Lecture 2

Hisam Sabouni, PhD

February 2020

Overview

In this lecture we will lay out the core ideas behind estimators. We will focus on analyzing their finite and asymptotic sample properties. The key ideas to take away relate to unbiasdenss, efficiency, and consistency. From these results we will discuss the law of large numbers and central limit theorem.

Again Heavy citation notice from Wooldridge!

Statistical inference involves learning something about a population given the availability of a sample from that population. By population, we mean any well-defined group of subjects, which could be individuals, firms, cities, or many other possibilities. By 'learning,' we can mean several things, which are broadly divided into the categories of estimation and hypothesis testing.

Let Y be a random variable representing a population with a probability density function $f(y;\theta)$, which depends on the single parameter θ. The probability density function (pdf) of Y is assumed to be known except for the value of θ ; different values of θ imply different population distributions, and therefore we are interested in the value of θ. If we can obtain certain kinds of samples from the population, then we can learn something about θ. The easiest sampling scheme to deal with is random sampling.

Given a random sample $\{Y_1, Y_2, \ldots, Y_n\}$ drawn from a population distribution that depends on an unknown parameter θ, an estimator of θ is a rule that assigns each possible outcome of the sample a value of θ. The rule is specified before any sampling is carried out; in particular, the rule is the same regardless of the data actually obtained.

What's an example of an estimator?

In general an estimator takes on the form:

$$W = h(\{Y_1, Y_2, \ldots, Y_n\})$$

Here unknown function $h(\cdot)$ is our estimator that takes in the random sample. Given each random sample the function will return a different result (as the input changes the output will change). For evaluating estimation procedures, we study various properties of the probability distribution of the random variable W. The distribution of an estimator is often called its sampling distribution, because this distribution describes the likelihood of various outcomes of W across different random samples. Because there are unlimited rules for combining data to estimate parameters, we need some sensible criteria for choosing among estimators, or at least for eliminating some estimators from consideration.

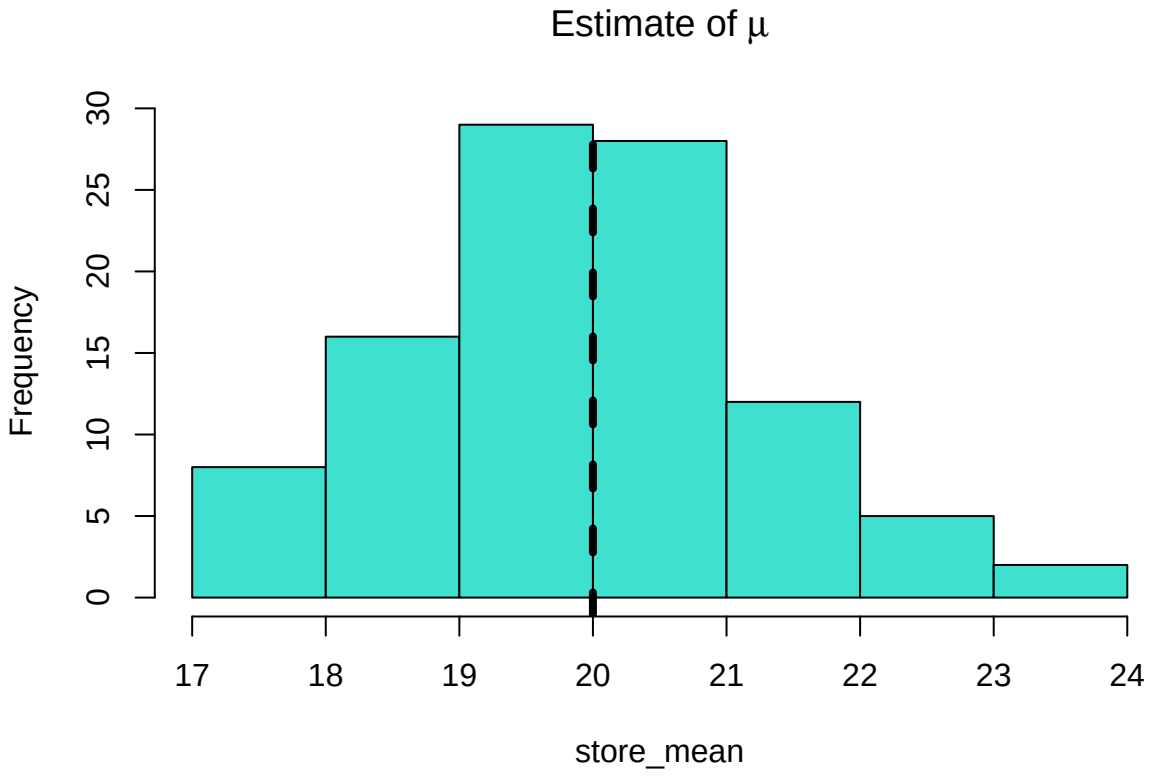
Unbiasedness: An estimator W, of θ, is an unbiased estimator if $E(W) = \theta$

Bias of an Estimator: If W is a biased estimator of θ, its bias is defined as $Bias(W) = E(W) - \theta$

This states that if we could indefinitely draw random samples on Y from the population, compute an estimate each time, and then average these estimates over all random samples, we would obtain θ.

Lets see this in action with a classical estimator: the average. We will simulate draws from a normal distribution that has a population average of 20 and a population standard deviation of 5. We will take 100 draws from this sample each of 10 observations. For each of the 100 draws we will record our estimate of the sample mean.

```
#Set the seed for reproducibility
set.seed(1)
#Create an empty vector to store our estimates of the sample mean
store_mean <- c()
#lets write a loop to simulate our 100 draws
draws <- 100
for(i in 1:draws){
  #We will use the rnorm() function to simulate draws from a normal distribution
  random_sample <- rnorm(n = 10,mean = 20,sd = 5)
  #estimate and store the sample mean
  store_mean[i] <- mean(random_sample)
}
#Lets generate a histogram of our estimates of the mean for each random sample
hist(store_mean,main=expression(paste('Estimate of ',mu)),col = '#3FE0D0')
#Add a vertical line for the true estimate
abline(v = 20,lty=2,lwd = 4)
```



We can see that the average estimator is unbiased through a simple proof using the laws of the expectations operator:

$$E(\bar{Y}) = E(n^{-1}\sum_i Y_i) = n^{-1}E(\sum_i Y_i) = n^{-1}\sum_i E(Y_i) = n^{-1}\sum_i \mu = n^{-1}n\mu = \mu$$

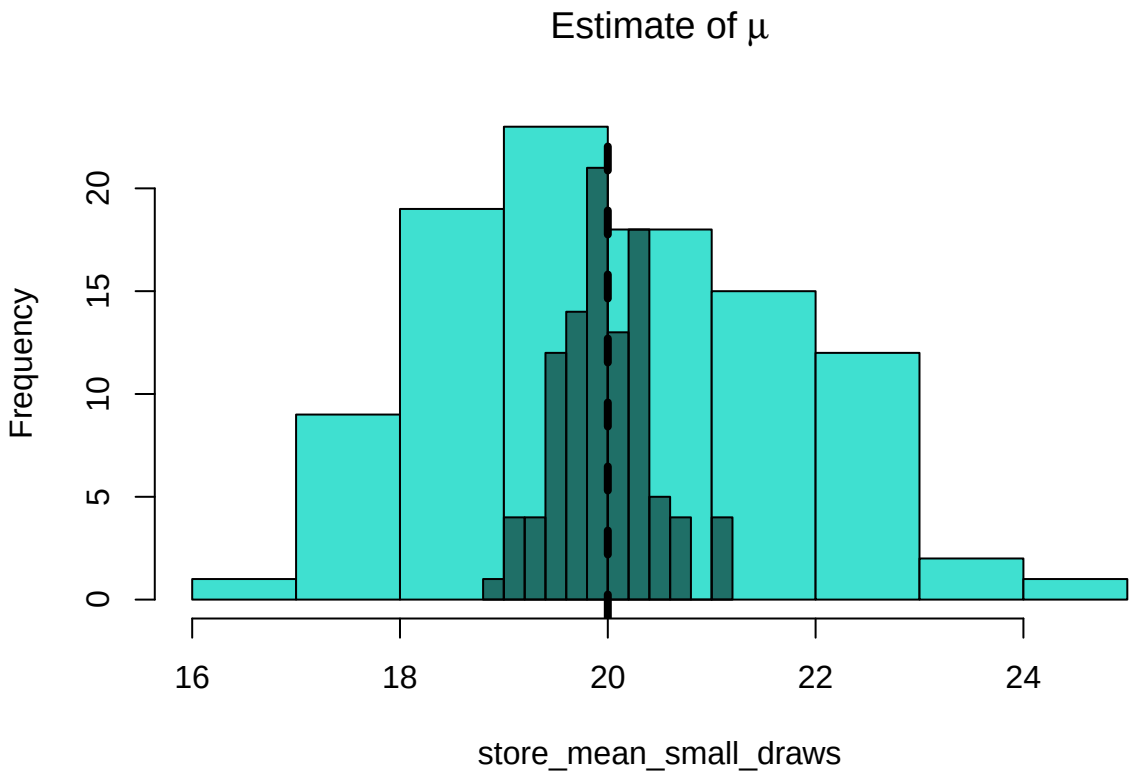
Unbiasedness only ensures that the sampling distribution of an estimator has a mean value equal to the parameter it is supposed to be estimating. This is fine, but we also need to know how spread out the distribution of an estimator is. An estimator can be equal to θ, on average, but it can also be very far away with large probability. The variance of an estimator is often called its sampling variance because it is the variance associated with a sampling distribution. Remember, the sampling variance is not a random variable; it is a constant, but it might be unknown. Lets take a look at the sampling variance of the average estimator:

$$Var(\bar{Y}) = Var(1/n\sum_i Y_i) = (1/n^2)Var(\sum_i Y_i) = (1/n^2)\sum_i Var(Y_i) = (1/n^2)\sum_i \sigma^2 = (1/n^2)n\sigma^2 = \sigma^2/n$$

An important implication of $Var(\bar{Y}) = \sigma^2/n$ is that it can be made very close to zero by increasing

the sample size n. This is a key feature of a reasonable estimator. Lets go back to our previous example an increase the sample size of our draws and empirically now estimate the sampling variance.

```
#Set the seed for reproducibility
set.seed(1)
#Create an empty vector to store our estimates of the sample mean for small draws (10)
store_mean_small_draws <- c()
#Create an empty vector to store our estimates of the sample mean for large draws(100)
store_mean_large_draws <- c()
#lets write a loop to simulate our 100 draws
draws <- 100
for(i in 1:draws){
  #We will use the rnorm() function to simulate draws from a normal distribution
  small_random_sample <- rnorm(n = 10,mean = 20,sd = 5)
  large_random_sample <- rnorm(n = 100,mean = 20,sd = 5)
  #estimate and store the sample mean
  store_mean_small_draws[i] <- mean(small_random_sample)
  store_mean_large_draws[i] <- mean(large_random_sample)
}
#Lets generate a histogram of our estimates of the mean for each random sample
hist(store_mean_small_draws,main=expression(paste('Estimate of ',mu)),col = '#3FE0D0')
hist(store_mean_large_draws,add = T,col = rgb(0,0,0,alpha = 0.5))
#Add a vertical line for the true estimate
abline(v = 20,lty=2,lwd = 4)
```



By increasing our sampling size by a factor of 10 we have nearly reduced our sampling variance by a factor of 10! This is quite an important finding for all of the empirical work you will be conducting in your research careers. You want to use estimators that are able to consistently improve in their estimates with the more data they use.

```
var(store_mean_small_draws)
```

```
## [1] 2.785808
```

```
var(store_mean_large_draws)
```

```
## [1] 0.1990663
```

In the case of having two unbiased estimators we should always go with the estimator that is more *efficient*.

> Relative Efficiency: If W_1 and W_2 are two unbiased estimators of θ, W_1 is more efficient than W_2 if $Var(W_1) \leq Var(W_2)$ for all θ.

One way to compare estimators that are not necessarily unbiased is to compute the mean squared error (MSE) of the estimators.

Again, note that the bias in our estimates is simply the difference between our estimate and the true

population parameter, here a $\hat{\cdot}$ of a variable refers to an estimate:

$$Bias(\hat{\theta}) = E(\hat{\theta}) - \theta$$

Bias tells us about the underlying accuracy of our point estimates.

The variance in our estimates is amount of uncertainty we have around our point estimates:

$$Var(\hat{\theta}) = E((\hat{\theta} - E(\hat{\theta}))^2) = E(E(\hat{\theta})^2 + \hat{\theta}^2 - 2E(\hat{\theta})\hat{\theta})$$

$$Var(\hat{\theta}) = E(\hat{\theta})^2 + E(\hat{\theta}^2) - 2E(\hat{\theta})^2 = E(\hat{\theta}^2) - E(\hat{\theta})^2$$

We can combine our two types of error by looking at the total mean squared error of our parameter estimates:

$$MSE = E((\hat{\theta} - \theta)^2) = E(\hat{\theta}^2 - 2\theta\hat{\theta} + \theta^2)$$

The true θ is a constant term. Using the linearity property of expectations:

$$\therefore \; MSE = E(\hat{\theta}^2) - 2\theta E(\hat{\theta}) + \theta^2$$

Note that the MSE can actually be decomposed into our bias and variance terms!

$$Bias^2(\hat{\theta}) = (E(\hat{\theta}) - \theta)^2 = E(\hat{\theta})^2 + \theta^2 - 2\theta E(\hat{\theta})$$

$$Var(\hat{\theta}) = E(\hat{\theta}^2) - E(\hat{\theta})^2$$

$$Bias^2(\hat{\theta}) + Var(\hat{\theta}) = E(\hat{\theta})^2 + \theta^2 - 2\theta E(\hat{\theta}) + E(\hat{\theta}^2) - E(\hat{\theta})^2$$

$$Bias^2(\hat{\theta}) + Var(\hat{\theta}) = E(\hat{\theta}^2) - 2\theta E(\hat{\theta}) + \theta^2 = MSE$$

Not bad! We can decompose our error into terms of Bias and Variance of our parameter estimates! Given this nice property of mean squared error we will use MSE throughout the class to evaluate a variety of the estimators we will cover.

Now lets get back to the idea of how our estimators behavior changes as the sample size grows. Consistency tells us how far we expect our estimator to be from the true population parameter as we increase the sample size indefinitely. Mathematically we can state this as:

> Consistency: Let W_n be an estimator of θ based on a sample $\{Y_1, Y_2, \ldots, Y_n\}$ of size n. Then W_n is a consistent estimator of θ if for every $\epsilon > 0$:

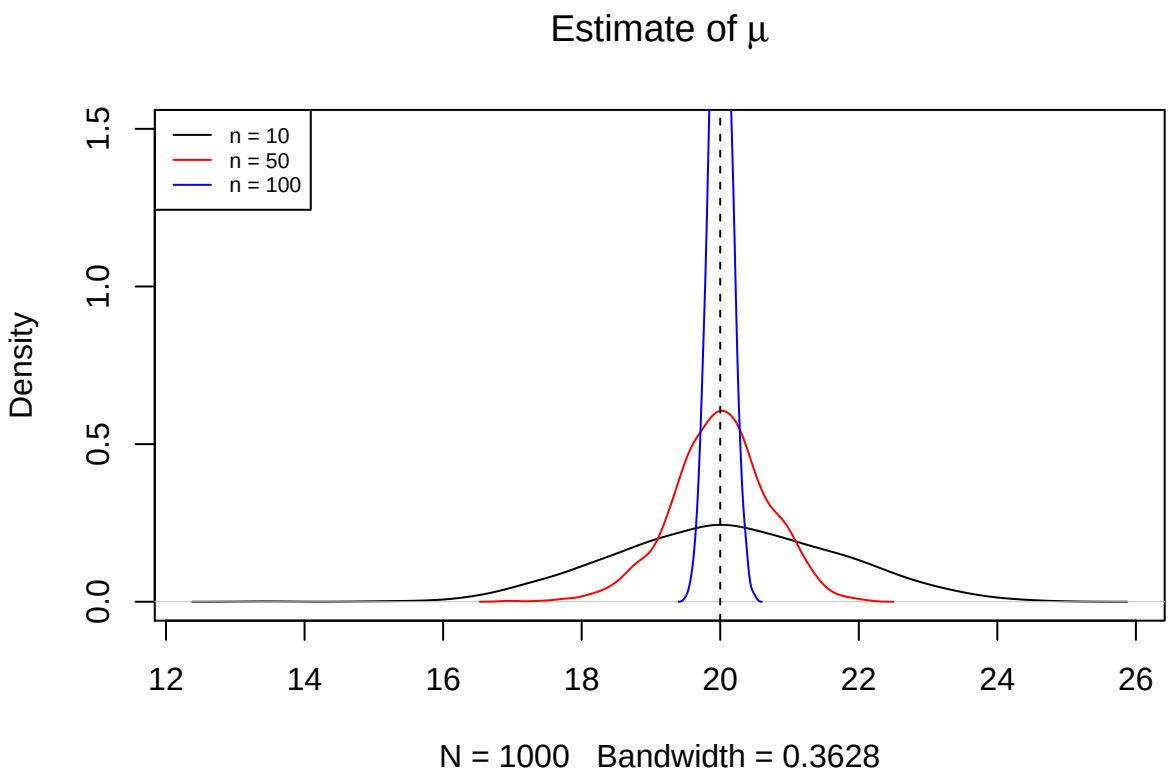
$$plim(|W_n - \theta| > \epsilon) \to 0 \text{ as } n \to \infty$$

All this is stating is that that the distribution of W_n becomes more and more concentrated about θ, which roughly means that for larger sample sizes, W_n is less and less likely to be very far from θ. If an estimator is not consistent, then it does not help us to learn about θ, even with an unlimited amount of data. For this reason, consistency is a minimal requirement of an estimator used in statistics or econometrics. Another way of stating this is: if W_n is an unbiased estimator of θ and $Var(W_n) \to 0$ as $n \to \infty$, then $plim(W_n) = \theta$

You can see this in the simulation below. As we are increasing the sample size, the probability density function is becoming more and more heavily concentrated around the true population mean of 20:

```
#Set the seed for reproducibility
set.seed(1)
#Create an empty vector to store our estimates of the sample mean for small draws (10)
store_mean_small_draws <- c()
#Create an empty vector to store our estimates of the sample mean for medium draws(50)
store_mean_medium_draws <- c()
#Create an empty vector to store our estimates of the sample mean for large draws(100)
store_mean_large_draws <- c()
#lets write a loop to simulate our 100 draws
draws <- 1000
for(i in 1:draws){
  #We will use the rnorm() function to simulate draws from a normal distribution
  small_random_sample <- rnorm(n = 10,mean = 20,sd = 5)
  medium_random_sample <- rnorm(n = 50,mean = 20,sd = 5)
```

```
  large_random_sample <- rnorm(n = 1000,mean = 20,sd = 5)
  #estimate and store the sample mean
  store_mean_small_draws[i] <- mean(small_random_sample)
  store_mean_medium_draws[i] <- mean(medium_random_sample)
  store_mean_large_draws[i] <- mean(large_random_sample)
}

plot(density(store_mean_small_draws),main=expression(paste('Estimate of ',mu)),col = 1,y
lines(density(store_mean_medium_draws),col = 2)
lines(density(store_mean_large_draws),col = 4)
abline(v = 20,lty = 2)
legend('topleft',legend = c('n = 10','n = 50','n = 100'),col = c(1,2,4),lty= 1,cex = 0.6
```



Consistency is a property of point estimators. Although it does tell us that the distribution of the estimator is collapsing around the parameter as the sample size gets large, it tells us essentially nothing about the shape of that distribution for a given sample size. Most econometric estimators have distributions that are well approximated by a normal distribution for large samples, which motivates the following definition.

Asymptotic Normality: Let $\{Z_n : n = 1, 2, \dots\}$ be a sequence of random variables, such that for

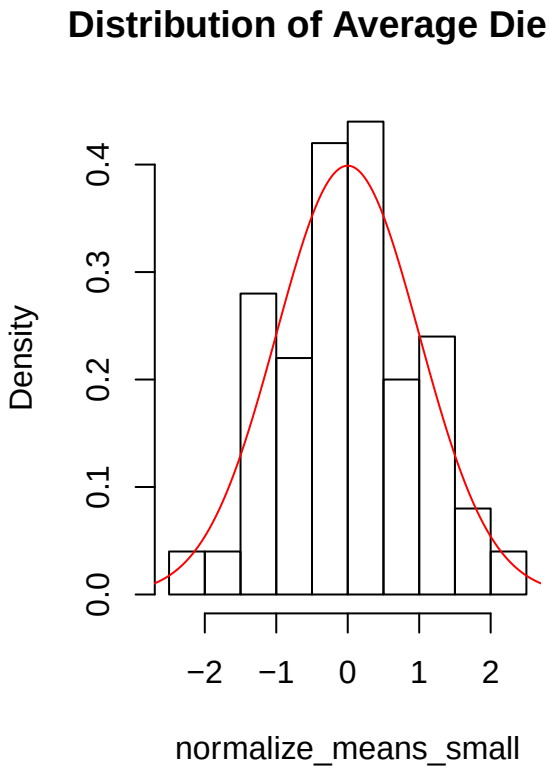
all numbers z:$P(Z_n < z) \to \Phi(z)$ as $n \to \infty$, where $\Phi(z)$ is the standard normal cumulative distribution function. This implies that the cumulative distribution function for Z_n gets closer and closer to the cdf of the standard normal distribution as the sample size n gets large. This is the foundation of the Central Limit Theorem, which states that the average from a random sample of any population, when standardized, has an asymptotic standard normal distribution.

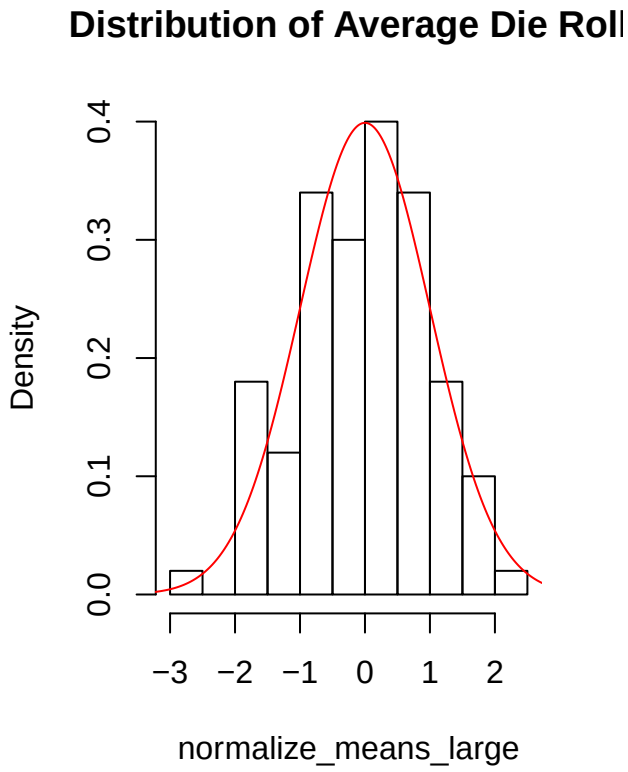
More formally, let $\{Y_1, Y_2, \ldots, Y_n\}$ be a random sample with mean μ and variance σ^2 then $Z_n = \frac{\bar{Y}-\mu}{\sigma/\sqrt{n}}$

While this seems complicated all that is being stated is that the sampling distribution of the mean of any independent, random variable will be normal or near normal, regardless of underlying distribution. Lets see this by rolling a die and analyzing the distribution of the mean outcome over a number of simulations. We know that a die has a uniform distribution. The Central Limit Theorem state that the estimate of the means of each of the rolls of the die should approximate a normal distribution.

```
#Empty vector to store average die roll
store_die_mean_small_sample <- c()
store_die_mean_large_sample <- c()
#Number of times to samples of die rolls
rolls <- 100
for(i in 1:rolls){
  #Simulate the rolls of the die for each sample
  roll_the_die_small <- sample(1:6,10,replace = T)
  roll_the_die_large <- sample(1:6,1000,replace = T)
  #Store the average
  store_die_mean_small_sample[i] <- mean(roll_the_die_small)
  store_die_mean_large_sample[i] <- mean(roll_the_die_large)
}
#Normalize the estimated means by subtracting the average estimate and dividing by the
normalize_means_small <- (store_die_mean_small_sample - mean(store_die_mean_small_sample


normalize_means_large <- (store_die_mean_large_sample - mean(store_die_mean_large_sample
par(mfrow=c(1,2))
#Generate a historgram
hist(normalize_means_small,freq = F,main='Distribution of Average Die Roll')
#Overlay a normal distribution
```

```r
lines(seq(-5,5,by = 0.01),dnorm(seq(-5,5,by = 0.01)),col = 2)

#Generate a historgram
hist(normalize_means_large,freq = F,main='Distribution of Average Die Roll')
#Overlay a normal distribution
lines(seq(-5,5,by = 0.01),dnorm(seq(-5,5,by = 0.01)),col = 2)
```






```r
par(mfrow=c(1,1))
```

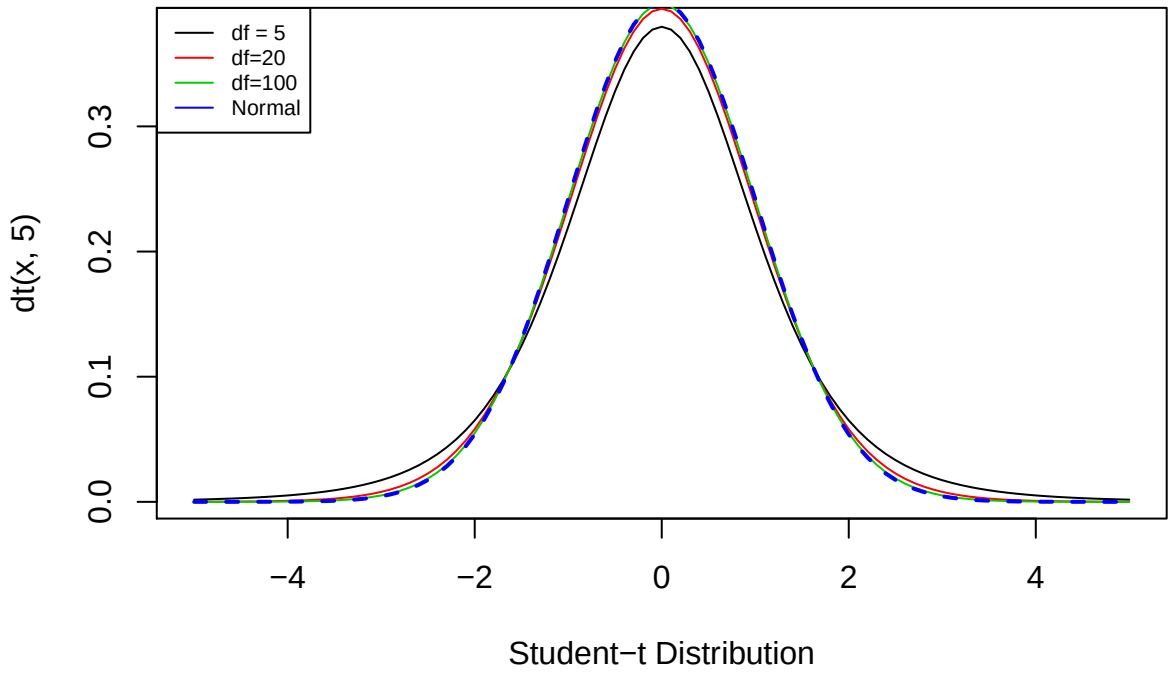
We can use the ideas of consistency along with the ideas of asymptotic normality to estimate confidence intervals around our point estimates. As we saw above each time we estimate an average from a random sample we get a slightly different estimate. When our sample size grew for each estimate we had a narrower range of estimated averages.

Using our simulation below we can estimate a 95% confidence interval around where our estimated averages fall from each random sample. A 95% confidence interval tells us where the middle 95% of the probability density falls. In other words, 95% of the samples result in a confidence interval that contains our true population parameter. In general, using the assumption of aysmptotic normality we can estimate a confidence interval for a mean estimate as follows:

$$\bar{Y} \pm \frac{Z\sigma}{\sqrt{n}} = \alpha$$

Here Z is the number of standard deivations from the mean we would like to construct our confience interval, σ is our standard deviation, and n is our sample size. All of this information gives us our $\alpha\%$ confidence interval. This formula assumes that the population standard deviation is known. In reality, we rarely know the population standard deviation are faced with estimating a sample standard deviation. As a result, we will use a cousin of the normal distribution, the student-t distribution which approximates a normal distribution with a sufficiently large dataset.

```
curve(dt(x,5),-5,5,xlab='Student-t Distribution')
curve(dt(x,20),-5,5,add = T,col = 2)
curve(dt(x,100),-5,5,add = T,col = 3)
curve(dnorm(x),-5,5,add = T,col = 4,lty = 2,lwd =2)
legend('topleft',legend = c('df = 5','df=20','df=100','Normal'),col = c(1,2,3,4),lty = 1
```



It can be shown that:

$$\frac{\bar{Y} - \mu}{S/\sqrt{n}} \sim t_{n-1}$$

Here S is our sample estimate of the standard deviation and t_{n-1} represents a student-t distribution with $n-1$ degrees of freedom. Lets use this information to derive our confidence intervals.

```r
#Set the seed for reproducibility
set.seed(1)
#Create an empty vector to store our estimates of the sample mean for large draws(100)
store_mean_large_draws <- c()
#lets write a loop to simulate our 100 draws
draws <- 1000
#A matrix to store our estimated confidence interval
ci_holder <- matrix(NA,draws,2)
colnames(ci_holder) <- c('LowerBound','UpperBound')
for(i in 1:draws){
  #We will use the rnorm() function to simulate draws from a normal distribution
  large_random_sample <- rnorm(n = 100,mean = 20,sd = 5)
  #estimate and store the sample mean
  n <- length(large_random_sample)
  store_mean_large_draws[i] <- mean(large_random_sample)
  #Here qt is the number of standard deviations from the mean using a student-t distri
  #We will store the lower and upper bound of our confidence interval
  ci_holder[i,1] <- store_mean_large_draws[i] - qt(0.975,n - 1)*sd(large_random_sample)/
  ci_holder[i,2] <- store_mean_large_draws[i] + qt(0.975,n - 1)*sd(large_random_sample)/
}
#How many of our estimated confidence intervals include the true population average of
print('---True Average in Estimated CI %---')
```

```
## [1] "---True Average in Estimated CI %---"
```

```r
sum(ci_holder[,1] <= 20 & 20 <= ci_holder[,2])/nrow(ci_holder)
```

```
## [1] 0.96
```

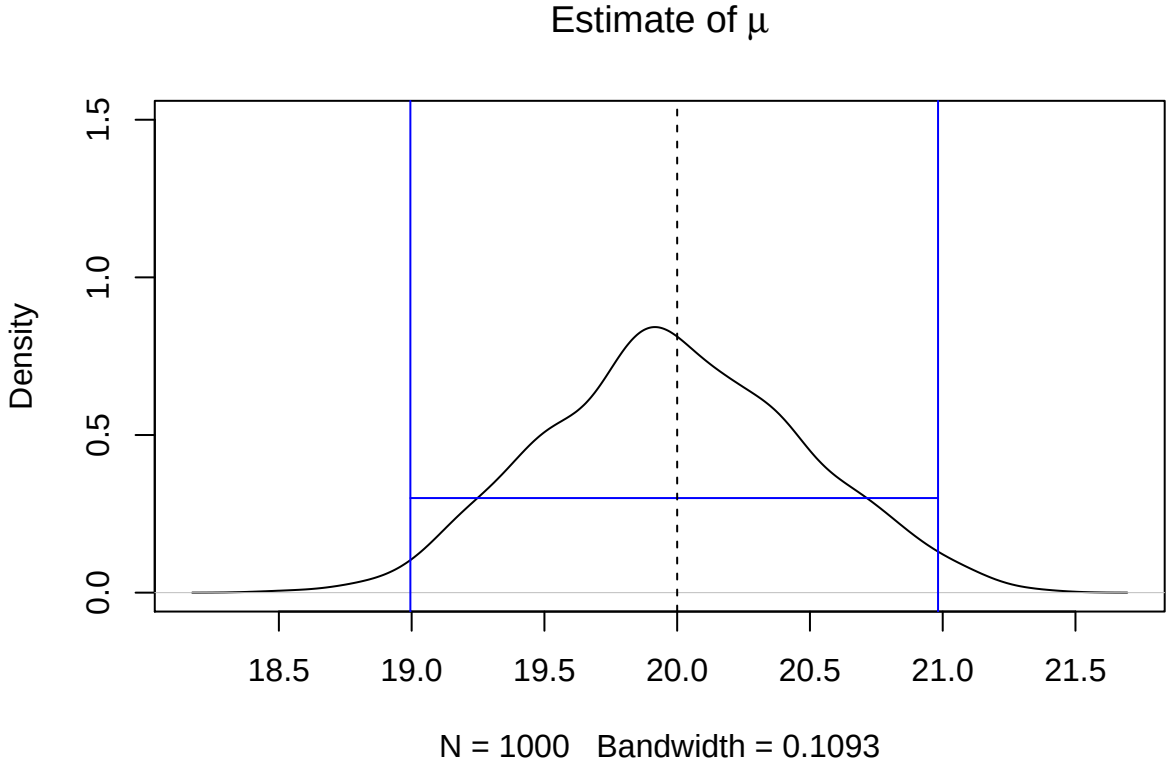
```r
#What is the the average size of our confidence interval estimate?
print('---Average Size of CI---')
```

```
## [1] "---Average Size of CI---"
```

```r
summary(ci_holder[,2] - ci_holder[,1])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.548   1.894   1.988   1.987   2.079   2.417
```

```r
#What is the average lower bound? What is the average upper bound?
avg_ci <- apply(ci_holder,2,mean)


plot(density(store_mean_large_draws),main=expression(paste('Estimate of ',mu)),col = 1,y
lines(avg_ci,c(0.3,0.3),col = 4)
abline(v = 20,lty = 2)
abline(v = avg_ci,col = 4)
```



In approximately 95% of our sample estimated confidence intervals the true mean falls. Note that in our simulation there were only 960 occurrences where the true mean fell in our estimated confidence interval. In general we can state the following:

In repeated sampling, the interval $\bar{Y} \pm \frac{Z\sigma}{\sqrt{n}}$covers the true value of μ with a probability of α%.

Using confidence intervals we can lay the ground work to begin what is known as hypothesis testing. In hypothesis testing we start with a **null hypothesis** (H_0) which we assume is the truth.

We compare the null hypothesis to an **alternative hypothesis** (H_1). We take the null hypothesis as true unless there is strong data suggesting otherwise.

When such testing is done we can run into cases of **type 1** and **type 2** error.

Type 1 error occurs when we reject the null hypothesis when it is in fact true.

Type 2 error occurs when we fail to reject the null hypothesis when it is in fact false.

When conducting a hypothesis test we define a **significance level** (α) which tells us the likelihood of making a type-1 error.

$$\alpha = P(\text{Reject}H_0|H_0).$$

which, is read as the probability of rejecting the null hypothesis given that the null hypothesis is true.

In general a test statistic is constructed and compared against a critical value, where depending on the relative position of the test statistic to the critical value we will reject, or, fail to reject the null hypothesis. If we have a sample estimate $\bar{Y}$ that we would like to compare to a null hypothesis we can do so as follows:

$$H_0 : \bar{Y} = \mu$$

$$H_1 : \bar{Y} \neq \mu$$

Here our null-hypothesis is that the estimated sample mean $\bar{Y}$ is equal to some value μ. The alternative hypothesis is that the estimated sample mean is not equal to μ. This is what is known as a two-sided test as we do not care if the sample mean is greater than or less than the value μ. If we cared about the direction we would use a one sided test, which would look something along the lines of:

$$H_0 : \bar{Y} = \mu$$

$$H_1 : \bar{Y} > \mu$$

or

$$H_1 : \bar{Y} < \mu$$

Regardless of the a one-sided or a two-sided test we will need to construct a test statistic. A test statistic that follows a student-t distribution with n-1 degrees of freedom can be constructed as:

$$T = \frac{\bar{Y} - \mu}{S/\sqrt{n}} \sim t_{n-1}$$

This test statistic can then be compared against critical values from a student-t distribution with n-1 degrees of freedom to determine whether or not we have a statistically different difference between $\bar{Y}$ and μ. Notice that this is the same thing as our constructed confidence intervals!

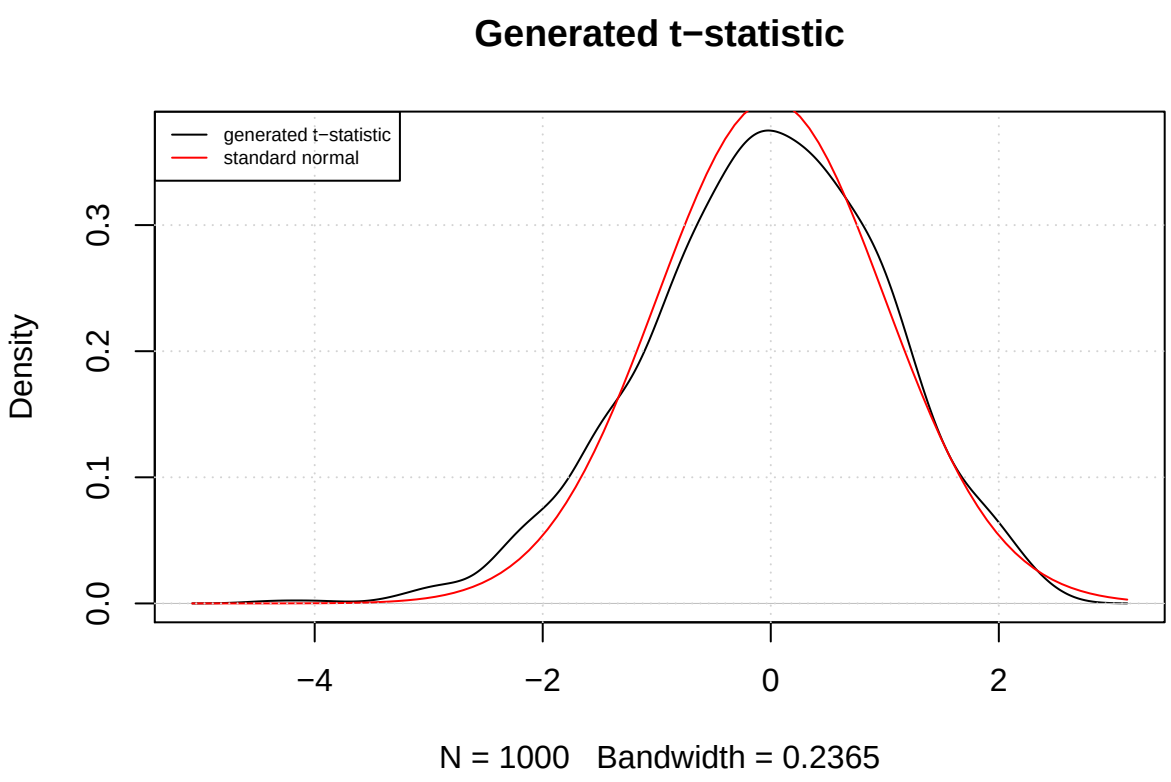
Lets walk through an example of generating t-statistics and check the asymptotic distribution. Recall that by the central limit theorem the standardized estimate of means should be asymptotically standard normally distributed. We'll simulate an unfair coin flip that has a 90% chance of landing on 0 (tails) and a 10% chance of landing on 1 (heads). For each series of coin flips we will test if the coin flip has a mean of 0.1. More formally:

$$H_0 : \bar{Y} = 0.1$$

$$H_1 : \bar{Y} \neq 0.1$$

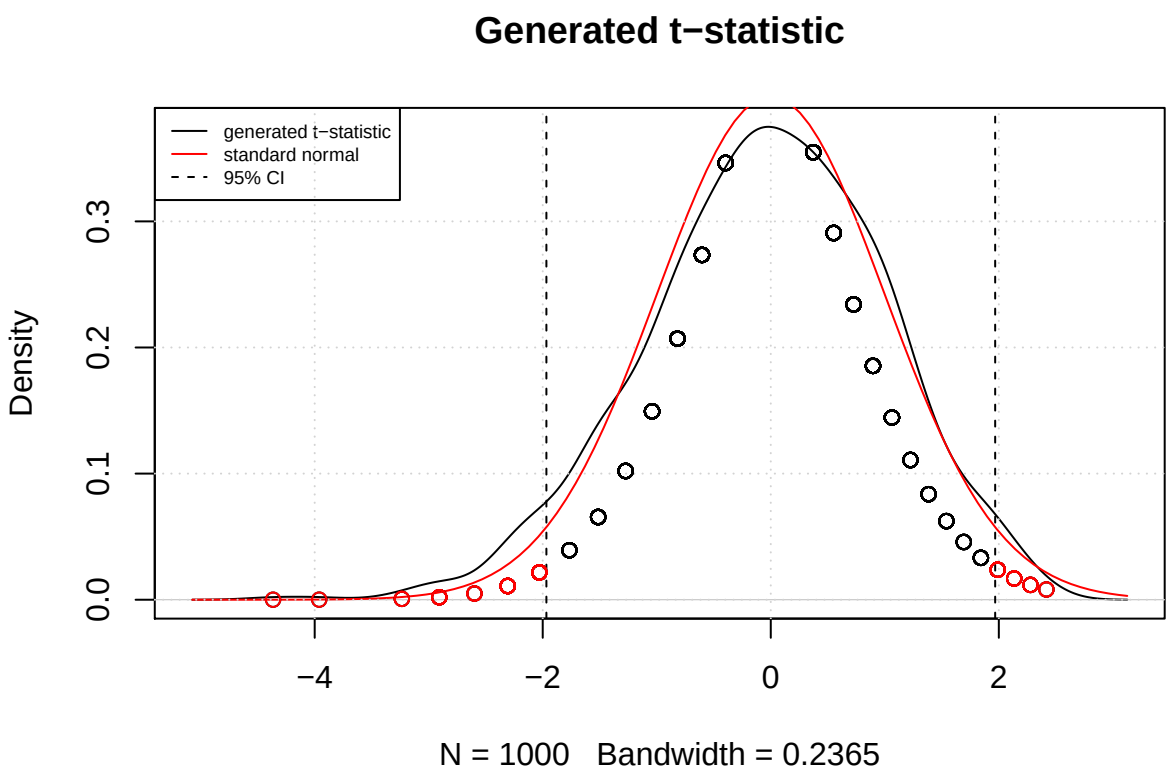
```r
#Set seed for reproducible results
set.seed(123)
#Create vector to store t-statistics
t_stat_holder <- c()
#Define a variable that contains likelihood of tails
prob_tails <- 0.9
#Number of coin flips per simulation
n <- 300
#Number of simulations to run
sim <- 1000
#Run the simulations
for(i in 1:sim){
```

```r
  #Simulate the coin flips with the parameters given outside of the loop
  flips <- sample(0:1,size= n, replace = T,prob = c(prob_tails,1-prob_tails))
  t_stat_holder[i] <- (mean(flips) - 0.1)/sqrt(var(flips)/(n-1))
}
#Plot empirical pdf
plot(density(t_stat_holder),main='Generated t-statistic')
#Add on the standard normal pdf
curve(dnorm(x),add = T,col = 2)
#add a legend (making it smaller with cex command)
legend('topleft',legend = c('generated t-statistic','standard normal'),lty = 1,col =1:2,
#add a grid
grid()
```



We know the true mean is in fact 0.1. As a result the majority of our generated t-statistics should be centered upon zero (which they are). There are however a number of occurences in our simulation where we actually could have generated large absolute t-statistics! Those large (in absolute terms) t-statistics could have rejected our null hypothesis at a variety of *significance levels*. For each of our generated t-statistics we can measure what is known as a p-value. Going back to our definition of type-1 error, if we assume that our null hypthesis is in fact true then our p-value tells us the likelihood of us rejecting the null hypothesis given our generated t-statistic from our observed sample.

```r
generated_p_values <- pt(abs(t_stat_holder),n-1,lower.tail = F)
#Plot empirical pdf
plot(density(t_stat_holder),main='Generated t-statistic')
#Add on the standard normal pdf
curve(dnorm(x),add = T,col = 2)
#add a legend (making it smaller with cex command)
legend('topleft',legend = c('generated t-statistic','standard normal','95% CI'),lty = c(
#add a grid
grid()
#Add vertical lines for top 2.5% of t-dist and bottom 2.5% of t-dist= total 5% likelih
abline(v=qt(0.975,n),lty = 2)
abline(v=qt(0.025,n),lty = 2)
#Add in our estimated p-values
points(t_stat_holder,generated_p_values)
#Highlighlight those simulations that had results significantly different from the tru
points(t_stat_holder[which(generated_p_values <= 0.025)],generated_p_values[which(genera
```




```r
#What percent of simulations did we reject that the mean is in fact 0.1?
length(t_stat_holder[which(generated_p_values <= 0.025)])/sim
```

```
## [1] 0.071
```

Lets look at a real world example of hypothesis testing evaluating student performance using data on California test scores

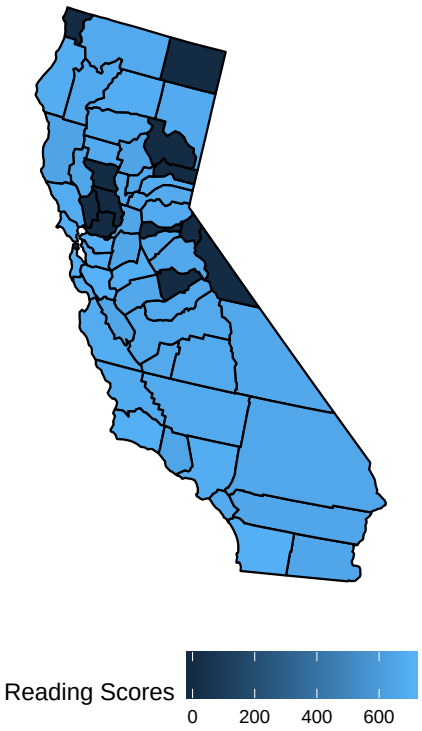
```
#install.packages('AER')
#install.packages('usmap')
#install.packages('ggplot2')
library(AER)
library(usmap)
library(ggplot2)
data(CASchools)
head(CASchools)
```

```
##   district                          school  county grades students
## 1    75119              Sunol Glen Unified Alameda  KK-08      195
## 2    61499            Manzanita Elementary   Butte  KK-08      240
## 3    61549     Thermalito Union Elementary   Butte  KK-08     1550
## 4    61457 Golden Feather Union Elementary   Butte  KK-08      243
## 5    61523        Palermo Union Elementary   Butte  KK-08     1335
## 6    62042         Burrel Union Elementary  Fresno  KK-08      137
##   teachers calworks   lunch computer expenditure    income   english  read
## 1    10.90   0.5102  2.0408       67    6384.911 22.690001  0.000000 691.6
## 2    11.15  15.4167 47.9167      101    5099.381  9.824000  4.583333 660.5
## 3    82.90  55.0323 76.3226      169    5501.955  8.978000 30.000002 636.3
## 4    14.00  36.4754 77.0492       85    7101.831  8.978000  0.000000 651.9
## 5    71.50  33.1086 78.4270      171    5235.988  9.080333 13.857677 641.8
## 6     6.40  12.3188 86.9565       25    5580.147 10.415000 12.408759 605.7
##    math
## 1 690.0
## 2 661.9
## 3 650.9
## 4 643.5
## 5 639.9
## 6 605.4
```

```r
head(countypop)
```

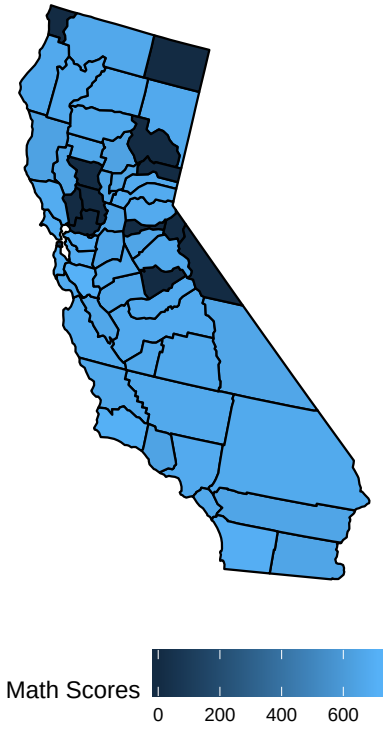
```
## # A tibble: 6 x 4
##   fips  abbr  county          pop_2015
##   <chr> <chr> <chr>              <dbl>
## 1 01001 AL    Autauga County     55347
## 2 01003 AL    Baldwin County    203709
## 3 01005 AL    Barbour County     26489
## 4 01007 AL    Bibb County        22583
## 5 01009 AL    Blount County      57673
## 6 01011 AL    Bullock County     10696
```

```r
ca_county <- countypop[substr(countypop$fips,1,2) == '06',]
ca_county$county_clean <- gsub(' County','',ca_county$county)
ca_county$test_score_read <- 0
ca_county$test_score_read[match(CASchools$county,ca_county$county_clean)] <- CASchools$r

ca_county$test_score_math <- 0
ca_county$test_score_math[match(CASchools$county,ca_county$county_clean)] <- CASchools$m

plot_usmap(data = ca_county,values = 'test_score_read',include ='CA') +
  labs(title = "CA Counties",
       subtitle = "") +
  theme(panel.background = element_rect(color = "white", fill = "white"),
        legend.position = 'bottom')+
  scale_fill_continuous('Reading Scores')
```




```
plot_usmap(data = ca_county,values = 'test_score_math',include ='CA') +
  labs(title = "CA Counties",
       subtitle = "") +
  theme(panel.background = element_rect(color = "white", fill = "white"),
        legend.position = 'bottom')+
  scale_fill_continuous('Math Scores')
```

CA Counties




```
ca_county$total_score <- ca_county$test_score_math + ca_county$test_score_read
plot_usmap(data = ca_county,values = 'total_score',include ='CA') +
  labs(title = "CA Counties",
       subtitle = "") +
  theme(panel.background = element_rect(color = "white", fill = "white"),
        legend.position = 'bottom')+
  scale_fill_continuous('Total Scores')
```

CA Counties