Univariate Descriptives and Uncertainty

Will Doyle

9/16/2021

Uncertainty in Univariate Statistics

When we calculate a summary statistic in univariate statistics, we're making a statement about what we can expect to see in other situations. If I say that the average height of a cedar tree is 75 feet, that gives an expectation for the average height we might calculate for any given sample of cedar trees. However, there's more information that we need to communicate. It's not just the summary measure– it's also our level of uncertainty around that summary measure. Sure, the average height might be 75 feet, but does that mean in every sample we ever collect we're always going to see an average of 75 feet?

Motivation for Today: How much do turnovers matter?

We're going to work with a different dataset covering every NBA game played in the seasons 2016-17 to 2018-19. I'm interested in whether winning teams have higher or lower values of turnovers, and whether winning teams tend to more often make over 80 percent of their free throws.

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(tidymodels)
```

```
## -- Attaching packages -------------------------------------- tidymodels 0.1.2 --

## v broom     0.7.6      v recipes   0.1.15
## v dials     0.0.9      v rsample   0.0.8
## v infer     0.5.3      v tune      0.1.2
## v modeldata 0.1.0      v workflows 0.2.1
## v parsnip   0.1.4      v yardstick 0.0.8

## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x scales::discard() masks purrr::discard()
## x dplyr::filter()   masks stats::filter()
## x recipes::fixed()  masks stringr::fixed()
## x dplyr::lag()      masks stats::lag()
## x yardstick::spec() masks readr::spec()
## x recipes::step()   masks stats::step()
```

The Data

```
gms<-read_rds("game_summary.Rds")
gms
```

```
## # A tibble: 7,380 x 15
##     idGame yearSeason dateGame   idTeam nameTeam  locationGame   tov   pts  treb
##      <dbl>      <int> <date>      <dbl> <chr>     <chr>        <dbl> <dbl> <dbl>
##  1  2.16e7       2017 2016-10-25 1.61e9 Clevelan~ H               14   117    51
##  2  2.16e7       2017 2016-10-25 1.61e9 New York~ A               18    88    42
##  3  2.16e7       2017 2016-10-25 1.61e9 Portland~ H               12   113    34
##  4  2.16e7       2017 2016-10-25 1.61e9 Utah Jazz A               11   104    31
##  5  2.16e7       2017 2016-10-25 1.61e9 Golden S~ H               16   100    35
##  6  2.16e7       2017 2016-10-25 1.61e9 San Anto~ A               13   129    55
##  7  2.16e7       2017 2016-10-26 1.61e9 Miami He~ A               10   108    52
##  8  2.16e7       2017 2016-10-26 1.61e9 Orlando ~ H               11    96    45
##  9  2.16e7       2017 2016-10-26 1.61e9 Dallas M~ A               15   121    49
## 10  2.16e7       2017 2016-10-26 1.61e9 Indiana ~ H               16   130    52
## # ... with 7,370 more rows, and 6 more variables: pctFG <dbl>, teamrest <dbl>,
## #   second_game <lgl>, pctFT <dbl>, isWin <lgl>, ft_80 <dbl>
```

The data for today is game by team summary data. Codebook here. It includes information for each team for every game played from 2017 to 2019. We're interested in knowing about how turnovers `tov` are different between game winners `isWin`.

Continuous Variables: Point Estimates

```
gms%>%
  filter(yearSeason==2017)%>%
  group_by(isWin)%>%
  summarize(mean(tov))
```

```
## # A tibble: 2 x 2
##   isWin `mean(tov)`
##   <lgl>       <dbl>
## 1 FALSE        13.8
## 2 TRUE         12.9
```

It looks like there's a fairly substantial difference– winning teams turned the ball over an average of 12.9 times, while losing teams turned it over an average of 13.8 times. One way to summarize this is that winning teams in general had one less turnover per game than losing teams.

What if we take these results and decide that these will apply in other seasons? We could say something like: "Winning teams over the course of a season will turn the ball over 12.9 times, and losing teams 13.8 times, period." Well let's look and see:

```
gms%>%
  filter(yearSeason==2018)%>%
  group_by(isWin)%>%
  summarize(mean(tov))
```

```
## # A tibble: 2 x 2
##   isWin `mean(tov)`
##   <lgl>       <dbl>
## 1 FALSE        14.1
## 2 TRUE         13.3
```

```
gms%>%
  filter(yearSeason==2019)%>%
  group_by(isWin)%>%
  summarize(mean(tov))
```

```
## # A tibble: 2 x 2
##   isWin `mean(tov)`
##   <lgl>       <dbl>
## 1 FALSE        13.9
## 2 TRUE         13.1
```

So, no, that's not right. In other seasons winning teams turned the ball over less, but it's not as simple as just saying it will always be the two numbers we calculated from the 2017 data.

What we'd like to be able to do is make a more general statement, not just about a given season but about what we can expect in general. To do that we need to provide some kind of range of uncertainty: what range of turnovers can we expect to see from both winning and losing teams? To generate this range of uncertainty we're going to use some key insights from probability theory and statistics that help us generate estimates of uncertainty.

Quick exercise: Are winning teams in 2017 more likely to make more than 80 percent of their free throws?

Sampling

We're going to start by building up a range of uncertainty from the data we already have. We'll do this by sampling from the data itself.

Let's just take very small sample of games– 100 games– and calculate turnovers for winners and losers.

```
set.seed(210916)

sample_size<-100

gms%>%
  filter(yearSeason==2017)%>% ## Filter to just 2017
  sample_n(size=sample_size) %>% ## Sample size is as set above
  group_by(isWin)%>% ## Group by win/lose
  summarize(mean(tov)) ## calculate mean
```

```
## # A tibble: 2 x 2
##   isWin `mean(tov)`
##   <lgl>       <dbl>
## 1 FALSE        14.8
## 2 TRUE         12.7
```

And again:

```
gms%>%
  filter(yearSeason==2017)%>% ## Filter to just 2017
  sample_n(size=sample_size) %>% ## Sample size is as set above
  group_by(isWin)%>% ## Group by win/lose
  summarize(mean(tov)) ## calculate mean
```

```
## # A tibble: 2 x 2
##   isWin `mean(tov)`
##   <lgl>       <dbl>
## 1 FALSE        14.4
## 2 TRUE          13.1
```

Sometimes we can get samples where the winning team turned the ball over more! These resamples on their own don't appear to be particularly useful, but what would happen if we calculated a bunch (technical term) of them?

I can continue this process of sampling and generating values many times using a loop. The code below resamples from the data 10,000 times, each time calculating the mean turnovers for winners and losers in a sample of size 100. It then adds those two means to a growing list, using the bind_rows function.

Warning: the code below will take a little while to run

```
gms_tov_rs<-NULL ##  Create a NULL variable: will fill this in later

for (i in 1:10000){ # Repeat the steps below 10,000 times
  gms_tov_rs<-gms%>% ## Create a dataset called gms_tov_rs (rs=resampled)
  filter(yearSeason==2017)%>%  ## Just 2017
  sample_n(size=sample_size) %>% ## Sample 100 games
  group_by(isWin)%>% ## Group by won or lost
  summarize(mean_tov=mean(tov))%>% ## Calculate mean turnovers for winners and losers
    bind_rows(gms_tov_rs) ## add this result to the existing dataset
}
```

Now I have a dataset that is built up from a bunch of small resamples from the data, with average turnovers for winners and losers in each small sample. Let's see what these look like.

```
gms_tov_rs
```

```
## # A tibble: 20,000 x 2
##    isWin mean_tov
##    <lgl>    <dbl>
##  1 FALSE     13.4
##  2 TRUE      12.7
##  3 FALSE     13.6
##  4 TRUE      11.8
##  5 FALSE     14
##  6 TRUE      12.8
##  7 FALSE     12.9
##  8 TRUE      13.2
##  9 FALSE     13.3
## 10 TRUE      12.3
## # ... with 19,990 more rows
```

This is a dataset that's just a bunch of means. We can calculate the mean of all of these means and see what it looks like:

```
gms_tov_rs%>%
  group_by(isWin)%>%
  summarise(mean_of_means=mean(mean_tov))
```

```
## # A tibble: 2 x 2
##   isWin mean_of_means
##   <lgl>         <dbl>
## 1 FALSE          13.8
## 2 TRUE           12.9
```

How does this "mean of means" compare with the actual?

```
gms%>%
  filter(yearSeason==2017)%>%
  group_by(isWin)%>%
  summarize(mean(tov))
```

```
## # A tibble: 2 x 2
##   isWin `mean(tov)`
##   <lgl>       <dbl>
## 1 FALSE        13.8
## 2 TRUE         12.9
```

Pretty similar! It's what we would expect, really, but it's super important. If we repeatedly sample from a dataset, our summary measures of a sufficiently large number of repeated samples will converge on the true value of the measure from the dataset.

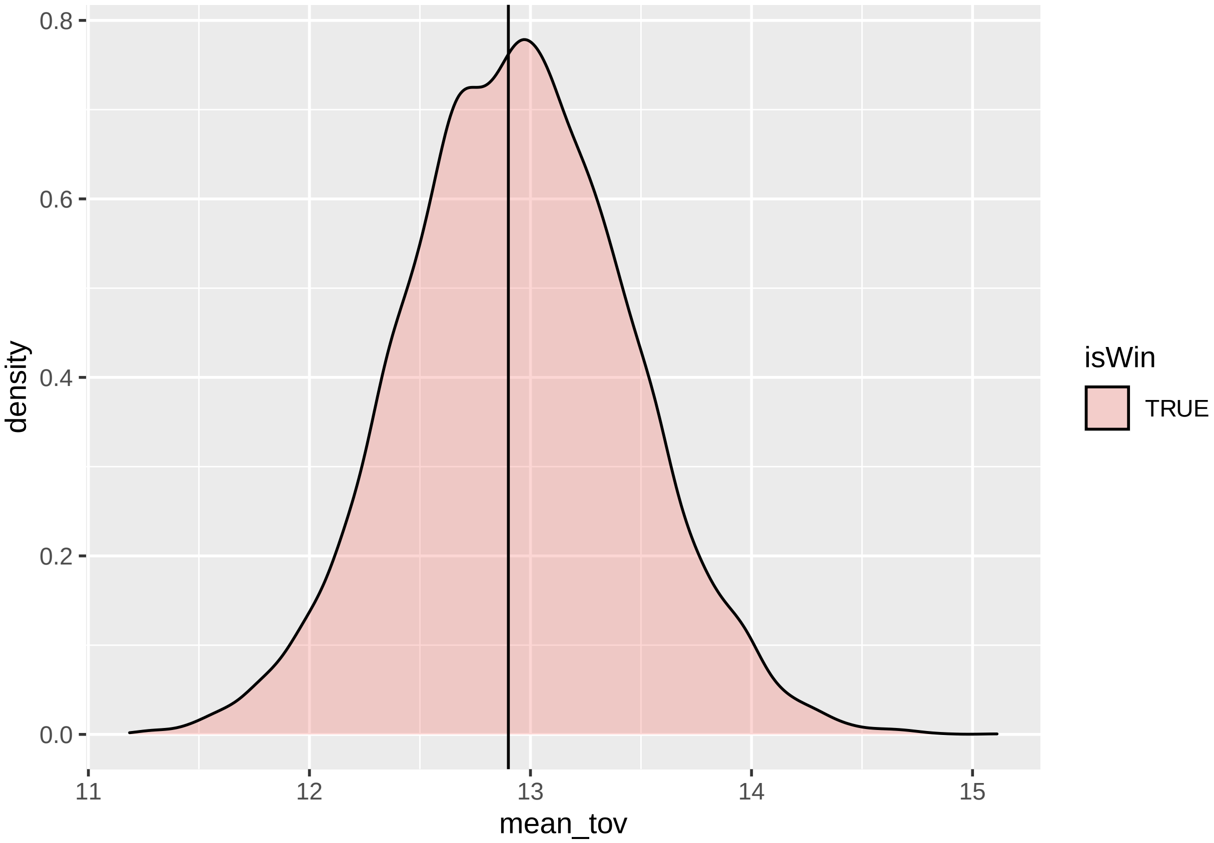
Quick Exercise: Repeat the above, but do it for Pct of Free Throws above .8

Distribution of Resampled Means

That's fine, but the other thing is that the *distribution* of those repeated samples will tell us about what we can expect to see in other, out of sample data that's generated by the same process.

Let's take a look at the distribution of turnovers for game winners:

```
gms_tov_rs%>%
  filter(isWin)%>%
  ggplot(aes(x=mean_tov,fill=isWin))+
  geom_density(alpha=.3)+
  geom_vline(xintercept =12.9)
```



We can see that the mean of this distribution is centered right on the mean of the actual data, and it goes from about 11 to about 15. This is different than the minimum and maximum of the overall sample, which goes from 3 to 24.

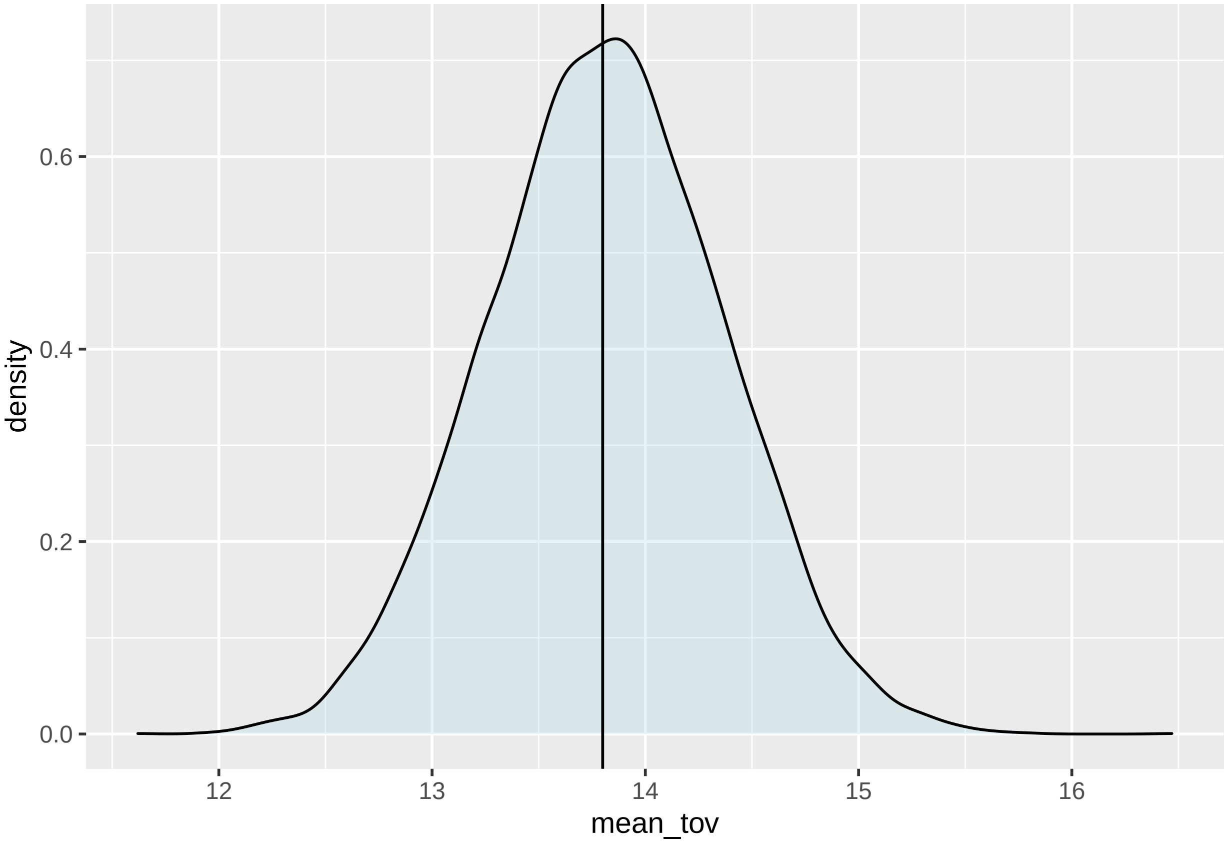
```
gms_tov_rs%>%
  filter(isWin)%>%
  summarize(value=fivenum(mean_tov))%>% ## Five number summary: described below
  mutate(measure=c("Min","25th percentile","Median","75th percentile","Max"))%>%
  select(measure, value)
```

```
## # A tibble: 5 x 2
##   measure         value
##   <chr>           <dbl>
## 1 Min              11.2
## 2 25th percentile  12.6
## 3 Median           12.9
## 4 75th percentile  13.3
## 5 Max              15.1
```

So what this tells us is that the minimum turnovers for winners in all of the samples we drew was 11.2, the maximum was about 15 and the median was 12.9.

And for game losers:

```
gms_tov_rs%>%
  filter(!isWin)%>%
  ggplot(aes(x=mean_tov,fill=isWin))+
  geom_density(alpha=.3,fill="lightblue")+
  geom_vline(xintercept =13.8)
```



```
gms_tov_rs%>%
  filter(!isWin)%>%
  summarize(value=fivenum(mean_tov))%>%
    mutate(measure=c("Min","25th percentile","Median","75th percentile","Max"))%>%
  select(measure, value)
```

```
## # A tibble: 5 x 2
##   measure         value
##   <chr>           <dbl>
## 1 Min              11.6
## 2 25th percentile  13.4
## 3 Median           13.8
## 4 75th percentile  14.2
## 5 Max              16.5
```

For game losers, minimum turnovers for winners in all of the samples we drew was 11.6, the maximum was about 16.5 (!!) and the median was 13.8.

Quick Exercise: Calculate the same summary, but do it for Pct of Free Throws above .8

So What? Using Percentiles of the Resampled Distribution

Now we can make some statements about uncertainty. Based on this what we can say is that in other seasons, we would expect that turnover for game winners will be in a certain range, and the same for game losers. What range? Well it depends on the level of risk you're willing to take as an analyst. Academics (a cautious bunch to be sure) usually use the middle 95 percent of the distribution: So for game winners:

```
gms_tov_rs%>%
  filter(isWin)%>%
  summarize(pct_05=quantile(mean_tov,.025),
            pct_95=quantile(mean_tov,.975))
```

```
## # A tibble: 1 x 2
##   pct_05 pct_95
##    <dbl>  <dbl>
## 1   11.9   14.0
```

This tells us we can expect that game winners in future seasons will turn the ball over between about 12 and 14 times.

And for game losers

```
gms_tov_rs%>%
  filter(!isWin)%>%
  summarize(pct_05=quantile(mean_tov,.025),
            pct_95=quantile(mean_tov,.975))
```

```
## # A tibble: 1 x 2
##   pct_05 pct_95
##    <dbl>  <dbl>
## 1   12.7   14.9
```

This tells us that we can expect that game losers in future seasons will turn the ball over between … 12.7 and 14.9 times.

Don't be disappointed! It just turns out that if we want to make accurate statements about out of sample data, we need to reflect our uncertainty.

Let's check to see if our expectations are borne out in future seasons:

```
gms%>%
  filter(yearSeason==2018)%>%
  group_by(isWin)%>%
  summarize(mean(tov))
```

```
## # A tibble: 2 x 2
##   isWin `mean(tov)`
##   <lgl>       <dbl>
## 1 FALSE        14.1
## 2 TRUE         13.3
```

```
gms%>%
  filter(yearSeason==2019)%>%
  group_by(isWin)%>%
  summarize(mean(tov))
```

```
## # A tibble: 2 x 2
##   isWin `mean(tov)`
##   <lgl>       <dbl>
## 1 FALSE        13.9
## 2 TRUE         13.1
```

So, our intervals for both winners and losers did include the values in future seasons.

Other intervals– the tradeoff between a "precise" interval and risk

You may be underwhelmed at this point, because the 95 percent range is a big range of possible turnover values. We can use narrower intervals– it just raises the risk of being wrong. Let's try the middle 50 percent.

```
gms_tov_rs%>%
  group_by(isWin)%>%
  summarize(pct_25=quantile(mean_tov,.25),
            pct_75=quantile(mean_tov,.75))
```

```
## # A tibble: 2 x 3
##   isWin pct_25 pct_75
##   <lgl>  <dbl>  <dbl>
## 1 FALSE   13.4   14.2
## 2 TRUE    12.6   13.3
```

Okay, now we're saying that winners will have between 12.6 and 13.3 turnovers. Is that right?

```
gms%>%
  filter(yearSeason==2018)%>%
  group_by(isWin)%>%
  summarize(mean(tov))
```

```
## # A tibble: 2 x 2
##   isWin `mean(tov)`
##   <lgl>       <dbl>
## 1 FALSE        14.1
## 2 TRUE         13.3
```

```
gms%>%
  filter(yearSeason==2019)%>%
  group_by(isWin)%>%
  summarize(mean(tov))
```

```
## # A tibble: 2 x 2
##   isWin `mean(tov)`
##   <lgl>       <dbl>
## 1 FALSE        13.9
## 2 TRUE         13.1
```

Yes, this checks out for subsequent seasons. What about a really narrow interval– the middle 10 percent?

```
gms_tov_rs%>%
  group_by(isWin)%>%
  summarize(pct_45=quantile(mean_tov,.45),
            pct_55=quantile(mean_tov,.55))
```

```
## # A tibble: 2 x 3
##   isWin pct_45 pct_55
##   <lgl>  <dbl>  <dbl>
## 1 FALSE   13.7   13.9
## 2 TRUE    12.9   13
```

```
gms%>%
  filter(yearSeason==2018)%>%
  group_by(isWin)%>%
  summarize(mean(tov))
```

```
## # A tibble: 2 x 2
##   isWin `mean(tov)`
##   <lgl>       <dbl>
## 1 FALSE        14.1
## 2 TRUE         13.3
```

In 2018, winning teams turned the ball over 13.3 times, on average. That's above the range we gave! If we used a 10 percent interval we'd be wrong. Similarly, in 2018 losing teams turned the ball over 14.1 times, again below our interval.

```
gms%>%
  filter(yearSeason==2019)%>%
  group_by(isWin)%>%
  summarize(mean(tov))
```

```
## # A tibble: 2 x 2
##   isWin `mean(tov)`
##   <lgl>       <dbl>
## 1 FALSE        13.9
## 2 TRUE         13.1
```

In 2019, winning teams turned the ball over 13.1 times, on average. That's below the range we gave! If we used a 10 percent interval we'd be wrong, again.

It turns out that the way this method works is that for an interval of a certain range, the calculated interval will include the true value of the measure in the same percent *of repeated samples*. We can think of each season as a repeated sample, so the middle 95 percent of this range will include the true value in 95 percent of seasons. When we call this a confidence interval, we're saying we have confidence in the approach, not the particular values we calculated.

The tradeoff here is between providing a narrow range of values vs. the probability of being correct. We can give a very narrow interval for what we would expect to see in out of sample data, but we're going to be wrong– a lot. We can give a very wide interval, but the information isn't going to be useful to decisionmakers. This is one of the key tradeoffs in applied data analysis, and there's no single answer to the question: what interval should I use? Academic work has settled on the 95 percent interval, but there's no real theoretical justification for this.

Empirical Bootstrap

What we just did is called the empirical bootstrap. It's massively useful, because it can be applied for any summary measure of the data: median, percentiles, and measures like regression coefficients. Here is the summary of steps for the empirical bootstrap:

- Decide on the summary measure to be used for the variable (it doesn't have to be the mean)
- Calculate the summary measure on a small subsample (called the bootstrap sample) of the data
- Repeat step 2 many times (how many? Start with 1000, but more is better.) Compile the estimates.
- Calculate the percentiles of the bootstrap distribution from the previous step.
- Describe your uncertainty using those percentiles.

Quick Exercise: Does 50 percent interval for free throws percent above 80 include the values for subsequent seasons?

Calculating Bootstraps Using Rsample

We can undertake the steps above using R's built-in capabilities. Below I create a dataset that's structured for bootstrap resampling:

```
boot_2017<-bootstraps(gms%>%filter(yearSeason==2017),times = 10000)
```

This is what's called a "splits" data structure. It splits the data into two parts: one part will be used in the analysis, one part will be held out. For reasons that escape me, the command only allows the data to be split 90/10, with 90 percent held out for analysis and 10 percent for assessment.

The function below takes the data (in split format), samples each element down to the specified sample size (100 in our case) and then pulls the turnover variable `tov`. It then returns a dataset that includes just the mean of the specified variable, in this case `tov`.

```
sample_size=100

calc_tov_mean_winners <- function(split){
  dat <- assessment(split) %>% ## create an object called dat from each "split" of the data
    filter(isWin)%>% ## filter just for winners
    sample_n(size=sample_size)%>% ## Sample the split down to 100
    pull(tov) ## pull just turnovers

  # Put it in this tidy format to use int_pctl
  return(tibble( ## return a tibble
    term = "mean", ## the variable will be named mean
    estimate = mean(dat))) ## the estimate is the mean of dat from above
}

calc_tov_mean_losers <- function(split){
  dat <- assessment(split) %>% ## create an object called dat from each "split" of the data
    filter(!isWin)%>% ## filter just for losers
    sample_n(size=sample_size)%>% ## Sample the split down to 100
    pull(tov) ## pull just turnovers

  # Put it in this tidy format to use int_pctl
  return(tibble( ## return a tibble
    term = "mean", ## the variable will be named mean
    estimate = mean(dat))) ## the estimate is the mean of dat from above
}
```

```
results_winners<-boot_2017%>% ## start with the resampled dataset
  mutate(tov_mean= ## mutate to create a column called tov_mean
           map(splits,calc_tov_mean_winners))  ## map the "calc" function onto each split

results_winners%>%int_pctl(tov_mean)
```

```
## # A tibble: 1 x 6
##   term  .lower .estimate .upper .alpha .method
##   <chr>  <dbl>     <dbl>  <dbl>  <dbl> <chr>
## 1 mean    12.2      12.9   13.6   0.05 percentile
```

```
results_losers<-boot_2017%>% ## start with the resampled dataset
  mutate(tov_mean= ## mutate to create a column called tov_mean
           map(splits,calc_tov_mean_losers))  ## map the "calc" function onto each split

results_losers%>%int_pctl(tov_mean)
```

```
## # A tibble: 1 x 6
##   term  .lower .estimate .upper .alpha .method
##   <chr>  <dbl>     <dbl>  <dbl>  <dbl> <chr>
## 1 mean    13.1      13.8   14.6   0.05 percentile
```