PMRCT\_data\_overview

#### ***don***

#### ***December 2, 2017***

R Markdown

This is the R markdown document for all of our data overview procedures.

library('readr')  
library(stats)  
library('dplyr')

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library('tidyr')  
library('tibble')  
library('stringr')  
library('ggplot2')

The first step is to import the data for later analyze. We focus on four files: training\_variants, training\_text, stage2\_test\_variants.csv, and stage2\_test\_text.csv. The variants files are easy to import with read.csv() fuction. The text files are a little bit more complicated because in the data files the first line which indiccates the name of each column is separated into colunmns as it should be, where as the following lines are not, ie. the lines are in column. For this reason, we take use of the tibble library to first read in each line as a long character object and then manually separate the data and name the features.

train\_gene\_variants <- read.csv('../data/training\_variants')  
test\_gene\_variants <- read.csv('../data/stage2\_test\_variants.csv')  
train\_gene\_variants2 <- read.csv('../data/test\_variants')  
  
train\_text\_temp <- tibble( text=read\_lines('../data/training\_text', skip = 1 ))  
train\_text <- train\_text\_temp %>%  
 separate(text, into = c("ID", "text"), sep = "\\|\\|")  
train\_text <- train\_text %>%  
 mutate(ID = as.integer(ID))  
  
test\_text\_dump <- tibble(text = read\_lines('../data/stage2\_test\_text.csv', skip = 1))  
test\_text <- test\_text\_dump %>%  
 separate(text, into = c("ID", "text"), sep = "\\|\\|")  
test\_text <- test\_text %>%  
 mutate(ID = as.integer(ID))

Gene and Variation Data

Next step, let take a look at what our genes and variants data look like.

train\_gene\_variants <- train\_gene\_variants %>%  
 mutate(Gene = factor(Gene),  
 Variation = factor(Variation),  
 Class = factor(Class))  
  
test\_gene\_variants <- test\_gene\_variants %>%  
 mutate(Gene = factor(Gene),  
 Variation = factor(Variation))

summary(train\_gene\_variants, maxsum = 10)

## ID Gene Variation Class   
## Min. : 0 BRCA1 : 264 Truncating Mutations: 93 1:568   
## 1st Qu.: 830 TP53 : 163 Deletion : 74 2:452   
## Median :1660 EGFR : 141 Amplification : 71 3: 89   
## Mean :1660 PTEN : 126 Fusions : 34 4:686   
## 3rd Qu.:2490 BRCA2 : 125 Overexpression : 6 5:242   
## Max. :3320 KIT : 99 G12V : 4 6:275   
## BRAF : 93 E17K : 3 7:953   
## ALK : 69 Q61H : 3 8: 19   
## ERBB2 : 69 Q61L : 3 9: 37   
## (Other):2172 (Other) :3030

train\_2\_unfilterd <- read\_csv('./stage1\_solution\_filtered.csv')

## Parsed with column specification:  
## cols(  
## ID = col\_integer(),  
## class1 = col\_integer(),  
## class2 = col\_integer(),  
## class3 = col\_integer(),  
## class4 = col\_integer(),  
## class5 = col\_integer(),  
## class6 = col\_integer(),  
## class7 = col\_integer(),  
## class8 = col\_integer(),  
## class9 = col\_integer()  
## )

train\_gene\_variants2 <- full\_join( train\_gene\_variants2, train\_2\_unfilterd, by= 'ID')  
train\_gene\_variants2 <- train\_gene\_variants2 %>%  
 filter(!is.na(class1))

summary(train\_gene\_variants2)

## ID Gene Variation class1   
## Min. : 12 BRCA1 : 29 Truncating Mutations: 18 Min. :0.0000   
## 1st Qu.:1458 TP53 : 19 Deletion : 14 1st Qu.:0.0000   
## Median :2689 EGFR : 16 Amplification : 8 Median :0.0000   
## Mean :2785 BRCA2 : 14 Fusions : 3 Mean :0.2554   
## 3rd Qu.:4082 PTEN : 14 G44D : 2 3rd Qu.:1.0000   
## Max. :5650 BRAF : 11 A126T : 1 Max. :1.0000   
## (Other):265 (Other) :322   
## class2 class3 class4 class5   
## Min. :0.000 Min. :0.00000 Min. :0.0000 Min. :0.00000   
## 1st Qu.:0.000 1st Qu.:0.00000 1st Qu.:0.0000 1st Qu.:0.00000   
## Median :0.000 Median :0.00000 Median :0.0000 Median :0.00000   
## Mean :0.125 Mean :0.01902 Mean :0.1766 Mean :0.06793   
## 3rd Qu.:0.000 3rd Qu.:0.00000 3rd Qu.:0.0000 3rd Qu.:0.00000   
## Max. :1.000 Max. :1.00000 Max. :1.0000 Max. :1.00000   
##   
## class6 class7 class8 class9   
## Min. :0.00000 Min. :0.0000 Min. :0.000000 Min. :0.0000   
## 1st Qu.:0.00000 1st Qu.:0.0000 1st Qu.:0.000000 1st Qu.:0.0000   
## Median :0.00000 Median :0.0000 Median :0.000000 Median :0.0000   
## Mean :0.05978 Mean :0.2745 Mean :0.005435 Mean :0.0163   
## 3rd Qu.:0.00000 3rd Qu.:1.0000 3rd Qu.:0.000000 3rd Qu.:0.0000   
## Max. :1.00000 Max. :1.0000 Max. :1.000000 Max. :1.0000   
##

summary(test\_gene\_variants, maxsum = 10)

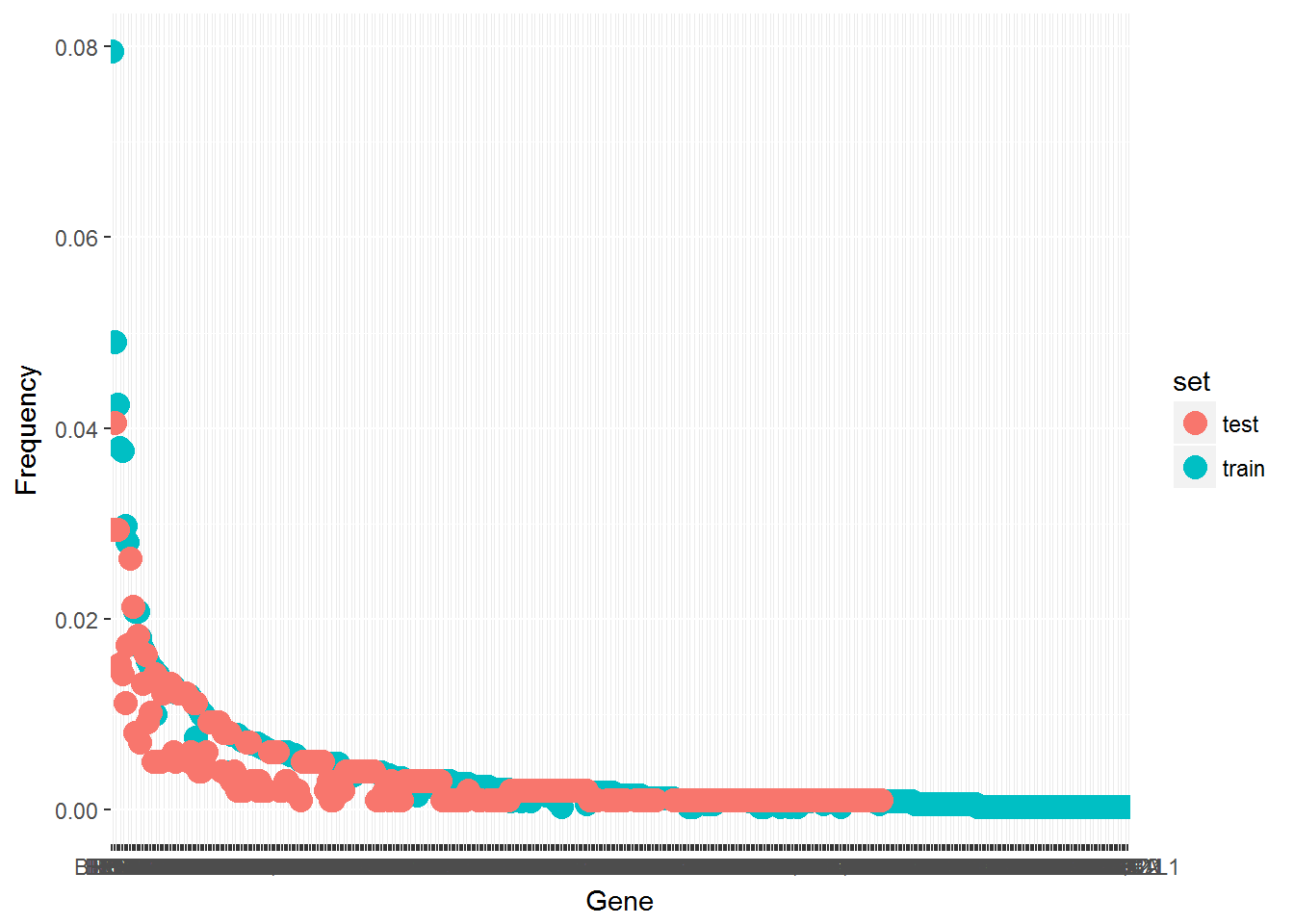
## ID Gene Variation   
## Min. : 1.0 TP53 : 40 Truncating Mutations: 18   
## 1st Qu.:247.2 BRCA1 : 29 Deletion : 14   
## Median :493.5 EGFR : 29 Amplification : 7   
## Mean :493.5 SCN4A : 26 Fusions : 3   
## 3rd Qu.:739.8 TSHR : 21 G13R : 2   
## Max. :986.0 ERBB2 : 18 G13S : 2   
## BRAF : 17 G44D : 2   
## TP63 : 16 A114V : 1   
## PTEN : 15 A1156T : 1   
## (Other):775 (Other) :936

From now, we would like to do little comparison between the train set and the test set, specifically the distribution of the genes in the data set.

#add a feature to train\_text  
count\_train\_gene <- train\_gene\_variants %>%   
 group\_by(Gene) %>%   
 summarise( count = n()/3321) %>%  
 filter( count > 0)  
count\_test\_gene <- test\_gene\_variants %>%   
 group\_by(Gene) %>%   
 summarise( count = n()/986) %>%  
 filter( count > 0)  
  
count\_train\_gene <- count\_train\_gene %>%  
 mutate(set = "train")  
count\_test\_gene <- count\_test\_gene %>%  
 mutate(set = "test")   
  
combine\_gene <- full\_join(count\_train\_gene, count\_test\_gene)

## Joining, by = c("Gene", "count", "set")

combine\_gene %>%  
 ggplot(aes(reorder(Gene, -count, FUN = min), count, fill = set)) +  
 geom\_point(size = 4, aes(colour=set)) +  
 labs(x = "Gene", y = "Frequency")



Draw some conclusion about the data here …

We would also want to investigate if there are combinations of gene and variation that are present in both training and testing set.

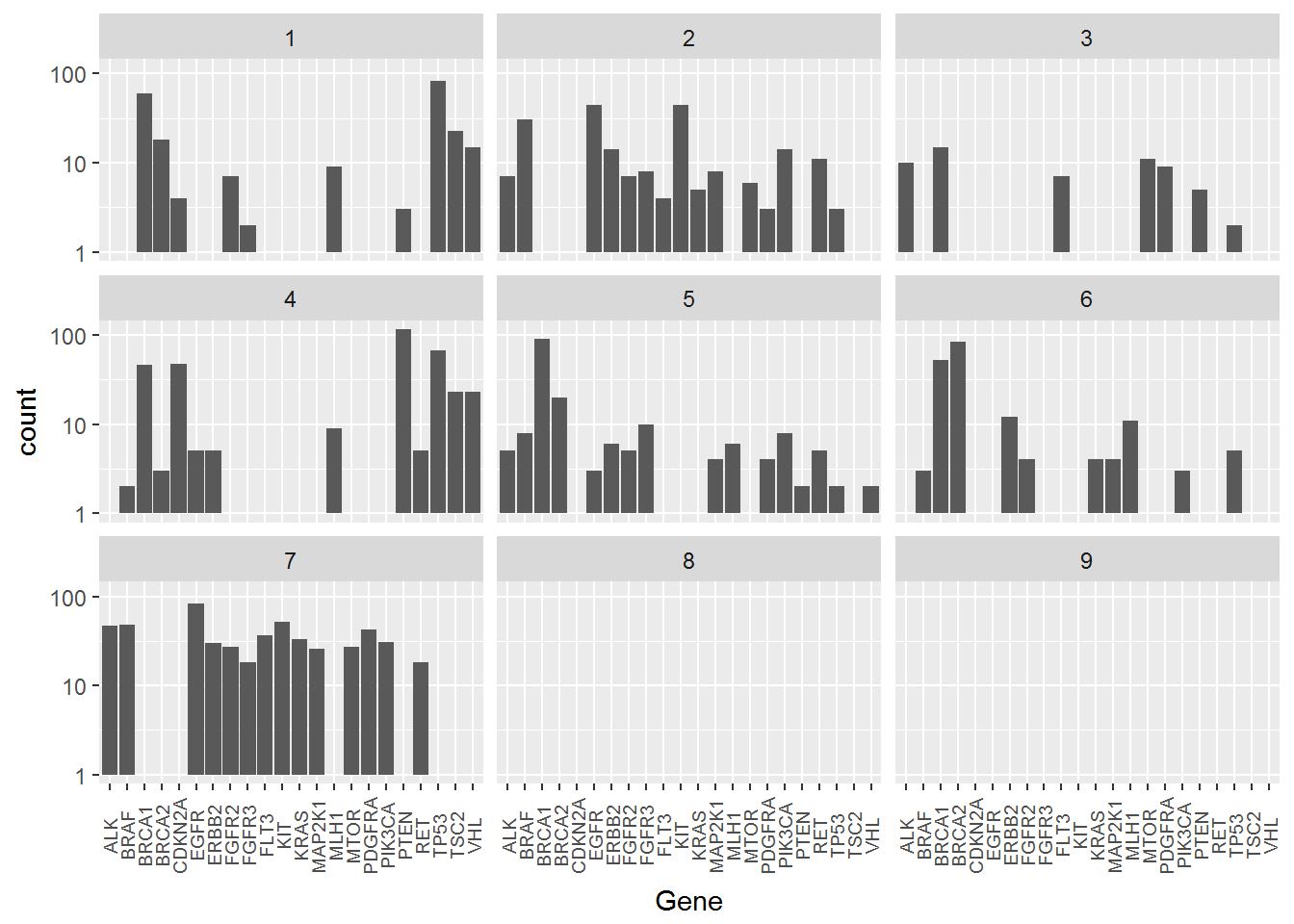
combination\_gene\_variation = merge(train\_gene\_variants, test\_gene\_variants, by=c('Gene', 'Variation'))  
summary(combination\_gene\_variation)

## Gene Variation ID.x Class ID.y   
## ABL1 :0 1\_2009trunc :0 Min. : NA 1 :0 Min. : NA   
## ACVR1 :0 2010\_2471trunc:0 1st Qu.: NA 2 :0 1st Qu.: NA   
## AGO2 :0 256\_286trunc :0 Median : NA 3 :0 Median : NA   
## AKT1 :0 3' Deletion :0 Mean :NaN 4 :0 Mean :NaN   
## AKT2 :0 385\_418del :0 3rd Qu.: NA 5 :0 3rd Qu.: NA   
## AKT3 :0 422\_605trunc :0 Max. : NA 6 :0 Max. : NA   
## (Other):0 (Other) :0 (Other):0

From the result (if you consider this a result), there is surprisingly 0 combination match between the train data and test data. In other words, none of the gene and variation combination that occurs in one data set can be found in the other.

After getting a general idea about what the gene and variants data look like. We want to investigate some basic and simple relationships between the features (gene and variation) and the label (class). We first look at the relationship between the gene and the class. We only look at genes that are frequent in the dataset, and we define >1% occurancy as frequent. We only look that the training data because the test data is not labeled.

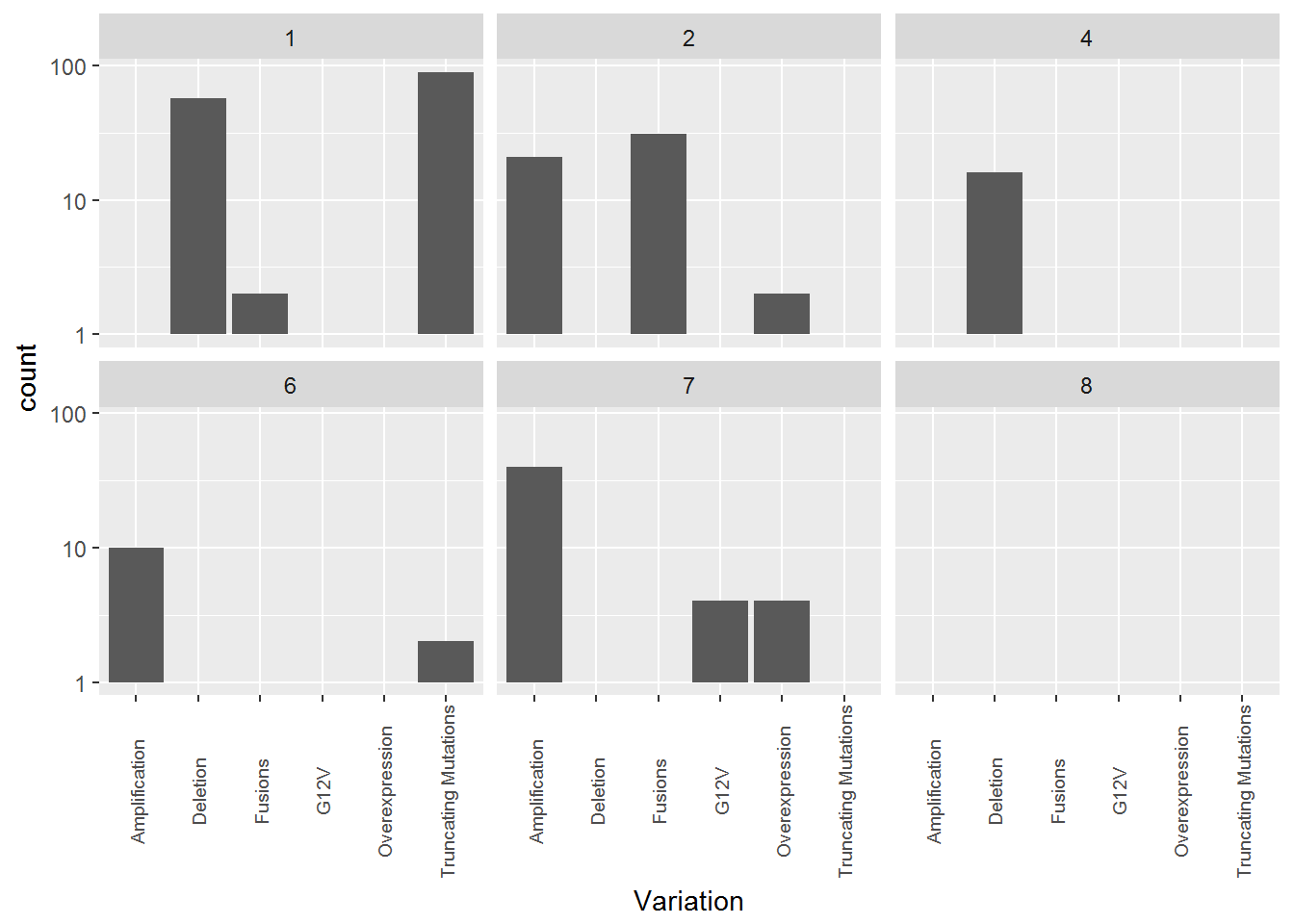
frequent\_train\_gene <- train\_gene\_variants %>%   
 group\_by(Gene) %>%   
 summarise( count = n()) %>%  
 filter( count > 33)  
  
train\_gene\_variants %>%  
 filter(Gene %in% str\_c(frequent\_train\_gene$Gene)) %>%  
 ggplot(aes(Gene)) +   
 geom\_bar() +   
 scale\_y\_log10() +   
 theme( axis.text.x = element\_text( angle=90, vjust=0.5, size=7)) +  
 facet\_wrap(~ Class)



Draw some conclusions here…

Next we look at the relationship between variants and class. Again We only look at variants that are frequent in the dataset with the same definition for frequent being >0.33%. And we only look that the training data because the test data is not labeled.

frequent\_train\_variants <- train\_gene\_variants %>%   
 group\_by(Variation) %>%   
 summarise( count = n()) %>%  
 filter( count > 3)  
  
train\_gene\_variants %>%  
 filter(Variation %in% str\_c(frequent\_train\_variants$Variation)) %>%  
 ggplot(aes(Variation)) +   
 geom\_bar() +   
 scale\_y\_log10() +   
 theme( axis.text.x = element\_text( angle=90, vjust=0.5, size=7)) +  
 facet\_wrap(~ Class)



Draw some conclusions here.

Text Data

First, let’s take a look at what ar text data look like.

glimpse(train\_text)

## Observations: 3,321  
## Variables: 2  
## $ ID <int> 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,...  
## $ text <chr> "Cyclin-dependent kinases (CDKs) regulate a variety of fu...

glimpse(test\_text)

## Observations: 986  
## Variables: 2  
## $ ID <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17...  
## $ text <chr> "The incidence of breast cancer is increasing in China in...

This doesn’t show us a lot about the data. We look at one specific entry. The str\_sub() function is used to control display length.

str\_sub(train\_text$text[1], start =1, end =1e3)

## [1] "Cyclin-dependent kinases (CDKs) regulate a variety of fundamental cellular processes. CDK10 stands out as one of the last orphan CDKs for which no activating cyclin has been identified and no kinase activity revealed. Previous work has shown that CDK10 silencing increases ETS2 (v-ets erythroblastosis virus E26 oncogene homolog 2)-driven activation of the MAPK pathway, which confers tamoxifen resistance to breast cancer cells. The precise mechanisms by which CDK10 modulates ETS2 activity, and more generally the functions of CDK10, remain elusive. Here we demonstrate that CDK10 is a cyclin-dependent kinase by identifying cyclin M as an activating cyclin. Cyclin M, an orphan cyclin, is the product of FAM58A, whose mutations cause STAR syndrome, a human developmental anomaly whose features include toe syndactyly, telecanthus, and anogenital and renal malformations. We show that STAR syndrome-associated cyclin M mutants are unable to interact with CDK10. Cyclin M silencing phenocopies CDK10"

As we can see here, each data entry of the text is a long piece of text of medical writing that one would classify a mutation type (our Class label) based on. Clearly, to turn this piece of text into usable information as a feature input to our machine learning models, we will need to perform some kind of Natural Language Processing on it. We can definitely extract meanings from the text by turning it into vectors, using one of many exisiting words-to-vec libraries. Such vectors can be used as feature inputs to our machine learning models. Additonally, since medical writings are often written with highly specific and precise terms and jargons, the occurence of certain terms in a text and in-occurence in others could also be a useful feature. Therefore, we believe TFIDF is also a viable approach to this problem.

Before we end our data overview, we can do some simple analysis with the text data, more specifically, the length of the texts.

#add a feature to train\_text  
train\_text <- train\_text %>%  
 mutate(text\_length = str\_length(text),   
 set = "train")  
test\_text <- test\_text %>%  
 mutate(text\_length = str\_length(text),   
 set = "test")  
  
combine\_text <- full\_join(train\_text, test\_text)

## Joining, by = c("ID", "text", "text\_length", "set")

combine\_text %>%  
 ggplot(aes(text\_length, fill = set)) +  
 geom\_histogram(bins = 50) +  
 labs(x = "Text Length")

