Sampling Distributions Lecture

Dr. J (PSYC417)

This is a markdown file alligning with the concepts we covered during the first part of the resampling methods lecture. The Rodgers (1999) reading is a great pairing to this code (it's not dry at all *for a methods piece*, for what that's worth).

To understand how our statistical methods work, consider the following *sampling distributions*:

- The **idealized sampling distribution**: The true sampling distribution for a statistic given a sample size and random samples from the population (computing the statistic in each random sample). We never have access to this. We don't often have access to the entire population, at least enough to conduct many, many experiments. Because we don't have this, we try to approximate it one of two ways described below.
- The **theoretical sampling distribution**: A distribution mathematically derived by statisticians that follows known properties. (For example, normal distributions, uniform distributions). Because these are mathematically derived, we know how these values group up in terms of density and probability of observation. Normal distributions have many observations toward the center (mean, median) and fewer toward the "tails", for example. We *assume* that the statistics we're interested in (e.g., regression coefficients) follow certain sampling distributions (e.g., normal) so we can compute things like p-values.
- The **empirical sampling distribution**: A computationally-generated sampling distribution generated from a resampling method (e.g., the bootstrap). The goal is to treat the sample data or estimates from the sample as representative of the population. With that, we can use that information to conduct many "fake studies" that allow us to create a distribution of estimates and, thus, approximate the uncertainty.

The data we'll be using to illustrate these concepts come from our class. You all filled out a survey on your enjoyment of Taylor Swift and video games. We're assuming the 18 of you in attendance are the "population" of interest. (Yes, populations are bigger than this, but we work with what we're given. This is for illustration.) Recall I had to triple the dataset just so we could have a somewhat reasonable population size. I will use expressions like "18 of you, randomly sample 10" and so on, but the code will have values triple to these to account for

the artificial increase in size. (i.e., When I say, "samples of size 10 from the population of 18", you will see samples of size 30 from the population of 54).

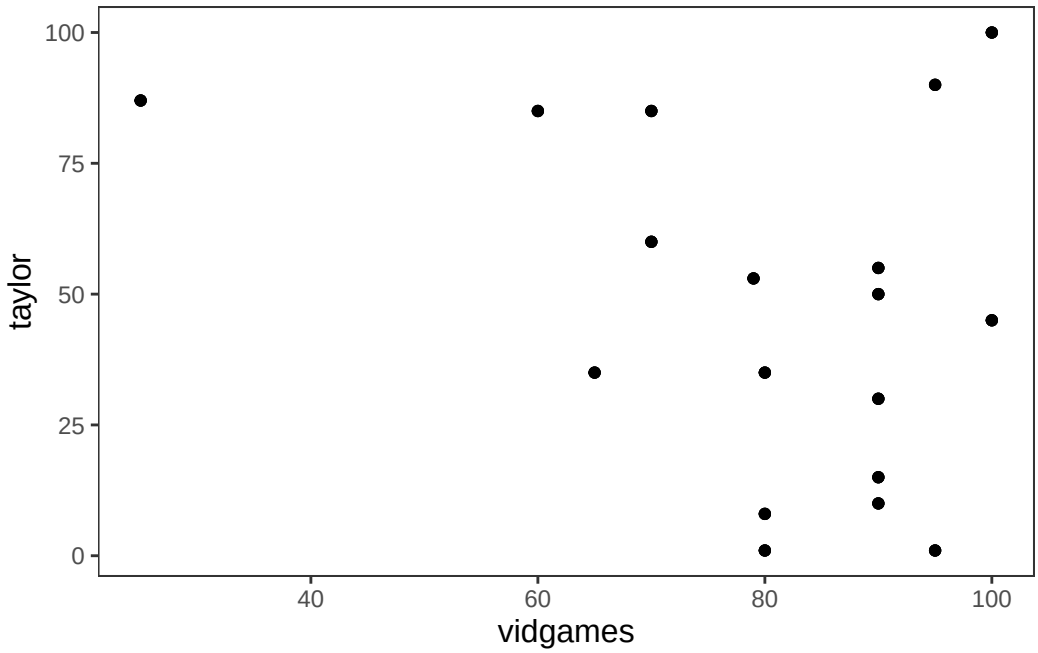
Forming an idealized sampling distribution

To do this, we first need to load relevant packages, read in the data, and duplicate it three times so we have more information to work with. Let's also compute descriptives and visualize the data while we're at it.

```
library(ggplot2) # for nice graphs
library(jtools) # for easy APA figures

# read in the data
tsvg <- read.table("E:/Teaching Materials/PSYC417 (Data Science)/Datasets/ts_vg.csv",heade

# triple the size of the data
tsvg_dat <- rbind(tsvg,tsvg,tsvg)

# summarize the data
summary(tsvg_dat)
```

```
    taylor          vidgames
Min.   :  1.00   Min.   : 25.0
1st Qu.: 15.00   1st Qu.: 70.0
Median : 47.50   Median : 85.0
Mean   : 46.94   Mean   : 80.5
3rd Qu.: 85.00   3rd Qu.: 90.0
Max.   :100.00   Max.   :100.0
```

```
# visualize the data
ggplot(tsvg_dat, aes(x = vidgames, y = taylor)) +
  geom_point() +
  jtools::theme_apa()
```



We have two variables: enjoyment of Taylor Swift and enjoyment of video games. Let's just see how these two variables are related in our "population."

```
cor.test(tsvg_dat$taylor, tsvg_dat$vidgames)
```

```
    Pearson's product-moment correlation

data:  tsvg_dat$taylor and tsvg_dat$vidgames
t = -2.3065, df = 52, p-value = 0.0251
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.52923885 -0.04016595
sample estimates:
     cor
-0.30465
```

```
# obtain the test statistic
cor.test(tsvg_dat$taylor, tsvg_dat$vidgames)[1]$statist
```

```
        t
```

```
-2.306504
```

It looks like they are negatively associated (r = -0.30), meaning that those who like Taylor Swift more tend to like video games less and vice versa. It's great we know this, but we don't have access to the population in the real world. Instead, we collect samples. Let's take a sample of size 10 here and see what the relationship between the two looks like. I'm going to set the seed to 42700 so the results are consistent.

```
set.seed(42700)
sample1 <- tsvg_dat[sample(c(1:nrow(tsvg_dat)),30),]
cor.test(sample1$taylor, sample1$vidgames)
```

```
    Pearson's product-moment correlation

data:  sample1$taylor and sample1$vidgames
t = -0.93468, df = 28, p-value = 0.3579
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.5027114  0.1987805
sample estimates:
       cor
-0.1739457
```

Overall, our effect is different from that in the population. (Which is to be expected). What if we did this again with another sample of size 10?

```
sample2 <- tsvg_dat[sample(c(1:nrow(tsvg_dat)),30),]
cor.test(sample2$taylor, sample2$vidgames)
```

```
    Pearson's product-moment correlation

data:  sample2$taylor and sample2$vidgames
t = -2.3551, df = 28, p-value = 0.02576
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.66889525 -0.05429725
sample estimates:
       cor
-0.4066125
```

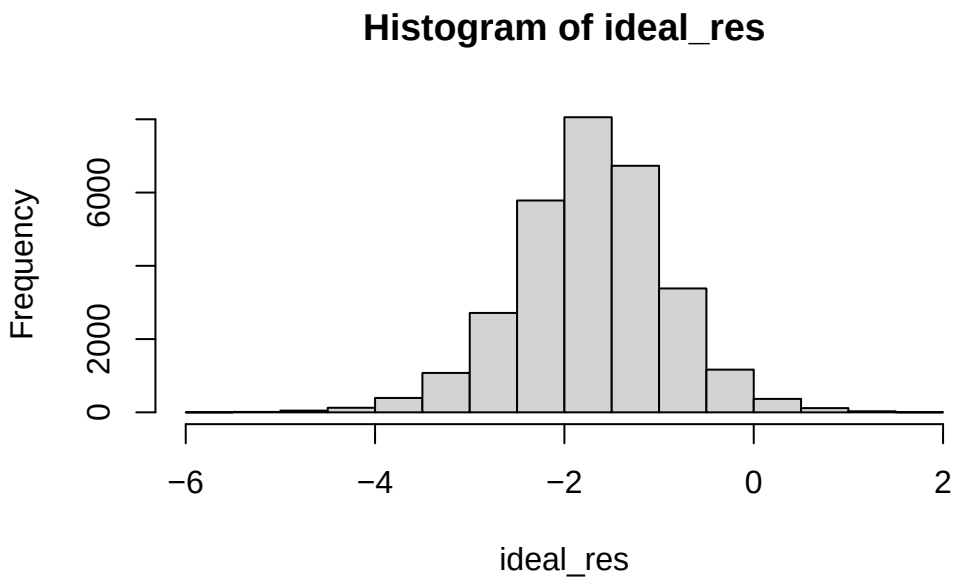
And another?

```
sample3 <- tsvg_dat[sample(c(1:nrow(tsvg_dat)),30),]
cor.test(sample3$taylor, sample3$vidgames)
```

```
    Pearson's product-moment correlation

data:  sample3$taylor and sample3$vidgames
t = -2.7533, df = 28, p-value = 0.01024
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.7046644 -0.1215117
sample estimates:
       cor
-0.4615746
```

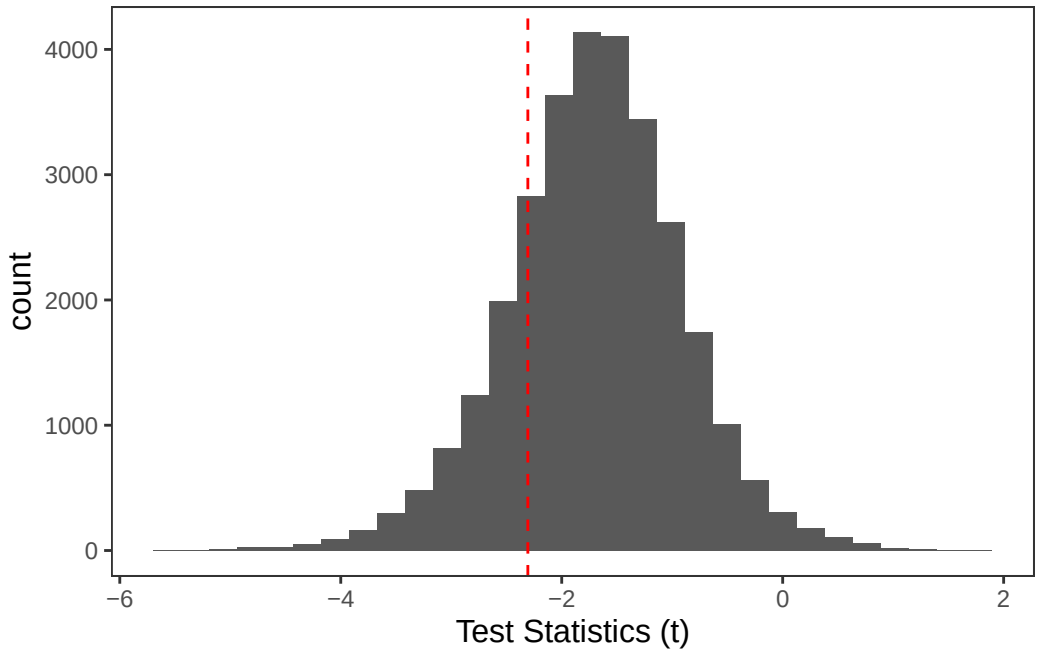
If we do this a large number of times, we can find the distribution of estimates we expect over repeated random samples of the same size (in this case, "10"). We're going to create a *for* loop for 30,000 imaginary studies where we randomly sample 10 of you and compute and store the correlation coefficient and *t*-statistic in each sample. We will plot these results with a histogram at the end.

```
# reset seed to 417
set.seed(417)

# create simulation parameters
trials <- 30000
ideal_res <- rep(NA,trials)
cor_res <- rep(NA,trials)

# loop through many "studies"
for(i in 1:trials){
  sample <- tsvg_dat[sample(c(1:nrow(tsvg_dat)),30),]
  ideal_res[i] <- cor.test(sample$taylor, sample$vidgames)[1]$statistic
  cor_res[i] <- cor(sample$taylor, sample$vidgames)
}

# quick plot of results
hist(ideal_res)
```



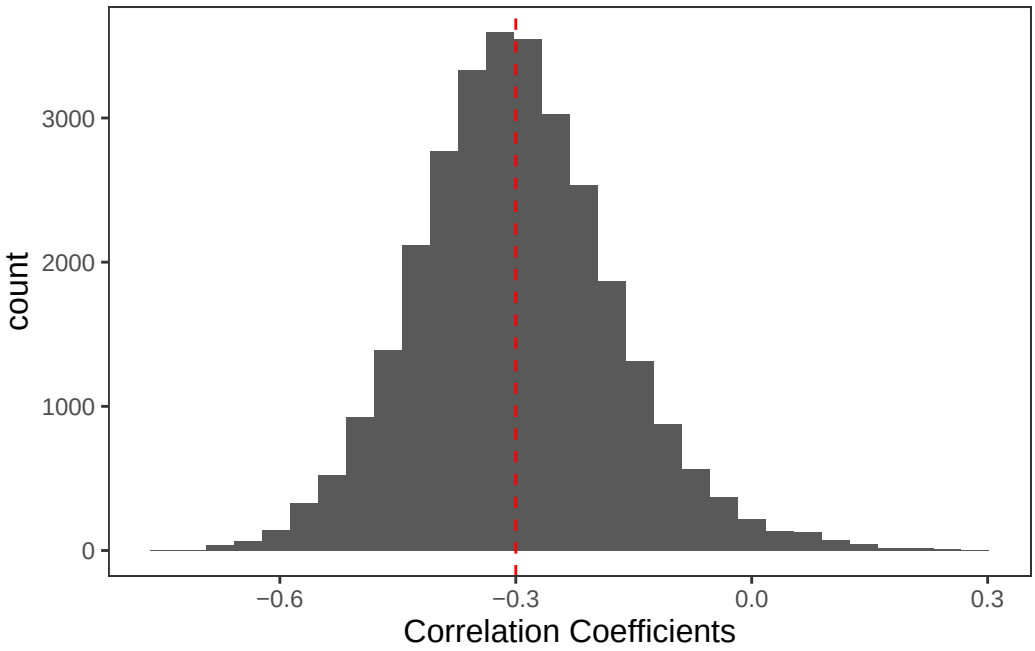

```
# prepare and plot in ggplot
ideal_df = data.frame(ideal_res = ideal_res, cor_res = cor_res)
ggplot(ideal_df, aes(x=ideal_res)) +
  geom_histogram() +
  jtools::theme_apa() +
  xlab("Test Statistics (t)") +
  geom_vline(xintercept=-2.3065, color = "red",linetype="dashed")
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
# prepare and plot in ggplot
ideal_df = data.frame(ideal_res = ideal_res, cor_res = cor_res)
ggplot(ideal_df, aes(x=cor_res)) +
  geom_histogram() +
  jtools::theme_apa() +
  xlab("Correlation Coefficients") +
  geom_vline(xintercept=-.30, color = "red",linetype="dashed")
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



That's quite the range/distribution of test statistics and coefficients, even in this example where our sample is a *majority* of the population. In traditional statistics, we assume that that sampling distribution is normal in shape. If that is true, we can rely on the properties of the normal distribution to estimate how unlikely it is we get a *test statistic* that extreme due to chance if the null hypothesis is true.

It looks like the distribution is normal, which is to be expected. It is often the case that regression coefficients (standardized and otherwise) are *normally distributed*. Thus, p-values and confidence intervals we construct can reliably be used for inference. For fun, let's look at the proportion of times in that study that we rejected the null hypothesis. (Since we know there's truly an effect in the population, this proportion will give us the observed statistical power.)

```
# find test statistic for statistical significance
ts = qt(.025, 28)

sum(abs(ideal_res) >= abs(ts))/trials
```

```
[1] 0.3154667
```

This isn't relevant to our current discussion, but it's interesting that only ~32% of the time we correctly reject the null at this sample/effect size.

The above is great, but we never know when the idealized sampling distribution isn't normal. We can be committing Type I and II errors far more/less often than we expect if this assumption is not met. To counteract this, we can rely on more flexible and robust methods of inference: *resampling methods* that create *empirical sampling distributions*.

A quick example involving bootstrapping sample2 from earlier is below. (The histogram has the endpoints of the confidence interval overlaid on it.) We'll talk about this more next time!

```
# normal theory confidence interval for sample2
confint(lm(scale(sample2$taylor)~scale(sample2$vidgames)), level=.95)
```

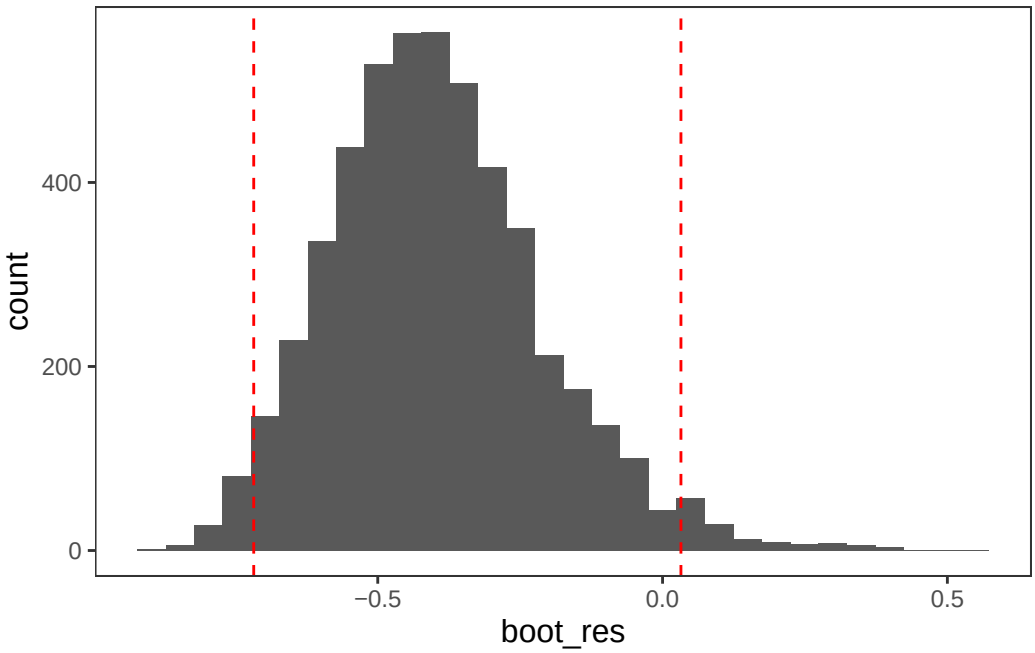
```
                              2.5 %      97.5 %
(Intercept)             -0.3477219  0.34772193
scale(sample2$vidgames) -0.7602788 -0.05294615
```

```
# bootstrap
bs_samp = 5000
boot_res <- rep(NA,bs_samp)
for(i in 1:bs_samp){
  boot_samp <- sample2[sample(1:nrow(sample2), nrow(sample2), replace=TRUE),]
  boot_res[i] <- cor(boot_samp$taylor,boot_samp$vidgames)
}

# find the upper and lower endpoints of the confidence interval
bs_ci <- quantile(boot_res, c(.025, .975))
bs_ci
```

```
      2.5%      97.5%
-0.71771474  0.03228677
```

```
# prepare data for visualization in ggplot
boot_res_df = data.frame(boot_res = boot_res)
ggplot(boot_res_df, aes(x=boot_res)) +
  geom_histogram() +
  jtools::theme_apa() +
  geom_vline(xintercept=bs_ci[1], color = "red",linetype="dashed") +
  geom_vline(xintercept=bs_ci[2], color = "red",linetype="dashed")
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Comparing the two, we can see that the bootstrap confidence interval is more narrow, which likely leads to greater power.

That's all for now. These concepts are extremely hard to understand so it's okay if you're confused. Mere exposure to these ideas will benefit you in the long run and help you learn it faster the next time.

End of script