R Workshop

Lecture 3: `tidyverse` and plotting

1 Quick review

`tidyverse` and data munging questions/review. Briefly discuss talk about `head()`, summary() mentioned in the previous lecture notes.

2 Plotting your data

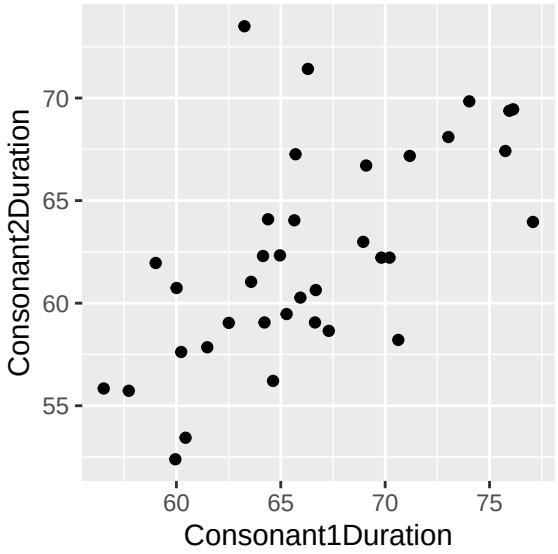
This is where **R** and, particularly, `tidyverse` really shine. The plots you get are really pretty, and super scalable (extendable to more complex cases/plots). The functions below are part of the `ggplot2` library, which is loaded when you load `tidyverse`.

3 Scatterplot

Say, you wanted to get a plot that sees how the values of "Consonant1Duration" change with those of "Consonant2Duration".

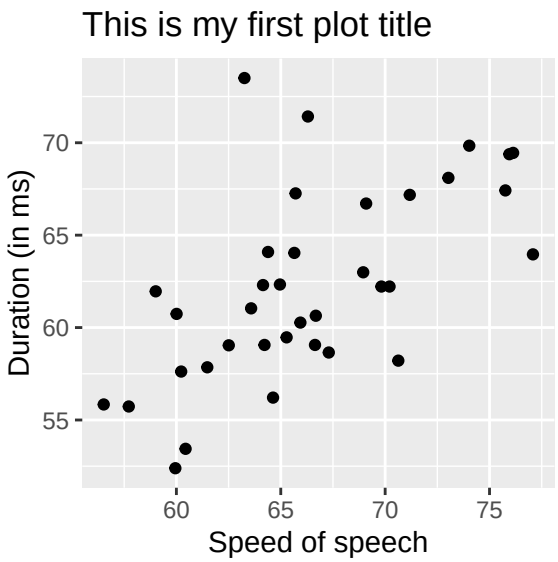
```
#setting the working directory
#Replace content with the relevant directory address
setwd("/.../.../.../")
```

```
#Opening the csv file and storing it to a (data.frame) variable
#The variable can be called anything, I am just calling it ''measurements''
measurements = read_csv("Lecture2-Measurements.csv")

## Rows: 36 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): Vowel, Speed
## dbl (3): Subject, Consonant1Duration, Consonant2Duration
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.

##Plot the values
ggplot(measurements, aes(x=Consonant1Duration,y=Consonant2Duration))+
  geom_point()
```



What if you want to give the plot a nice title and nice x-axis and y-axis labels?

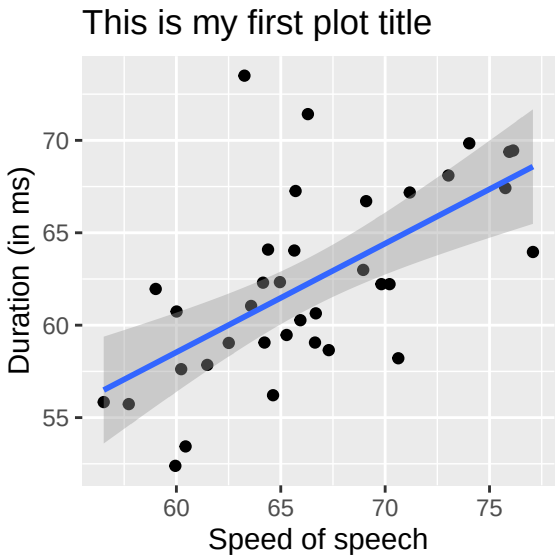
```
#Plotting
ggplot(measurements, aes(x=Consonant1Duration,y=Consonant2Duration))+
  geom_point()+
  ggtitle("This is my first plot title") +
  xlab("Speed of speech") +
  ylab("Duration (in ms)")
```



You can add a 'line of best fit' with the addition of another function.

```
##Plot the values
ggplot(measurements, aes(x=Consonant1Duration,y=Consonant2Duration))+
  geom_point()+
  geom_smooth(method="lm")+
  ggtitle("This is my first plot title") +
  xlab("Speed of speech") +
  ylab("Duration (in ms)")
```

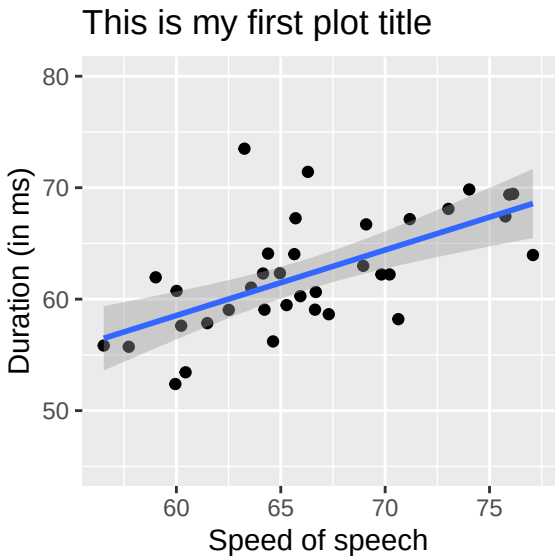
```
## `geom_smooth()` using formula = 'y ~ x'
```



What if you want to restrict the range of y-values to be between 45 and 80?

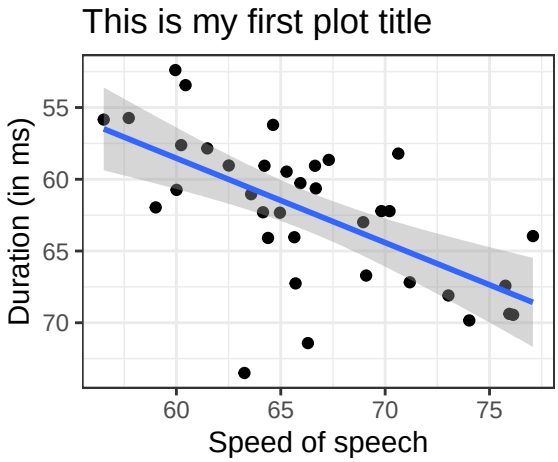
NOTE: yes, you can do this, but you need to make sure that you are not excluding values in the distribution.

```
#Plotting
ggplot(measurements, aes(x=Consonant1Duration,y=Consonant2Duration))+
  geom_point()+
  geom_smooth(method="lm")+
  ggtitle("This is my first plot title") +
  xlab("Speed of speech") +
  ylab("Duration (in ms)") +
  ylim(45,80)

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



What if you don't like the background theme and want to flip the y-axis orientation?

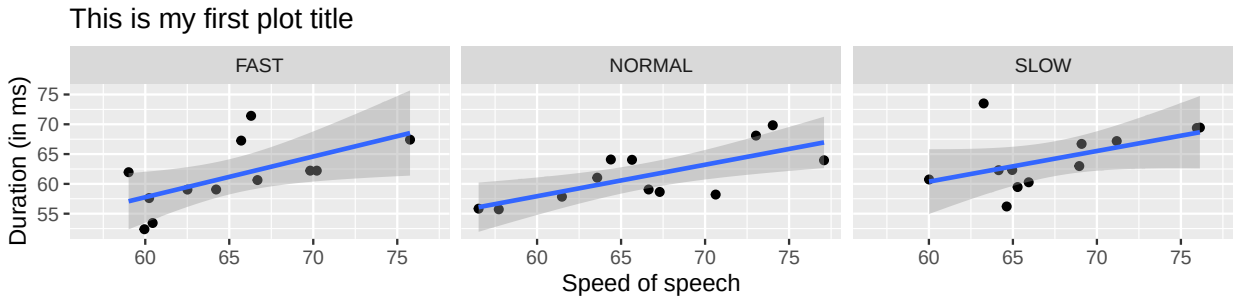
```
#Plotting
ggplot(measurements, aes(x=Consonant1Duration,y=Consonant2Duration))+
  geom_point()+
  geom_smooth(method="lm")+
  ggtitle("This is my first plot title") +
  xlab("Speed of speech") +
  ylab("Duration (in ms)") +
  ylim(45,80)+
  theme_bw()+
  scale_y_reverse()

## Scale for y is already present.
## Adding another scale for y, which will replace the existing scale.
## `geom_smooth()` using formula = 'y ~ x'
```



You can also try `theme_classic()`. There are other themes too, and you can make your own theme if you want.

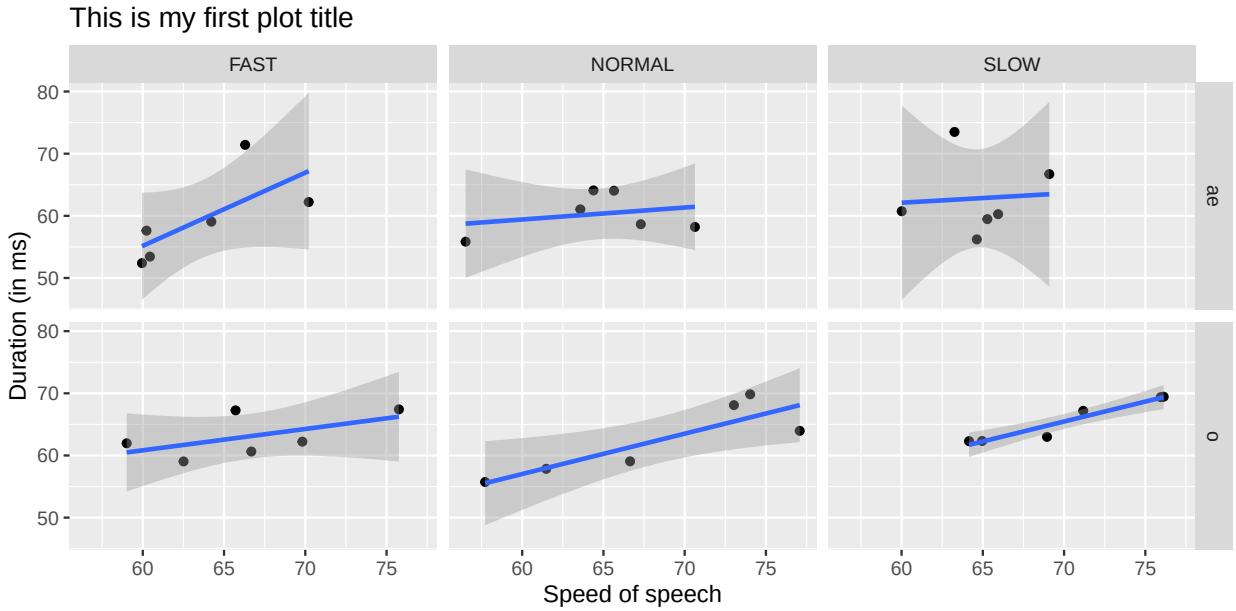
What if you wanted to see how the two measures vary separately for each "Speed"?

```
##Plot the values
ggplot(measurements, aes(x=Consonant1Duration,y=Consonant2Duration))+
  geom_point()+
  geom_smooth(method="lm")+
  ggtitle("This is my first plot title") +
  xlab("Speed of speech") +
  ylab("Duration (in ms)") +
  facet_grid(~Speed)

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



What if you wanted to see how the two measures vary separately for each "Speed' and for each "Vowel'?

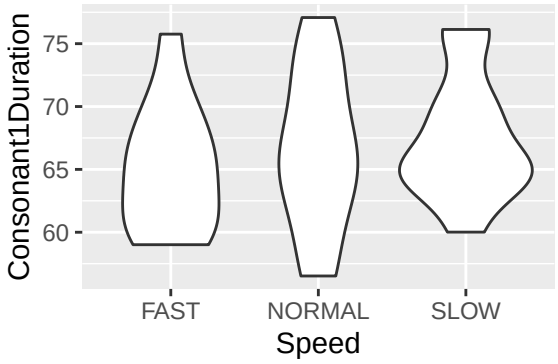
```
##Plot the values
ggplot(measurements, aes(x=Consonant1Duration,y=Consonant2Duration))+
  geom_point()+
  geom_smooth(method="lm")+
  ggtitle("This is my first plot title") +
  xlab("Speed of speech") +
  ylab("Duration (in ms)") +
  facet_grid(Vowel~Speed)

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



3.1 Violin plots

Say, you wanted to get a distribution of the values in "Consonant1Duration" for each "Speed'.

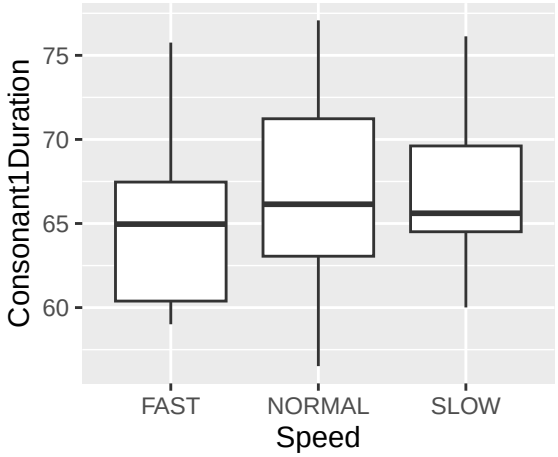
```
##Plot the values
ggplot(measurements, aes(x=Speed,y=Consonant1Duration))+
  geom_violin()
```



3.2 Box plots

Say, you wanted to get a box-plot for the mean value of "Consonant1Duration" at each "Speed": Say, you wanted to get a distribution of the values in "Consonant1Duration" for each ''Speed'.

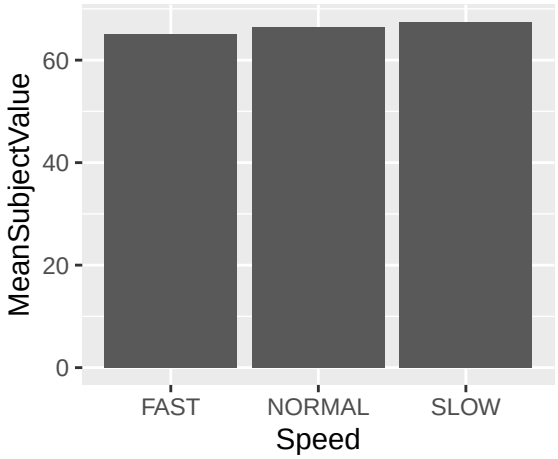
```
##Plot the values
ggplot(measurements, aes(x=Speed,y=Consonant1Duration))+
  geom_boxplot()
```



3.3 Bar plots

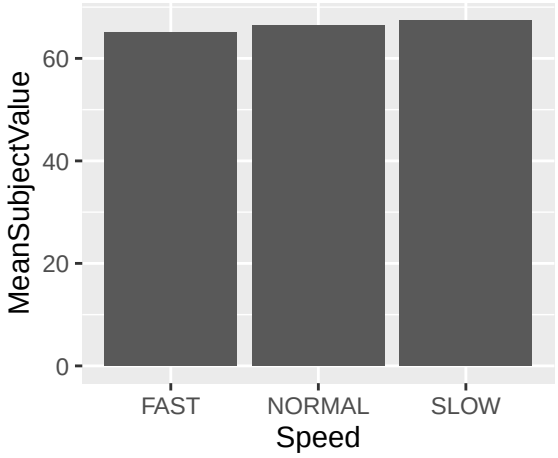
Say, you wanted to get a bar-plot for the mean value of "Consonant1Duration" at each "Speed":

```
#First get the mean value of Consonant1Duration at each speed
measurements2 = measurements %>%
  group_by(Speed) %>%
  summarise(MeanSubjectValue = mean(Consonant1Duration))

#Then plot the values
ggplot(measurements2, aes(x=Speed,y=MeanSubjectValue))+
  geom_bar(stat="identity")
```



You can infact combine the data analysis steps and the plotting steps. But, remember, the pipe for data analysis is %>%, and for plotting using `ggplot2` functions is +.

```
#Bar-plot for the mean value of Consonant1Duration at each speed
measurements %>%
  group_by(Speed) %>%
  summarise(MeanSubjectValue = mean(Consonant1Duration)) %>%
  ggplot(aes(x=Speed,y=MeanSubjectValue)) +
  geom_bar(stat="identity")
```



3.4 Saving your plots

Ok, you are done with the plotting, and you want to save your plot. You can do it manually using the options in the plotting window, but you could also automate it:

```
#If you want to save the plot as a file, first it has to be saved to a variable.
#Again, the variable can be called anything. I will call it "plot".
plot = ggplot(measurements, aes(x=Speed,y=Consonant1Duration))+
  geom_bar(stat="identity") +
  ggtitle("This is my second bar plot title") +
  xlab("Speed of speech") +
  ylab("Duration (in ms)") +
  coord_cartesian(ylim=c(600,850))
```

```r
#Now, set the directory where you want to save it
setwd("/.../.../.../")

#Now you can save it
#Note: There are a variety of plot filetypes - png, jpeg, tiff,... I suggest png.
ggsave("Plot.png",plot)

#You can also specify the dimensions of the saved plot
ggsave("Plot.png",plot,width=4,height=4,units="in")
```

4 Some useful references

You should keep a copy of the ggplot2 cheatsheet with you — it's super helpful!

5 Homework

1. Imagine you have a `data.frame` named **ExperimentData**, which has a column named *Response*, and you are trying to print out the 8^{th} value in the column. Write the r-code to do it. Note, there are multiple columns in the `data.frame`, and you don't know the column number.
2. What does a violin plot allow us to see?
3. List all the errors in the following piece of code, and then include the correct r-code:
   (a) When trying to loop through 6 times and printing "Hello World!" or "Good bye, World!" based on the value of x:

```r
x = 15
for(i in c(1:6)
    if(x = 10){
      print("Hello, World!")
      else
        print("Good bye, World!")
      }
```

   (b) When trying to exclude subjects numbered 10 or above from the *Subject* column from a `data.frame` named **ExperimentData** and then plotting a scatterplot for two columns *ConsonantDuration1* and *ConsonantDuration2*:

```r
ExperimentData %>%
  filter(Subject < 10) +
  ggplot(aes(ConsonantDuration1,ConsonantDuration2) +
         geom_scatterplot()
```

   (c) When trying to get a mean value for each *Subject* of a column named *ReactionTime* from a `data.frame` named **ExperimentData**, and plotting separate boxplots within the same plot area based on two columns, *Speed* and *ConsonantDuration2*:

```r
ExperimentData %>%
  group_by(Subject) %>%
  summarise(MeanReactionTime = mean(ReactionTime)) +
  ggplot(aes(x=Speed,y=ConsonantDuration1)+
         geom_boxplot()
```

4. Write a script using `tidyverse` functions and a single set of piped (or chained) `tidyverse` functions that does the following things. Make sure to comment your script properly, and include the output plot in the PDF.

   i. Open the file **Measurements.csv** that we used in class (uploaded to D2L).

   ii. Save the csv file to a `data.frame` **ExperimentData**.

   iii. Remove the rows that contain the value "Normal" in the *Speed* column.

   iv. Remove any rows where *Consonant2Duration* is greater than 70.

   v. Find the average *Consonant1Duration* for each *Speed* ("FAST" vs. "SLOW") for each *Subject*.

   vi. Plot a boxplot where the x-axis is *Speed* and the y-axis has the average values of *Consonant1Duration* that you calculated in the previous step.