Week 5: Distributions

SOC6302 Winter 2023

1 By the end of this lab you should know

- how to calculate z-scores
- how to calculate probabilities of normal distribution
- how to simulate from a normal distribution
- for loops
- how to plot simulation results as a histogram

2 Call the packages we need

In most labs we are going to use functions in the tidyverse package, so we load the package first.

```
library(tidyverse) # need to load in this to define a tibble!
```

3 Calculating z-scores

Let's look at three different heights: Monica's is 168cm, Rohan's is 180cm, and Teddy's is 87cm. Define these as variables

```
m_height <- 168
r_height <- 180
t_height <- 87
```

Now pretend we know that the distribution of adult heights in the population has a mean of 170cm and a standard deviation of 8cm. Define these parameters as variables too:

```r
mu <- 170
sigma <- 8
```

To calculate the z-scores, we just follow the standardization formula given in the slides. Can define these as new variables. NOTE: order of operations means we need to put parentheses around the numerator:

```r
m_z_score <- (m_height - mu)/sigma
r_z_score <- (r_height - mu)/sigma
t_z_score <- (t_height - mu)/sigma
```

3.1 Questions

Take a look at these z-scores. Who is closest to the mean? What's the interpretation of Teddy's z-score? Intuitively, how likely do you think Teddy is a fully grown adult, based on the z-score?

4 Calculating probabilities

We can calculate the probability of observing a value less than a specified value in a normal distribution using the `pnorm` function. For example, let's calculate the probability of observing a value less than Monica's z-score:

```r
?pnorm
pnorm(q = m_z_score, mean = 0, sd = 1)
```

```
[1] 0.4012937
```

Note: the default of `pnorm` is to return P(X <= x), i.e. P(any value <= the observed value). To calculate the probability than any observed value is **greater than** the observed value, you can change the argument in the function to be `lower.tail = FALSE`

```r
pnorm(q = m_z_score, mean = 0, sd = 1, lower.tail = FALSE)
```

```
[1] 0.5987063
```

Alternatively, we know that the sum of all probabilities has to equal 1, so could calculate

```
1 - pnorm(q = m_z_score, mean = 0, sd = 1)
```

```
[1] 0.5987063
```

We also need not use z-score, could get the probability of observing a height less than Monica's by changing the arguments for mean and standard deviation in the function:

```
pnorm(q = m_height, mean = mu, sd = sigma)
```

```
[1] 0.4012937
```

To get the probability that a value is between two values, take the difference e.g probability that someone's height is between 190 and 200cm

```
pnorm(q = 200, mean = mu, sd = sigma) - pnorm(q = 190, mean = mu, sd = sigma)
```

```
[1] 0.006121248
```

4.1 Questions

- Calculate the probability of observing a height between Rohan and Monica's
- Calculate the probability of observing a height that is greater than Teddy's

5 Simulating from a normal distribution

We can simulate hypothetical values from a normal distribution using the `rnorm` function. For example, let's simulate 6 different heights from our underlying distribution of adult heights:

```
rnorm(n = 6, mean = mu, sd = sigma)
```

```
[1] 174.9920 177.5328 166.3086 177.4444 173.9764 171.3699
```

Note: execute the above code over and over and notice that the numbers change (they are random draws). To make them the same each time we execute the code, you can set a seed, which tells the computer where to start its random generation.

```
set.seed(876)
rnorm(n = 6, mean = mu, sd = sigma)
```

```
[1] 171.3672 176.2573 161.4867 167.8852 170.9095 185.5180
```

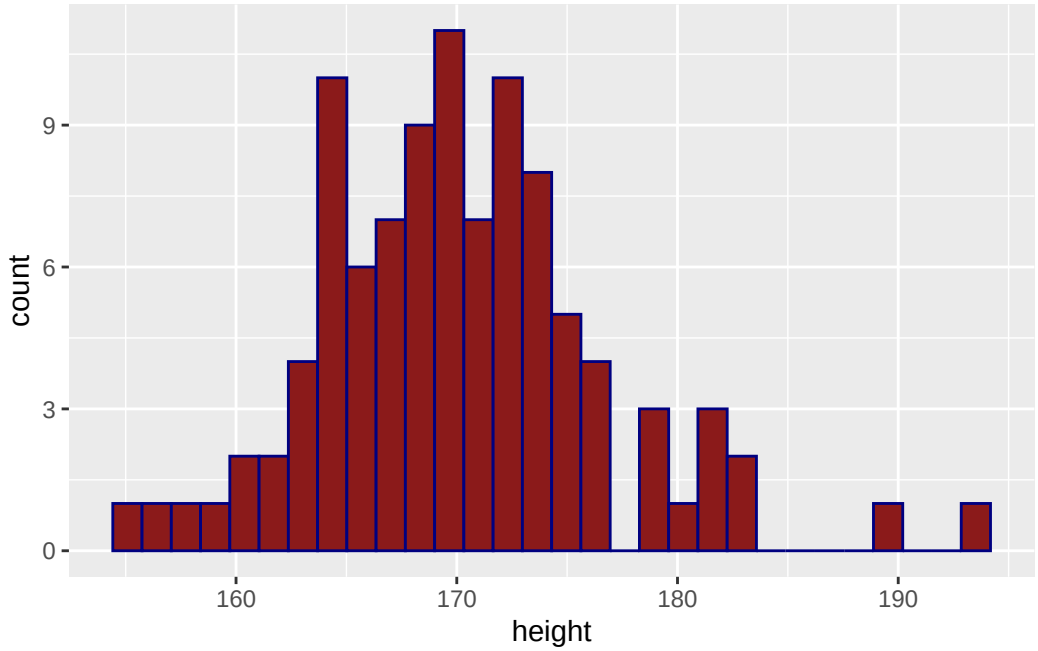
Run this over and over and notice the numbers don't change. The number in the `set.seed` function can be anything you want.

Now let's simulation 1,000 heights, and save these in a tibble.

```
sample_heights <- tibble(person_number = 1:100, height = rnorm(100, mean = mu, sd = sigma))
```

Now plot these heights as a histogram (based on skills from last week)

```
ggplot(data = sample_heights, aes(height)) +
  geom_histogram(color = "navy", fill = "firebrick4")
```



5.1 Question

Calculate the mean and standard deviation of the heights in `sample_heights`

6 More than one sample

We have one set of 100 heights. Now pretend we wanted another sample of 100 heights. We could do exactly the same thing, and call our second dataframe `sample_heights_2`, for example

```
sample_heights_2 <- tibble(person_number = 1:100, height = rnorm(100, mean = mu, sd = sigm
```

We could create an extra column called sample_number that would tell us which sample the person's height was drawn from, and then join `sample_heights` and `sample_heights_2` together into the one dataframe/tibble using a function called `bind_rows`

```
sample_heights <- sample_heights |> mutate(sample_number = 1)
sample_heights_2 <- sample_heights_2 |> mutate(sample_number = 2)

all_sample_heights <- bind_rows(sample_heights, sample_heights_2)
```

6.1 Questions

- Take a look at `all_sample_heights` and make sure you understand what happened.
- Calculate the mean and standard deviation of each sample of heights (hint: use `group_by`)

7 Many, many samples

The above is okay, but say I wanted to get 1000 samples like this. This is a lot of copy pasting and would be very time consuming. Instead, we can use a for loops to get R to do this for us automatically.

Remember that for loops are based on looping over an iterating variable (usually called `i`) and telling R to do something each time. The following code tells R to loop over `i` 3 times and each time print out a little message, which changes based on the loop

```
for(i in 1:3){ # for iterator i going from number 1 to number 3
  print(paste("Hello, you are in loop", i))
}
```

```
[1] "Hello, you are in loop 1"
[1] "Hello, you are in loop 2"
[1] "Hello, you are in loop 3"
```

We could do the same thing N times, where N is a variable we create and can change (below N is defined as 10)

```
N <- 10

for(i in 1:N){ # for iterator i going from number 1 to N
  print(paste("Hello, you are in loop", i))
}
```

```
[1] "Hello, you are in loop 1"
[1] "Hello, you are in loop 2"
[1] "Hello, you are in loop 3"
[1] "Hello, you are in loop 4"
[1] "Hello, you are in loop 5"
[1] "Hello, you are in loop 6"
[1] "Hello, you are in loop 7"
[1] "Hello, you are in loop 8"
[1] "Hello, you are in loop 9"
[1] "Hello, you are in loop 10"
```

For loops are useful if you want to do the same thing over and over just changing something slightly. In our case, that's exactly what we want to do – sample 100 people over and over (1000 times), and just note down the sample number.

Here's the code to do it twice. Firstly BEFORE the loop, I'm defining an empty tibble to store my results in. Then within the loops for each iteration

- Create a sample of 100 people, and define the column `sample_number` based on the iteration number i
- Bind that sample onto the tibble that all results are stored in

```
sample_heights <- tibble()

for(i in 1:2){
  this_sample <- tibble(person_number = 1:100,
                        height = rnorm(100, mean = mu, sd = sigma),
                        sample_number = i)
  sample_heights <- bind_rows(sample_heights, this_sample)
}
```

7.1 Questions

- Alter the code above to do this 1000 times
- Calculate the mean height for each sample
- Plot the means as a histogram