Exercise set 1 – Peter Tempfli

tempflip@gmail.com (mailto:tempflip@gmail.com)

1.

The initial state

```
vec1<-c(0,2,3,0,2,11,0,7,NA)
```

Use indexing to remove the NA.

```
vec1 <- vec1[!is.na(vec1)]
vec1
```

```
## [1]  0  2  3  0  2 11  0  7
```

Make a logical vector indicating the elements equal to zero

```
myZeroPositions <- vec1==0
myZeroPositions
```

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

Use the logical vector to pick out the zero values and store them in a vector called ‘zeros’.

```
zeros <- vec1[myZeroPositions]
zeros
```

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

Check how many zeros you have in vec1 by taking the length of the vector zeros (use function length()).

```
length(zeros)
```

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

2.

```
w <- c(109, 112, 115, 121, 128, 132, 135, 140, 148)
m <- c(120, 122, 124, 130, 136, 140, 143, 150, 155)
df <- data.frame(M = m, W = w)
row.names(df) <- seq(2003, 2011)
df <- t(df)

df
```

```
##   2003 2004 2005 2006 2007 2008 2009 2010 2011
## M  120  122  124  130  136  140  143  150  155
## W  109  112  115  121  128  132  135  140  148
```

Construct a data frame from the table below, including the three variables W (average wage/h), YEAR (including the years for each observation) and Gender (including characters for Women/Men).

Explanation:

I use the `melt` function in order to transform the data to long-format (= every observation has a separate row in the DF). Than I set manually the column names.

```
library(reshape2)
each_observastion <- melt(df)
names(each_observastion) <- c('gender', 'year', 'wage')
each_observastion
```

```
##    gender year wage
## 1       M 2003  120
## 2       W 2003  109
## 3       M 2004  122
## 4       W 2004  112
## 5       M 2005  124
## 6       W 2005  115
## 7       M 2006  130
## 8       W 2006  121
## 9       M 2007  136
## 10      W 2007  128
## 11      M 2008  140
## 12      W 2008  132
## 13      M 2009  143
## 14      W 2009  135
## 15      M 2010  150
## 16      W 2010  140
## 17      M 2011  155
## 18      W 2011  148
```

```
setwd('~/prog/r')
write.csv(each_observastion, file='mydata.csv')
```

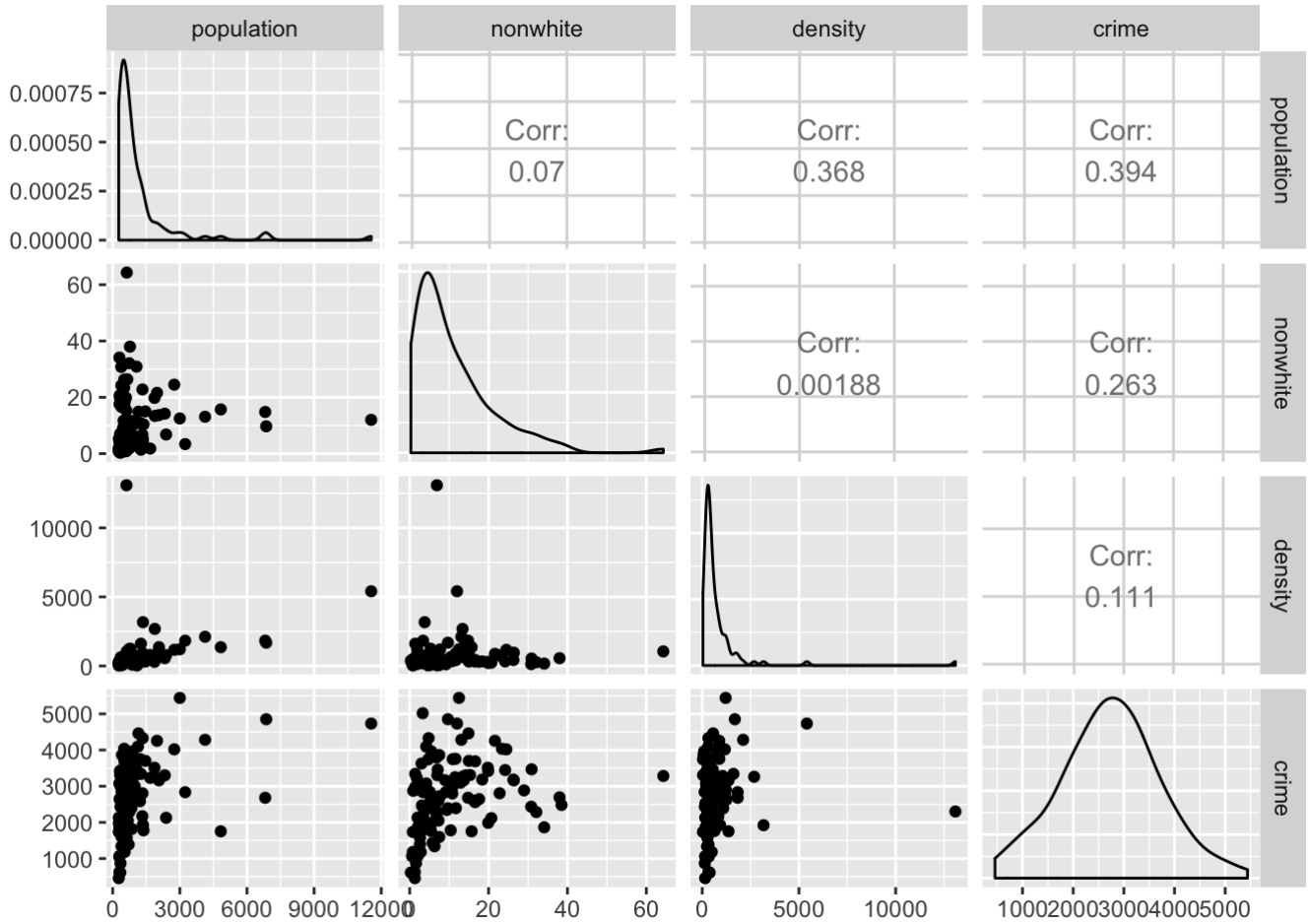
3.

```
library(openxlsx)
fr <- read.xlsx('~/Desktop/R/Freedman.xlsx')
fr$population <- as.numeric(fr$population)
fr$nonwhite <- as.numeric(fr$nonwhite)
fr$density <- as.numeric(fr$density)

summary(fr)
```

```
##      City             population        nonwhite        density
##  Length:108         Min.   :  270.0   Min.   : 0.30   Min.   :   37.0
##  Class :character   1st Qu.:  398.5   1st Qu.: 3.40   1st Qu.:  265.0
##  Mode  :character   Median :  675.0   Median : 7.20   Median :  405.0
##                     Mean   : 1142.1   Mean   :10.89   Mean   :  768.2
##                     3rd Qu.: 1185.5   3rd Qu.:14.93   3rd Qu.:  776.5
##                     Max.   :11551.0   Max.   :64.30   Max.   :13087.0
##                     NA's   :9                         NA's   :9
##      crime
##  Min.   : 458
##  1st Qu.:2123
##  Median :2716
##  Mean   :2736
##  3rd Qu.:3307
##  Max.   :5441
##
```

Some basic visualization of the data



4.

```
library(car)
```

```
## Loading required package: carData
```

```
pr <- Prestige
```

Select a subset of the data for occupations with more than 50% women and call the subset 'sub Prestige'.

```
sub_Prestige <- pr[pr$women>50,]
sub_Prestige
```

```
##                         education income women prestige census type
## economists                  14.44   8049 57.31     62.2   2311 prof
## social.workers              14.21   6336 54.77     55.1   2331 prof
## librarians                  14.15   6112 77.10     58.1   2351 prof
## primary.school.teachers     13.62   5648 83.78     59.6   2731 prof
## nurses                      12.46   4614 96.12     64.7   3131 prof
## nursing.aides                9.45   3485 76.14     34.9   3135   bc
## physio.therapsts            13.62   5092 82.66     72.1   3137 prof
## medical.technicians         12.79   5180 76.04     67.5   3156   wc
## secretaries                 11.59   4036 97.51     46.0   4111   wc
## typists                     11.49   3148 95.97     41.9   4113   wc
## bookkeepers                 11.32   4348 68.24     49.4   4131   wc
## tellers.cashiers            10.64   2448 91.76     42.3   4133   wc
## computer.operators          11.36   4330 75.92     47.7   4143   wc
## file.clerks                 12.09   3016 83.19     32.7   4161   wc
## receptionsts                11.04   2901 92.86     38.7   4171   wc
## postal.clerks               10.07   3739 52.27     37.2   4173   wc
## telephone.operators         10.51   3161 96.14     38.1   4175   wc
## claim.adjustors             11.13   5052 56.10     51.1   4192   wc
## office.clerks               11.00   4075 63.23     35.6   4197   wc
## sales.clerks                10.05   2594 67.82     26.5   5137   wc
## cooks                        7.74   3116 52.00     29.7   6121   bc
## babysitters                  9.46    611 96.53     25.9   6147 <NA>
## launderers                   7.33   3000 69.31     20.8   6162   bc
## canners                      7.42   1890 72.24     23.2   8221   bc
## electronic.workers           8.76   3942 74.54     50.8   8534   bc
## sewing.mach.operators        6.38   2847 90.67     28.2   8563   bc
## bookbinders                  8.55   3617 70.87     35.2   9517   bc
```

Use the subset and compute the average prestige score.

```
mean(sub_Prestige$prestige)
```

```
## [1] 43.52593
```

Now compute the average prestige score for occupations with less than 50% women

```
mean( subset(pr, women < 50)$prestige )
```

```
## [1] 48.024
```

For this question use the complete Prestige data again. Make a for-loop to compute the average prestige score for the three dif- ferent types of occupations. Automatically store the three means in a vector. The loop should be general, i.e. even if the types of occupations were 100000 one should be able to use your loop.

Explanation:

The function calculates the means of a property, grouped by another property. In the example above calculates the ‘prestige’ mean value, grouped by ‘type’.

The group-by column must be a factor.

```
mean_by_key <- function(df, valname, keyname) {

  counters <- rep(0, length(levels(df[,keyname]) ))
  sums <- rep(0, length(levels(df[,keyname]) ))

  for (i in seq(0:nrow(df))) {

    val <- df[i,valname]
    key <- df[i, keyname]
    counters[as.numeric(key)] <- counters[as.numeric(key)] + 1
    sums[as.numeric(key)] <- sums[as.numeric(key)] + val

  }

  r <- data.frame(sums/counters)
  rownames(r) <- levels(df[,keyname])
  colnames(r) <- c(valname)
  return(r)
}
```

Demo

```
pr <- Prestige
means_for_types <- mean_by_key(pr, 'prestige', 'type') ## column name and the grouped
 by are dynamic
means_for_types
```

```
##      prestige
## bc   35.52727
## prof 67.84839
## wc   42.24348
```

We can check if our function output is correct, using the built-in `by` function.

```
by(pr$prestige, pr$type, mean)
```

```
## pr$type: bc
## [1] 35.52727
## ------------------------------------------------------------
## pr$type: prof
## [1] 67.84839
## ------------------------------------------------------------
## pr$type: wc
## [1] 42.24348
```