Exercises for statistical inference and stuff

Timothée Bonnet

December 5, 2018

Contents

1 Statistical inference and random numbers

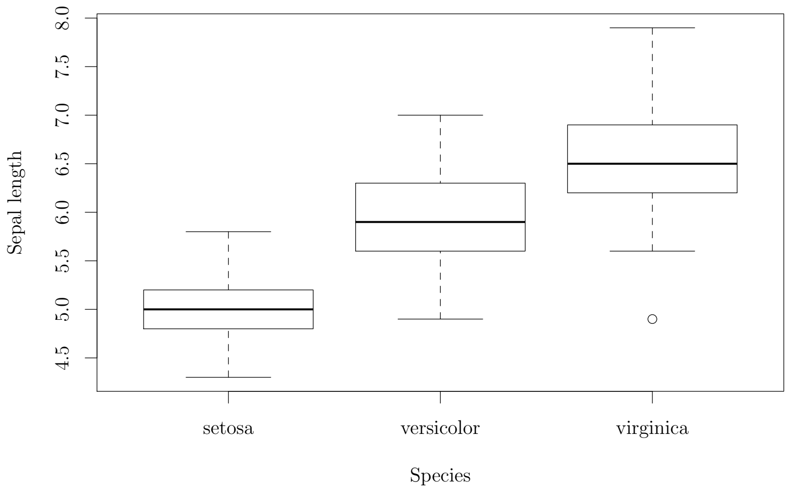
1.1 Iris

Some datasets are shipped with R (in R-base or in packages) and you can load them with the function data:

```
data("iris")
```

The dataset looks like that:

```
boxplot(Sepal.Length ~ Species,
    data = iris,
    drop = TRUE, ylab="Sepal length", xlab="Species")
```



2 R-studio tricks

2.1 Column selection

2.2 Short-cuts

3 Linear models

1. Load Cdata.csv, fit models of y predited by x1 and x2, or x2 and x3. Something is weird, what is going on? What to do?

2. For model that can be fitted with t.test, aov, and lm, is one of the function faster?

3. Write your own code to obtain a prediction from a lm (that is, a simpler version of the predict function), with confidence interval. (extra toughness: do it using the matrix formulation of the analytical solution to a linear model)

4 While-loop

4.1 What you need to know

```
    while(condition TRUE)
    {
      something
    }
```

For instance:

```
x <- 0
while(x<10)
    {
      x <- x+1
      print(x)
    }

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

4.2 Practice

The function sample() takes 5 number between 1 and 6 (like 5 dice!):

```
x <- sample(x = 1:6, size = 5, replace = TRUE)
```

Are all die equal?

```
all(x == x[1])

## [1] FALSE
```

Are they ever going to be equal?
Write a while loop to find a case with all die equal
How many attempts does it take
Write a for while loop within a for loop to estimate how long it take on average.

5 If-else statement

5.1 What you need to know

```
if(condition)
{
  do something
}
```

```
if(condition)
{
  do something
}else{
  do something else
}
```

For instance:

```
for (i in 1:10)
{
  if(i < 6)
  {
    print("tofu")
  }else{
    print("bacon")
  }
}

## [1] "tofu"
## [1] "tofu"
## [1] "tofu"
## [1] "tofu"
## [1] "tofu"
## [1] "bacon"
## [1] "bacon"
## [1] "bacon"
## [1] "bacon"
## [1] "bacon"
```

5.2 Practice

We can draw 100 random number following a random distribution of mean 0 and variance one with:

```
x <- rnorm(n = 100, mean = 0, sd = 1)
```

If we take their logarithm we obtain many “NaN” (Not A Number), because the log of a negative number is undefined:

```
log(x)

## Warning in log(x):  NaNs produced

##   [1]          NaN          NaN -0.589344163          NaN          NaN
##   [6] -1.664655179          NaN          NaN  0.341789894          NaN
##  [11] -0.794460074          NaN -0.984556273          NaN -0.279575110
##  [16]          NaN -1.899465646          NaN  0.187291755          NaN
##  [21]  0.108289852 -1.102364730          NaN          NaN -0.327273048
##  [26] -2.285902113          NaN -1.282928209          NaN  0.778312086
##  [31] -0.901221503          NaN          NaN          NaN          NaN
##  [36]          NaN  0.003695511          NaN          NaN -0.413186144
##  [41] -0.810298198          NaN -1.212548705          NaN          NaN
##  [46]  0.085032800          NaN          NaN  0.087356314 -2.332427983
##  [51] -0.148211628          NaN          NaN          NaN -1.043687298
##  [56]          NaN          NaN -1.265321312          NaN          NaN
##  [61]          NaN -0.100223277          NaN -0.641165705          NaN
##  [66]          NaN -1.545501844 -2.355299437          NaN -0.349517255
##  [71]  0.891404300          NaN          NaN  0.345827362 -1.662390625
##  [76] -0.289272687          NaN -0.335576748          NaN -2.269379373
##  [81]          NaN -0.825110689 -0.032402413          NaN          NaN
##  [86] -1.222536575 -0.363257036          NaN  0.060941438 -0.238537886
##  [91] -0.683101916 -1.288282157 -1.278254302          NaN -0.126683029
##  [96]          NaN -2.328577470          NaN -0.015353381          NaN
```

Let’s say we want 0 instead of NaN.

Use a for loop and an if-else statement to do that.

More difficult: Use a for loop and a while loop to re-draw random numbers until they are all positive.