Notebook 1: Exploratory data analysis

prerequisite: swirl exploratory data analysis

In this notebook, numerical and visual explorative data analysis is demonstrated. We will focus on visualizations that are an intermediate step in the analysis of data. For plotting, the 'ggplot2' plotting system is used, but occasionally we will switch to the base 'graphics' package that is part of R.

Before running the scripts, we need you to set the directory in which the associated datasets are situated. After completing the path, hit the key combination <Ctrl><Shift><Enter> inside the code block to execute it.

```
rm(list = ls(all = TRUE)) # Clean up workspace
# type the full path between the quotes, replacing the placeholder:
path <- "C:/Users/koenn/OneDrive/School/DAV/Notebook 1"
```

First, we load the data (a binary file).

```
load(file.path(path,"ads_prevENGs.RData"))
dat <- dat[1:4000,]
# It's always a good idea to view the first few rows of your dataset to check
# for loading errors or behaviour you don't expect:
head(dat)
```

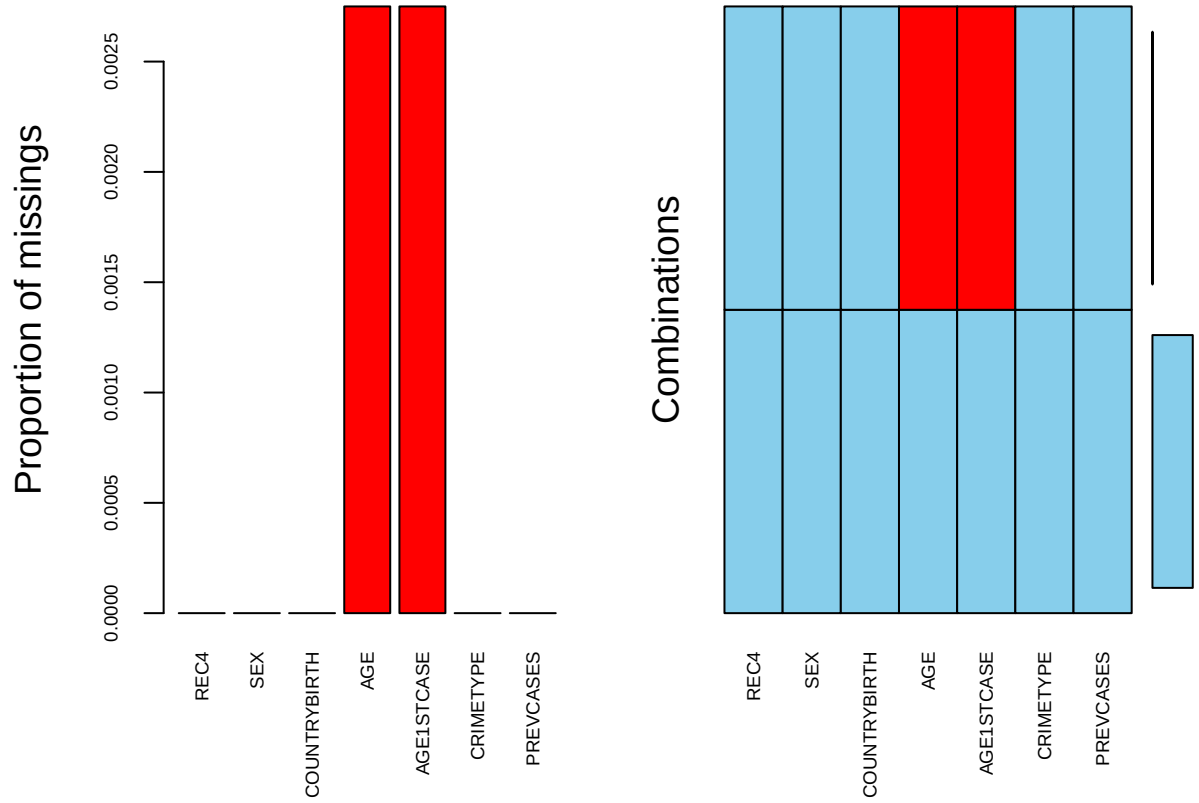
```
##   REC4   SEX  COUNTRYBIRTH AGE AGE1STCASE                 CRIMETYPE
## 1    0   man Other Western  23         23                     drugs
## 2    1   man   Netherlands  25         19                  violence
## 3    0   man   Netherlands  35         35                  violence
## 4    1   man       Surinam  44         17 property without violence
## 5    0 vrouw   Netherlands  56         56 property without violence
## 6    0   man   Netherlands  38         38                  violence
##   PREVCASES
## 1         0
## 2         3
## 3         0
## 4         8
## 5         0
## 6         0
```

The data file is now stored in memory in an object named 'dat'. This is a data.frame object (i.e. an object of class 'data.frame'). This can be checked by typing class(dat) in the console. A data.frame is a matrix of that contains units of observation (i.e. *cases*) in the rows and variables ('features' of the observations) in the columns. It is the central data structure for R.

The actual data consist of adult convicts in the Netherlands, convicted in the year 2010. The variables include sex (SEX), country of birth (COUNTRYBIRTH), age in years (AGE), age at first conviction in years (AGE1STCASE), crime type of the index case (CRIMETYPE), number of previous convictions (PREVCASES) and an indicator whether convicts had a new penal case within four years (REC4).

At first, we inspect the (co)occurrence of missing values in the data set.

```
library(VIM)
# aggr is a function for calculation (and/or plotting) of missing values.
miss <- aggr(dat, plot = FALSE)
plot(miss, cex.axis = 0.55) # cex.axis determines the point size of the axis label font
```



Apparently, there are only very few missing data. On the left graph, we can see that only the variables AGE and AGE1STCASE have missing instances in a very small proportion of the data. In the right graph, we find out that AGE and AGE1STCASE are always simultaneously missing. This has to do with missing dates of birth, on which both variables are derived.

Because of their rarity in these data, we can safely remove cases with missing values.

```
dat <- na.omit(dat)
```

We are now ready to perform exploratory data analysis. First, we will try the numerical approach.

As a first step, we look at the summary statistics. With summary(), we get the five number summary (i.e. the minimum, 1st quartile, median, 3rd quartile and the maximum) and the mean for continuous variables. For categorical variables we obtain frequency distributions.

```
summary(dat)
```

```
##  REC4        SEX                    COUNTRYBIRTH       AGE
##  0:2568   man  :3195   Netherlands      :2764   Min.   : 12.00
##  1:1421   vrouw: 794   Morocco          : 123   1st Qu.: 23.00
##                        Dutch Antilles   : 126   Median : 32.00
##                        Surinam          : 169   Mean   : 34.83
##                        Turkey           : 109   3rd Qu.: 44.00
##                        Other Western    : 397   Max.   :104.00
##                        Other non-Western: 301
##    AGE1STCASE                          CRIMETYPE      PREVCASES
##  Min.   :11.00   traffic                    :1117   Min.   :  0.000
##  1st Qu.:18.00   property without violence: 983   1st Qu.:  0.000
##  Median :22.00   misc                       : 643   Median :  1.000
```

```
##  Mean   :27.11   violence                : 571   Mean   :  4.409
##  3rd Qu.:33.00   public order            : 376   3rd Qu.:  4.000
##  Max.   :91.00   drugs                   : 219   Max.   :190.000
##                  (Other)                 :  80
```

Categorical variables

When summarizing the data as above, the maximum number of categories that is summarized for categorical variables is seven. As you can see with crime type, all remaining categories are lumped together in '(Other)'.

We can generate a table for a single variable using the `table` command. The resulting table can be converted into a table of proportions with the `prop.table` command, in order to get an idea of the proportional occurrence of crime types.

```
tab_crimetype <- table(dat$CRIMETYPE)
tab_crimetype
```

```
##
##                  violence                    sexual
##                       571                        26
##    property with violence property without violence
##                        54                       983
##              public order                     drugs
##                       376                       219
##                   traffic                      misc
##                      1117                       643
```

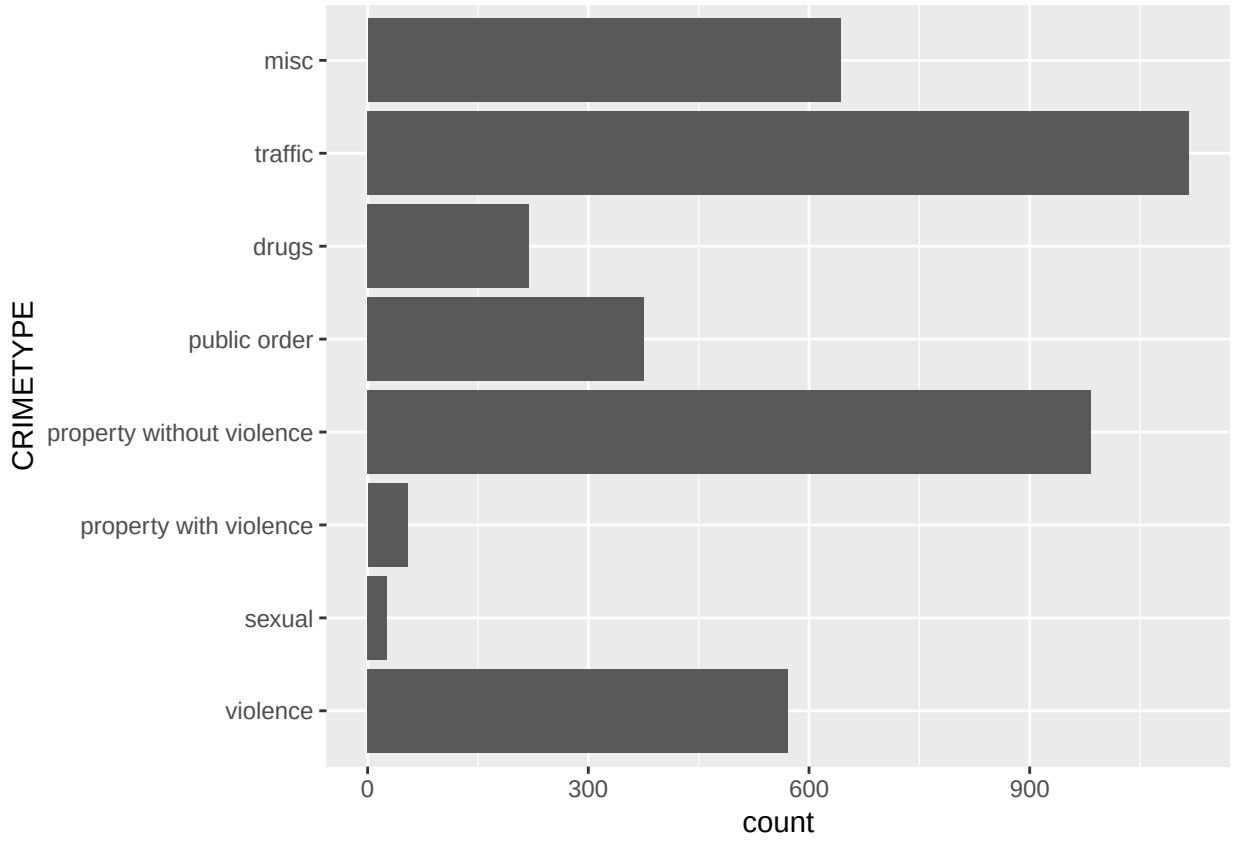
```
prop.table(tab_crimetype)
```

```
##
##                  violence                    sexual
##               0.143143645               0.006517924
##    property with violence property without violence
##               0.013537227               0.246427676
##              public order                     drugs
##               0.094259213               0.054900978
##                   traffic                      misc
##               0.280020055               0.161193282
```

The frequency table shows us that traffic and property offenses without violence are by far the largest categories. The proportion table shows that their relative frequencies are 0.28 and 0.24.

Categorical variables can be visualized by using bar charts. Using `ggplot`, we can use the `geom_bar` for this purpose. It automatically converts a categorical variable into counts that can indicate bar length.

```
library(ggplot2)
ggplot(data = dat) +
  geom_bar(mapping = aes(x = CRIMETYPE)) +
  coord_flip() # flip the coordinates for legibility of categories
```



A cross table can display the simultaneous distribution of counts over two or more variables. This is also done with the `table` command.

This is the cross table of country of birth and crime type and its corresponding proportion table. With more than one variable we need to choose whether we want column (`margin = 2`) or row (`margin = 1`) proportions, depending on what one is interested in.

```
tab_crimetype_countrybirth <- table(dat$CRIMETYPE, dat$COUNTRYBIRTH)
tab_crimetype_countrybirth
```

```
##
##                            Netherlands Morocco Dutch Antilles Surinam
##   violence                         392      15              27      25
##   sexual                            23       0               1       0
##   property with violence            27       4               6       7
##   property without violence        602      35              43      54
##   public order                     278      11               9      12
##   drugs                            126      10              10      15
##   traffic                          857      26              21      42
##   misc                             459      22               9      14
##
##                            Turkey Other Western Other non-Western
##   violence                     23            46                43
##   sexual                        0             2                 0
##   property with violence        1             5                 4
##   property without violence    16           138                95
##   public order                 10            31                25
```

```
##   drugs                          5            33                20
##   traffic                       26            81                64
##   misc                          28            61                50
```

```
cat('Column proportions') # prop.table where columns sum to 1 (margin = 2)
```

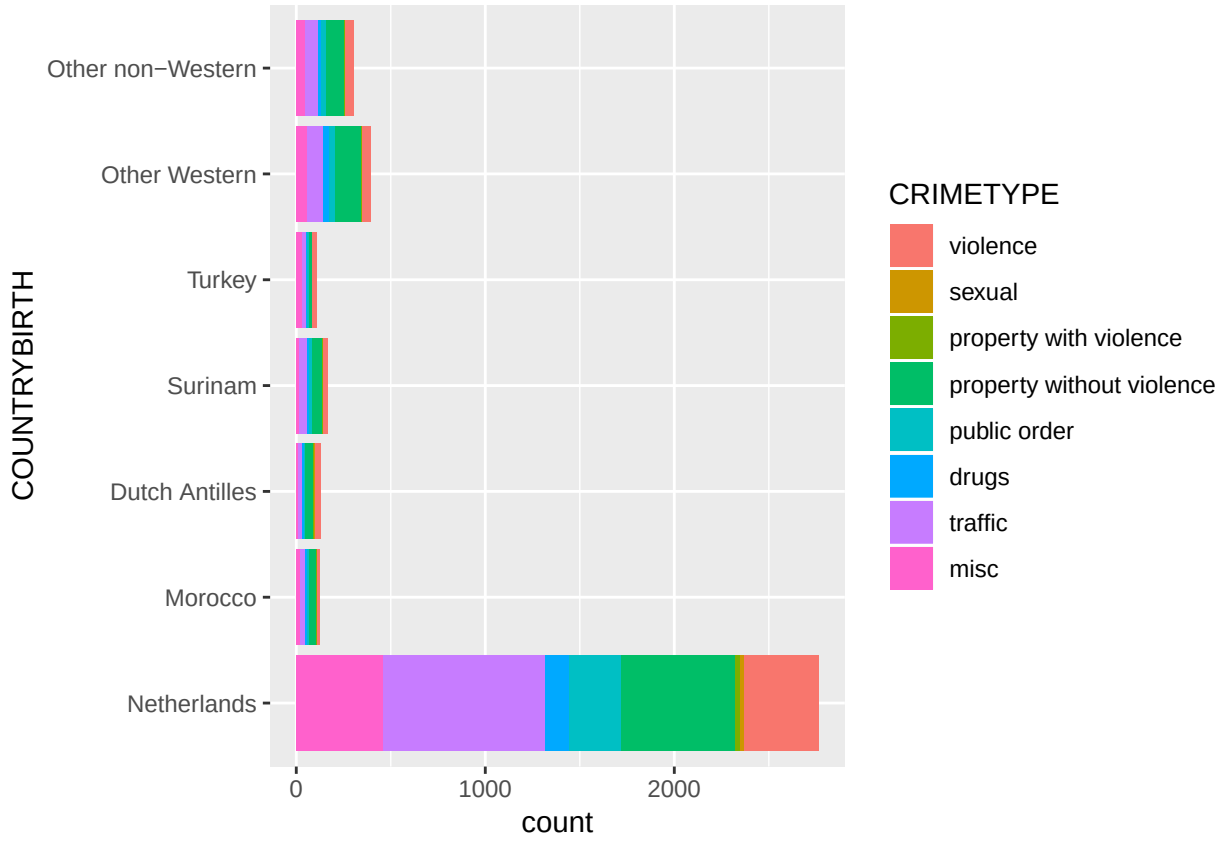
```
## Column proportions
```

```
prop_crimetype_countrybirth <- prop.table(tab_crimetype_countrybirth, margin = 2)
round(prop_crimetype_countrybirth, 2) # round to 2 decimals
```

```
##
##                              Netherlands Morocco Dutch Antilles Surinam
##   violence                          0.14    0.12           0.21    0.15
##   sexual                            0.01    0.00           0.01    0.00
##   property with violence            0.01    0.03           0.05    0.04
##   property without violence         0.22    0.28           0.34    0.32
##   public order                      0.10    0.09           0.07    0.07
##   drugs                             0.05    0.08           0.08    0.09
##   traffic                           0.31    0.21           0.17    0.25
##   misc                              0.17    0.18           0.07    0.08
##
##                              Turkey Other Western Other non-Western
##   violence                     0.21          0.12              0.14
##   sexual                       0.00          0.01              0.00
##   property with violence       0.01          0.01              0.01
##   property without violence    0.15          0.35              0.32
##   public order                 0.09          0.08              0.08
##   drugs                        0.05          0.08              0.07
##   traffic                      0.24          0.20              0.21
##   misc                         0.26          0.15              0.17
```

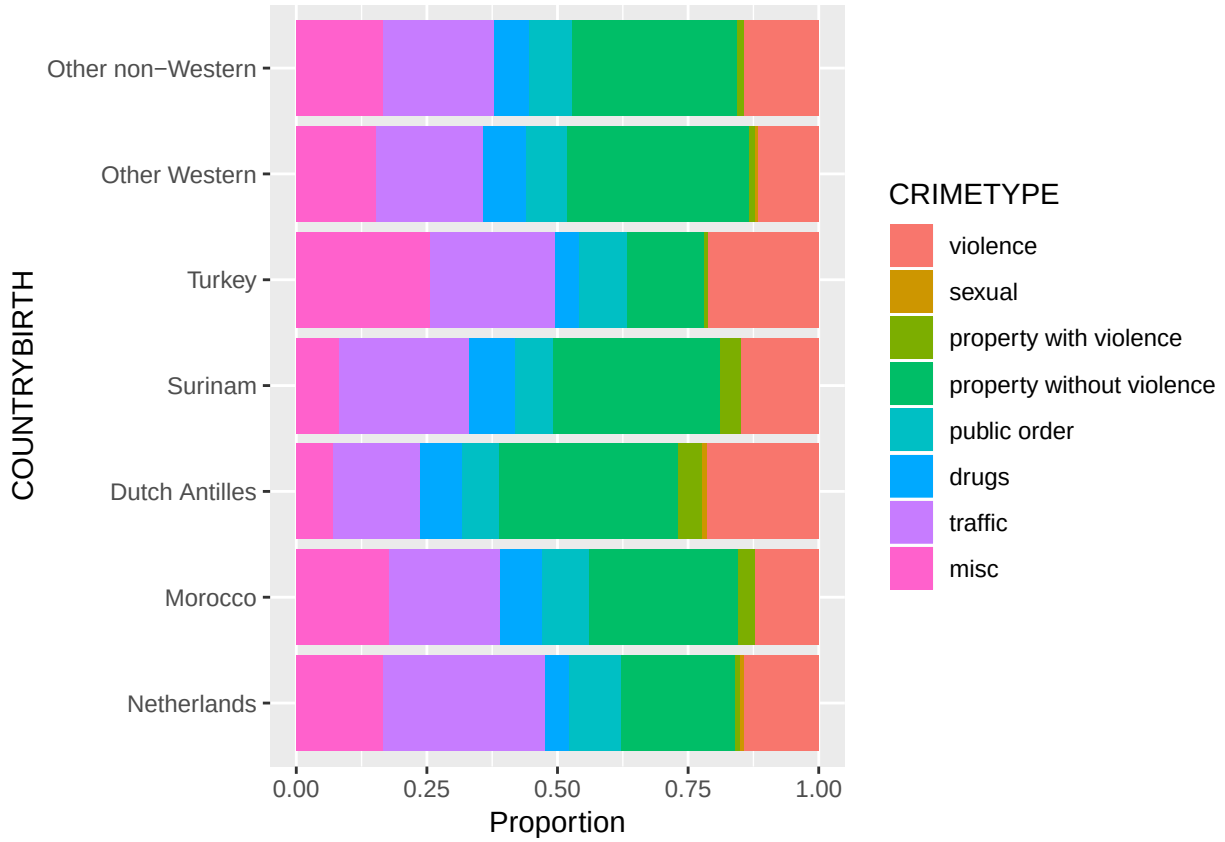
The larger a table becomes, the harder it becomes to discern its results or see patterns. Therefore, in this case, a combined bar plot might be more appropriate.

```
ggplot(dat, aes(COUNTRYBIRTH)) +
  geom_bar(aes(fill = CRIMETYPE)) +
  coord_flip()
```



Although this plot gives a good indication of relative counts of country of birth, it is hard to compare the proportions. Let's use a 100% stacked bar chart by adding `position = fill` to the `geom_bar` arguments. Then we should also change the axis label to "proportion" instead of the default "count".

```
ggplot(dat, aes(COUNTRYBIRTH)) +
  geom_bar(aes(fill = CRIMETYPE), position = "fill") +
  ylab("Proportion") + # ylab because we flip the coordinates on the next line
  coord_flip()
```



Perpetrators born in the Netherlands are apparently relatively more into committing traffic-related offences, whereas those born in Turkey are more inclined to commit offences in the category "miscellaneous". Note that this is all relative, as in absolute terms many more "misc" offences come from Netherlands-born than from Turkey-born perpetrators.

Continous variables

The strength and direction of the associations between continuous variables can be quickly summarized using a correlation matrix.

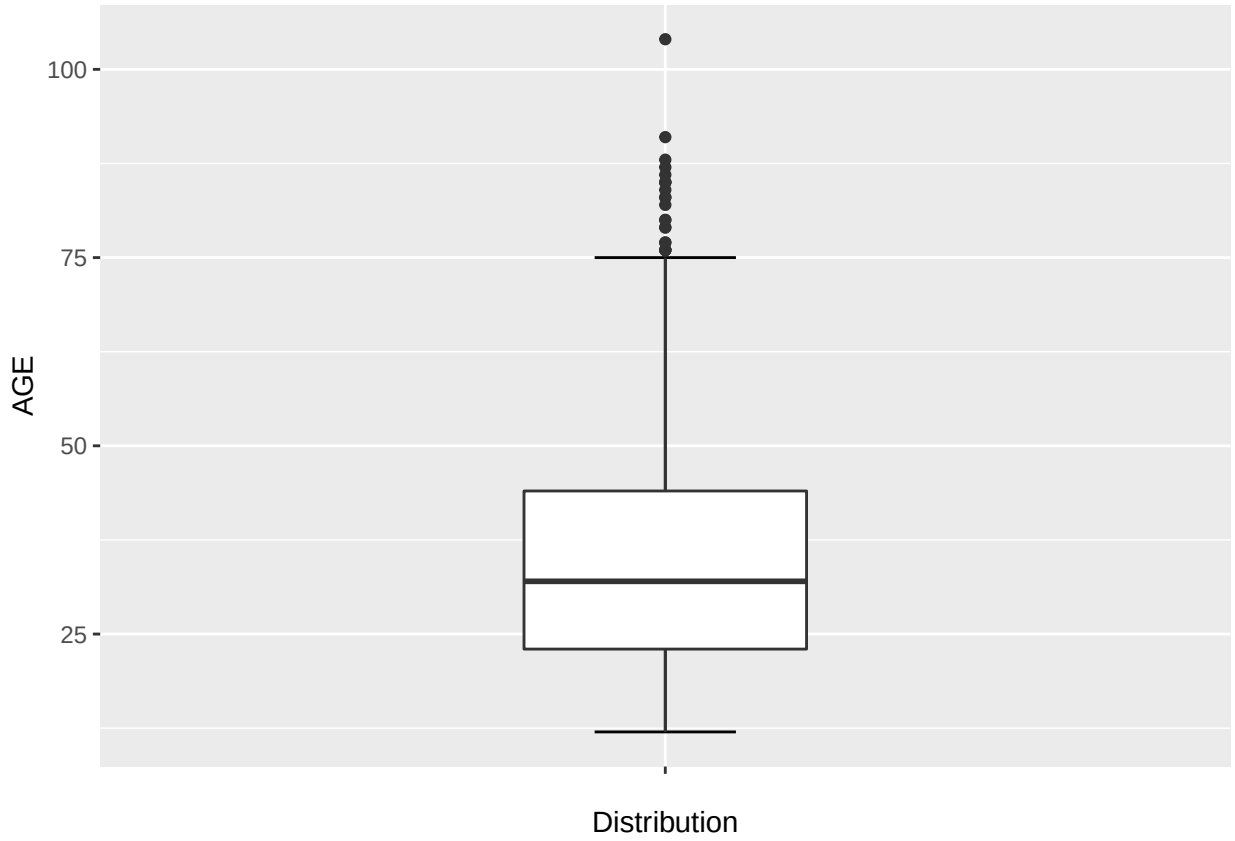
```
cor(dat[,c("AGE","AGE1STCASE","PREVCASES")])
```

```
##                    AGE AGE1STCASE  PREVCASES
## AGE         1.0000000  0.7246927  0.1346413
## AGE1STCASE 0.7246927  1.0000000 -0.2403469
## PREVCASES  0.1346413 -0.2403469  1.0000000
```

For instance, now we know that the higher the age at the first penal case was, the lower the number of previous cases.

Boxplots (a.k.a. Box and whisker plots) can give a quick impression of the distribution of continuous (numeric) variables. It plots the three quartiles (median, 1st quartile and 3rd quartile, see the five number summary above).

```
ggplot(dat, mapping = aes(y = AGE, x = "")) +
  stat_boxplot(geom = 'errorbar', width = 0.15) + # the horizontal whisker lines
  geom_boxplot(width = 0.3) + # the box
  xlab("Distribution")
```

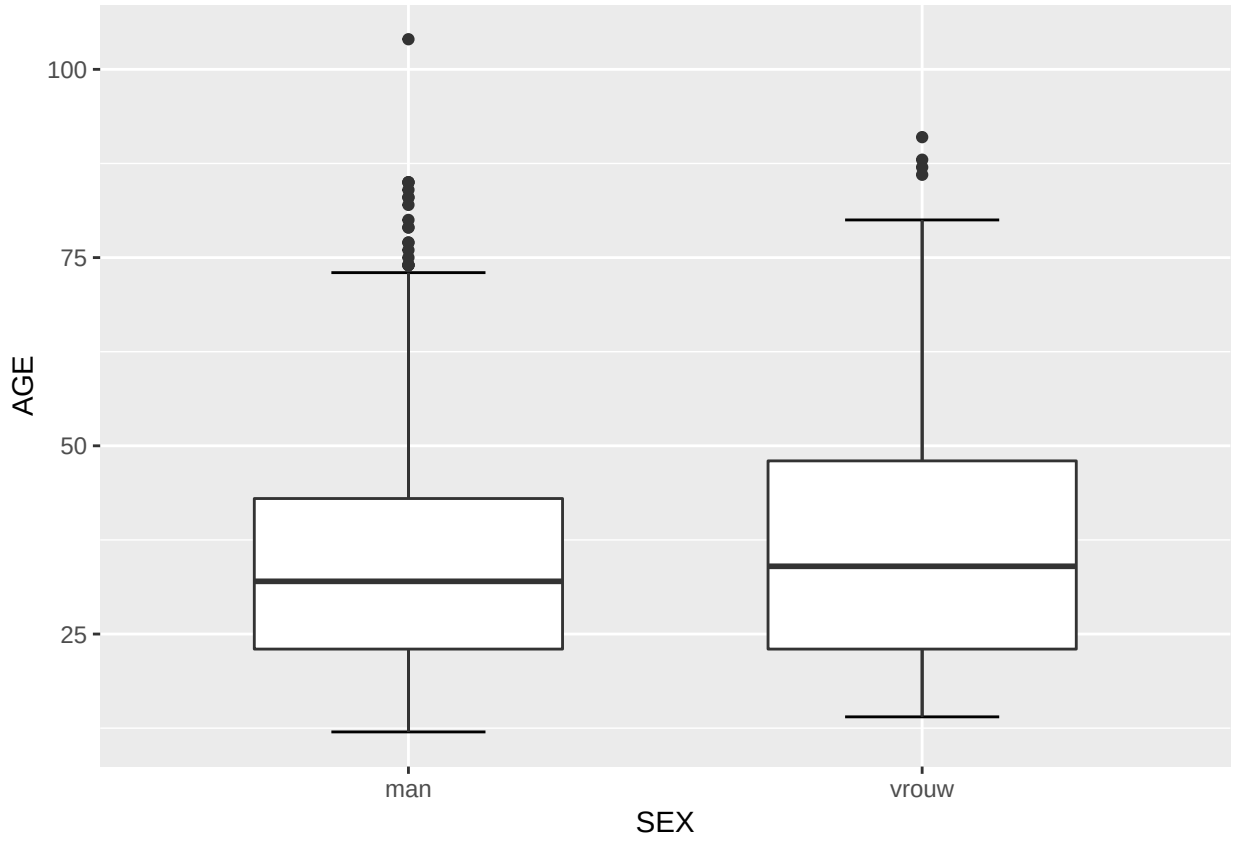


The definition of the 'whiskers' of the box plot (the outer edges) can vary across implementations. In this implementation, they are the first data values that still lie within the $1.5 \times IQR$, where IQR is the interquartile range, i.e. the 3rd minus the 1st quartile (or the whiskers are the minimum or maximum). Observations outside this interval are plotted as 'outliers' (i.e. marked as observations with extreme values).

This particular box plot shows that age has a very skew distribution. Specifically, the distribution has a positive skew or is said to be right-skewed.

We can generate separate boxplots by mapping a grouping variable on the x-axis, for exanple grouping by sex:

```
ggplot(dat, mapping = aes(y = AGE, x = SEX)) +
  stat_boxplot(geom = 'errorbar', width = 0.3) + # the whiskers
  geom_boxplot(width = 0.6) # the box
```



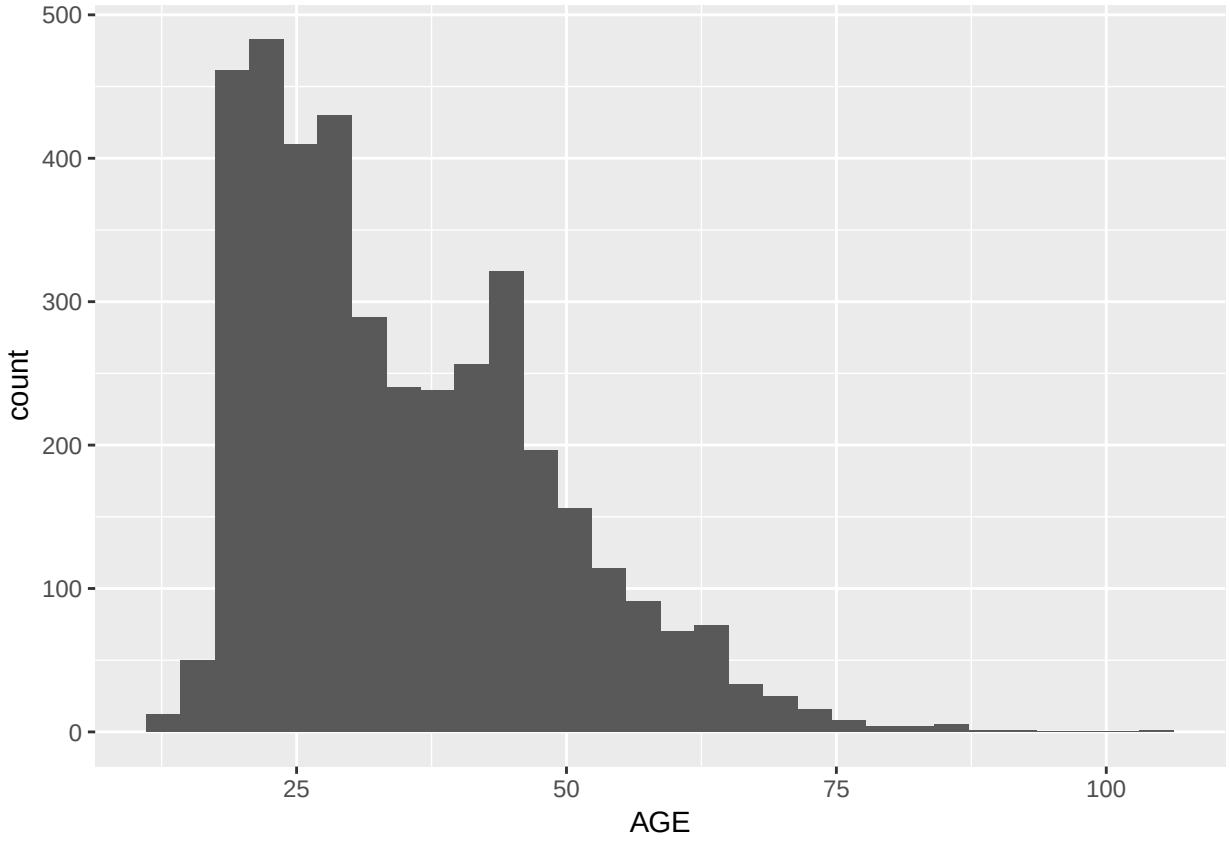
Apparently, female perpetrators are slightly older, but the age distributions are basically identical.

One disadvantage of boxplots to be aware of, is that they cannot visualize that a distribution has more than one peak (i.e. multimodality).

For that, we need other techniques.

Another tool to visualize distributions of continuous variables is the histogram. It applies binning in a fixed number of bins, decided by the "bins" argument.

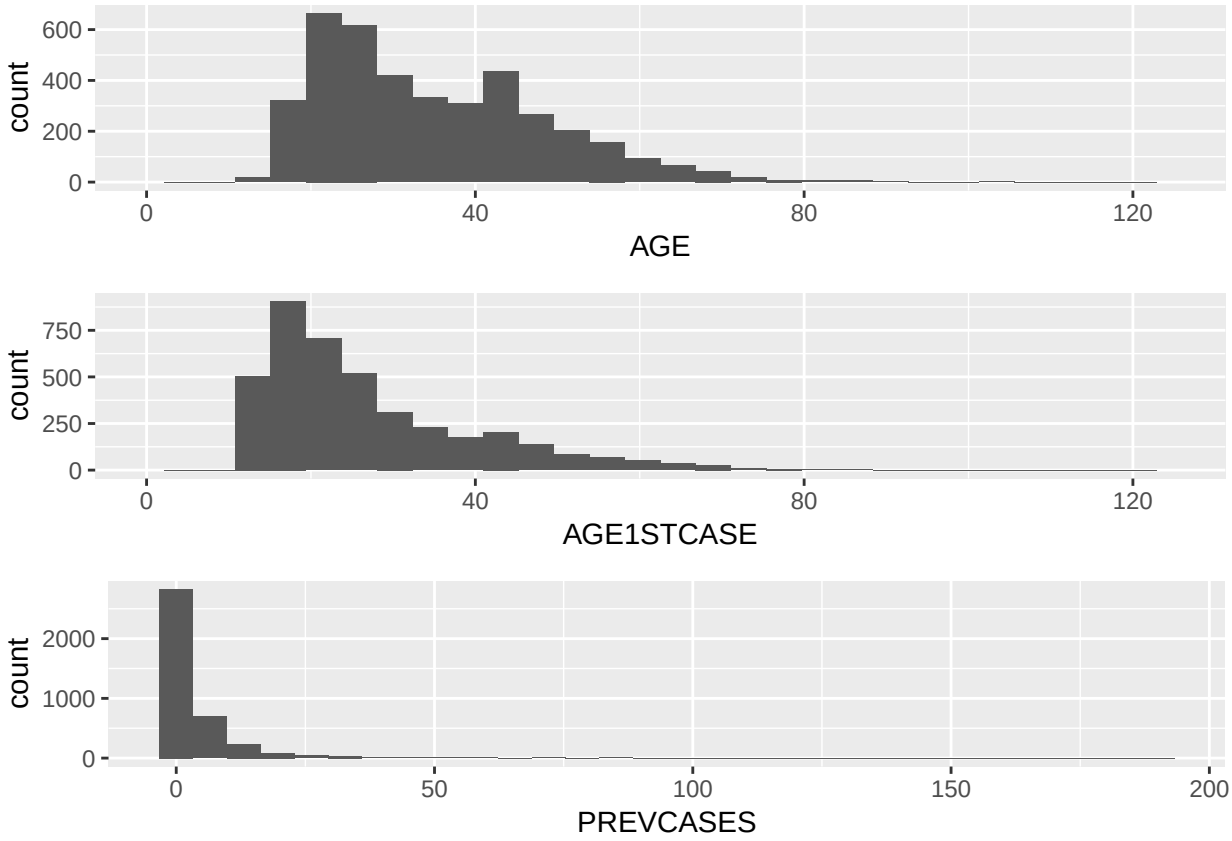
```
ggplot(dat) + geom_histogram(mapping = aes(x = AGE), bins = 30)
```



Now, the second peak around age 40 becomes apparent.

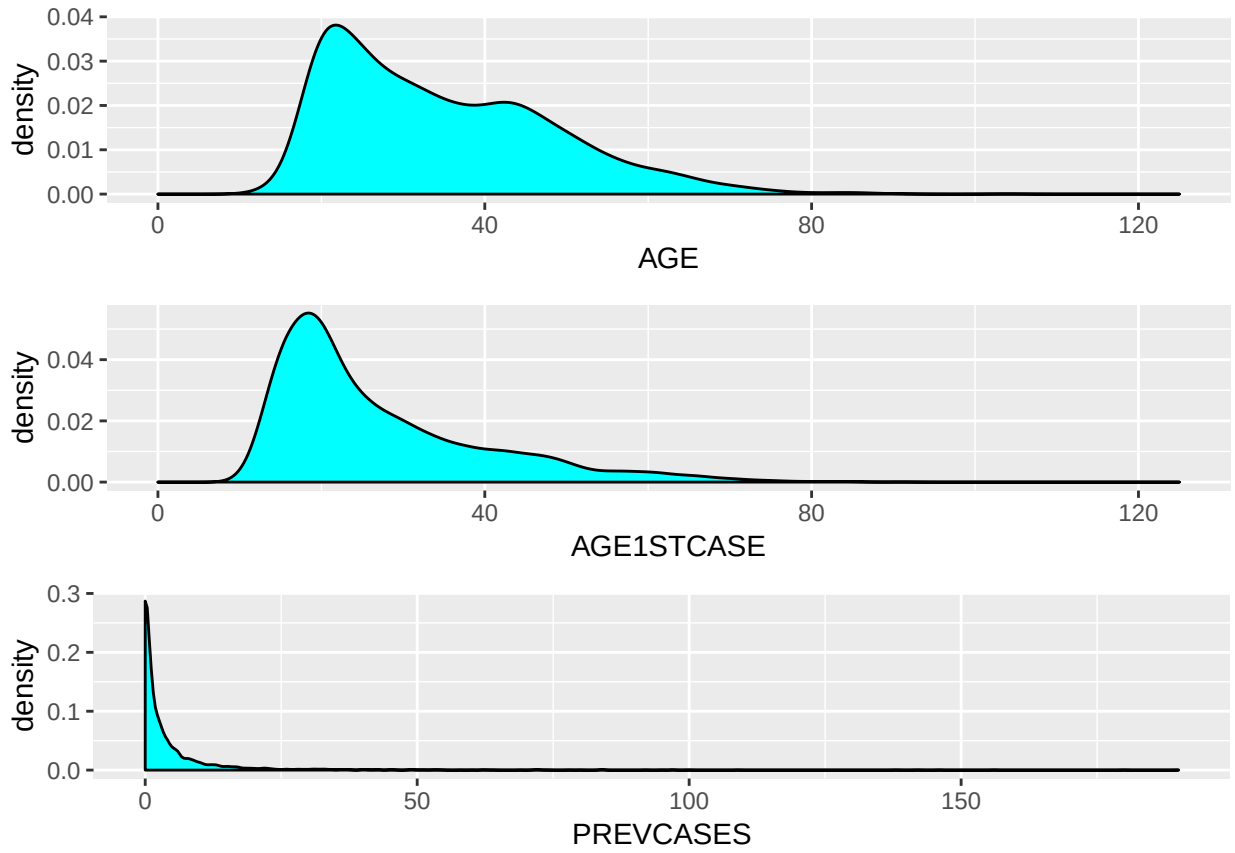
With the package `gridExtra` we can render multiple plots on one page:

```
library(gridExtra)
p1 <- ggplot(dat) +
  geom_histogram(mapping = aes(x = AGE), bins = 30) +
  xlim(0, 125)

p2 <- ggplot(dat) +
  geom_histogram(mapping = aes(x = AGE1STCASE), bins = 30) +
  xlim(0, 125)

p3 <- ggplot(dat) +
  geom_histogram(mapping = aes(x = PREVCASES), bins = 30)

grid.arrange(p1, p2, p3, nrow = 3)
```



A continuous, smooth version of the histogram is the (kernel) density plot. These are the plotted densities of the previous three histogram variables.

```
p1 <- ggplot(dat) +
  geom_density(mapping = aes(x = AGE), fill = "cyan") +
  xlim(0, 125)

p2 <- ggplot(dat) +
  geom_density(mapping = aes(x = AGE1STCASE), fill = "cyan") +
  xlim(0, 125)

p3 <- ggplot(dat) +
  geom_density(mapping = aes(x = PREVCASES), fill = "cyan")

grid.arrange(p1, p2, p3, nrow = 3)
```



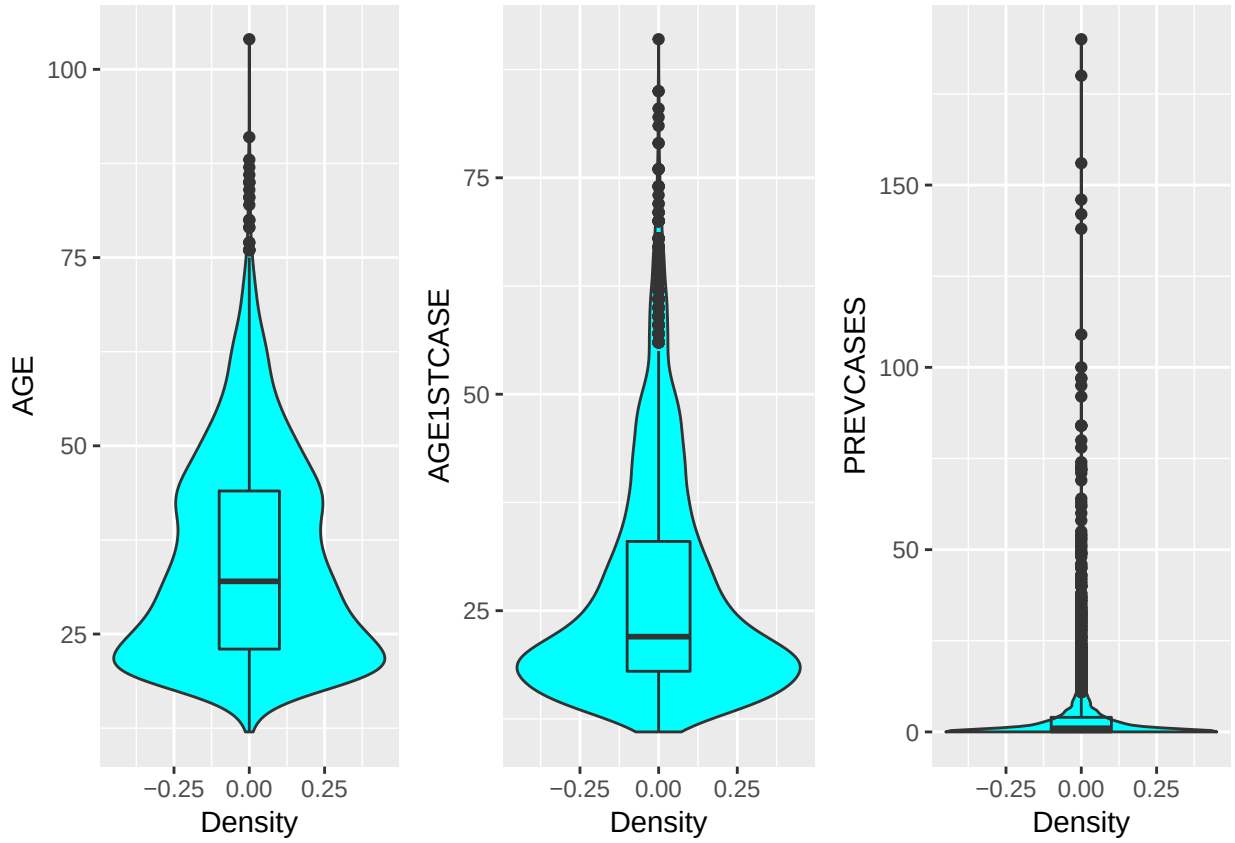

```
# try to change the kernel density bandwidth by adding the argument `bw = 1` to
# the geom_density function in plot 1. Play around with different bandwidths!
```

The multimodality of AGE again becomes apparent.

The violin plot combines elements of the box plot and the density plot. It contains a box plot in the middle and plots the density on both sides.

This is the violin plot for all continuous variables in the data set.

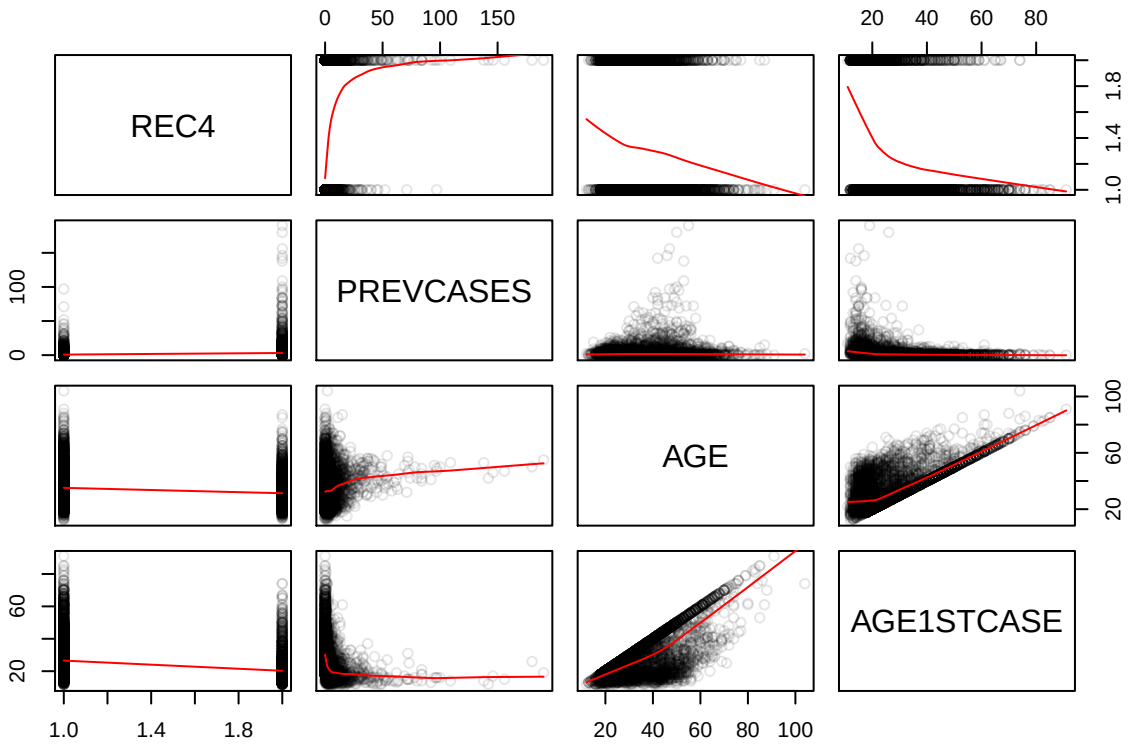
```
p1 <- ggplot(dat, mapping = aes(y = AGE, x = 0)) +
  geom_violin(fill = "cyan") +
  geom_boxplot(width = 0.2, fill = "transparent") +
  xlab("Density")

p2 <- ggplot(dat, mapping = aes(y = AGE1STCASE, x = 0)) +
  geom_violin(fill = "cyan") +
  geom_boxplot(width = 0.2, fill = "transparent") +
  xlab("Density")

p3 <- ggplot(dat, mapping = aes(y = PREVCASES, x = 0)) +
  geom_violin(fill = "cyan") +
  geom_boxplot(width = 0.2, fill = "transparent") +
  xlab("Density")

grid.arrange(p1, p2, p3, ncol = 3)
```



Up until now, we have only focused on single variables, sometimes split by a group variable. We can however also explore several variables simultaneously. This may enhance insight into the interrelations of variables.

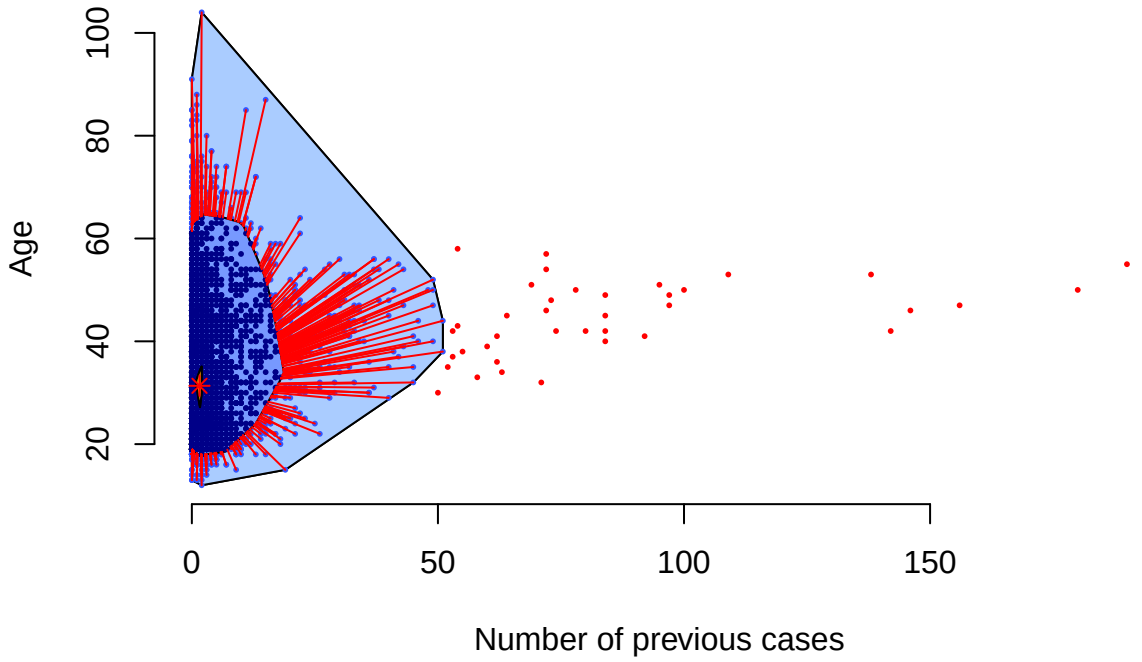
A quick way to visualize this in a data set is the scatterplot matrix, using function `pairs()` from the base `graphics` package that comes installed with R.

```
pairs(dat[,c("REC4", "PREVCASES", "AGE", "AGE1STCASE")],
      col = rgb(0, 0, 0, alpha = 0.1), # transparent (alpha) black for points
      panel = panel.smooth) # add smooth fit curves
```



We can get an idea of the simultaneous distribution of two variables by the bagplot, a bivariate generalization of the box plot.

```
library(aplpack)
bagplot(dat$PREVCASES, dat$AGE, xlab = "Number of previous cases", ylab = "Age")
```



It clearly shows there are some outliers in the age range of 40 to 60. Let us view the other variables of these cases:

```
dat[dat$PREVCASES > 90,]
```

```
##      REC4   SEX COUNTRYBIRTH AGE AGE1STCASE                 CRIMETYPE
## 447     1   man  Netherlands  47         15 property without violence
## 551     1   man  Netherlands  42         12 property without violence
## 562     1   man  Netherlands  46         17 property without violence
## 1153    1 vrouw  Netherlands  47         16 property without violence
## 1371    0   man  Netherlands  49         16                  violence
## 1871    1   man  Netherlands  41         13 property without violence
## 2045    1   man  Netherlands  55         19                    sexual
## 2643    1   man      Surinam  53         14    property with violence
## 2797    1 vrouw  Netherlands  50         26 property without violence
## 2890    1   man      Morocco  51         16 property without violence
## 3223    1   man  Netherlands  53         14 property without violence
## 3363    1   man  Netherlands  50         16 property without violence
##      PREVCASES
## 447        156
## 551        142
## 562        146
## 1153        97
## 1371        97
## 1871        92
## 2045       190
## 2643       109
```

```
## 2797       180
## 2890        95
## 3223       138
## 3363       100
```

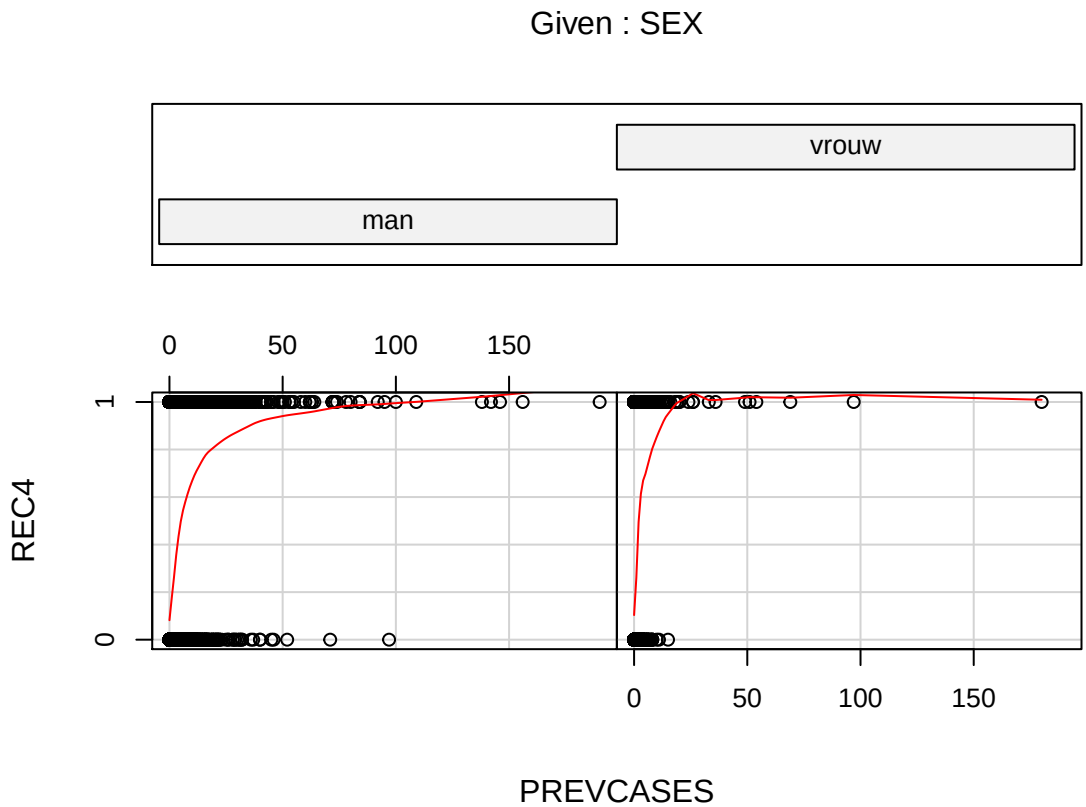
```
# or alternatively using the package dplyr:
# dat %>% filter(PREVCASES > 90)
```

The data shows these individuals started early (AGE1STCASE) and they are relatively old, so there is no reason to suspect the extreme values of PREVCASES.

With `coplot`, plots can be made conditioned on the values of variables, using the formula interface built into R (see `?formula`). It is especially useful because it features automatic (overlapping) categorization of continuous variables.

For instance, is the relation between REC2 and PREVCASES the same for different values of SEX?

```
coplot(REC4 ~ PREVCASES | SEX, # Read: REC4 predicted by PREVCASES given SEX
       data = dat,
       panel = panel.smooth) # add local fit curves to panels
```



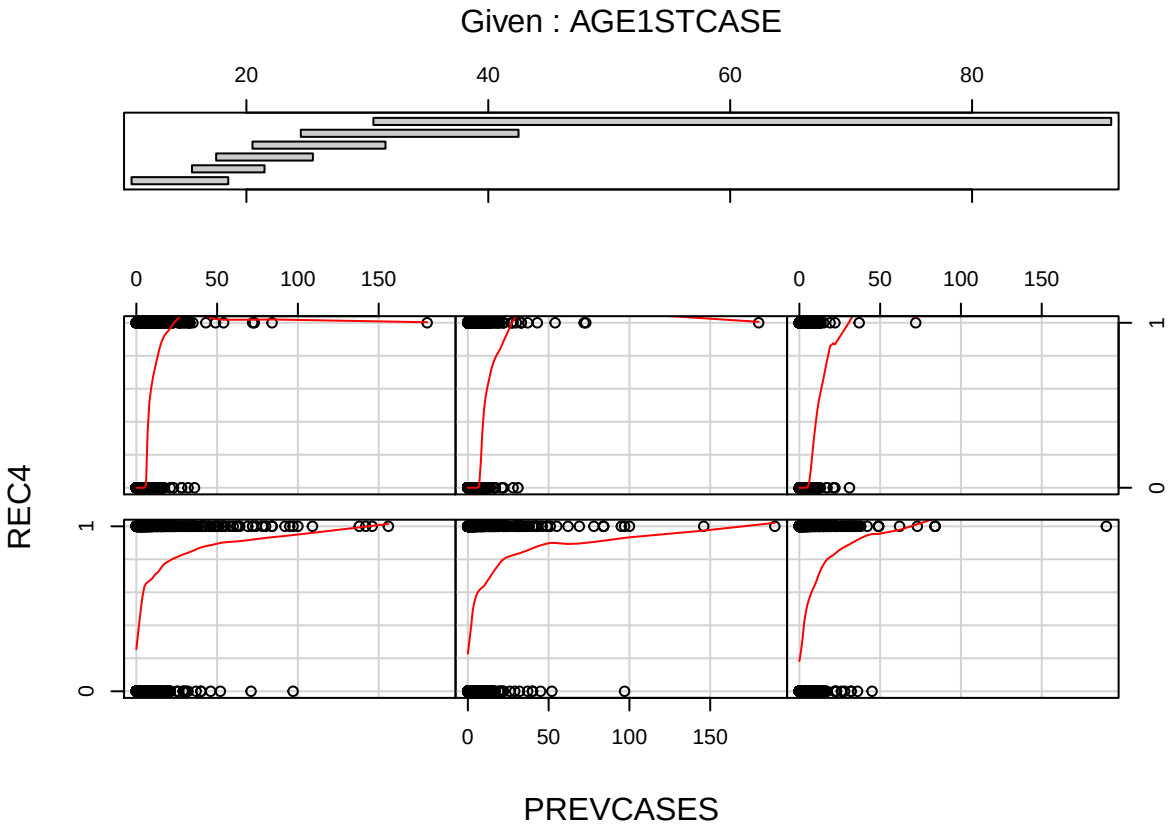
The function passed to the argument `panel` are optimal and specifies extra's to be plotted in each panel. In this case, a smooth fit of y on x.

By plotting a smooth (a local regression, i.e., lowess. See `?loess` in R), a relation between a binary and a continuous variable can be visualized.

Apparently, when you are female, you will more quickly approach the maximum recidivism probability.

You can also condition on continuous variables.

```
coplot(REC4 ~ PREVCASES | AGE1STCASE,
       data = dat,
       panel = panel.smooth)
```

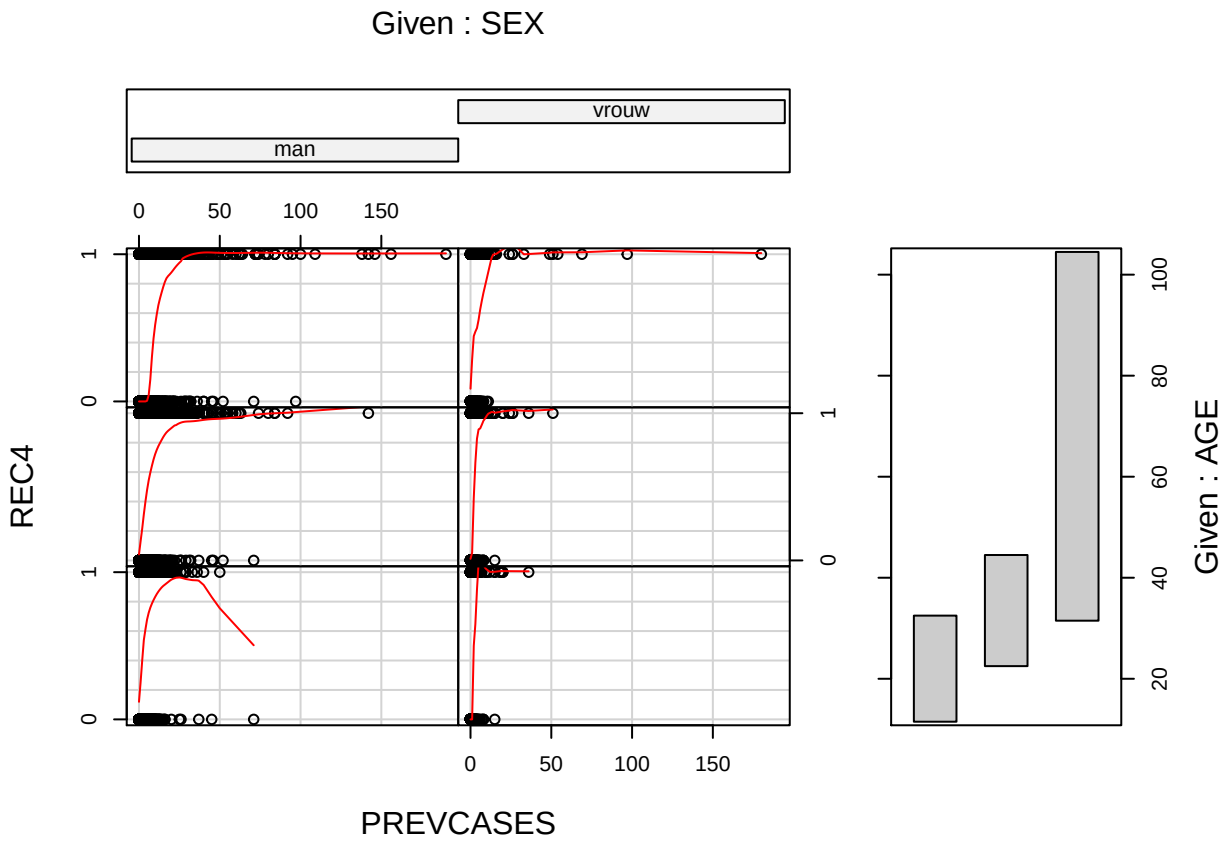


As mentioned before, coplots use overlapping categories when conditioning on continuous variables. This is an advantage when you have a smaller data set.

In other words, the younger someone is at his/her first penal case, the more steep the effect of PREVCASES on recidivism.

Conditioning on two variables is also possible:

```
coplot(REC4 ~ PREVCASES | SEX * AGE,
       number = 3,
       data = dat,
       panel = panel.smooth)
```



The automatic categorization of continuous variables is unfortunately not supported by ggplot2.

Finally, interactive 3d-plots can be generated with package `rgl` (a separate window is opened. Try dragging the plot with the left mouse button clicked)

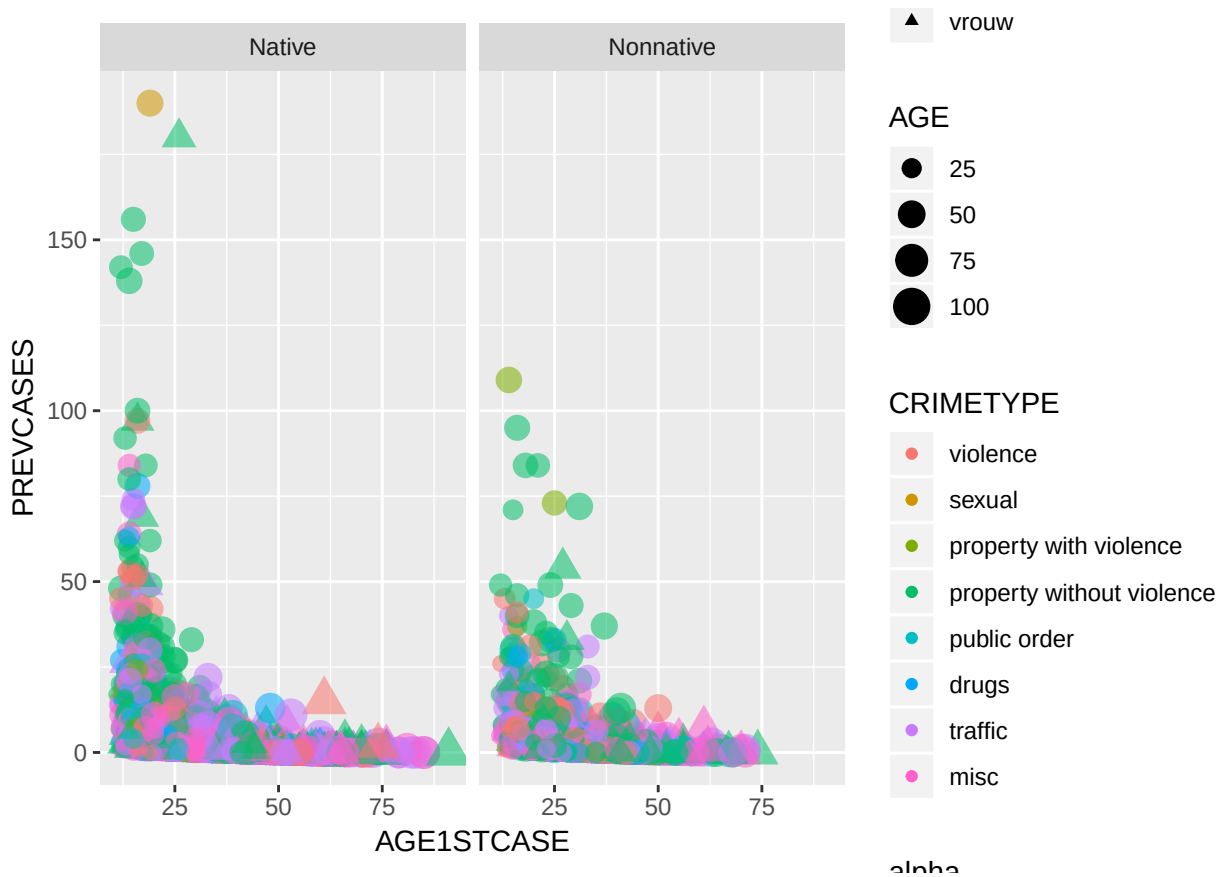
```
library(rgl)
spheres3d(x = dat$AGE,
          y = dat$AGE1STCASE,
          z = dat$PREVCASES,
          col = as.numeric(dat$REC4))
axes3d(labels = FALSE)
title3d(main = '3d scatterplot', xlab = 'AGE', ylab =  'AGE1STCASE', zlab = 'PREVCASES')
```

Interactive plotting with ggplot2

The grammar of graphics in the `ggplot2` package allows for quick interactive plotting of multiple variable:

```
# binarize country of birth.
dat$ETN <- factor(ifelse(dat$COUNTRYBIRTH == "Netherlands",
                         "Native",
                         "Nonnative"))

p <- ggplot(data = dat, aes(AGE1STCASE, PREVCASES)) +
  geom_point(aes(shape = SEX,
                 size = AGE,
                 color = CRIMETYPE,
                 alpha = 0.15)) +
```

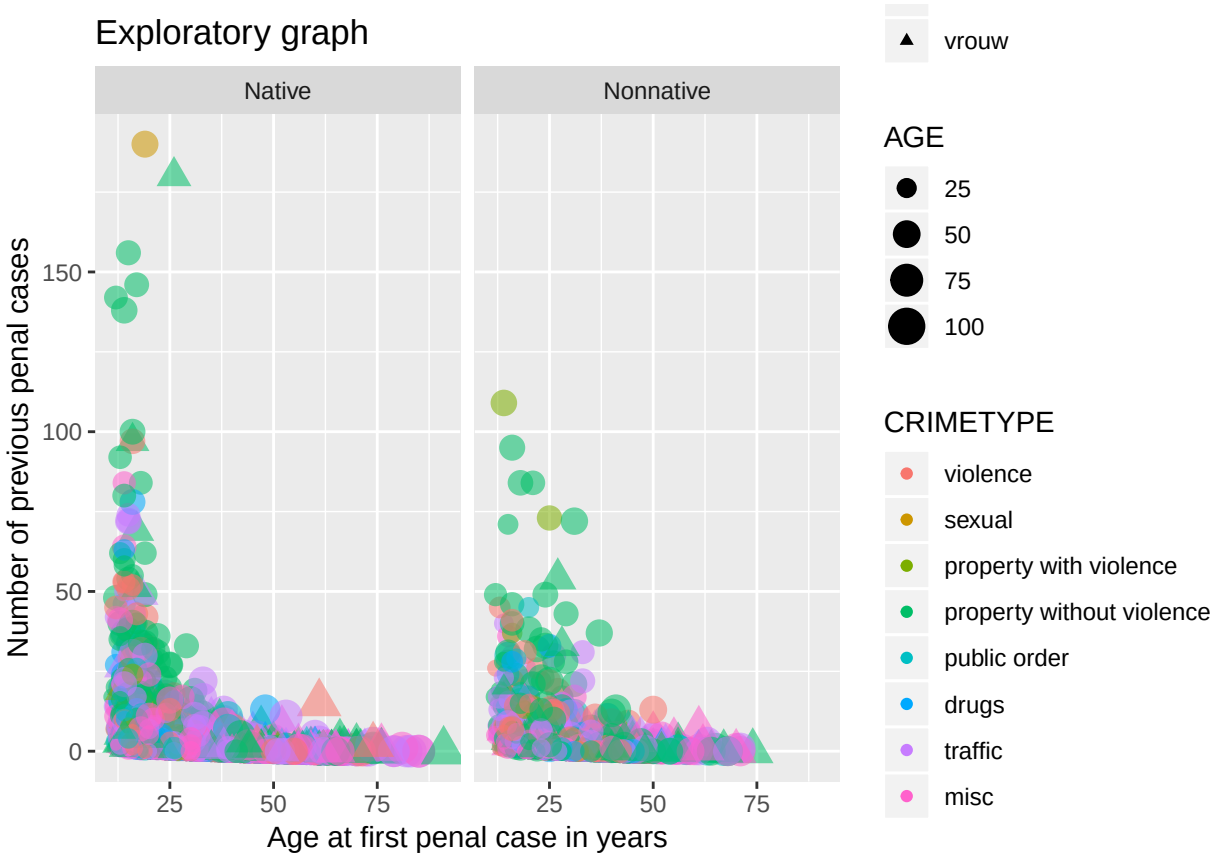
```
  facet_wrap( ~ ETN) # facet (different plots) by ethnicity
p
```



As you can see, we managed to put six variables into one plot. However, obtaining *meaningful* vizualizations requires experimentation. For example, in the above graph, `size` and `shape` aesthetics do not convey a lot of information because of overplotting (i.e. many points occupying the same space).

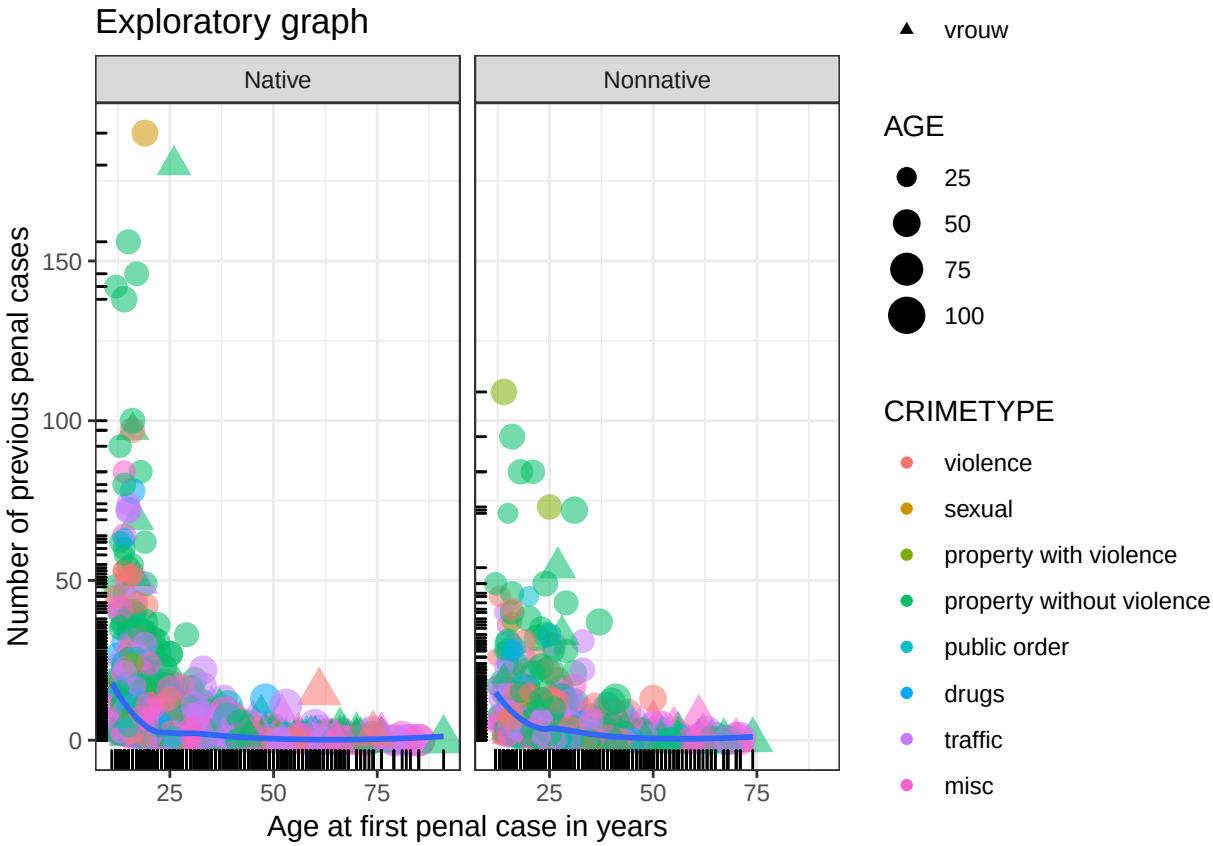
All the information for building the plot is stored in the object p. Calling p causes R to display the actual plot. We can now easily add different labels and a title, change the theme and store it in the object.

```
p <- p +
  labs(x = "Age at first penal case in years",
       y = "Number of previous penal cases",
       title = "Exploratory graph")
p
```



Adding a smooth line without confidence bands, a rug (i.e. a 1-dimensional scatter on an axis) and changing the theme, after the plot was created is then trivial.

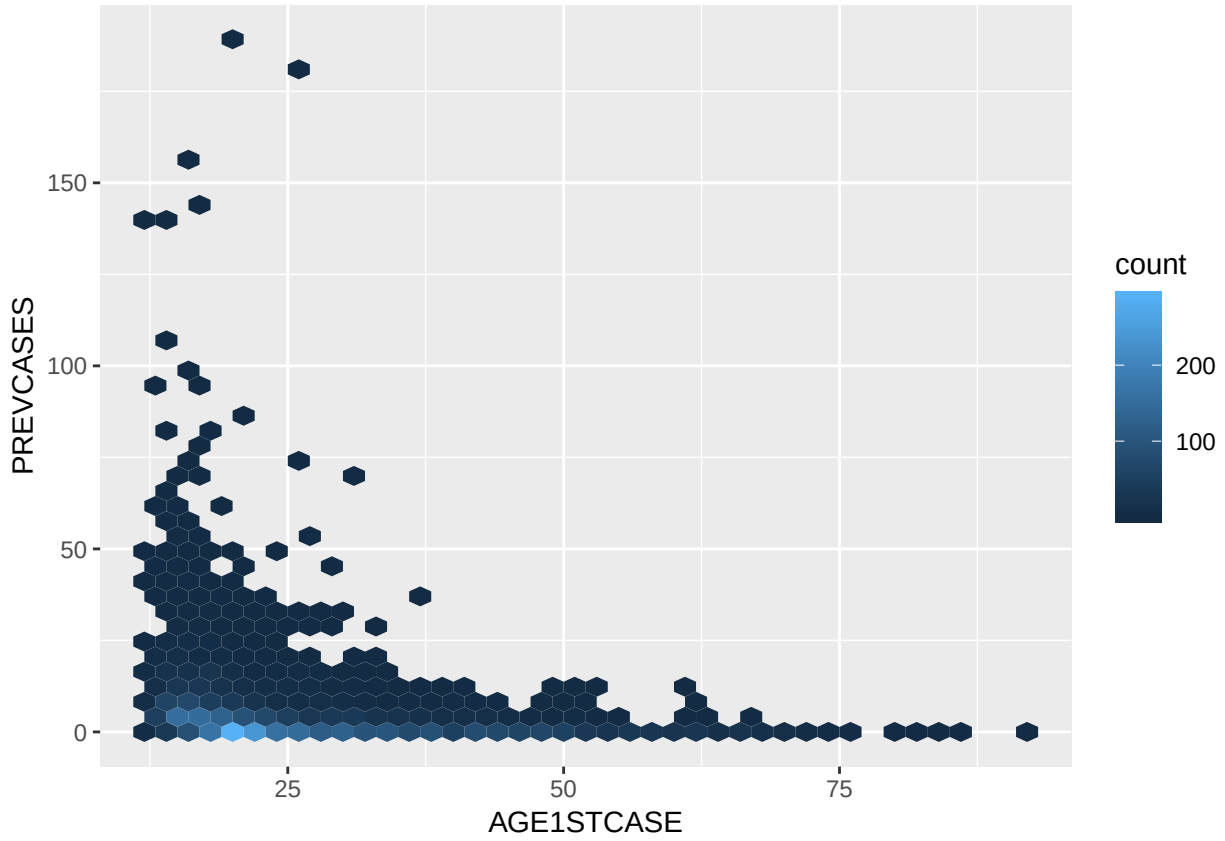
```
p +
  geom_smooth(se = FALSE, method = "loess") +
  geom_rug() +
  theme_bw()
```



This makes ggplot very suitable to interactive explorative plot building.

If you have a serious issue with overplotting, the binhex plot might be a solution. Being a combination of a scatterplot and a histogram, it creates hexagonal bins that are color coded for the number of observations that fall inside the bins.

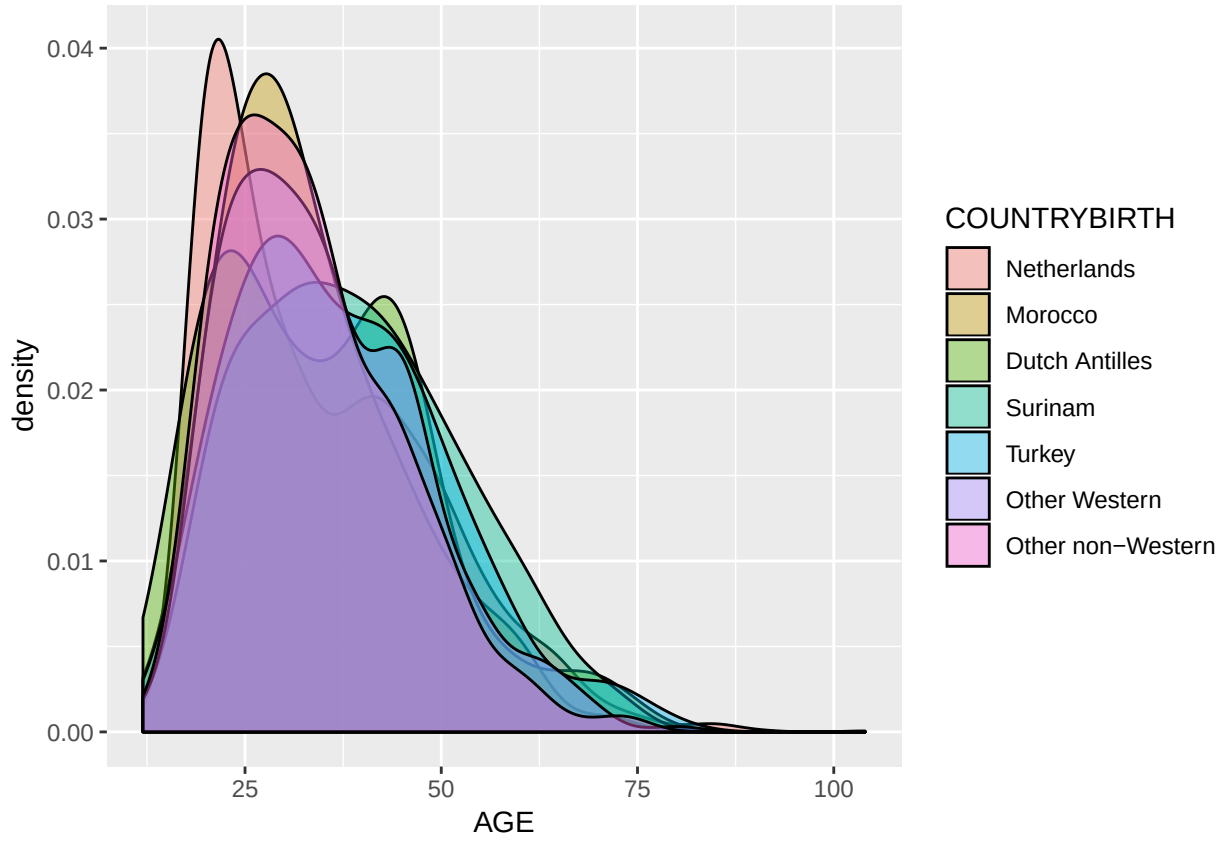
```
library(ggplot2)
p <- ggplot(data = dat, aes(AGE1STCASE, PREVCASES)) +
  stat_binhex(bins = 40)
 p
```



It now become more apparent in which region most observations are concentrated.

Finally, we create a density plot by country of birth:

```
ggplot(dat, aes(AGE, fill = COUNTRYBIRTH)) +
stat_density(aes(), position = "identity", alpha = 0.4, color = "black")
```



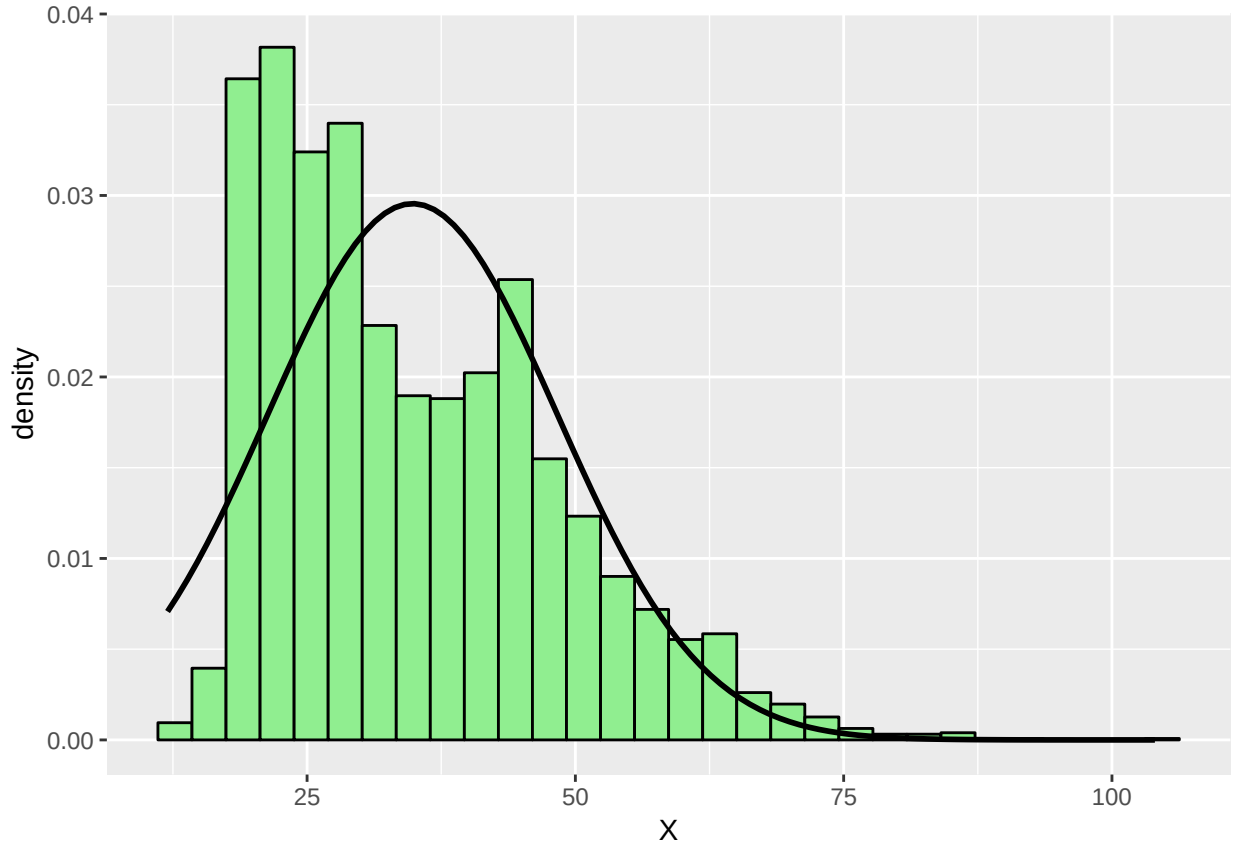
Those not born in the Netherlands can have a strongly different age distribution than the total sample.

Exploratory data analysis for modelling

Some statistical models have assumptions like normal distribution or linearity. Exploratory data analysis can be applied prior to the modelling to explore violations and find transformations that alleviate these violations.

Linear regression, for instance, requires normally distributed outcomes. Say we want to model the AGE from the remaining variables. Then we plot this variable with an overlaid normal distribution. The mean and standard deviation for the distribution are estimated from the data.

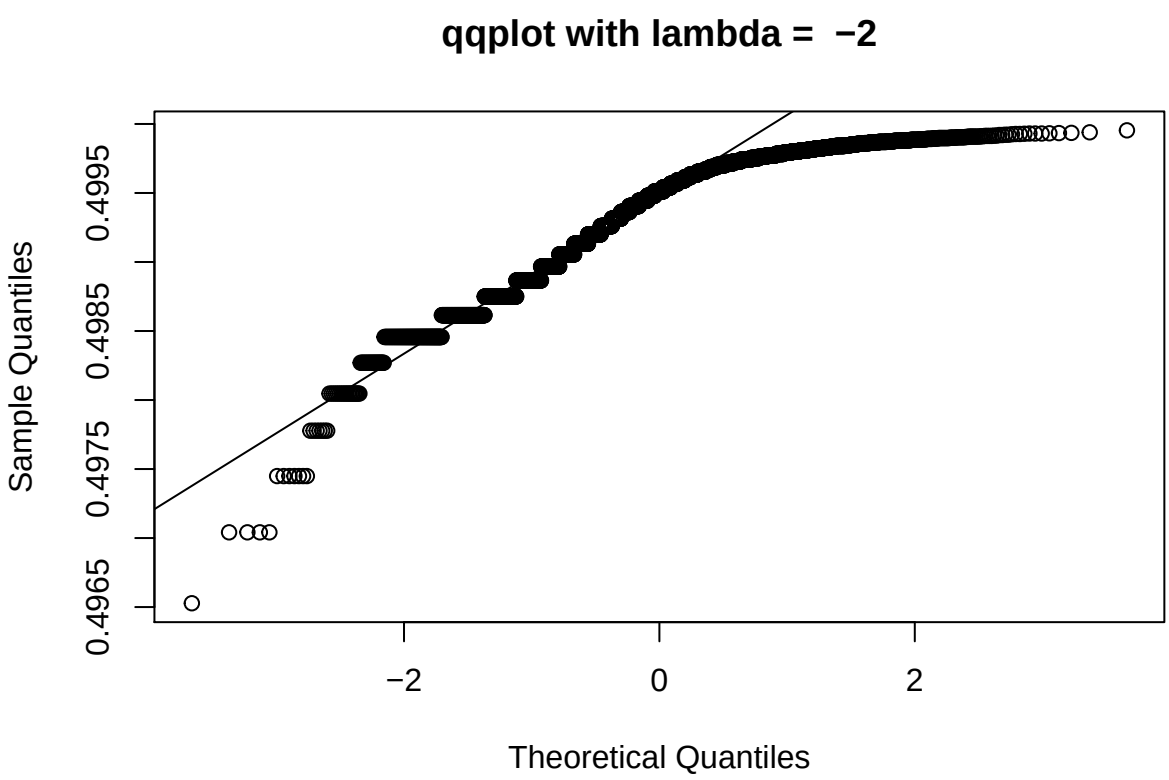
```
# Define a function for a combined histogram and estimated normal distribution
histnorm <- function(X) {
  p <- ggplot(data = data.frame(X), aes(x=X)) +
    geom_histogram(aes(y=..density..),
                   bins = 30,
                   colour = "black",
                   fill="lightgreen") +
    stat_function(fun = dnorm,
                  args = list(mean = mean(X),
                              sd = sd(X)),
                  size = 1)
  return(p)
}
histnorm(dat$AGE)
```

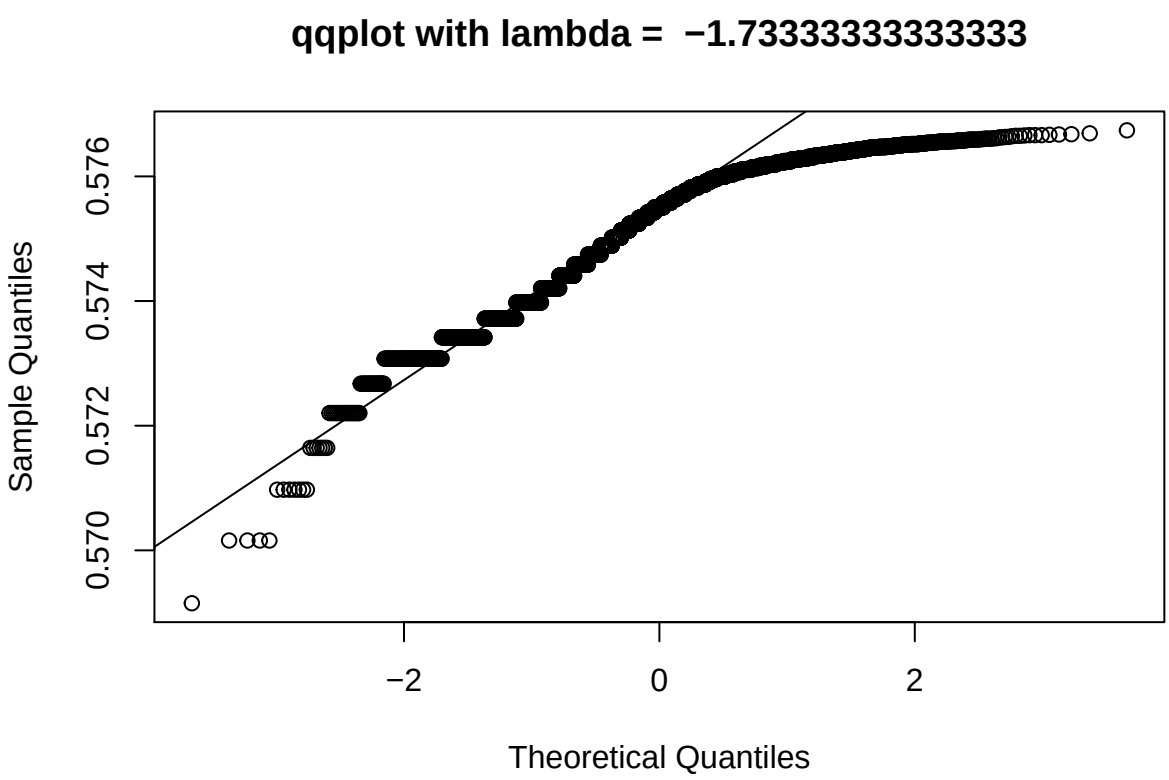


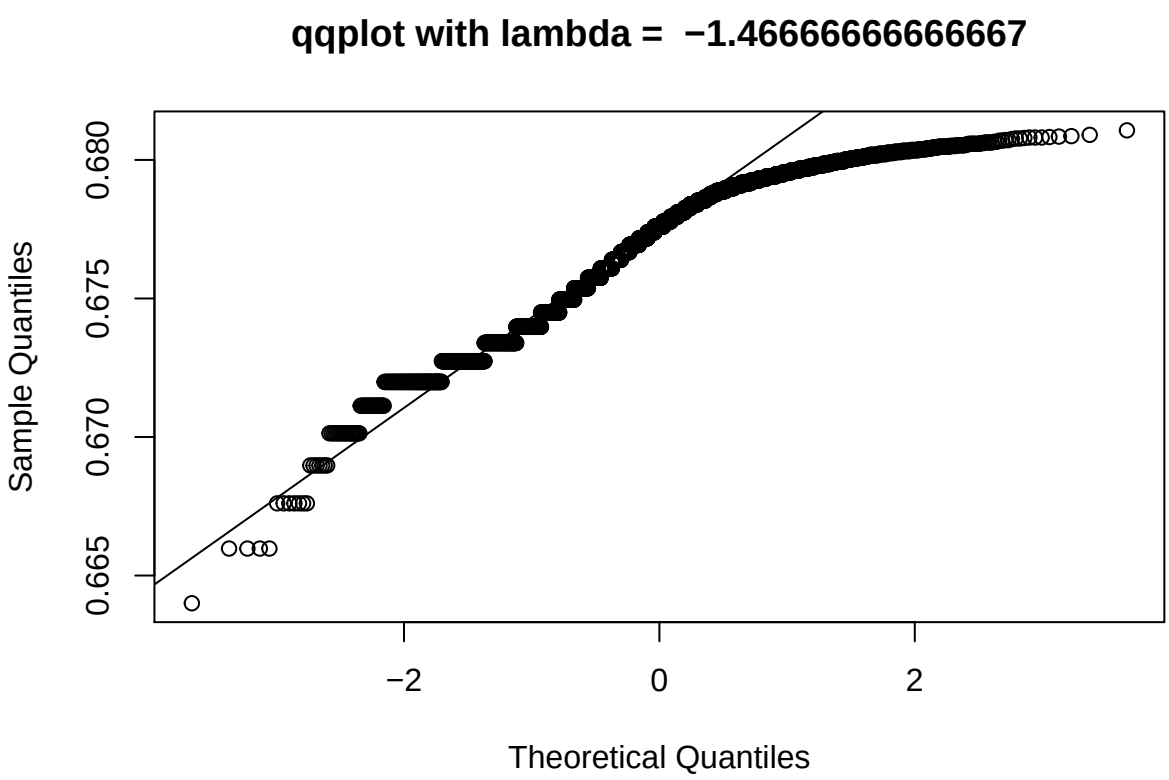
Visually, AGE does not follow a normal distribution.

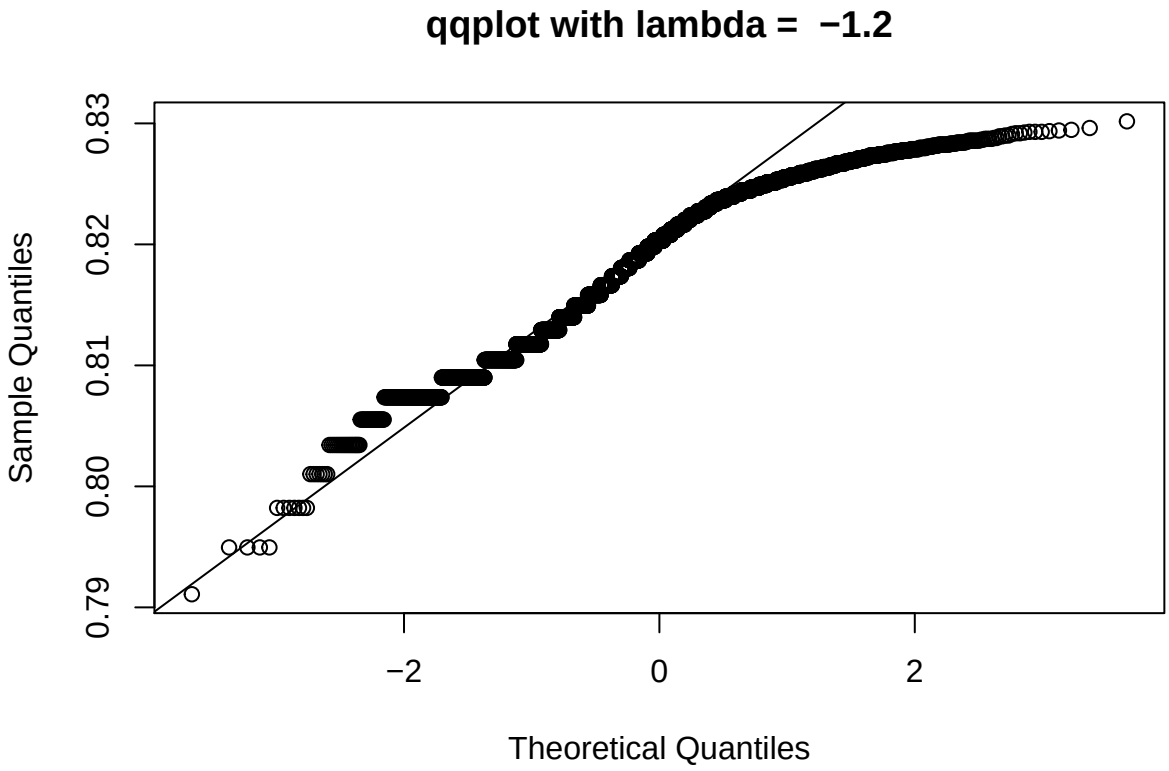
The Box-Cox family of transformations can be used to transform AGE to normality. Normality can be visually assessed by a qqplot. It plots the observed quantiles against the theoretically expected distribution, in our case the normal distribution. When the qqplot follows a straigth line, the data is normally distributed.

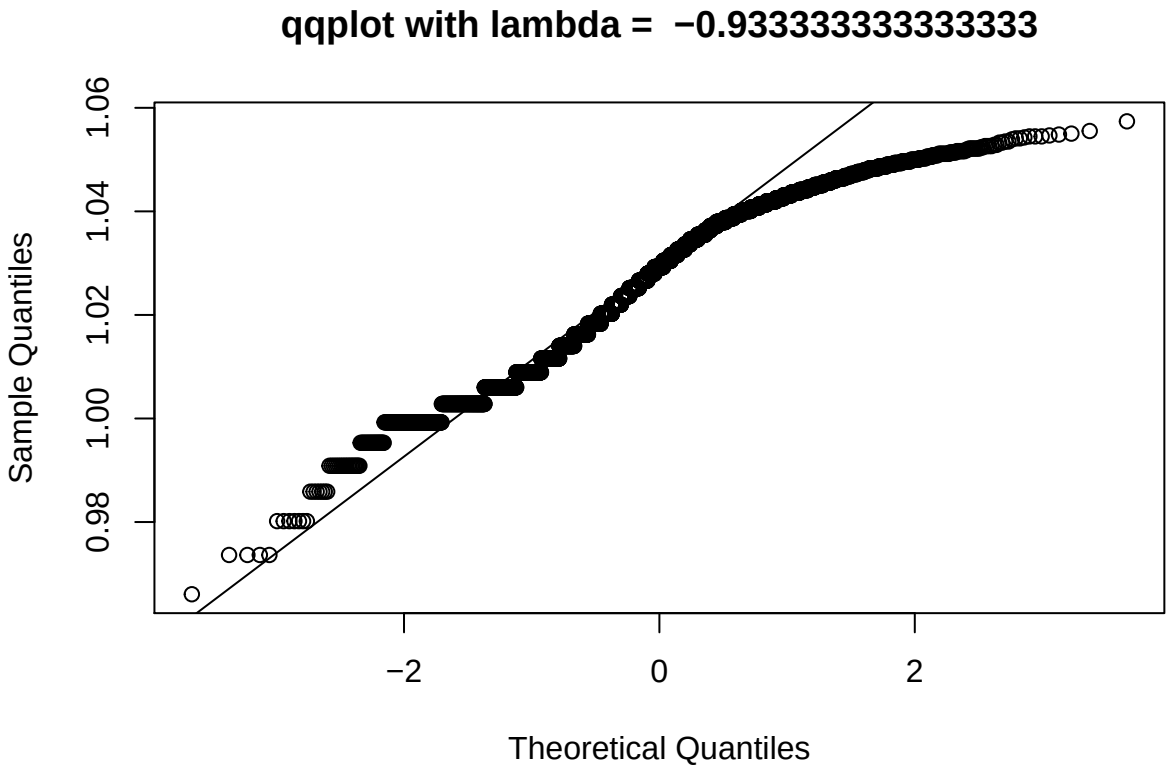
```
box.cox <- function(x, lambda){
  # applies the box-cox family of transformations
  # special values are:
  # -1 -> 1/x
  # 0 -> logarithm
  # 0.5 -> square root
  # 2 -> square
  if (lambda==0) {
    xprime <- log(x)
  } else {
  xprime <- (x^lambda - 1) / lambda
  }
}
lambdavec <- seq(-2,2, length.out = 16)
for (i in 1:length(lambdavec)){
  # histnorm(box.cox(dat$AGE, lambdavec[i]))
  # title(paste("lambda = ",round(lambdavec[i],1)), cex.main = .5)
  transvar <- box.cox(dat$AGE, lambdavec[i])
  qqnorm(transvar, main = paste("qqplot with lambda = ", lambdavec[i]))
  qqline(transvar, prob = c(0.1, 0.6))
}
```

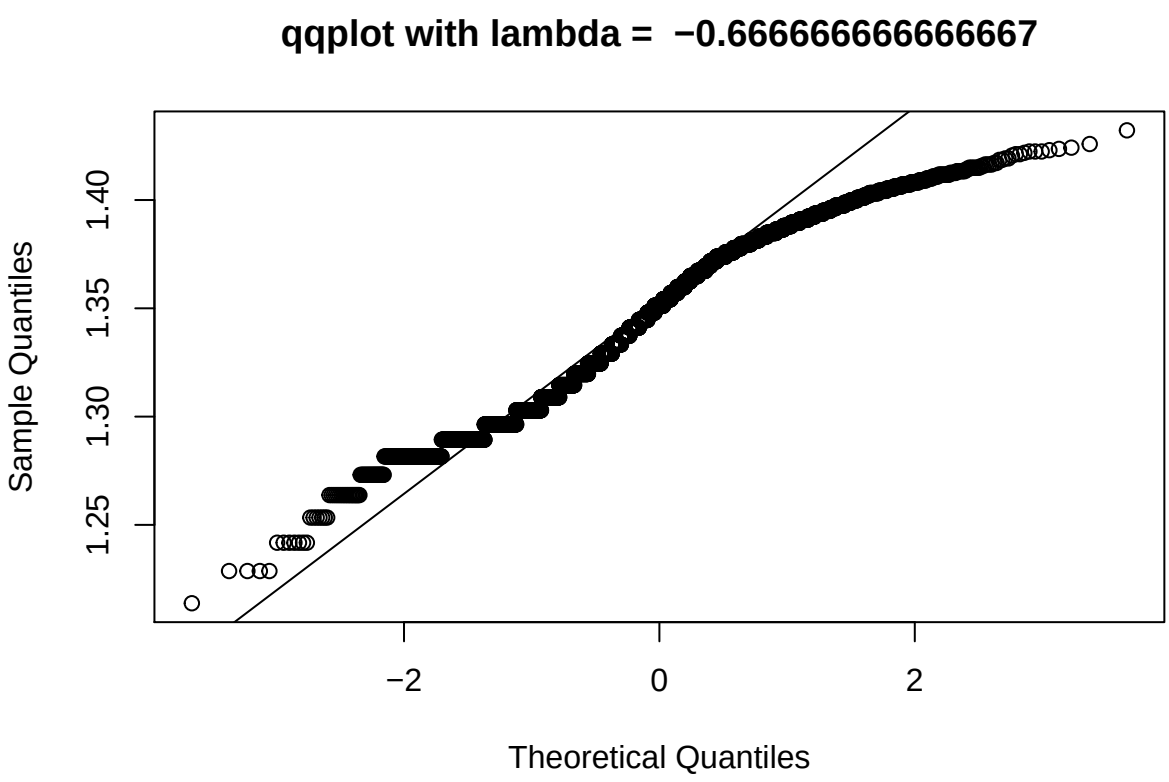

qqplot with lambda = −2
Sample Quantiles
0.4965
0.4975
0.4985
0.4995
−2
0
2
Theoretical Quantiles

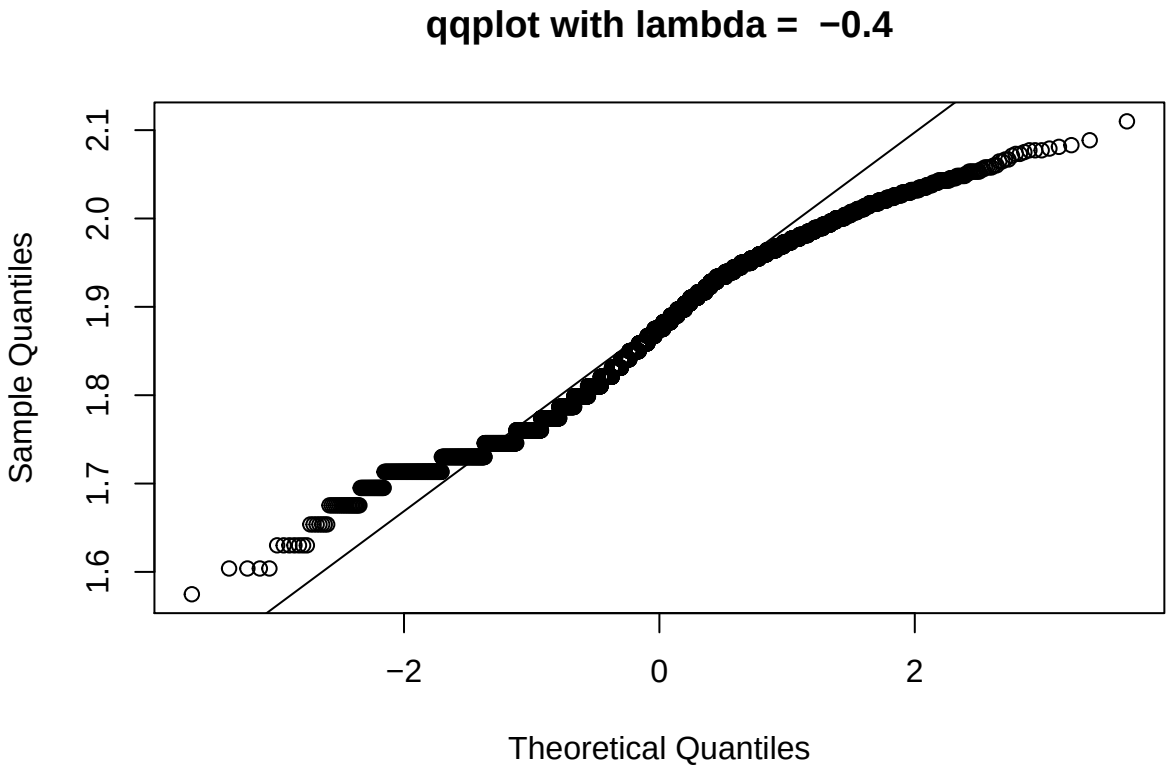

qqplot with lambda = −1.73333333333333
Sample Quantiles
0.570
0.572
0.574
0.576
−2
0
2
Theoretical Quantiles

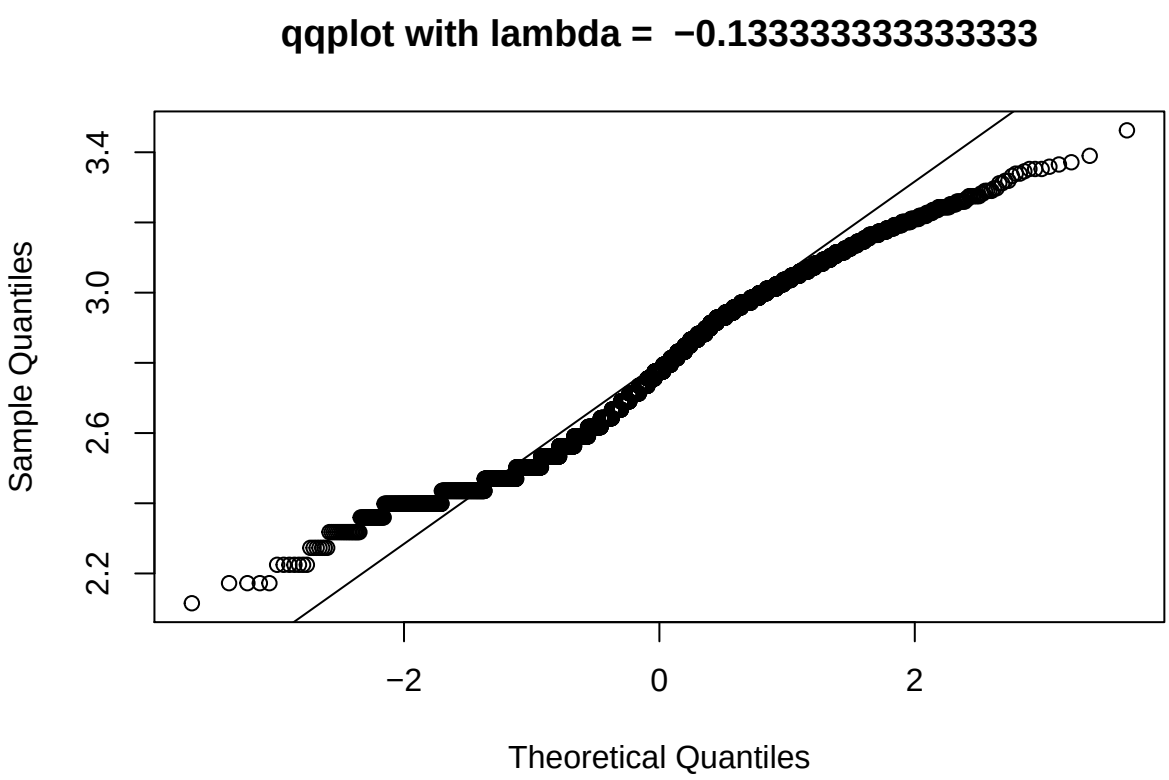

qqplot with lambda = −1.46666666666667
Sample Quantiles
0.665
0.670
0.675
0.680
−2
0
2
Theoretical Quantiles

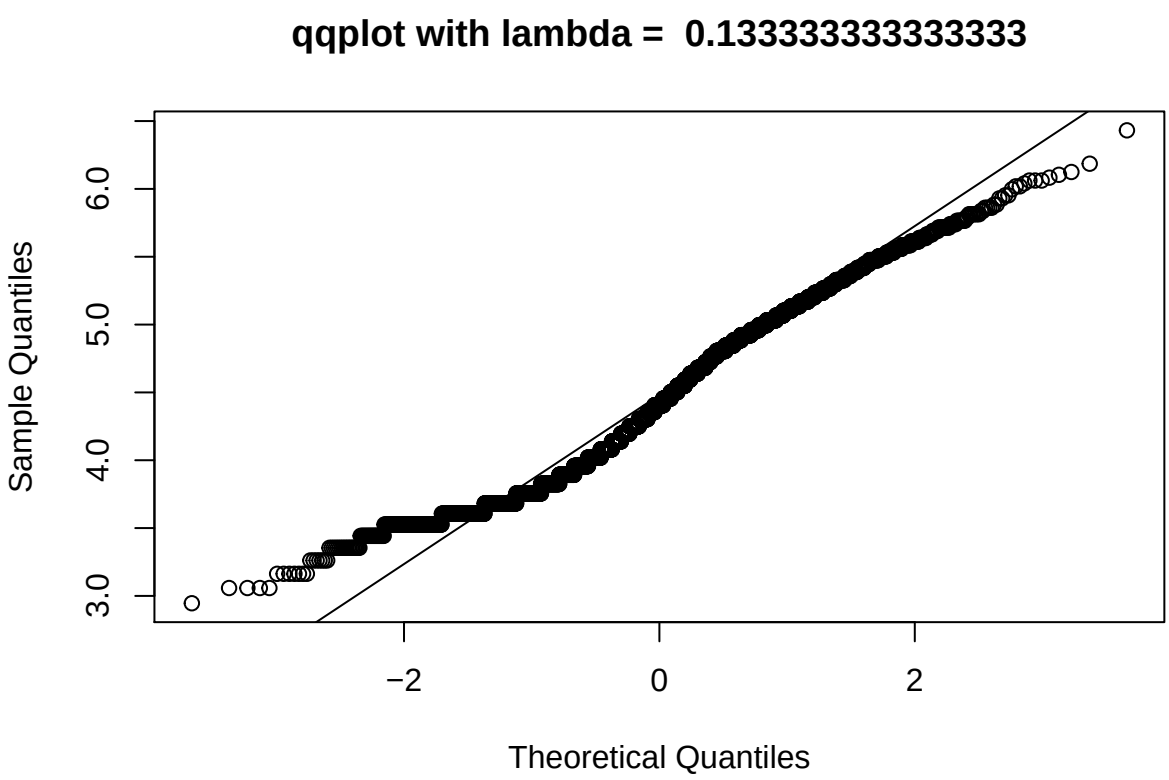

qqplot with lambda = −1.2
Sample Quantiles
0.79
0.80
0.81
0.82
0.83
−2
0
2
Theoretical Quantiles

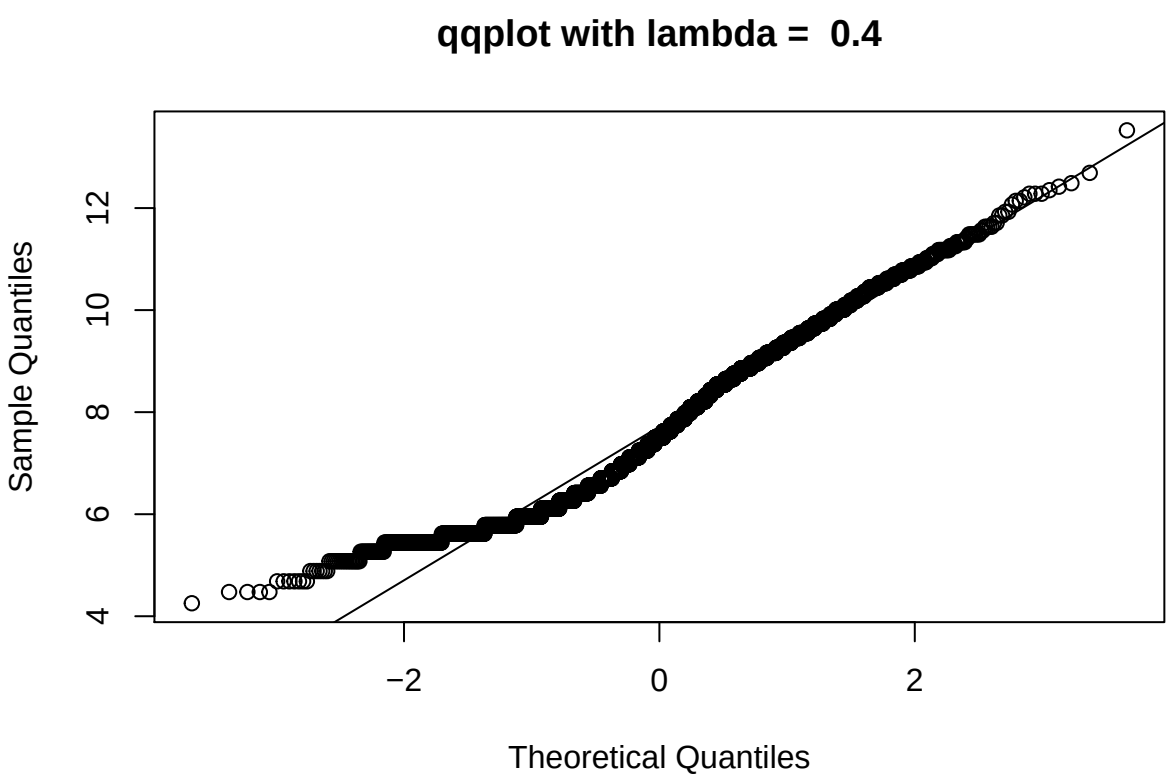

qqplot with lambda = −0.933333333333333
Sample Quantiles
0.98
1.00
1.02
1.04
1.06
−2
0
2
Theoretical Quantiles

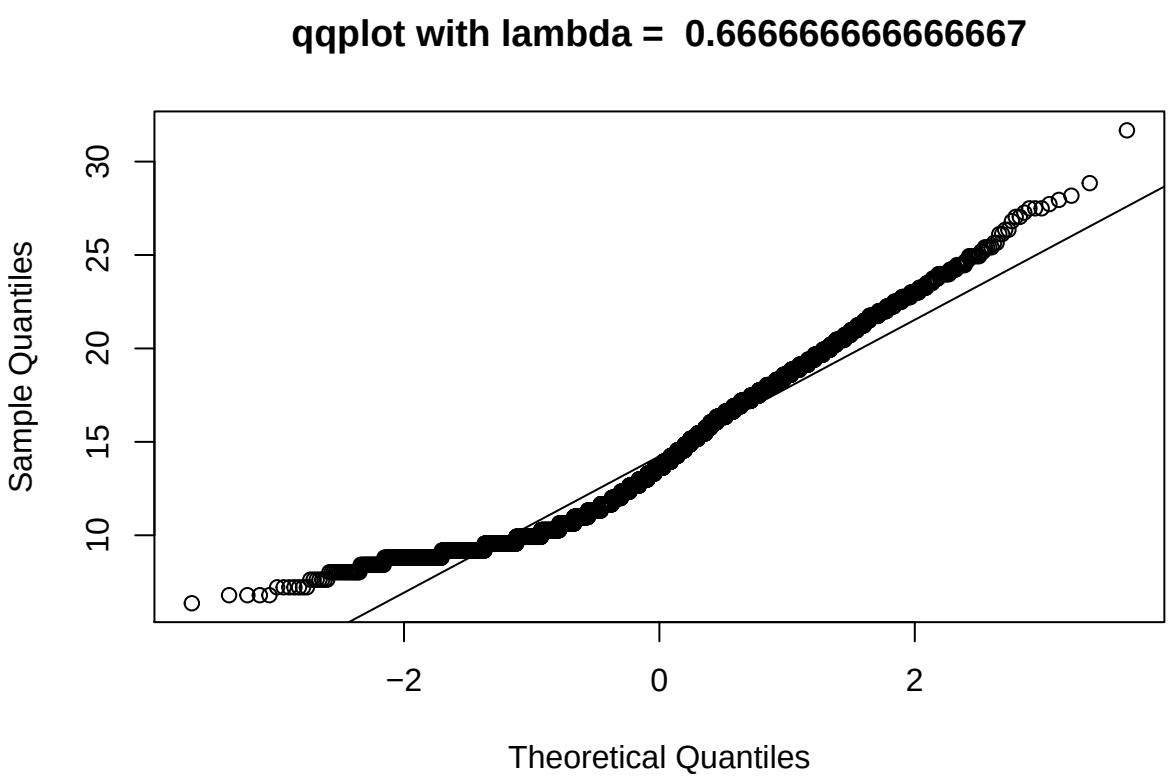

qqplot with lambda = −0.666666666666667
Sample Quantiles
1.25
1.30
1.35
1.40
−2
0
2
Theoretical Quantiles

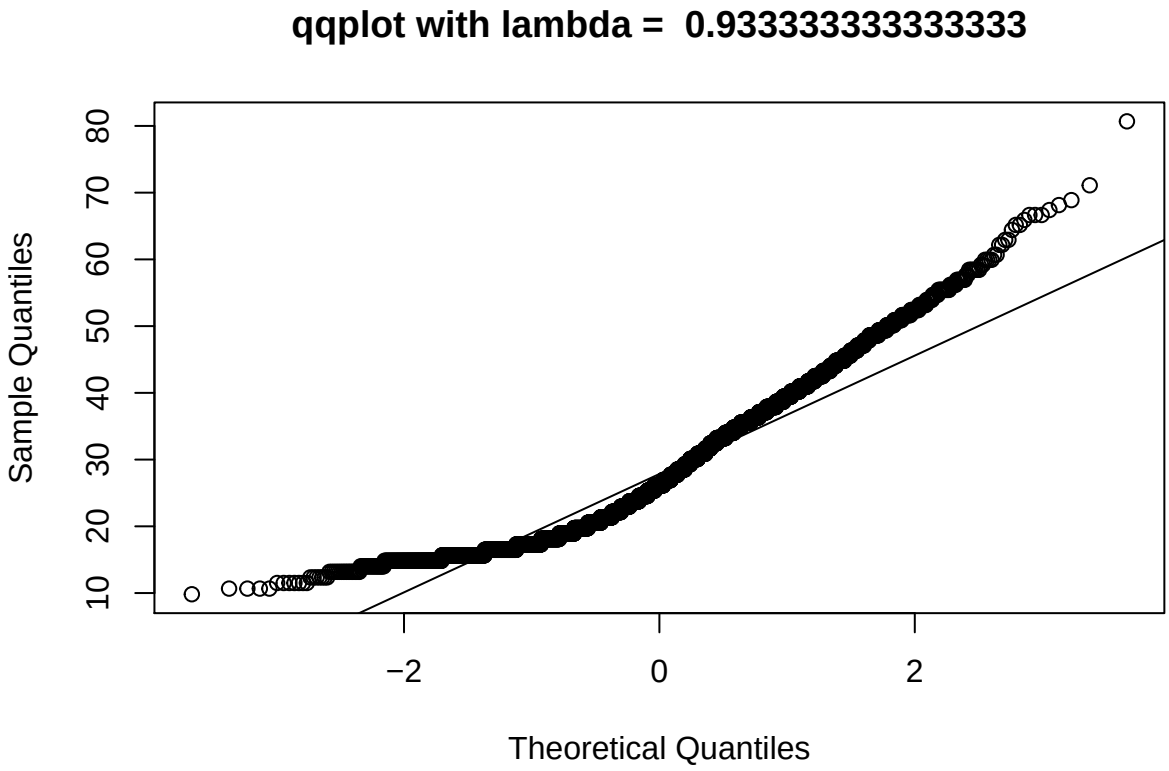

qqplot with lambda = −0.4
Sample Quantiles
1.6
1.7
1.8
1.9
2.0
2.1
−2
0
2
Theoretical Quantiles

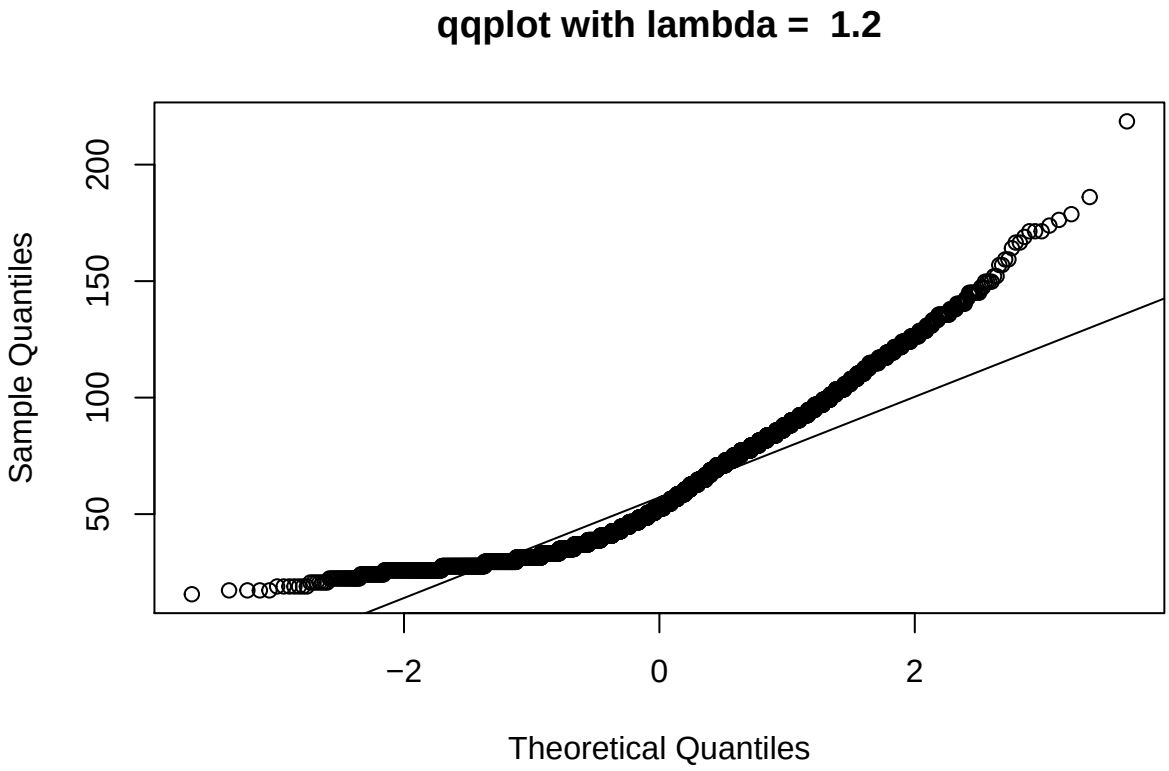

qqplot with lambda = −0.133333333333333
Sample Quantiles
2.2
2.6
3.0
3.4
−2
0
2
Theoretical Quantiles

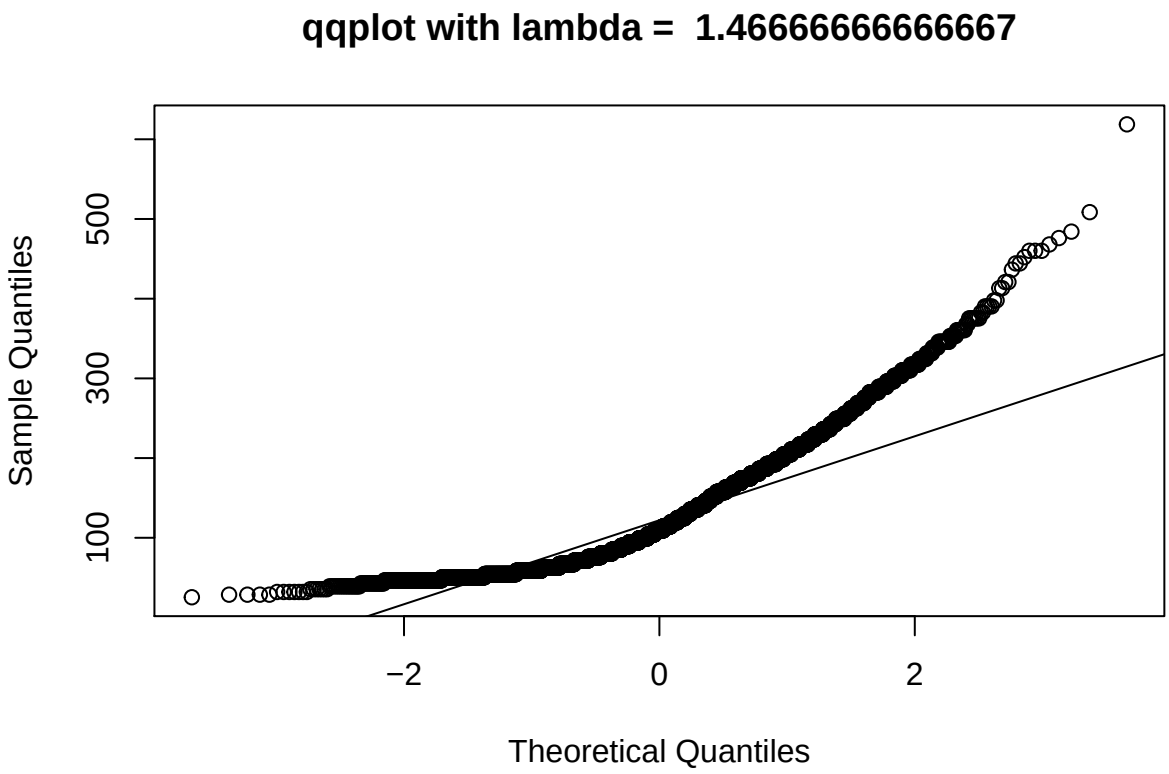

qqplot with lambda = 0.133333333333333
Sample Quantiles
3.0
4.0
5.0
6.0
−2
0
2
Theoretical Quantiles

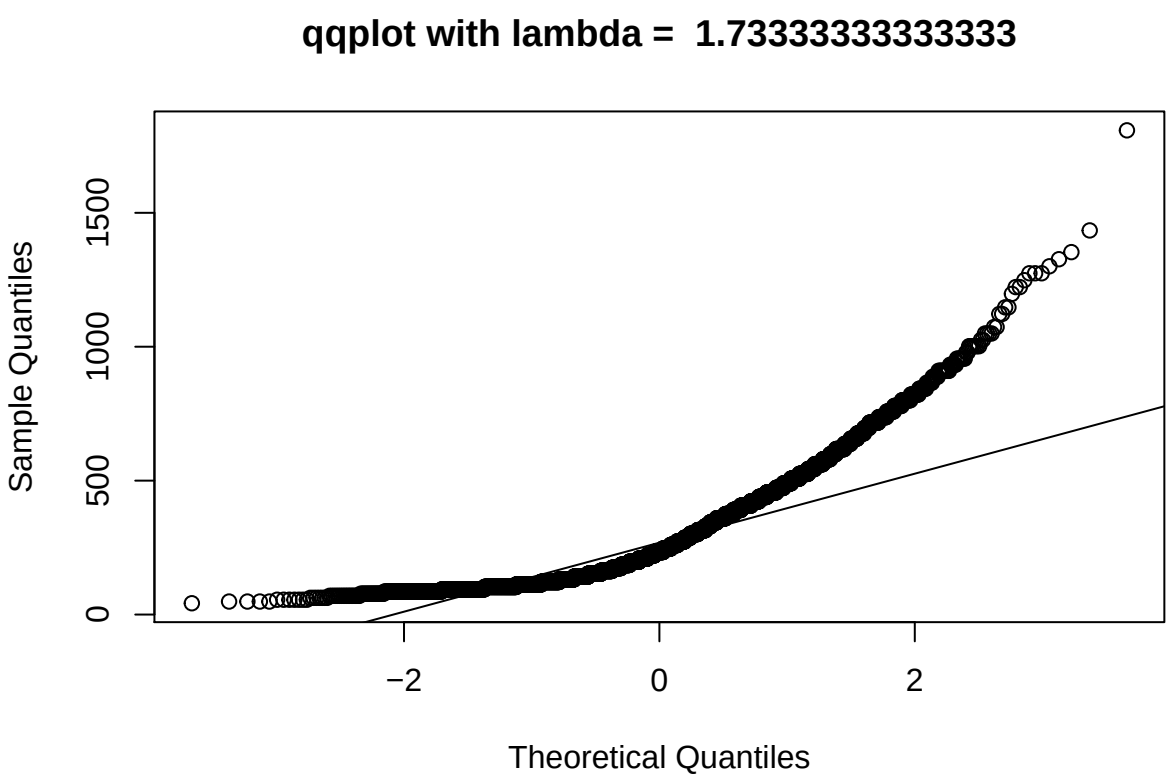

qqplot with lambda = 0.4
Sample Quantiles
4
6
8
10
12
−2
0
2
Theoretical Quantiles

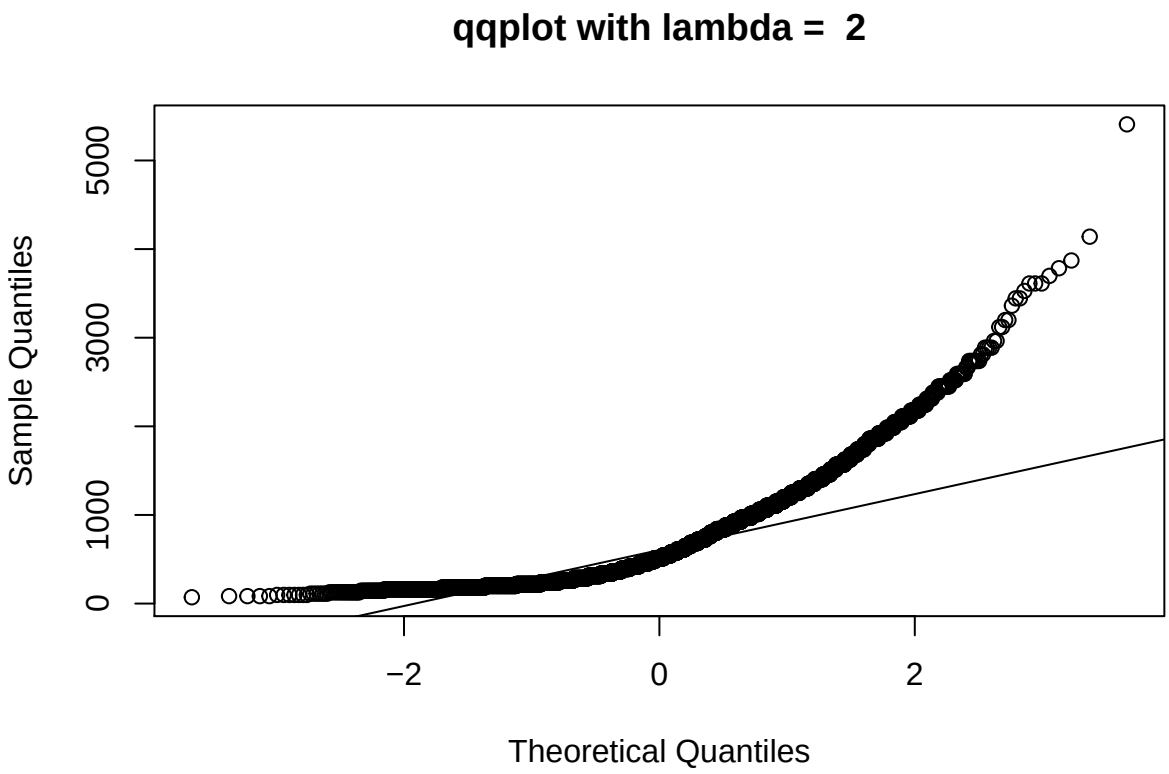

qqplot with lambda = 0.666666666666667
Sample Quantiles
10
15
20
25
30
−2
0
2
Theoretical Quantiles


qqplot with lambda = 0.933333333333333
Sample Quantiles
10
20
30
40
50
60
70
80
−2
0
2
Theoretical Quantiles


qqplot with lambda = 1.2
Sample Quantiles
50
100
150
200
−2
0
2
Theoretical Quantiles

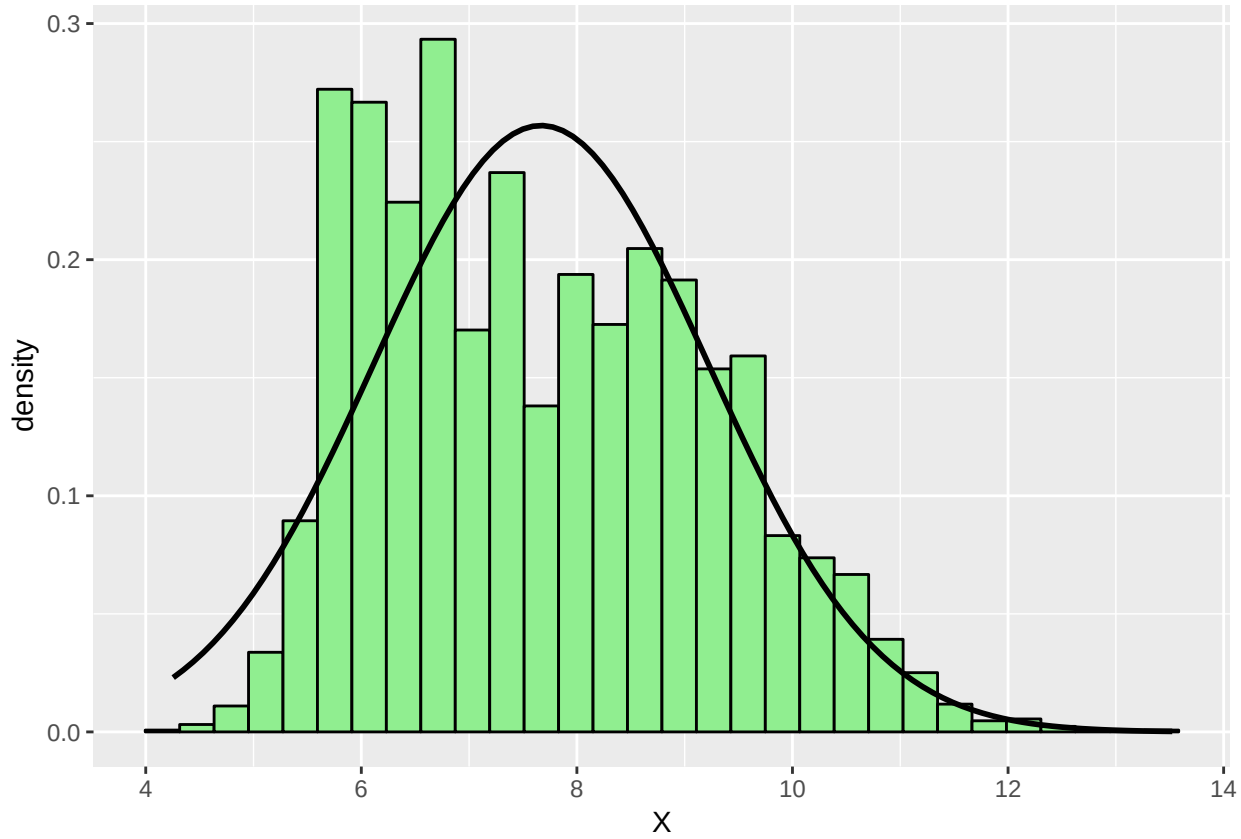

qqplot with lambda = 1.46666666666667
Sample Quantiles
100
300
500
−2
0
2
Theoretical Quantiles


qqplot with lambda = 1.73333333333333
Sample Quantiles
0
500
1000
1500
−2
0
2
Theoretical Quantiles



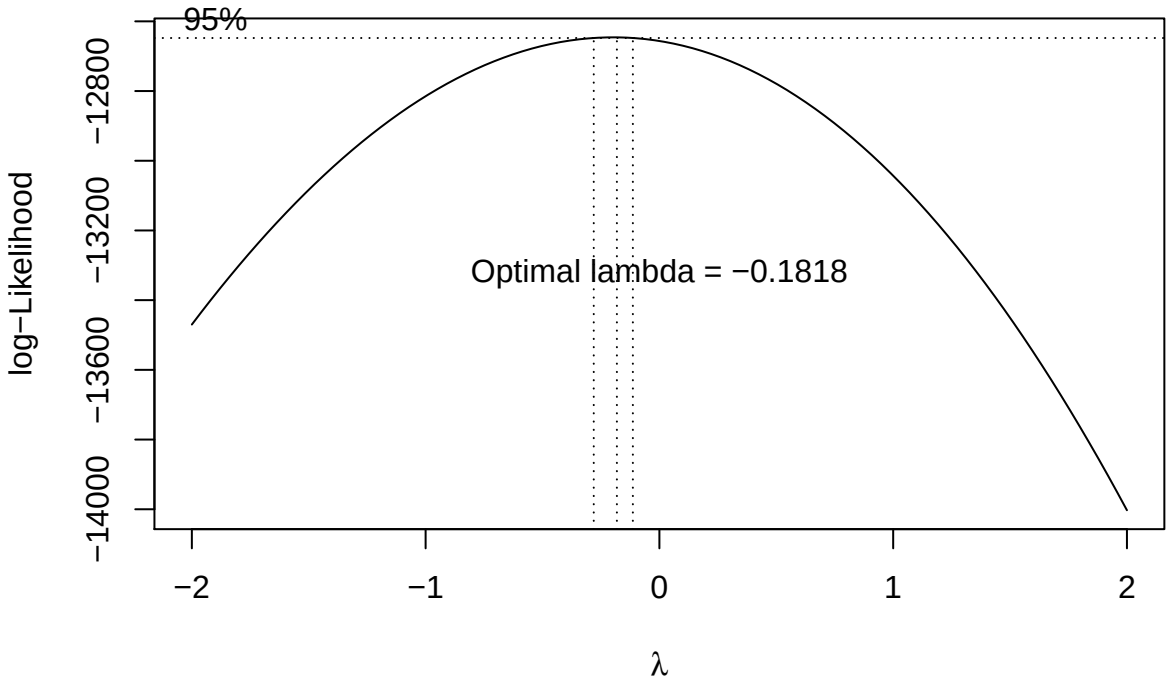
Although far from perfect, a lambda 0.4 seems to fit best. This is the corresponding histogram:

```
histnorm(box.cox(dat$AGE, 0.4))
```



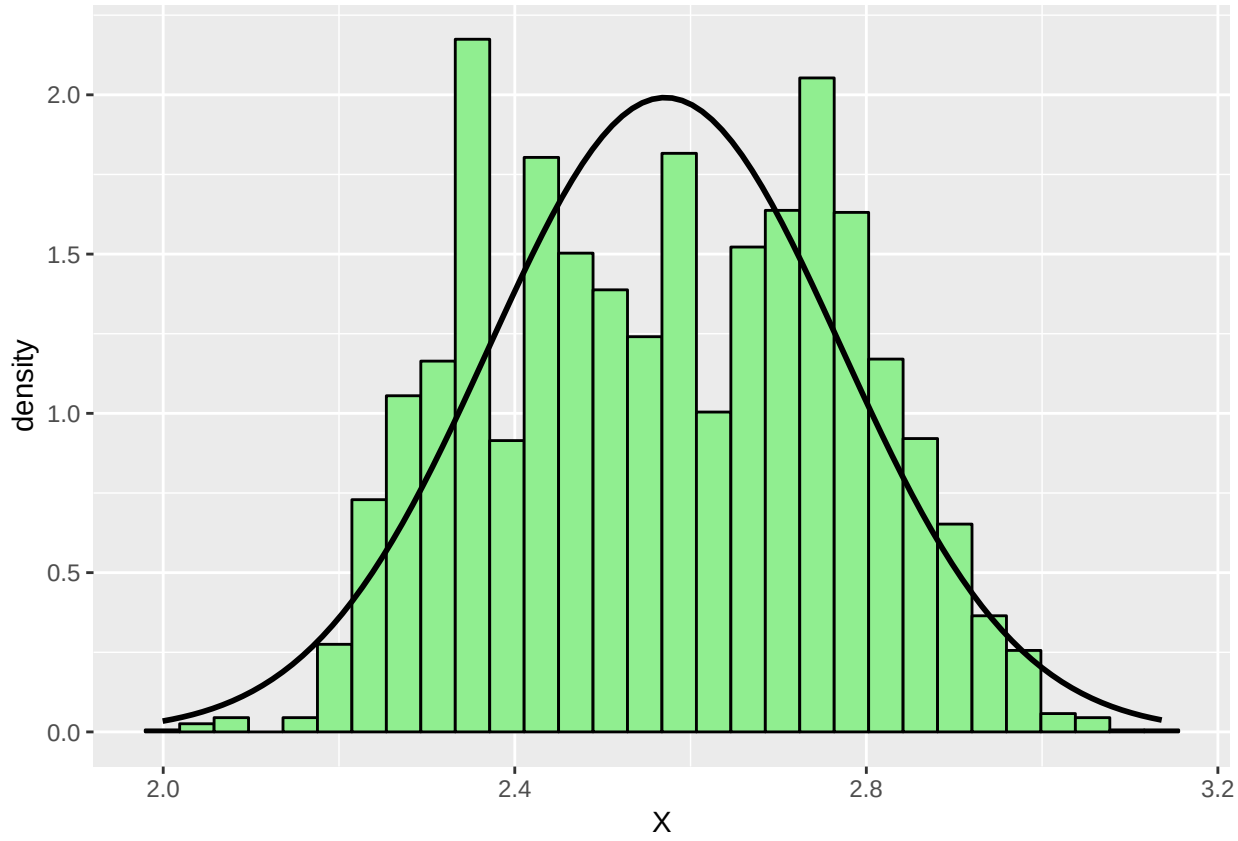
We can also find the optimal lambda using the log-likelihood criterion, using the `boxcox` function and fitting a constant only model. It calculates the log-likelihood over a grid of lambda values and estimates the optimal lambda through spline interpolation:

```
bc <- MASS::boxcox(AGE ~ 1, data = dat)
optlambda <- bc$x[which.max(bc$y)]
text(0, min(bc$y) + (max(bc$y)-min(bc$y))/2, sprintf("Optimal lambda = %.4f", optlambda))
```

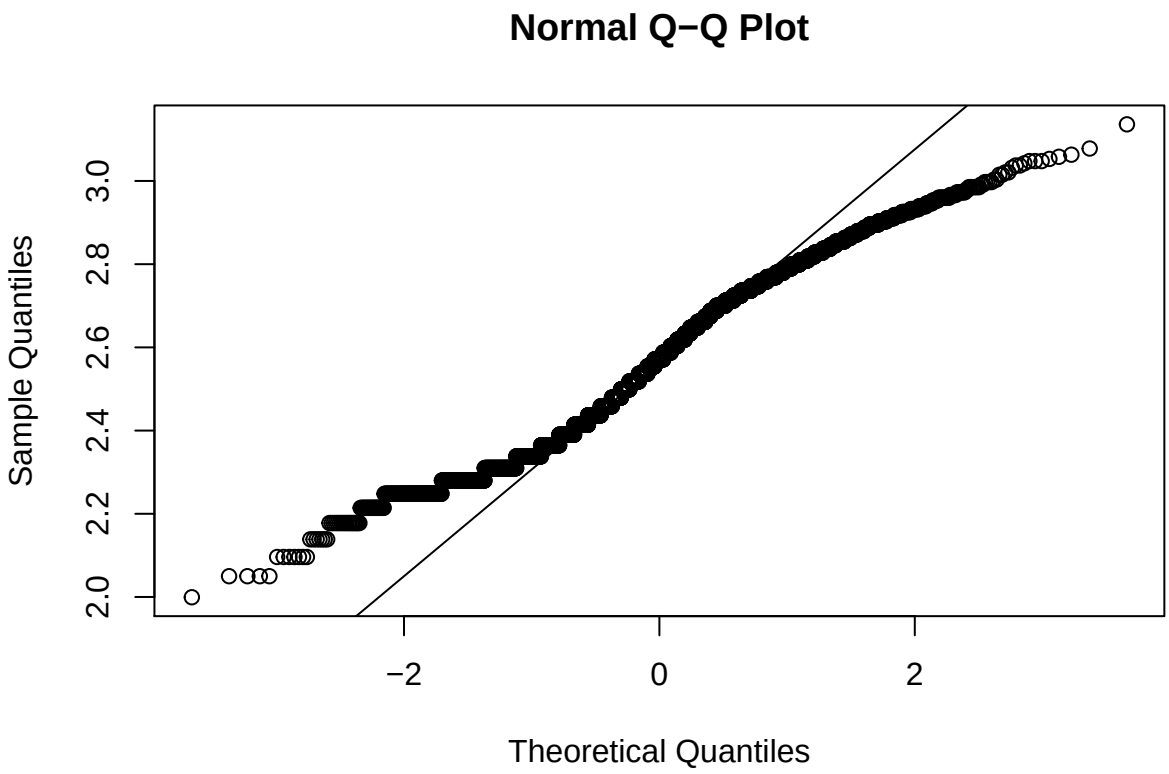


Let's see how that histogram looks

```
histnorm(box.cox(dat$AGE, optlambda))
```




```
qqnorm(box.cox(dat$AGE, optlambda))
qqline(box.cox(dat$AGE, optlambda))
```



This is a different value of λ from the one we found using visual inspection.

We could now incorporate the transformed variable in a statistical model.

Temporal and spatial data

Thus far, we have only considered cross-sectional data.

We now add a spatial and temporal dimension to our visualizations.

We can obtain aggregated crime data broken down by municipality and year from the Dutch Census Bureau, the CBS.

http://opendata.cbs.nl/dataportaal/portal.html?_catalog=CBS&_la=nl&tableId=83363NED&_theme=494

and read in the csv-file

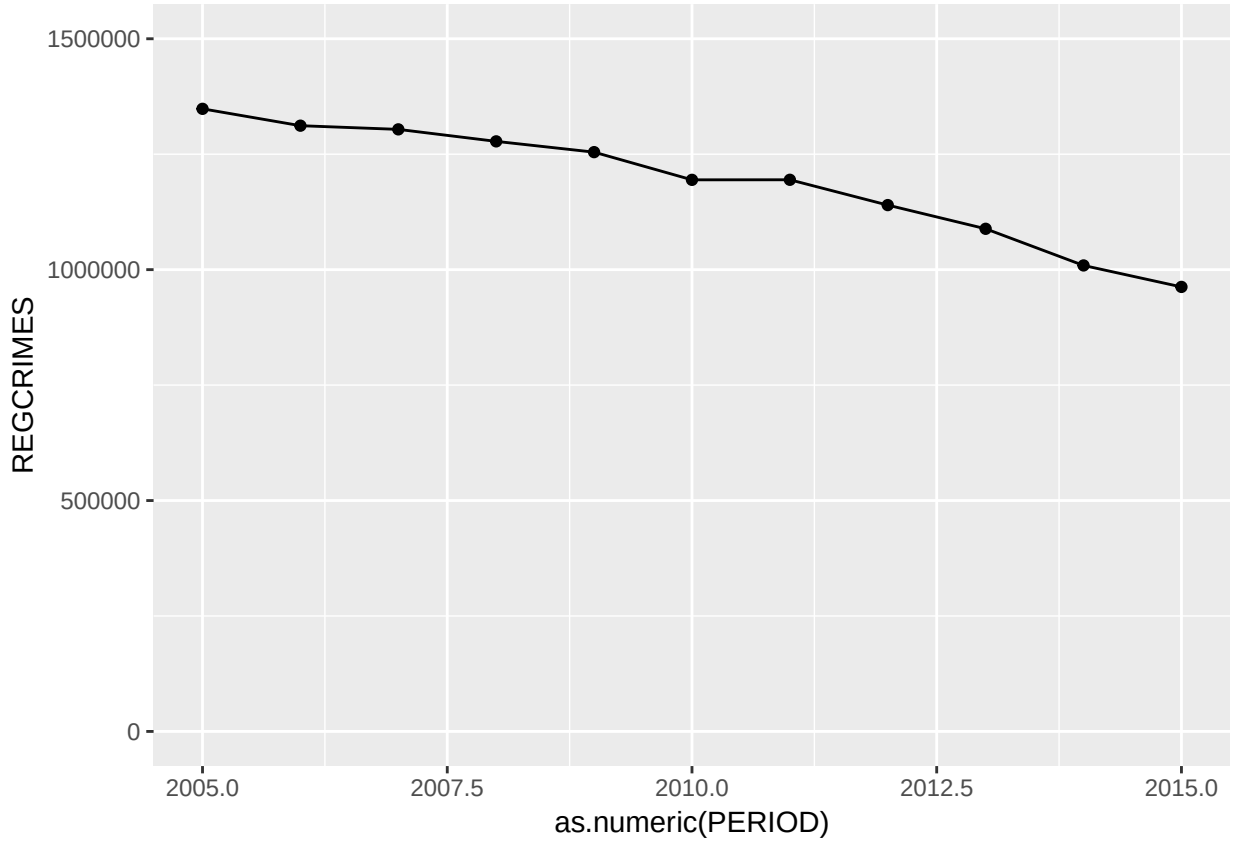
```
dat2 <- read.csv(file.path(path,"83363NED_UntypedDataSet_10112016_143641.csv"), sep = ";")
names(dat2) <- c("ID","CRIMETYPE","MUNICIPALITY","PERIOD","REGCRIMES")
dat2$PERIOD <- substr(dat2$PERIOD,1,4)
summary(dat2)
```

```
##        ID         CRIMETYPE  MUNICIPALITY     PERIOD
##  Min.   : 209   Min.   :0   GM0003 :  11   Length:4323
##  1st Qu.:1290   1st Qu.:0   GM0005 :  11   Class :character
##  Median :2370   Median :0   GM0007 :  11   Mode  :character
##  Mean   :2370   Mean   :0   GM0009 :  11
##  3rd Qu.:3450   3rd Qu.:0   GM0010 :  11
```

```
##  Max.   :4531   Max.   :0   GM0014 :  11
##                             (Other):4257
##    REGCRIMES
##  Min.   :     0
##  1st Qu.:   735
##  Median :  1245
##  Mean   :  3027
##  3rd Qu.:  2450
##  Max.   :105595
##
```

Let's plot a trendline of crimes in the Netherlands.

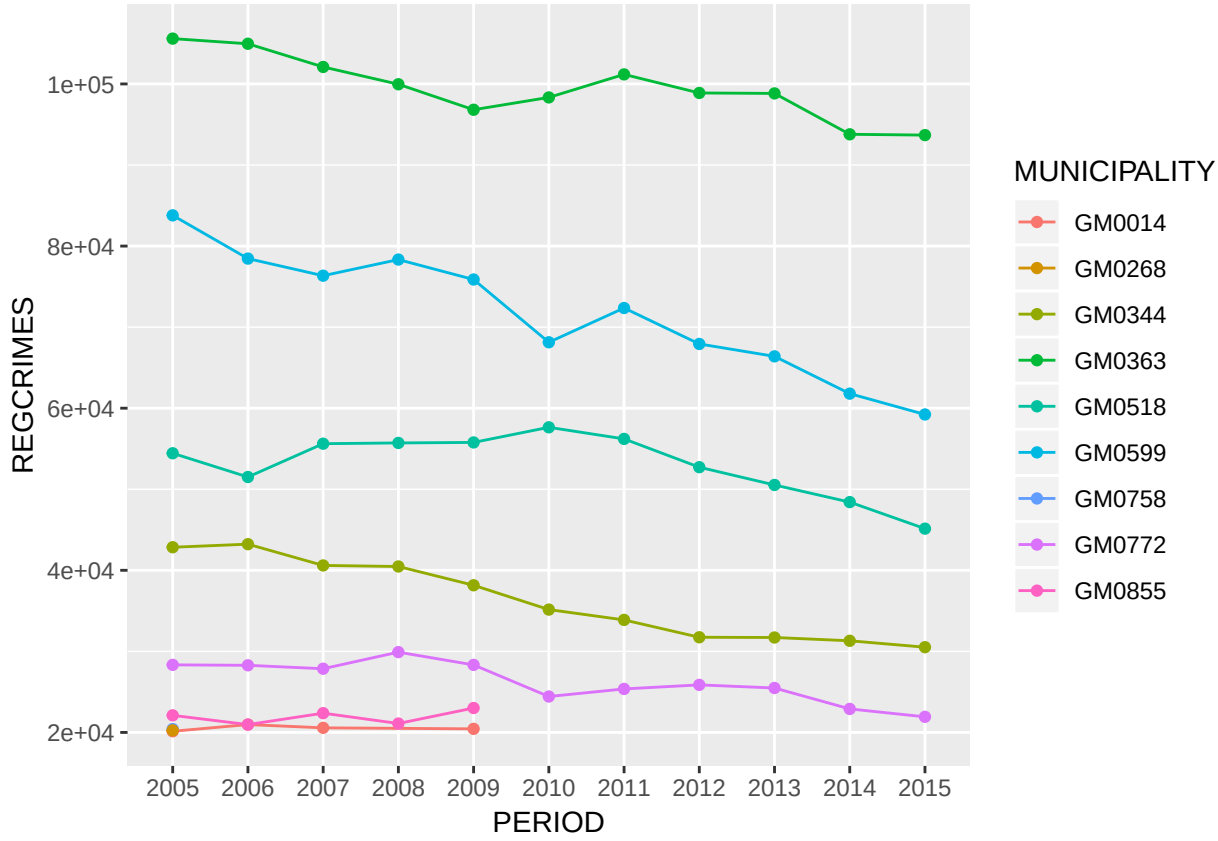
```
trends <- aggregate(REGCRIMES ~ PERIOD, data = dat2, sum)
ggplot(trends, aes(y = REGCRIMES, x = as.numeric(PERIOD))) +
  ylim(0, 15e5) + # let axis start at 0! NB: 15e5 == 15 x 10^5 == 1500000
  geom_line() +
  geom_point()
```



Using the package `dplyr` we can easily manipulate our data with the pipe operator `%>%`. Both `ggplot2` and `dplyr` are part of the `tidyverse`, a set of packages that play nice together, programmed by Hadley Wickham. We will use it to plot the trendline for crimes in locations with more than 20000 crimes per year.

```
library(dplyr)
dat2 %>%
  filter(REGCRIMES > 20000) %>%
  ggplot(aes(PERIOD, REGCRIMES, group = MUNICIPALITY, color = MUNICIPALITY)) +
  geom_line() +
```

```
  geom_point()
```



If you're interested in `dplyr`, look it up on the internet!

Now we will dive into plotting geographic data in a map.

First, we obtain files that contain the division of the Netherlands into municipalities from the following website:

http://www.twiav.nl/files/NL_Gemeenten2014.zip.

We already downloaded the shape files for municipality.

Read the shape files into memory, and plot it.

```
library(rgdal)
dsn <- file.path(path, "shapefiles2")
municip.shp <- readOGR(dsn = dsn, layer = 'NL_Gemeenten2014')
```

```
## OGR data source with driver: MapInfo File
## Source: "C:\Users\koenn\OneDrive\School\DAV\Notebook 1\shapefiles2", layer: "NL_Gemeenten2014"
## with 403 features
## It has 34 fields
```

```
plot(municip.shp)
```

This is the shape of all Dutch municipalities in 2014. Now we can incorporate municipality level statistics by joining the statistics file with the shape file. Then we can derive two new variables:

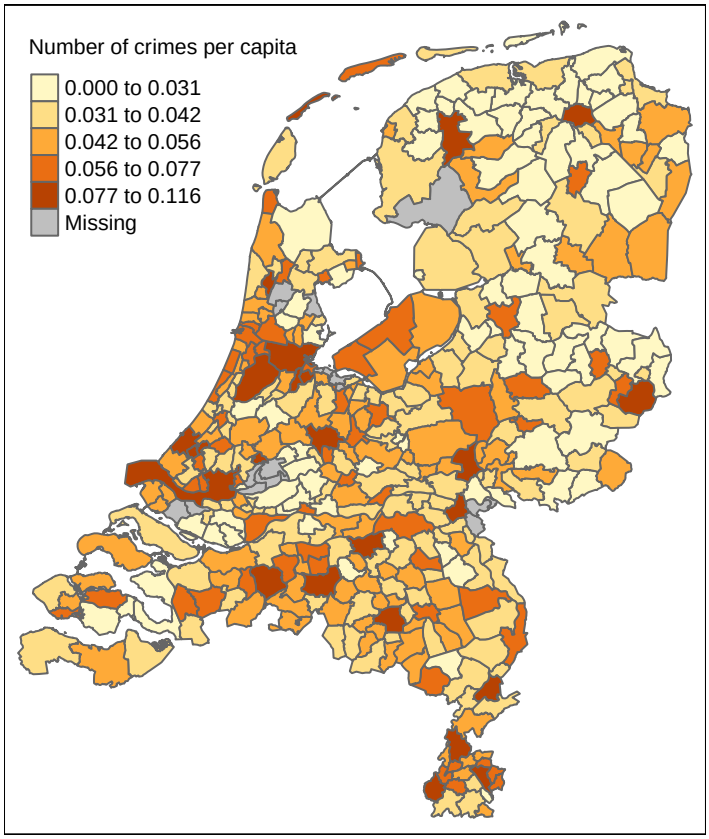
- The number of crimes per inhabitant
- The number of crimes per square kilometer

```
regcrimes <-
  dat2 %>%
  mutate(CBSCODE = as.character(MUNICIPALITY)) %>%
  filter(PERIOD == 2014) %>%
  select(REGCRIMES, CBSCODE) %>%
  right_join(data.frame(CBSCODE = as.character(municip.shp$CBSCODE)),
             by = "CBSCODE")
```

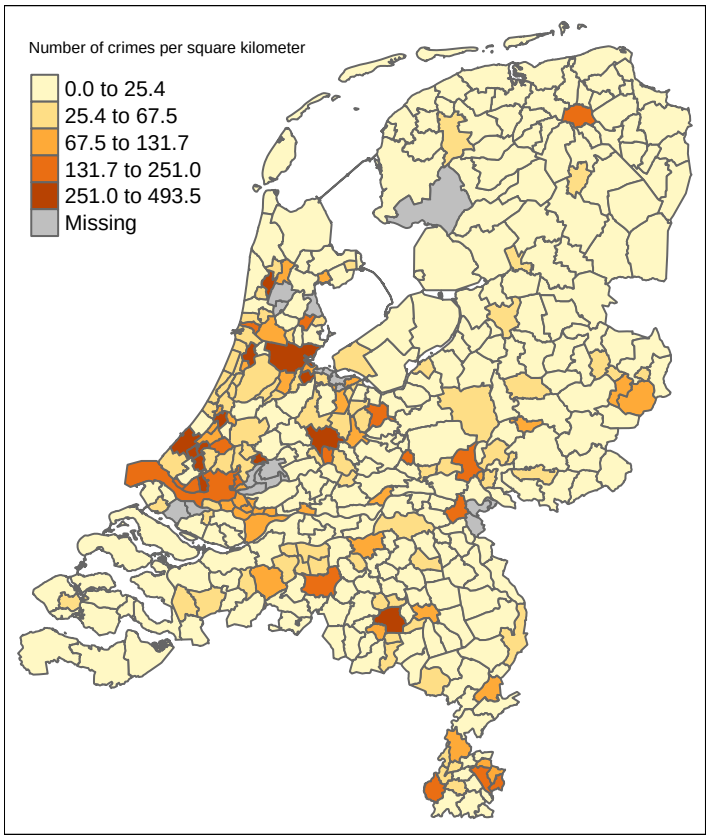
```
## Warning: Column `CBSCODE` joining character vector and factor, coercing
## into character vector
```

```
# Adding relevant variables to the shape object.
municip.shp$regcrimes <- regcrimes$REGCRIMES

municip.shp$crimespcapita <- # Number of crimes per capita
  as.numeric(municip.shp$regcrimes) / as.numeric(municip.shp$Inwoners)

municip.shp$crimespkm <- # Number of crimes per square kilometer
  as.numeric(municip.shp$regcrimes) / as.numeric(municip.shp$Oppervlakte)
```

```
# plot the map of crimes per capita
library(tmap)
qtm(shp = municip.shp, fill = "crimespcapita", fill.style = "kmeans",
    fill.title = "Number of crimes per capita")
```




```
qtm(shp = municip.shp, fill = "crimespkm", fill.style = "kmeans", fill.title = "Number of crimes per sq
```



The number of crimes per km^2 gives a radically different result than the number of crimes per inhabitant. Note that the crimes per km^2 map simply resembles a map of the major urban centers in the Netherlands, which seems logical!

Which municipalities have the highest score on that statistic?

```
# List the municipalities with the highest number of crimes per km (in 2014)
municip.shp@data %>%
  filter(crimespkm > 250) %>%
  select(Gemeentenaam, crimespkm) %>% rename(MUNICIPALITY = Gemeentenaam) %>%
  arrange(desc(crimespkm)) %>%  # Sort descending by crimespkm
  print
```

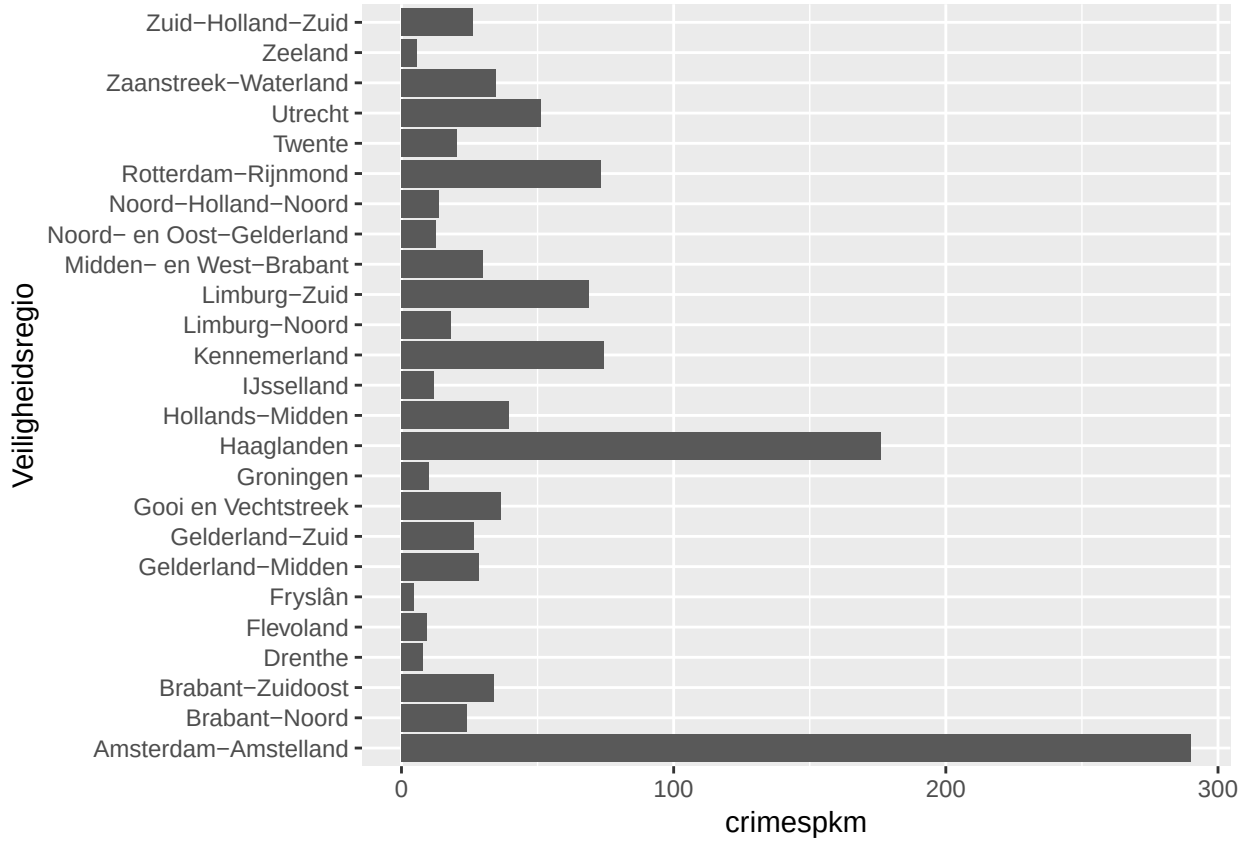
```
##     MUNICIPALITY crimespkm
## 1  's-Gravenhage  493.4767
## 2      Amsterdam  427.3088
## 3         Leiden  381.1775
## 4          Gouda  340.1436
## 5        Haarlem  332.3465
## 6        Utrecht  315.5428
## 7          Delft  291.9784
## 8       Schiedam  272.6586
## 9        Alkmaar  260.8974
## 10     Eindhoven  257.6798
## 11      Rijswijk  257.0738
```

The city in which the government is seated has the highest crime rate per square kilometer, followed by the capital of the Netherlands. Utrecht ranks sixth on this statistic.

The Netherlands are divided into so-called 'safety regions' (i.e. *veiligheidsregio's*).

Let's find out how this statistic is divided along these regions.

```
t1 <- aggregate(regcrimes ~ Veiligheidsregio_naam, municip.shp@data, sum)
t2 <- aggregate(Oppervlakte ~ Veiligheidsregio_naam, municip.shp@data, sum)
df <- data.frame(Veiligheidsregio = t1$Veiligheidsregio_naam, crimespkm = (t1[,2]/t2[,2]))
df %>%
  ggplot(aes(Veiligheidsregio, crimespkm))  +
       geom_bar(stat = "identity") + coord_flip()
```



If fear of crime is an issue, you should consider living in FryslÃ¢n or Zeeland.

You can now try plotting your own statistics from `municip.shp@data`.