DSCI/MATH530 RLab 1

Drew Remmenga

Overview

This first lab in R is an introduction to using R for computing and looking at data. It is also a different perspective on Module 1. Here we will be computing probabilities and simulating random processes rather working out analytic formulas.

Submission

- Complete this assignment using R Markdown and convert to pdf.
- Submit through gradescope.

You can also refer to the .Rmd file for this assignment to have an example of this markdown document. Note however, that this version is setup to also use Latex for some math and conversion directly to pdf.

Background on R

This assignment will give some specific hints about using the R language. However, you should look at the R tutorial included in the course pages. (See **»Start Here** in the top menu and then look for Getting Ready – the world of R.) for much more details.

Be careful to use `` ```{r} `` to begin a code block that will be run. Problem 5 has some formatted code blocks but are not run by omitting the `{r}`

Note: In R anything in the workspace is usually referred to as an *object*. Sometmes these are data sets read in or other results from doing some computations. They might also have specific types, such as vectors, matrices or data frames, but we would call them all objects.

E.g. `obj<- 1:10` and we would call `obj` an *object* in the workspace and more specifically in this case it is a vector.

1 Counting the easy way.

The following R code generates all possible combinations of rolling 3 dice.

```r
sampleSpace<- expand.grid( die1= 1:6, die2= 1:6, die3 = 1:6)
```

a) We know that `sampleSpace` should have 6x6x6 = 216 elements. How would you find this out using functions in R and this created object?

```r
lengths(sampleSpace)
```

```
## die1 die2 die3
##  216  216  216
```

b) Is `sampleSpace` a matrix OR a table OR a list OR a dataframe? Dataframe c) Assign `ind<-sampleSpace$die1 == 4`. Is `ind` numeric, logical, character? What is the result of `sampleSpace[ind, ]`? How many cases are in this subset?

```r
ind <- sampleSpace$die1 == 4
length(ind)
```

```
## [1] 216
```

```r
sampleSpace[ind,]
```

```
##     die1 die2 die3
## 4      4    1    1
## 10     4    2    1
## 16     4    3    1
## 22     4    4    1
## 28     4    5    1
## 34     4    6    1
## 40     4    1    2
## 46     4    2    2
## 52     4    3    2
## 58     4    4    2
## 64     4    5    2
## 70     4    6    2
## 76     4    1    3
## 82     4    2    3
## 88     4    3    3
## 94     4    4    3
## 100    4    5    3
## 106    4    6    3
## 112    4    1    4
## 118    4    2    4
## 124    4    3    4
## 130    4    4    4
## 136    4    5    4
## 142    4    6    4
## 148    4    1    5
## 154    4    2    5
## 160    4    3    5
## 166    4    4    5
## 172    4    5    5
## 178    4    6    5
## 184    4    1    6
## 190    4    2    6
## 196    4    3    6
## 202    4    4    6
## 208    4    5    6
## 214    4    6    6
```

```r
lengths(sampleSpace[ind,])
```

```
## die1 die2 die3
##   36   36   36
```

This is the outcome where we have four on the first die. Logical ## 2 Counting a nasty event. Based on the setup from problem 1 describe in words the compound event.

```r
ind2<- abs(sampleSpace$die1 -sampleSpace$die2) <=2 &
        abs(sampleSpace$die2 -sampleSpace$die3) <=2
```

True or false the absolute value of the difference between die1 and die2 is greater than two and the absolute value of the difference between die 2 and die 3 is greater than two. Call this event A. By counting elements in A using `sampleSpace` and `ind2` what is $P(A)$?

```r
lengths(sampleSpace[ind2,])
```

```
## die1 die2 die3
##  100  100  100
```

```r
A<- 100/216
A
```

```
## [1] 0.462963
```

3 Computing probabilities using built in functions.

The function `phyper` will compute probabilities based on the hypergeometric distribution (see `help(phyper)` for more details), the function `pbinom` will work for the binomial, and `ppois` for the poisson. Note that for each of these R may choose some strange variables names for the distribution parameters. E.g. `size` in the binomial is our n from the text.

a) Given these functions redo problem 2.62 parts b) and c).

```r
pbinom(0,3,2 /50)
```

```
## [1] 0.884736
```

b) Redo a) and b) from problem 2.64.

```r
1-pbinom(4, 200,.05, lower.tail = TRUE)
```

```
## [1] 0.9735532
```

```r
1-ppois(4,10)
```

```
## [1] 0.9707473
```

4 The normal distribution.

For the normal distribution in R there are a group of functions: `pnorm` the CDF, `dnorm` the density function, `rnorm` generates a random sample and `qnorm` finds the quantile (inverse CDF). See `help(Normal )` for all the details. Note that in R the argument *mean* is μ and *sd* is σ from the textbook.

a) Explain why we get the following results:

```r
c(
  pnorm(0),
  qnorm(0.5),
  pnorm(100, mean =100, sd=10),
  pnorm(100, mean =100, sd=5)
)
```

```
## [1] 0.5 0.0 0.5 0.5
```

pnorm dives the cdf of a standard normal distribution centered aroun dzero giving a probability of .5. Qnorm is the inverse of this. Pnorm asks what the probability of getting the mean is. Same with pnorm with an sd of 5.

b) Explain why the following samples are not the same,

```r
rnorm(3)
```

```
## [1]  1.3475867  0.1611734 -1.2320601
```

```r
rnorm(3)
```

```
## [1] 0.4616570 0.8534462 0.2152273
```

Its random. If we set the seed we would get the same result.

but the following results are the same.

```r
set.seed(530)
rnorm(3)
```

```
## [1] 1.034420 1.174682 1.027885
```

```r
set.seed(530)
rnorm(3)
```

```
## [1] 1.034420 1.174682 1.027885
```

Here we set the seed so our random numbers are the same.

c) Solve problem 2.80 b) by Monte Carlo: Generate a large sample of normal random variables (using `rnorm`) and determine the fraction in $[16.0, 16.5]$.

A common shortcut in R is if you have a vector that is a logical (TRUE or FALSE). Then summing up the vector (use the `sum` function) is the same as counting the number of TRUE values.

For the CS folks: In applying a numeric function to logicals TRUE is coerced to 1 and FALSE to 0.

d) Solve problem 2.80 c) by Monte Carlo: Simulate a large sample of normal random variables, sort them, then find the lower 10th percentile (or use the `quantile` function applied to your random sample). Assign this sample to an R object (e.g. `simNorm`) so that you can use it in problem 5.

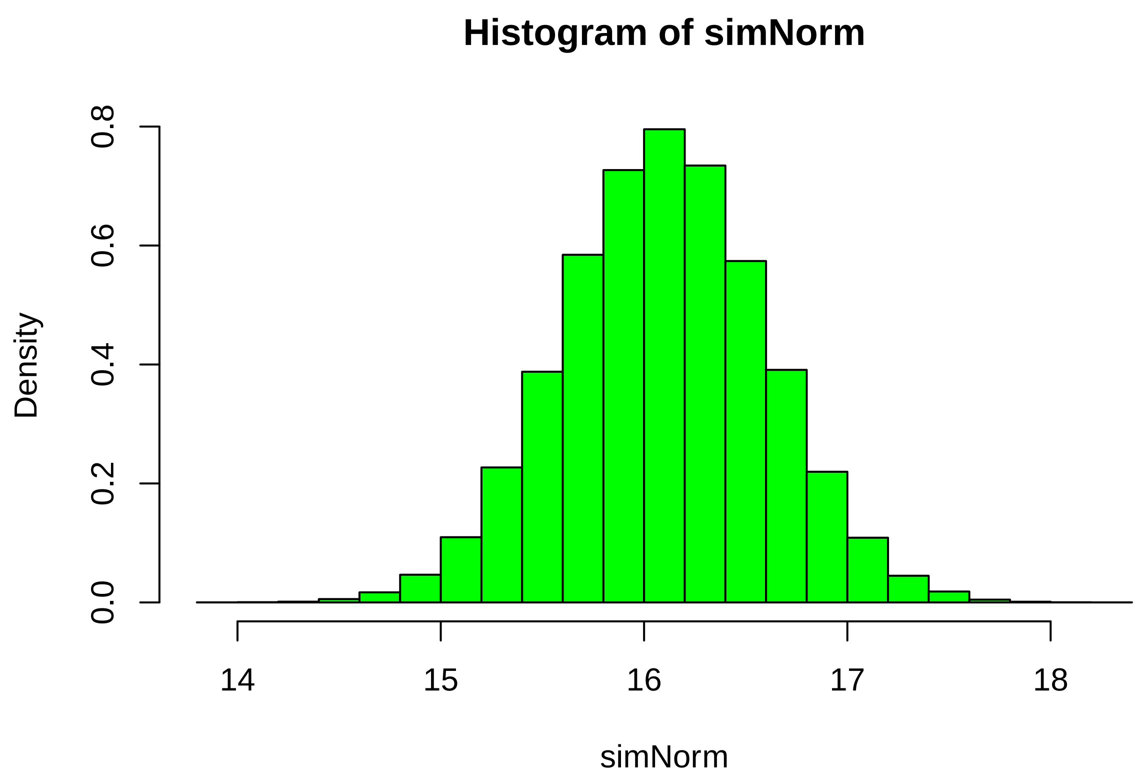
```
simNorm<-rnorm(100000,mean=16.1, sd=.5)
quantile(simNorm,probs=c(.10,.90))
```

```
##      10%      90%
## 15.45665 16.73949
```

5 Matching the sample and theoretical densities.

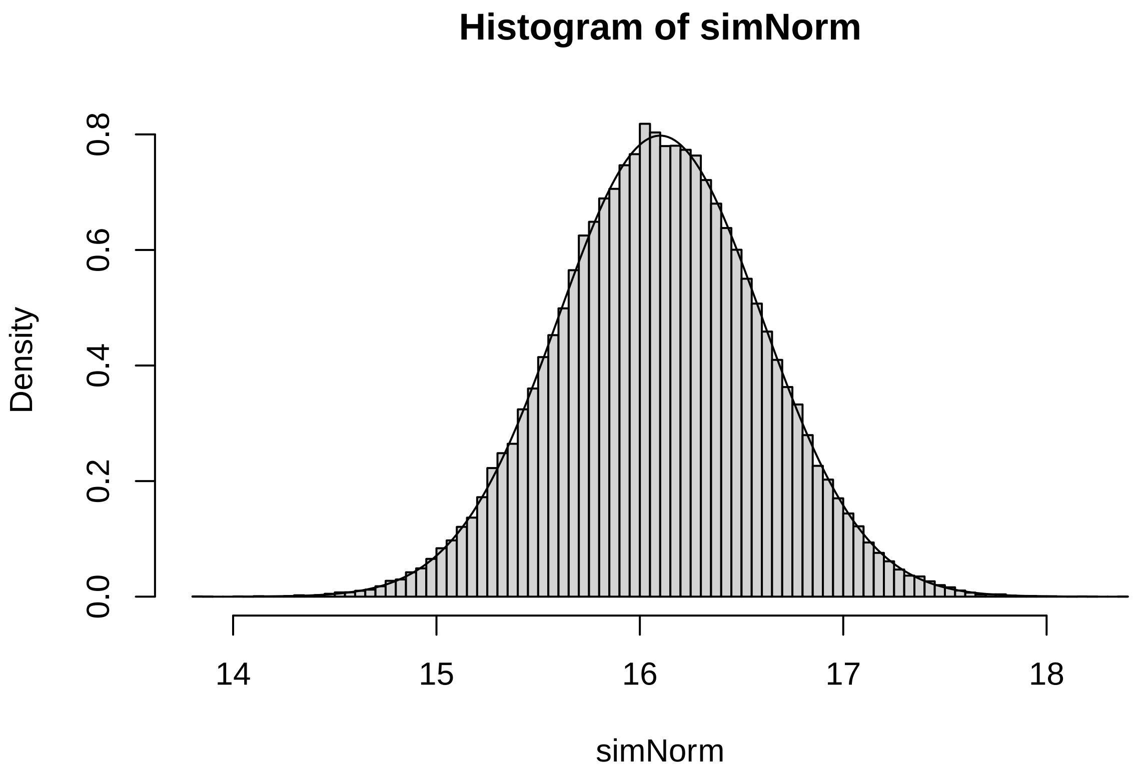
Finally let's see how well your random sample matches the theoretical density. A histogram of your sample (with green bars and density scaling)

```
hist( simNorm, probability=TRUE, col="green")
```



Now add the normal pdf on top. Range is plus/minus 4 standard deviations around mean.

```r
xValues<- seq( 16.1 - 4*.5, 16.1 + 4*.5, length.out=250 )
pdf<- dnorm( xValues, mean=16.1, sd=.5)
hist(simNorm,breaks=150,freq=FALSE)
lines(xValues,pdf)
```



Comment on how well your sample histogram tracks the density curve. Where does it do well? What does it miss?

The normal curve is fit very well by the distribution.