Practical Exercise 1

Niels Richard Hansen

September 4, 2017

This exercise uses the data set `infrared`, which is also used in Problem 10.1 in the book *Computational Statistics*. Start by locating this data set from the book homepage and load it into R using the `read.table` function. As in Problem 10.1 you will throughout study the distribution of the logarithm of the variable called F12.

The purpose of this exercise is two-fold. First, you will get familiar with the data and see how different choices of visualizations using histograms can affect your interpretation of the data. Second, you will learn more about how to write functions in R and gain a better understanding of how they work.

Problem 1.1

Plot a histogram of `log(F12)` using the default value of the argument `breaks`. Experiment with alternative values of `breaks`.

Problem 1.2

Write your own function, called `myBreaks`, which takes two arguments, `x` (a vector) and `h` (a positive integer). Let `h` have default value `5`. The function should first sort `x` into increasing order and then return the vector that: starts with the smallest entry in `x`; contains every hth unique entry from the sorted `x`; ends with the largest entry in `x`.

For example, if `h = 2` and `x = c(1, 3, 2, 5, 10, 11, 1, 1, 3)` the function should return `c(1, 3, 10, 11)`. To see this, first sort `x`, which gives the vector `c(1, 1, 1, 2, 3, 3, 5, 10, 11)`, whose unique values are `c(1, 2, 3, 5, 10, 11)`. Every second unique entry is `c(1, 3, 10)`, and then the largest entry `11` is concatenated.

Hint: The functions `sort` and `unique` can be useful.

Use your function to construct *breakpoints* for the histogram for different values of `h`, and compare with the histograms obtained in Problem 1.1.

Problem 1.3

If there are no ties in the data set, the function above with produce breakpoints with `h` observations in the interval between two consecutive breakpoints (except the last two perhaps). If there are ties, the function will by construction return unique breakpoints, but there may be more than `h` observations in some intervals.

The intention is now to rewrite `myBreaks` so that if possible each interval contains `h` observations.

Modify the `myBreaks` function with this intention and so that is has the following properties:

- All breakpoints must be unique.
- The range of the breakpoints must cover the range of `x`.
- For two subsequent breakpoints, a and b, there must be at least `h` observations in the interval $(a, b]$, provided `h < length(x)`. (With the exception that for the first two breakpoints, the interval is $[a, b]$.)