Stat 133, Fall 2016, Simulations

Gaston Sanchez

October 28, 2016

Introduction

The goals of this lab are:

- getting started with simulations in R
- learn how to create a basic shiny app
- put in practice concepts from your introductory statistics course(s)

The idea is to simulate tossing a coin n times in order to compute:

- the total number of heads and tails in n tosses
- the relative frequencies of heads and tails
- how does the relative frequencies (of heads) varies over the series of tosses

From the conceptual point of view, you will use a shiny app to visualize the notion of **chance error** when simulating tossing a coin a given number of times.

Simulating Tossing a coin

To toss a coin using R, we first need an object that plays the role of a coin. So let's start by creating a `coin` object using a character vector with two elements: `"heads"` and `"tails"`:

```
# coin object
coin <- c("heads", "tails")
```

Tossing a coin is a random experiment: you either get heads or tails. To get a random output in R we can use the function `sample()` which takes a random sample of a given vector. Here's how to simulate a coin toss using `sample()` to take a random sample of size 1 from `coin`:

```
# one toss
sample(coin, size = 1)
```

```
## [1] "tails"
```

You can also use `sample()` to take samples of sizes different than one, and also to sample with replacement. To simulate multiple tosses, change the value of the `size` argument, and set `replace = TRUE`:

```
# 3 tosses
sample(coin, size = 3, replace = TRUE)
```

```
## [1] "heads" "heads" "heads"
```

```
# 6 tosses
sample(coin, size = 6, replace = TRUE)
```

```
## [1] "heads" "tails" "tails" "heads" "tails" "heads"
```

Loaded Coins

The function `sample()` also has an argument `prob` that lets you set different probabilities for each element in the provided vector. By default, `prob = NULL` means using a uniform probability. In the case of your `coin`, this means that `heads` and `tails` have a 0.5 chance of being sampled (i.e. a *fair* coin).

To simulate a loaded coin, you can specify `prob = c(0.4, 0.6)`:

```
# tossing a loaded coin
sample(coin, size = 5, prob = c(0.4, 0.6), replace = TRUE)
```

```
## [1] "heads" "tails" "tails" "tails" "tails"
```

Function `toss()`

To make your code reusable, it's better to create a function that lets you `toss()` a coin multiple times:

```
toss <- function(x = c("heads", "tables"), times = 1, prob = c(0.5, 0.5)) {
  sample(x, size = times, replace = TRUE, prob = prob)
}
```

Test it:

```
toss(times = 10)
```

```
##  [1] "heads"  "heads"  "heads"  "tables" "tables" "tables" "tables"
##  [8] "tables" "heads"  "heads"
```

Frequencies and Relative Frequencies

Typical probability problems that have to do with coin tossing, require to compute the total proportion of `"heads"` and `"tails"`:

```
# five tosses
five <- toss(coin, times = 5)

# total frequencies of heads and tails
sum(five == "heads")
```

```
## [1] 2
```

```
sum(five == "tails")
```

```
## [1] 3
```

It is also customary to compute the relative frequencies of `"heads"` and `"tails"` in a series of tosses:

```r
# relative frequencies of heads
cumsum(five == "heads") / 1:length(five)
```

```
## [1] 1.0000000 0.5000000 0.3333333 0.5000000 0.4000000
```

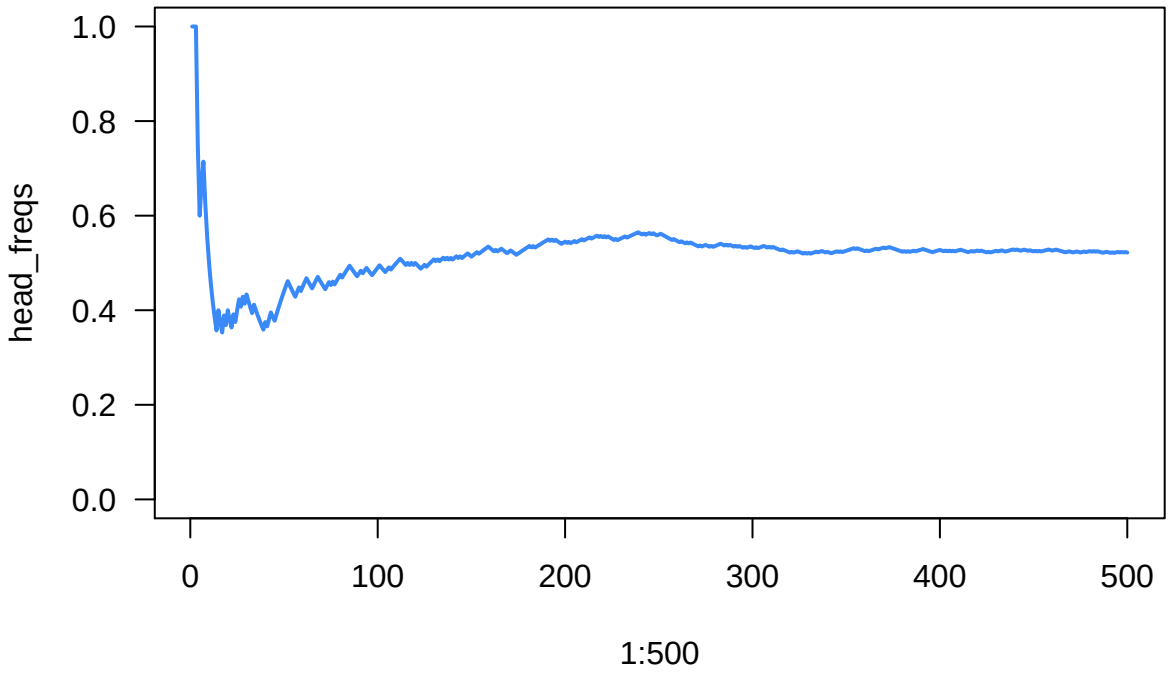
```r
# relative frequencies of tails
cumsum(five == "tails") / 1:length(five)
```

```
## [1] 0.0000000 0.5000000 0.6666667 0.5000000 0.6000000
```

Likewise, it is common to look at how the relative frequencies of heads or tails change over a series of tosses:

```r
set.seed(5938)
hundreds <- toss(coin, times = 500)
head_freqs = cumsum(hundreds == "heads") / 1:500

plot(1:500, head_freqs, type = "l", ylim = c(0, 1), las = 1,
     col = "#3989f8", lwd = 2)
```



So far we have written code in R that simulates tossing a coin one or more times. We have included commands to compute proportion of heads and tails, as well the relative frequencies of heads (or tails) in a series of tosses. In addition, we have produced a plot of the relative frequencies and see how, as the number of tosses increases, the frequency of heads (and tails) approach 0.5.

Shiny App

- Open RStudio.
- Go to the **File** option from the menu bar.
- Select **New File** and choose **Shiny Web App**.

- Give a name to your App, choose a location for it, and click the **Create** button.

These steps should create a new folder in the specified directory containing an R script file called `app.R`. This file contains a basic template with the following main ingredients:

- a call to `library(shiny)` at the top of the file
- the User Interface "function" `ui <- fluidPage(...)`
- the Server "function" `server <- function(input, output) {...}`
- a call to `shinyApp(ui = ui, server = server)` to run your app

By default, shiny creates a basic template with a histogram of the variable `waiting` from the data set `faithful`. You can try running the app by clicking on the **Run App** button (see buttons at the top of the source pane).

App scripts

Instead of using the default `app.R` script, you will be playing with your own scripts to simulate the coin tossing experiment.

In the folder of this lab, you will find several app R scripts: `app1.R`, `app2.R`, `app3.R`, and `app4.R`. Each of them adds a new element to the sidebar, so that your app becomes more flexible.

- `app1.R`: basic skeleton that includes input for number of tosses
- `app2.R`: includes input for probability of heads
- `app3.R`: includes input for random seed

The file `app4.R` is a bit more complex. First, we redefine `toss()` by adding another argument for the random seed. Notice also the use of `reactive()` to create reactive objects `tosses()` and `proportions()`. Likewise, in the main panel of outputs, we display a data table showing summary results.