Week 11: Sampling Live R

Monica Truelove-Hill

2022-11-26

In the lecture this week, we covered Samples and Sampling Distributions. Today, we'll explore these ideas further. Specifically, we'll examine random samples from a large dataset, and use a new function to explore variability within these samples, given a specific n and sample number.

First, let's load the tidyverse. I'm also going to set a fill color for my histograms here.

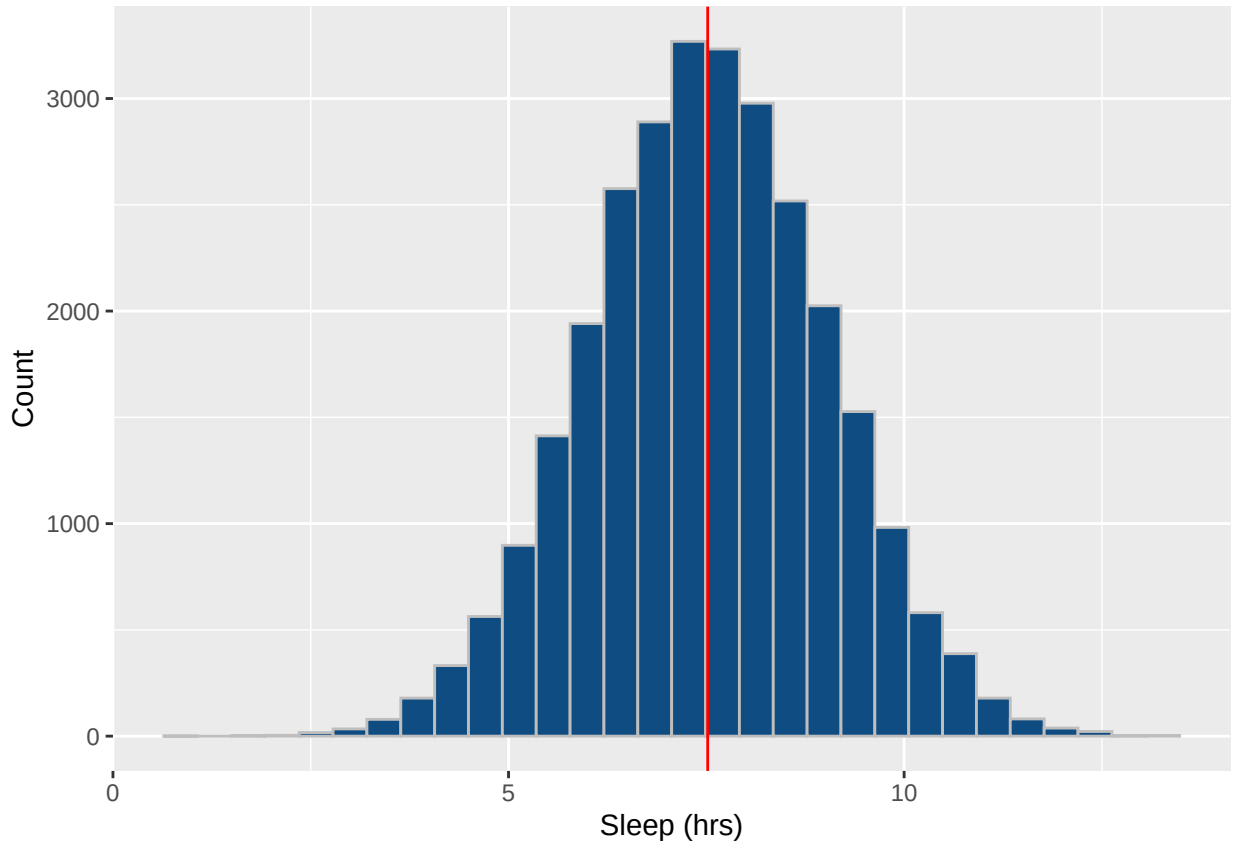
```
library(tidyverse)
baseColor <- '#0F4C81'
```

Let's say that we are interested in the sleeping habits of all students within the University of Edinburgh. To investigate this, let's first create a dataset that represents our population of interest. The variable we'll examine is average hours of sleep per night. We'll assume that hours of sleep is normally distributed, so we'll use the `rnorm` function to generate sample data.

```
set.seed(1022)
dat <- tibble(mSleep = rnorm(28755, mean = 7.53, sd = 1.5),
              Year=sample(c('Y1', 'Y2', 'Y3', 'Y4'), size = 28755, replace = T))
```

Now, let's visualise the data using a histogram:

```
ggplot(dat, aes(mSleep)) + geom_histogram(color='gray', fill = baseColor) +
  geom_vline(xintercept = mean(dat$mSleep), colour = 'red') +
  labs(x = 'Sleep (hrs)', y = 'Count')
```



We can see here that the data are normally distributed, which is as expected since we used the `rnorm` function to generate them, and `rnorm` produces data that follow a normal distribution with a given mean and standard deviation.

Since everything appears to be in order with our population, let's take some samples and see how well they represent our population. To do this, we'll use a new function, `rep_sample_n`.

```
source('https://uoepsy.github.io/files/rep_sample_n.R')
```

What happens when you run this line of code? You'll notice that the function appears as an object in your environment. You can now use it as you would any other function. If you try to run the function before sourcing it (or if you've removed it from your environment), you will get an error. You will need to source the function every time you open RStudio (in the same way you need to load packages from the library each time you open RStudio).

`rep_sample_n` is a function that selects samples from a dataset. Unlike the `sample` function, however, it allows you to specify how many samples you would like to produce. It requires three arguments: `rep_sample_n(data, n = <sample size>, samples = <how many samples>)`

Let's take two samples of only 5 participants from our data:

```
rep_sample_n(dat, 5, 2)
```

```
## # A tibble: 10 x 3
##    sample mSleep Year
##     <dbl>  <dbl> <chr>
##  1      1   7.84 Y3
```

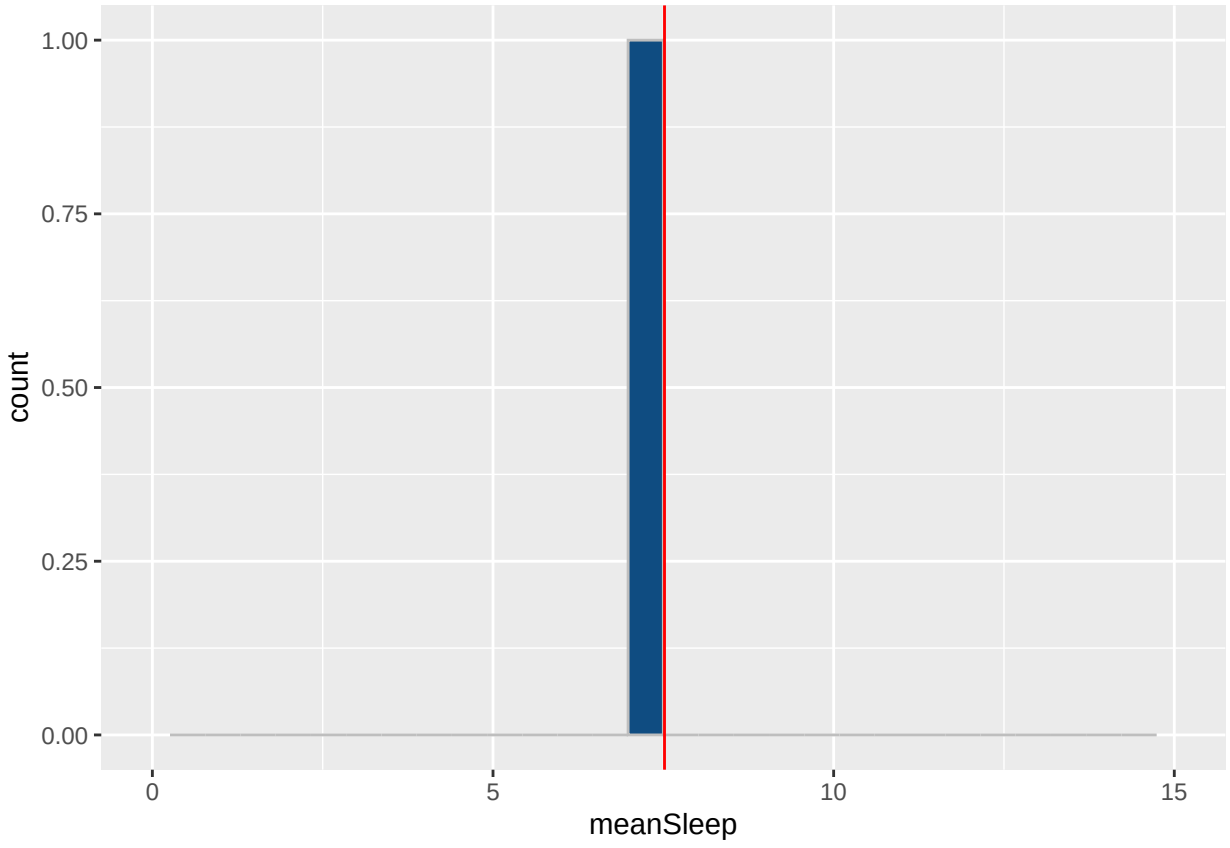
```
##  2      1   8.03 Y4
##  3      1   5.64 Y3
##  4      1   7.07 Y4
##  5      1   8.65 Y4
##  6      2   6.65 Y2
##  7      2   9.07 Y3
##  8      2   7.45 Y2
##  9      2   7.58 Y4
## 10      2   7.37 Y4
```

`rep_sample_n` takes the above arguments, and outputs a tibble with the sample number in one column and the sample values in a second column. In our example, the output has ten rows (2 samples X 5 participants each) and two columns. The column *sample* gives us the sample number, and the column *mSleep* gives us each participant's average hours of sleep.

How does sample number affect the sampling distribution?

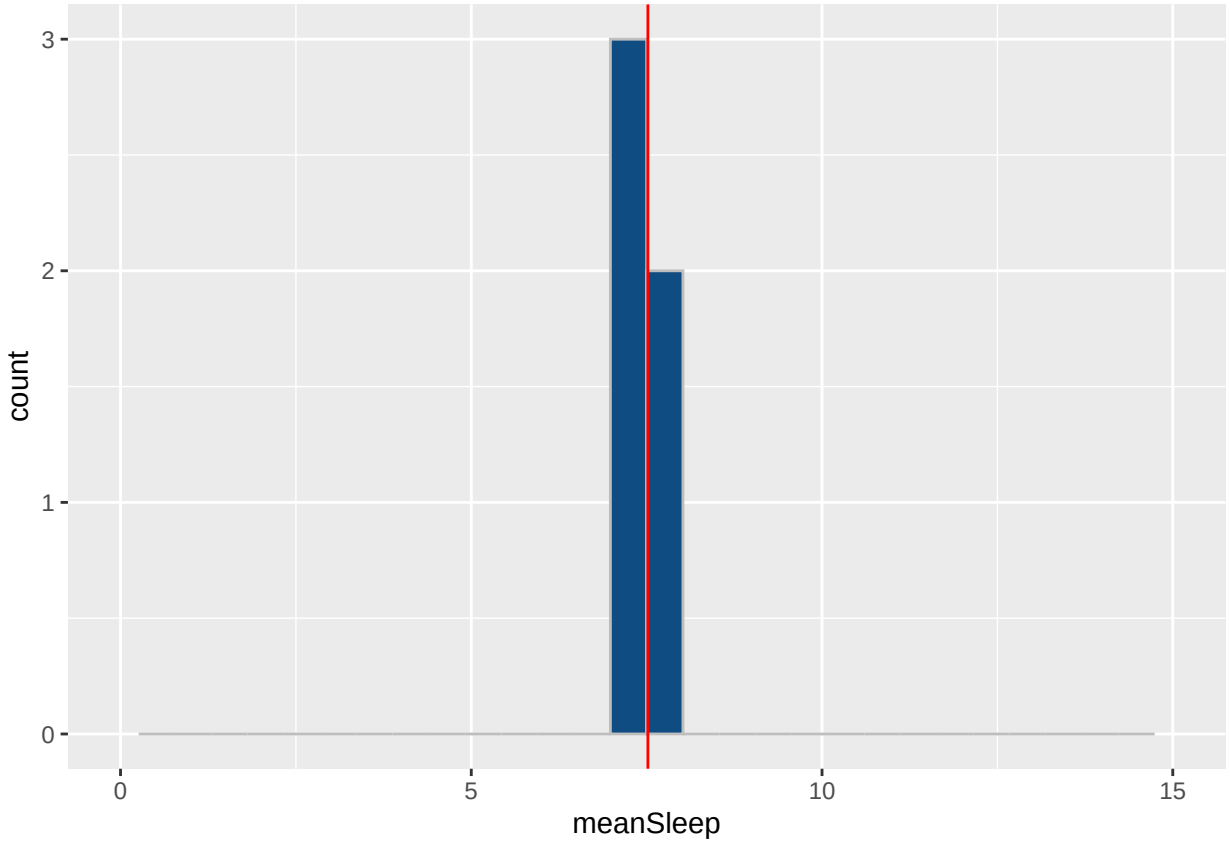
Let's start with a single sample of 10 participants.

```
rep_sample_n(dat, 10, 1) %>%
  summarise(meanSleep = mean(mSleep)) %>%
  ggplot(., aes(meanSleep)) + geom_histogram(color = 'gray', fill = baseColor) +
  scale_x_continuous(limits = c(0, 15)) +
  geom_vline(xintercept = mean(dat$mSleep), colour = 'red')
```



What happens to the distribution if we increase the number of samples? Note that we'll have to add the `group_by` function to the code to get this to work properly. The `group_by` function allows us to specify a column within our data by which we'd like to group rows. In this case, we want to specify that each sample is an individual group, and summarise the data by group rather than across the dataset as a whole. In this way, a mean will be computed separately for each sample.

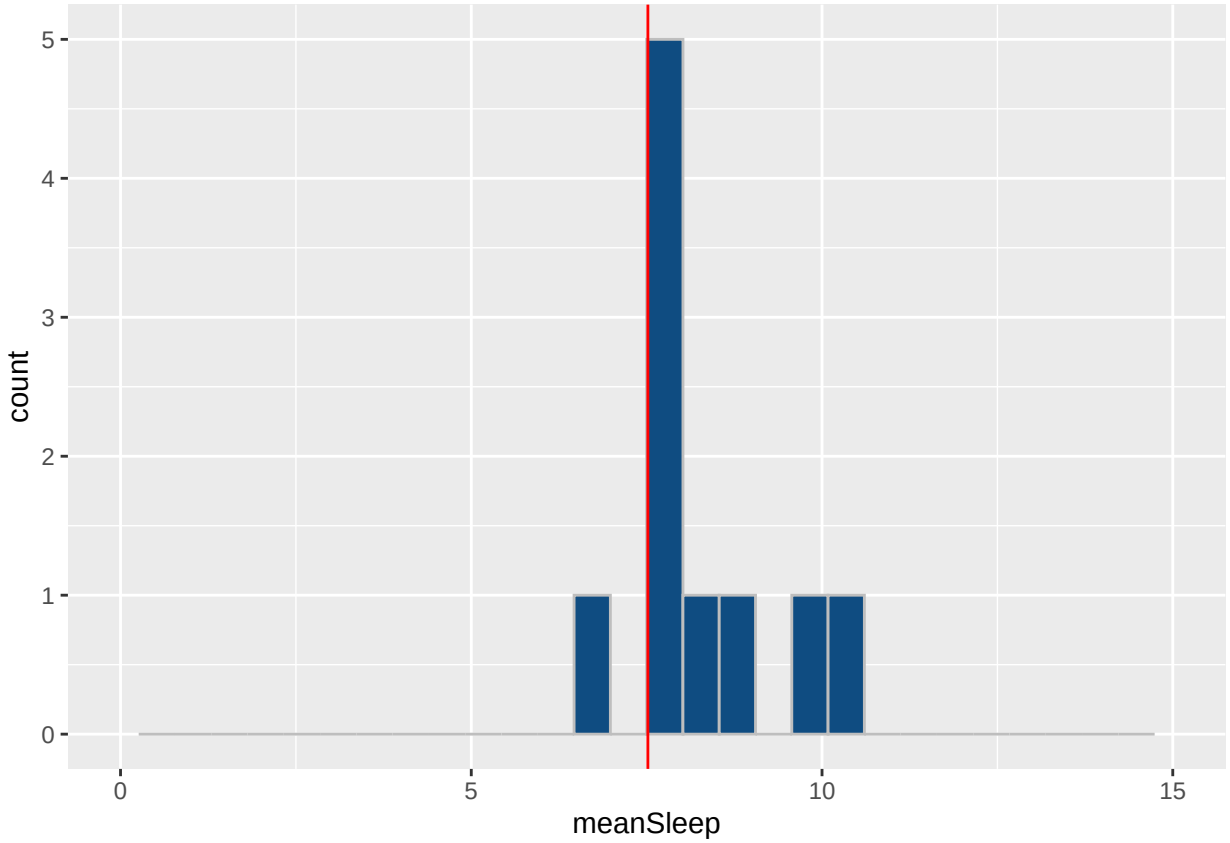
```
rep_sample_n(dat, 10, 5) %>%
  group_by(sample) %>%
  summarise(meanSleep = mean(mSleep)) %>%
  ggplot(., aes(meanSleep)) + geom_histogram(color = 'gray', fill = baseColor) +
  scale_x_continuous(limits=c(0, 15)) +
  geom_vline(xintercept = mean(dat$mSleep), colour = 'red')
```



How does sample size affect the sampling distribution?

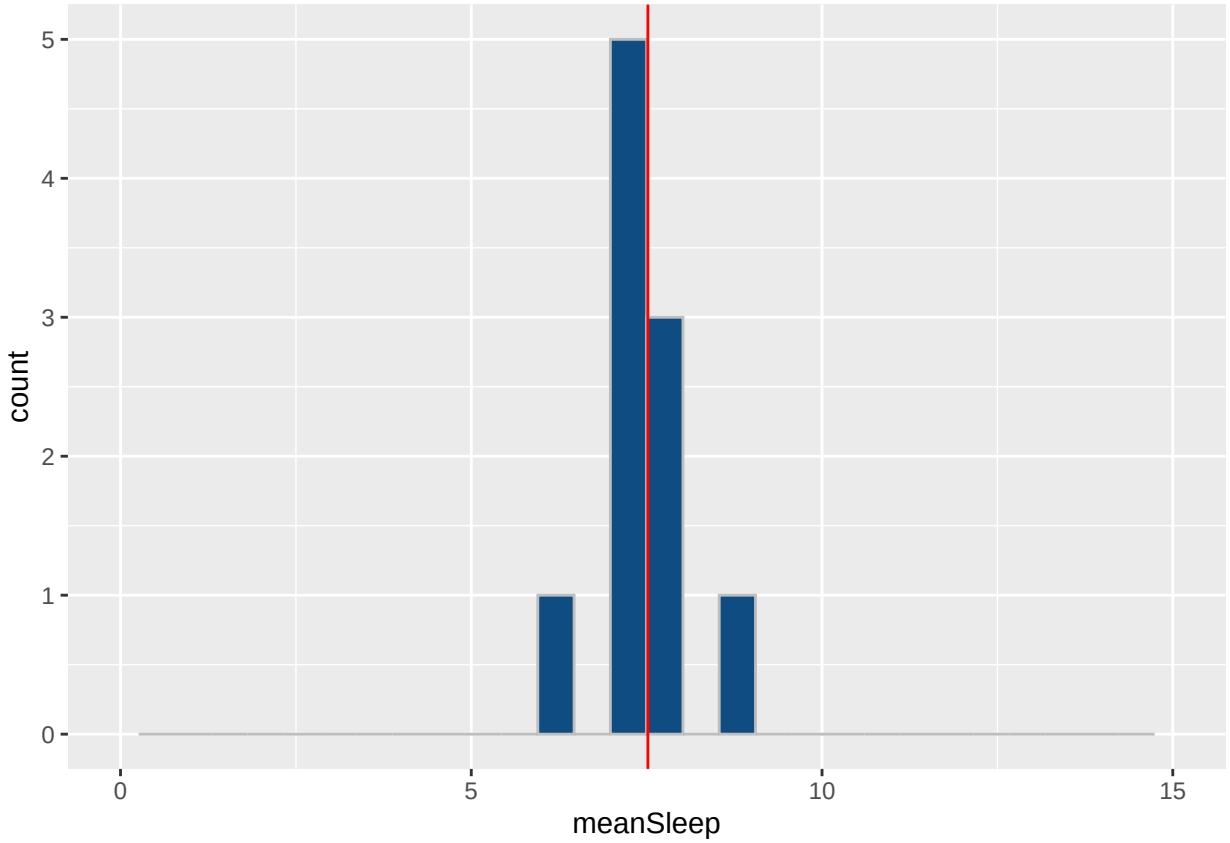
Let's start with 10 samples of only 1 participant.

```
rep_sample_n(dat, 1, 10) %>%
  group_by(sample) %>%
  summarise(meanSleep = mean(mSleep)) %>%
  ggplot(., aes(meanSleep)) + geom_histogram(color = 'gray', fill = baseColor) +
  scale_x_continuous(limits=c(0, 15)) +
  geom_vline(xintercept = mean(dat$mSleep), colour = 'red')
```



What happens as we increase our sample size? Let's incrementally increase it by 5.

```
rep_sample_n(dat, 5, 10) %>%
  group_by(sample) %>%
  summarise(meanSleep = mean(mSleep)) %>%
  ggplot(., aes(meanSleep)) + geom_histogram(color = 'gray', fill = baseColor) +
  scale_x_continuous(limits=c(0, 15)) +
  geom_vline(xintercept = mean(dat$mSleep), colour = 'red')
```

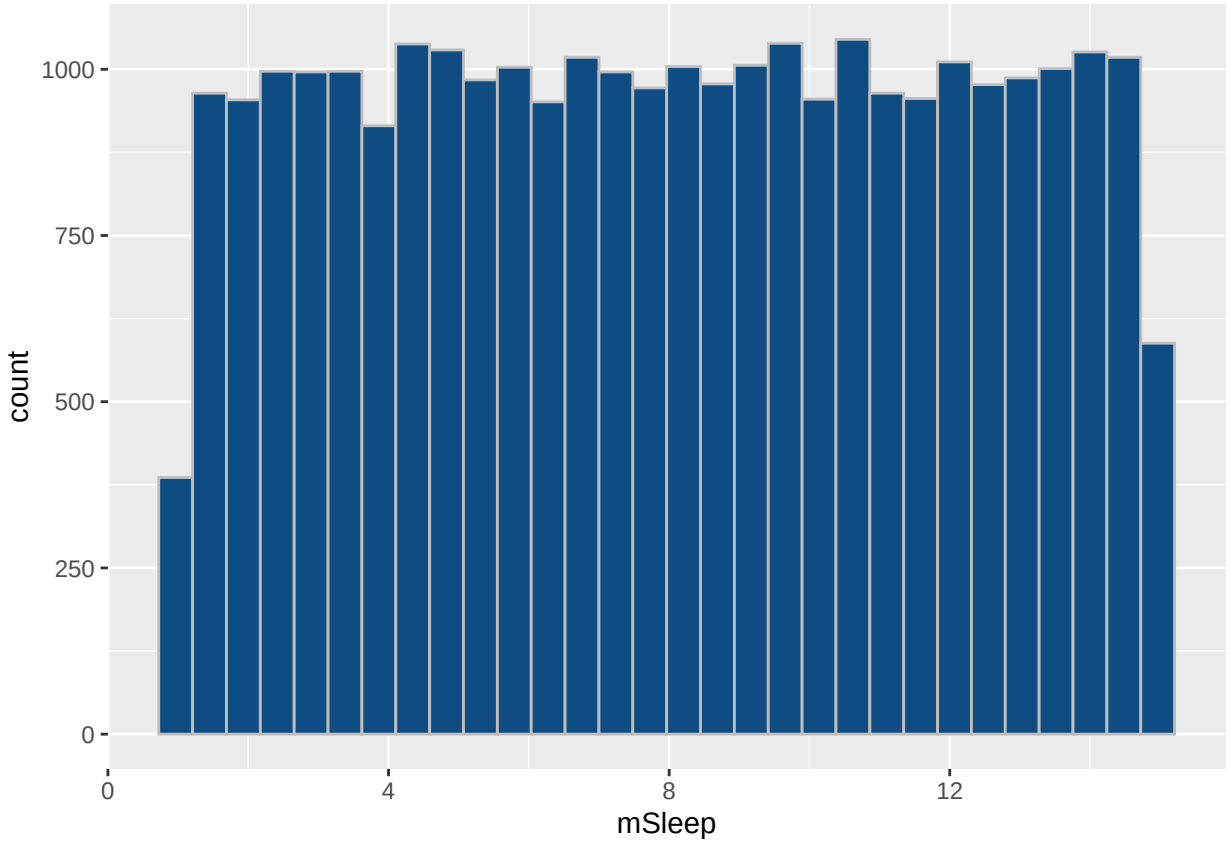


Other distributions

Now, let's imagine that somehow, sleep is actually uniformly distributed, so that people are equally likely to be sleeping 1 hour on average or 15 hours on average.

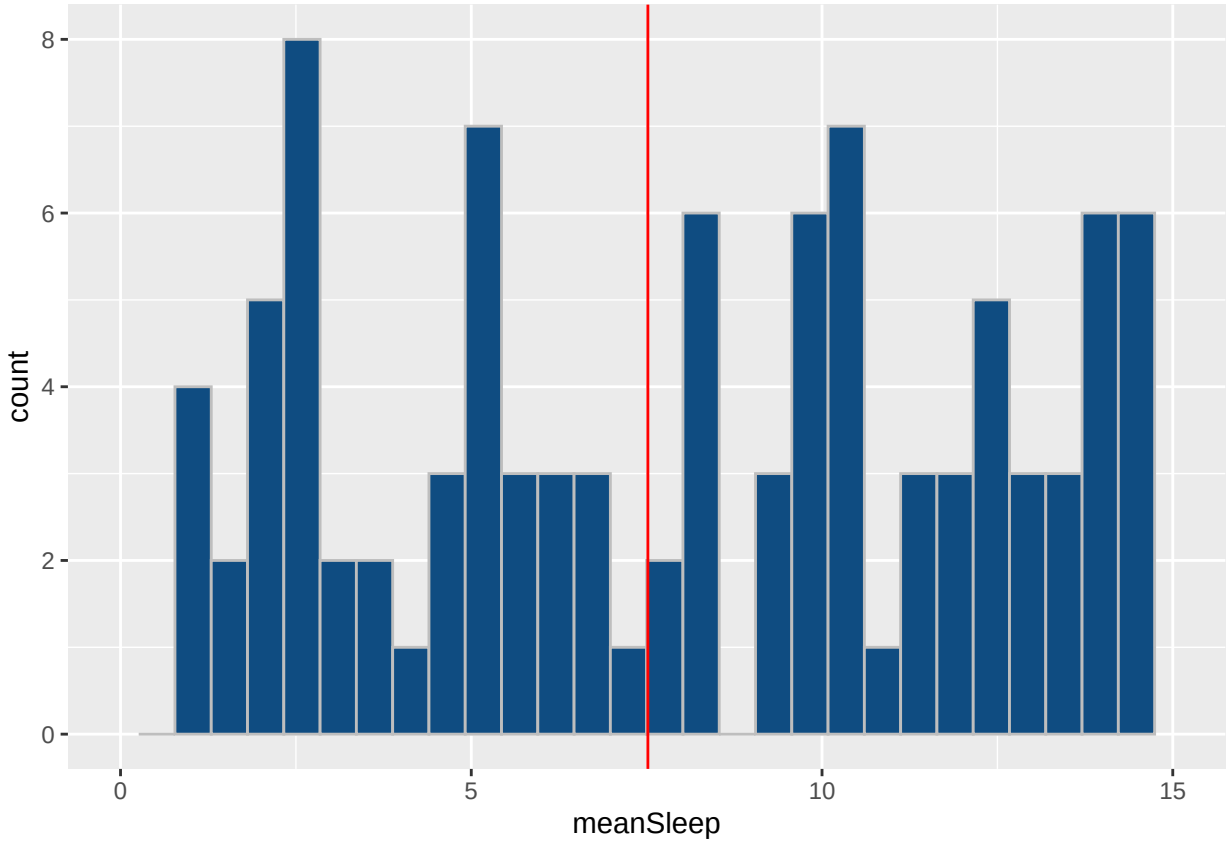
```
sleepUni <- tibble(mSleep=runif(nrow(dat), min=1, max = 15))

ggplot(sleepUni, aes(mSleep)) + geom_histogram(colour = 'gray', fill = baseColor)
```

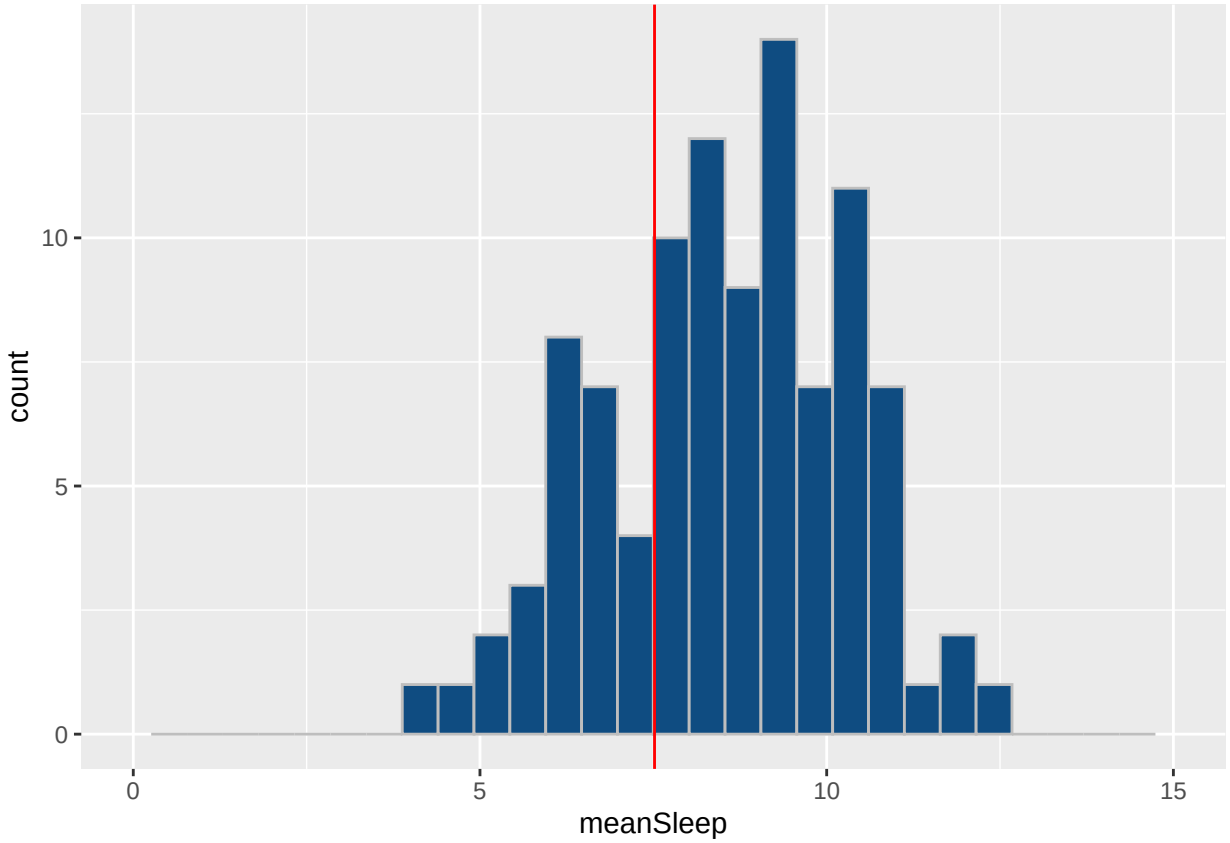


To demonstrate the central limit theorem, let's draw increasingly larger samples and see what happens to our sampling distribution.

```
rep_sample_n(sleepUni, 1, 100) %>%
  group_by(sample) %>%
  summarise(meanSleep = mean(mSleep)) %>%
  ggplot(., aes(meanSleep)) + geom_histogram(color = 'gray', fill = baseColor) +
  scale_x_continuous(limits=c(0, 15)) +
  geom_vline(xintercept = mean(dat$mSleep), colour = 'red')
```




```
rep_sample_n(sleepUni, 5, 100) %>%
  group_by(sample) %>%
  summarise(meanSleep = mean(mSleep)) %>%
  ggplot(., aes(meanSleep)) + geom_histogram(color = 'gray', fill = baseColor) +
  scale_x_continuous(limits=c(0, 15)) +
  geom_vline(xintercept = mean(dat$mSleep), colour = 'red')
```



How are our data distributed by year in Uni?

Finally, let's use the grouping variable, `Year` to visually investigate any differences in sleep by year in Uni.

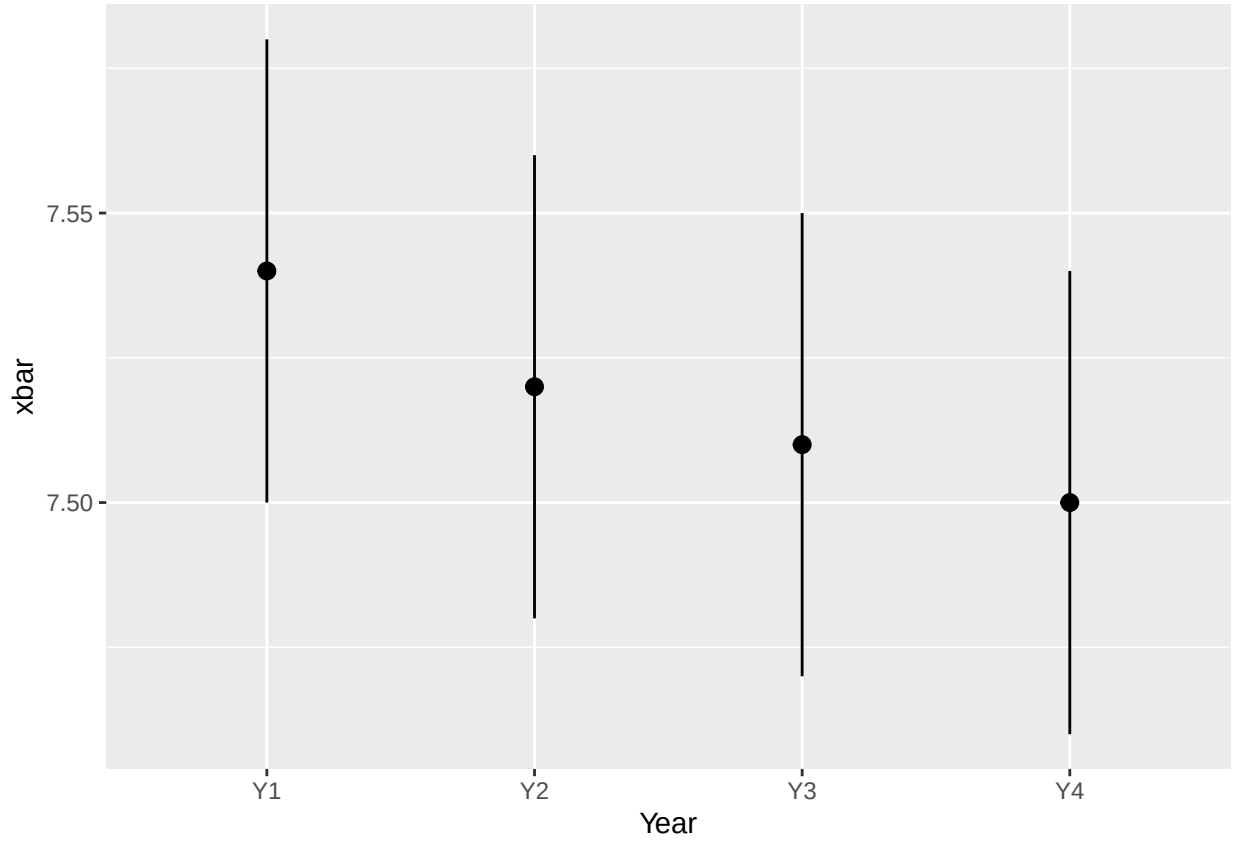
```
tbl_Year <- dat %>%
    group_by(Year) %>%
    summarise(n = n(),
              xbar = mean(mSleep),
              se = sd(mSleep) / sqrt(n)) %>%
    mutate(xbar = round(xbar, 2),
           se = round(se, 2)) %>%
    as.data.frame()

tbl_Year
```

```
##   Year    n xbar   se
## 1   Y1 7187 7.54 0.02
## 2   Y2 7260 7.52 0.02
## 3   Y3 7085 7.51 0.02
## 4   Y4 7223 7.50 0.02
```

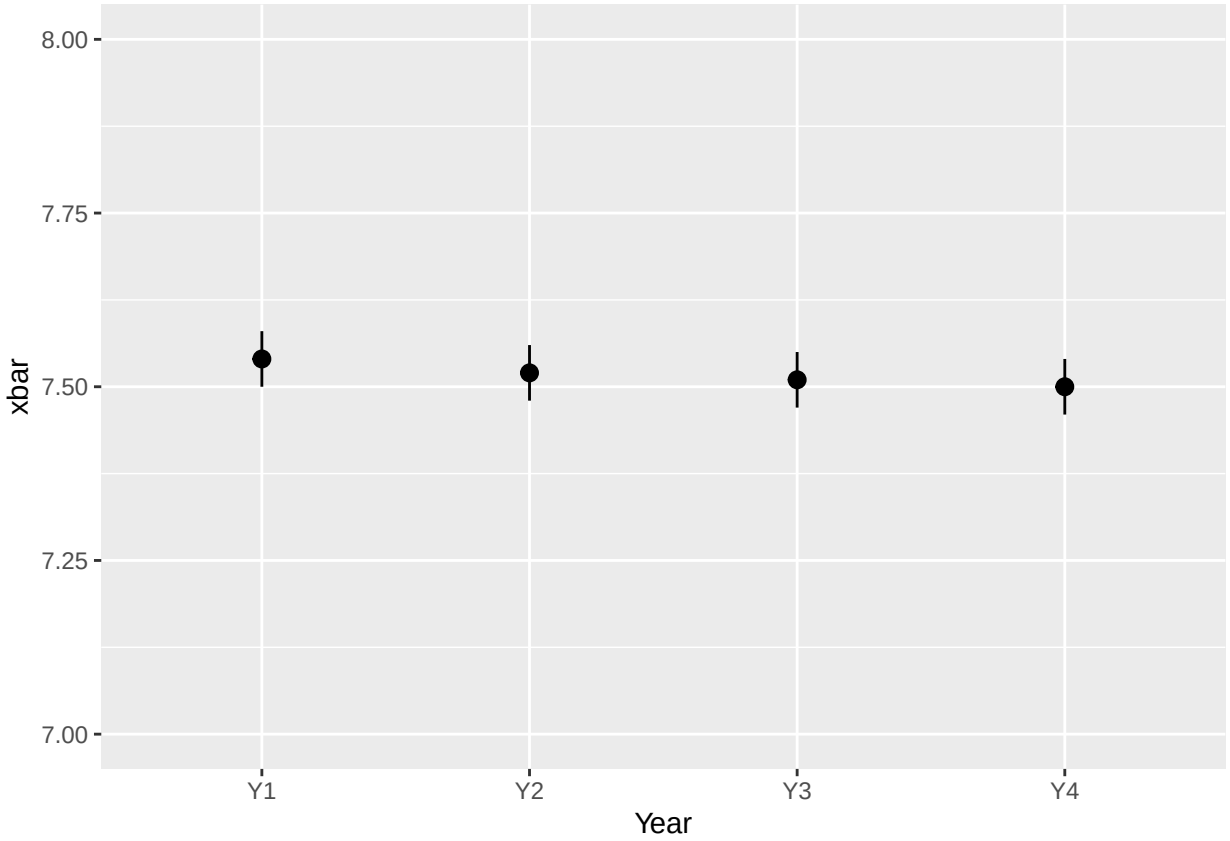
Here, we've created a table that has the mean hours of sleep by students in each year in Uni. Let's plot this table to visualise these means by group:

```
ggplot(tbl_Year, aes(x=Year, y=xbar, ymin=xbar-2*se, ymax=xbar+2*se)) + geom_pointrange()
```



If we look at this plot as is, it looks as though there's a negative relationship between year in Uni and mean sleep. However, the y-axis is a bit misleading here. Look at the scaling! It's quite small. Let's adjust the scaling on the y-axis using `scale_y_continuous`:

```
ggplot(tbl_Year, aes(x=Year, y=xbar, ymin=xbar-2*se, ymax=xbar+2*se)) + geom_pointrange() +
  scale_y_continuous(limits=c(7, 8), breaks = seq(7, 8, .25))
```



By adjusting the scale, we can see that there's really not much difference between our groups in the overall population.