Exploring Sentencing Data Notebook

```r
setwd('~/Desktop/Sentencing/')
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(palmerpenguins)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
library(plotly)
```

```
##
## Attaching package: 'plotly'
```

```
## The following object is masked from 'package:ggplot2':
##
##     last_plot
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following object is masked from 'package:graphics':
##
##     layout
```

```r
peng = penguins %>% na.omit()
knitr::opts_chunk$set(fig.width=4, fig.height=4)
```

Data Types

What is halfway between 0 and 1? It is 1/2. What is halfway between horse and dog? There is no such thing! Thinking about each type of data is very important so that we don't code silly things like the mean of animal species!

There are two main types of data: **categorical data** and **numerical data**. Some examples of categorical data would be color, ethnicity, employment status, or states/countries. These have unique values, like California or Oregon.

Some examples of numerical data are temperature, height and salary. It makes perfect sense to be 165.8 cm tall or for the temperature to be 82.4 degrees outside.

There are some confusing data types that use numbers to *represent* categorical data, like zip code. You may live in the zip code 90201, which is a number, but you can't live in the zip code 90210.3. Only whole numbers, and specific ones at that, make sense here. We will learn more about using numbers to represent categorical data in this lesson.

Penguins Data

Take a look at the penguin data. There are 8 columns in the table, each giving an attribute about penguins. Can you identify which data type each variable is?

```r
head(penguins)
```

```
## # A tibble: 6 x 8
##   species island bill_length_mm bill_depth_mm flipper_length_~ body_mass_g sex
##   <fct>   <fct>           <dbl>         <dbl>            <int>       <int> <fct>
## 1 Adelie  Torge~           39.1          18.7              181        3750 male
## 2 Adelie  Torge~           39.5          17.4              186        3800 fema~
## 3 Adelie  Torge~           40.3          18                195        3250 fema~
## 4 Adelie  Torge~           NA            NA                 NA          NA <NA>
## 5 Adelie  Torge~           36.7          19.3              193        3450 fema~
## 6 Adelie  Torge~           39.3          20.6              190        3650 male
## # ... with 1 more variable: year <int>
```

Explorations

A good practice to do with a new data set is to explore it through visualization. We will walk through a few graphs in R so you can see how to plot. We will examine each of these so we can see relationships and learn more about our data. Then, explore on your own by modifying this code!

How to make a scatter plot with quantitative variables

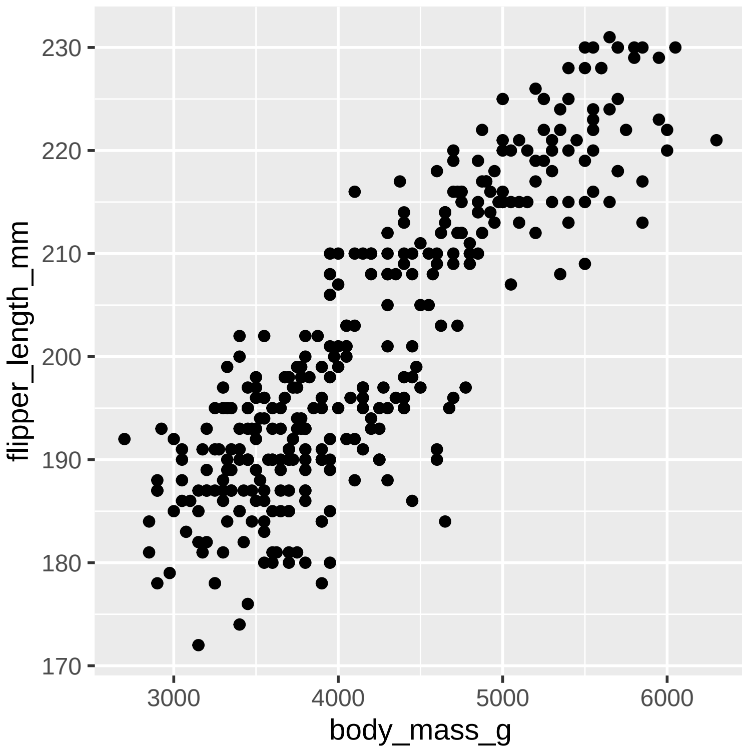
In this first plot, we will look at numerical variables only. We will see how each penguin's flipper length relates to its body mass.

Each dot in the scatterplot we produce with the ggplot command represents a row in our penguin data. Scatterplots help us to visualize and understand numerical data better. We will compare each penguin's body mass to their flipper length through visualization and observation. We will use this scatterplot to identify any patterns that exist in our penguin data set.

%%%%%%%% come back to this to explain R code %%%%%%%%%%

```r
ggplot(penguins, aes(x=body_mass_g, y=flipper_length_mm)) + geom_point()
```

```
## Warning: Removed 2 rows containing missing values (geom_point).
```



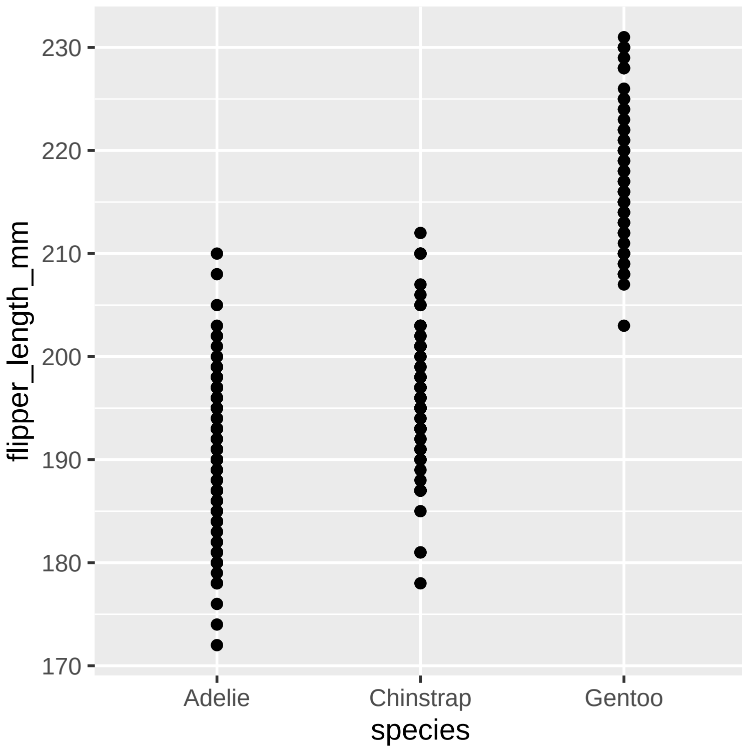
What do you notice in this scatterplot? What do you wonder?

What happens when you try to make a scatter plot with categorical variables?

We just created a scatterplot with two numerical variables. Now we will see what happens if one variable is numerical and the other is categorical? Run the code below tht plots species (a categorical variable) against flipper length (a numerical variable).

```
ggplot(penguins, aes(x=species, y=flipper_length_mm)) + geom_point()
```

```
## Warning: Removed 2 rows containing missing values (geom_point).
```

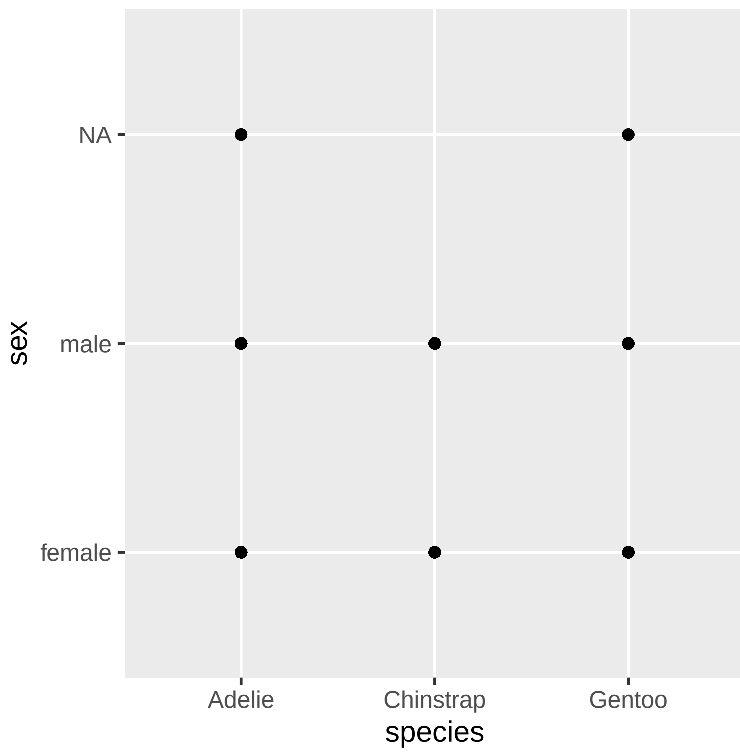


What do you notice in this scatterplot?

What do you wonder?

Let's try one more scenario. What happens when you try to plot two categorical variables against each other? Run the code below to plot sex againstagainst species (both categorical variables).

```
ggplot(penguins, aes(x=species, y=sex)) + geom_point()
```



What do you notice in this scatterplot?

What do you wonder?

Scatterplots of two categorical variables are not that useful for analysis and inference. In the remainder of this lesson we will focus on comparisons where we have at least one numerical variable.

Let's momentarily restrict our data to two data points to think more about linear patterns.

Fitting Lines to Data

- motivation on why we would do this in other contexts
- motivation on the simple case

Let's make two example points, one at (1,2) and one at (3,5). In R, we will save this into a data frame using a vector of the x values, 1 and 3, and a vector of the matching y values, 2 and 5.

```
twopoints <- data.frame(xvals = c(1,3), yvals = c(2,5))
head(twopoints)
```

```
##   xvals yvals
## 1     1     2
## 2     3     5
```

We can make a fairly boring plot of these two points.

```
twoplot <- ggplot(twopoints, aes(x=xvals, y=yvals)) +
  geom_point(color='red') +
  xlim(0,6) +
  ylim(0,6)
```

In order to create a linear regression on these two points, we can think back to algebra and use the formula for a line.

$$y = mx + b$$

Fitting our line to the data is easy, we can solve in a number of ways. We can plug both these points into the equation and then solve the system together.

$$2 = m + b$$

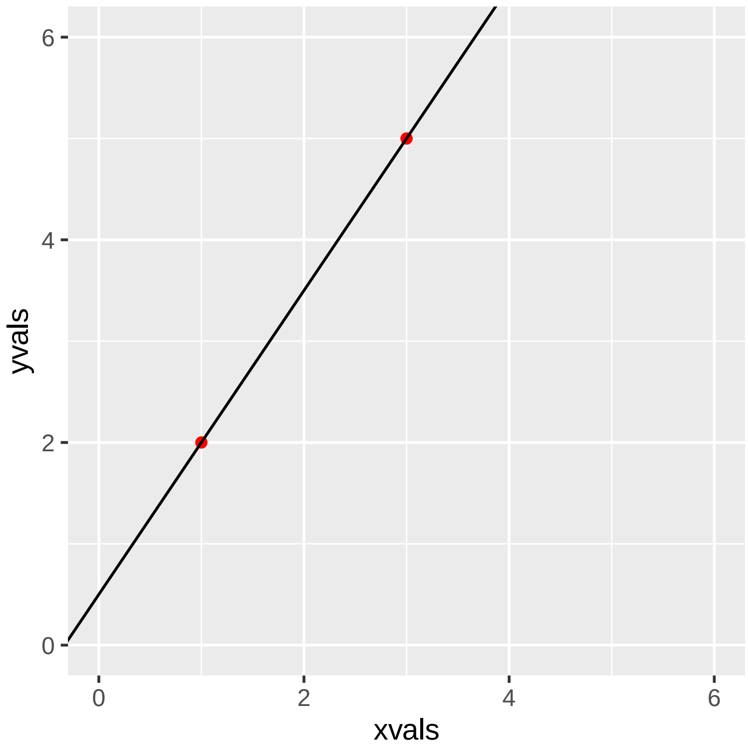
$$5 = 3m + b$$

Here we have a system that has two equations and two unknowns, m and b. We know this has a unique solution! We can solve this system using a variety of techniques. Try this using a technique you are comfortable with and verify that the solution passes through each of the two points.

$$y = \frac{3}{2}x + \frac{1}{2}$$

We can plot the results. Here we use the `abline()` function which can be done inslope-intercept form.

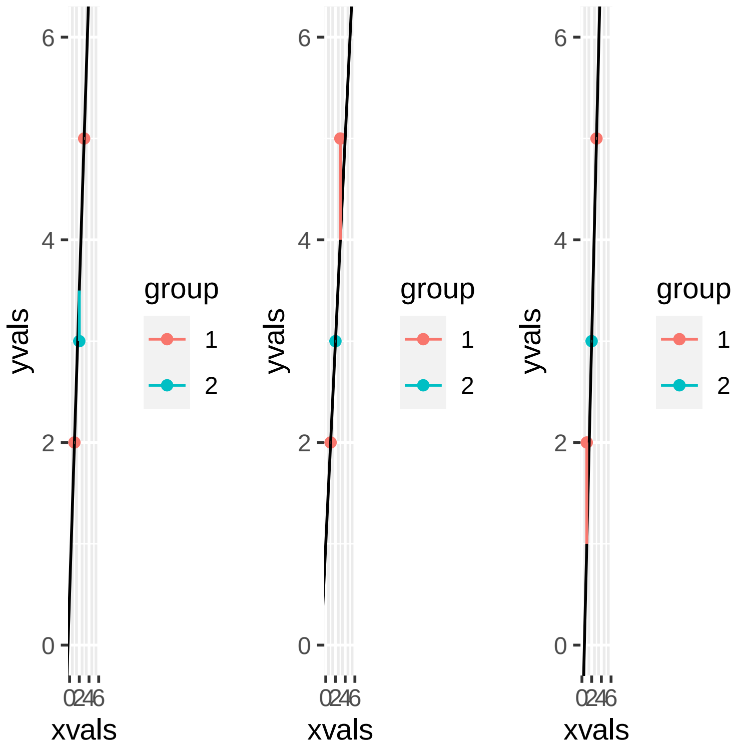
```
twoplot + geom_abline(slope=3/2, intercept = 1/2)
```




```
twolinear <- lm(formula = yvals ~ xvals, data=twopoints)
twolinear
```

```
##
## Call:
## lm(formula = yvals ~ xvals, data = twopoints)
##
## Coefficients:
## (Intercept)        xvals
##         0.5          1.5
```

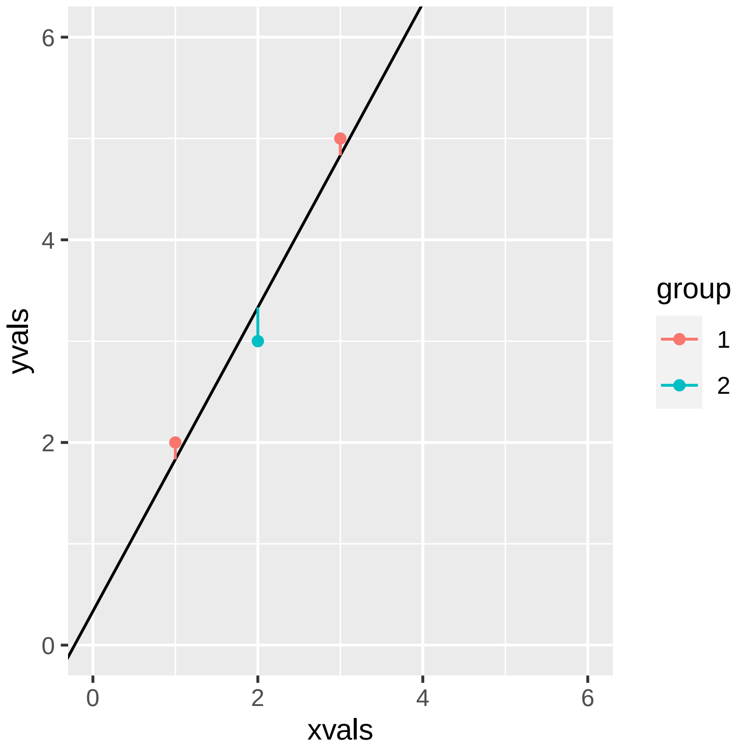
```
twopoints$group = as.factor(1)
threepoints = rbind(twopoints, data.frame(xvals = 2, yvals = 3, group=as.factor(2)))
threepoints$yfit1 = threepoints$xvals*3/2+1/2
threepoints$yfit2 = threepoints$xvals+1
threepoints$yfit3 = threepoints$xvals*2-1

threeplot = ggplot(threepoints, aes(x=xvals, y = yvals, color=group)) + geom_point()  + xlim(0,6) + ylim

grid.arrange(
threeplot + geom_abline(slope=3/2, intercept = 1/2)+  geom_segment(aes(xend = xvals, yend = yfit1)),
threeplot + geom_abline(slope=1, intercept = 1) + geom_segment(aes(xend = xvals, yend = yfit2)),
threeplot + geom_abline(slope=2, intercept = -1) +  geom_segment(aes(xend = xvals, yend = yfit3)),
ncol=3
)
```




```
threelinear = lm(formula=yvals~xvals, data=threepoints)
threelinear
```

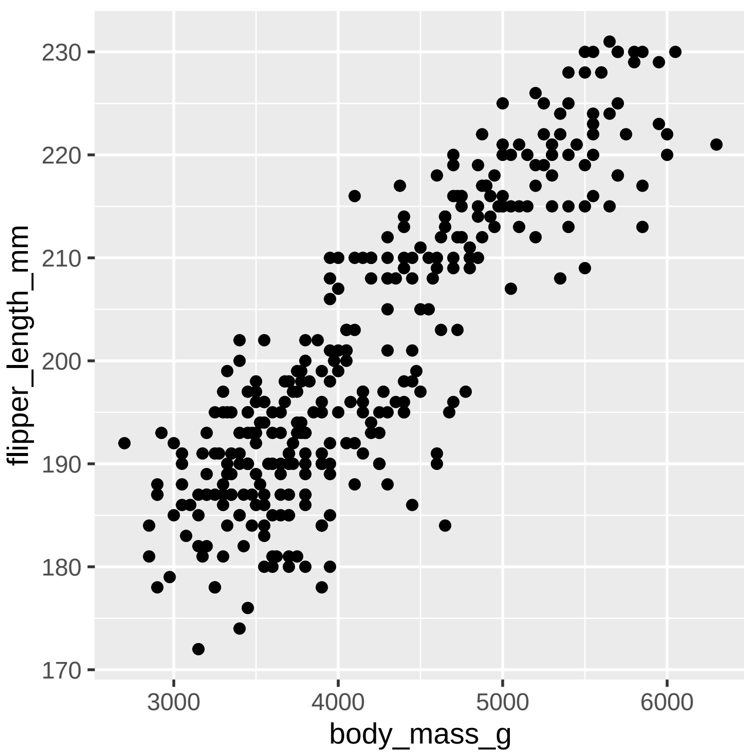
```
##
## Call:
## lm(formula = yvals ~ xvals, data = threepoints)
##
## Coefficients:
## (Intercept)        xvals
##      0.3333       1.5000
```

```
threepoints$linfit = 1.5*threepoints$xvals + 0.3333
ggplot(threepoints, aes(x=xvals, y = yvals, color=group)) +
  geom_point()  +
  xlim(0,6) + ylim(0,6)+
  geom_abline(slope=1.5, intercept=0.3333) +
  geom_segment(aes(xend = xvals, yend = linfit))
```




```
pengscat = ggplot(penguins, aes(x=body_mass_g, y=flipper_length_mm)) + geom_point()
pengscat
```

```
## Warning: Removed 2 rows containing missing values (geom_point).
```

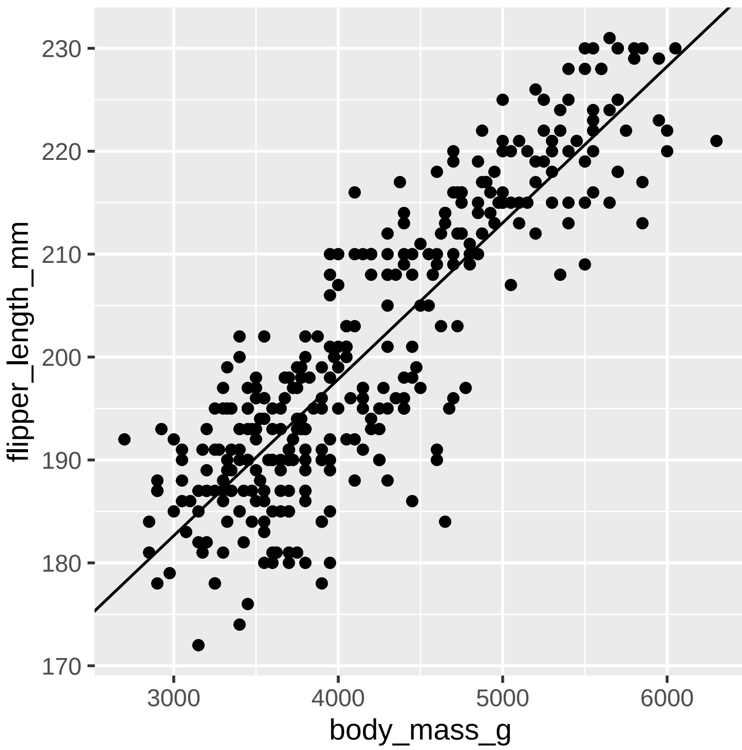



```
pengfit = lm(formula = flipper_length_mm ~ body_mass_g, data = peng)
pengfit
```

```
##
## Call:
## lm(formula = flipper_length_mm ~ body_mass_g, data = peng)
##
## Coefficients:
## (Intercept)  body_mass_g
##    137.0396       0.0152
```

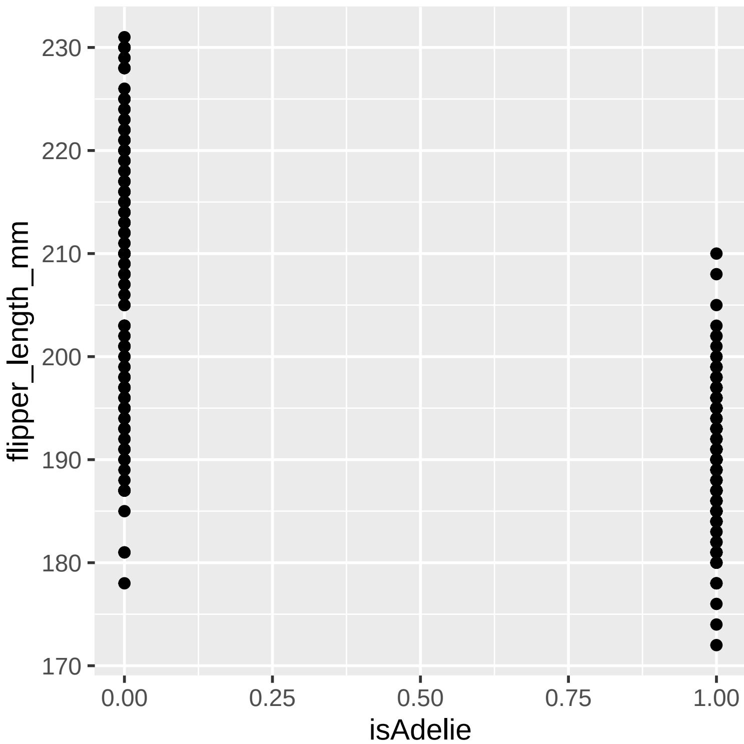
```
pengscat + geom_abline(slope= 0.0152, intercept = 137.0396)
```

```
## Warning: Removed 2 rows containing missing values (geom_point).
```



Categorical Data to Numerical Representations

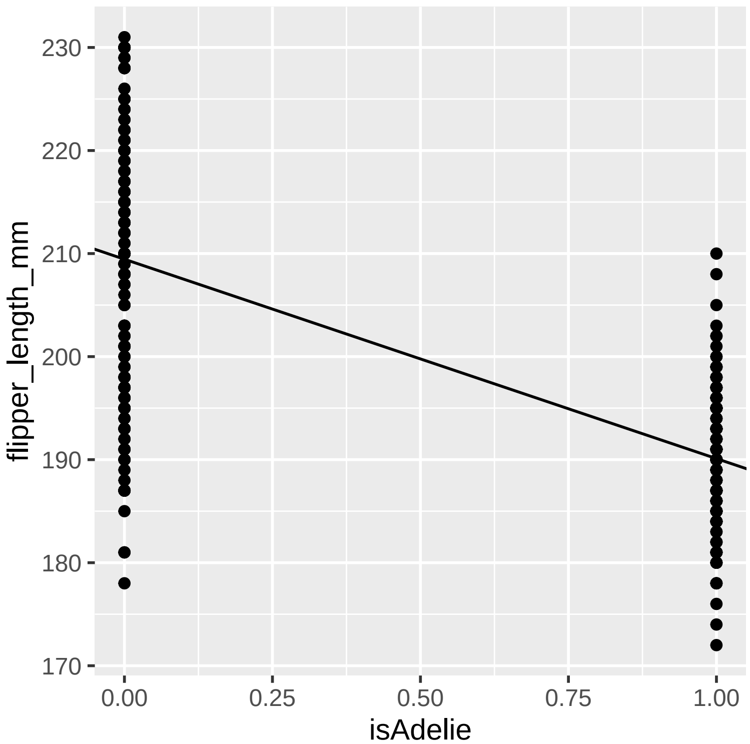
```
peng$isAdelie = ifelse(peng$species=='Adelie',1,0)
adelieplot = ggplot(peng, aes(x=isAdelie, y=flipper_length_mm)) + geom_point()
adelieplot
```



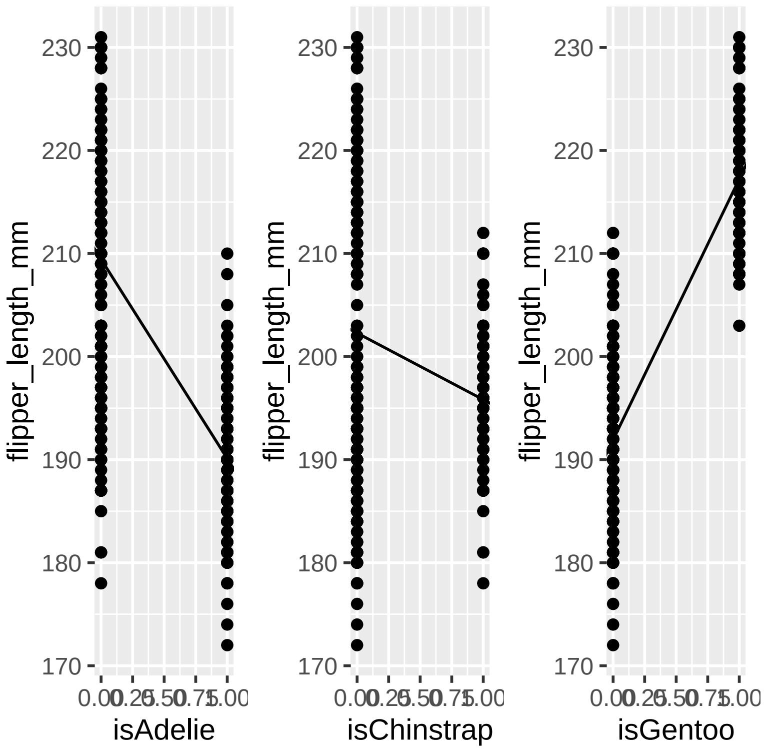

```
amodel = lm(formula = flipper_length_mm~isAdelie, data=peng)
amodel
```

```
##
## Call:
## lm(formula = flipper_length_mm ~ isAdelie, data = peng)
##
## Coefficients:
## (Intercept)     isAdelie
##      209.45       -19.35
```

```
adelieplot + geom_abline(slope=-19.35, intercept=209.45)
```




```
peng$isChinstrap = ifelse(peng$species=='Chinstrap',1,0)
peng$isGentoo = ifelse(peng$species=='Gentoo',1,0)

cmodel = lm(formula = flipper_length_mm~isChinstrap, data=peng)
gmodel = lm(formula = flipper_length_mm~isGentoo, data=peng)

speciesbase = ggplot(peng, aes(y=flipper_length_mm))
aplot= speciesbase + geom_point(aes(x=isAdelie)) + geom_abline(slope=amodel$coefficients[2], intercept=a
cplot = speciesbase + geom_point(aes(x=isChinstrap)) + geom_abline(slope=cmodel$coefficients[2], interc
gplot = speciesbase + geom_point(aes(x=isGentoo)) + geom_abline(slope=gmodel$coefficients[2], intercept=

grid.arrange( aplot, cplot, gplot, ncol=3)
```




```r
linmodel <- lm(flipper_length_mm ~ species, data = peng)
linmodel
```

```
##
## Call:
## lm(formula = flipper_length_mm ~ species, data = peng)
##
## Coefficients:
##      (Intercept)  speciesChinstrap     speciesGentoo
##          190.103             5.721            27.133
```

```r
peng_encoded = peng %>% mutate(value = 1) %>% spread(species, value, fill = 0 )
head(peng_encoded)
```

```
## # A tibble: 6 x 13
##   island    bill_length_mm bill_depth_mm flipper_length_~ body_mass_g sex    year
##   <fct>              <dbl>         <dbl>            <int>       <int> <fct> <int>
## 1 Torgers~            39.1          18.7              181        3750 male   2007
## 2 Torgers~            39.5          17.4              186        3800 fema~  2007
## 3 Torgers~            40.3          18                195        3250 fema~  2007
## 4 Torgers~            36.7          19.3              193        3450 fema~  2007
## 5 Torgers~            39.3          20.6              190        3650 male   2007
## 6 Torgers~            38.9          17.8              181        3625 fema~  2007
## # ... with 6 more variables: isAdelie <dbl>, isChinstrap <dbl>, isGentoo <dbl>,
## #   Adelie <dbl>, Chinstrap <dbl>, Gentoo <dbl>
```