Vowel formants example

Steven Moran

(16 February, 2023)

Contents

Getting started

R packages

In R, you need to install (once) any R software package (aka library) that you want to use, before you load it with the `library()` function each time that you run a script or RMarkdown file.

This code will check whether or not you have those packages, and if not, will install them and load the packages.

```r
if(!require("tidyverse")) install.packages("tidyverse")
if(!require("ggpubr")) install.packages("ggpubr")
if(!require("devtools")) install.packages("devtools")
library(devtools)
if(!require("ggConvexHull")) devtools::install_github("cmartin/ggConvexHull")
```

Normally, once a package is already installed, you simply load it with the `library()` function, like this:

```r
library(tidyverse)
library(ggpubr)
library(ggConvexHull)
```

The tidyverse package will install several R packages including:

- ggplot for creating nice looking plots
- dplyr for manipulating data easily

We are going to additionally use ggpubr package because it is helpful in making professional looking plots.

And the ggConvexHull package, which extends the `geom_polyon()` function in the ggplot package, so that we can add a convex hull around our vowel polygon data. This package is available via GitHub and hence the devtools package.

Load our vowel data

Let's load the vowel formants data from our measurements in class.

First download it from the shared Google sheets as a CSV file.

I renamed my downloaded file to `data.csv` to remove the spaces in the filename (you don't have to) and I saved it in the same directory as this `README.Rmd` file. If you want to get the current working directory in RStudio (or R) you can use this command:

```r
getwd()
```

```
## [1] "/Users/stiv/GitHub/APY313/case_studies/formants"
```

In RStudio, I set the working directory to the directory that contains this script. In this folder with the script, I also put the downloaded vowels data.

Make your life easier and **put both files in the same folder!**

You can set the working directory a la the link above or you can also do it manually with the `setwd()` function, e.g.:

```r
# setwd('/Users/stiv/GitHub/APY313/case_studies/formants')
```

Let's load the data that's in the same directory as this script.

```r
df <- read_csv('data.csv')

# If you want the data in a different directory, you can give `read_csv()` the full path! E.g.:
# df <- read_csv('/Users/stiv/Downloads/data.csv')
```

With the `read_csv()` function we read the CSV file – a comma delimited text file that represents tabular (i.e. table) data as rows and columns.

The `read_csv()` function reads the file into R as a data frame, essentially a table![1]

Have a look at the raw data

Let's have a look at the data. The function structure `str()` shows the data structure of the loaded data frame. It tells you what the columns are, what each column's data type is (e.g. `chr` means the rows in that column include characters; `num` means they include numbers) – followed by human rows there are and examples of the first few rows.

```r
str(df)
```

```
## spc_tbl_ [35 x 10] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ID    : num [1:35] 1 1 1 1 1 2 2 2 2 2 ...
##  $ Word  : chr [1:35] "heed" "hayed" "hawed" "who'd" ...
##  $ Vowel : chr [1:35] "i" "e" "a" "u" ...
##  $ F0    : num [1:35] 120 115 115 122 119 180 181 177 201 185 ...
##  $ F1    : num [1:35] 319 418 558 317 457 383 503 785 383 544 ...
##  $ F2    : num [1:35] 2522 2119 1044 874 1044 ...
```

[1] It's actuall a "tibble", which is tidyverse's version of the data frame, but that's not important for now.

```
##  $ F3    : num [1:35] 3299 3631 2677 2668 2587 ...
##  $ Sex   : chr [1:35] "M" "M" "M" "M" ...
##  $ L1    : chr [1:35] "English" "English" "English" "English" ...
##  $ Height: num [1:35] 69 69 69 69 69 62 62 62 62 62 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   ID = col_double(),
##   ..   Word = col_character(),
##   ..   Vowel = col_character(),
##   ..   F0 = col_double(),
##   ..   F1 = col_double(),
##   ..   F2 = col_double(),
##   ..   F3 = col_double(),
##   ..   Sex = col_character(),
##   ..   L1 = col_character(),
##   ..   Height = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

The function `head()` shows us the first few rows, so we can have an idea of its contents.

```
head(df)
```

```
## # A tibble: 6 x 10
##      ID Word  Vowel    F0    F1    F2    F3 Sex   L1      Height
##   <dbl> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <chr> <chr>    <dbl>
## 1     1 heed  i       120   319  2522  3299 M     English     69
## 2     1 hayed e       115   418  2119  3631 M     English     69
## 3     1 hawed a       115   558  1044  2677 M     English     69
## 4     1 who'd u       122   317   874  2668 M     English     69
## 5     1 hoed  o       119   457  1044  2587 M     English     69
## 6     2 heed  i       180   383  2756  3159 F     English     62
```

The function `tail()` gives us the last few rows.

```
tail(df)
```

```
## # A tibble: 6 x 10
##      ID Word  Vowel    F0    F1    F2    F3 Sex   L1      Height
##   <dbl> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <chr> <chr>    <dbl>
## 1     6 hoed  o      122    385   963  2952 M     English     62
## 2     7 heed  i      180.  2756  3616  3930 <NA>  spanish     68
## 3     7 hayed e      180.  2756  3343  3972 <NA>  spanish     68
## 4     7 hawed a      179.  1204  2819  4056 <NA>  spanish     68
## 5     7 who'd u      198.  1393  2798  4098 <NA>  spanish     68
## 6     7 hoed  o      184.  1037  2693  3700 <NA>  spanish     68
```

We can also simply ask for the dimensions of the data frame (table!) with the `dim()` function. This tells us the number of **rows** (i.e. observations) by the number of **columns** (i.e. variables).

```
dim(df)
```

```
## [1] 35 10
```

Recall from class that every data has a data type. We can ask R to tell us the data type with the `class()` function:

```
class(df)
```

```
## [1] "spec_tbl_df" "tbl_df"      "tbl"         "data.frame"
```

Or if you want to know the data type of a specific column in the table.

```
class(df$ID)
```

```
## [1] "numeric"
```

The $ in R is used to access specific columns! You can access each column by its name, e.g.:

```
df$Word
```

```
##  [1] "heed"  "hayed" "hawed" "who'd" "hoed"  "heed"  "hayed" "hawed" "who'd"
## [10] "hoed"  "heed"  "hayed" "hawed" "whod"  "hoed"  "heed"  "heyed" "hawed"
## [19] "who'd" "hoed"  "heed"  "hayed" "hawed" "who'd" "hoed"  "heed"  "hayed"
## [28] "hawed" "who'd" "hoed"  "heed"  "hayed" "hawed" "who'd" "hoed"
```

Working with the data

Overview

Above we had a quick look at the raw data, which you can also do directly with RStudio by clicking on the dataframe in the Environment tab.

There are lots of ways of doing preliminary data analysis and one great way is to visualize the data!

Recall our discussion of how the International Phonetic Alphabet (IPA) vowel chart looks – head faces left, vertical access is the jaw's height (closed to open) and the horizontal access is the position of the tongue (front to back in the mouth).

Single speaker

Let's look at one speaker in our sample. This is where the data manipulation R package dplyr comes in handy! We will talk about how to use these functions in class.

The dplyr package has several functions that allow us to `filter()` rows and to `select()` columns (among many other useful things!).

Let's filter out a single speaker and plot their vowels. You can set the ID to yourself!

```
single_speaker <- df %>% filter(ID == 5)
```

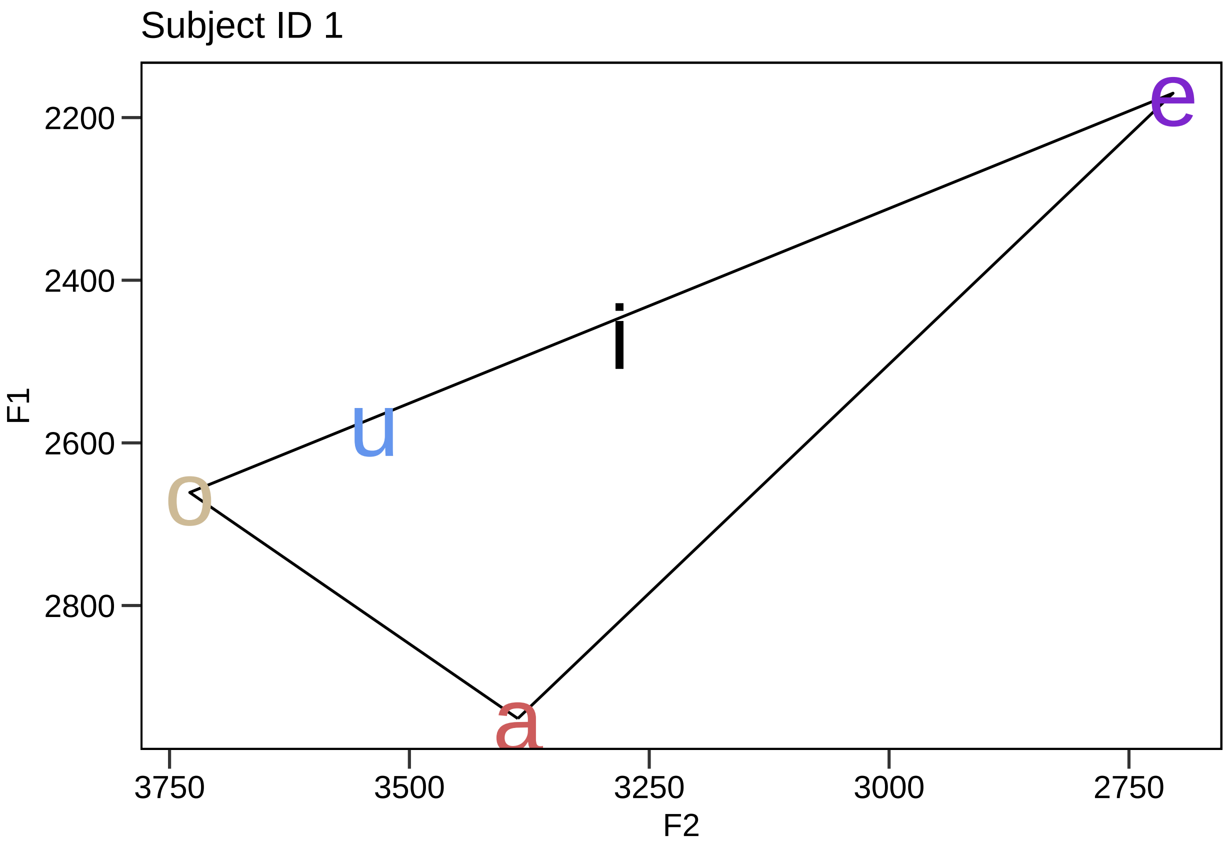
Above, we have saved the output of filtering by `ID == 1` (read "subject ID equals one") to a new data frame and named it `single_speaker`. Let's have a look.

```
single_speaker
```

```
## # A tibble: 5 x 10
##      ID Word  Vowel    F0    F1    F2    F3 Sex   L1      Height
##   <dbl> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <chr> <chr>    <dbl>
## 1     5 heed  i      169.  2469  3281  3772 M     Spanish     67
## 2     5 hayed e      142.  2170  2704  3750 M     Spanish     67
## 3     5 hawed a      133.  2939  3387  4583 M     Spanish     67
## 4     5 who'd u      144.  2576  3537  4220 M     Spanish     67
## 5     5 hoed  o      127.  2661  3729  4583 M     Spanish     67
```

Now let's plot Subject ID 1's vowel formant values for F1 and F2.

```r
ggplot(single_speaker, aes(x = F2, y = F1, color = Vowel)) +
  geom_convexhull(alpha = 0, colour = "black") +
  geom_text(aes(label = Vowel), size = 12) +
  scale_x_reverse() +
  scale_y_reverse() +
  coord_cartesian() +
  theme_pubr(border = TRUE, legend = "none") +
  theme(axis.ticks.length = unit(.25, "cm")) +
  scale_color_manual(
    name = "Vowel",
    values = c(
      "a" = "indianred",
      "i" = "black",
      "u" = "cornflowerblue",
      "o" = "wheat3",
      "e" = "purple3"
    )
  ) +
  ggtitle('Subject ID 1')
```



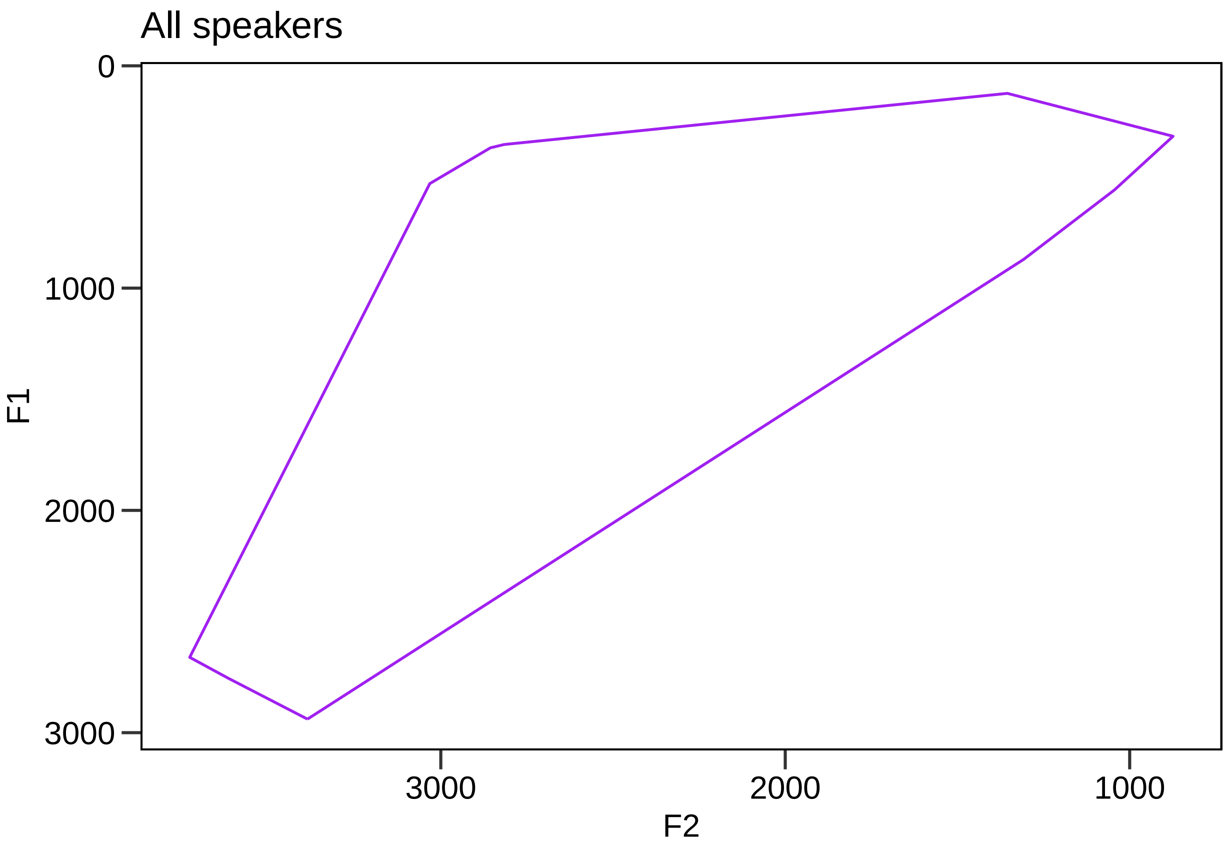
```r
summary(df)
```

```
##        ID         Word              Vowel                 F0       
##  Min.   :1   Length:35          Length:35          Min.   :115.0  
##  1st Qu.:2   Class :character   Class :character   1st Qu.:123.5  
##  Median :4   Mode  :character   Mode  :character   Median :142.3  
##  Mean   :4                                         Mean   :154.9  
##  3rd Qu.:6                                         3rd Qu.:182.4  
##  Max.   :7                                         Max.   :207.0  
##        F1               F2             F3            Sex           
##  Min.   : 124.0   Min.   : 874   Min.   :2307   Length:35         
##  1st Qu.: 344.5   1st Qu.:1208   1st Qu.:2824   Class :character  
##  Median : 461.0   Median :2268   Median :3104   Mode  :character  
##  Mean   : 925.0   Mean   :2132   Mean   :3287                     
##  3rd Qu.:1120.5   3rd Qu.:2818   3rd Qu.:3736                     
##  Max.   :2939.0   Max.   :3729   Max.   :4583                     
##       L1                Height     
##  Length:35          Min.   :62.00  
##  Class :character   1st Qu.:62.00  
##  Mode  :character   Median :67.00  
##                     Mean   :66.43  
##                     3rd Qu.:69.00  
##                     Max.   :70.00
```

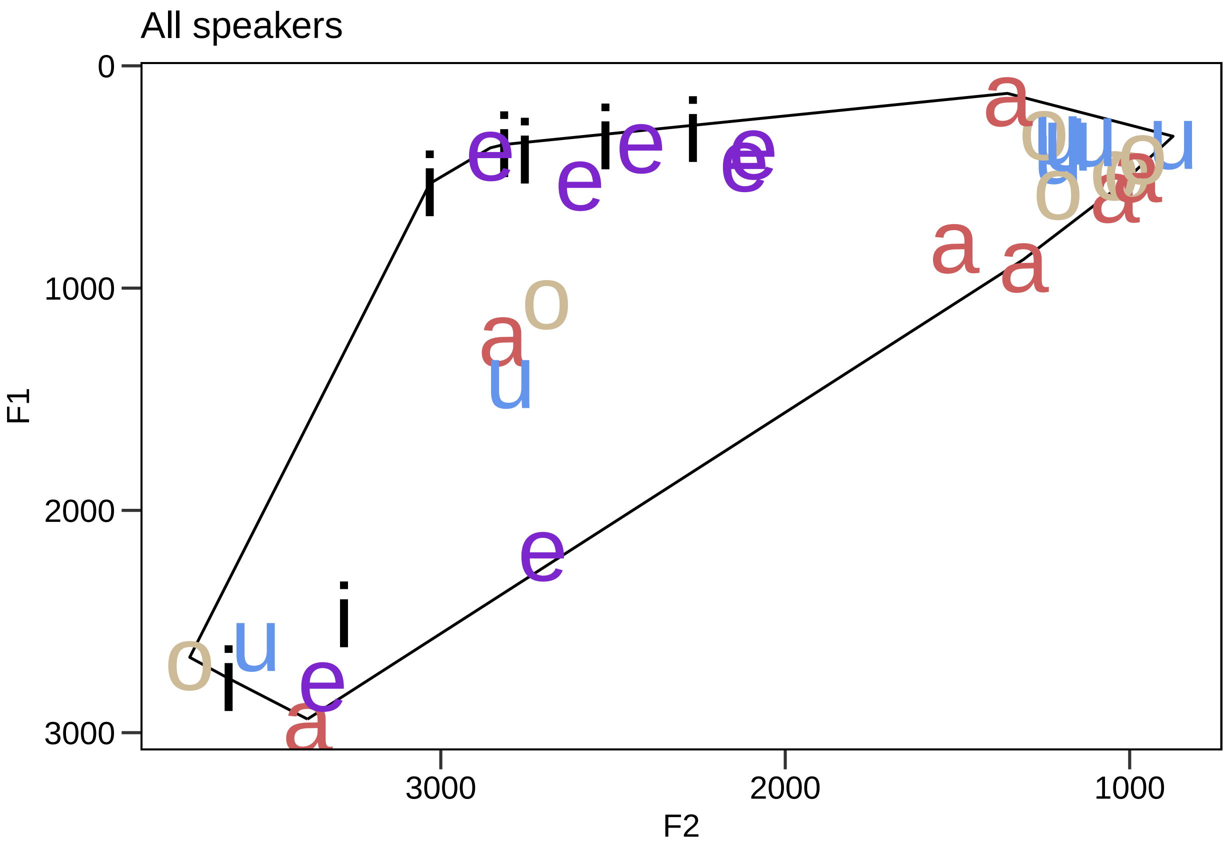
All speakers

Let's plot all of the F1 and F2 formants in our sample and see what they look like. We use the `ggplot()` function with lots of other functions. This is complicated – do not worry, we will go through this mess in class.

```r
ggplot(df, aes(x = F2, y = F1, color = Vowel)) +
  geom_convexhull(alpha = 0, colour = "purple") +
  scale_x_reverse() +
  scale_y_reverse() +
    coord_cartesian() +
  theme_pubr(border = TRUE, legend = "none") +
  theme(axis.ticks.length = unit(.25, "cm")) +
  scale_color_manual(
    name = "Vowel",
    values = c(
      "a" = "indianred",
      "i" = "black",
      "u" = "cornflowerblue",
      "o" = "wheat3",
      "e" = "purple3"
    )
  ) +
  ggtitle('All speakers')
```



```r
ggplot(df, aes(x = F2, y = F1, color = Vowel)) +
  geom_convexhull(alpha = 0, colour = "black") +
  geom_text(aes(label = Vowel), size = 12) +
  scale_x_reverse() +
  scale_y_reverse() +
  coord_cartesian() +
  theme_pubr(border = TRUE, legend = "none") +
  theme(axis.ticks.length = unit(.25, "cm")) +
  scale_color_manual(
    name = "Vowel",
    values = c(
      "a" = "indianred",
      "i" = "black",
      "u" = "cornflowerblue",
      "o" = "wheat3",
      "e" = "purple3"
    )
  ) +
  ggtitle('All speakers')
```



This isn't a very normal looking vowel chart! We seem to have some outliers and/or mistakes in formant measurements, e.g., the "e" vowel in the bottom left corner is way too high in terms of F1 and F2 frequencies.

Filter out outliers

If we have outliers in our data, we can filter them out for exploratory purposes. For example, perhaps we find some data points (observations) that are missing data or perhaps some mistakes were made in the data collection. (Note that you should never remove outliers in real scientific experiments!)

First, let's check that all subjects have recorded all five vowels.

This code is a bit complex, but we will discuss it in class. By using `dplyr` we can "pipe" data frames (table data) into functions and manipulate the data.

Here we group by `ID`, i.e., we group each subjects responses into their own groups, and then we count (by "summarizing") how many rows (`n()` function) are in each group – and we do so by creating a new column called `vowels`.

```
df %>% group_by(ID) %>% summarize(vowels = n())
```

```
## # A tibble: 7 x 2
##      ID vowels
##   <dbl>  <int>
## 1     1      5
## 2     2      5
## 3     3      5
## 4     4      5
## 5     5      5
## 6     6      5
## 7     7      5
```

```
df %>% group_by(Vowel) %>% select(Vowel, F1) %>% summarize(count = n())
```

```
## # A tibble: 5 x 2
##   Vowel count
##   <chr> <int>
## 1 a         7
## 2 e         7
## 3 i         7
## 4 o         7
## 5 u         7
```

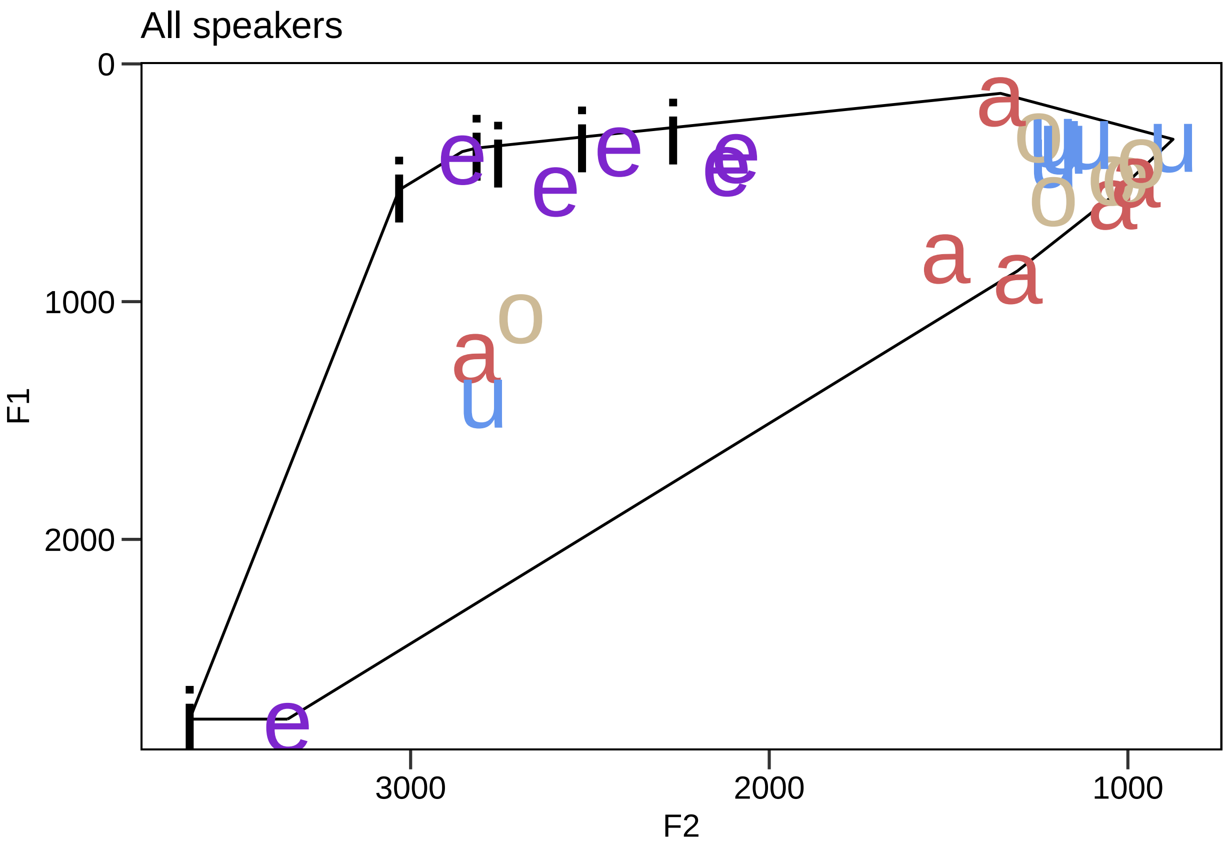
We will pretend that subject ID 2 does not have 5 observations. What we can do is remove that subject from our data set like this:

```
temp <- df %>% filter(ID != 5)
```

We saved the new data frame in a "temporary" data frame, so that we still have the full data in the `df` data frame.

Now let's revisualize what's left. We can use the same code we used above.

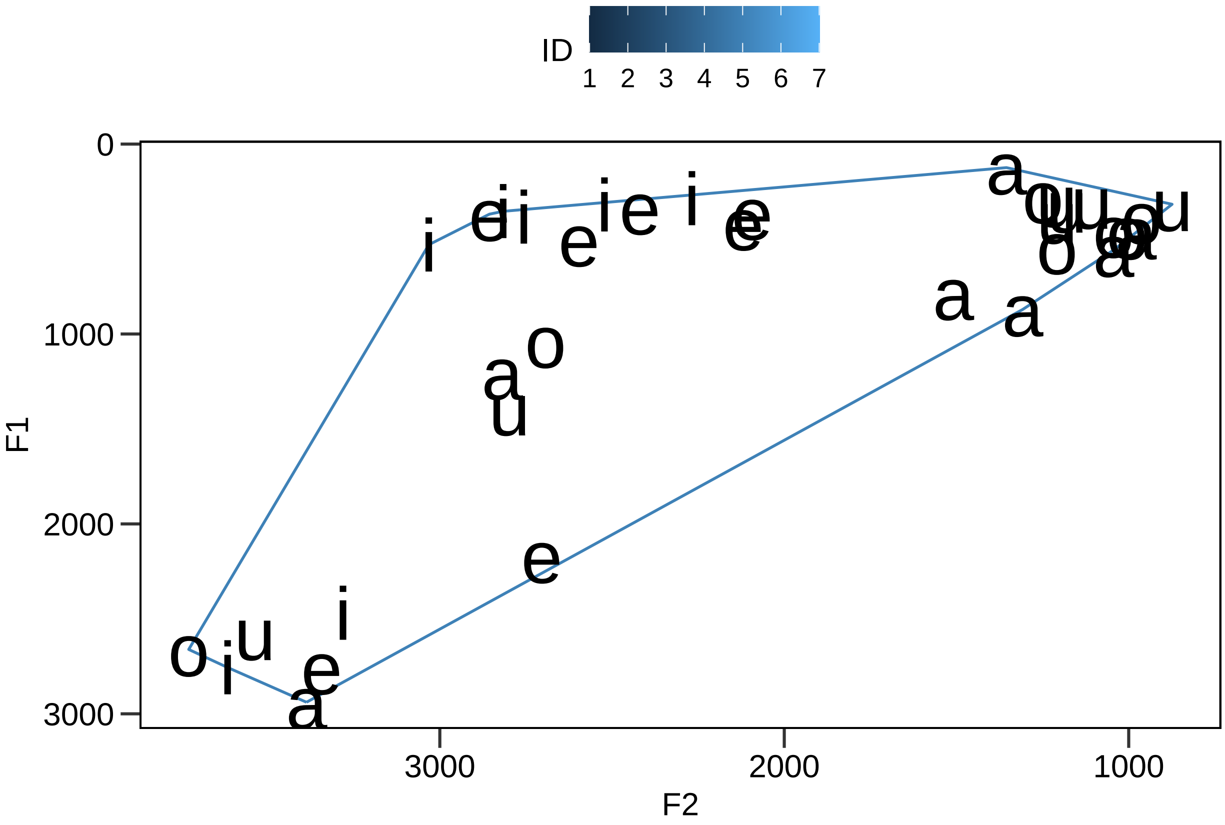
```
ggplot(temp, aes(x = F2, y = F1, color = Vowel)) +
  geom_convexhull(alpha = 0, colour = "black") +
  geom_text(aes(label = Vowel), size = 12) +
  scale_x_reverse() +
  scale_y_reverse() +
  coord_cartesian() +
  theme_pubr(border = TRUE, legend = "none") +
  theme(axis.ticks.length = unit(.25, "cm")) +
  scale_color_manual(
    name = "Vowel",
    values = c(
      "a" = "indianred",
      "i" = "black",
      "u" = "cornflowerblue",
      "o" = "wheat3",
      "e" = "purple3"
    )
  ) +
  ggtitle('All speakers')
```



See how the vowel chart changes. If we have great or more outliers, we will see more change.

For each speaker, we can add the vowel polygon lines.

```
ggplot(df, aes(x = F2, y = F1)) +
  geom_convexhull(alpha = 0, aes(colour = ID)) +
  geom_text(aes(label = Vowel), size = 10) +
  scale_x_reverse() +
  scale_y_reverse() +
  coord_cartesian() +
  theme_pubr(border = TRUE) +
  theme(axis.ticks.length = unit(.25, "cm"))
```

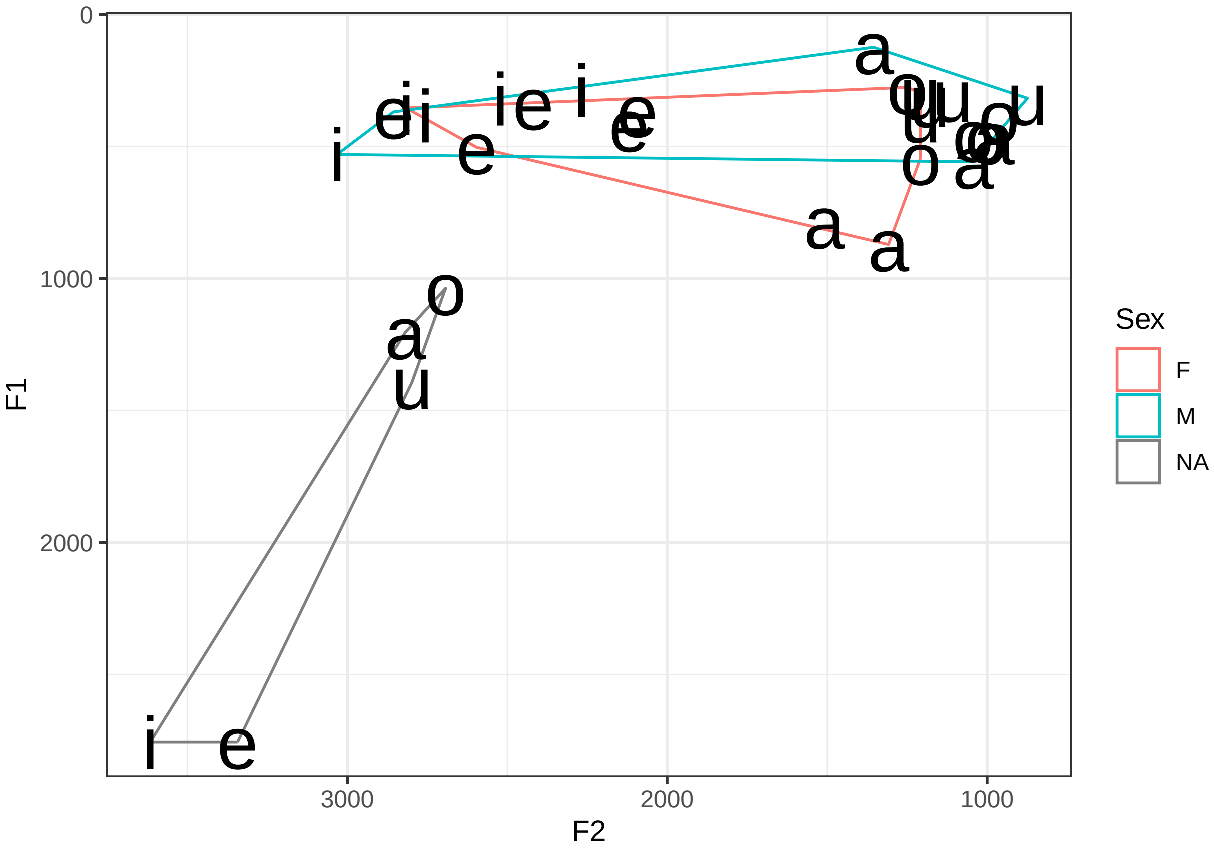


Analyses

Phonetic differences by sex

Let's plot individual vowel polygons by the reported sex of the speaker. As we expect from the literature on phonetic differences between male and female speech, the vowel polygon of the women in our sample displays greater acoustic range than the men.

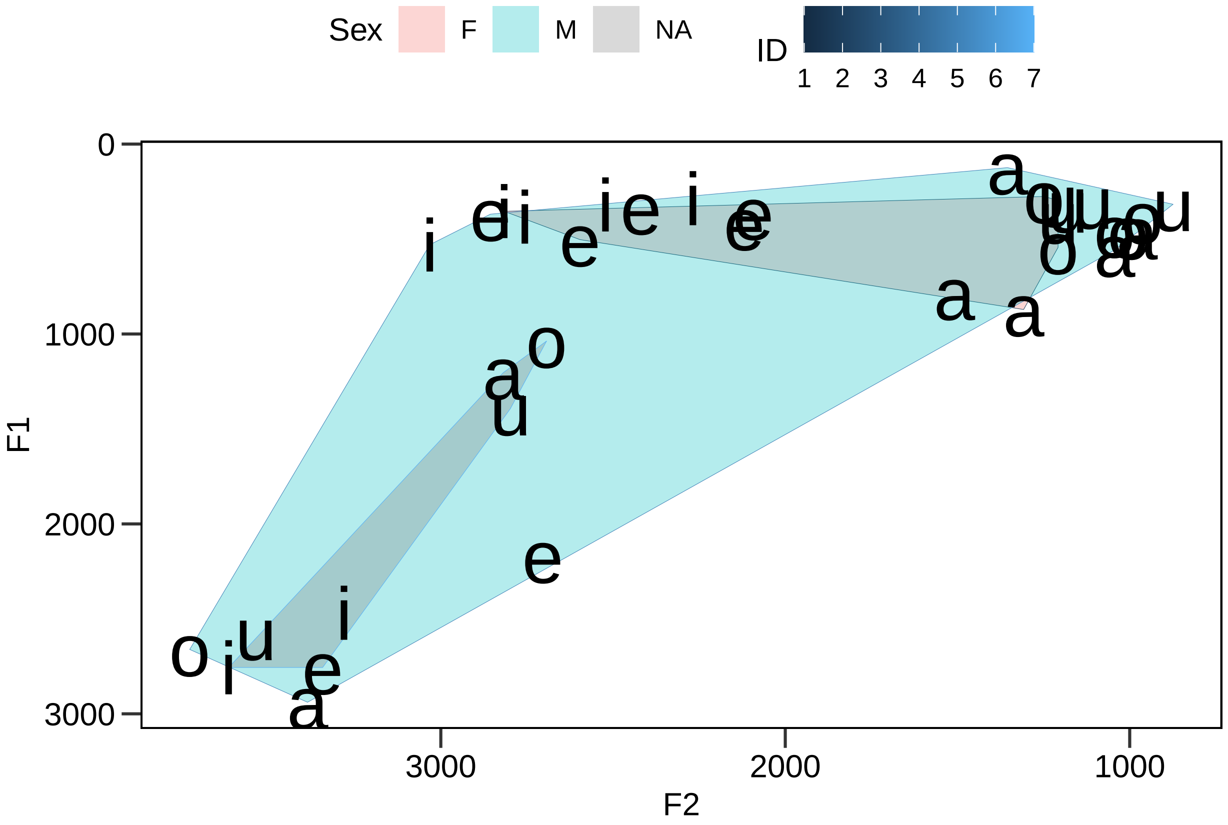
```
temp <- df %>% filter(ID != 5)
ggplot(temp, aes(x = F2, y = F1)) +
  geom_convexhull(alpha = 0, aes(colour = Sex)) +
  geom_text(aes(label = Vowel), size = 10) +
  scale_x_reverse() +
  scale_y_reverse() +
  coord_cartesian() +
  theme_pubr(border = TRUE) +
  theme(axis.ticks.length = unit(.25, "cm")) +
  theme_bw()
```



We can also plot individual vowel polygon spaces by sex and filled in by color.

```
ggplot(df, aes(x = F2, y = F1)) +
  geom_convexhull(alpha = 0.3, lwd = 0, aes(colour = ID, fill = Sex)) +
  geom_text(aes(label = Vowel), size = 10) +
  scale_x_reverse() +
  scale_y_reverse() +
  coord_cartesian() +
  theme_pubr(border = TRUE) +
  theme(axis.ticks.length = unit(.25, "cm"))
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
```



Differences in pitch

Recall that we also recorded our F0 – our fundamental frequency, which we perceive as our pitch. This is in the F0 column of our data frame:

```
head(df)
```

```
## # A tibble: 6 x 10
##      ID Word  Vowel    F0    F1    F2    F3 Sex   L1      Height
##   <dbl> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <chr> <chr>    <dbl>
## 1     1 heed  i       120   319  2522  3299 M     English     69
## 2     1 hayed e       115   418  2119  3631 M     English     69
## 3     1 hawed a       115   558  1044  2677 M     English     69
## 4     1 who'd u       122   317   874  2668 M     English     69
## 5     1 hoed  o       119   457  1044  2587 M     English     69
## 6     2 heed  i       180   383  2756  3159 F     English     62
```

Again, we access any column in the data frame with the $ operator with the name of the column as the suffix, e.g. let's get all values (aka observations, rows) of F0:

```
df$F0
```

```
##  [1] 120.0 115.0 115.0 122.0 119.0 180.0 181.0 177.0 201.0 185.0 201.0 203.0
## [13] 205.0 207.0 206.0 126.0 127.0 124.0 126.0 124.0 169.3 142.3 132.9 143.6
## [25] 127.3 122.0 133.0 120.0 123.0 122.0 180.4 180.2 179.3 198.4 183.8
```

We can also use various R functions on columns. For example, you can summarize numerical values from a column with the `summary()` function!

```r
summary(df$F0)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   115.0   123.5   142.3   154.9   182.4   207.0
```

Perhaps we want to summarize the F0 by sex? In “base R”:

```r
# Get a column by name
df$F0
```

```
##  [1] 120.0 115.0 115.0 122.0 119.0 180.0 181.0 177.0 201.0 185.0 201.0 203.0
## [13] 205.0 207.0 206.0 126.0 127.0 124.0 126.0 124.0 169.3 142.3 132.9 143.6
## [25] 127.3 122.0 133.0 120.0 123.0 122.0 180.4 180.2 179.3 198.4 183.8
```

```r
# Get rows by filtering the contents of a colum; here by "sex is F(emale)"
df[df$Sex == "F", ]
```

```
## # A tibble: 15 x 10
##       ID Word  Vowel    F0    F1    F2    F3 Sex   L1      Height
##    <dbl> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <chr> <chr>    <dbl>
##  1     2 heed  i       180   383  2756  3159 F     English     62
##  2     2 hayed e       181   503  2596  2937 F     English     62
##  3     2 hawed a       177   785  1509  2817 F     English     62
##  4     2 who'd u       201   383  1208  2837 F     English     62
##  5     2 hoed  o       185   544  1208  2817 F     English     62
##  6     3 heed  i       201   354  2816  3392 F     English     67
##  7     3 hayed e       203   335  2419  2995 F     English     67
##  8     3 hawed a       205   871  1308  3074 F     English     67
##  9     3 whod  u       207   296  1209  3034 F     English     67
## 10     3 hoed  o       206   276  1249  2975 F     English     67
## 11    NA <NA>  <NA>     NA    NA    NA    NA <NA>  <NA>        NA
## 12    NA <NA>  <NA>     NA    NA    NA    NA <NA>  <NA>        NA
## 13    NA <NA>  <NA>     NA    NA    NA    NA <NA>  <NA>        NA
## 14    NA <NA>  <NA>     NA    NA    NA    NA <NA>  <NA>        NA
## 15    NA <NA>  <NA>     NA    NA    NA    NA <NA>  <NA>        NA
```

```r
# Get just the contents of the column we want by the filter -- ugh this is painful!
df[df$Sex == "F", ]$F0
```

```
##  [1] 180 181 177 201 185 201 203 205 207 206  NA  NA  NA  NA  NA
```

```r
# Now let's summarize that mess
summary(df[df$Sex == "F", ]$F0)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
##   177.0   182.0   201.0   194.6   204.5   207.0       5
```

```r
# And the men
summary(df[df$Sex == "M", ]$F0)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
##   115.0   121.5   124.0   127.7   128.7   169.3       5
```

Men in the sample have on average lower pitch.

Pitch versus height

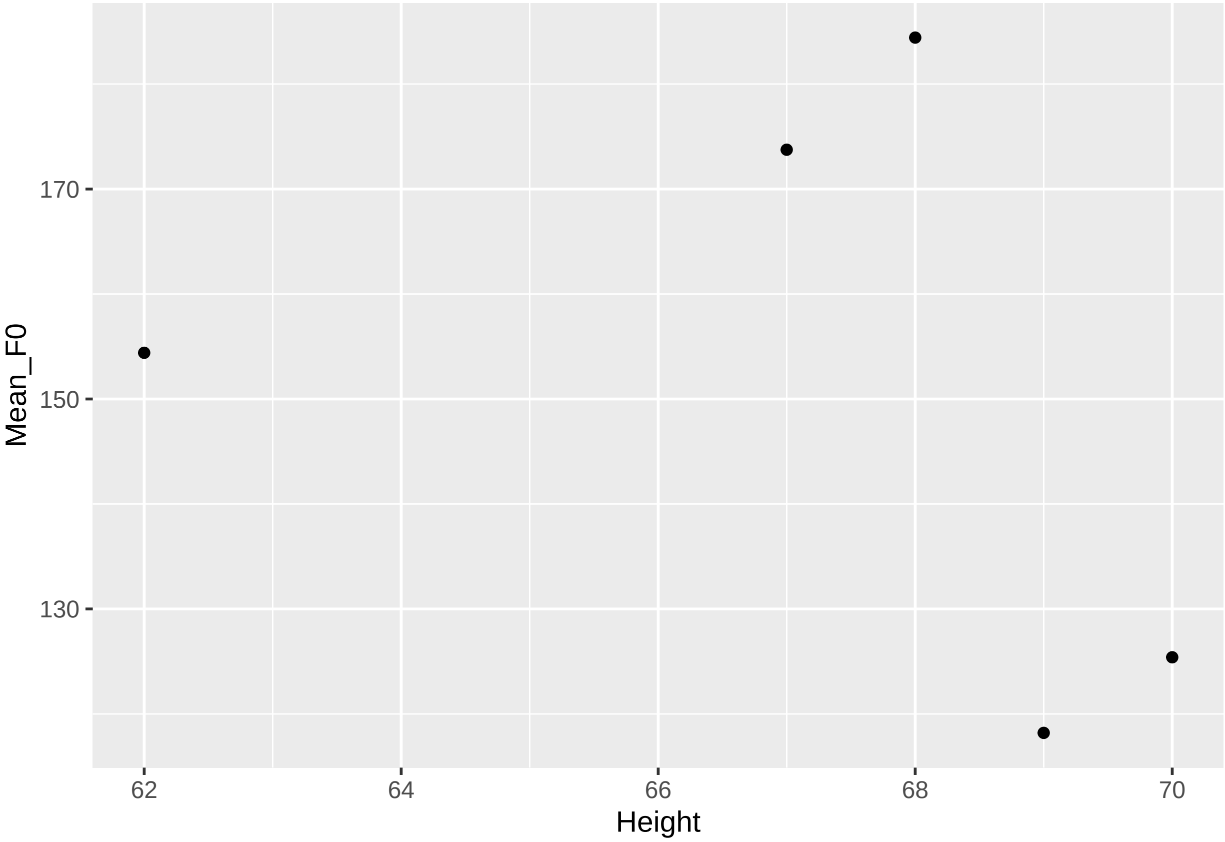
Lastly, let’s see if there is a correlation between one’s height and one’s F0.

First, let's get the data we want by "grouping" the height variable and summarizing the mean F0 for each by height.

```
tmp <- df %>% group_by(Height) %>% summarize(Mean_F0 = mean(F0))
```

Now let's plot the results as a linear regression.

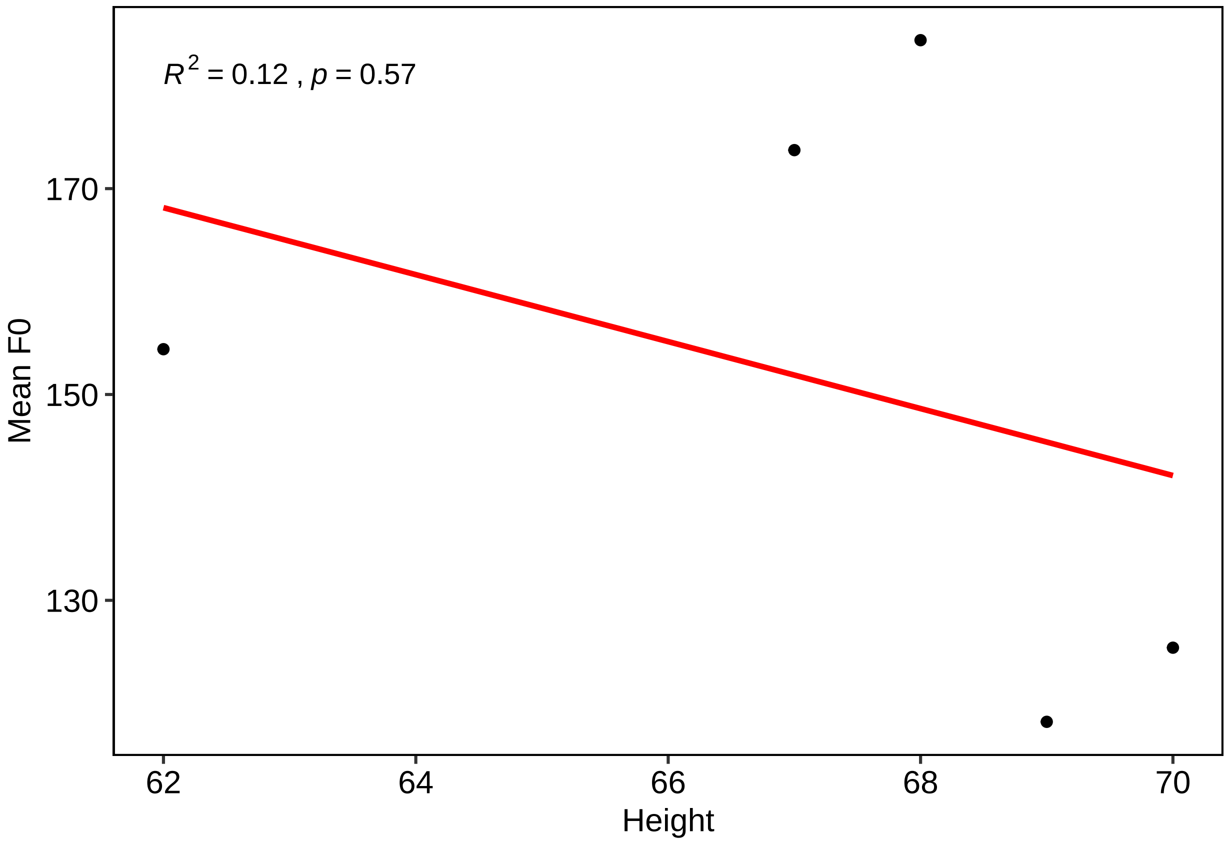
```
ggplot(tmp, aes(x=Height, y=Mean_F0)) +
  geom_point()
```



```
ggplot(tmp, aes(Height, Mean_F0)) +
  geom_point(colour = "black", alpha = 1) +
  xlab("Height") +
  ylab("Mean F0") +
  theme_pubr(border = TRUE, margin = TRUE) +
  geom_smooth(method = "lm", se = FALSE, colour = "red") +
  ggpubr::stat_cor(aes(label = paste(..rr.label.., ..p.label.., sep = "~`,`~")))
```

```
## Warning: The dot-dot notation (`..rr.label..`) was deprecated in ggplot2 3.4.0.
## i Please use `after_stat(rr.label)` instead.

## `geom_smooth()` using formula = 'y ~ x'
```



And indeed it looks like as one gets taller, their pitch on average gets lower.

We can also get the model statistics directly from the data with these functions.

```
model1 <- lm(Mean_F0 ~ Height, data = tmp)
summary(model1)
```

```
##
## Call:
## lm(formula = Mean_F0 ~ Height, data = tmp)
##
## Residuals:
##      1      2      3      4      5
## -13.75  21.86  35.79 -27.18 -16.72
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  369.826    339.506   1.089    0.356
## Height        -3.253      5.048  -0.644    0.565
##
## Residual standard error: 31.44 on 3 degrees of freedom
## Multiple R-squared:  0.1216, Adjusted R-squared:  -0.1712
## F-statistic: 0.4153 on 1 and 3 DF,  p-value: 0.5652
```