STA333 Lecture 20
Introduction to bootstrapping

20.1 Introduction

Statistics has changed drastically in the past decade or so as a result of modern computing and software that enable us to look at data graphically and numerically in ways that were previously inconceivable. These advances have also opened up doors to methods of data analysis that would have been completely impractical (if not impossible) before. Permutation tests are an example of this: the theory behind them was developed long ago and is so straightforward and elegant, but they were impossible to perform in a practical sense with real world problems before resampling software came on the scene.

The "**bootstrap**" is another method of resampling your data, but it has a different goal in mind as opposed to resampling as we did in permutation tests. But it is a nonparametric resampling method, so it has a lot of the same benefits that permutation tests had:

1. **Fewer assumptions.** Resampling methods such as permutation tests or bootstrapping do not require normally distributed populations or large samples to work.

2. **Better accuracy.** Resampling methods can be more accurate that traditional methods.

3. **Generalizability.** Permutation tests and bootstrap methods are remarkably similar for a wide array of different statistics, so they do not require new formulas or "starting from scratch" for every new statistic you encounter. If you understand the general process taking place, it is easy to adapt the process to new problems.

So now, on to this bootstrap thing. What is it, anyway?

20.2 Bootstrapping 101

Bootstrapping is a method of resampling your observed data so as to estimate the standard error (SE) and sampling distribution of some statistic, like the sample mean.

To introduce the notion of bootstrapping, let's see how it works with a particular example. I'll start by showing you how to bootstrap, and then relate the results to ideas of standard errors and sampling distributions.

► **"Baby steps" example.** Suppose you had the following data sampled from some population:

23.3 26.1 19.0 28.8 29.0

We want to compute a 95% confidence interval for the mean of the population using these $n = 5$ measurements. Tradition would be to just apply the usual *t*-based confidence interval formula:

$$\bar{x} \pm t_{0.025} \frac{s}{\sqrt{n}}$$

The quantity $s/\sqrt{n}$ is called the **standard error** of the sample mean, and may be written as $SE_{\bar{x}}$. However, *what if your population were not normal*? Then this CI formula isn't valid. Alright, then check your sample for normality using something like a normal quantile-quantile plot. That's difficult, though, because the plot won't show much if *n* is small. So what can we do?

Statistical inference is based on the sampling distribution of sample statistics. **The bootstrap, first and foremost, is a way of finding the sampling distribution (at least approximately) from just one sample.** Here is how you do it:

1. **Resample.** Create hundreds, if not thousands, of new samples called **bootstrap samples**, by sampling *with replacement* from the original sample. Each bootstrap sample is the same size as the original sample.
2. **Calculate the bootstrap distribution.** Calculate the statistic of interest from each resampling. The distribution of these resample statistics is called the *bootstrap distribution*.
3. **Use the bootstrap distribution.** The bootstrap distribution gives us information about the shape, center and spread of the sampling distribution of the statistic.

For the example above, here is an illustration of the process using R to do the resampling. The code below creates three different bootstrap samples:

```
> x <- c(23.3,26.1,19.0,28.8,29.0)
> x
[1] 23.3 26.1 19.0 28.8 29.0

> bootsample1 <- sample(x, size=length(x), replace=TRUE)
> bootsample1
[1] 26.1 23.3 23.3 19.0 19.0
```

```
> bootsample2 <- sample(x, size=length(x), replace=TRUE)
> bootsample2
[1] 26.1 26.1 19.0 19.0 29.0

> bootsample3 <- sample(x, size=length(x), replace=TRUE)
> bootsample3
[1] 29.0 26.1 26.1 29.0 26.1
```

Since bootstrapping resamples the original data with replacement, any value may be drawn once, more than once, or not at all.

This example illustrates the notion on a very small scale. **In practice, we would draw literally hundreds or thousands of bootstrap samples, not just three!** To facilitate this in software, using loops much like we did with permutation tests will automate the process. Here is a simple R program that collects 1000 bootstrap samples and finds the means of each:

```
R <- 1000
bootmean <- numeric(R)

x <- c(23.3,26.1,19.0,28.8,29.0)
mean(x)     # mean of the original sample

for (i in 1:R) {
  bootsample <- sample(x, size=length(x), replace=TRUE)
  bootmean[i] <- mean(bootsample)
  }
```

We'll try this out in class and see what we get. ◄

The bootstrap idea

The original sample represents the population from which it was drawn. So, resamples from *this* sample represent what we would get if we took many resamples from the population.
The bootstrap distribution of the statistic, based on many resamples, represents the sampling distribution of the statistic.

Why does bootstrapping work? It might seem that bootstrapping creates data out of nothing. This seems a little suspicious. But we are not using the resampled observations as if they were real data—the bootstrap is *not* a substitute for gathering more data to improve accuracy. Instead, the bootstrap idea is to use the resample means to estimate how the sample mean from a sample of size $n = 5$ from this population varies because of random sampling.

Using the data twice—once to estimate the population mean $\bar{x}$, and again to estimate the variation in the sample mean—is perfectly legal. In fact, you've done this many times before when you used the same sample to calculate both $\bar{x}$ and $s/\sqrt{n}$. What is different now is that:

1. We compute the standard error $SE_{\bar{x}}$ via resampling instead of the formula $s/\sqrt{n}$.
2. We use the bootstrap distribution to see if the sampling distribution is approximately normal, rather than just hoping that *n* is large enough for the CLT to apply.

Moreover, the beauty of this method is that it applies to statistics other than the sample mean. We can use bootstrapping in situations where traditional statistical approaches completely break down. For example, we can use bootstrapping to estimate the standard error and sampling distribution of

- the sample median
- a ratio of sample means
- a difference in sample means
- a ratio of sample medians
- a correlation (Pearson or Spearman, you pick!)
- a regression slope
- a ratio of regression slopes
- etc, etc, etc…

Bootstrapping is highly flexible and adaptable to many scenarios. I will show you some applications in the next couple of lectures.

20.2 Standard errors and sampling distributions via bootstrapping

The **standard error (SE) of a statistic** is a measure of the variability that the statistic has due to random sampling. In short, it is the standard deviation of the statistic. Knowing the standard error is crucial for quantifying the level of certainty (or uncertainty) we may place in our sample estimate.

The **bootstrap standard error (SE) of a statistic** is just the standard deviation of the bootstrap distribution of that statistic. I illustrate this using the previous example:

```
x <- c(23.3,26.1,19.0,28.8,29.0)
se.trad <- sd(x)/sqrt(length(x))
se.trad          # SE of the mean by traditional formula
[1] 1.874193

for (i in 1:R) {
   bootsample <- sample(x, size=length(x), replace=TRUE)
   bootmean[i] <- mean(bootsample)
   }

sd(bootmean)     # SE of the mean estimated by bootstrapping
[1] 1.672297
```

The standard error of $\bar{x}$ via the traditional formula is SE = 1.874. However, a bootstrap estimate of the standard error based on 1000 bootstrap samples is SE = 1.672.

Sampling distributions

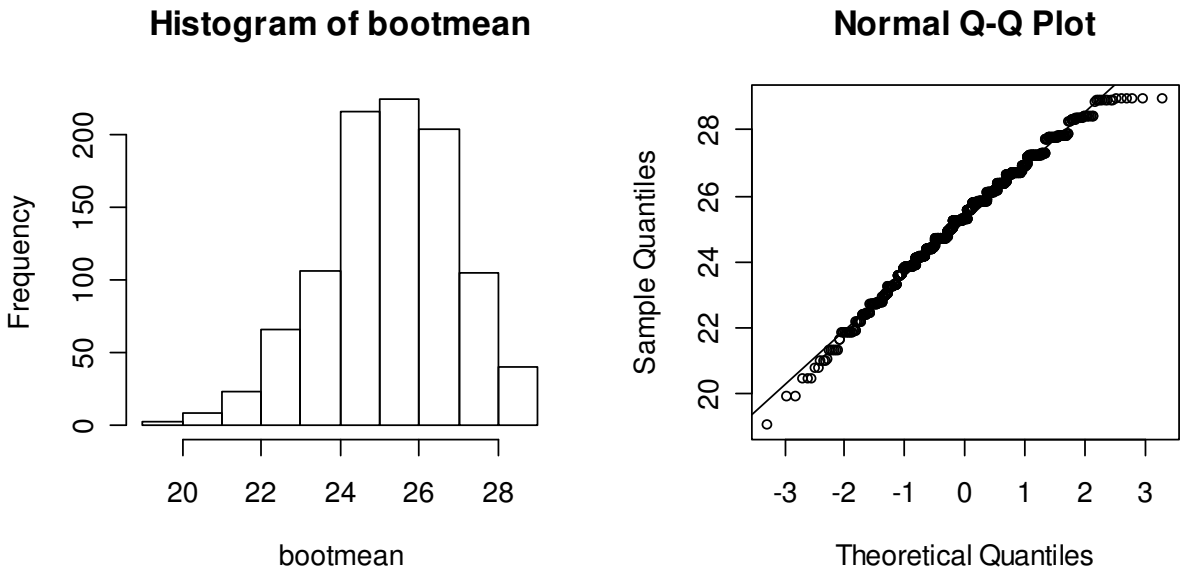
We should also inspect the *shape* of the sampling distribution of our statistic.

Why is this important? Recall from introductory statistics that the **sampling distribution of a statistic** is the distribution of *all possible values* that the statistic could take on from *all possible samples* of a given fixed size *n*. It is an important concept to wrap your brain around, because every confidence interval and hypothesis test ever conceived is based upon the idea of the sampling distribution of a statistic. In traditional statistics, we rely on a preconceived model for specifying the shape of the population, probability theory and certain assumptions to ensure that the sampling distribution takes on a particular shape (e.g. normal).

But in many settings, we have no model for the population. We then can't appeal to probability theory, so traditional approaches get stuck. This is where bootstrapping can save you.

► **"Baby steps" example.** We may check the overall shape of the sampling distribution of $\bar{x}$ (as approximated by bootstrapping) by drawing a histogram and a normal quantile-quantile plot of the bootstrap distribution. Here it is for the "baby steps" example (I've already generated the vector of bootstrapped means, called `bootmean`):

```
> par(mfrow=c(1,2))    # sets up a plot window for two side-by-side plots
> hist(bootmean)       # make a histogram of the bootstrapped means
> qqnorm(bootmean)     # make a normal Q-Q plot of the bootstrapped means
> qqline(bootmean)     # add a reference line for assessing normality
```



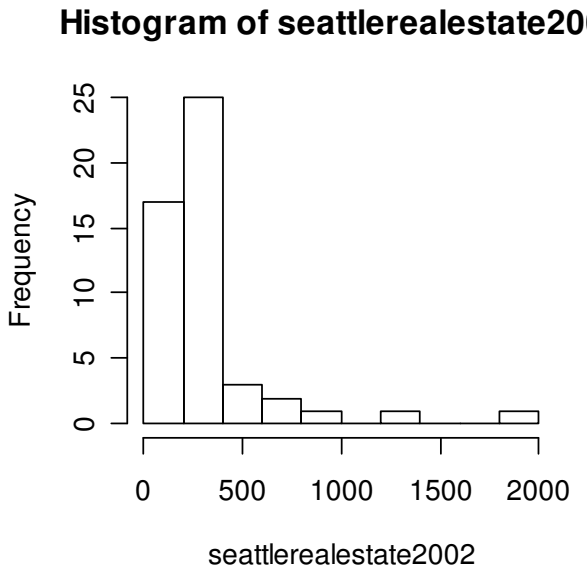
The sampling distribution appears to be left skewed. ◄

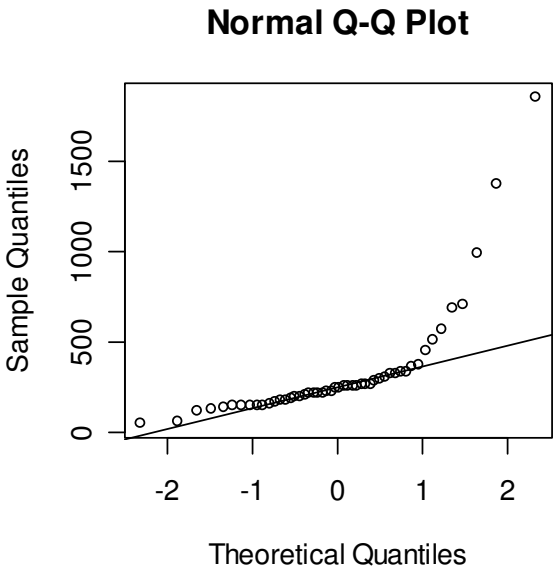
So you see that bootstrapping can be used to estimate the variability *and* the overall distribution of a statistic. This will eventually enable us to form **nonparametric confidence intervals** for population parameters. I conclude with a different example.

► **Example: Seattle real estate prices.** We are interested in the sales prices of residential properties in Seattle. Unfortunately, the data available from the county assessor's office do not distinguish residential properties from commercial properties. While most of the sales were residential, the fewer large commercial sales in the sample can greatly increase the mean selling price. The data are for 2002, and appear in the R workspace `seattlerealestate2002`. Investigate both the sample mean and the sample median as ways of characterizing "typical" sales prices.

Solution. Let's begin by looking at the data using a histogram and a normal quantitle-quantile plot. First open the workspace.

```
> par(mfrow=c(1,2))
> hist(seattlerealestate2002)
> qqnorm(seattlerealestate2002)
> qqline(seattlerealestate2002)
```





It's pretty easy to see the impact of the commercial properties: they make the sample (representative of all property sales in Seattle) highly right skewed. The normal Q-Q plot indicates strong non-normality in the data.

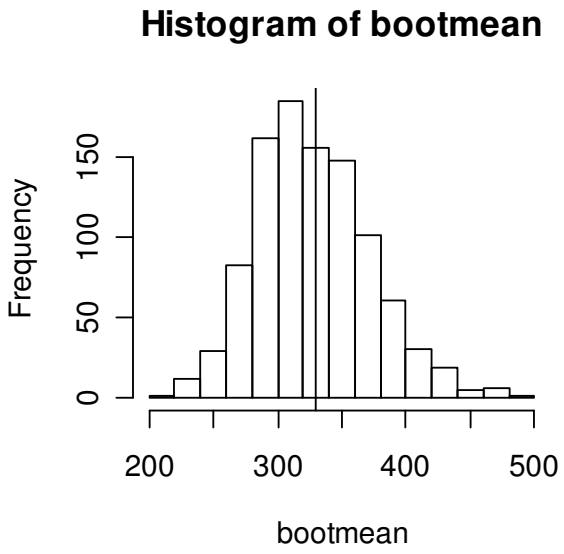
Let's collect 1000 bootstrap samples and generate the bootstrap distributions of both the sample mean and sample median. Here's an R program to do it:

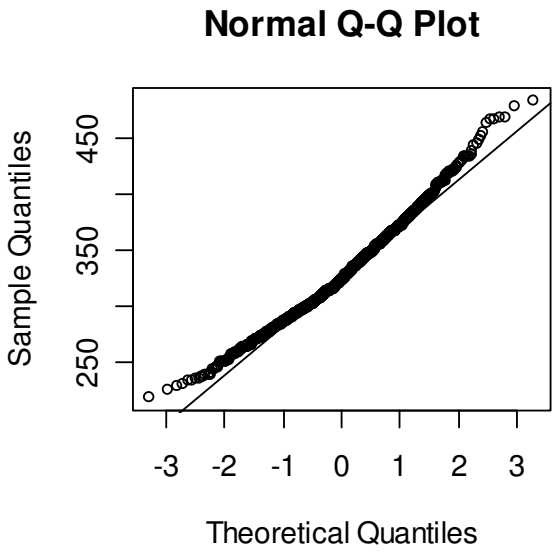
```
d <- seattlerealestate2002    # shorten the dataframe name!

R <- 1000
bootmean <- numeric(R)
bootmedian <- numeric(R)

for (i in 1:R) {
  bootsample <- sample(d, size=length(d), replace=TRUE)
  bootmean[i] <- mean(bootsample)
  bootmedian[i] <- median(bootsample)
  }

sd(bootmean)      # SE of the mean estimated by bootstrapping
sd(bootmedian)    # SE of the median estimated by bootstrapping
```

Run this and see what you get. Which statistic (the mean or median) might provide a more stable estimate of "typical" sales price?

Here is a visualization of the sampling distribution of the sample **mean** via bootstrapping:

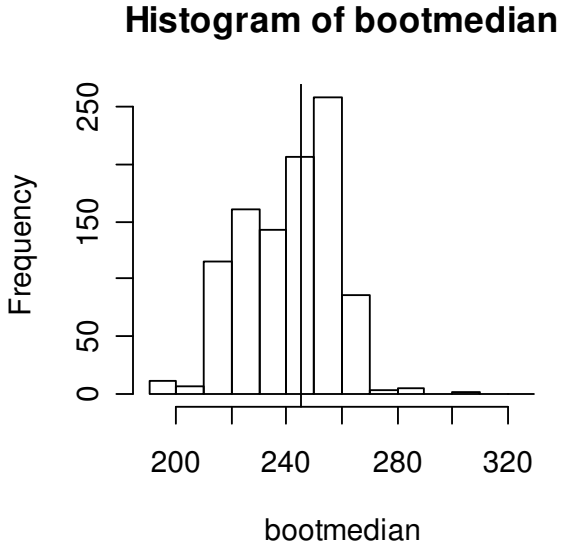
```
par(mfrow=c(1,2))     # set up for two plots side by side
hist(bootmean)        # histogram
abline(v=mean(d))     # lay in a vertical line at the actual sample mean
qqnorm(bootmean)      # create a normal Q-Q plot
qqline(bootmean)      # lay in a normal reference line
```

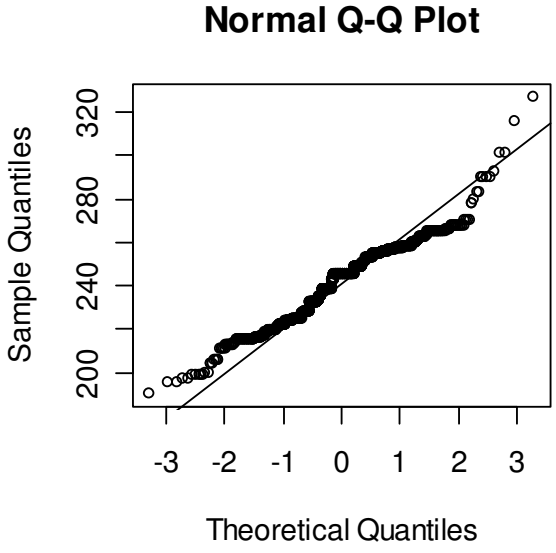




And here is the sampling distribution of the sample **median** via bootstrapping:

```
par(mfrow=c(1,2))
hist(bootmedian)
abline(v=median(d))
qqnorm(bootmedian)
qqline(bootmedian)
```





Does either statistic appear to behave normally? What about the precision of each statistic? Make some assessments about which statistic you would prefer to use to form a confidence interval for residential sales. ◄

Lecture 20 Practice Exercises

A random sample of 300 customer weekday noon-hour waiting times at a downtown Columbus Chipotle restaurant were recorded (in minutes). The data are in the R data frame `waittime.Rdata`.

1. Make a histogram of the sample and use it to describe the distributional pattern of the population of waiting times. Also, calculate and interpret the sample mean wait time $\bar{x}$.

2. Create a single bootstrap sample from the waiting times.

3. Estimate the SE and sampling distribution of the sample mean $\bar{x}$ using bootstrapping. Use 1000 bootstraps. Describe the shape of the sampling distribution of the sample mean $\bar{x}$. Is it approximately normal? Are you surprised? Why or why not?

4. With the observed sample, the traditional estimate for the SE of $\bar{x}$ is 0.05438. How does the bootstrap estimate of the SE of $\bar{x}$ compare? What might you attribute the difference to?

5. Estimate the SE and sampling distribution of the sample variance s^2 (the square of the standard deviation) using bootstrapping. Use 1000 bootstraps. Describe the shape of the sampling distribution of the sample variance. Is it approximately normal?

STA333 Lecture 21
Bootstrap confidence intervals (part 1)

21.1 Preliminaries: an introduction to the notion of bias

One of the fundamental uses of the bootstrap is to find confidence intervals for population parameters. In traditional statistics when required assumptions are met, the approach to doing this is formulaic, but if we are dealing with a situation where usual assumptions are violated and/or we are trying to estimate some atypical parameter, the bootstrap can provide us with a mechanism for constructing confidence intervals for such parameters *nonparametrically*.

We said in the last lecture that the shape of the bootstrap distribution approximates the true shape of the sampling distribution of the statistic we are using. So, for example, we can use the bootstrap distribution to check for normality. If the sampling distribution appears to be normal and centered at the true parameter, we can use the bootstrap standard error to calculate a *t*-based confidence interval. So, it turns out that we need to use the bootstrap to check the **center** of the sampling distribution as well as the shape and spread. As it turns out, the bootstrap does not reveal the center directly, but rather reveals the **bias**.

Bias. A statistic used to estimate a parameter is biased if its sampling distribution is not centered at the true value of the parameter being estimated. The bias of a statistic is found thus:

$$bias = mean\ of\ sampling\ distribution - true\ parameter\ value$$

Of course, we do not know the true parameter value, so we cannot directly calculate the bias. However, the bootstrap method allows us to check for bias by seeing whether the bootstrap distribution of a statistic is centered at the statistic's value from the original random sample. The bootstrap estimate of bias is given by

bootstrap estimate of bias = mean of bootstrap distribution – statistic for original data

► **A normal example using the sample mean.** It is a known statistical fact that the sample mean $\bar{x}$ is an unbiased estimator for the population mean μ, i.e. the true bias is 0. Let's run a little example using some randomly generated normal data to check this using bootstrapping.

The steps are:

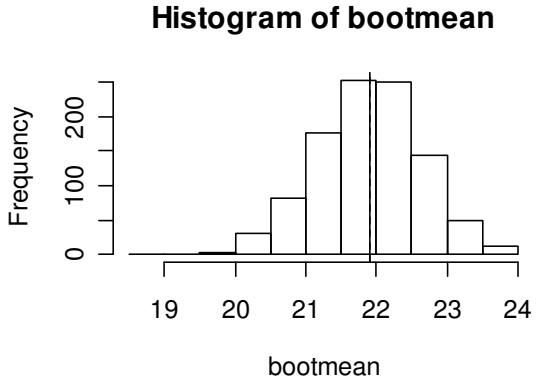
1. Randomly generate a sample of size $n = 50$ from the normal distribution with mean $\mu = 22$ and standard deviation $\sigma = 5$.
2. Calculate the sample mean $\bar{x}$ for our random sample.
3. Generate 1000 bootstrap samples from this sample, calculating the bootstrap sample mean for each.
4. From the bootstrap distribution for $\bar{x}$, calculate the estimated bias using the formula in the box on the previous page.

Here goes:

```
x <- rnorm(50,22,5)                          # step 1
xbar <- mean(x)                              # step 2

bootmean <- numeric(1000)                    # step 3
for (i in 1:1000) {
  bootsample <- sample(x, size=length(x), replace=TRUE)
  bootmean[i] <- mean(bootsample)
  }
est.bias <- mean(bootmean) - xbar            # step 4
est.bias
```

Upon running this R program, I found my estimate of bias to be 0.00497. I encourage all of you to do the same to see what results you get. Because you are resampling, you will all get different answers, but they should all be very close to 0. Here is a picture of what is happening:

```
hist(bootmean)                    # picture of the bootstrap distribution
abline(v=xbar)                    # add vertical line at the actual sample mean
abline(v=mean(bootmean), lty=2) # add dashed line mean of the bootstrap dist
```

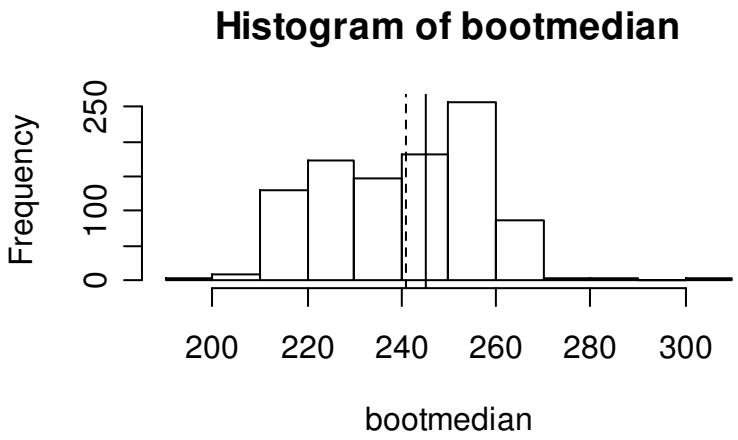


What about other kinds of statistics? Let's check for bias in the median estimate for the Seattle real estate problem from last lecture.

► **Example: Seattle real estate prices.** We are interested in the sales prices of residential properties in Seattle. Unfortunately, the data available from the county assessor's office do not distinguish residential properties from commercial properties. Because of this, we are considering the median selling price. (Remember that the data appear in the R workspace `seattlerealestate2002`.) Let's estimate the bias in using the sample median to estimate the true median selling price.

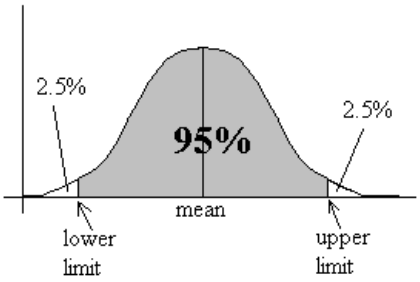
```
d <- seattlerealestate2002        # shorten the vector name!
median <- median(d)

bootmedian <- numeric(1000)
for (i in 1:1000) {
  bootsample <- sample(d, size=length(d), replace=TRUE)
  bootmedian[i] <- median(bootsample)
  }

est.bias <- mean(bootmedian) - median
est.bias                          # print the bias estimate

hist(bootmedian)                  # picture of the bootstrap distribution
abline(v=median)                  # add line at the actual sample median
abline(v=mean(bootmedian),lty=2) # add line at the mean of the bootstrap dist
```

My bias estimate was –3.9995. Upon running the program a few times, you will see that this is clearly a systematic bias in using the median. The bias is that the sampling distribution tends to underestimate the true median of the population. This is clearly shown in the picture below. Thus, a good confidence interval procedure using the bootstrap should try to compensate for the bias that we have estimated to be there. We'll see this soon. ◄



21.2 Basic bootstrap confidence intervals

In traditional (parametric) statistics, finding a CI for a mean is a snap once you have identified the sampling distribution of the statistic. Once you do that, you determine the lower and upper limits of (say) a 95% confidence interval by finding the appropriate 2.5% and 97.5% quantiles on the appropriate sampling distribution (e.g. a *t*-curve): see picture at right for an illustration.



We then calculated the CI using a formula, such as $\bar{x} \pm t_{0.025} \frac{s}{\sqrt{n}}$.

How do we find a nonparametric confidence interval? Well, the notion is essentially the same, except we use the bootstrap distribution as our estimate of the actual sampling distribution of the statistic. Once we find the 2.5% and 97.5% quantiles of the bootstrap distribution, we have essentially found a 95% confidence interval for the parameter. This type of interval is known as a **bootstrap percentile confidence interval**. It is easy to do in R using the `quantile()` function.

A little later, we will see that there are built-in routines in R for bootstrapping automatically and then calculating bootstrap CIs. We'll start, however, by building one ourselves using R code.

► **A normal example using the mean.** Let's find a 95% CI for the mean of the population using our simulated data. Here are the things we know:

1. We randomly sampled $n = 50$ observations from a normal population with mean $\mu = 22$ and standard deviation $\sigma = 5$. Thus, we know that the true mean is 22. Hopefully our CI captured this!
2. Since $\bar{x}$ is an unbiased estimator for the population mean μ, we do not need to worry about bias messing up the CI results.
3. Since the normality assumption is met and the estimator we use is unbiased, we can see how consistent the bootstrap results are with the usual parametric results you'd get using a t-based CI.

Here's the program to do it:

```
> x <- rnorm(50,22,5)
>
> bootmean <- numeric(1000)
> for (i in 1:1000) {
+   bootsample <- sample(x, size=length(x), replace=TRUE)
+   bootmean[i] <- mean(bootsample)
+   }
> quantile(bootmean, c(0.025, 0.975))

    2.5%    97.5%
19.95437 22.44248
```

I got (19.95, 22.44) as my 95% nonparametric bootstrap CI for μ. Let's compare this to the usual t-based CI from the formula $\bar{x} \pm t_{0.025} \frac{s}{\sqrt{n}}$. This is easy to get using `t.test()`:

```
> t.test(x)

        One Sample t-test

data:  x
t = 32.2378, df = 49, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 19.91887 22.56728
sample estimates:
mean of x
 21.24308
```

Buried in this R output is the 95% parametric CI result: (19.92, 22.57). We see that the parametric and nonparametric results are very consistent. ◄

► **Example: Seattle real estate prices.** We now find a 95% *bootstrap percentile confidence interval* for the median of Seattle real estate selling prices in 2002.

```
> d <- seattlerealestate2002        # shorten the vector name!
>
> bootmedian <- numeric(1000)
> for (i in 1:1000) {
+   bootsample <- sample(d, size=length(d), replace=TRUE)
+   bootmedian[i] <- median(bootsample)
+   }
> quantile(bootmedian, c(0.025, 0.975))
   2.5%   97.5%
213.975 266.000
```

Using this method, we can be 95% confident that the true median selling price of residential properties in Seattle in 2002 was between $213,975 and $266,000. ◄

A few notes:

1. Try finding a 95% CI for the mean using both a *t*-based approach and the bootstrap approach for the last example. Compare the results and comment on what you see.
2. We can even improve on the interval result above. How? Remember that the median was biased. Our method (percentile-based bootstrap) does not account for this bias. Next time we will see an improved method using bootstrapping that does account for the estimated bias
3. The R add-on package `boot` handles many kinds of bootstrap problems and calculates bootstrap CIs automatically. On the next page I will re-do the Seattle real estate problem using the `boot` package.

► **Example: Seattle real estate prices.** I re-do the problems in this section but using the R add-on package `boot`. The code below will be discussed in class.

```
> d <- seattlerealestate2002        # shorten the vector name!
>
> library(boot)
> mymedian <- function(d,i)  median(d[i])
> myboot <- boot(d, mymedian, R=1000)
> myboot

ORDINARY NONPARAMETRIC BOOTSTRAP

Call:
boot(data = d, statistic = mymedian, R = 1000)

Bootstrap Statistics :
    original    bias    std. error
t1*  244.925 -4.304875    16.26544
```

The original sample median is 244.925, the estimated bias is –4.305, and the SE of the bootstrap distribution is 16.27.

We now ask for the 95% percentile-based bootstrap confidence interval for the median. This is done via the `boot.ci()` function in the `boot` package:

```
> boot.ci(myboot, type="perc")

BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1000 bootstrap replicates

CALL :
boot.ci(boot.out = myboot, type = "perc")

Intervals :
Level     Percentile
95%   (213.2, 266.0 )
Calculations and Intervals on Original Scale
```

The 95% confidence interval is found to be (213.2, 266.0). ◄

We will see more of this next time, along with several different problems in varying contexts and by using several different statistics. In the meantime, do the following:

1. Check the help page by typing `?boot` in R.
2. Visit http://www.statmethods.net/advstats/bootstrapping.html on the Quick-R website.
3. Visit http://www.mayin.org/ajayshah/KB/R/documents/boot.html for another writeup on the basics of the `boot` package.

Lecture 21 Practice Exercises

A random sample of 300 customer weekday noon-hour waiting times at a downtown Columbus Chipotle restaurant were recorded (in minutes). The data are in the R data frame `waittime.Rdata`.

1. Find and interpret a 95% bootstrap confidence interval for the true mean weekday noon-hour waiting times at the downtown Columbus Chipotle restaurant.

2. Repeat problem 1, but using the R add-on package `boot`.

3. Recall from the Lecture 20 practice exercises that the sampling distribution of the sample mean (via bootstrapping) appeared fairly normal, except for a bit of discrepancy in the tails of the distribution. Due to this, go ahead and find the usual 95% *t*-based CI for the true mean weekday noon-hour waiting times at the downtown Columbus Chipotle restaurant. How does the traditional approach result vary from the bootstrap CI from problem 1?

4. Find and interpret a 95% bootstrap confidence interval for the true median weekday noon-hour waiting times at a downtown Columbus Chipotle restaurant. Estimate the bias and comment. How does this CI differ from the CI you got for the mean in problem 1? Can you explain why the difference exists?

STA333 Lecture 22
Bootstrap confidence intervals (part 2)

22.1 Better bootstrap CIs: the bias-corrected accelerated (BCa) interval

No method for obtaining CIs exactly the intended confidence level in practice. When you compute what is supposed to be, say, a 90% CI, the method you use may in fact give intervals that capture the true parameter value less often, say 87% of the time. Or instead of "missing" 5% of the time on each side, a method may in some settings miss 3% of the time on one side and 7% of the time on the other, giving a biased picture of where the true parameter is.

Accuracy. We say that a method for obtaining a 90% CI is **accurate** in a particular setting when 90% of the time it produces an interval that covers the true parameter value, and produces intervals that "miss" the true parameter value 5% of the time on the high side and 5% of the time on the low side. No CIs are perfectly accurate in practice because the conditions under which they work are never perfectly satisfied. The common culprits resulting in inaccurate CIs are:

1. sampling from a highly skewed population; or,
2. using a statistic that is a biased estimator of its corresponding population parameter.

One advantage of bootstrapping is that it allows you to check for skewness in the sampling distribution. When this skewness is present, it can produce a **CI bias** in the direction of the skew. So, a percentile-based bootstrap CI may not be accurate enough if:

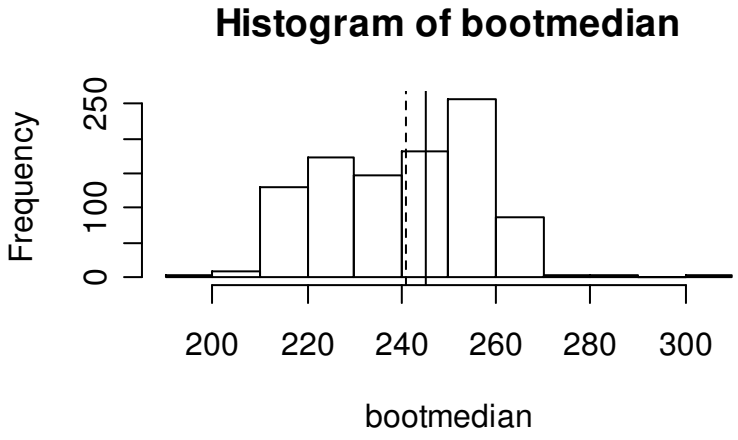
- The statistic itself is strongly biased, as indicated by the bias estimate from the bootstrap,
- The sampling distribution of the statistic is clearly skewed, as indicated by the bootstrap distribution itself, or
- High accuracy is needed because the stakes are high (e.g. large amounts of money, public welfare, etc.)

The BCa interval

The BCa, or *bias-corrected and accelerated bootstrap* confidence interval, is a modification of the percentile-based bootstrap confidence interval. **BCa CI endpoints are percentiles of the bootstrap distribution that are adjusted to correct for bias and skewness in the distribution.** For example, if the statistic is biased upward (that is, it tends to be too large), the BCa bias correction shifts the endpoints to the left. If the bootstrap distribution is skewed to the right, the BCa incorporates a correction to shift the endpoints even farther to the right (this may seem counterintuitive, but it is the correct action).

Details of these computations are pretty advanced stuff, so we will rely on software to calculate BCa intervals. In R, BCa intervals may be found using the `boot.ci()` function in the `boot` package. Ask for `method="bca"` rather than `method="perc"`.

► **Example: Seattle real estate prices.** We earlier saw (and estimated) the downward bias in the median for these data. Remember this? –




```
ORDINARY NONPARAMETRIC BOOTSTRAP

Call:
boot(data = d, statistic = mymedian, R = 1000)

Bootstrap Statistics :
    original    bias    std. error
t1*  244.925 -4.304875    16.26544
```

The estimated bias was –4.305. Because of this, a 95% percentile-based bootstrap confidence interval for the median may not be accurate. So, we instead ask for the BCa interval. Just for kicks, I'll ask for both types from R:

```
> boot.ci(myboot, type=c("perc", "bca"))


BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1000 bootstrap replicates
```

```
CALL :
boot.ci(boot.out = myboot, type = c("perc", "bca"))

Intervals :
Level     Percentile            BCa
95%   (213.2, 266.0 )   (213.0, 265.5 )
Calculations and Intervals on Original Scale
```

The 95% BCa bootstrap interval is (213.0, 265.5). The lack of a noticeable difference between the percentile-based CI and the BCa CI actually suggests that the bias does not have an impact on the results. ◄

22.2 Parting comments on the bootstrap

Some comments are in order before we look at several examples using bootstrapping in the next section. Here goes:

1. Bootstrapping and conclusions based on them include *two* sources of random variation:
   - The original sample was chosen at random from the population.
   - Bootstrap samples are chosen at random from the original sample.

2. For most statistics, **almost all of the variation in bootstrap distributions comes from the selection of the original sample**, *not* from what bootstrap samples you happen to select. A bootstrap resampling process using 1000 or more resamples introduces very little additional variation to the outcome.

3. While bootstrapping allows us to relax assumptions like normality, **you must still respect the structure of your data**; any form of dependence in the data must be taken into account. For instance, how you draw bootstrap samples for a comparison of means using independent samples must be done differently than if you had paired samples.

4. **Bootstrapping is *not* a cure for tiny sample sizes!** If your sample size n is small, there is no getting around the fact that you do not have much information from the population from which to generate an accurate or usable inference. *This is always true, whether you are bootstrapping or not!* As always, bigger samples yield more trustworthy results.

5. **Bootstrapping does not work on <u>all</u> kinds of statistics.** In particular, I recommend against using the bootstrap for statistics that are largely functions of the sample extremes: for example, the minimum value, the maximum value, the range, etc. Bootstrapping works best on statistics that are functions of the *entire* sample: this includes means, standard deviations, correlations, regression-based quantities such as model coefficients and predictions, etc. It also works very well for medians and quartiles, though usually larger samples are desirable for bootstrapping these statistics.

22.3 Examples

Now, several examples using bootstrapping in different context.

► **Example: Diet comparisons.** These data were first presented in Lecture 11. In a comparison of the effect on growth of two diets ***A*** and ***B***, a number of growing rats were placed on these two diets, and the following growth figures were observed after 7 weeks:

A	156	183	120	113	138	145	142			
B	109	107	119	162	121	123	76	111	130	115

Earlier, we did hypothesis tests of H_0: $\mu_A = \mu_B$ versus H_a: $\mu_A \neq \mu_B$ using both a parametric independent samples *t*-test and a nonparametric permutation test. Now, let's use bootstrapping to estimate, with 90% confidence, the *true mean difference in growth* between diets ***A*** and ***B***.

Solution. To do this, we can enter the samples into two separate R vectors, bootstrap each one, and then calculate the difference between the means of the bootstrapped samples. **We bootstrap each sample independently because the samples are independent** (see Comment 3 in the previous section).

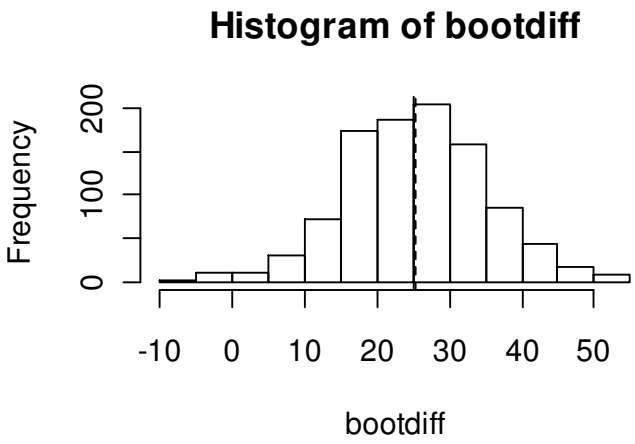
The true mean difference in growth is $\mu_A - \mu_B$. So, we use $\bar{x}_A - \bar{x}_B$ as our statistic. The steps are:

1. Collect a bootstrap sample from the diet ***A*** sample. Calculate $\bar{x}_A$ from this.
2. Collect a bootstrap sample from the diet ***B*** sample. Calculate $\bar{x}_B$ from this.
3. Calculate $\bar{x}_A - \bar{x}_B$. This is a bootstrapped mean difference.
4. Repeat steps 1-3 a very large number of times (say 1000). This produces our bootstrap distribution for $\bar{x}_A - \bar{x}_B$.
5. Form a CI for $\mu_A - \mu_B$ from the bootstrap distribution.

Here is an R program to do the bootstrapping, including a check of bias:

```
A <- c(156,183,120,113,138,145,142)
B <- c(109,107,119,162,121,123,76,111,130,115)
obs.diff <- mean(A) - mean(B)

bootdiff <- numeric(1000)

for (i in 1:1000) {
  bootsampleA <- sample(A, size=length(A), replace=TRUE)
  bootsampleB <- sample(B, size=length(B), replace=TRUE)
  bootdiff[i] <- mean(bootsampleA) - mean(bootsampleB)
  }
est.bias <- mean(bootdiff) - obs.diff
est.bias

hist(bootdiff)                  # picture of the bootstrap distribution
abline(v=obs.diff)              # add vertical line at the actual sample mean diff
abline(v=mean(bootdiff),lty=2)  # add dashed line mean of the bootstrap dist
```

Upon running, I got a bias estimate very close to zero. The plot of the bootstrap distribution reveals that bias is not an important issue here:

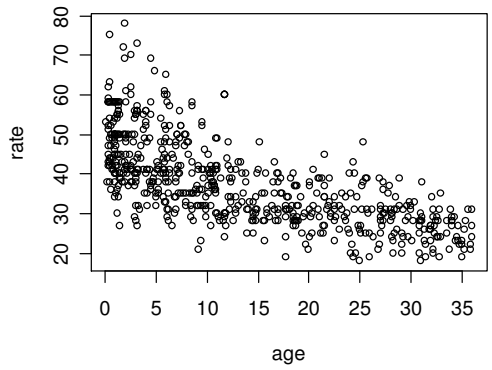


Now, I calculate a 90% percentile-based bootstrap CI for $\mu_A - \mu_B$:

```
> quantile(bootdiff, c(0.05, 0.95))
       5%       95%
 9.826429 41.687857
```

We can be 90% confident that the true mean growth due to diet ***A*** is between 9.82 to 41.68 higher than for diet ***B***. ◄

► **Example: Respiratory rates in children.** A high respiratory rate is a potential diagnostic indicator of respiratory infection in children. To judge whether a respiratory rate is truly "high," however, a physician must have a clear picture of the distribution of normal respiratory rates. To this end, Italian researchers measured the respiratory rates of $n = 618$ children between the ages of 15 days and 3 years (given in months). The data appear in the R workspace `respiratory`. Here is a plot of the data:



Find a 95% bootstrap confidence interval for the true rank (Spearman) correlation relating age to respiratory rate.

Solution. To do this, we must perform the bootstrapping by resampling the children. In this context, that means **we must bootstrap entire rows of the R dataframe**. This will retain the connection between each child's age and their respiratory rate.

For this example, I will demonstrate by using the `boot` package in R. After opening the dataframe in R, run the following program. The code will be explained in class, but brief comments are included in-line below.

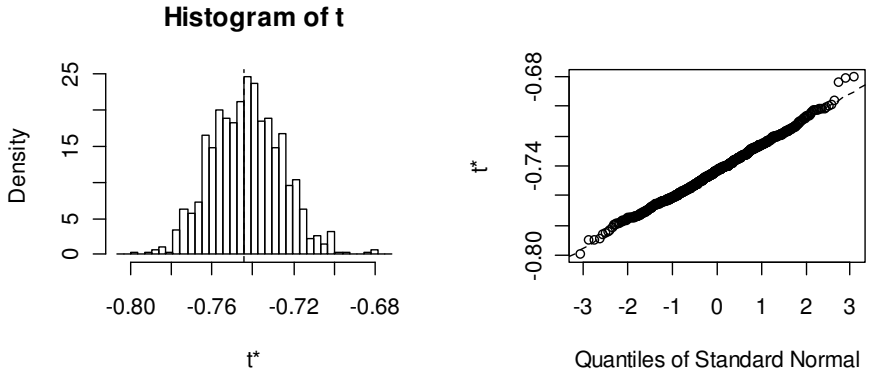
```
library(boot)

# First, create a function to obtain Spearman correlations from the data:
mycorr <- function(data, indices) {
               d <- data[indices,]
               return(cor(d$age, d$rate, method="spearman"))
             }

# Next, run the bootstrapping using 1000 replications:
results <- boot(data=respiratory, statistic=mycorr, R=1000)

# Finally, view results of bootstrapping:
results
```

Here's the output for the code above:

```
ORDINARY NONPARAMETRIC BOOTSTRAP

Call:
boot(data = respiratory, statistic = mycorr, R = 1000)

Bootstrap Statistics :
      original      bias    std. error
t1* -0.7445018 0.001416301  0.01725391
```

The Spearman correlation between rate and age is -0.7445. The standard error of the correlation estimate is 0.0173, and the bias seems negligible. We can plot the bootstrap distribution using the `plot()` function applied to the `boot` object that we named `results`:

```
plot(results)
```



*t** is a generic name that `boot` gives to the bootstrapped statistic. In this problem, the bootstrap distribution actually appears reasonably normal! Now, the CI:

```
> boot.ci(results, type=c("perc","bca"))

BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1000 bootstrap replicates

CALL :
boot.ci(boot.out = results, type = c("perc", "bca"))

Intervals :
Level     Percentile            BCa
95%   (-0.7745, -0.7079 )   (-0.7766, -0.7115 )
Calculations and Intervals on Original Scale
```

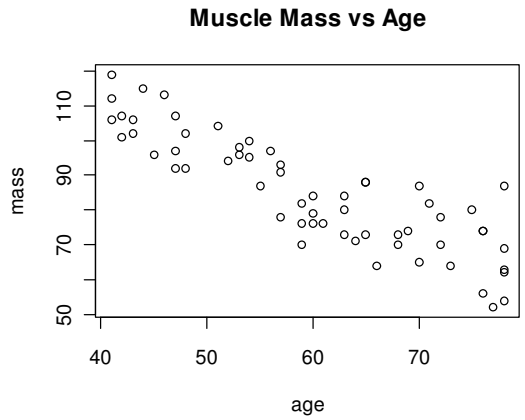
We can be 95% confident that the true rank correlation between age and respiratory rate for children aged 15 days and 3 years is between –0.77 and –0.71. Thus, there is statistical evidence of a quite strong negative association between age and respiratory rate. Due to the symmetry and lack of bias, either bootstrap CI is fine to use here. ◄

► **Example: Muscle mass.** A person's muscle mass is expected to decrease with age. To explore this relationship in women, a nutritionist randomly selected 15 women from each 10-year age group beginning with age 40 and ending with age 79. The data reside in the R workspace `musclemass`. The variables in the dataset are `mass` and `age`. Let's do the following:

1. Investigate the relationship visually using a scatterplot.
2. Fit a simple linear regression model relating muscle mass to age.
3. Find a 95% bootstrap CI for the true mean rate of change in muscle mass for each additional year of age. (In other words, find a 95% bootstrap CI for the regression slope.)

Solution. First, here's the scatterplot:

```
> plot(mass~age, data=musclemass, main="Muscle Mass vs Age")
```



We can clearly see the negative trend one would expect: as age increases, muscle mass tends to decrease. You should also notice that it decreases in roughly a linear fashion, so it is reasonable to fit a simple linear regression model to this data. (However, this point is arguable … we may investigate this in the next lecture!)

We now fit the model in R. To do so, the `lm()` function may be used, followed by a `summary()` function call to see results of the regression fit:

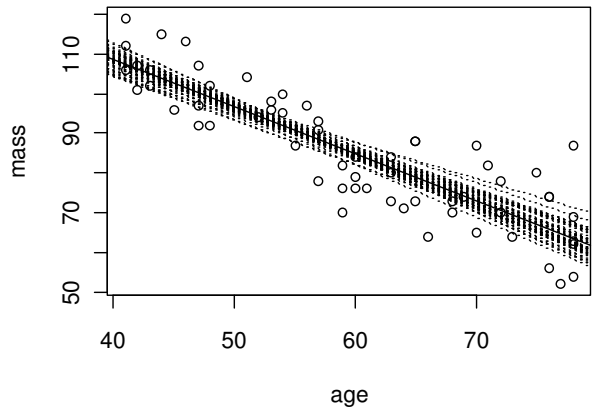
```
> fit <- lm(mass~age, data=musclemass)
> summary(fit)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 156.3466     5.5123   28.36   <2e-16 ***
age          -1.1900     0.0902  -13.19   <2e-16 ***
```

The fitted regression model is $\widehat{mass} = 156.345 - 1.19(age)$. The slope estimate is $b_1 = -1.19$. This means that we estimate a drop of 1.19 in mean muscle mass per year of age in women aged 40-79.

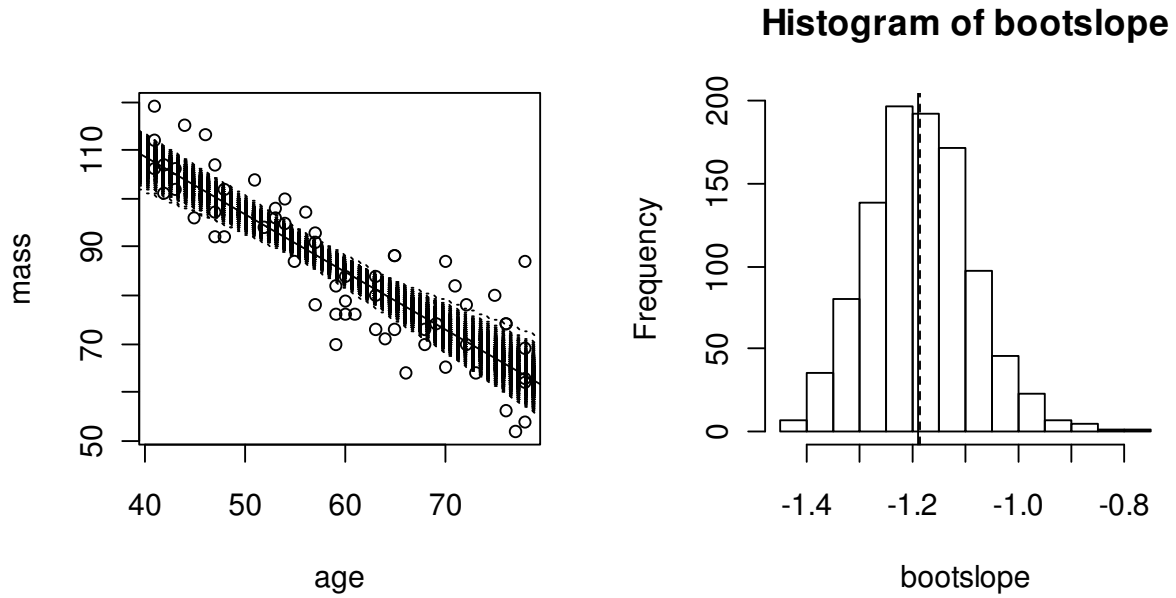
What we want now is a 95% confidence interval for the true mean rate of change in muscle mass for each additional year of age. There is a parametric way of doing this using a *t*-based CI, but if the usual regression assumptions aren't met, a bootstrap CI may be used instead. This approach is detailed below. The following R program was built by yours truly from the ground up. I will explain it in class in detail, but one of the nice things it does is show you the bootstrap distribution of the slopes visually by plotting all the bootstrapped slopes over the scatterplot. (Kinda cool, if you ask me.)

```
## A sexy nonparametric bootstrap program for the slope in a simple linear regression.
## This example uses the R dataframe "musclemass".

par(mfrow=c(1,2))                # set up two panels for our plots
bootslope <- numeric(1000)       # create vector to hold 1000 bootstrapped slopes

plot(mass~age, data=musclemass)        # make scatterplot of the data points
fit <- lm(mass~age, data=musclemass)   # fit the simple linear regression to orig data
summary(fit)                           # print a summary of fitted model results
slope <- summary(fit)$coeff[2]         # extract the slope estimate
abline(fit)                            # draw in the fitted line

for (i in 1:1000) {
  n <- nrow(musclemass)
  index <- sample(1:n, size=n, replace=TRUE)
  bootdata <- data.frame(mass=musclemass$mass[index],age=musclemass$age[index])
  bootfit <- lm(mass~age, data=bootdata)
  bootslope[i] <- summary(bootfit)$coeff[2]     # extract bootstrapped slope
  abline(bootfit, lty=2)                        # draw in bootstrapped slope on plot
   }

hist(bootslope)                          # picture of the bootstrap dist of slopes
abline(v=slope)                          # add vertical line at the actual sample slope
abline(v=mean(bootslope),lty=2)          # add dashed line at mean of the bootstrap dist
quantile(bootslope, c(0.025, 0.975))     # percentile-based 95% bootstrap CI for slope
```

We'll run this in class (and do so yourself, too). For the sake of illustration, here is the scatterplot with just 50 bootstrapped slopes superimposed. This provides you with a sense of the variation you might expect in the slope if you had collected a different random sample (of the same size) of women:



A full run of the program using 1000 bootstraps gives this:



There appears to be a very minor right skew in the distribution of slope estimates (can you look at the scatterplot and guess why?). The percentile-based bootstrap CI is given below:

```
> quantile(bootslope, c(0.025, 0.975))

      2.5%      97.5%
-1.3698876 -0.9774157
```

The 95% CI is (–1.37, –0.97). We can be 95% confident that each additional year of age will result in a mean decrease in muscle mass of between 0.97 to 1.37. This CI is entirely below 0, so we are confident that muscle mass and age are negatively linearly related in women aged 40-79.

Side note. The usual parametric *t*-based CI for the regression slope can be found by applying the `confint()` function in R to the original fitted model from `lm()`:

```
> confint(fit)
                 2.5 %     97.5 %
(Intercept) 145.312572 167.380556
age          -1.370545  -1.009446
```

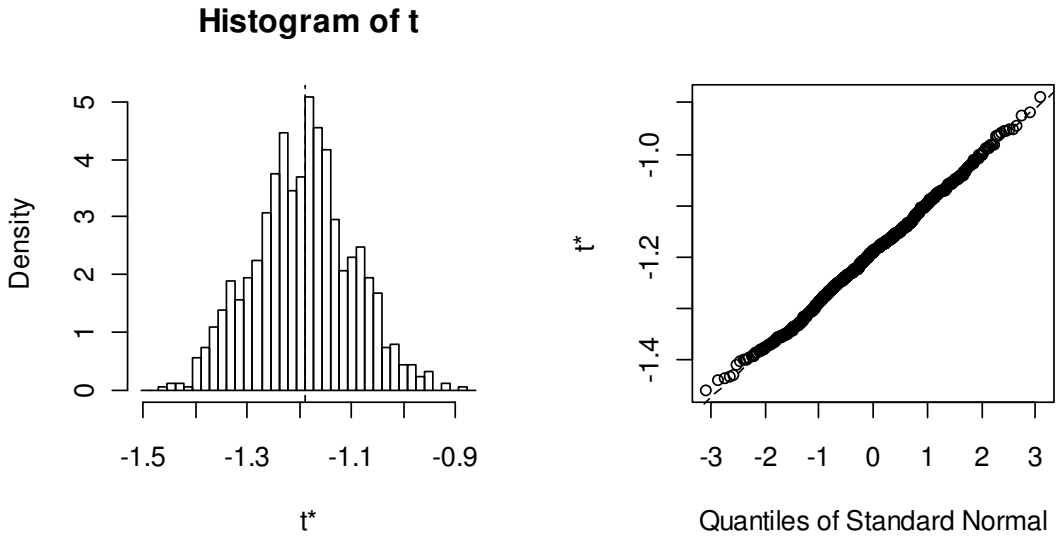
Note how close the parametric CI is to our bootstrap CI. This is because the standard regression assumptions were reasonably met in this case. ◄

► **Example: Muscle mass (again).** Here's the same problem I just did, but now done using the `boot` package in R:

```
library(boot)                                   # load the 'boot' package

# First, create a function to obtain the slope from the data:
myslopes <- function(data, indices) {
               d <- data[indices,]              # tells 'boot' to resample ROWS of data
               fit <- lm(mass~age, data=d)      # fit regression using bootstrap dataset
               return(summary(fit)$coeff[2])    # returns a bootstrapped slope estimate
}

# Next, run the bootstrapping using 1000 replications:
results <- boot(data=musclemass, statistic=myslopes, R=1000)

# Finally, view results of bootstrapping:
results
plot(results)

# ... and obtain 95% bootstrap CIs (both percentile and BCa):
boot.ci(results, type=c("perc","bca"))
```

Here are the results. Compare these to what I got above.

```
ORDINARY NONPARAMETRIC BOOTSTRAP

Call:
boot(data = musclemass, statistic = myslopes, R = 1000)

Bootstrap Statistics :
     original       bias    std. error
t1* -1.189996 -0.002642710  0.09428257
```




```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1000 bootstrap replicates

CALL :
boot.ci(boot.out = results, type = c("perc", "bca"))

Intervals :
Level     Percentile            BCa
95%   (-1.374, -1.001 )   (-1.370, -0.996 )
Calculations and Intervals on Original Scale
```

All methods (parametric *t*-based CI, percentile-based bootstrap CI, BCa bootstrap CI) produce highly consistent results here and lead to the same conclusions. ◄

Lecture 22 Practice Exercises

These practice problems use data sets from the examples in Lecture Notes section 22.3.

1. Use the diet comparisons data to calculate and interpret a 90% bootstrap confidence interval for σ_1/σ_2, the ratio of the population standard deviations between diets *A* and *B*. Use the ratio of the sample standard deviations s_1/s_2 as your bootstrapping statistic.

2. Use the muscle mass data to calculate and interpret a 99% bootstrap CI for the Pearson correlation. Use the R add-on package `boot` to do the problem. What is your bias estimate? Should you use a percentile-based bootstrap CI or a BCa bootstrap interval?

3. *Challenge problem:* Use the muscle mass data to calculate and interpret a 95% bootstrap CI for the true mean muscle mass for a 60-year old woman. (*Hint:* this prediction is found via the fitted regression equation $\hat{y} = b_0 + b_1(60)$. Look into the `predict()` function in R.)