Sampling Distributions Lecture

Dr. J (PSYC417)

This is a markdown file aligning 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 those 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.)

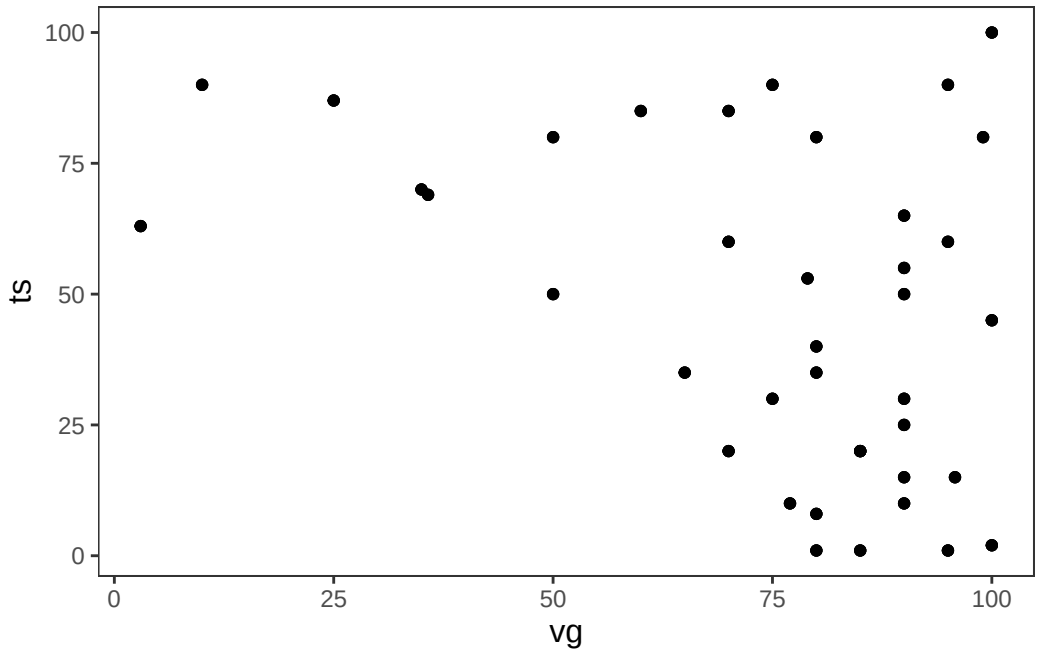
Forming an idealized sampling distribution

To do this, we first need to load relevant package and read in the data. 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.csv("vg_ts.csv")

tsvg <- rbind(tsvg, tsvg, tsvg)

# summarize the data
summary(tsvg)
```

```
      ts               vg
 Min.   :  1.00   Min.   :  3.00
 1st Qu.: 20.00   1st Qu.: 70.00
 Median : 50.00   Median : 80.00
 Mean   : 46.79   Mean   : 74.73
 3rd Qu.: 80.00   3rd Qu.: 90.00
 Max.   :100.00   Max.   :100.00
```

```
# visualize the data
ggplot(tsvg, aes(x = vg, y = ts)) +
  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$ts, tsvg$vg)
```

```
    Pearson's product-moment correlation

data:  tsvg$ts and tsvg$vg
t = -4.5781, df = 115, p-value = 1.195e-05
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.5359624 -0.2272982
sample estimates:
       cor
-0.3926304
```

```
# obtain the test statistic
pop_value = cor.test(tsvg$ts, tsvg$vg)[4]$estimate; pop_value
```

```
       cor
```

```
-0.3926304
```

It looks like they are negatively associated (r = -0.393), 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 9 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[sample(c(1:nrow(tsvg)),30),]
cor.test(sample1$ts, sample1$vg)
```

```
    Pearson's product-moment correlation

data:  sample1$ts and sample1$vg
t = -1.724, df = 28, p-value = 0.09573
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.60277567  0.05683368
sample estimates:
       cor
-0.3097784
```

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 9?

```
sample2 <- tsvg[sample(c(1:nrow(tsvg)),30),]
cor.test(sample2$ts, sample2$vg)
```

```
    Pearson's product-moment correlation

data:  sample2$ts and sample2$vg
t = -3.9269, df = 28, p-value = 0.000511
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.7871987 -0.3001026
sample estimates:
       cor
-0.5959401
```

And another?

```
sample3 <- tsvg[sample(c(1:nrow(tsvg)),30),]
cor.test(sample3$ts, sample3$vg)
```

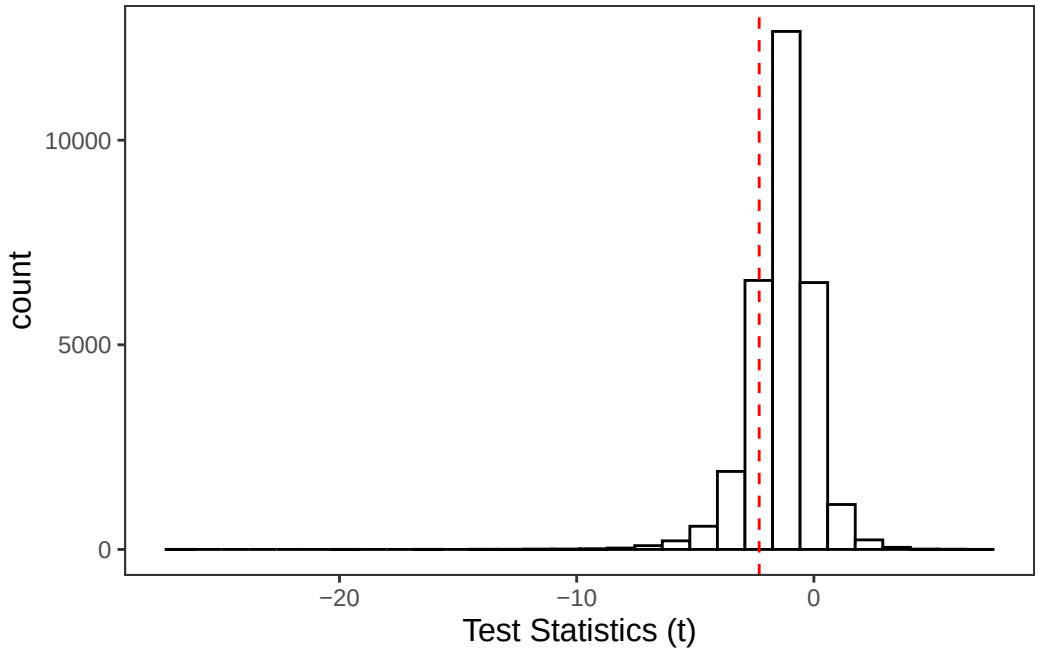
```
    Pearson's product-moment correlation

data:  sample3$ts and sample3$vg
t = -2.5017, df = 28, p-value = 0.01848
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.68257999 -0.07937109
sample estimates:
       cor
-0.4274183
```

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, 9). We're going to create a *for* loop for 30,000 imaginary studies where we randomly sample 9 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[sample(c(1:nrow(tsvg)),9),]
  ideal_res[i] <- cor.test(sample$ts, sample$vg)[1]$statistic
  cor_res[i] <- cor(sample$ts, sample$vg)
}

# prepare and plot in ggplot
ideal_df = data.frame(ideal_res = ideal_res, cor_res = cor_res)
ggplot(ideal_df, aes(x=ideal_res)) +
```

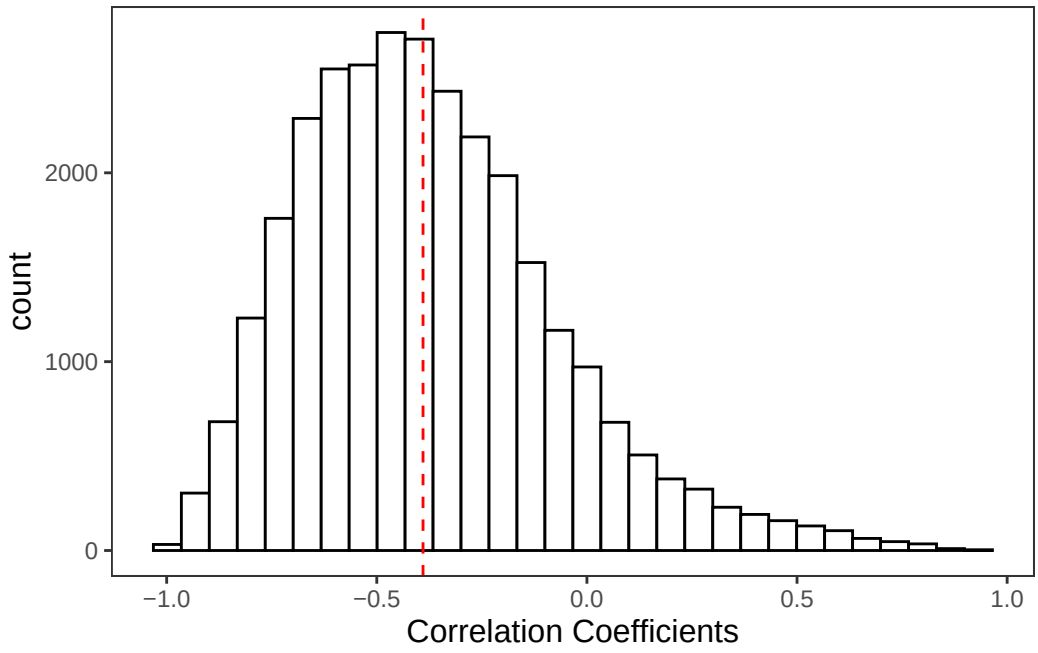
```
    geom_histogram(fill = "white", color = "black") +
    jtools::theme_apa() +
    xlab("Test Statistics (t)") +
    geom_vline(xintercept=-2.3065, color = "red",linetype="dashed")
```

```
`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(fill = "white", color = "black") +
    jtools::theme_apa() +
    xlab("Correlation Coefficients") +
    geom_vline(xintercept=-.39, color = "red",linetype="dashed")
```

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



That's quite the range/distribution of test statistics and correlation coefficients, even in this example where our sample is a large proportion 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 **not** normal. However, 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
tstat = qt(.025, 7)

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

```
[1] 0.1731667
```

This isn't relevant to our current discussion, but it's interesting that only ~15% 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$ts)~scale(sample2$vg)), level=.95)
```

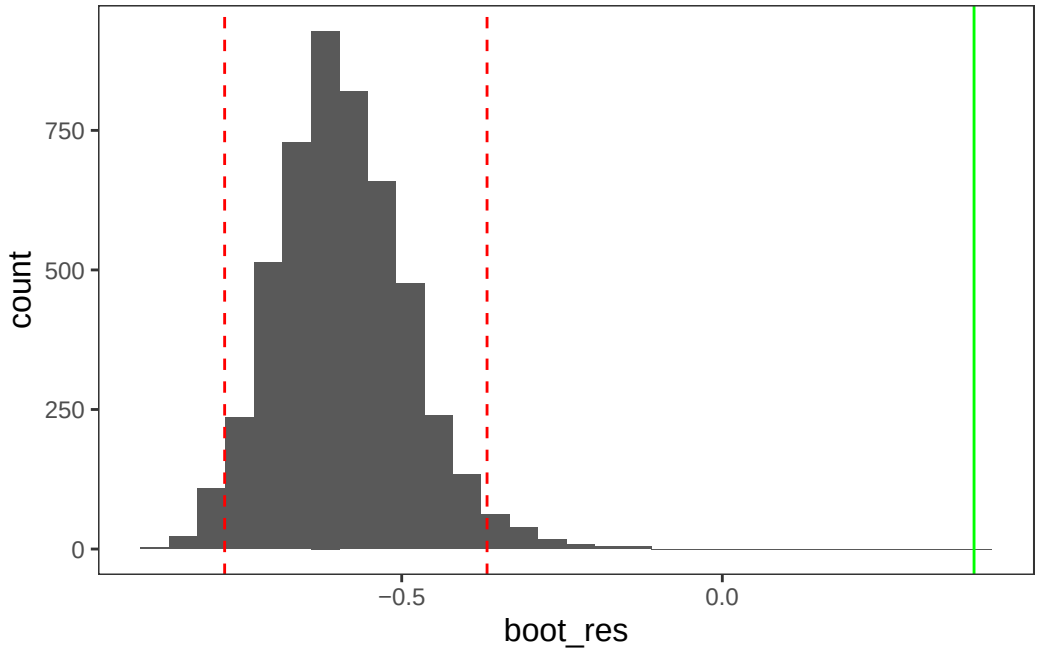
```
                       2.5 %     97.5 %
(Intercept)       -0.3056376  0.3056376
scale(sample2$vg) -0.9068027 -0.2850775
```

```
# 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$ts,boot_samp$vg)
}

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

```
     2.5%      97.5%
-0.7761393 -0.3668844
```

```
# 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") +
  geom_vline(xintercept=-pop_value, color = "green")
```

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



Now let's take a look at coverage across repeated samples. Recall the interpretation of confidence intervals is that "95% of infinitely many confidence intervals constructed the same way with the same sample size will contain the true population value." For a 95% confidence interval, we expect this rate of "coverage" to be 95%.

```
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
bs_samp = 500
boot_res <- rep(NA,bs_samp)
tot_rep = 500
coverage = rep(NA, tot_rep)

for(j in 1:tot_rep){
  for(i in 1:bs_samp){
    boot_samp <- sample2[sample(1:nrow(sample2), nrow(sample2), replace=TRUE),]
    boot_res[i] <- cor(boot_samp$ts,boot_samp$vg)
  }
  bs_ci <- quantile(boot_res, c(.025, .975))

  coverage[j] = dplyr::between(pop_value, bs_ci[1], bs_ci[2])
}

paste0("The population value of ", round(pop_value,3), " was covered in ", round(sum(cover
```

```
[1] "The population value of -0.393 was covered in 93% of samples. We expect this to be 95%.'
```

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