dviz-practice

1 README

- This assignment is based on Pearson's archived DataCamp course "Data Visualization in R" (2016), chapter 2, "Different plot types".
- For the exercises, create or complete R code blocks as needed. If you use the Org-mode header `#+PROPERTY` and `#+STATUS` remember that you have may have to activate these with `C-c C-c`.
- When you completed all exercises and watched all videos, update the `#+AUTHOR:` information with your name and `(pledged)`, and submit.

2 TODO Characterizing a single variable (video)

- Watch the video (4 min) and the PDF slides (GDrive).

3 TODO The `hist` and `truehist` functions

Histograms are probably the best-known way of looking at how the values of a numerical variable are distributed over their range, and R provides several different histogram implementations.

The purpose of this exercise is to introduce two of these:

`hist` is part of base R and its default option yields a histogram based on the number of times a record falls into each of the bins on which the histogram is based. `truehist` is from the `MASS` package and scales these counts to give an estimate of the probability density.

1. Use the `par` function to set the `mfrow` parameter for a side-by-side array of two plots and display the value of `mfrow`.

```
par(mfrow=c(1,2))  # you can also use mfrow=2
par()$mfrow
```

```
[1] 1 2
```

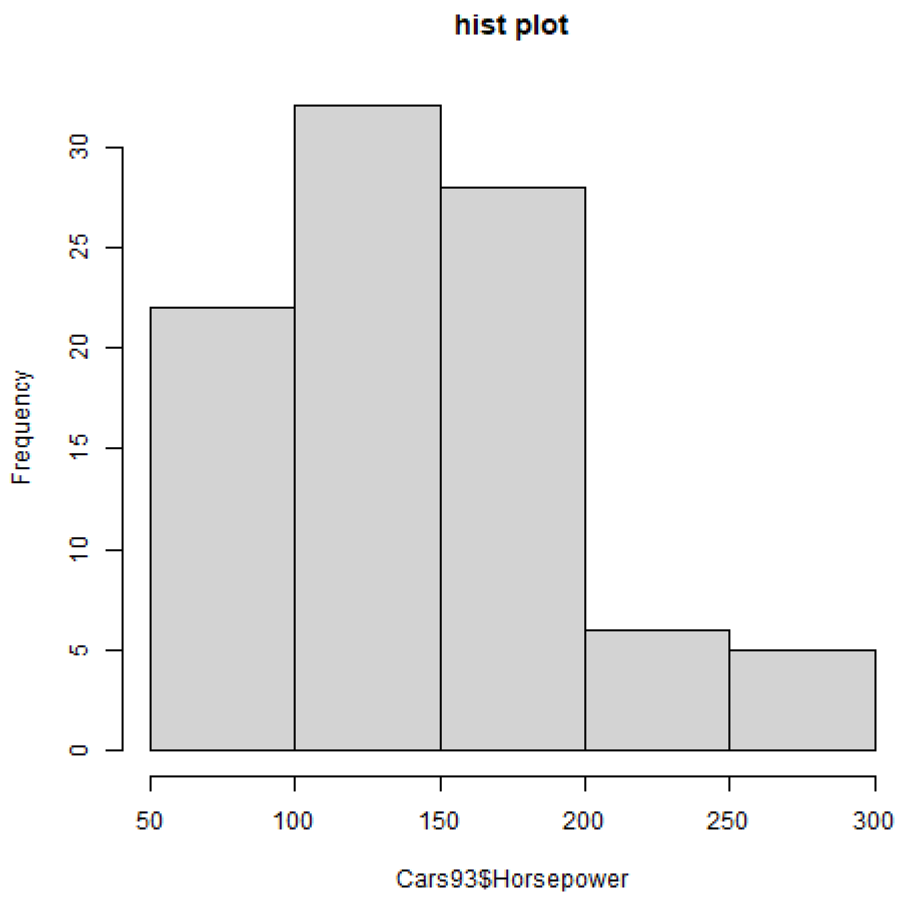
   1. Get an overview of `Cars93`.

```
str(Cars93)
```

```
'data.frame':	93 obs. of  27 variables:
 $ Manufacturer      : Factor w/ 32 levels "Acura","Audi",..: 1 1 2 2 3 4 4 4 4 5 ..
 $ Model             : Factor w/ 93 levels "100","190E","240",..: 49 56 9 1 6 24 54
 $ Type              : Factor w/ 6 levels "Compact","Large",..: 4 3 1 3 3 3 2 2 3 2
 $ Min.Price         : num  12.9 29.2 25.9 30.8 23.7 14.2 19.9 22.6 26.3 33 ...
 $ Price             : num  15.9 33.9 29.1 37.7 30 15.7 20.8 23.7 26.3 34.7 ...
 $ Max.Price         : num  18.8 38.7 32.3 44.6 36.2 17.3 21.7 24.9 26.3 36.3 ...
 $ MPG.city          : int  25 18 20 19 22 22 19 16 19 16 ...
 $ MPG.highway       : int  31 25 26 26 30 31 28 25 27 25 ...
 $ AirBags           : Factor w/ 3 levels "Driver & Passenger",..: 3 1 2 1 2 2 2 2 2
 $ DriveTrain        : Factor w/ 3 levels "4WD","Front",..: 2 2 2 2 3 2 2 3 2 2 ...
 $ Cylinders         : Factor w/ 6 levels "3","4","5","6",..: 2 4 4 4 2 2 4 4 4 5 ..
 $ EngineSize        : num  1.8 3.2 2.8 2.8 3.5 2.2 3.8 5.7 3.8 4.9 ...
 $ Horsepower        : int  140 200 172 172 208 110 170 180 170 200 ...
 $ RPM               : int  6300 5500 5500 5500 5700 5200 4800 4000 4800 4100 ...
 $ Rev.per.mile      : int  2890 2335 2280 2535 2545 2565 1570 1320 1690 1510 ...
 $ Man.trans.avail   : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 1 1 1 1 1 ...
 $ Fuel.tank.capacity: num  13.2 18 16.9 21.1 21.1 16.4 18 23 18.8 18 ...
 $ Passengers        : int  5 5 5 6 4 6 6 6 5 6 ...
 $ Length            : int  177 195 180 193 186 189 200 216 198 206 ...
 $ Wheelbase         : int  102 115 102 106 109 105 111 116 108 114 ...
 $ Width             : int  68 71 67 70 69 69 74 78 73 73 ...
 $ Turn.circle       : int  37 38 37 37 39 41 42 45 41 43 ...
 $ Rear.seat.room    : num  26.5 30 28 31 27 28 30.5 30.5 26.5 35 ...
 $ Luggage.room      : int  11 15 14 17 13 16 17 21 14 18 ...
 $ Weight            : int  2705 3560 3375 3405 3640 2880 3470 4105 3495 3620 ...
 $ Origin            : Factor w/ 2 levels "USA","non-USA": 2 2 2 2 2 1 1 1 1 1 ...
 $ Make              : Factor w/ 93 levels "Acura Integra",..: 1 2 4 3 5 6 7 9 8 10
```

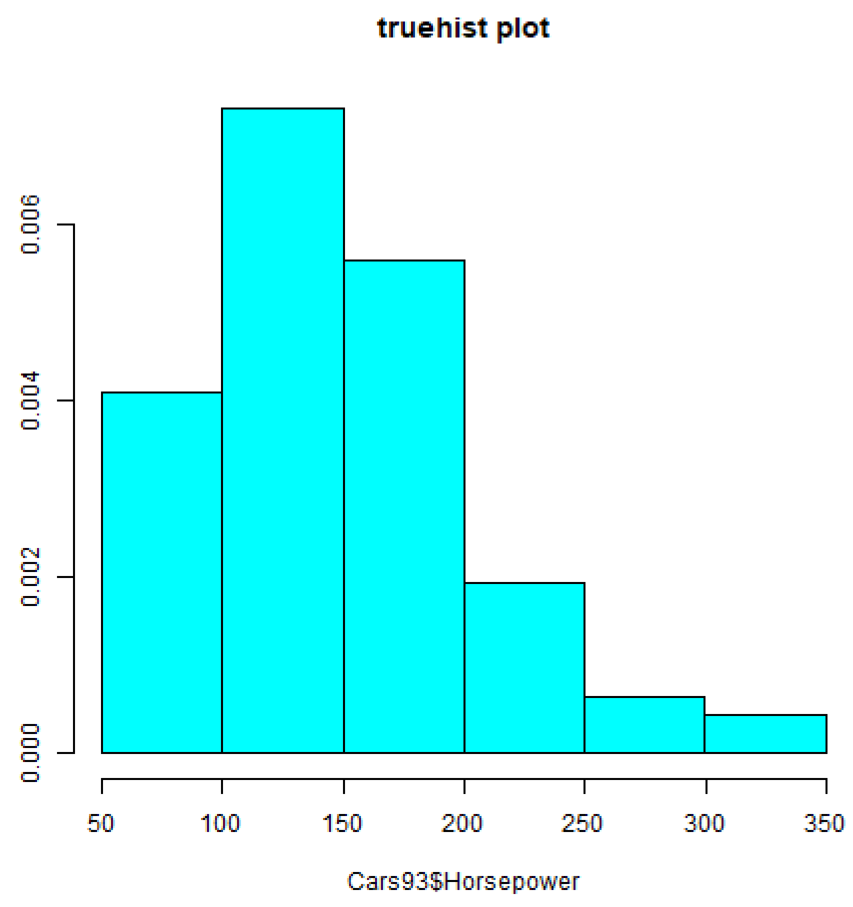
1. Use the `hist` function to generate a histogram of the `Horsepower` variable from the `Cars93` data frame in the `MASS` package. Set its `main` argument equal to the title of the plot `"hist plot"`.

```
hist(Cars93$Horsepower, main="hist plot")
```



2. Use the `truehist` function to generate an alternative histogram of the same variable. Title this plot `"truehist plot"` using its `main` argument.

```
truehist(Cars93$Horsepower, main="truehist plot")
```



4 TODO Density plots as smoothed histograms

While they are probably not as well known as the histogram, **density estimates** may be regarded as smoothed histograms, designed to give a better estimate of the density function for a random variable.

In this exercise, you'll use the built-in `ChickWeight` dataset, which contains a collection of chicks' weights. You will first select for the chicks that are 16 weeks old. Then, you'll create a histogram using the `truehist` function, and add its density plot on top, using the `lines` and `density` functions with their default options.

The density plot of this type of variable is often expected to conform approximately to the bell-shaped curve, otherwise known as the Gaussian distribution. Let's find out whether that's the case for this dataset.

1. Display the first few lines of `ChickWeight` to get an idea of the data set.

```
head(ChickWeight)
```

```
  weight Time Chick Diet
1     42    0     1    1
2     51    2     1    1
3     59    4     1    1
4     64    6     1    1
5     76    8     1    1
6     93   10     1    1
```

2. Create an index vector `index16` using the `which` function that selects records from the `ChickWeight` data frame with `Time` equal to 16.

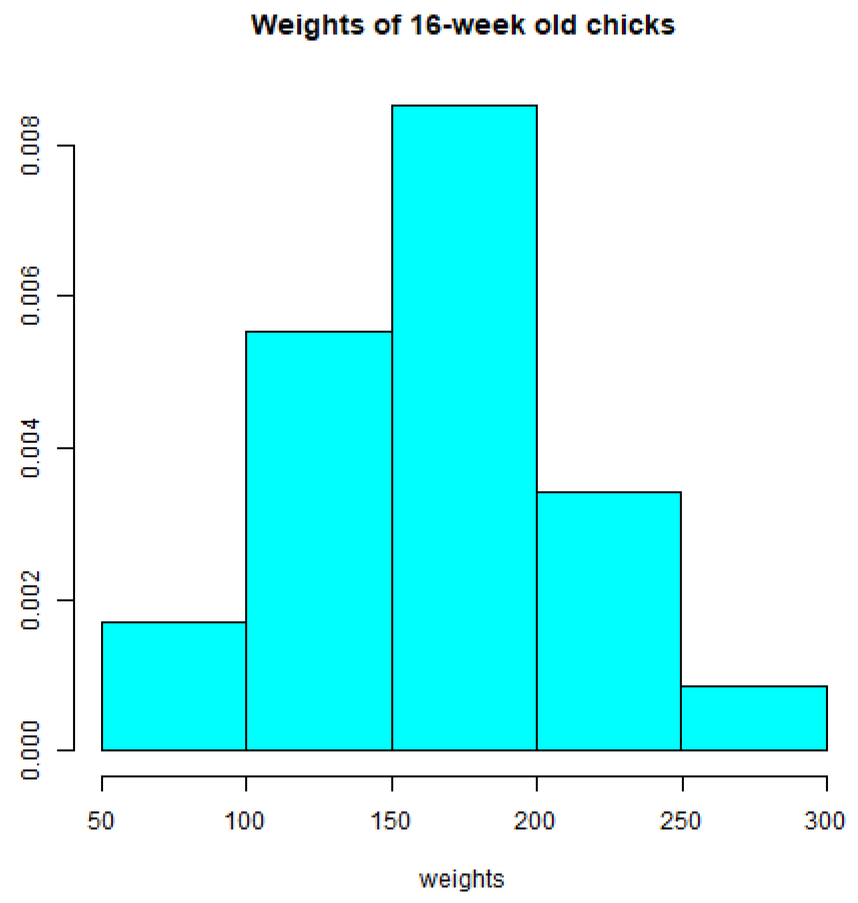
```
index16 <- which(ChickWeight$Time == 16)
```

3. Create a variable `weights` that gives the weights of the 16-week old chicks by using the index vector you just defined on the respective variable of `ChickWeight`.

```
weights <- ChickWeight$weight[index16]
```

4. Use `truehist` to generate a histogram from `weights` and title it `"Weights of 16-week old chicks"`.

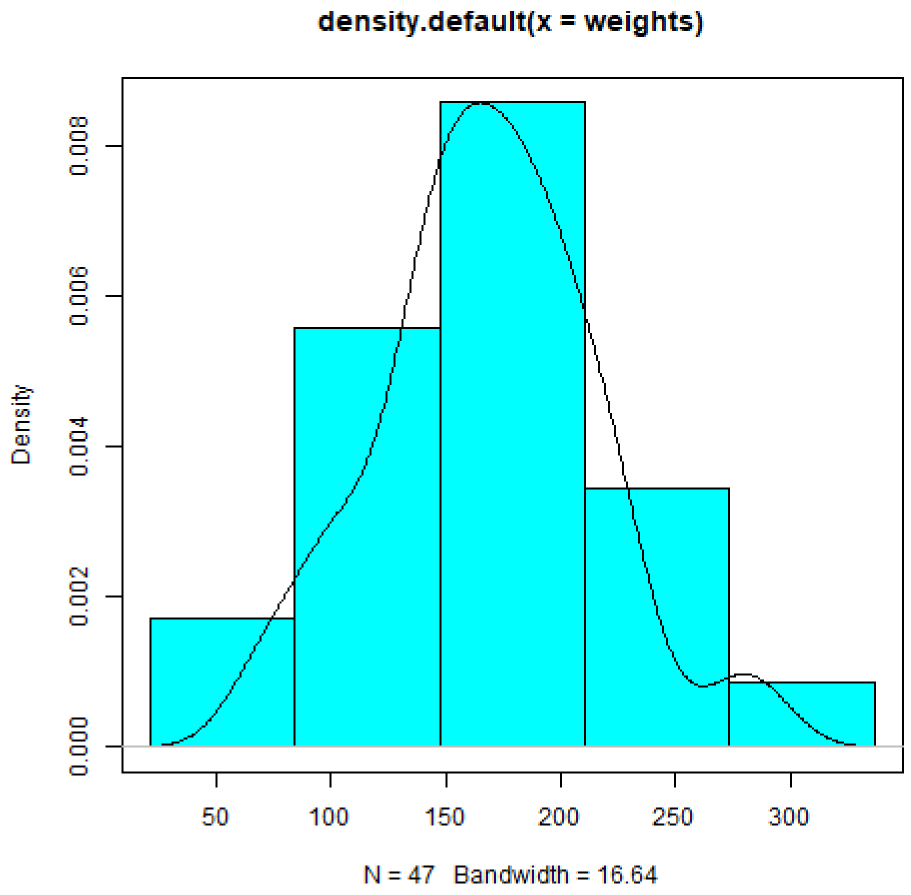
```
truehist(weights, main="Weights of 16-week old chicks")
```



5. Use the `lines` and `density` functions to overlay a density plot of the `weights` values on the histogram with three lines of code:

   1. Call the `truehist` function again for `weights`, but set the parameters `main` and `xlab` to `""` and the parameter `axes` to `FALSE`
   2. Call `par(new=TRUE)` to plot over the histogram
   3. Call `lines` with the argument `plot(density(weights))`

```
truehist(weights,main="",xlab="",axes=FALSE)
par(new=TRUE)
lines(plot(density(weights)))
```



5 TODO Using `qqPlot` to see many details in data

A practical limitation of both histograms and density estimates is that, if we want to know whether the Gaussian distribution assumption is reasonable for our data, it is difficult to tell.

The quantile-quantile plot, or QQ-plot, is a useful alternative. For such a plot, the data are sorted, then they are plotted against a specially-designed x-axis based on our reference distribution (e.g., the Gaussian "bell curve"), and finally we look to see whether the points lie approximately on a straight line.

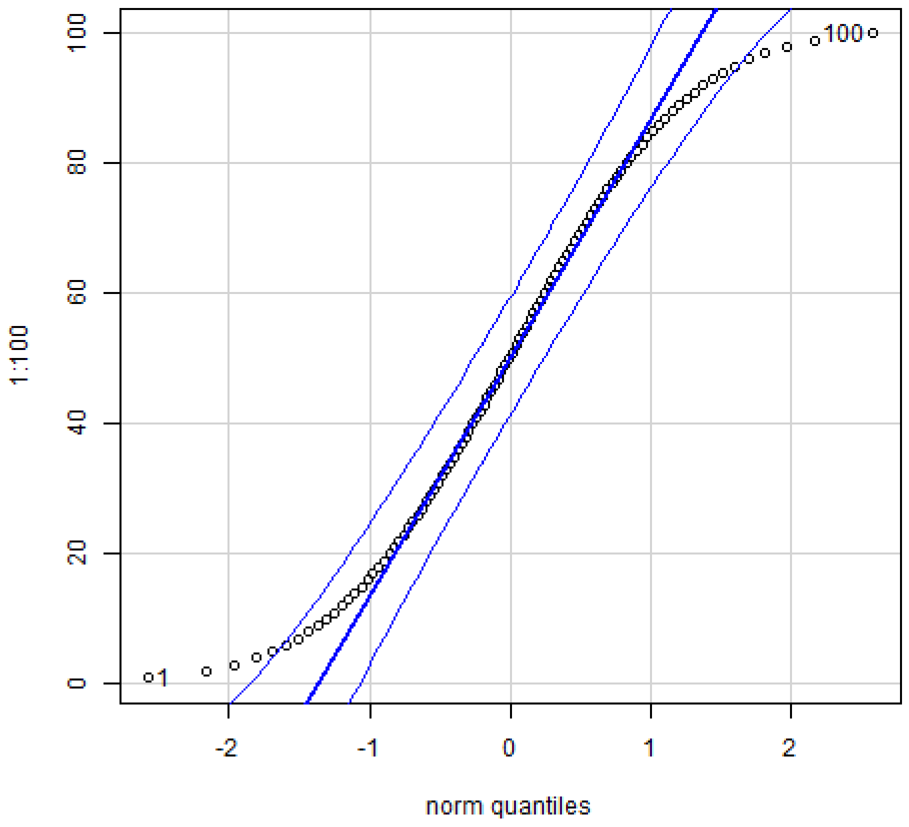
In R, several QQ-plot implementations are available, but the most convenient one is the `qqPlot` function in the `car` package.

6 TODO Using `qqPlot` to show that the Gaussian assumption is a good fit

The first part of this exercise applies this function to the 16-week chick weight data considered in the last exercise, to show that the Gaussian distribution appears to be reasonable here.

1. Load the `car` package to make the `qqPlot` function available for use, and call `qqPlot` on the vector `1:100`

```
library("car")
qqPlot(1:100)
```



2. Create an index vector `index16` using the `which` function that selects records from the `ChickWeight` data frame with `Time` equal `16`.

```
index16 <- which(ChickWeight$Time == 16)
```

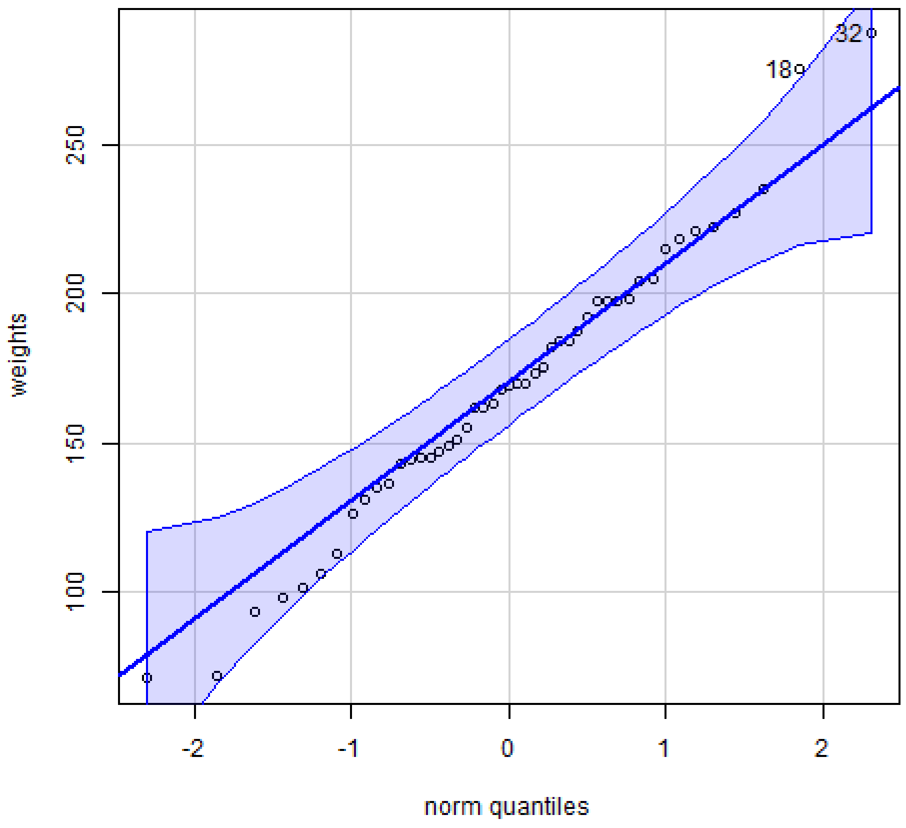
3. Create the variable `weights` that gives the weights of 16-week old chicks. Print the weights of the chicks that are heavier than 250 grams. Then print the indices of these outliers.

```
weights <- ChickWeight$weight[index16]
weights[weights>250]
which(weights>250)
```

```
[1] 275 287
[1] 18 32
```

4. Apply the `qqPlot` function to the `weights` data. Note that almost all points fall within the confidence intervals around the reference line. This indicates conformance with the Gaussian distribution. The indices of the outliers with weight greater than 250 grams are referenced.

```
qqPlot(weights)
```



7 TODO Using `qqPlot` to show that the Gaussian assumption is a poor fit

The second part of the exercise applies this function to another variable where the Gaussian distribution is obviously a poor fit, but the results also show the presence of repeated values (flat stretches in the plot) and portions of the data range where there are no observations (vertical "jumps" in the plot).

1. Make the `MASS` package available for use, load the `Boston` data frame from the `MASS` package, and look at the structure of the data frame.

```
library("MASS")
data("Boston")
str(Boston)
dim(Boston)
```

```
'data.frame':	506 obs. of  14 variables:
 $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
 $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
 $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
 $ chas   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
 $ rm     : num  6.58 6.42 7.18 7 7.15 ...
 $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
 $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
 $ rad    : int  1 2 2 3 3 3 5 5 5 5 ...
 $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
 $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
 $ black  : num  397 397 393 395 397 ...
 $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
 $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
[1] 506  14
```

2. The `Boston` data set contains 14 different housing values of 506 houses in suburbs of Boston, MA. Show the dimensions of the data frame using the function `dim` on the argument `Boston`.

```
dim(Boston)
```

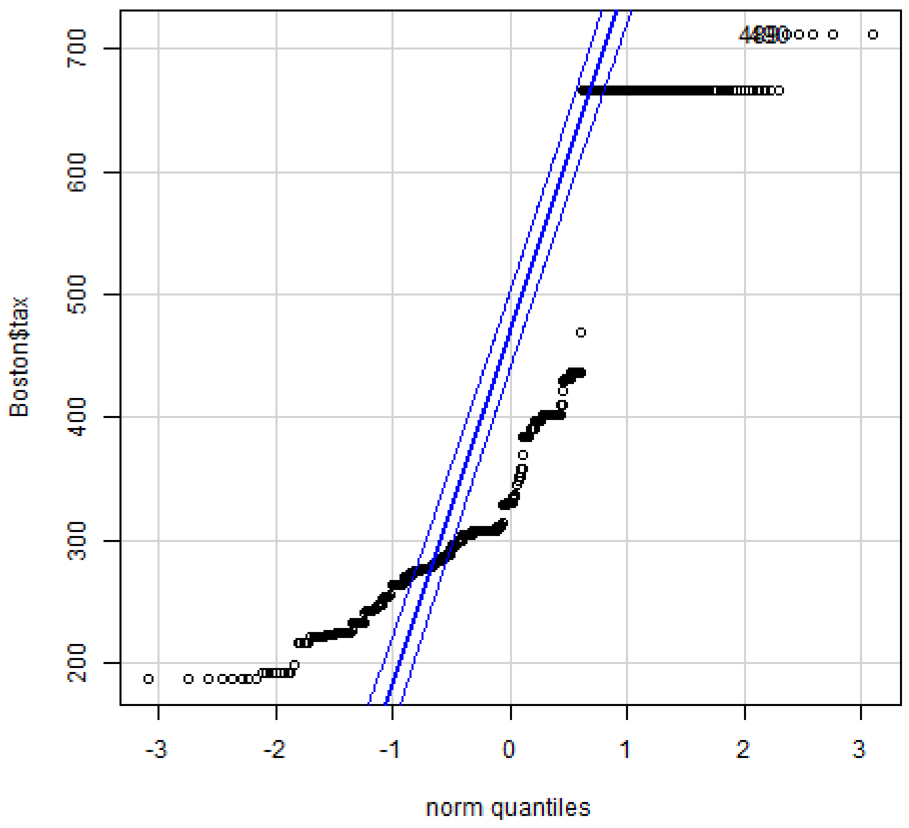
```
[1] 506  14
```

3. Make the `qqPlot` function available by loading the `car` package into the current session.

```
library("car")
```

4. Show the normal QQ-plot of the tax data from the `Boston` data frame. The result shows that the Gaussian assumption is not justified for this data set. Horizontal stretches in the plot indicate repeated values, and vertical jumps indicate missing observations - the evenness of the bell curve is lost here.

```
qqPlot(Boston$tax)
```



Created: 2022-10-28 Fri 16:31