Lab Tutorial 2: Inferential Statistics Review

Chris M. Fiacconi

Variability Among Sample Means

Just as individual IQ scores can vary within a given sample, so too can sample means vary *across* samples. Imagine we took 300 samples from our Guelph population, each sample containing 300 observations. We could compute the mean IQ for each of those 300 samples, and look at how the 300 means are distributed. Sampling variability ensures that you will not obtain the same means in all 300 samples. Variability among means from sample to sample is very important and forms the bedrock of many inferential test statistics (e.g., t, F, etc.)

Let's get 300 samples of 300 IQ scores:

```
set.seed(4910) # set random number generator
IQpop<-rnorm(n=20000,mean=100,sd=15) # Create population of UG students' IQ scores
samp.list<-list() # initialize list to hold samples
nIterations<-300 # set number of samples to take
nObs<-300 # set number of observations per sample
set.seed(83557)

# Create a loop in which we take a sample of 50 observations and repeat this process 100 times.
# Each sample of 50 observations is stored in a list variable named 'samp.list'

for (i in 1:nIterations) {
samp.list[[i]]<-sample(x=IQpop,size=nObs)
}

# Now let's get the mean for each of our 100 samples and store them in a new variable
# called 'smplmeans'

smplmeans<-sapply(X=samp.list,FUN=mean)

# Now let's plot a histogram of the sample means
hist(smplmeans,breaks=8,las=1,col="orange",xlab="Sample Mean",xlim=c(90,110),
     ylab="Frequency",main="Guelph IQ Sample Means")
```

```
# Get the mean of the sample means, and the standard deviation of the sample means
grdsmplmean<-round(mean(smplmeans),digits=2)
print(grdsmplmean)
```

```
## [1] 99.87
```

```
SEM<-round(sd(smplmeans),digits=2)
print(SEM)
```

```
## [1] 0.86
```

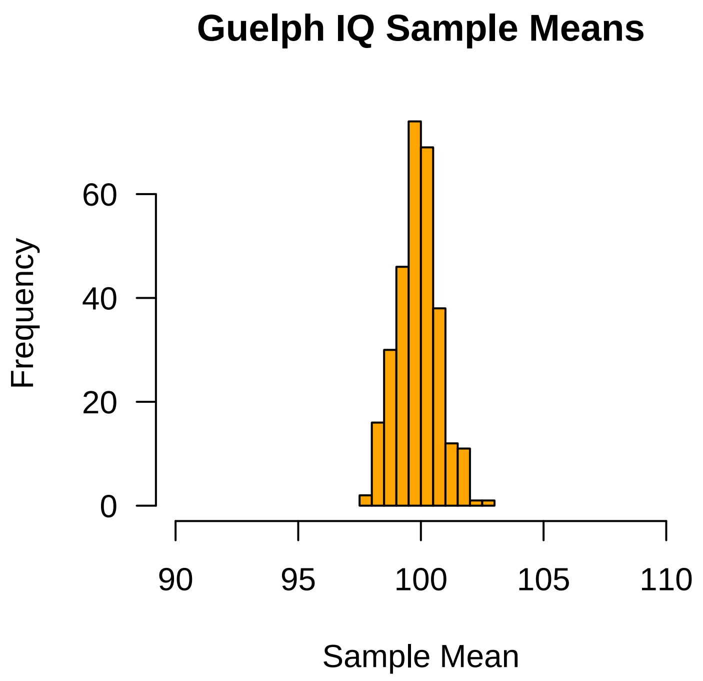

Figure 1: Histogram depicting variability among sample means

The standard deviation of the sample means has a special name - the **standard error of the mean**, and it represents how far, on average, a sample mean will fall from the true population mean.

Fortunately, we do not have to take multiple samples to estimate the SEM. It turns out that we can estimate it directly from ONE sample, by dividing the sample SD (s) by the square root of our sample size (n):

$$SEM = \frac{\sigma}{\sqrt{n}} \approx \frac{s}{\sqrt{n}}$$

Let's get three different estimates of the SEM from our first three samples:

```
stderr1<-round(sd(samp.list[[1]])/sqrt(nObs),digits=2)
print(stderr1)
```

```
## [1] 0.9
```

```
stderr2<-round(sd(samp.list[[2]])/sqrt(nObs),digits=2)
print(stderr2)
```

```
## [1] 0.86
```

```
stderr3<-round(sd(samp.list[[3]])/sqrt(nObs),digits=2)
print(stderr3)
```

```
## [1] 0.88
```

Estimates of the SEM from each sample are similar to the direct estimate of the SD of the sample means (.84). Both of these methods are aimed at measuring the **SAME** quantity, but they can be slightly different in this case because of sample size. We took 300 samples, but each sample consisted of only 300 IQ scores. These different procedures for estimating the SEM will converge as the number of observations per sample increases.

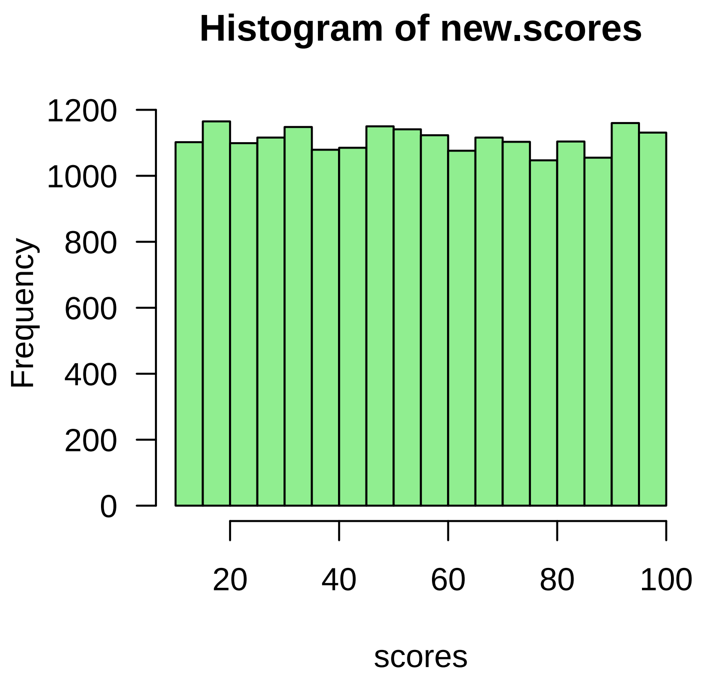

Figure 2: Histogram depicting a population of scores with a uniform distribution

The Central Limit Theorem

Consider the following population of scores:

```r
set.seed(41924)
new.scores<-runif(n=20000,min=10,max=100)

hist(x=new.scores,breaks=20,las=1,col="lightgreen",xlab="scores",
     ylab="Frequency",main="Histogram of new.scores")
```

The central limit theorem states that the distribution of sample means will *always* be normally distributed even if the parent population distribution is non-normally distributed. Let's draw a series of 500 samples with 1000 observations each from the uniform population distribution shown above and plot the sample means in a histogram:

```r
nIterations<-500
nObs<-100
sample.list<-list()

for (i in 1:nIterations) {
  sample.list[[i]]<-sample(x=new.scores,size=nObs)
}

sample.means<-sapply(X=sample.list,FUN=mean)

hist(x=sample.means,breaks=20,las=1,col="lightgreen",xlab="Sample Means",
     ylab="Frequency",main="Histogram of sample.means")
```

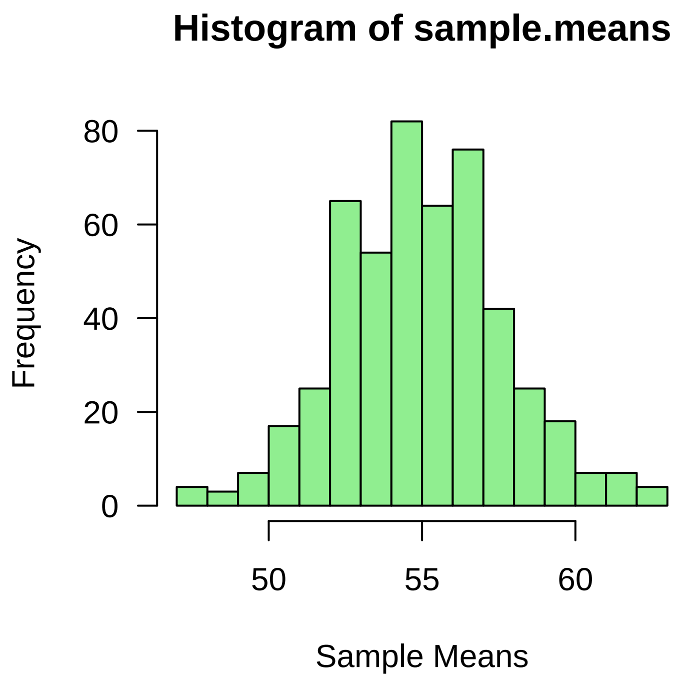

Figure 3: Histogram depicting normal disribution of sample means where each sample was drawn from a uniform distribution

95% Confidence Intervals

Confidence intervals represent a plausible range of values within which the true population mean is likely to fall. Here, we will learn to create 95% confidence intervals, plot the results in a figure, and evaluate hypotheses about a sample mean. Given that we now know that sample means are distributed normally, we can estimate the bounds within which sample means will fall 95% of the time using the *t*-distribution along with the standard error of the mean (SEM). Let's use our population of IQ scores from the example above.

```
set.seed(49921)
nIterations<-20 # number of samples drawn from population
nObs<-100 # size of each sample

sample.list<-list() # initialize list to hold individual samples

for (i in 1:nIterations){
sample.list[[i]]<-sample(x=IQpop,size=nObs)
}
```

Let's Calculate the mean, standard deviation, and SEM for each of 20 samples of 100 observations using the **sapply** function:

```
means<-sapply(sample.list,mean)
SD<-sapply(sample.list,sd)
SEM<-SD/sqrt(nObs)


samples<-(seq(1,nIterations)) # Create samples variable (1:20)
```

```
all.sampledata<-data.frame(samples,means,SEM) # Combine samples, means, and SE variables into one dataf
head(all.sampledata)
```

```
##   samples     means      SEM
## 1       1  97.80609 1.616347
## 2       2  99.84247 1.590904
## 3       3  96.85179 1.524665
## 4       4  97.39699 1.621966
## 5       5 102.12052 1.673676
## 6       6 100.10265 1.761509
```

Now, were going to plot each sample mean along with its 95% CI:

```
tvalue<-qt(p=.975,df=nObs-1) # get t-value associated with the 97.5th percentile and 99 df

MoE<-SEM*tvalue # calculate margin of error (MoE) for 95% CI

plot(x=all.sampledata$samples,y=all.sampledata$means,type="p",las=1,cex=1.5,pch=17,ylim=c(90,110),
     xlab="Sample No.",ylab="Sample Mean")

arrows(x0=all.sampledata$samples,y0=all.sampledata$means,x1=all.sampledata$samples,
       y1=all.sampledata$means-MoE,angle=90,length=.1) # Draw upper half of 95% CI

arrows(x0=all.sampledata$samples,y0=all.sampledata$means,x1=all.sampledata$samples,
       y1=all.sampledata$means+MoE,angle=90,length=.1) # Draw lower half of 95% CI

abline(100,0,col="red",lty=2,lwd=2) # Draw line representing population mean
```

You'll notice that the 95% CI around ONE of our samples does not include the population mean. This is because, by definition, our confidence interval for each sample will, on average, contain the true population mean 95% (19/20) times.

t-tests

We will now learn how to test hypotheses about population means with the *t*-statistic. *t*-tests are used to evaluate hypotheses about means when the standard deviation of the underlying population(s) is unknown (which is usually the case).

One-Sample *t*-test

Suppose that research on alcohol abuse has shown that individuals who consume more than 15 alcoholic beverages per week are at risk of developing an alcohol dependency. Let's ask whether Guelph students, on average, consume more than 15 drinks per week. To do so we need to take a random sample of students, and ask them how many drinks they consume per week. We don't actually know the true mean or sd across ALL students at Guelph, so we must infer these parameters from our sample.

```
set.seed(123)
UGpop<-rnorm(n=20000,mean=16,sd=6) # Create population - 20,000 students, mean = 16, sd=6
```

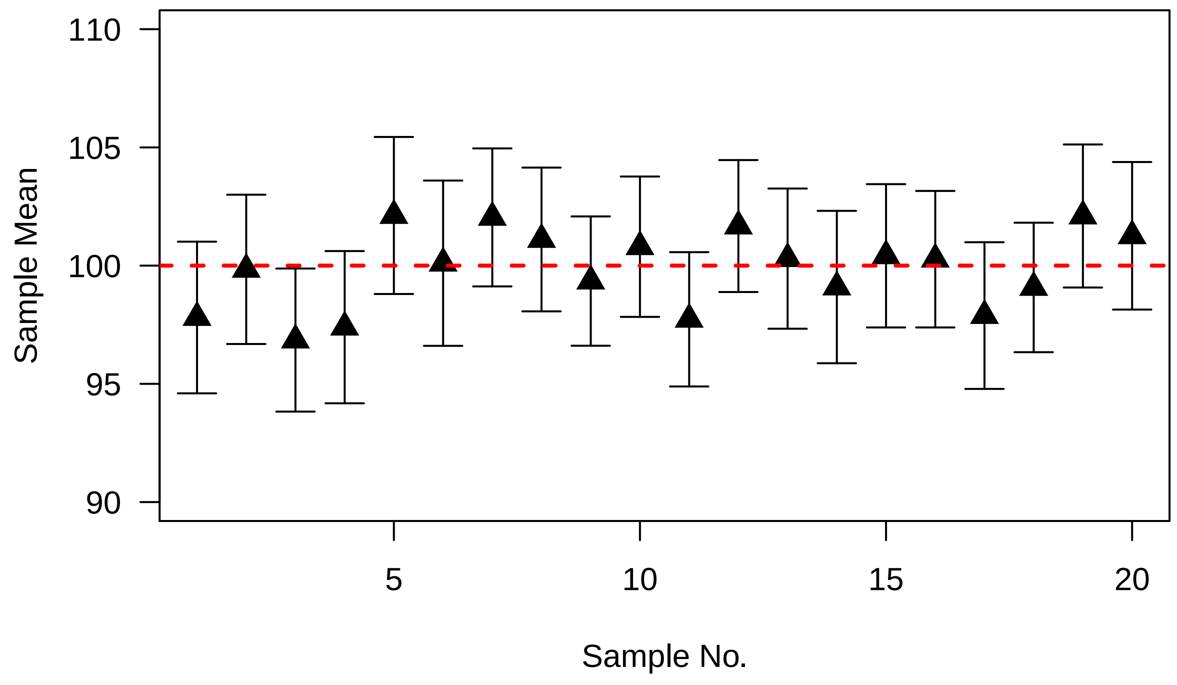

Figure 4: Figure shows dance of the 95% CIs. On average, 1/20 such CIs will not contain the true population mean

Although we don't actually know the true population parameters in practice, to illustrate the logic, I am making them up here. Now, let's take a random sample of 20 students from this population, and calculate the mean and SD of our sample.

```
set.seed(923)
sample1<-sample(x=UGpop,size=20)

s1mean<-round(mean(sample1),digits=2)
s1sd<-round(sd(sample1),digits=2)

print(s1mean)
```

```
## [1] 16.06
```

```
print(s1sd)
```

```
## [1] 6.4
```

Our estimate of the mean number of drinks consumed by Guelph students is 16.06. Is this significantly different than the reference value of 15?

```
t.test(x=sample1,mu=15)
```

```
##
##  One Sample t-test
##
```

```
## data:  sample1
## t = 0.74322, df = 19, p-value = 0.4664
## alternative hypothesis: true mean is not equal to 15
## 95 percent confidence interval:
##  13.06686 19.06196
## sample estimates:
## mean of x
##  16.06441
```

You'll notice, that our *t*-test failed to reject the null hypothesis that Guelph students on average drink more (or less) than 15 drinks per week. But wait, why? We set up the population of Guelph students such that the mean was 16 drinks per week. So, in reality, they do drink more than 15 per week. Why wasn't this reflected in our *t*-test results? **POWER!!**

Power

Power is the probability that our experiment will be able to detect a true effect in the population. You can think of it as a measure of how sensitive our experiment is to true differences in means in the population. Power is very useful concept to consider when planning an experiment, as it can help us decide how many participants to test to achieve a desired level of power. As n increases, so does our power. A widely accepted power value is 0.8, which means that our statistical test would be able to detect a true difference in population means 80% of the time. We can use a power calculation to determine the number of participants we need to test to reach a power of 0.8.

To calculate the n needed for a power = 0.8, we need to know a few things first. First and foremost, we need an estimate of the *effect size* in the population. A standard measure of effect size for an experiment with one or two groups is Cohen's d, which expresses the difference in population means in units of population standard deviation. In the case of a one-sample t-test when the mean and SD of the population from which our sample is drawn is known, calculation of Cohen's d is pretty straightforward:

$$d = \frac{\bar{Y} - \mu_0}{\sigma} \approx \frac{\bar{Y} - \mu_0}{s}$$

Simply subtract the reference mean (15) from the population mean (16) and divide by the population standard deviation = (16 - 15)/6. Otherwise, if you didn't know the true population mean or SD (which is usually the case), you would have to make an educated guess as to the effect size in the population. In this case, Cohen's d = .167, which is a relatively small effect size. How many participants do we need to test to achieve a power of 0.8?

We'll use the **pwr.t.test** function in the *pwr* package.

```
install.packages("pwr") # make sure you're connected to the Internet!!
```

```
library(pwr) # load the pwr package into the R environment
```

Now, let's do the calculation

```
pwr.t.test(n=NULL,d=0.167,sig.level=0.05,power=0.8,type="one.sample",
           alternative="two.sided") # Set alpha level (sig.level) = 0.05
```

```
##
##      One-sample t test power calculation
```

```
##
##                n = 283.3589
##                d = 0.167
##        sig.level = 0.05
##            power = 0.8
##      alternative = two.sided
```

We need a sample of 284 students to achieve a power of 0.8!!

Going back to our initial sample, we can also ask: What power was achieved with our initial sample of 20 participants?

```
pwr.t.test(n=20,d=0.167,sig.level=0.05,power=NULL,type="one.sample",
           alternative="two.sided") # Set alpha level (sig.level) = 0.05
```

```
##
##      One-sample t test power calculation
##
##                n = 20
##                d = 0.167
##        sig.level = 0.05
##            power = 0.1093992
##      alternative = two.sided
```

An experiment with 20 participants would detect an effect = .167 only ~11% of the time! No wonder we did not find a significant difference between our estimate of drinks consumed by UG students and the reference mean.

We can also use the pwr.t.test function to calculate the obtained power with a sample of 300 students. Given the previous calculation, the power associated with an $n = 300$ should be greater than .80.

```
pwr.t.test(n=300,d=0.167,sig.level=0.05,power=NULL,type="one.sample",
           alternative="two.sided") # Set alpha level (sig.level) = 0.05
```

```
##
##      One-sample t test power calculation
##
##                n = 300
##                d = 0.167
##        sig.level = 0.05
##            power = 0.8220643
##      alternative = two.sided
```

With a sample of 300 participants, our experiment will be able to detect an effect size in the population = .167 on ~82% of occasions.

Let's do the same experiment again, but now take a larger sample of 300 students.

```
set.seed(123456)
sample2<-sample(x=UGpop,size=300)

s2mean<-round(mean(sample2),digits=2)
s2sd<-round(sd(sample2),digits=2)

print(s2mean)
```

```
## [1] 16.13
```

```
print(s2sd)
```

```
## [1] 6.2
```

Our new estimate of the mean number of drinks consumed by Guelph students is 16.13. Is this significantly different than the reference point of 15?

```
t.test(x=sample2,mu=15)
```

```
##
##  One Sample t-test
##
## data:  sample2
## t = 3.1644, df = 299, p-value = 0.001714
## alternative hypothesis: true mean is not equal to 15
## 95 percent confidence interval:
##  15.42832 16.83737
## sample estimates:
## mean of x
##  16.13285
```

Our p-value is now $< .05$, so we reject the null hypothesis that Guelph students, on average, consume 15 drinks per week. Notice that the *size* of the difference in the second sample (16.13 - 15 = **1.13**) is roughly the same as that obtained in our first sample (16.06 - 15 = **1.06**). Therefore, the effect of increasing sample size is to reduce the error variability (noise), making it easier for us to see the true difference between our population mean (drinks among UG students) and the reference point of 15.

Independent samples *t*-test

In the case of the one-sample t-test, we were interested in evaluating the null hypothesis that Guelph students consumed a particular number (15) of drinks per week. What if we wanted to know whether Guelph students drank more or less than Western students? How could we evaluate the null hypothesis that U of G and Western students consume an equal number of drinks per week? We need to take two independent samples - one from Guelph and one from Western and compare them. Let's set the population parameters for Western students.

```
set.seed(41273)
UWOpop<-rnorm(n=30000,mean=18,sd=6) # Create population - 30,000 students, mean = 18, sd=6
```

Now, let's draw a sample of 200 students from each university

```
set.seed(92742)
UGsample<-sample(x=UGpop,size=200)
UWOsample<-sample(x=UWOpop,size=200)
```

Let's create a boxplot to visually inspect each sample

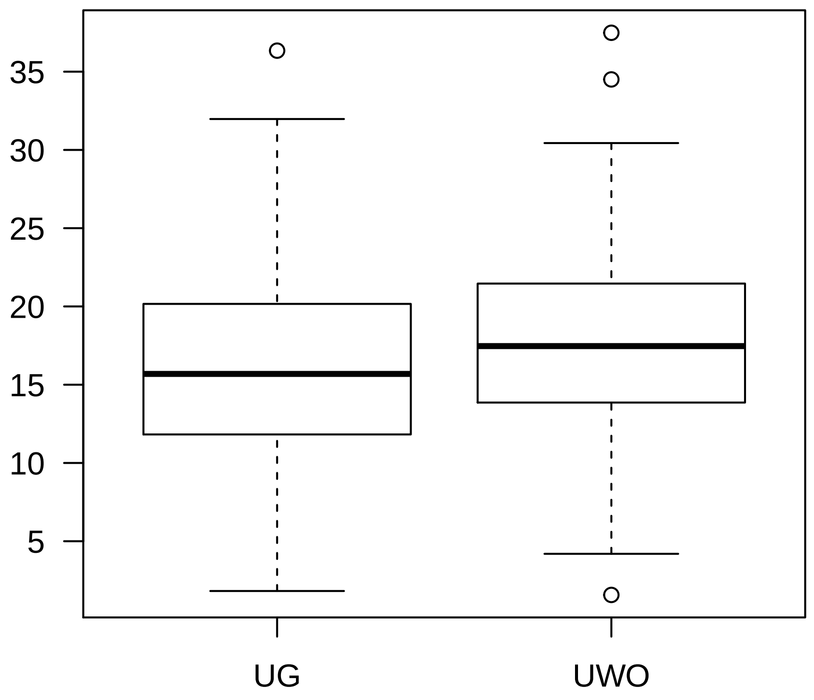

Figure 5: Boxplot depicting number of drinks consumed by students at two Ontario universities

```r
scores<-c(UGsample,UWOsample)
school<-rep(c("UG","UWO"),times=1,each=200)
drinkdata<-data.frame(school,scores)
head(drinkdata) # take a peek at the new data frame with a column for school and scores
```

```
##   school    scores
## 1     UG 14.652082
## 2     UG 16.780095
## 3     UG  9.084823
## 4     UG 11.483564
## 5     UG 22.872464
## 6     UG  1.819821
```

```r
boxplot(scores~school,las=1,data=drinkdata)
```

To evaluate the null hypothesis that Guelph and Western students consume an equal number of drinks per week, we need to run an independent-samples t-test.

```r
t.test(scores~school,paired=F,var.equal=T,data=drinkdata)
```

```
##
##  Two Sample t-test
##
## data:  scores by school
## t = -2.7419, df = 398, p-value = 0.006384
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
```

```
##  -2.8063881 -0.4625642
## sample estimates:
##  mean in group UG mean in group UWO
##          15.85134          17.48582
```

Our p-value is $< .05$, so we reject the null hypothesis that UG and UWO students consume an equal number of drinks per week.

Effect Size

What is our effect size? For an independent groups design Cohen's d is calculated as follows:

$$d = \frac{\bar{Y_2} - \bar{Y_1}}{SD_{pooled}}$$

Again, let's let the software do the work. We'll install and load the *effsize* package and use the function **cohen.d**.

Calculate Cohen's d:

```r
install.packages("effsize") # install package - make sure you're online!!
```

```r
library(effsize) # load in the effsize package
chn_d<-cohen.d(scores~school,data=drinkdata,noncentral=TRUE)
print(chn_d)
```

```
##
## Cohen's d
##
## d estimate: -0.2741917 (small)
## 95 percent confidence interval:
##       lower       upper
## -0.47094064 -0.07710153
```

Although the estimate of Cohen's d is -.27, the negative sign is unimportant and simply reflects the order in which the two means were subtracted. What's important is the absolute magnitude, so for our purposes, let's interpret d = -.27 as d = .27. This would be considered a small effect according to Cohen's criteria (.2 to .5 = small; .5 to .8 = medium; > .8 = large).

You'll also notice that a 95% CI for Cohen's d is provided by this function. Similar to 95% CI's around sample means, this CI represents a range of plausible values within which the true *population* effect size is likely to fall.

Paired-samples *t*-test

Although some research questions necessitate the use of an independent-groups design, repeated-measure designs (or *paired designs* in the case of a study with two conditions), are generally a more powerful design. This increase in statistical power stems from the fact that each individual is tested in both experimental conditions, and so variability in our dependent measure owing to differences *between* individuals can be ignored. This can drastically reduce the amount of error variability in our design, and consequently makes it easier to detect a true difference between population means.

Let's go through an example: A researcher wants to know if people can improve their math skills with practice. The researcher takes a random sample of 20 individuals, and tests the speed with which they complete 40 multiplication questions on two consecutive days. Because each participant is tested twice (once on each day), this constitutes an example of a paired design - that is, each participant's score on day one is *paired* with his/her score on day two.

First, let's load in the data file:

```
mathdata<-read.table(file="mathdata.txt",header=T)
```

Let's take a peek at the data! Note that it is arranged in long format:

```
head(mathdata,n=10)
```

```
##    Subject  Day CompleteTime
## 1        1 Day1     4.829410
## 2        2 Day1     4.480651
## 3        3 Day1     3.696141
## 4        4 Day1     4.600279
## 5        5 Day1     6.779562
## 6        6 Day1     4.802778
## 7        7 Day1     5.658276
## 8        8 Day1     5.267206
## 9        9 Day1     3.200148
## 10      10 Day1     4.969112
```

Notice how the data are organized - separate columns for Subject, Day, and our dependent variable (CompleteTime) - **This is important**

Now let's get the means for each condition, and then plot the data to see how it looks:

```
library(tidyverse) # Load in tidyverse
sample.size<-20

mathmeans<-mathdata %>% group_by(Day) %>% summarize(Mean=mean(CompleteTime))
print(mathmeans)
```

```
## # A tibble: 2 x 2
##   Day    Mean
##   <fct> <dbl>
## 1 Day1   5.51
## 2 Day2   4.60
```

```
bp_mean<-barplot(height=mathmeans$Mean,names.arg=mathmeans$Day,beside=T,las=1,
                 ylim=c(0,10),ylab="Completion Time (sec)")


points(rep(bp_mean[1],times=sample.size),filter(mathdata,Day=="Day1")[,3],cex=2,
       pch=19)
points(rep(bp_mean[2],times=sample.size),filter(mathdata,Day=="Day2")[,3],cex=2,
       pch=19)
segments(bp_mean[1],filter(mathdata,Day=="Day1")[,3],bp_mean[2],
         filter(mathdata,Day=="Day2")[,3])
```

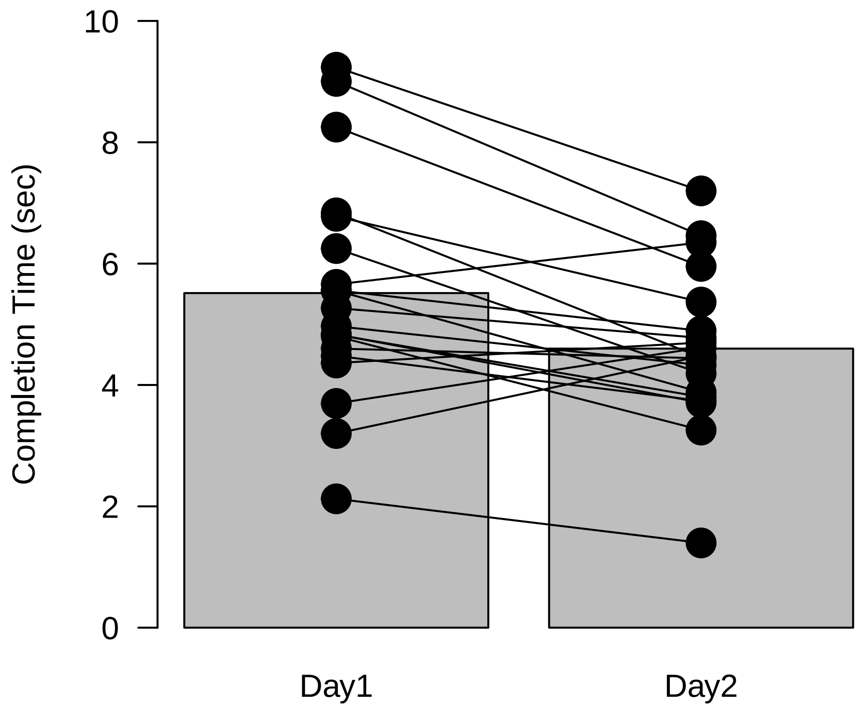

Figure 6: Bar graph showing math performance as a function of Day. Lines represent performance of individual subjects on days 1 and 2.

```r
# The segments function draws connecting lines for each subject
```

In this plot, the gray bars represent the mean completion times (in seconds) for Day 1 and Day 2. The filled circles represent individual data points, and the lines connect data points from the same subject.

A few things should be noted here:

- 1) Not all subjects show a reduction in completion time from Day 1 to Day 2.
- 2) There are individual differences in completion time within each Day condition, but it doesn't matter! We only care about how a given individual *changes* from Day 1 to Day 2

How do we evaluate the null hypothesis that the completion times in the population do NOT differ by Day?

We need our *t*-test to account for the fact that we no longer need to worry about differences between individuals in completion times, but rather we care only about how *consistent* the reduction in completion times are from Day 1 to 2 **within** individuals.

```r
t.test(CompleteTime~Day,paired=T,data=mathdata)
```

```
##
##  Paired t-test
##
## data:  CompleteTime by Day
## t = 3.655, df = 19, p-value = 0.001684
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
```

```
##  0.3911375 1.4393710
## sample estimates:
## mean of the differences
##               0.9152543
```

Note how I change the *paired* argument in the t.test function to **TRUE.** Our p-value is $< .05$, so we reject the null hypothesis that the time to solve math problems does not change with practice.

Effect Size

What is our effect size? For a paired design, Cohen's d can be calculated as follows:

$$d = \frac{\bar{Y_2} - \bar{Y_1}}{SD_{avg}}$$

We'll use the **cohensd_rm** function in the *itns* package built by Geoff Cumming & Robert Calin-Jageman.

```r
install.packages("remotes") # make sure you're online!!
remotes::install_github("gitrman/itns") # use install_github function from remotes package
```

```r
library(itns) # load itns package into R

# Store Completion Times for each Day in seperate variables
d1<-filter(mathdata,Day=="Day1")[,3] # Get completion times for Day 1
d2<-filter(mathdata,Day=="Day2")[,3] # Get completion times for Day 2

# Calculate Cohen's d
chn_d_rm<-cohensd_rm(x=d1,y=d2,ci=95) # Get Cohen's d for paired design
print(chn_d_rm)
```

```
##       est        ll        ul
## 0.5835008 0.2146888 0.9409143
```

The *est* column contains our estimate of the population effect size (d = .58), and the *ll* and *ul* columns represent the lower and upper limits of the 95% CI for the population effect size.