.

```r
par(mfrow = c(1,1))
clt(500, 2, df = 1.5)
```

IntroRExercisesWithSolutions_files/figure-latex/unnamed-chunk-172-1.pdf

---

Linear regression

1. Load the Stata data set **wages.dta**. The variables are `earnings` (in Euro, 2009), `age`, `gender` (male=1, female=2), `education` (years of education), `hours` (hours worked during 2009), and `weight`.

```
library(foreign)
wages <- read.dta("data/wages.dta")
```

```
## Warning in read.dta("data/wages.dta"): value labels ('d1110211') for
## 'gender' are missing
```

```
head(wages)
```

```
##   gender age education hours earnings weight
## 1      2  51        18  1039     3900 737.73
## 2      2  59        18  2026    55550 459.86
## 3      1  22        13   312     2400 459.86
## 4      1  37        15  2338    54000 459.86
## 5      2  31        18  2858    28800 585.21
## 6      1  32        15  2078     9000 585.21
```

```
earnings <- wages$earnings
age <- wages$age
gender <- wages$gender
education <- wages$education
hours <- wages$hours
weight <- wages$weight
```

- Compute the (unweighted) wage equation

$$\ln \text{earnings}_i = \alpha + \beta_1 \text{age}_i + \beta_2 \text{age}_i^2 + \beta_3 \text{education}_i + \beta_4 \text{gender}_i + u_i,$$

  print the summary of the `lm`-object, and interpret the output.

```
regr <- lm(log(earnings) ~ age + I(age^2) + education + gender)
summary(regr)
```

```
##
## Call:
## lm(formula = log(earnings) ~ age + I(age^2) + education + gender)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.4564 -0.3610  0.1617  0.5563  3.6001
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.903e+00  1.163e-01   50.76   <2e-16 ***
## age          1.663e-01  5.213e-03   31.91   <2e-16 ***
## I(age^2)    -1.726e-03  6.001e-05  -28.76   <2e-16 ***
## education    1.067e-01  3.043e-03   35.08   <2e-16 ***
## gender      -7.237e-01  1.660e-02  -43.60   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8947 on 11643 degrees of freedom
```

```
## Multiple R-squared:  0.2888, Adjusted R-squared:  0.2886
## F-statistic:  1182 on 4 and 11643 DF,  p-value: < 2.2e-16
```

- Add an interaction term for `education` and `gender` to the regression.

```r
regr2 <- lm(log(earnings) ~ age + I(age^2) + education + gender + education:gender)
summary(regr2)
```

```
##
## Call:
## lm(formula = log(earnings) ~ age + I(age^2) + education + gender +
##     education:gender)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.4788 -0.3621  0.1572  0.5510  3.5676
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       6.610e+00  1.631e-01  40.527  < 2e-16 ***
## age               1.659e-01  5.205e-03  31.873  < 2e-16 ***
## I(age^2)         -1.717e-03  5.993e-05 -28.649  < 2e-16 ***
## education         5.139e-02  9.472e-03   5.426 5.88e-08 ***
## gender           -1.204e+00  7.954e-02 -15.134  < 2e-16 ***
## education:gender  3.763e-02  6.099e-03   6.170 7.05e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8933 on 11642 degrees of freedom
## Multiple R-squared:  0.2911, Adjusted R-squared:  0.2908
## F-statistic: 956.2 on 5 and 11642 DF,  p-value: < 2.2e-16
```

- Compute the weighted hourly wage equation

$$\ln \frac{\text{earnings}_i}{\text{hours}_i} = \alpha + \beta_1 \text{age}_i + \beta_2 \text{age}_i^2 + \beta_3 \text{education}_i + \beta_4 \text{gender}_i + u_i,$$

  print the summary of the `lm`-object, and interpret the output.

```r
regr3 <- lm(log(earnings/hours) ~ age + I(age^2) + education + gender, weights = weight)
summary(regr3)
```

```
##
## Call:
## lm(formula = log(earnings/hours) ~ age + I(age^2) + education +
##     gender, weights = weight)
##
## Weighted Residuals:
##     Min      1Q  Median      3Q     Max
## -540.17   -9.18    0.00   14.33  367.44
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.929e-01  9.095e-02   5.419 6.11e-08 ***
## age          5.823e-02  4.010e-03  14.524  < 2e-16 ***
## I(age^2)    -5.494e-04  4.596e-05 -11.954  < 2e-16 ***
## education    8.069e-02  2.368e-03  34.072  < 2e-16 ***
```

```
## gender      -2.728e-01  1.264e-02 -21.581  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 36.63 on 10128 degrees of freedom
## Multiple R-squared:  0.1757, Adjusted R-squared:  0.1753
## F-statistic: 539.5 on 4 and 10128 DF,  p-value: < 2.2e-16
```

```r
plot(regr3$residuals)
```

IntroRExercisesWithSolutions_files/figure-latex/unnamed-chunk-178-1.pdf

- Activate the packages `lmtest` and `sandwich`. Use the function `coeftest` to compute the heteroskedasticity robust standard errors (`vcov=vcovHC`) for the estimated coefficients.

```r
library(lmtest)
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
##
## Attaching package: 'lmtest'
```

```
## The following object is masked _by_ '.GlobalEnv':
##
##     wages
```

```r
library(sandwich)
```

```r
# Robust standard errors
coeftest(regr, vcov = vcovHC)
```

```
##
## t test of coefficients:
##
##                Estimate  Std. Error t value  Pr(>|t|)
## (Intercept)  5.9035e+00  1.4065e-01  41.974 < 2.2e-16 ***
## age          1.6634e-01  6.3384e-03  26.243 < 2.2e-16 ***
## I(age^2)    -1.7256e-03  7.2473e-05 -23.810 < 2.2e-16 ***
## education    1.0675e-01  2.9736e-03  35.898 < 2.2e-16 ***
## gender      -7.2369e-01  1.6586e-02 -43.633 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
coeftest(regr2, vcov = vcovHC)
```

```
##
## t test of coefficients:
##
```

```
##                      Estimate  Std. Error  t value  Pr(>|t|)    
## (Intercept)       6.6102e+00  1.7589e-01  37.5822 < 2.2e-16 ***
## age               1.6591e-01  6.3388e-03  26.1741 < 2.2e-16 ***
## I(age^2)         -1.7168e-03  7.2509e-05 -23.6775 < 2.2e-16 ***
## education         5.1394e-02  8.8301e-03   5.8203 6.028e-09 ***
## gender           -1.2037e+00  7.9682e-02 -15.1059 < 2.2e-16 ***
## education:gender  3.7634e-02  6.0272e-03   6.2441 4.410e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
coeftest(regr3, vcov = vcovHC)
```

```
## Warning in residuals^2/(1 - diaghat)^2: Länge des längeren Objektes
##       ist kein Vielfaches der Länge des kürzeren Objektes
```

```
## 
## t test of coefficients:
## 
##                 Estimate  Std. Error  t value  Pr(>|t|)    
## (Intercept)  4.9288e-01  1.3111e-01   3.7592 0.0001714 ***
## age          5.8234e-02  6.1032e-03   9.5417 < 2.2e-16 ***
## I(age^2)    -5.4943e-04  7.1504e-05  -7.6838 1.687e-14 ***
## education    8.0690e-02  3.4333e-03  23.5025 < 2.2e-16 ***
## gender      -2.7276e-01  1.6960e-02 -16.0825 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- Predict the hourly wage of a male person aged 60 years as a function of education (vary the years of education between 9 and 18). Set the option `se.fit=TRUE`. Inspect the object returned by the `predict` command. Plot the predicted values and add the ± 2 standard deviations confidence intervals.

```r
forecast <- predict(regr3, newdata = data.frame(education = seq(9, 18, by = 0.5), age = 60, gender = 1)
names(forecast)
```

```
## [1] "fit"            "se.fit"         "df"             "residual.scale"
```

```r
plot(seq(9, 18, by = 0.5), forecast$fit, type = "l", lwd = 2)
lines(seq(9, 18, by = 0.5), forecast$fit + 2 * forecast$se.fit, type = "l", col = "red", lwd = 1.5)
lines(seq(9, 18, by = 0.5), forecast$fit - 2 * forecast$se.fit, type = "l", col = "red", lwd = 1.5)
```

IntroRExercisesWithSolutions_files/figure-latex/unnamed-chunk-181-1.pdf

2. Load the data set **bsp4.txt**.

```r
bsp4 <- read.csv("data/bsp4.txt")
head(bsp4)
```

```
##        y     x
## 1  20.40 20.23
## 2 218.92 66.01
## 3 189.06 64.83
## 4 197.56 66.10
## 5 304.33 87.48
```

```
## 6 281.04 67.63
```

- Plot the scatter plot of y against x.

```
plot(bsp4$x, bsp4$y)
```

IntroRExercisesWithSolutions_files/figure-latex/unnamed-chunk-183-1.pdf

- Perform a simple linear regression of y on x and save the results as an `lm`-object `obj`. Add the regression line of y on x to the plot.

```
plot(bsp4$x, bsp4$y)
obj <- lm(bsp4$y ~ bsp4$x)
abline(obj)
```

IntroRExercisesWithSolutions_files/figure-latex/unnamed-chunk-184-1.pdf

- Extract the fitted values from `obj` and add them as red points to the plot (use the command `points`).

```
plot(bsp4$x, bsp4$y)
abline(obj)
points(bsp4$x, obj$fitted.values, col = "red")
```

IntroRExercisesWithSolutions_files/figure-latex/unnamed-chunk-185-1.pdf

- Extract the residuals of the regression and calculate the sum of the squared residuals, $SSR = \sum_{i=1}^{100} \hat{u}_i^2$

```
ssr <- sum((obj$residuals)^2)
print(ssr)
```

```
## [1] 223587.4
```

- Compute the total sum of squares and the explained sum of squares,

$$TSS = \sum_{i=1}^{100} (y_i - \bar{y})^2$$
$$ESS = \sum_{i=1}^{100} (\hat{y}_i - \bar{y})^2$$

and show that $ESS + SSR = TSS$.

```r
tss <- sum((bsp4$y - mean(bsp4$y))^2)
ess <- sum((obj$fitted.values - mean(bsp4$y))^2)
ess + ssr - tss # this is numerically zero
```

```
## [1] 1.164153e-10
```

```r
round(ess + ssr) == round(tss)
```

```
## [1] TRUE
```