Week 1

Santiago Barreda

Contents

1 Inspecting a single sample of values

1.1 Data and research questions

Voice fundamental frequency (related to pitch) is a very important cue in speech communications. It relates to phoneme identification, prosody, and to the communication of social and indexical information (talker gender, age, . . . etc.).

We are going use a well-known data set, the Hillenbrand et al. (1995) data of Michigan English. We are going to focus on a single vector (w$f0) representing the f0 produced by a set of female talkers. The variable `uspeaker` is a speaker number that is unique across all speaker groups, and `type` indicates speaker group from among `b` (boys), `g` (girls), `m` (men), and `w` (women).

```
url1 = "https://raw.githubusercontent.com/santiagobarreda"
url2 = "/stats-class/master/data/h95_vowel_data.csv"
h95 = read.csv (url(paste0 (url1, url2)))

# select women only
w = h95[h95$type == 'w',]
head (w)
```

```
##     file duration  f0  f1   f2   f3   f4 f1_2 f2_2 f3_2 f1_5 f2_5 f3_5 f1_8 f2_8 f3_8 type speaker
## 46 w01ae      305 225 678 2293 2861 4412  681 2295 2868  711 2160 2867  705 1968 2906    w       1
## 47 w02ae      486 214 624 2442 3091 5306  621 2470 3042  755 2237 3033  712 2000 3069    w       2
## 48 w03ae      293 192 666 2370 2814 3706  701 2328 2768  799 2074 2595  925 1983 2695    w       3
## 49 w04ae      353 233 743 2230 3055 4476  759 2193 3073  812 1948 2951  805 1923 3119    w       4
## 50 w05ae      338 223 677 2320 2987 5230  678 2263 2987  804 2056 2808  806 1988 2810    w       5
## 51 w06ae      362 223 627 2266 2875 4085  628 2224 2918  730 1983 2791  763 1788 2760    w       6
##    vowel uspeaker
## 46    ae       92
## 47    ae       93
## 48    ae       94
## 49    ae       95
## 50    ae       96
## 51    ae       97
```

```
# this turns the subject numbers into factors
w$speaker = factor (w$speaker)
# this is unique subject numbers across all groups
w$uspeaker = factor (w$uspeaker)
# select only the vector of interest
f0s = w$f0
```

These talkers represent a sample from a larger population. The sample is a finite set of observations that you actually have. The population is the larger group of possible observations that you are *actually* interested in. For example, Hillenbrand et al. collected this data not to study these talkers in particular, but instead to make inferences about Michigan speakers more generally.

Similarly, we want to answer a few basic questions about the population of female talkers from Michigan, not about the sample itself:

1) What is the average f0 of the whole *population* likely to be?
2) Can we set bounds on likely mean f0 values based on the data we collected?

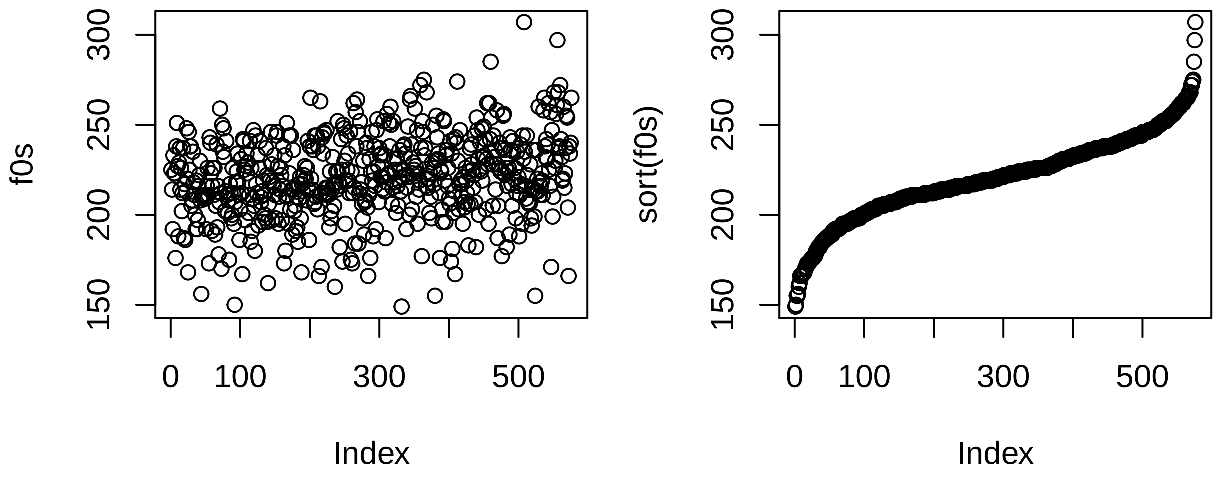
The third point is crucial. First, our sample will never exactly match the population. But it should be *representative* of it, meaning it should not be *too* far off from the real mean (otherwise, it is likely not a good sample from that population). For example, the mean f0 in the data we will discuss below is 220 Hz. This seems to suggest that, for example, a *true* mean of 100 Hz is unlikely. Is a true mean of 150 Hz also unlikely? What about 190 Hz?

Below we will discuss how to measure the spread and location of a set of numbers, and how to establish likely, credible values for the variable.

1.1.1 Inspecting the central location and spread of values

Plotting the values does not really make a very nice plot by default, but it does help to see what the data looks like in general. We can also plot the observations sorted by increasing value.

```r
par (mfrow = c(1,2))
plot (f0s)
plot ( sort ( f0s ) )
```



We can easily find descriptive statistics like the mean, the standard deviation, and important quantiles. The quantiles below correspond to the values of ordered observations, like in the right plot above. The 0% quantile is the smallest observation, while 100% is the highest. Any other quantile is found by ordering the observations and selecting the observation that is higher than x% of the sample values. For example, the 50% quantile (the median) is higher than 50% of values, and the 25% quantile is higher than 1/4 of the values in the sample.

```r
mean (f0s)
```

```
## [1] 220.401
```

```r
sd (f0s)
```

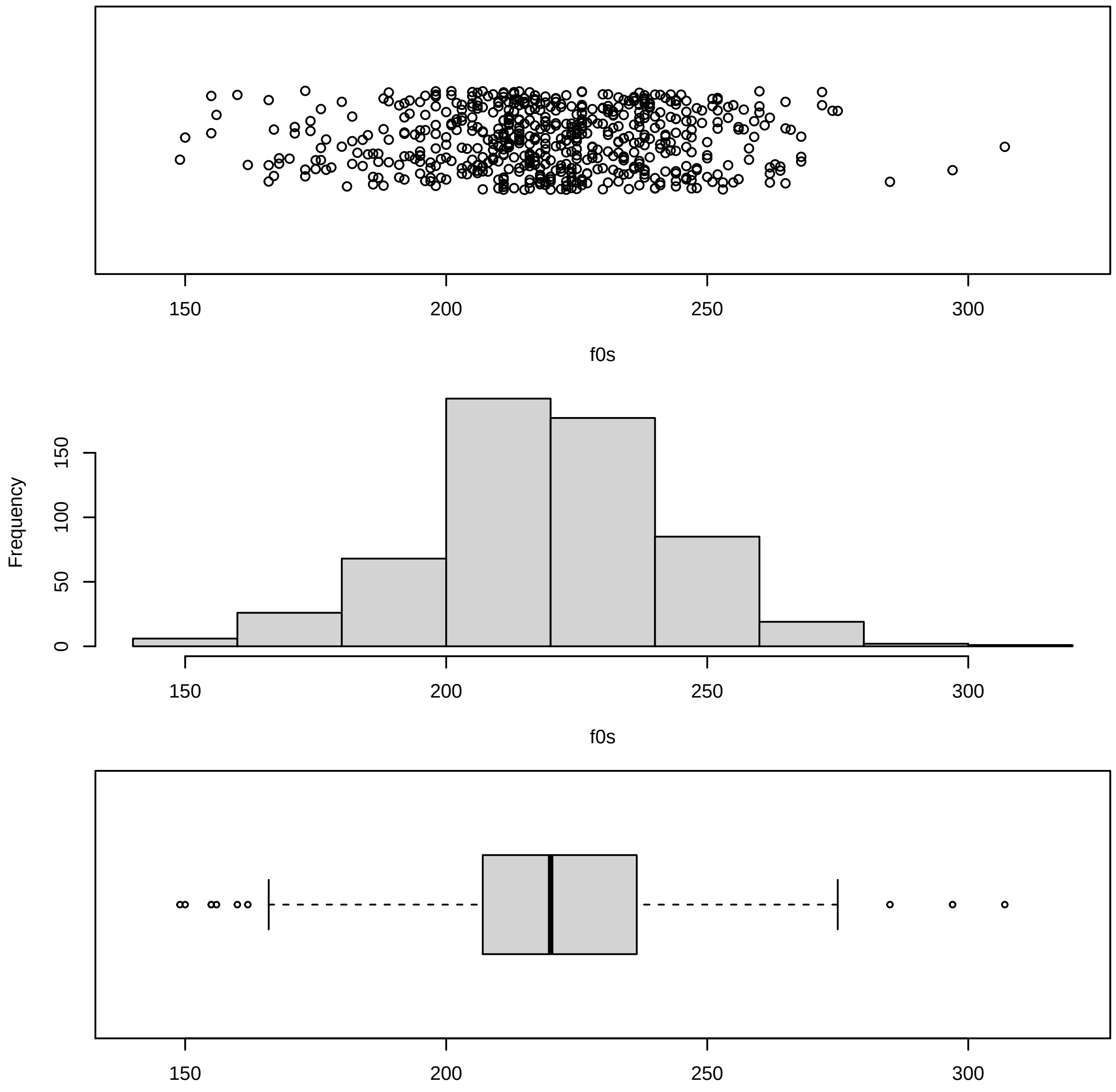
```
## [1] 23.22069
```

```r
quantile (f0s)
```

```
##     0%    25%    50%    75%   100%
## 149.00 207.00 220.00 236.25 307.00
```

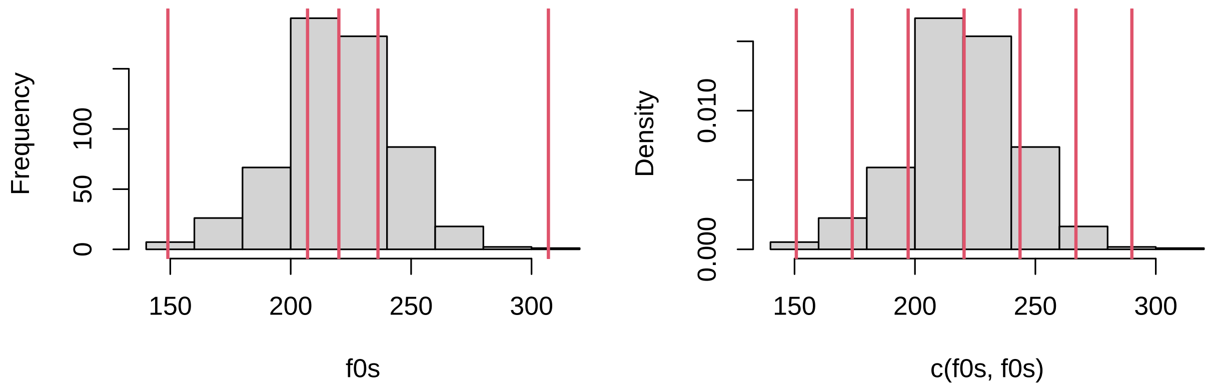
We can look at the distribution of talkers. In the top row, points indicate individual productions, and are jittered along the y axis to make them easier to see. In the middle row we see a histogram of the same data. The histogram gives you the count of observations in each bin. In the bottom row we see a box plot of the same data. The edges of the box correspond to the 25 and 75% quantiles of the distribution, and the line in the middle of it corresponds to the median. As a result, 50% of observations are contained in the box.

```r
par (mfrow = c(3,1))

plot (f0s, jitter (rep(1,nrow(w))), xlim = c(140, 320), ylim = c(.95,1.05),yaxt='n',ylab='')
hist (f0s,main="")
boxplot (f0s, horizontal = TRUE, ylim = c(140, 320))
```



The summary statistics basically tell you about the shape of your distribution, given some assumptions. The left panel shows the locations of quantiles (0%, 25%, 50%, 75%, 100%), the right panel shows you the mean and standard deviations from the mean (-3, -2, 0, +1, +2, +3). Notice that ±3 standard deviations enclose the entire distribution.

```r
par (mfrow = c(1,2))

hist (f0s, main = "")
abline (v = quantile (f0s), lwd = 2, col = 2)
hist (c(f0s,f0s), freq = FALSE, breaks = 10, main = "")
abline (v = seq (mean(f0s)-3*sd(f0s),mean(f0s)+3*sd(f0s),sd(f0s)), lwd = 2, col = 2)
```



1.1.2 Probability Distributions

Histograms are particularly useful to understand because of how they relate to probability distributions. For our purposes, the probability is the number of times an event is expected to occur, out of all the other observed events and outcomes. This can also be thought of as the *percent* of times an event is expected to occur. The total probability of all events is always equal to 1. This is like using 100 to communicate percent. Its just easier that way. As a result of this convention, you know that a probability of 0.5 means something is expected to occur half the time (i.e., on 50% of trials). For example, suppose we want to know the probability of being an adult female in our sample who produces an f0 under 175 Hz. Finding the probability of observing this event is easy:

```r
sum (f0s < 175)   ## number of observations the fall below threshold
```

```
## [1] 22
```

```r
sum (f0s < 175) / length (f0s)   ## divided by total number of events
```

```
## [1] 0.03819444
```

```r
mean (f0s < 175) ## another way to calculate probability
```

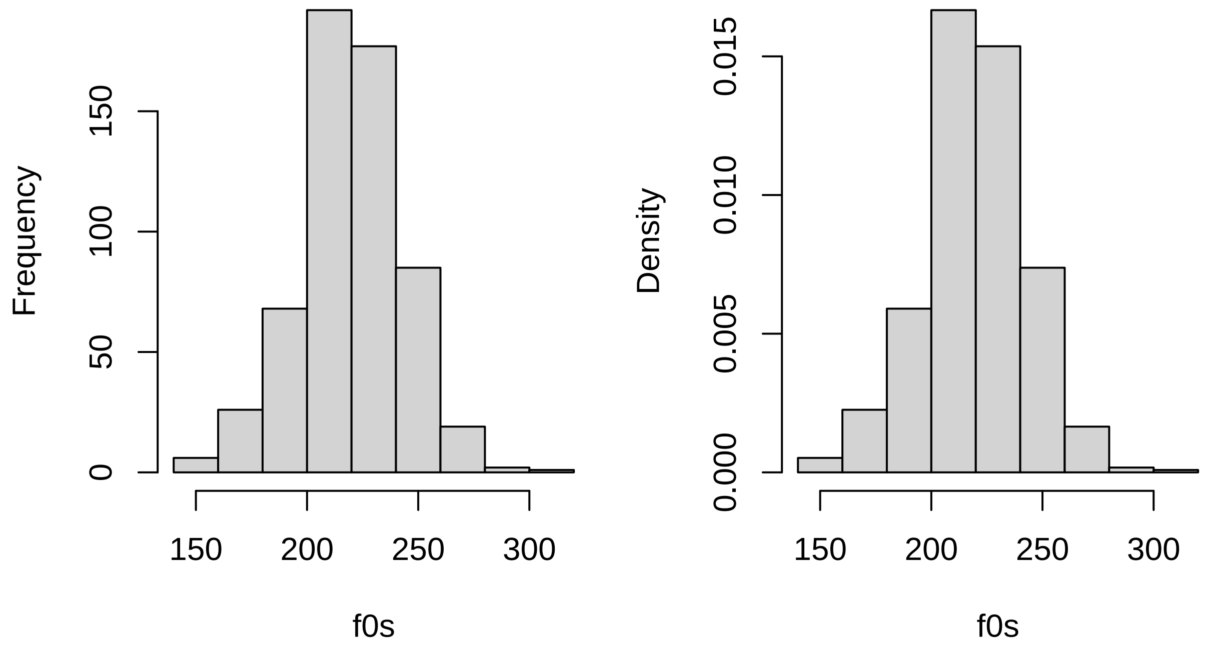
```
## [1] 0.03819444
```

The top value is the frequency of the occurrence. This is not so useful because this number takes can mean different things given different sample sizes (e.g., 22/23, 22/10000). The middle and bottom values have been divided by the total number of observations. As a result, these now represent a proportion, or probability.

Histograms can also show this difference between total counts and probabilities. Below, the histogram on the left shows the number of observations in each bin. The histogram on the right shows *density* on the y axis. When you see *density* on the y axis, that means that y axis values have been scaled to make the area under the curve equal to 1. This has two benefits:

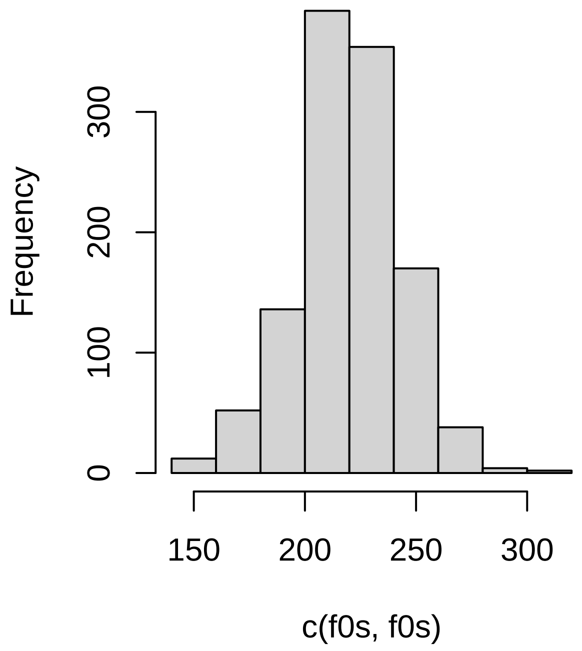
1) It lets you compare the distribution of values across different sample sizes.
2) It makes the histrogram more comparable to a probability distribution.

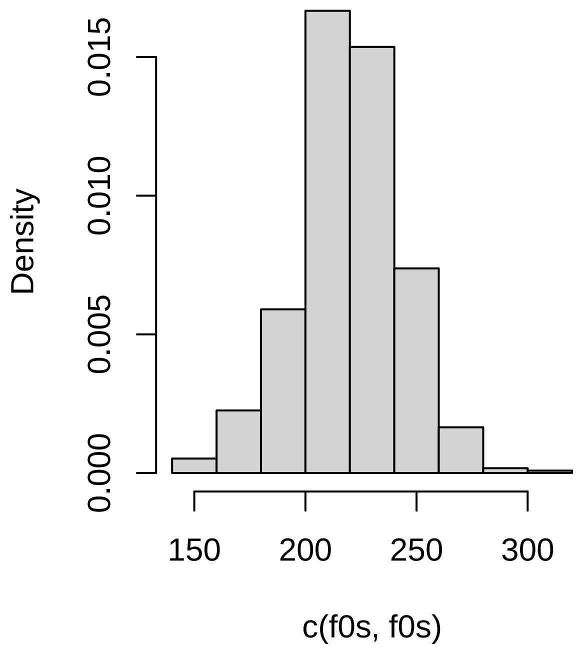
```
par (mfrow = c(1,2))

hist (f0s, main="")
hist (f0s, freq = FALSE, main = "")
```



Below I've repeated the data, doubling the counts. Notice that the y axis in the right panel does not change. This is because more observations change your counts but not the relative frequencies of observations. For instance, increasing the number of coin flips will not change the fact that 50% will be heads, but it will change the number of heads observed.

```
par (mfrow = c(1,2))

hist (c(f0s,f0s), breaks = 10, main = "")
hist (c(f0s,f0s), freq = FALSE, breaks = 10, main = "")
```





The distribution of many variables (including f0) follows what is called a normal distribution. This means if you take a random sample and arrange observations into bins, they will tend to resemble the shape of a normal distribution. This distribution is also called a Gaussian distribution and has a familiar, bell-shaped curve.

The normal distribution has the following important characteristics.

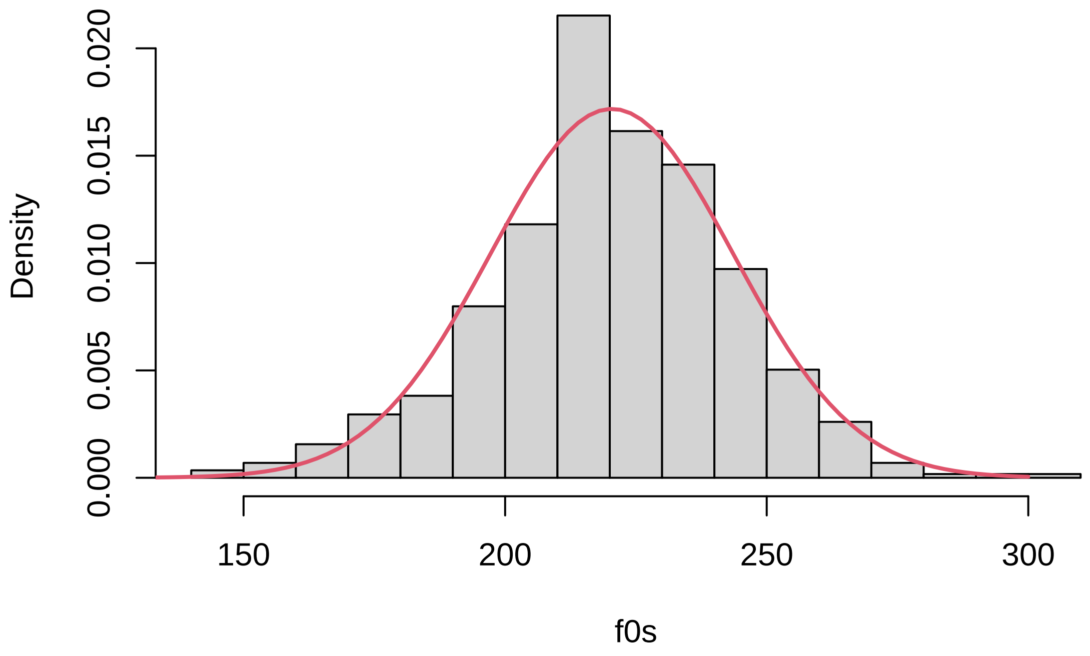
1. The distribution is approximately symmetrical - i.e., producing a higher or lower than average f0 is about equally likely.
2. The probability of observing a given value decreases as you get further from the mean.
3. It is easy to work with, very well understood, and naturally arises in basically all domains.

Normal distributions have two parameters. This means they vary from each other in only two ways. These parameters are:

1. A mean, which determines where the distribution is located along the x axis. The mean is the 50% halfway point of the 'mass' of the distribution. If the distribution were an physical object, its mean would be its center of gravity.
2. A standard deviation (or variance) that determines its *spread* along the x axis. Since every distribution has an area under the curve equal to one (they all have the same 'volume'), the smaller the variance the higher the peak of the density along the y axis.

Below, I compare the histogram of f0 values to the density of a normal distribution with the same mean and standard deviation of our sample. The density was drawn using the *dnorm()* function. This function will help draw a curve representing the shape of a theoretical normal distribution with a given mean and standard deviation.

```r
par (mfrow = c(1,1))
hist (f0s, freq = FALSE, main = "", breaks = 20)
abline (v = 63.8, lwd = 2, col = 2, lty=3)
## plots the normal density (red line) using stats calculated form our sample.
curve (dnorm (x, mean(f0s), sd(f0s)), from = 100, to = 300,
       lwd=2, col = 2, add = TRUE)
```



Notice that our real life measurements follow the 'shape' predicted by the theoretical distribution. Since they often match the real-life distribution of observations, probability distributions are often used to make inferences about the real-life populations and observations.

1.1.3 Referring to the normal distribution to make inferences

Can we use our data to figure out what the probability of observing a production from a female speaker with an f0 of under 175 Hz?

Since is that your data is unlikely to match the exact characteristics of the true population, we can't just take our observations at face value. Unfortunately, finding the *true* probability of observing an event is not possible! We would need to go out and interview every woman in the population. By the time we were done, some would have died and others would have aged into the category, voices might have changed, people might have moved, etc.

What scientists we do instead is refer to the properties of theoretical distribution to make inferences about real data. The basic process is as follows:

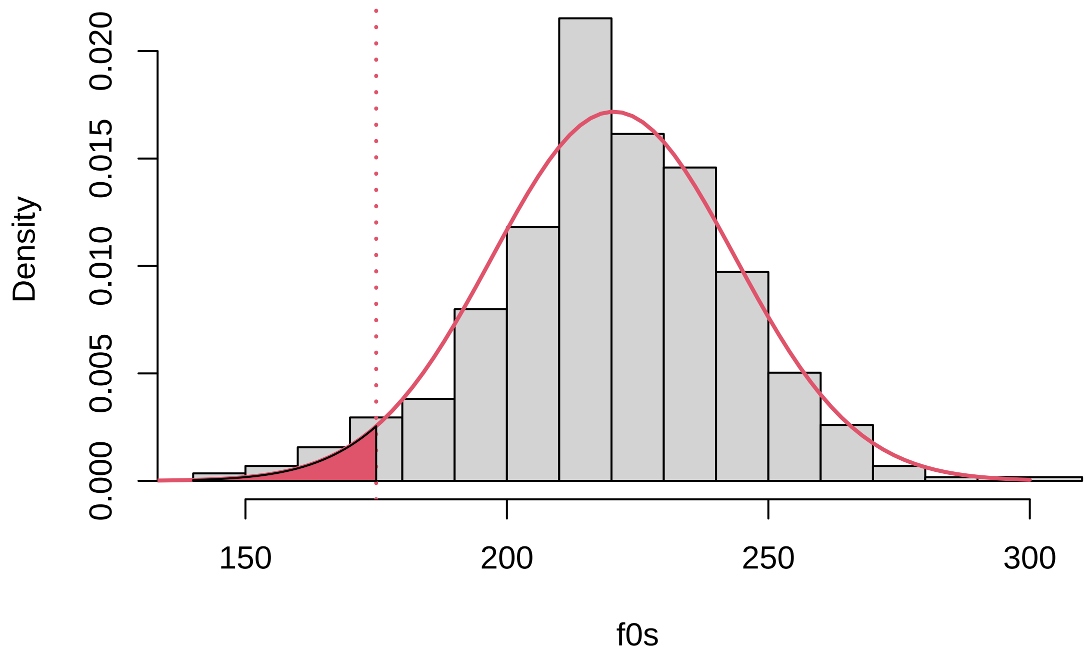
1. Sample from a population, and collect measurements (in our case Michigan adult females).

2. Establish which theoretical distribution best reflects the variation in our data. For continuous data, this will almost always be the normal distribution (looks pretty good above).
3. Estimate the distribution parameters from the sample. In the case of the normal distribution, this is the mean and standard deviation.
4. Refer to the theoretical distribution to make inferences about the real distribution.

So, we can use the theoretical normal density to figure out the probability of observing a female production with an f0 of under 175 Hz, based on the assumptions that our sample is representative of the population.

We do this by referring to the proportion of values expected to be greater/less than some reference value. This can be found by finding the area under the curve of the probability density to the left/right of that point (the red area below). Since the *total* area is always equal to 1, the area of the red portion below correponds to a percentage or a probability. Below, I use the function *pnorm* to find out the proportion of values that are expected to be greater/less than 175 Hz. I use the parameters estimated form our sample to run the prnom function, as these are our best guesses of the population parameters. As we can see, this value is reasonably close to our empirical proportion, which was 0.038 (3.8%).

```r
hist (f0s, freq = FALSE, main = "", breaks = 20)
abline (v = 175, lwd = 2, col = 2, lty=3)
## plots the normal density (red line) using stats calculated form our sample.
curve (dnorm (x, mean(f0s), sd(f0s)),from=100, to=300, lwd=2, col = 2, add=TRUE)

x = c(140,seq(140,175,length.out = 100),175)
y = c(0,dnorm(seq(140,175,length.out = 100), mean (f0s), sd (f0s)),0)
polygon(x, y, col='2')
abline (v = 63.8, lwd = 2, col = 2, lty=3)
abline (v = 70, lwd = 2, col = 1, lty=3)
```



```r
## probability that a woman is *less than* 70 inches tall
pnorm (175, mean (f0s), sd(f0s))
```

```
## [1] 0.02527988
```

```r
## probability that a woman is *more than* 70 inches tall
1 - pnorm (175, mean (f0s), sd(f0s))
```

```
## [1] 0.9747201
```

This may sound complicated but its surprisingly simple conceptually. Imagine you had 1 pound of clay and I asked you to make a shape **exactly** like the normal density (red curve) above with a constant depth. The 'area under the curve' would just correspond to the amount of play dough in a certain area. So, if you made the density just right and I took a knife and cut the shape left of 175 Hz (the red part) and we weighed it, it should weigh 2.5% of a pound. So, the area under the curve, the probability, is literally just the amount of the *stuff* in the density that falls below/above a certain point, or between two points.

Since the total number of observations is always one. This helps us compare across many different actual numbers of observations. The probability above suggests the following:

```r
## probability of observing a production with an f0 under 175 Hz
pnorm (175, mean (f0s), sd(f0s))
```

```
## [1] 0.02527988
```

```r
## expected count
pnorm (175, mean (f0s), sd(f0s)) * length (f0s)
```

```
## [1] 14.56121
```

```r
## actual count
sum (f0s < 175)
```

```
## [1] 22
```

We can also use this theoretical distribution to think about things we *did not observe*:

```r
min (f0s)
```

```
## [1] 149
```

```r
pnorm (149, mean (f0s), sd(f0s)) # probability of observing our smallest value
```

```
## [1] 0.001052907
```

```
pnorm (140, mean (f0s), sd(f0s)) # probability of observing a smaller value
```

```
## [1] 0.0002676171
```

```
## predicted count for larger experiment
pnorm (175, mean (f0s), sd(f0s)) * 1000
```

```
## [1] 25.27988
```

1.2 Considering what *other* samples might have been like

Our sample of f0 values is finite and randomly chosen. If our data had been even slightly different we would have arrived at different parameter estimates, potentially leading to different conclusions. We can use the *sample()* function to imagine this variability. Below we randomly sample 10 values form our sample repeatedly.

```
sample (f0s, size = 10)
```

```
##  [1] 258 236 211 239 233 248 227 185 237 225
```

```
sample (f0s, size = 10)
```

```
##  [1] 156 244 219 227 210 285 227 211 195 252
```

```
sample (f0s, size = 10)
```

```
##  [1] 173 257 225 226 226 229 211 221 251 221
```

```
sample (f0s, size = 10)
```

```
##  [1] 226 224 255 206 188 234 238 202 236 238
```

Obviously, the means calculated from these different samples would be different from each other, and from the population mean. In general, the parameters estimated from any sample will be randomly different from the true values, due to a combination of error and inherent variability. This is called sampling variability (or sampling error, or sampling variation) and it is an **extremely** important concept.

1.2.1 Sampling Variation

We can use our data to answer the question: what might other data look like? Below we randomly sample 10 observations from our full sample, and calculate the mean each time. This general approach is called (resampling)[https://en.wikipedia.org/wiki/Resampling_(statistics)] (since you are 're'-sampling from your single sample].

Resampling allows us to consider the *distribution* of our parameter estimates across different possible experiments. Below, I have written a function that takes a sample of a given size and calculates the mean of that sample. If I take 5000 samples of size 20 and calculate the mean, for example, I am simulating 5000 parallel universes where people took different samples and came to different conclusions about the population mean.

The distribution of sample means seen below is called the *sampling distribution*. It reflects the values your parameters might take given replications of the experiment in roughly the same conditions. Keep in mind, this is a distribution of *parameters*, not of data. Since these distributions are of parameters, we don't say they have a standard deviation, we say they have a *standard error*. It's the same thing, just with a different name to highlight the differences (variation in data vs. variation in parameters).

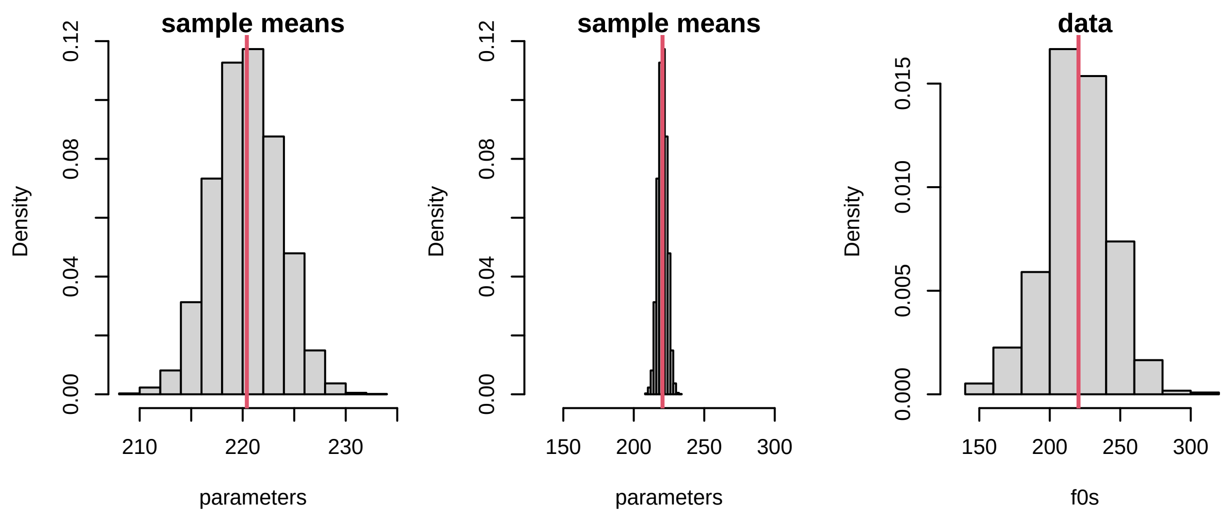
```r
## special function that samples from our data and calculates parameter values
sample_fnc = function (size = 10){
  tmp_sample = sample (f0s, size = size, replace = TRUE)
  mean (tmp_sample)
}

# sampling our data 500 times and transposing (it gives you data across columns)
parameters = replicate (5000, sample_fnc(50))

## have a look at the data, left column is the mean, right column is the sd
head (parameters)
```

```
## [1] 220.22 218.46 221.92 224.76 224.44 224.80
```

```r
str (parameters)
```

```
##  num [1:5000] 220 218 222 225 224 ...
```

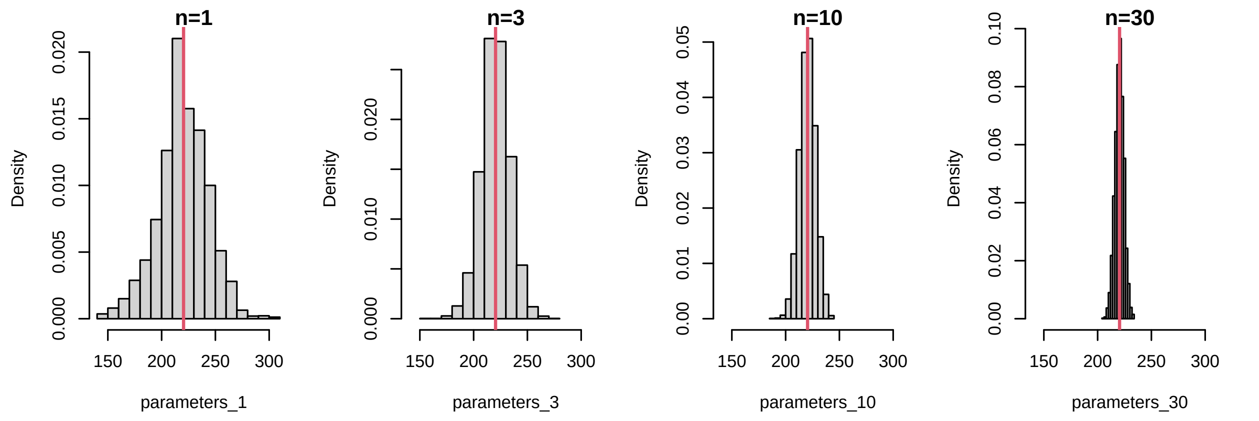
```r
par (mfrow =c(1,3))
## sample means, narrow x axis
hist (parameters, freq = FALSE, main = "sample means")
abline (v = mean (f0s), lwd=2,col=2)
## sample means plotted with same x range as data
hist (parameters, freq = FALSE, main = "sample means", xlim = c(130,320))
abline (v = mean (f0s), lwd=2,col=2)
# underlying data
hist (f0s, freq = FALSE, main = "data", add = FALSE, xlim = c(130,320))
abline (v = mean (f0s), lwd=2,col=2)
```



Notice that the sampling distribution of sample means is much narrower than the distribution of our original data. This is because our mean is based on 10 samples and so positive and negative deviations from the average will tend to cancel out. Another way to think of this is that the more data you have the more *precise* your estimates are, and the less *uncertainty* is associated with estimates.

It turns out that the standard error of the sample mean is *extremely* predictable based on the underlying variability in your data. Basically, the more data you have the less variation you have in your sample mean (in the limiting case, if your sample includes your entire sample you have 0 variation). Another way to think of this is that the more data you have, the less uncertainty you have regarding the properties of that data. Below, we can see that taking larger samples decreases the standard error of the resulting sampling distribution.

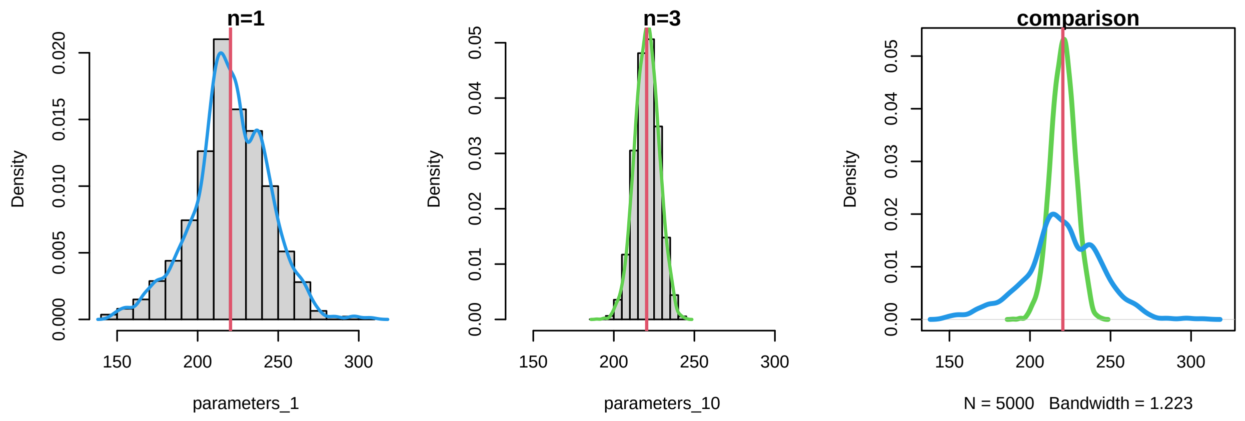
```
parameters_1 = t ( replicate (5000, sample_fnc(1)) )
parameters_3 = t ( replicate (5000, sample_fnc(3)) )
parameters_10 = t ( replicate (5000, sample_fnc(10)) )
parameters_30 = t ( replicate (5000, sample_fnc(30)) )

par (mfrow =c(1,4))
hist (parameters_1, freq = FALSE, main = "n=1", xlim = c(140,320))
abline (v = mean (f0s), lwd=2,col=2)
hist (parameters_3, freq = FALSE, main = "n=3", xlim = c(140,320))
abline (v = mean (f0s), lwd=2,col=2)
hist (parameters_10, freq = FALSE, main = "n=10", xlim = c(140,320))
abline (v = mean (f0s), lwd=2,col=2)
hist (parameters_30, freq = FALSE, main = "n=30", xlim = c(140,320))
abline (v = mean (f0s), lwd=2,col=2)
```



Before, I plotted curves representing theoretical probability densities for the normal distribution. We can also use functions in R to estimate the density of the distribution of our data (or any other values). A density is basically just a continuous histogram, and it can be calculated using the *density()* function.

Below, I compare some of the histograms above to their estimated densities. Note how much less variability/uncertainty there is in our estimates just by taking 3 as opposed to 1 sample. Note also that using densities rather than histograms makes distributions easier to compare.

```r
par (mfrow =c(1,3))
hist (parameters_1, freq = FALSE, main = "n=1", xlim = c(140,320))
lines (density (parameters_1), lwd = 2, col = 4)
abline (v = mean (f0s), lwd=2,col=2)
hist (parameters_10, freq = FALSE, main = "n=3", xlim = c(140,320))
lines (density (parameters_10), lwd = 2, col = 3)
abline (v = mean (f0s), lwd=2,col=2)
plot (density (parameters_10), type = 'l', xlim = c(140,320), col = 3, lwd=3, main="comparison")
lines (density (parameters_1), type = 'l', col = 4, lwd=3)
abline (v = mean (f0s), lwd=2,col=2)
```



Here are some things I am just going to assert as facts without much explanation (but I swear they're true!):

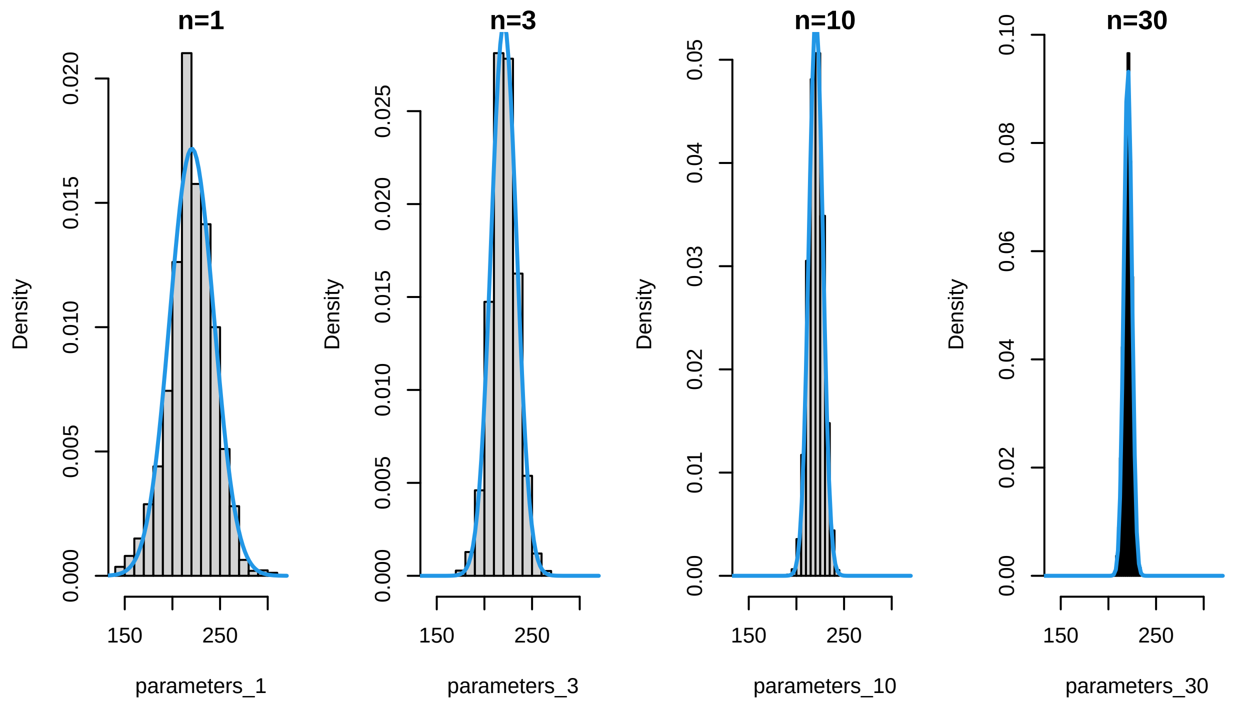
1. The sample mean will be normally distributed in a wide variety of cases, no matter what the underlying data looks like. The larger the sample, the more this is true.

2. The mean of the sample means will be equal to the population mean (in a wide variety of circumstances). This means that the sample mean is our best as to the population mean.

3. The standard error of the sample mean is equal to the standard deviation of the data, divided by the square root of N (the sample size).

Below I compare the histograms of our observed samples (presented above) compared to their **theoretical** densities. These densities were drawn using parameters estimated from the entire sample, but adjusted for sample size. Notice that I am able to exactly predict the distribution of sample means for *real data* using a completely theoretical distribution, and based solely on statistical principles.

```
N = length (f0s)
mean_f0s = mean (f0s)
sd_f0s = sd (f0s)

## in Each case below, dividing the standard deviation by the square root of the
## sample size perfectly adjusts the theoretical normal density
par (mfrow =c(1,4))
hist (parameters_1, freq = FALSE, main = "n=1", xlim = c(140,320))
curve (dnorm (x, mean_f0s, sd_f0s / sqrt ( 1 )), add = TRUE,
from = c(100,300), lwd = 2, col = 4)

hist (parameters_3, freq = FALSE, main = "n=3", xlim = c(140,320))
curve (dnorm (x, mean_f0s, sd_f0s / sqrt ( 3 )), add = TRUE,
from = c(100,300), lwd = 2, col = 4)

hist (parameters_10, freq = FALSE, main = "n=10", xlim = c(140,320))
curve (dnorm (x, mean_f0s, sd_f0s / sqrt ( 10 )), add = TRUE,
from = c(100,300), lwd = 2, col = 4)

hist (parameters_30, freq = FALSE, main = "n=30", xlim = c(140,320))
curve (dnorm (x, mean_f0s, sd_f0s / sqrt ( 30 )), add = TRUE,
from = c(100,300), lwd = 2, col = 4)
```



1.2.2 Estimating sampling variability using theoretical distributions

As shown in the plot above, if we know the mean, standard deviation, and size of a sample, we can predict the amount of variability we expect in estimates of the mean (or any other parameter). This also means we can predict the *shape* of the distribution, including the amount of the curve above/below a given point.

Before explaining this I just want to clarify: You are never going to have to do this. I am only going over this because understanding this can be useful to really understand whats going on with statistical models.

In Section 1.3, I mentioned using the normal distribution to make inferences. When variables are normally distributed we can use the theoretical normal distribution and functions such as *pnorm()* to answer questions about values we expect, and don't expect, to see. We can take this same approach to make inferences about *parameters*.

For example, we can use the results of the calculations below:

```
mean (f0s)
```

```
## [1] 220.401
```

```
sd (f0s)
```

```
## [1] 23.22069
```

```
length (f0s)
```

```
## [1] 576
```

```r
sd (f0s) / sqrt ( length (f0s) ) ## the standard error
```

```
## [1] 0.9675289
```

To draw the expected distribution of these values. This distribution is called a likelihood function, because it shows the distribution of a parameter and not of data. We can use likelihood functions to rule out implausible values for our population parameters. The reasoning goes like this:

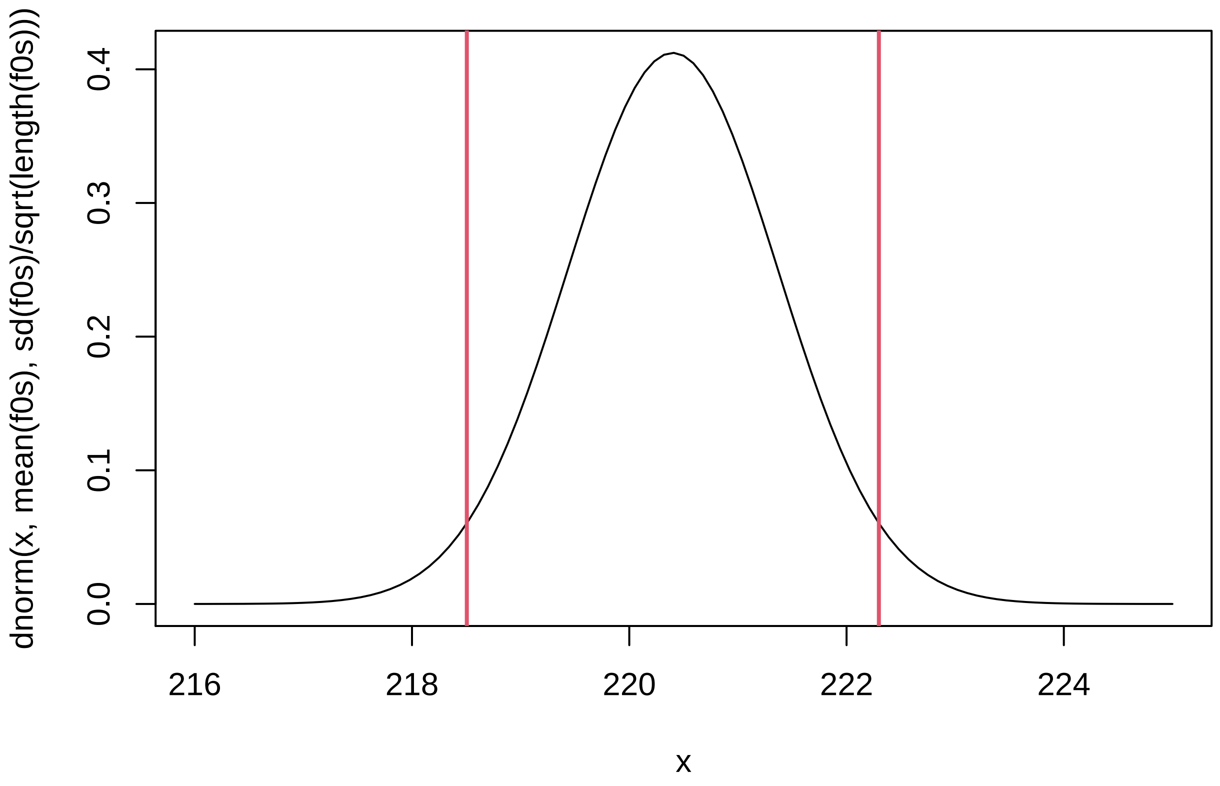
1) We know that sample means usually equal population means, but are also slightly off.
2) The sampling distribution tells us how wrong these estimates will usually be wrong by.
3) If the sampling distribution says a parameter value is very unlikely, that means is it not a plausible conclusion.
4) if the sampling distribution says a value is very likely, it may be close to the true value.

For example, we can use estimate the sampling distribution of our entire sample by using the sample mean, and the sample standard deviation divided by the squae root of the sample size. We can then use the *qnorm()* function to calculate quantiles for our sampling distribution, presented below. I added vertical lines at the 2.5% and 97.5% quantiles of our distribution. These vertical lines enclose 95% of the probability density, and so represent the range of values representing 95% of possible outcomes.

```r
par (mfrow =c(1,1))
curve (dnorm (x, mean(f0s), sd(f0s)/sqrt(length(f0s))), xlim = c(216,225))
quantiles = qnorm (c(0.025, 0.975), mean (f0s), sd (f0s) / sqrt (length (f0s) ) )
quantiles
```

```
## [1] 218.5047 222.2974
```

```r
abline (v = quantiles, lwd=2,col=2)
```



Given the information presented in the figure above, we expect 95% of samples of this size from this population to have a sample mean between 218 Hz and 222 Hz. This means that it is reasonable that the true value might be 221 Hz, as this value is very likely given our sample. Basically, maybe our sample mean is wrong and is arose by accident, and 221 Hz is correct. This outcome is compatible with our data.

However, a value of 216 Hz is extremely unlikely to arise given our sample. It is just too far from our sample mean relative to the amount of variation in our sample. This is like if you measured the heights of 100 women in a small town (pop. 1500) and found the average was 5'4", you might accept that the actual population average is 5'5", but may find it difficult to accept that it was actually 6'0". It would mean you happened to measure all of the shortest women in the town, an extremely unlikely event.

So, since we think that 216 Hz is not a plausible mean f0 given our sample, this also means that it is very unlikely that the true mean is 216 Hz. This is because a distribution centered at 216 would be extremely unlikely to generate a sample mean of 220 Hz (based on the same reasoning outlined above). So, based on the above we can rule out implausible values of the *true* population mean based on the characteristics of our data, and what this suggests about the sampling distribution of the parameters of interest (e.g., the sample mean).

By convention, researchers usually express 95% intervals: bounds which enclose 95% of possible/credibly parameter values. This is what the plot above expresses, it shows the range of the 95% most likely population means given our data and model structure.

At this point we can offer conventional responses to the research questions posed in Section 1.1.

Q1) What is the average f0 of the whole *population* likely to be?

A1) The most likely value for the population mean is our sample mean, 220.4 Hz.

Q2) Can we set bounds on likely mean f0 values based on the data we collected?

A2) Yes, there is a 95% probability that the population mean is between 218.5 222.3 Hz, given our data and model structure.

1.3 Using brms to estimate parameter values and credible intervals

We are going to be using an R package called *brms*. This package lets you use *STAN* easily, nearly as easily as fitting a more traditional model. We are going to fit our first model, to answer the questions I first posed in Section 1.1.1.

Everything so far has been to help understand the output brms gives you. I am going to be focusing on analyzing different research designs in *brms*, and on how to interpret the output of the functions in this package.

Below, I load the package and fit a model that estimates population parameters given our sample. It assumes we want to use a normal distribution by default. The first term is the formula, it tells the function about the structure of the model. We will talk about this in a lot more detail later, but the formula basically tells it to predict f0 based on only a mean value, and no predictors. The single predictor is referred to as an 'intercept' because our models are based on lines. Shifting the intercept up or down brings all values up or down, thereby affecting the mean value. The number 1 is used by convention to represent a model in which only a mean is being estimated. The other terms tell the function to take 4000 samples using one processing core, using a single set of samples.

```
library (brms)
```

```
 set.seed (1)
 # I am giving it the original dataframe because brms won't accept vectors
 model = brm (f0 ~ 1, data = w, chains = 1, cores = 1, iter = 4000)
```

```
## Compiling Stan program...

## Start sampling

##
## SAMPLING FOR MODEL '98dae0f1caaef07c210aac2156c73749' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 1: Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 1: Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 1: Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 1: Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 1: Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 1: Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 1: Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 1: Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 1: Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 1: Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 1: Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 1:
```

```
## Chain 1:  Elapsed Time: 0.101 seconds (Warm-up)
## Chain 1:                0.084 seconds (Sampling)
## Chain 1:                0.185 seconds (Total)
## Chain 1:
```

The *brm* function allows you to specify a statistical model, and to take random samples from the likelihood function. Remember, the likelihood function tells us the values that are most plausible given our data and model structure. As a result, the distribution of the samples made by brms ends up telling us about the range of plausible parameter values.

The output above shows you that the sampler is working, and tells you about the progress as it works. This is the last time I will be actually fitting a model in the code chunks. I am going to be relying on pre-fit models that you can load directly from GitHub like this:

```
model = readRDS ('1_model.RDS')
```

We can eavaluate the model name to show the default `brms` model print tatement:

```
model
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: f0 ~ 1
##    Data: w (Number of observations: 576)
## Samples: 1 chains, each with iter = 4000; warmup = 2000; thin = 1;
##          total post-warmup samples = 2000
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept   220.40      0.95   218.46   222.19 1.00     1912     1457
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma    23.24      0.66    21.95    24.57 1.00     1901     1441
##
## Samples were drawn using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

The model above shows us that our mean estimate ('Population-Level Effects: Intercept') is 220.4. Notice that the the 2.5% and 97.5% quantiles estimated for this parameter are an excellent match to the theoretically determined values first shown in Section 1.2.2:

```
qnorm (c(0.025, 0.975), mean (f0s), sd (f0s) / sqrt (length (f0s) ) )
```

```
## [1] 218.5047 222.2974
```

They almost exactly match because they are the same information estimated in two different ways. As the models we are interested in become more and more complicated, using brms becomes very useful. We can also get a lot of useful information from our brms models, notice that in addition to estimating the population mean, we also get an estimate of the population standard deviation (Family Specific Parameters: sigma), and an estimate of its plausible ranges (2.5% = 21.93, 97.5% = 24.74).

1.3.1 Sampling from the likelihood function

In Section 1.2.1 I talked about the likelihood function for the sample mean. The are statistical programs called samplers that can sample values from the likelihood function for a model and parameters. What they do is they repeatedly sample from the likelihood in a certain way, repeatedly keeping or discarding observations. Given certain conditions, the result of this is that after enough samples, the distributions of samples collected by the sampler will converge on (become more similar to) the likelihood of the parameter.

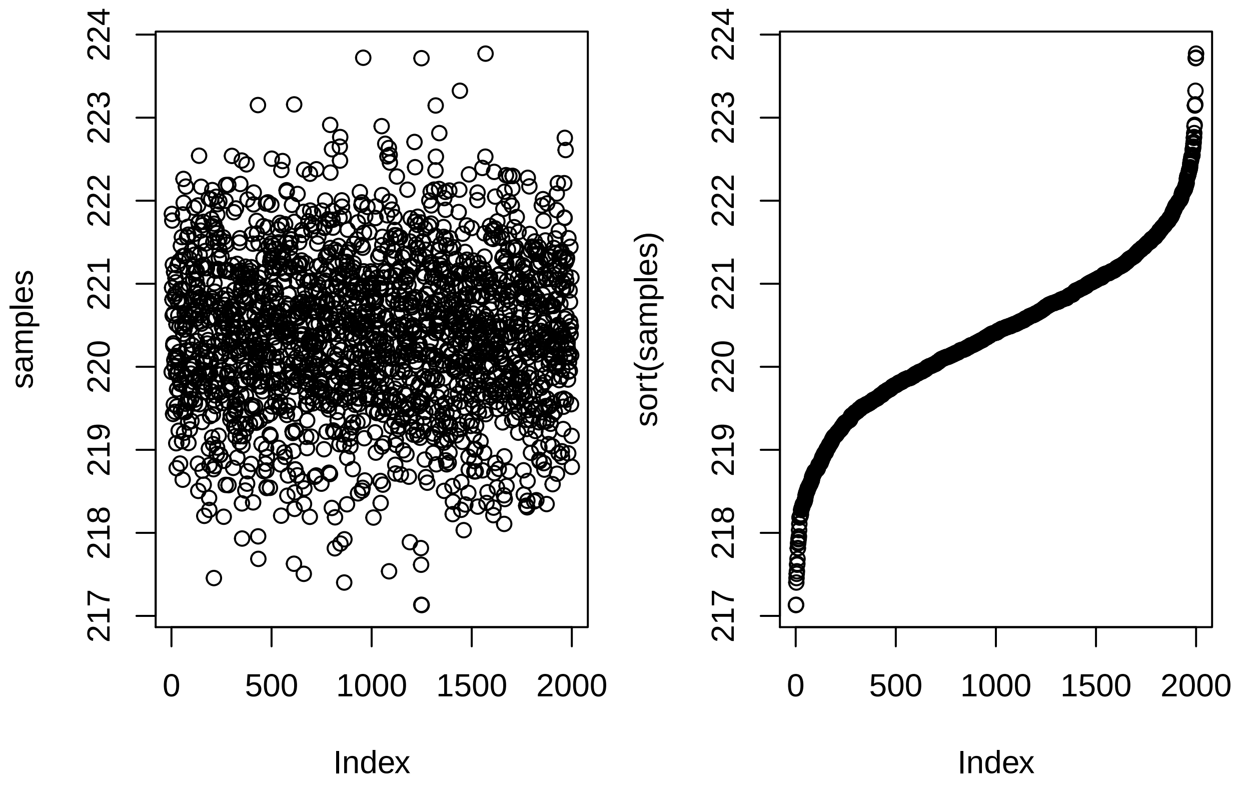
brms does all this for us, and summarizes the characteristics of the sample. But we can collect these samples and see how they look just like what we expected, and how *brms* summarizes these samples and provides them to us in a digestible manner.

First, we can get the samples and plot them just to have a look.

```r
# this fetches the samples associated with the parameter we want
samples = posterior_samples (model, "Intercept")[,1]
head (samples)
```

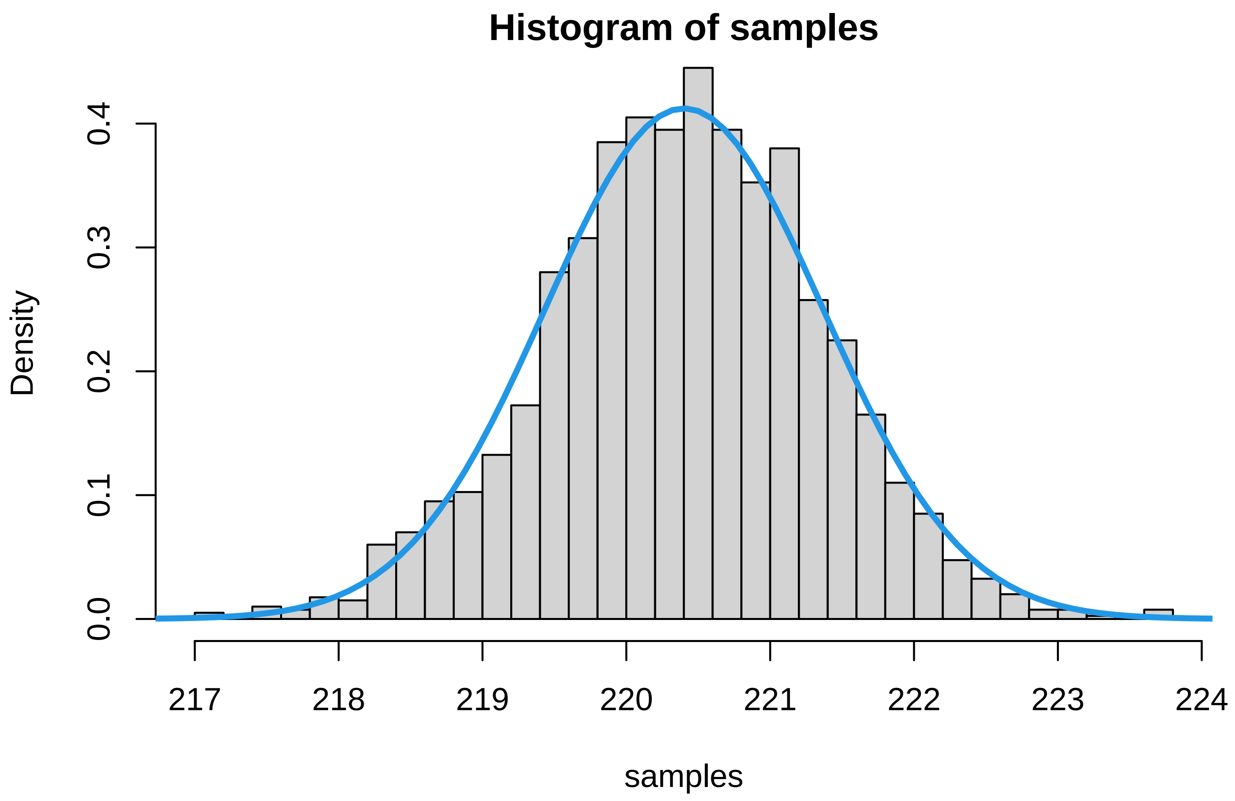
```
## [1] 219.9382 221.8407 220.9520 221.7609 220.8087 220.6191
```

```r
par (mfrow =c(1,2))
## we can plot the individual samples
plot (samples)
## or sort and then plot them.
plot ( sort ( samples ) )
```



Then, I can compare the histogram of our samples collected by brms to the theoretical distribution of the likelihood of the sample mean. I can also verify that the actual quantiles form our samples almost exactly match the quantiles calculated from the theoretical distribution. Finally, we can see that the sampled quantiles exactly correspond to the interval reported for this parameter in the model output above. That is because *brms* is relying these samples to provide you the information in the model output.

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

# the theoretical and sampled distributions are about the same
hist (samples, freq = FALSE, breaks = 30)
curve (dnorm (x, mean(f0s), sd(f0s)/sqrt(length(f0s))), xlim = c(216,225),
       add = TRUE, lwd = 3, col=4)
```



```
# the quantiles calculated from both are also about the same
## theoretical quantiles
qnorm (c(0.025, 0.975), mean (f0s), sd (f0s) / sqrt (length (f0s) ) )
```

```
## [1] 218.5047 222.2974
```

```
## sample quantiles
quantile (samples, c(0.025, 0.975))
```

```
##     2.5%    97.5%
## 218.4614 222.1905
```

1.4 Explaining ‘multilevel Bayesian models’

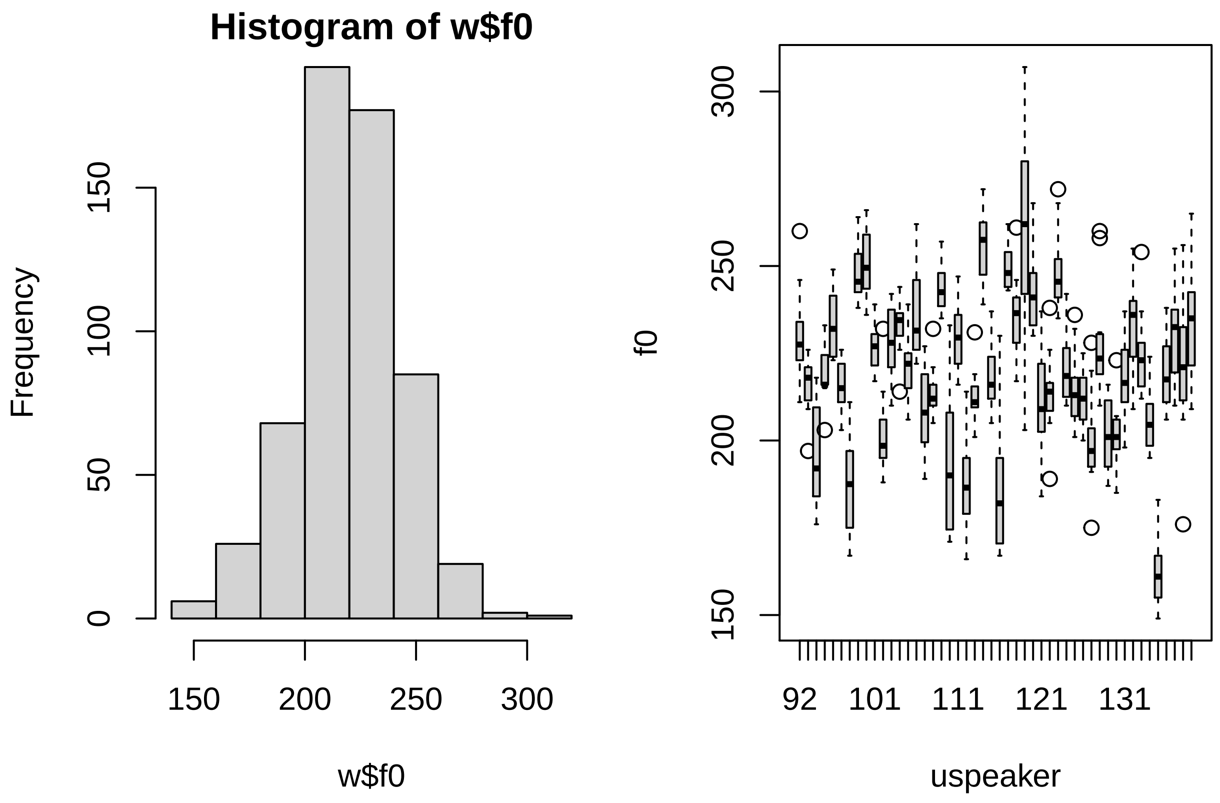
In this class we are going to learn about ‘multilevel’ models. These models are also called ‘mixed effects’ or ‘random effects’ models. Basically, these are models where some of your parameters are like random variables. These models often arise whenever you have ‘repeated-measures’ data, that is when you collect multiple observations from any given source (e.g., a single person). In linguistics and most other social sciences, almost all of our data is usually repeated-measures data, meaning we actually need to analyze it using a multilevel model.

Our approach will be to analyze this kind of data using multilevel Bayesian models, which will be explained below.

1.4.1 Repeated-measures data

The reason repeated-measures designs need special treatment can be seen in the plots below. On the left is a histrogram of the f0s produced by the women, boys and girls in our sample. Everything appears to be normal, with values normally-distributed around the mean. On the right however, tokens are grouped by talker.

```r
par (mfrow = c(1,2))
hist (w$f0)
boxplot (f0 ~ uspeaker, data = w)
```



It's clear that our distribution of f0s does not consist of totally independent observations. There are obvious clusters according to speaker. As we can see below there are 94 speakers, each of which provided 12 tokens each. This means we do not actually have 1128=94*12 totally independent observations here.

```
table (w$uspeaker) ## table of number of observations per speaker
```

```
##
##   92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116
##   12  12  12  12  12  12  12  12  12  12  12  12  12  12  12  12  12  12  12  12  12  12  12  12  12
## 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139
##   12  12  12  12  12  12  12  12  12  12  12  12  12  12  12  12  12  12  12  12  12  12  12
```

```
length(table (w$uspeaker)) ## number of unique speakers
```

```
## [1] 48
```

This matters because it gives us a warped perspective of how much variability there really is in the sample. For example, if I told you I had 1,000,000 samples of speech from male speakers from Los Angeles, you may be confident that I can estimate average pitch for male speakers from Los Angeles. But what if I told you all samples were from only three different people? You know instinctively that this would be a problem. The reason why is because the measurments are correlated: multiple measurements from the same person are obviously going to be related to each other.

To demonstrate how this can cause problems, we can return briefly to the question of what the average f0 is for adult females in our sample, addressed in the previous section. In the previous analysis, we treated every single token as an independent observation, ignoring the fact that multiple tokens were produced by each talker.

Below, I fit two versions of the same intercept-only model as before: one using all the individual data, and another using only one observation per person. Solely based on the fact that we have reduced our number of observations by a factor of 12, we obviously expect more variation in our estimates in the speaker-average model.

```
## fit the model with all the individual observations
set.seed (1)
all_data_model = model
  brm (f0 ~ 1, data = w, chains = 1, cores = 1, iter = 4000)
```

```
## aggregate the data so that there is one averaged observation per talker
agg_data = aggregate (f0 ~ uspeaker, data = w, FUN = mean)
```

```
## fit the model with only the averaged data
set.seed (1)
speaker_average_model =
  brm (f0 ~ 1, data = agg_data, chains = 1, cores = 1, iter = 4000)
```

```
## read data from pre-fit model
all_data_model = readRDS ("1_all_data_model.RDS")
speaker_average_model = readRDS ("1_speaker_average_model.RDS")
```

Notice that the means are nearly identical across the two models. However, the credible interval (the boundary of believable values) for the mean parameter is substantially wider than the range we find using the model using all our observations. Basically, not acknowledging the fact that we have correlated observations in our data results in a misleadingly-precise estimate of the mean parameter.

```
all_data_model
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: f0 ~ 1
##    Data: w (Number of observations: 576)
## Samples: 1 chains, each with iter = 4000; warmup = 2000; thin = 1;
##          total post-warmup samples = 2000
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept   220.40      0.95   218.46   222.19 1.00     1912     1457
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma    23.24      0.66    21.95    24.57 1.00     1901     1441
##
## Samples were drawn using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

```
speaker_average_model
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: f0 ~ 1
##    Data: agg_data (Number of observations: 48)
## Samples: 1 chains, each with iter = 4000; warmup = 2000; thin = 1;
##          total post-warmup samples = 2000
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept   220.46      2.86   214.57   226.02 1.00     1405     1222
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma    20.29      2.09    16.70    24.67 1.00     1552     1391
##
## Samples were drawn using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

1.4.2 'multilevel': What are the multiple levels?

The 'multilevel' nomenclature comes from the recognition that in many repeated measures models, the units that provide our data behave much like random variables themselves. As a result, this results in at least two levels of variation for f0 production:

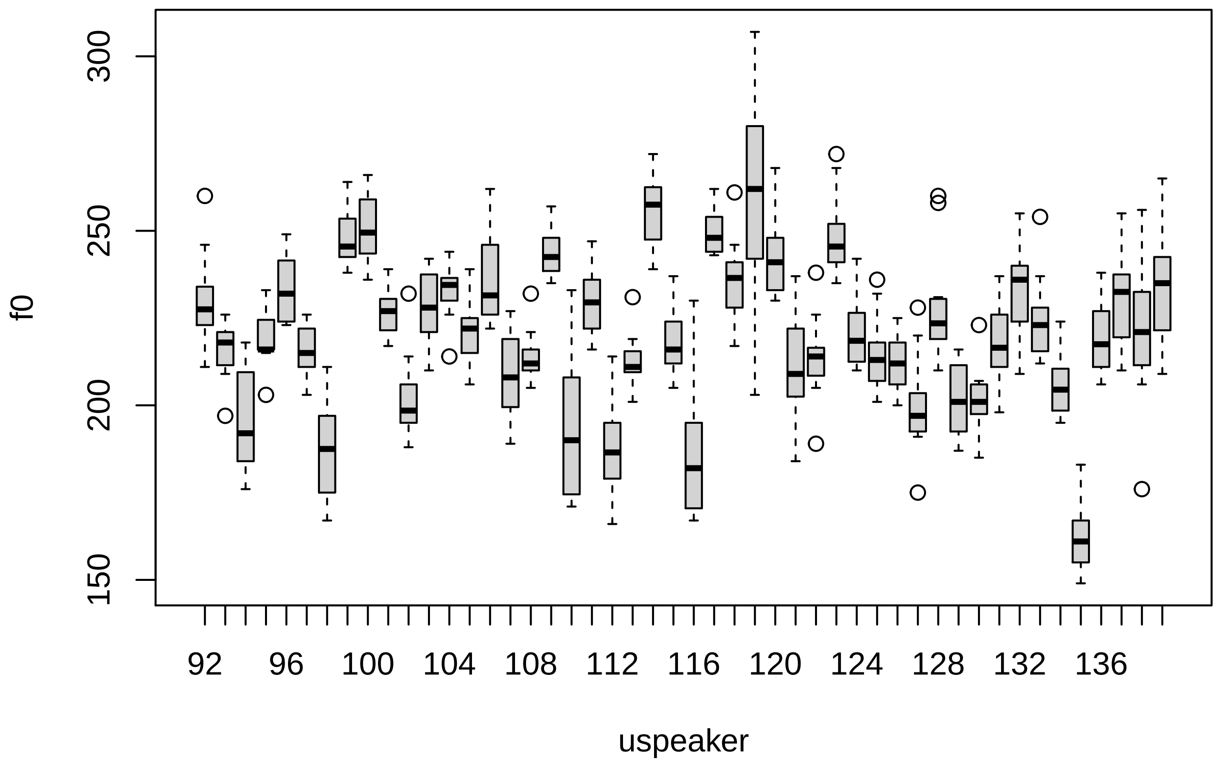
The 'upper' level: Between-speaker variation. Speakers are chosen randomly from a larger population. Each speaker has an average f0 that they produce over time.

The 'lower' level: Within-speaker variation. When an individual speaker produces tokens, their productions will vary around their average.

The variation at the lower and upper levels are analogous. Just like individual speakers will rarely have an average f0 exactly like the population average, individual speakers will rarely produce f0 values exactly at their talker average.

We can think of each of the speaker-specific boxes below as a tiny little normal distribution, centered at the speaker average. Furthermore, each speaker average can be thought of as a single point in the population-level, normal distribution of speakers. It is the accumulation of each of these tiny normal distributions into one group that leads to the single, sample-wide distribution seen above.

```r
par (mfrow = c(1,1))
boxplot (f0 ~ uspeaker, data = w)
```



Importantly, variation at the two levels is quasi-independent and logically distinct. For example, within-speaker variation can be small or large independently of whether between speaker. Further, although perhaps within-speaker variation can be though of as 'error', between-speaker variation is meaningful as it tells us about the variation between different speakers in the population. By using multilevel models we can independently estimate multiple sources of variation.

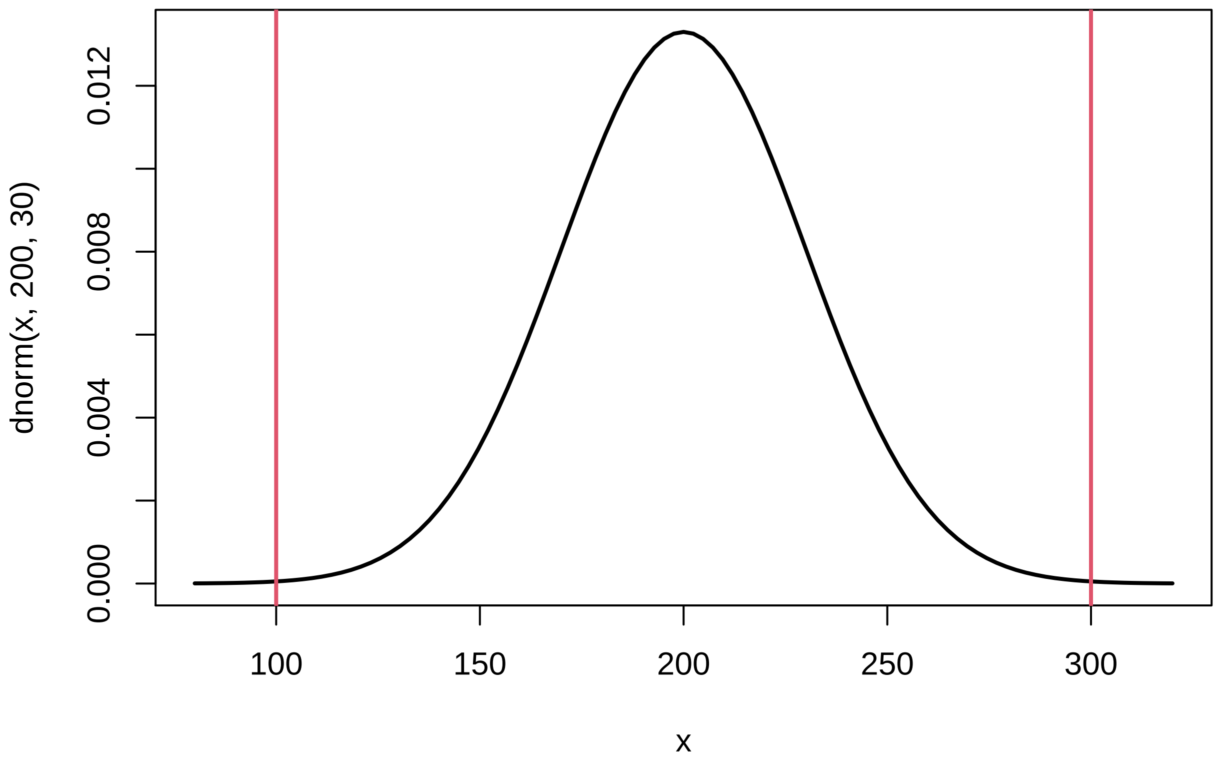
1.4.3 'Bayesian': What's 'Bayesian' about it?

There are two important ways that what we are learning is different from 'regular' models such as ordinary least-square regression and linear mixed-effects models as implemented in the *lmer* function in the *lme4* package.

1) We are estimating parameter values using sampling and not maximum likelihood estimation. *lmer* and traditional approaches use something more like maximum likelihood estimation. This is a technical difference and is not something to worry about. The two approaches are equivalent given equivalent designs. The only way this affects you is that your model consists of a bunch of samples, whereas a traditional model only gives you the final estimates for all the things you are interested in.
2) When using brms, you have to set prior probabilities for all your parameter estimates. This consists of specifying a probability distribution for what you expect evry parameter to be *a priori* (before experimentation). *brms* helps make this very easy.

For example, I can use the normal distribution to specify a prior probability for female mean f0. I can make the mean 200 Hz, and the standard deviation 30 Hz. This effectively means I believe that the value is very likely to be within 100-300 Hz, but not outside those values. This prior probability is based on many measurements that exist in the literature, which suggest typical mean f0s for adult females are around 200 Hz.

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

## illustration of a normal distribution with a mean of 200 and an sd of 30.
curve (dnorm (x, 200, 30), xlim = c(80,320), lwd=2)
abline (v = c(100,300), lwd = 2, col = 2)
```



The use of prior probabilities is often said to make Bayesian models ‘subjective’ but its not really a big deal. First, every model involves arbitrary decisions which can substantially affect our results. Second, a researcher will always use common sense to interpret a model. For example, before collecting my sample I can say that I expect my female average f0 to be 200 Hz or so, but think its reasonable to expect anything from 100 to 300 Hz. Based on everything we know about human speech, even these bounds are too wide, and anything outside would suggest something is very wrong. So, even if I did not use a prior, I would use my expectations to 'screen' my results, and be very wary of anything that did not meet my expectations.

A Bayesian model simply requires that you build your expectation into your model. It formalizes it, makes it definable and replicable. Also, being 'objective' does not quite make sense in many cases. Is it really being objective to ignore common sense and act as if a mean f0 of 250 is exactly as likely a priori as one of 20,000 Hz? Because not using a prior is equivalent to using a 'flat' prior and acting like almost any value is equally likely a priori, when this is hardly the case.

So, a 'Bayesian' model combines information from the like likelihood function of different parameters with the prior distributions you specified for those parameters.

1.4.4 'models': How do I make a research question, or experimental data, into a model?

This is something we are going to cover little by little. Models become more complicated to specify as the relationships they represent become more complicated.

In regression models, formulas specify the relationships we think exist between our dependent variable, which is the thing we are interested in understanding, and our predictors. Predictors are things we have measured or manipulated in order to better understand variation in the dependent variable. We are going to incrementally learn how to specify more and more complicated relationships in our formulas.

Writing formulas involves learning a bunch of conventions. The simplest formulas consist of a single predictor and look like this `formula ~ predictor`. The `~` sign can be read like "is distributed according to". For example, the formula: `f0 ~ 1` specifies that we expect that the values of f0 are distributed according to a single constant value (the mean). This is basically a model without predictors, and the number '1' is used in these situations by convention.

To specify a multilevel model, you need to write a slightly more complicated model formula. When you have experimental units that you repeatedly sample from, and you expect these to be like random variables, you place then within `()` in the formula. Our updated formula would then look like this:

`f0 ~ 1 + (1|uspeaker)`

This new formula now says: f0 is distributed only according to differences in intercepts (means), however, we expect the mean to vary randomly from person to person (uspeaker) so calculate a different intercept (mean) for each speaker.

1.5 Fitting an appropriate multilevel model for our data.

We can fit a model with a formula that appropriately specifies the clustering we expect in our data. As a result, this model can estimate both between and within speaker variability.

```r
set.seed (1)
multilevel_model =
  brm (f0 ~ (1|uspeaker), data = w, chains = 1, cores = 1, iter = 4000)
```

```r
## read data from pre-fit model
multilevel_model = readRDS ("1_multilevel_model.RDS")
```

```r
## check out model output
multilevel_model
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: f0 ~ (1 | uspeaker)
##    Data: w (Number of observations: 576)
## Samples: 1 chains, each with iter = 4000; warmup = 2000; thin = 1;
##          total post-warmup samples = 2000
##
## Group-Level Effects:
## ~uspeaker (Number of levels: 48)
##               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)    20.29      2.40    16.47    26.02 1.01      162      178
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept   220.03      2.71   214.95   225.36 1.00       63      338
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma    12.53      0.39    11.79    13.37 1.00     1640     1458
##
## Samples were drawn using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

This new model contains more information than our previous models. The 'Group-Level Effects' tell us about variation in our 'higher' level variable, in this case the between-speaker means and the variation between them variation in means. The section on 'Population-Level Parameters' contains information about systematic variation at the lowest level, variation that is common to all units. In this case, the population level mean is the overall mean for all of the data.

We can see that the information provided by *brms* is about the same as what we can estimate directly using our data. However, *brms* does this all for us, in addition to giving us a lot more information.

```r
## find mean f0 for each speaker
speaker_means = aggregate (f0 ~ uspeaker, data = w, FUN = mean)
## find the within speaker variance. This is the within-talker 'error'.
speaker_vars = aggregate (f0 ~ uspeaker, data = w, FUN = var)

## the mean of the speaker means corresponds to our overall mean estimate
mean (speaker_means$f0)
```

```
## [1] 220.401
```

```r
## sd(Intercept) in the model reflects the amount of variation in talker
## intercepts. This is the between speaker variation in our model. See how it is
## similar to the sd of the actual speaker means.
sd (speaker_means$f0)
```

```
## [1] 20.07397
```

```r
## sigma in the model reflects the amount of variation in talker intercepts.
## This is the between speaker variation in our model.
sqrt (sd (speaker_vars$f0))
```

```
## [1] 12.42719
```