Introduction to R statistical package

Joseph J. Luczkovich, PhD

January 14, 2015

Abstract

This is a brief introduction to the use of a new, free statistical package called R. It is down-loadable from the website: http://www.r-project.org/ You will find this software difficult to use at first, because to use it you must learn to write "code", but after a while you will find it far easier to use than menu-driven software packages like SAS or SYSTAT. R is better, because R is free. And many statisticians are using it to develop and enhance statistical tests for their unique problems, which is a good thing, because you can use their code, too. Here we explore R and use some simple commands to add numbers, store results in variables, compute some simple probabilities based on coin-flipping simulations, and make some plots of data.

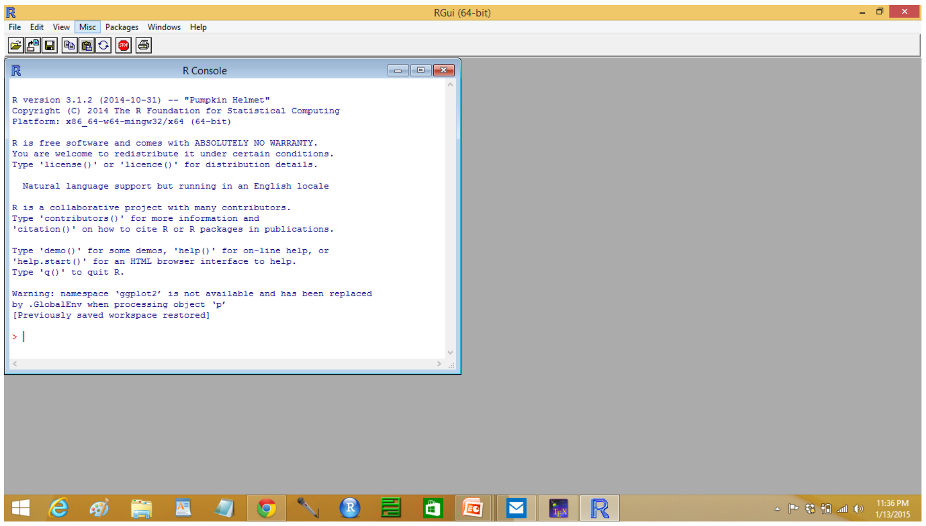
1 Introduction to R

R is a statistical package [1] that has been developed for free non-commercial use by students, professors, scientists and statisticians all over the world. Because it is free, it is extremely valuable to people who have little or no money for software licenses and subscriptions, like most students. Because it is written by some of the best statisticians out there, it is very robust and full of excellent procedures. But R has a rather steep learning curve. Fortunately, there are ample R resources and help files, vignettes, help blogs, and even a journal devoted to the use of R. It is like Wikipedia in that sense, because it has all the information as a regular paid-for-subscription encyclopedia, but is totally FREE. And it runs equally well on Linux, Apple Macintosh and Microsoft Windows personal computers. For these reasons, I have adopted R as the primary software for this course in Data Analysis.

Getting Started First, you need to download and install the latest R package (3.1.2) for your system (Mac, Windows PC, Linux, there are versions for all):

1. go to http://www.r-project.org/ and find a mirror site (any one is OK)

2. download R 3.1.2 and install in the default directory (R)



```
R version 3.1.2 (2014-10-31) -- "Pumpkin Helmet"
Copyright (C) 2014 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

Warning: namespace ‘ggplot2’ is not available and has been replaced
by .GlobalEnv when processing object ‘p’
[Previously saved workspace restored]

> 
```

Figure 1: The R desktop console on a Windows 64-bit system

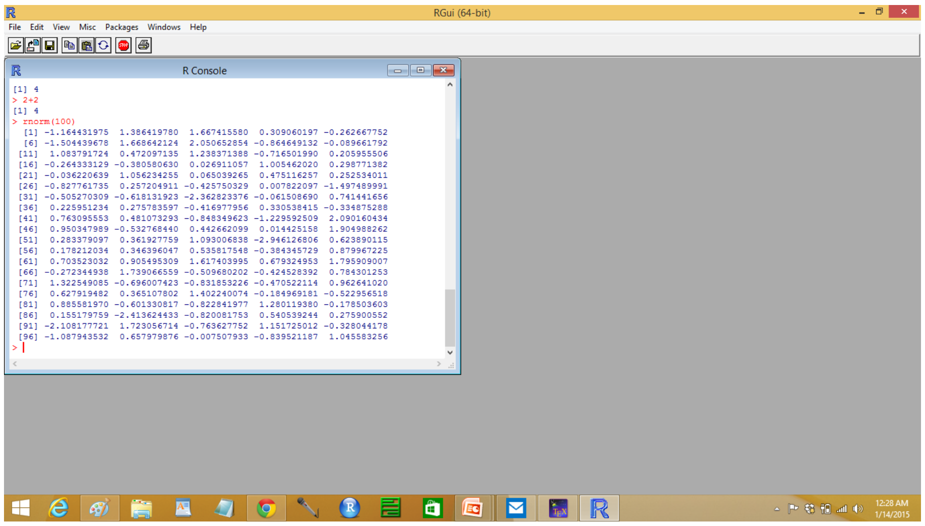
3. You can choose 32-bit or 64-bit (depends on the computer processor and operating system you have installed)

4. When finished, open R by clicking on the icon on the desktop

You should see something like this (Figure 1). The main window is called the console, and it is where you type the codes you will use. The desktop is where other windows will open, when results of the commands are presented (plots) or you want to see the command history. At the top of the R desktop, you will see a menu listing with ”File”, ”Edit”, ”View”, ”Misc.”, ”Packages”, ”Windows”, ”Help” are listed. You can click on each of these drop-down menus and see what the sub- menus are (more later). In the next row are some icons showing some common tasks - Open Script, Load Workspace, Save Workspace, Copy, Paste, Copy and Paste, and Print. In the console, at the bottom you will see a ”>”. That is the prompt. You will type commands after that.

2 Basic Calculations

R is basically a calculator, upon which all statistical calculations can be computed. There are many built in commands, which we will learn gradually. You can see that it is able to do computations with simple math, and with complex math. Try to run this simple command: type 2+2 after the command prompt, then hit enter. You should see the answer to this simple problem. [1] 4. The answer is 4, as expected, but what is the [1]? That number inside the [ ] is the row index for the answer. Some answers have many rows. Let’s try something more complicated. Type rnorm(100) after the command prompt and then hit



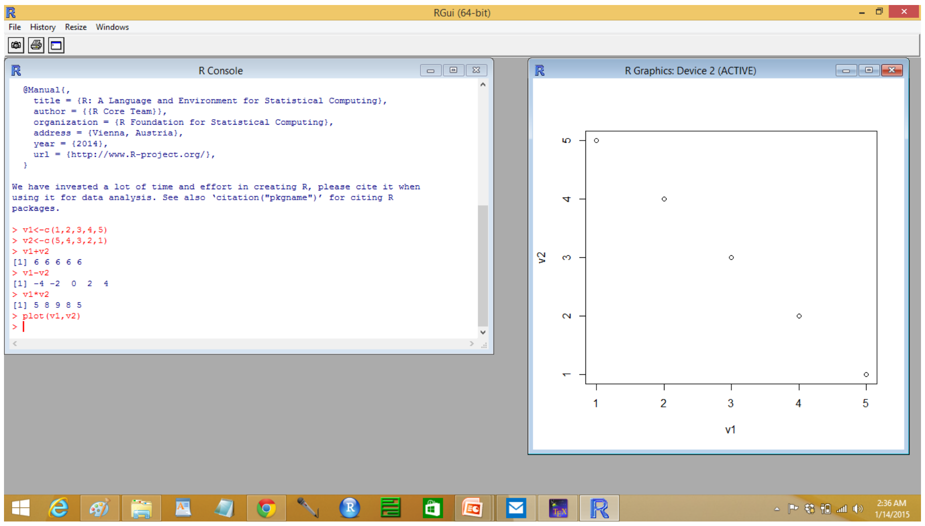
```
[1] 4
> 2+2
[1] 4
> rnorm(100)
  [1] -1.164431975  1.386419780  1.667415580  0.309060197 -0.262667752
  [6] -1.504439678  1.668642124  2.050652854 -0.864649132 -0.089661792
 [11]  1.083791724  0.472097135  1.238371388 -0.716501990  0.205955506
 [16] -0.264333129 -0.380580630  0.026911057  1.005462020  0.298771382
 [21] -0.036220639  1.056234255  0.065039265  0.475116257  0.252534011
 [26] -0.827761735  0.257204911 -0.425750329  0.007822097 -1.497489991
 [31] -0.505270309 -0.618131923 -2.362823376 -0.061508690  0.741441656
 [36]  0.225951234  0.275783597 -0.416977956  0.330538415 -0.334875288
 [41]  0.763095553  0.481073293 -0.848349623 -1.229592509  2.090160434
 [46]  0.950347989 -0.532768440  0.442662099  0.014425158  1.904988262
 [51]  0.283379097  0.361927759  1.093006838 -2.946126806  0.623890115
 [56]  0.178212034  0.346396047  0.535817548 -0.384345729  0.879967225
 [61]  0.703523032  0.905495309  1.617403995  0.679324953  1.795909007
 [66] -0.272344938  1.739066559 -0.509680202 -0.424528392  0.784301253
 [71]  1.322549085 -0.696007423 -0.831853226 -0.470522114  0.962641020
 [76]  0.627919482  0.365107802  1.402240074 -0.184969181 -0.522956518
 [81]  0.885581970 -0.601330817 -0.822841977  1.280119380 -0.178503603
 [86]  0.155179759 -2.413624433 -0.820081753  0.540539244  0.275900552
 [91] -2.108177721  1.723056714 -0.763627752  1.151725012 -0.328044178
 [96] -1.087943532  0.657979876 -0.007507933 -0.839521187  1.045583256
> 
```

Figure 2: The result of 2+2 and rnorm(100)

enter. That command will return a long list of numbers, 100 numbers to be exact (Figure 2). They are values drawn at random from a normal distribution. Try doing this yourself. Let's assign the result of our commands to a variable,which is easy to do, after you learn the R notation. the notation $< -$ means "assign" or "equal to". Type: $x < -2 + 2$ and hit enter, what do you see? Not the answer 4 as before, but nothing happened, just a new command prompt was returned. But something did happen, internally in R. The result of the command on the right of the $< -$ was stored in the new variable you created, x. Now, type x and hit enter. You see the answer is stored in x, $x = 4$, displayed just as before [1] 4. What do you think will happen if you type $y =$ rnorm(100)? Try it and see.

3 Vectors and entering data

R can store many numbers in a single variable. Such a variable is termed a vector in R, because it is a single line or row of data, even if the rows wrap around when displayed on the screen. Vectors can be added together, subtracted from one another, and multiplied or divided by a scalar, resulting in another vector. R can represent matrices as well, where there are numbers arranged in both rows and columns. For now, we will work with vectors. To enter some numbers as a vector, use the command c(x,y,z,...) to create a vector. The "c" means "concatenate". Try entering $v1 < -c(1, 2, 3, 4, 5)$ and $v2 < -c(5, 4, 3, 2, 1)$. Then add the two vectors v1+v2. What is the result? What is the result of v1-v2? v1*v2? Of 2 * v1? Try these on your R workstation. Answers are given



```
  @Manual{,
    title = {R: A Language and Environment for Statistical Computing},
    author = {{R Core Team}},
    organization = {R Foundation for Statistical Computing},
    address = {Vienna, Austria},
    year = {2014},
    url = {http://www.R-project.org/},
  }

We have invested a lot of time and effort in creating R, please cite it when
using it for data analysis. See also 'citation("pkgname")' for citing R
packages.

> v1<-c(1,2,3,4,5)
> v2<-c(5,4,3,2,1)
> v1+v2
[1] 6 6 6 6 6
> v1-v2
[1] -4 -2  0  2  4
> v1*v2
[1] 5 8 9 8 5
> plot(v1,v2)
> 
```

Figure 3: The plot of the two vectors v1 and v2

below. [1].

There are many better ways to enter data in R as vectors, matrices, or data frames, which we will see later. Data can be imported in to R from text files (as comma separated values, or csv), many stand-alone stat packages (SYSTAT and SAS among others). That is mainly how you will enter data in this course, but this simple method of creating a vector is useful sometimes.

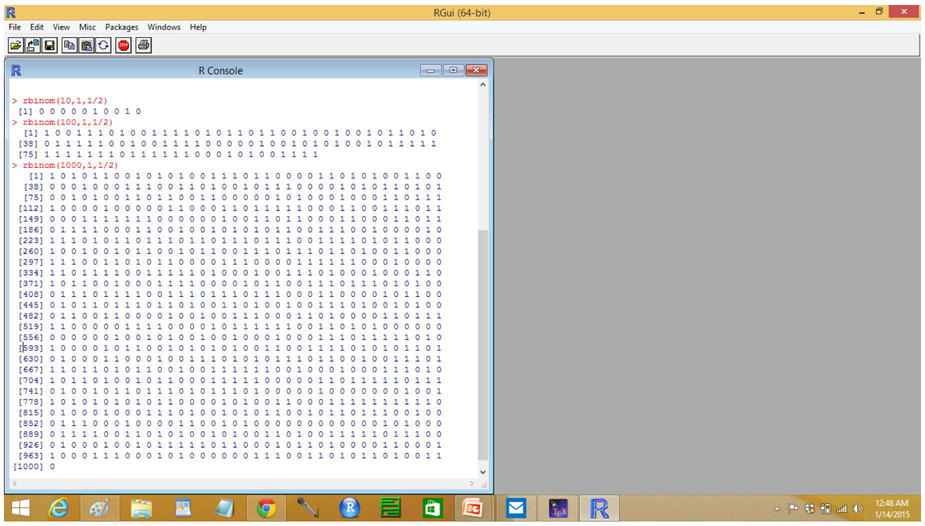
4 Plotting

It is easy to make a plot of two vectors of equal length in R. The command to create a plot is plot(x,y). Let's make a plot of the two vectors we just created (Figure 3)

5 Probability and coin flipping

Let's flip some coins in R to learn about probabilities. The definition of probability is the number of successes/number of attempts of any process. If we flip a coin 10 times, and it comes up heads 5 times, we say the probability of getting a head is 0.5 or 5 heads/10 flips. But that does not always happen. Sometimes there are 6,7,8,9 and even 10 heads in 10 flips. What is the probability in these cases? It is still p = 0.5, in reality, if the coin is not weighted so that heads will come up (a fair coin). These unusual results for number of heads are due to random chance events, and our estimate of the probability is off because 10

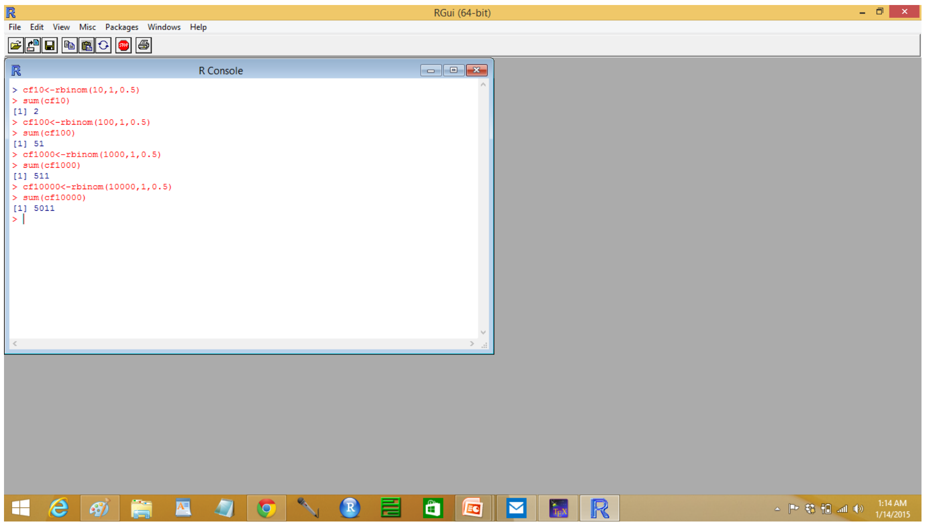
[1]a) v1+v2 = 6,6,6,6,6 v1-v2=-4,-2,0,2,4 v1*v2=5,8,9,8,5 2*v1=2,6,8,10



```
> rbinom(10,1,1/2)
 [1] 0 0 0 0 0 1 0 0 1 0
> rbinom(100,1,1/2)
  [1] 1 0 0 1 1 1 0 1 0 0 1 1 1 1 0 1 0 1 1 0 1 1 0 0 1 0 0 1 0 0 1 0 1 1 0 1 0
 [38] 0 1 1 1 1 1 0 0 1 0 0 1 1 1 1 0 0 0 0 0 1 0 0 1 0 1 0 1 0 0 1 0 1 1 1 1 1
 [75] 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 0 0 1 0 1 0 0 1 1 1 1
> rbinom(1000,1,1/2)
   [1] 1 0 1 0 1 1 0 0 1 0 1 0 1 0 0 1 1 1 0 1 1 0 0 0 0 1 1 0 1 0 1 0 0 1 1 0 0
  [38] 0 0 0 1 0 0 0 1 1 1 0 0 1 1 0 1 0 0 1 0 1 1 1 0 0 0 0 1 0 1 0 1 1 0 1 0 1
  [75] 0 0 1 0 1 0 0 1 1 0 1 1 0 0 1 1 0 0 0 0 0 1 0 1 0 0 0 1 0 0 0 1 1 0 1 1 1
 [112] 1 0 0 0 0 1 0 0 0 0 0 1 1 0 0 0 1 1 0 1 1 1 1 1 0 0 0 1 1 0 0 1 1 1 0 1 1
 [149] 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 1 0 0 1 1 0 1 1 0 0 0 1 1 0 0 0 1 1 0 1 1
 [186] 0 1 1 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 0 1 0 1 1 0 0 1 1 1 0 0 1 0 0 0 0 1 0
 [223] 1 1 1 0 1 0 1 1 0 1 1 1 0 1 1 0 1 1 1 0 1 1 1 0 0 1 1 1 1 0 1 0 1 1 0 0 0
 [260] 1 0 0 1 0 0 1 0 1 1 0 0 1 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 0 1 0 0 1 1 0 0 0
 [297] 1 1 1 0 0 1 1 0 1 0 1 1 0 0 0 0 1 1 1 0 0 0 0 1 1 1 1 1 1 0 0 0 1 0 0 0 0
 [334] 1 1 0 1 1 1 1 0 0 1 1 1 1 1 0 1 0 0 0 1 0 0 1 1 1 0 1 0 0 0 1 0 0 0 1 1 0
 [371] 1 0 1 1 0 0 1 0 0 0 1 1 1 1 0 0 0 0 1 0 1 1 0 0 1 1 1 0 1 1 1 0 1 0 1 0 0
 [408] 0 1 1 1 0 1 1 1 1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 0 0 1 1 0 0 0 0 1 0 1 1 0 0
 [445] 0 1 0 1 1 0 1 1 1 0 1 1 0 1 0 0 1 1 0 1 0 0 1 0 0 1 1 1 0 1 0 0 1 0 1 0 0
 [482] 0 1 1 0 0 1 1 0 0 0 0 1 0 0 1 0 0 1 1 1 0 0 0 1 1 0 1 0 0 0 0 1 1 0 1 1 1
 [519] 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 1 0 1 1 1 1 1 1 0 0 1 1 0 1 0 1 0 0 0 0 0 0
 [556] 0 0 0 0 0 0 1 0 0 1 0 1 0 0 1 0 0 1 0 0 0 1 0 0 0 1 1 1 0 1 1 1 1 1 0 1 0
 [593] 1 0 0 0 0 1 0 1 1 0 0 1 0 1 0 1 0 1 0 0 1 1 0 0 1 1 1 1 0 1 0 1 0 1 1 0 1
 [630] 0 1 0 0 0 1 1 0 0 0 1 0 0 1 1 1 0 1 0 1 0 1 1 1 0 1 1 0 0 1 0 0 1 1 1 0 1
 [667] 1 1 0 1 1 0 1 0 1 1 0 0 1 0 0 1 1 1 1 1 1 0 0 1 0 0 0 1 0 0 0 1 1 1 0 1 0
 [704] 1 0 1 1 0 1 0 0 1 0 1 1 0 0 0 1 1 1 1 1 0 0 0 0 0 1 1 0 1 1 1 1 1 0 1 1 1
 [741] 0 1 0 0 1 0 1 1 0 1 1 1 0 1 0 1 1 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 1
 [778] 1 0 1 0 1 0 1 0 1 0 1 1 0 0 0 0 1 0 1 0 0 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 0
 [815] 0 1 0 0 0 1 0 0 0 1 1 1 0 1 0 0 1 0 1 0 1 1 0 0 1 0 1 1 0 1 1 1 0 0 1 0 0
 [852] 0 1 1 1 0 0 0 1 0 0 0 0 1 1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0
 [889] 0 1 1 1 1 0 0 1 1 0 1 0 1 0 0 1 0 1 0 0 1 1 0 1 0 0 1 1 1 1 1 0 1 1 1 0 0
 [926] 0 1 0 0 0 1 0 0 1 0 1 1 1 1 1 0 1 1 0 0 0 1 0 1 1 0 1 0 0 0 0 1 1 0 0 0 1
 [963] 1 0 0 0 1 1 1 0 0 0 1 0 1 0 0 0 0 0 1 1 1 0 0 1 1 0 1 0 1 1 0 1 0 0 1 1
[1000] 0
```

Figure 4: Coin flips of 10, 100, and 1000 replicates

flips is too few to see the true (long-term) probability of the flipping process. We know, even with 10 heads in a row, that intuitively the $p \neq 1.0$. We could flip the coin 100, 1000, 10,000 times or more, count the heads, and get a better idea of the true probability. That would be time consuming in practice, but we can simulate a fair coin in R and see what the true probability is. The commands to do 10, 100, and 1000 coin flips are shown in Figure 4. The command is rbinom(10,1,0.5). This tells R to flip a coin (rbinom is a random binomial function in R) ten times using the parameter or argument: (10,1,0.5) 10 for ten replicates, 1 for the specific binomial distribution called Bernouli, and for the coins bias, in this case 0.5. Now you could count the heads (1's in tis case) by hand, but why not let R do it? Here's how: Assign the result in each case to a variable, and use the built-in command sum(variable) to compute the number of heads in each variable. Let's do that next (Figure 5). Notice that I called the variable a unique name to keep the results for the coin flips separate, so 10 flips was assigned to variable cf10 (coin flip 10), 100 flips assigned to cf100 and 1000 coin flips to cf1000. I did 10,000 flips here as well, calling the result cf10000. Variables can be named any way you please, as long as they don't start with a period (.) or a number. They should also not be the same as an existing R built-in commands (avoid "mean" and "SD", among others). To get the number of heads, we simply use the built-in command sum( ) to add up the heads. There are 2/10, 51/100, 511/1000, and 5011/10000 heads in this example. You will get different numbers each time you run the command, that is because it is random selection from a binomial distribution. What values do you get? What happened as the number of coin flips increased? Did the estimate of the probability of getting a head get closer to the true probability? Make a plot of the probability of getting a head (p on y-axis) in 10, 100, 1000, and 10,000 coin flips



```
> cf10<-rbinom(10,1,0.5)
> sum(cf10)
[1] 2
> cf100<-rbinom(100,1,0.5)
> sum(cf100)
[1] 51
> cf1000<-rbinom(1000,1,0.5)
> sum(cf1000)
[1] 511
> cf10000<-rbinom(10000,1,0.5)
> sum(cf10000)
[1] 5011
>
```

Figure 5: The number of heads resulting from 10, 100, 1000, 10000 coin flips

(on x-axis). Where does the p converge on the ”true” probability?

6 Homework 1

For Homework 1, turn in the following as a pdf document. 1) Plot the data from 10,000 points drawn from a normal distribution. Typically, the normal distribution will have an average (mean) of 0.0 with equal numbers of positive and negative points, varying between -4 and +4. Plot the frequency (probability) of getting each point of this distribution along the y-axis, and the points ranging from -4.0 through +4.0 on the x axis. Hint: You may wish to try this with fewer random points (try 50) to figure out the probabilities, and group the points into frequency bins, like -1.5 to -1.0, -1.0 to -0.5, -0.5 to - 0.0, 0.0 to 0.5, 0.5 to 1.0, 1.0 to 1.5 etc., computing the ”successes” for teh number of random points falling within each frequency bin in 50 trials. Hint 2: There is a very easy built-in way to do this in R with a single command. 2) Create an x,y plot for the coin flip experiment (10 flips through 10,000 flips) as just described above. 3) Then devise a procedure for testing whether or not a new coin is ”fair” with the data given below: 10 flips has 8 heads 100 flips has 76 heads 1000 flips has 795 heads 10000 flips has 8035 heads If it is a ”fair” coin what would the expected probability be? Is the asymptotic probability for this trail with a new unknown coin the same as that of a ”fair” coin? How can we test this difference statistically? Hint, try using the binomial distribution with the long-term probability suspected for this coin, then look at any statistical test that you are familiar with to assess the significance of the difference.

References

[1] R Core Team , R: A Language and Environment for Statistical Computing , R Foundation for Statistical Computing , Vienna, Austria, 2014, http://www.R-project.org/