

Q&A Material

Ari Anisfeld & Terence Chau

9/28/2020

What's the Deal with Fall Coding Lab?

Two tracks:

Accelerated: 2 lessons

- ▶ 2 lessons covering loops and functions.
- ▶ No final project (you already did it).

Not accelerated: 5 lessons.

- ▶ 3 lessons review summer camp material.
- ▶ 2 lessons covering loops and functions.
- ▶ Final project:
 - ▶ Find a data set that speaks to you.
 - ▶ Try to uncover something interesting. Graph it and tab it.
 - ▶ We'll give you feedback.

Logistics for Both Tracks

- ▶ Instructors Ari and Terence + wonderful TAs.
- ▶ 80 minutes per week: brief review and Q&A, then work in groups.
- ▶ Not graded.
- ▶ Access to TAs for coding specific problems throughout the quarter.
 - ▶ TA office hours 30 minutes before and after lecture time.
- ▶ Github website with all material.
 - ▶ We'll post solutions (eventually).
- ▶ Use Piazza for questions.
 - ▶ Rules of engagement: coding questions only, no Stats homework!
 - ▶ How to ask a good question?

Poll: How much coding experience do you have?

- ▶ First timer.
- ▶ Beginner.
- ▶ Intermediate.
- ▶ Proficient.

Please include your email.

Class 1: Why R? & Vectors

Key Points: R Basics

- ▶ Rstudio has a console to access R and a text editor to write code for reproducible projects.
 - ▶ Analogy: R is to RStudio as Tony Stark is to Iron Man's suit.
- ▶ R extensible through packages.
 - ▶ use `install.packages("")` once and then `library()` each session.
- ▶ Use `<-` to assign *any* object to a name.
- ▶ Functions take inputs and return outputs.
 - ▶ Input “understood” based on position or name.
 - ▶ Find out more about functions with `?` (e.g. `?filter`).

Questions

- ▶ Any questions on this? Feel free to ask on chat.

Key points: Vectors

- ▶ Vectors are the fundamental way to store data in R.
- ▶ We can operate on vectors element-by-element without loops.
 - ▶ `dplyr` verbs rely on this!
- ▶ We introduced built-in functions to build vectors and do operations on vectors.
- ▶ NAs are sticky!

Key points: Data Types and Coercion

- ▶ (Atomic) Vectors have a single data type.
 - ▶ Most often: `logical`, `integer`, `double`, or `character`.
- ▶ Certain operations expect a certain data type and R will try to coerce the data if it can.
- ▶ Usually, simpler types can be coerced to more complex types.
 - ▶ `logical < integer < double < character`.
 - ▶ Example on slides: `paste0(1L, "ing")`.
- ▶ Caution! Coercion can lead to unexpected behavior such as making NAs.

One More Thing

Logicals are coercible to numeric or character. This is very useful!

Determine the rule for how R treats TRUE and FALSE in math.

```
TRUE + 4
```

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

```
FALSE + 4
```

```
## [1] 4
```

Questions?

- ▶ Any questions on this? Feel free to ask on chat.

Warm up & Lab 1

Warm up

- ▶ Solve the questions at the beginning of the lab in small (random) groups.

Lab 1

- ▶ Two “types” of breakout room:
 - ▶ Work along: larger group with more guidance from a TA.
 - ▶ Small groups: 4 people, TAs will come in and out to answer questions. Use the help button!
- ▶ Add “(work along)” or “(small group)” to your Zoom name so we can sort you.
- ▶ Get as far as you can, then finish it up after class.
- ▶ Before you leave, fill out the exit poll.

Lab 1: Exit poll

- ▶ What does `hist()` return?
 - ▶ A histogram plot of the data you give it.
 - ▶ A history of the commands you've run.

Class 2: Reading files and 'dplyr'

Course logistics:

- ▶ When should we start working on the final project?
 - ▶ Start looking for a dataset now.
 - ▶ Write code to read it into R and start investigating with `dplyr` verbs.
 - ▶ Ask simple questions that can be addressed with your current tools.

Key points: Reading files

- ▶ Tabular data is stored in a lot of different formats.
 - ▶ e.g. `.csv`, `.xlsx`, `.dta`
- ▶ Read tabular data of a given type with the proper function.
 - ▶ e.g. for csvs we have `read_csv()`
 - ▶ If you get a new type, Google “How to read xxx files into R tidyverse”.
- ▶ We need to be aware of the file path and can `setwd()`.
- ▶ We know there are useful tools built into the `read_xxx()` functions.
 - ▶ Though we just scratched the surface.

Key points: Manipulating data with `dplyr()`

- ▶ Choose columns with `select()`.
- ▶ Choose rows based on a match criteria with `filter()`.
 - ▶ We were introduced to comparison operators like `==` and `%in%`.
- ▶ Make new columns with `mutate()`.
- ▶ Sort data with `arrange()` and `arrange(desc())` or `arrange(-x)`.
- ▶ Create summary statistics with `summarize()`.

Key points: Grouped analysis with `group_by()`

- ▶ *Groups* are a set of rows that belong together.
 - ▶ `group_by()` adds information about groups without changing the “data”.
- ▶ Use `group_by()` with `summarize()` to create summary data at group-level.
 - ▶ Use with functions that *reduce* data from a vector to a single value per group.
 - ▶ Expected output: a table with one row per group and one column per summary statistic and one column per grouping column.
- ▶ We can also use `group_by` to do grouped analysis with:
 - ▶ `mutate` with window functions or to add a summary stat as column for further analysis.
 - ▶ It also can impact `arrange` and `filter`.

Warm up & Lab 2

Warm up & Lab 2

- ▶ Solve the questions at the beginning of the lab in small (random) groups.
- ▶ After: add “(small group)” in front of your name if you want to work in one, if you want to work along just stay in the main room.
- ▶ Exit poll:
 1. What `dplyr` command allows you to create or modify variables?
 2. What `dplyr` command allows you to sort your data?

Class 3: Control Flow

Key points: control flow with `if ()` and `ifelse ()`

- ▶ Review how logical operators (i.e. `!`, `|`, `&`) and comparison operators work.
 - ▶ If this all still seems confusing a refresher in propositional logic will seriously pay off!
- ▶ Use `if ()`, `else`, and `else if ()` to control when and how an action is completed.
 - ▶ Can use shorthand for conditions: `if (thing_thats_true)` is equivalent to `if (thing_thats_true == TRUE)`.
- ▶ `ifelse()` as a vectorized, NA friendly alternative.
- ▶ Use `ifelse()` with `mutate()` to create new columns contingently.

Two more things (1/2): case_when()

- ▶ case_when() allows you to chain many conditions and outputs:

```
case_when(condition_1 ~ value_1,  
          condition_2 ~ value_2,  
          condition_3 ~ value_3,  
          TRUE ~ value_4)
```

Somewhat equivalent to:

```
ifelse(condition_1, value_1,  
       ifelse(condition_2, value_2,  
              ifelse(condition_3, value_3, value_4)))
```

Two more things (2/2): NA

- ▶ How to think about logicals and NA? If the answer is ambiguous \rightarrow NA. If the answer isn't \rightarrow not NA.

```
TRUE | NA
```

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

```
FALSE & NA
```

```
## [1] FALSE
```

Warm up & Lab 3

Warm up & Lab 3

- ▶ This warm up is a little longer. Aim to get up to the “Vectorized booleans” part!
- ▶ After: add “(small group)” in front of your name if you want to work in one.
- ▶ Exit poll:
 1. What is FALSE | NA?
 2. What is TRUE & NA?

Acc Class 1 / Class 4: Functions

What's the Deal with Fall Coding Lab?

Two tracks:

Accelerated: 2 lessons

- ▶ 2 lessons covering loops and functions.
- ▶ No final project (you already did it).
- ▶ Occurs every other week.
 - ▶ Ideally the sections are balanced.

Not accelerated: 5 lessons.

- ▶ 3 lessons review summer camp material.
- ▶ 2 lessons covering loops and functions.
- ▶ Final project:
 - ▶ Find a data set that speaks to you.
 - ▶ Try to uncover something interesting. Graph it and tab it.
 - ▶ We'll give you feedback.

Logistics for Both Tracks

- ▶ Instructors Ari and Terence + wonderful TAs.
- ▶ 80 minutes per week: brief review and Q&A, then work in groups.
- ▶ Not graded.
- ▶ Access to TAs for coding specific problems throughout the quarter.
 - ▶ TA office hours 30 minutes before and after lecture time.
- ▶ Github website with all material.
 - ▶ We'll post solutions (eventually).
- ▶ Use Piazza for questions.
 - ▶ Rules of engagement: coding questions only, no Stats homework!
 - ▶ How to ask a good question?

Today

- ▶ Review what functions are
- ▶ Do a warm-up
- ▶ Provide background for the lab
- ▶ Do the lab

Functions

What are some of the key take aways you learned about functions from the video?

```
do_monte_carlo <- function(N, true_mean, B, alpha){  
  sample_statistics <- monte_carlo_samples(N, true_mean, B)  
  z_scores <- get_zscores(sample_statistics$mean,  
                           true_mean,  
                           sample_statistics$sd,  
                           N)  
  test_significance(z_scores, alpha) %>% mean()  
}
```

Key points: Functions

- ▶ Write functions when you want to use a set of operations repeatedly
 - ▶ Don't Repeat Yourself (DRY)
- ▶ Functions consist of arguments and a body and are usually assigned to a name.
 - ▶ The arguments are things that you want to change in the code
 - ▶ The body is what you would be repeating if you wrote sloppy code
- ▶ Functions are for humans
 - ▶ pick names for the function and arguments that are clear and consistent
- ▶ Debug your code as much as you can as you write it.
 - ▶ if you want to use your code with `mutate()` test the code with vectors

Warm up

- ▶ Solve the questions at the beginning of the lab in small (random) groups.

Background for the lab

Concepts: r/q/p/d functions

R has built-in functions for working with distributions.

| | example | what it does? |
|---|-----------------------|--|
| r | <code>rnorm(n)</code> | generates a random sample of size n |
| p | <code>pnorm(q)</code> | returns CDF value at q |
| q | <code>qnorm(p)</code> | returns inverse CDF (the quantile) for a given probability |
| d | <code>dnorm(x)</code> | returns pdf value at x |

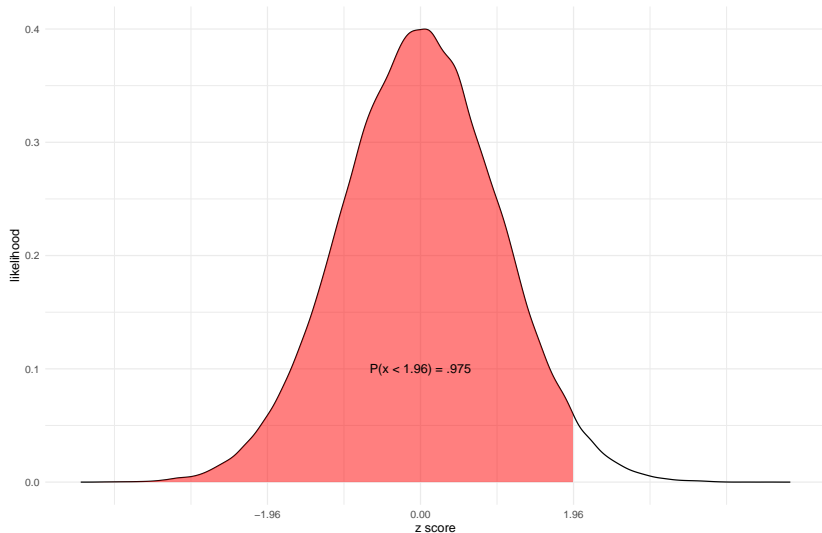
We should already be familiar with `r` functions like `rnorm()` and `runif()`.

- ▶ These concepts will be taught in stats. We don't expect you to learn them here. We can help you reason through the material enough to do the coding.

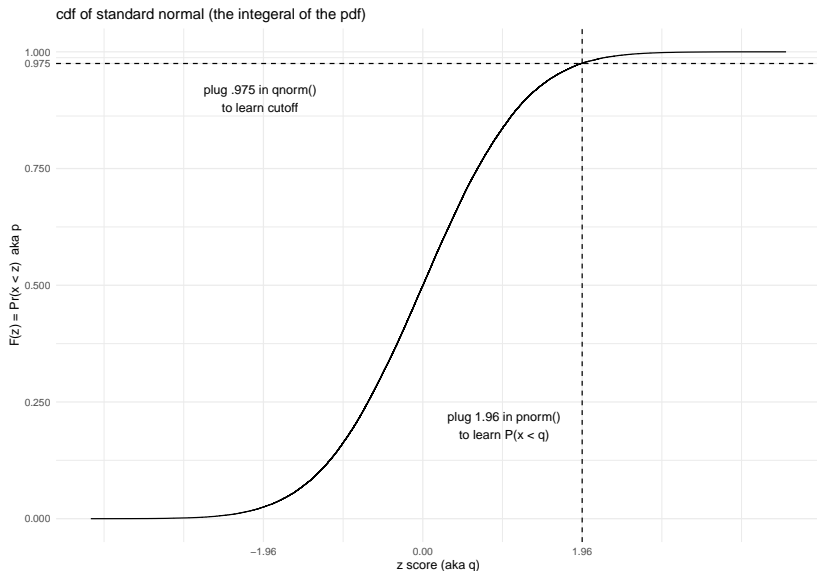
What are p and q?

pdf of standard normal

area under curve is the probability of being less than a cutoff



What are p and q?



What are p and q?

`pnorm` returns the probability we observe a value less than or equal to some value q .

```
pnorm(1.96)
```

```
## [1] 0.9750021
```

```
pnorm(0)
```

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

`qnorm` returns the inverse of `pnorm`. Plug in the probability and get the cutoff.

```
qnorm(.975)
```

```
## [1] 1.959964
```

```
qnorm(.5)
```

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


Monte Carlo experiments

Monte Carlo is a world gambling hub.

- ▶ Gamblers know that roulette wheels are not made perfectly.
- ▶ If you watch the wheel long enough and take notes you can figure out the empirical probability



Monte Carlo experiments

Statisticians use the same idea.

- ▶ If we're not sure how to calculate something exactly, we can simulate it and get the result.
- ▶ Often used for difficult to compute integrals.

An example

In real life experiments we usually have one sample.

```
# Setting a seed ensures replicability
set.seed(4)

# we set our parameters
true_mean <- 0.5
N <- 30

# We simulate and observe outcomes
simulated_data <- rnorm(N, mean = true_mean)
obs_mean <- mean(simulated_data)
obs_mean

## [1] 0.9871873
```

Put that number in perspective with a z score

```
obs_sd <- sd(simulated_data)
zscore <- (obs_mean - true_mean) / (obs_sd / sqrt(N))
zscore
```

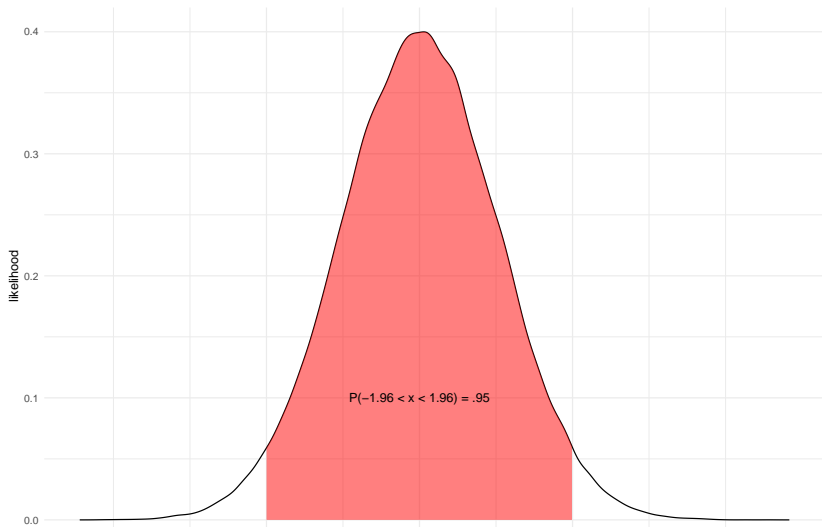
```
## [1] 3.303849
```

```
1 - pnorm(zscore)
```

```
## [1] 0.000476836
```

Our monte carlo experiment

Show that the sample mean from a sample drawn from a normal distribution falls outside the 95 percent confidence region 5 percent of the time.



How to do a Monte Carlo Simulation

1. Generate random samples of data using a known process (e.g. `rnorm()`).
2. Make calculations based on the random sample.
3. Aggregate the results.

```
do_monte_carlo <- function(N, true_mean, B, alpha){  
  sample_statistics <- make_mc_samples(N, true_mean, B)  
  z_scores <- get_zscores(sample_statistics$mean, true_mean)  
  test_significance(z_scores, alpha) %>% mean()  
}
```

Lab 4

- ▶ Two “types” of breakout room:
 - ▶ Work along: larger group with more guidance from a TA.
 - ▶ Small groups: 4 people, TAs will come in and out to answer questions. Use the help button!
- ▶ Add “(work along)” or “(small group)” to your Zoom name so we can sort you.
- ▶ Get as far as you can, then finish it up after class.
- ▶ Before you leave, fill out the exit poll.