STA238 Tutorial 1

Luis Ledesma

2023-01-25

Announcements

- You can upload your work on Crowdmark from the end of the tutorial session to 5pm Friday of that week.
- All questions must be solved using RStudio.

Getting started

In order to run R on your computer, you need to carry out the following steps:

1. Install R
2. Install RStudio

Alternatively, you can use RStudio on Jupyter notebooks for your work (I would still highly recommend that you install R and RStudio on your own computer).

Remark: Please make sure that you have properly set up R and RStudio in your computer!

Knowing how to use R and RStudio

An important part of R are packages. These are 'libraries' that can be imported into our instance of R to enable various auxiliary functions. If we want to install and use the package `tidyverse`, one would write:

```
## install.packages("tidyverse")
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.0      v purrr   0.3.5
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.3      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

The first line of code installs the package `tidyverse` (if we remove the comments indicated by `#`). The second line will load it into our instance of R.

Remark: Usually, it's good practice to **install R packages in the console**, and include the loading procedure in your R scripts or RMarkdowns.

Basic R functions and definitions

In R, one can assign values to a variable with `<-`:

```
var1 <- "a"
var2 <- 5
var3 <- TRUE
var1
```

```
## [1] "a"
```

```
var2
```

```
## [1] 5
```

```
var3
```

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

Moreover, one can also define vectors and matrices:

```
var4 <- numeric(length=2)
var5 <- matrix(data=0,nrow=2,ncol=2)
var4
```

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

```
var5
```

```
##      [,1] [,2]
## [1,]    0    0
## [2,]    0    0
```

The above chunk will initialize a vector of length 2 and 0s as entries, and a matrix of dimensions 2×2 and 0s as entries, which is based on the parameters indicated.

Remark: To seek help, you can write `?function` (e.g. if the function's name is `function`) in the console to make the help menu pop-up on the lower right-hand window in RStudio.

Moreover, one can also define logical statements, `if-else` statements, and loops as in other programming languages. See the base R cheatsheet for more details.

Simulating data from a probability distribution

In R, there are different functions related to the probability distributions (not just limited to the ones below):

1. `rnorm`: normal distribution
2. `rbinom`: binomial distribution
3. `rchisq`: chi-squared distribution
4. `rt`: t-distribution
5. `rpois`: Poisson distribution

For the normal distribution, one has the auxiliary functions:

1. `dnorm`: density function for the normal distribution (pdf)
2. `pnorm`: probability function for the normal distribution (CDF)
3. `qnorm`: quantiles from the normal distribution
4. `rnorm`: random numbers sampled from a normal distribution

These functions are also analogously present for the other probability distributions, see the help menu for the appropriate parameters that must be used.

The Central Limit Theorem

The CLT indicates that for X_i iid, with finite mean and variance:

$$\frac{\overline{X_n} - \mu}{\sigma/\sqrt{n}} \rightarrow_d N(0,1)$$

Where convergence is in distribution. How can we demonstrate this through a coding simulation?

Coding the Central Limit Theorem in R

Given our problem, we have a binomial with parameter $t = 5$ and $p = 0.1$. Thus, we would be simulating values from $\text{Bin}(5, 0.1)$. What would be the mean and variance of this random variable?

For fixed n, by properties of distributions (and limits), $\overline{X_n}$ should be close in distribution to a normal (not necessarily the standard normal).

Question: Look at the section 'Simulating data from a probability distribution'. Which function do you think would be the most appropriate to simulate data from for a binomial distribution?

Suppose that $n = 50$ and that we simulate 60000 times. Then, we can simulate the data in the following manner:

```
t <- 5
p <- 0.1
n <- 50
s <- 60000

rbinom(n,t,p)
```

```
##  [1] 0 1 1 1 0 0 1 2 0 0 1 1 0 1 1 0 0 1 1 0 0 0 1 1 1 0 0 0 2 0 0 0 1 0 1 1 1 1
## [39] 0 0 0 0 0 1 0 3 0 0 2 1
```

```
mean(rbinom(n,t,p))
```

```
## [1] 0.52
```

The last line in this chunk of code should be a realization of $\overline{X_n}$. We now want to repeat this 60000 times, visualize the empirical distribution, and compare the shape of the density function with the one of a normal (should be roughly getting a bell curve).

We can initialize an empty vector and empty matrix that will store our values, each column will correspond to a simulation vector from the $s = 60000$ simulations, so we should have a $n \times s$ matrix.

```
SampleMeans <- numeric(s)
SimValues <- matrix(0, n, s)
```

We can iterate on the columns, and compute the sample mean of each of them, and store it in the vector `SampleMeans`:

```
for (i in 1:s){
  SimValues[,i] <- rbinom(n, t, p)
  SampleMeans[i] <- mean(SimValues[,i])
}
```

We now have multiple simulations from $\overline{X_n}$. If we were to compute a histogram of these values, we should be getting the empirical density function for the random variable $\overline{X_n}$:

```r
hist(SampleMeans, breaks = 30)
```