MIE237 Week 2 Lab

Neil Montgomery

Preliminaries

You should have installed R and RStudio. Latex is optional but nice. You should seek out some online intro to R to work through just to get started. This one seems fine (and is one I indirectly recommended last week):

http://tryr.codeschool.com/

You don't need to become a big expert (unless you want to).

Git

Something I didn't mention but should have... install Git! This lets you "clone" all course repositories as a new RStudio project and is probably the easiest way to get course files. Note that Git is a program, while github.com is a website and a place to store code repositories. If you want to get repositories from github.com you need to install Git on your system. See:

https://support.rstudio.com/hc/en-us/articles/200532077-Version-Control-with-Git-and-SVN

for details.

RStudio best practices

Use RStudio projects. Each project in its own fresh clean directory. Keep the data files in that directory.

This lab

This lab assumes you have already learned a little about R basics on your own. You should have this very file opened up in RStudio. We'll import some data, compute a numerical summary, make a plot, calculate a confidence interval.

Installing packages.

R is pretty powerful on its own but there is a also vast quantity of packages available for doing just about anything in statistics. You'll need to be able to install packages when required. Let's try one now. You'll need to be connected to the internet.

Three packages we'll use in this week's lab are called: `dplyr`, `rio`, and `ggplot2`. To install them, select the "Packages" tab in the bottom-right pane of RStudio and click "Install". Just type dplyr, rio, ggplot2 and hit enter and they (along with other packages they themselves need) will be installed. If something goes wrong, well, look at the error message, try Google, or as a last resort ask Jenna or me for help. (We will in turn ask Google for help.)

To use packages you have to load them using the library function as follows. Some messages might come up. They usually warn you that the package has renamed something - usually this doesn't matter and you can ignore the messages. This week we'll only be using `rio`.

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
##
## The following objects are masked from 'package:stats':
##
##     filter, lag
##
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(rio)
library(ggplot2)
```

Import some basic data and do a few things

A dataset comes with this lab called `Ex09.27.txt` and is the data from question 9.27 from the book and is a weeny little text file. Go read the question to get the story behind it.

For importing data I like to use the `rio` package that I loaded. To import the textbook data use `rio`'s `import` function, that does a good job of figuring out how to interpret the file without the user having to supply too many details.

```r
ex9.27 <- import("Ex09.27.txt")
```

(RStudio has nice typing hints accessing external files but you need to use the Tab key to get these hints. Try typing everything up to the `Ex` and hitting the Tab key to see how this works.)

This object appears in the Environment tab in the top right pane of RStudio. It is categorized as "Data". The object is what R calls a *data frame*. All datasets we use will be data frames. You can think of a data frame as a cross between a matrix and a spreadsheet.

You can look at the data by clicking on its name. You can look at the "structure" of the object by clicking on the blue dot beside the name (and click the dot again to collapse this information.)

Let's compute some numerical summaries. You can cut and paste, but try typing at least one of these out so you can see how the RStudio typing hints work.

```r
nrow(ex9.27) # returns the sample size
```

```
## [1] 16
```

```r
mean(ex9.27$diameter)
```

```
## [1] 1.0025
```

```r
sd(ex9.27$diameter)
```

```
## [1] 0.02016598
```

```
summary(ex9.27)
```

```
##     diameter    
##  Min.   :0.970  
##  1st Qu.:0.990  
##  Median :1.000  
##  Mean   :1.002  
##  3rd Qu.:1.012  
##  Max.   :1.040  
```

Let's get the 95% confidence interval. A function we'll use quite a bit is called `t.test` and implements a number of types of statistical inferences based on t distributions.

```
t.test(ex9.27$diameter)
```

```
## 
##  One Sample t-test
## 
## data:  ex9.27$diameter
## t = 198.85, df = 15, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  0.9917543 1.0132457
## sample estimates:
## mean of x 
##    1.0025 
```

This gives more output than we need. It gives us the 95% interval, as well as giving us the results of a hypothesis test of $H_0 : \mu = 0$ versus $H_a : \mu \neq 0$ even though we didn't ask it to.

Also, when you execute the `t.test` command the results just fly past and if we need any of its parts (like for a report or some other calculation) it's not clear how we would do that. It is very common in R to assign a results to its own named object, like this.

```
ex9.27_t.test <- t.test(ex9.27)
```

This appears under "Values" in the Environment tab. Click on the blue circle. This object is a "List of 9" elements called `statistic`, `parameter`, and so-on. These elements can be accessed directly, for example:

```
ex9.27_t.test$statistic
```

```
##        t 
## 198.8498 
```

```
ex9.27_t.test$p.value
```

```
## [1] 4.449457e-27
```

```r
ex9.27_t.test$conf.int
```

```
## [1] 0.9917543 1.0132457
## attr(,"conf.level")
## [1] 0.95
```

```r
ex9.27_t.test$conf.int[1] # First element of the C.I.
```

```
## [1] 0.9917543
```

The `t.test` function has options that we might wish to change. If we wanted to get the 99% C.I. we can do this (type this out manually to see how RStudio helps with suggestions):

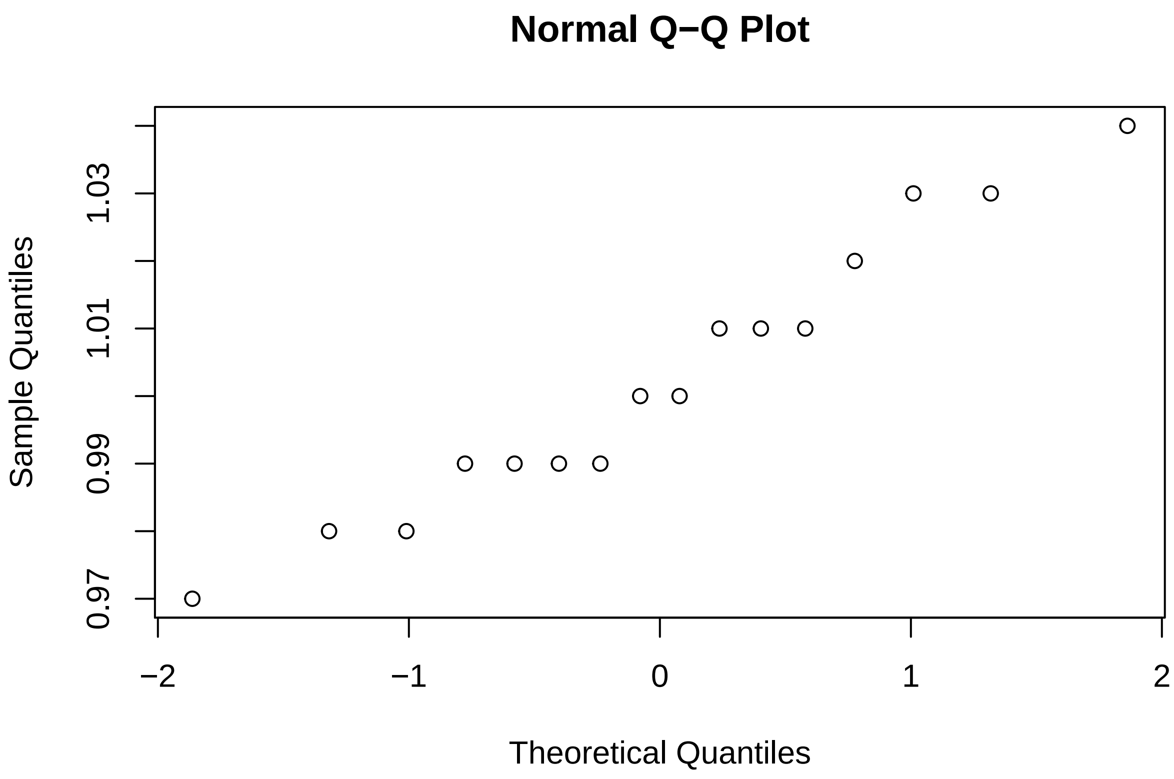
```r
t.test(ex9.27, conf.level = 0.99)
```

```
##
##  One Sample t-test
##
## data:  ex9.27
## t = 198.85, df = 15, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 99 percent confidence interval:
##  0.9876442 1.0173558
## sample estimates:
## mean of x
##    1.0025
```

To find out more about `t.test` (or any other function) just type `?t.test` at the console and the documentation appears in the bottom right pane.

The only useful plot I can think of would be a normal quantile plot (to see if the output of `t.test` is valid) and one way of doing this is:

```r
qqnorm(ex9.27$diameter)
```



but this is really ugly. I'll show you another way later.