Moneyball: DATA621 Project #1

Team 2

03/07/2021

Contents

0.1 Introduction

- Data
- Purpose of Analysis
- Method?

0.2 Setup

This analysis requires installation of `tidyverse`,`corrplot`, and `reshape2`.

```
## -- Attaching packages ---------------------------------------------------------------------

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0

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

## Loading required package: reshape2
```

```
##
## Attaching package: 'reshape2'

## The following object is masked from 'package:tidyr':
##
##     smiths

## Loading required package: corrplot

## corrplot 0.84 loaded
```

0.2.1 Load Raw Data

The data has been divided in advance into a training set, `training_raw`, and a test set, `test_raw`.
We will load both data sets and perform exploratory data analysis on the training set, only.

```
url1 = "https://raw.githubusercontent.com/Jagdish16/CUNY_DATA_621/main/project_1/moneyball-trai
url2 = "https://raw.githubusercontent.com/Jagdish16/CUNY_DATA_621/main/project_1/moneyball-eval

training_raw = read_csv(url1)
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   INDEX = col_double(),
##   TARGET_WINS = col_double(),
##   TEAM_BATTING_H = col_double(),
##   TEAM_BATTING_2B = col_double(),
##   TEAM_BATTING_3B = col_double(),
##   TEAM_BATTING_HR = col_double(),
##   TEAM_BATTING_BB = col_double(),
##   TEAM_BATTING_SO = col_double(),
##   TEAM_BASERUN_SB = col_double(),
##   TEAM_BASERUN_CS = col_double(),
##   TEAM_BATTING_HBP = col_double(),
##   TEAM_PITCHING_H = col_double(),
##   TEAM_PITCHING_HR = col_double(),
##   TEAM_PITCHING_BB = col_double(),
##   TEAM_PITCHING_SO = col_double(),
##   TEAM_FIELDING_E = col_double(),
##   TEAM_FIELDING_DP = col_double()
## )
```

```
test_raw=read_csv(url2)
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   INDEX = col_double(),
##   TEAM_BATTING_H = col_double(),
##   TEAM_BATTING_2B = col_double(),
##   TEAM_BATTING_3B = col_double(),
##   TEAM_BATTING_HR = col_double(),
##   TEAM_BATTING_BB = col_double(),
##   TEAM_BATTING_SO = col_double(),
##   TEAM_BASERUN_SB = col_double(),
##   TEAM_BASERUN_CS = col_double(),
##   TEAM_BATTING_HBP = col_double(),
##   TEAM_PITCHING_H = col_double(),
##   TEAM_PITCHING_HR = col_double(),
##   TEAM_PITCHING_BB = col_double(),
##   TEAM_PITCHING_SO = col_double(),
##   TEAM_FIELDING_E = col_double(),
##   TEAM_FIELDING_DP = col_double()
## )
```

```
head(training_raw)
```

INDEX	TARGET_WINS	TEAM_BATTING_H	TEAM_BATTING_2B	TEAM_BATTING_3B	TEAM_BATTING_HR	TEAM_BATTING_BB	TEAM_BATTING_SO	TEAM_BASERUN_SB	TEAM_BASERUN_CS	TEAM_BATTING_HBP	TEAM_PITCHING_H	TEAM_PITCHING_HR	TEAM_PITCHING_BB	TEAM_PITCHING_SO	TEAM_FIELDING_E	TEAM_FIELDING_DP
1	39	1445	194	39	13	143	842	NA	NA	NA	9364	84	927	5456	1011	NA
2	70	1339	219	22	190	685	1075	37	28	NA	1347	191	689	1082	193	155
3	86	1377	232	35	137	602	917	46	27	NA	1377	137	602	917	175	153
4	70	1387	209	38	96	451	922	43	30	NA	1396	97	454	928	164	156
5	82	1297	186	27	102	472	920	49	39	NA	1297	102	472	920	138	168
6	75	1279	200	36	92	443	973	107	59	NA	1279	92	443	973	123	149

First, rename the columns to be more human-readable and drop INDEX, which has no meaning.

```
training <- training_raw %>%
  select(-INDEX) %>%
  rename_with(~ gsub("TEAM_", "", .x)) %>%
  rename_with(stringr::str_to_title) %>%
  dplyr::rename(
    Wins = Target_wins,
    Hits = Batting_h,
    Doubles = Batting_2b,
    Triples = Batting_3b,
    HomeRuns = Batting_hr,
    Walks_AtBat = Batting_bb,
    StrikeOuts_AtBat = Batting_so,
```

```
    BasesStolen = Baserun_sb,
    OutStealingBases = Baserun_cs,
    Hits_Allowed = Pitching_h,
    HitByPitch_AtBat = Batting_hbp,
    Errors = Fielding_e,
    HomeRuns_Allowed = Pitching_hr,
    Walks_Allowed = Pitching_bb,
    StrikeOuts = Pitching_so,
    DoublePlays = Fielding_dp
  )

colnames(training)
```

```
##  [1] "Wins"             "Hits"             "Doubles"          "Triples"
##  [5] "HomeRuns"         "Walks_AtBat"      "StrikeOuts_AtBat" "BasesStolen"
##  [9] "OutStealingBases" "HitByPitch_AtBat" "Hits_Allowed"     "HomeRuns_Allowed"
## [13] "Walks_Allowed"    "StrikeOuts"       "Errors"           "DoublePlays"
```

Second, we will add a variable called singles.

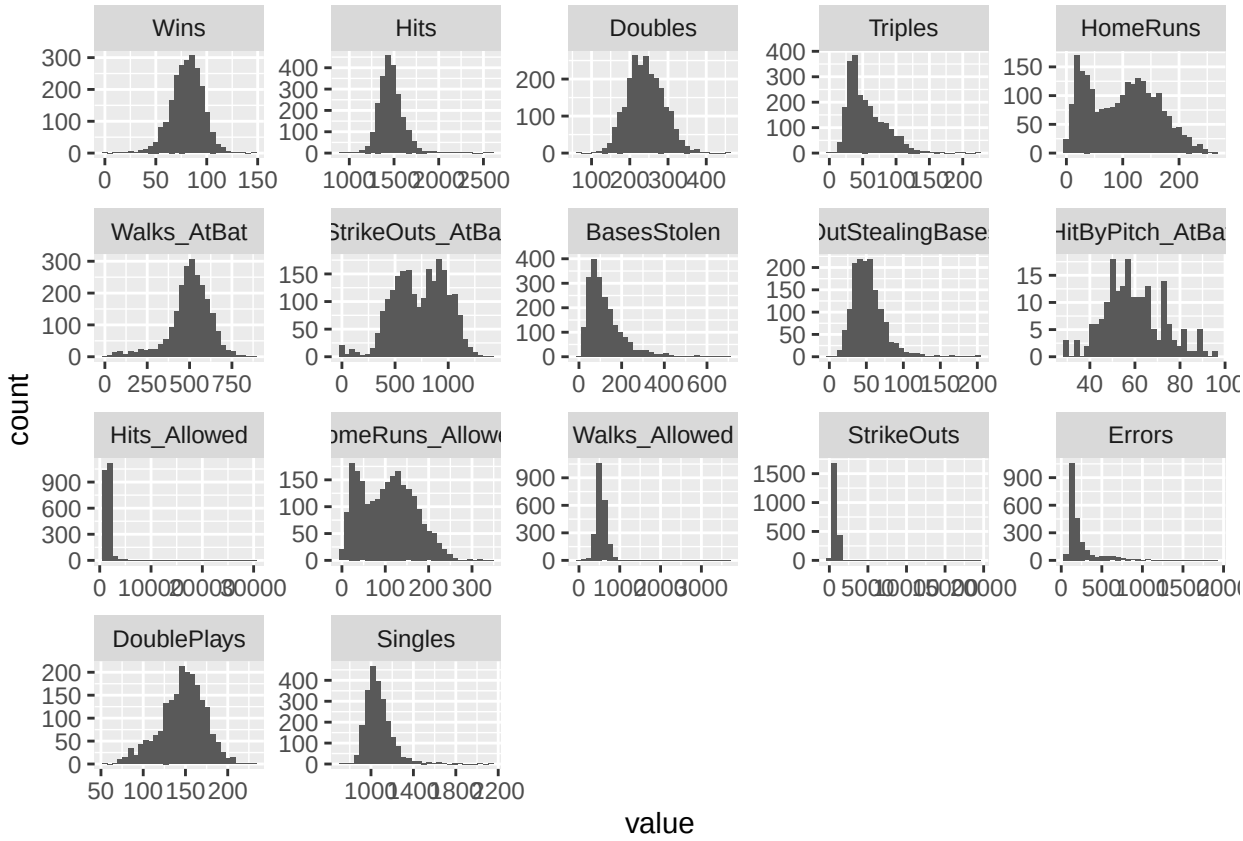
```
training <- training %>% mutate(Singles = Hits - HomeRuns - Doubles - Triples)
```

```
long <- training %>% as.data.frame() %>% melt
```

```
## No id variables; using all as measure variables
```

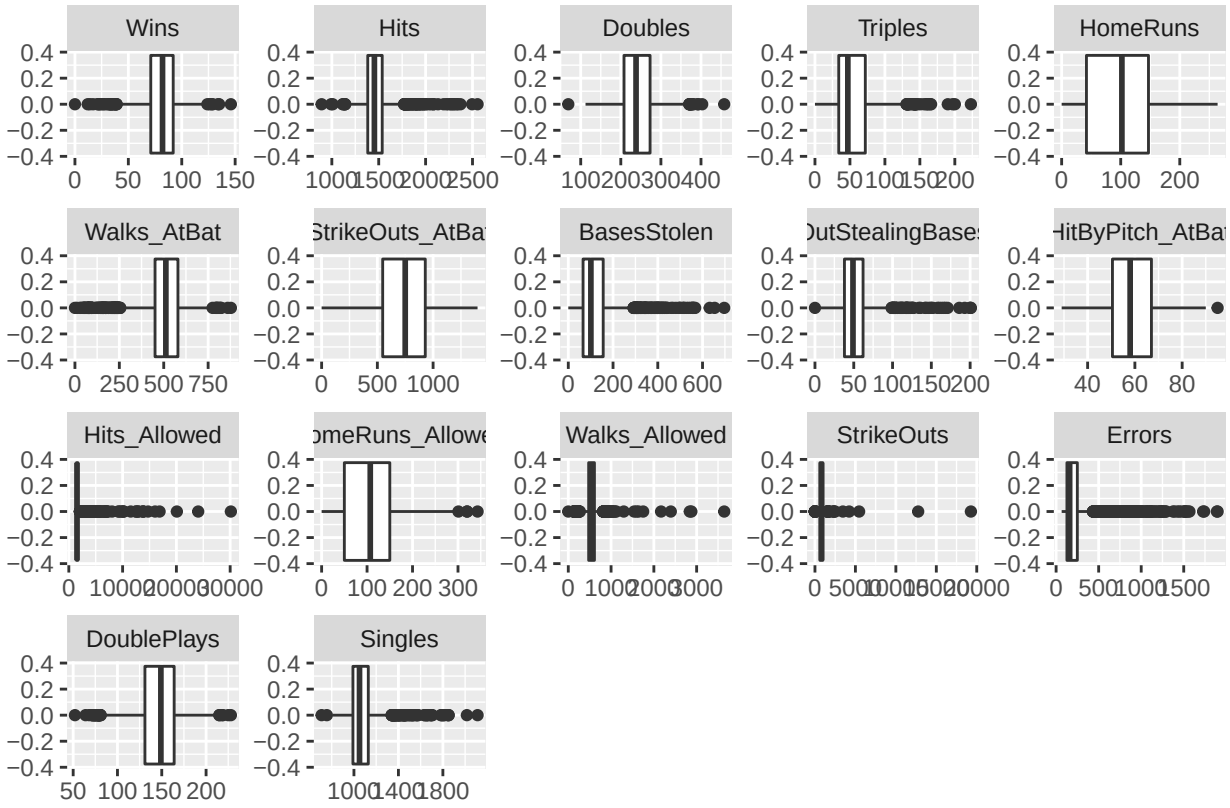
```
long %>%
  ggplot(aes(x=value)) +
  geom_histogram() +
  facet_wrap(~variable, scales='free')
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

## Warning: Removed 3478 rows containing non-finite values (stat_bin).
```




```
long %>%
  ggplot(aes(x=value)) +
  geom_boxplot() +
  facet_wrap(~variable, scales='free')
```

```
## Warning: Removed 3478 rows containing non-finite values (stat_boxplot).
```

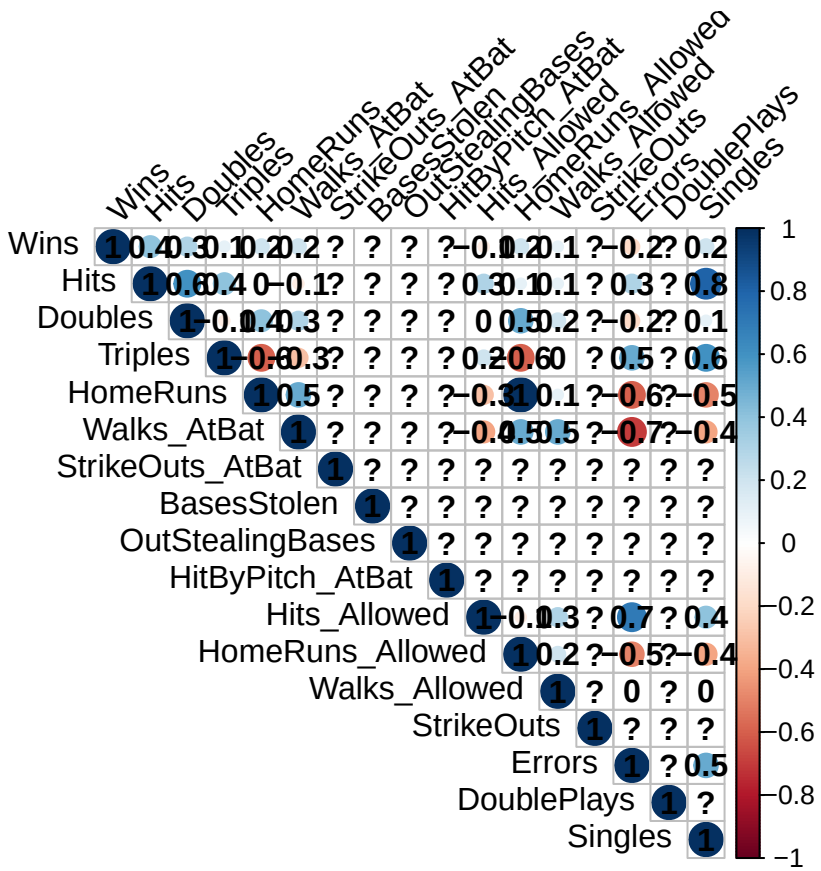



```
# removing index column
# how to handle NA values? TEAM_BATTING_HBP has 2085 NA values and TEAM_BASERUN_CS has 772
# removing them for now
# listwise deletion of missing values

#moneyball_training <- na.omit(moneyball_training[-c(1, 10, 11)])

#
# # removing outliers <- I think this may be too agressive (Alice)
# out_ind <- c()
# outliers <- c()
#
# for (i in 1:ncol(training)){
#
#   out <- boxplot.stats(training[, i])$out
#   outliers <- append(outliers, out)
#   out_ind <- append(out_ind, which(training[, i] %in% c(out)))
#
# }
#
# # to see the outliers
# # cbind(outliers, moneyball_training[out_ind,])
#
```

```
# # removing the outliers
# # moneyball_training <- moneyball_training[-out_ind, ]
#
# # changing the column names to be more human readable
#
#
#
#
# str(moneyball_training)
```

0.3 Including Plots

```
# computing correlation matrix
corr <- round(cor(training), digits = 1)

corrplot(corr, type = 'upper', addCoef.col = 'black', tl.col = 'black', tl.srt = 45)
```



Feature Selection We should drop highly correlated values, but which ones to drop?

Using our understanding of baseball, we can also tell which features are highly correlated because they are in fact calculated from a common statistic (e.g. Runs Hit, and Double Hit both include how many doubles were hit!) versus variables which are correlated in fact but not in function

(e.g. Runs Allowed Pitcher vs Runs Allowed Field – this is a measure of the picthing staff versus the batting so while these tend to be similar they are in fact measuring the skill of different players at different points in the game.)

Oddly, although pitching hits and pitching homeruns are likewise connecting in the real world, they are negatively correlated at just the 0.1 value in the data.

Because Team Batting Hits includes the data from singles, double, triples and home runs, it might sense to drop Team Batting Hits and add a new variable – Team Batting Singles

```
#re-factoring

FancyHist <- function(field, title){
  ggplot(data = moneyball_training, aes(x = get(field)))  +
    geom_histogram( color = 'black', fill =  'gray') +
    geom_vline(aes(xintercept = mean(get(field))),
               linetype = 'dashed', size = 2, color = 'blue') +
    #geom_label(aes(x = 50, y = 125,
    #               label = str_replace_all(toString(summary(moneyball_training['WINS'])), ','
    #               )) +
    labs(title = title, y = 'Count', x=field)
}

FancyHist('WINS', "Wins")

#Hits
ggplot(data = moneyball_training, aes(x = B_H))  +
  geom_histogram( color = 'black', fill =  'gray') +
  geom_vline(aes(xintercept = mean(B_H)),
             linetype = 'dashed', size = 2, color = 'blue') +
  geom_label(aes(x = 1250, y = 100,
                 label = str_replace_all(toString(summary(moneyball_training['B_H'])), ',', '\n
                 )) +
  labs(title = 'Base Hits Histogram Plot', y = 'Count', x = 'Base Hits')

# Doubles
ggplot(data = moneyball_training, aes(x = B_2B))  +
  geom_histogram( color = 'black', fill =  'gray') +
  geom_vline(aes(xintercept = mean(B_2B)),
             linetype = 'dashed', size = 2, color = 'blue') +
  geom_label(aes(x = 160, y = 100,
                 label = str_replace_all(toString(summary(moneyball_training['B_2B'])), ',', '
                 )) +
  labs(title = 'Doubles Histogram Plot', y = 'Count', x = 'Doubles')


# Triples
ggplot(data = moneyball_training, aes(x = B_3B))  +
  geom_histogram( color = 'black', fill =  'gray') +
```

```
  geom_vline(aes(xintercept = mean(B_3B)),
             linetype = 'dashed', size = 2, color = 'blue') +
  geom_label(aes(x = 75, y = 130,
                 label = str_replace_all(toString(summary(moneyball_training['B_3B'])), ',', '
                 )) +
  labs(title = 'Triples Histogram Plot', y = 'Count', x = 'Triples')


# Homeruns
ggplot(data = moneyball_training, aes(x = B_HR))  +
  geom_histogram( color = 'black', fill =  'gray') +
  geom_vline(aes(xintercept = mean(B_HR)),
             linetype = 'dashed', size = 2, color = 'blue') +
  geom_label(aes(x = 25, y = 90,
                 label = str_replace_all(toString(summary(moneyball_training['B_HR'])), ',', '
                 )) +
  labs(title = 'Homeruns Histogram Plot', y = 'Count', x = 'Homeruns')


# Walks
ggplot(data = moneyball_training, aes(x = B_BB))  +
  geom_histogram( color = 'black', fill =  'gray') +
  geom_vline(aes(xintercept = mean(B_BB)),
             linetype = 'dashed', size = 2, color = 'blue') +
  geom_label(aes(x = 380, y = 100,
                 label = str_replace_all(toString(summary(moneyball_training['B_BB'])), ',', '
                 )) +
  labs(title = 'Walks Histogram Plot', y = 'Count', x = 'Walks')


# Strike Out by Batters
ggplot(data = moneyball_training, aes(x = B_SO))  +
  geom_histogram( color = 'black', fill =  'gray') +
  geom_vline(aes(xintercept = mean(B_SO)),
             linetype = 'dashed', size = 2, color = 'blue') +
  geom_label(aes(x = 380, y = 100,
                 label = str_replace_all(toString(summary(moneyball_training['B_SO'])), ',', '
                 )) +
  labs(title = 'Strike Out by Batters Histogram Plot', y = 'Count', x = 'Strike Out by Batters


# Stolen Bases
ggplot(data = moneyball_training, aes(x = BR_SB))  +
  geom_histogram( color = 'black', fill =  'gray') +
  geom_vline(aes(xintercept = mean(BR_SB)),
             linetype = 'dashed', size = 2, color = 'blue') +
  geom_label(aes(x = 200, y = 100,
```

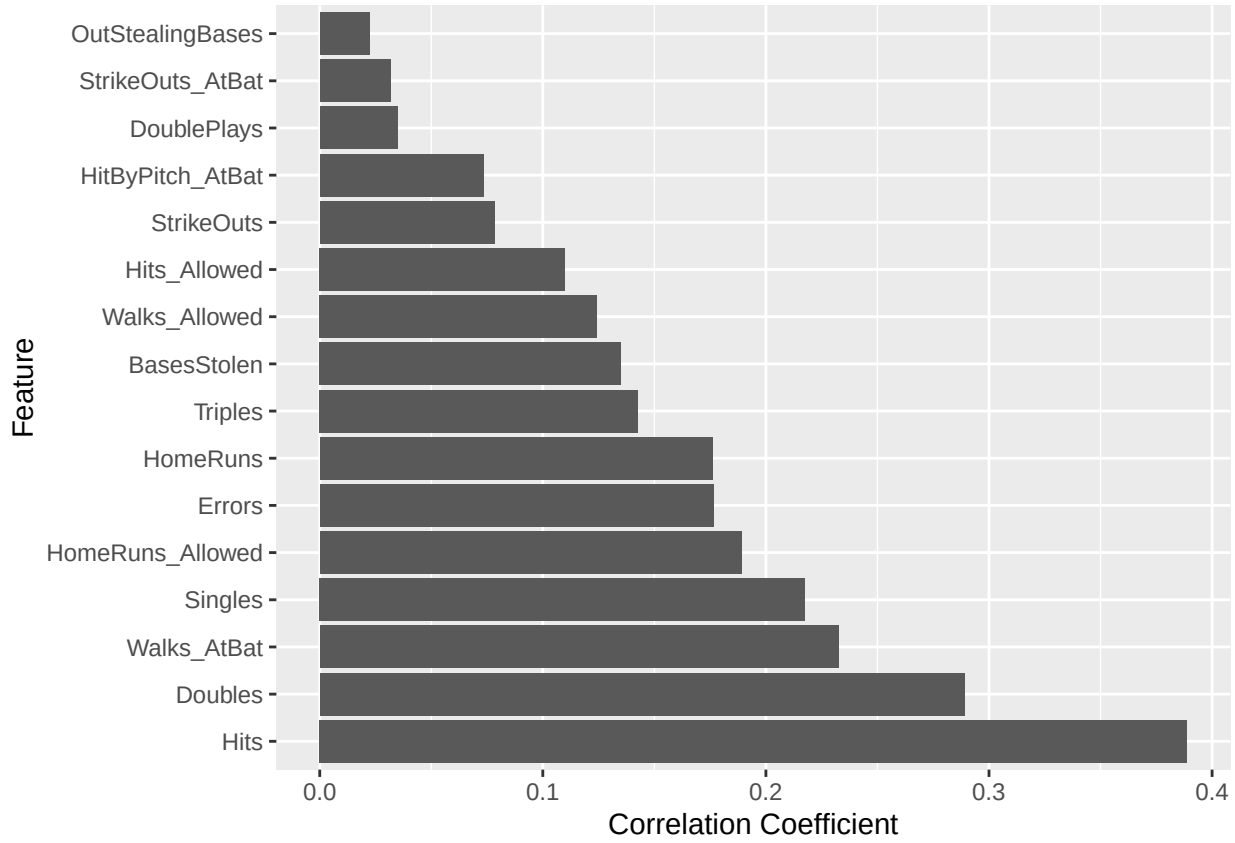
```
                label = str_replace_all(toString(summary(moneyball_training['BR_SB'])), ',',
                )) +
  labs(title = 'Stolen Bases Histogram Plot', y = 'Count', x = 'Stolen Bases')


# Hits Allowed
ggplot(data = moneyball_training, aes(x = P_H))  +
  geom_histogram( color = 'black', fill =  'gray') +
  geom_vline(aes(xintercept = mean(P_H)),
             linetype = 'dashed', size = 2, color = 'blue') +
  geom_label(aes(x = 1250, y = 100,
                label = str_replace_all(toString(summary(moneyball_training['P_H'])), ',', '\n
                )) +
  labs(title = 'Hits Allowed Histogram Plot', y = 'Count', x = 'Hits Allowed')


# Homeruns Allowed
ggplot(data = moneyball_training, aes(x = P_HR))  +
  geom_histogram( color = 'black', fill =  'gray') +
  geom_vline(aes(xintercept = mean(P_HR)),
             linetype = 'dashed', size = 2, color = 'blue') +
  geom_label(aes(x = 30, y = 90,
                label = str_replace_all(toString(summary(moneyball_training['P_HR'])), ',', '
                )) +
  labs(title = 'Homeruns Allowed Histogram Plot', y = 'Count', x = 'Homeruns Allowed')


# Walks Allowed
ggplot(data = moneyball_training, aes(x = P_BB))  +
  geom_histogram( color = 'black', fill =  'gray') +
  geom_vline(aes(xintercept = mean(P_BB)),
             linetype = 'dashed', size = 2, color = 'blue') +
  geom_label(aes(x = 375, y = 100,
                label = str_replace_all(toString(summary(moneyball_training['P_BB'])), ',', '
                )) +
  labs(title = 'Walks Allowed Histogram Plot', y = 'Count', x = 'Walks Allowed')


# Strikeouts by Pitchers
ggplot(data = moneyball_training, aes(x = P_SO))  +
  geom_histogram( color = 'black', fill =  'gray') +
  geom_vline(aes(xintercept = mean(P_SO)),
             linetype = 'dashed', size = 2, color = 'blue') +
  geom_label(aes(x = 425, y = 100,
                label = str_replace_all(toString(summary(moneyball_training['P_SO'])), ',', '
                )) +
  labs(title = 'Strikeouts by Pitchers Histogram Plot', y = 'Count', x = 'Strikeouts by Pitche
```

```
# Errors
ggplot(data = moneyball_training, aes(x = F_E))  +
  geom_histogram( color = 'black', fill =  'gray') +
  geom_vline(aes(xintercept = mean(F_E)),
             linetype = 'dashed', size = 2, color = 'blue') +
  geom_label(aes(x = 225, y = 100,
                 label = str_replace_all(toString(summary(moneyball_training['F_E'])), ',', '\n
                 )) +
  labs(title = 'Errors Histogram Plot', y = 'Count', x = 'Errors')

# Double Plays
ggplot(data = moneyball_training, aes(x = F_DP))  +
  geom_histogram( color = 'black', fill =  'gray') +
  geom_vline(aes(xintercept = mean(F_DP)),
             linetype = 'dashed', size = 2, color = 'blue') +
  geom_label(aes(x = 110, y = 100,
                 label = str_replace_all(toString(summary(moneyball_training['F_DP'])), ',', '
                 )) +
  labs(title = 'Double Plays Histogram Plot', y = 'Count', x = 'Double Plays')
```

To see what has the biggest impact on the results, we can evaluate the absolute value of the correlation of every feature.

```
# adapted from https://www.kaggle.com/reisel/how-to-handle-correlated-features

#start with field names
features <- training %>% select(-Wins) %>% names()

#initiate blank data frame
corr_coefs<- data.frame(feature = features, coef = rep(NA, length(features)))

#for every feature, calculate the correlation with the target, "Wins"
i <- 1
for (feature in features){
  df <- training[, c(feature, "Wins")] %>% na.omit() #omit NAs for each feature
  corr_coefs$coef[i] <- abs(cor(df[, feature], df$Wins))
  # print(feature)
  # print(i)
  # print(corr_coefs$coef[i])
  i <- i + 1
}

# sort by correlation coefficient
corr_coefs<- corr_coefs[order(corr_coefs$coef, decreasing = TRUE), ]

ggplot(corr_coefs, aes(x = factor(feature, levels = feature), y = coef)) +
```

```
    geom_bar(stat = "identity") +
    coord_flip() +
    xlab("Feature") +
    ylab("Correlation Coefficient")
```



0.4 Model Testing

Then we can test some models

```
lm_original <- lm(Wins ~ . -Singles, data = training)
summary(lm_original)
```

```
##
## Call:
## lm(formula = Wins ~ . - Singles, data = training)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -19.8708  -5.6564  -0.0599   5.2545  22.9274
##
## Coefficients:
```

```
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)      60.28826   19.67842   3.064  0.00253 **
## Hits              1.91348    2.76139   0.693  0.48927
## Doubles           0.02639    0.03029   0.871  0.38484
## Triples          -0.10118    0.07751  -1.305  0.19348
## HomeRuns         -4.84371   10.50851  -0.461  0.64542
## Walks_AtBat      -4.45969    3.63624  -1.226  0.22167
## StrikeOuts_AtBat  0.34196    2.59876   0.132  0.89546
## BasesStolen       0.03304    0.02867   1.152  0.25071
## OutStealingBases -0.01104    0.07143  -0.155  0.87730
## HitByPitch_AtBat  0.08247    0.04960   1.663  0.09815 .
## Hits_Allowed     -1.89096    2.76095  -0.685  0.49432
## HomeRuns_Allowed  4.93043   10.50664   0.469  0.63946
## Walks_Allowed     4.51089    3.63372   1.241  0.21612
## StrikeOuts       -0.37364    2.59705  -0.144  0.88577
## Errors           -0.17204    0.04140  -4.155 5.08e-05 ***
## DoublePlays      -0.10819    0.03654  -2.961  0.00349 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.467 on 175 degrees of freedom
##   (2085 observations deleted due to missingness)
## Multiple R-squared:  0.5501, Adjusted R-squared:  0.5116
## F-statistic: 14.27 on 15 and 175 DF,  p-value: < 2.2e-16
```

```
lm_all <- lm(Wins ~ . - Hits, data = training)
summary(lm_all)
```

```
##
## Call:
## lm(formula = Wins ~ . - Hits, data = training)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -19.8708  -5.6564  -0.0599   5.2545  22.9274
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)      60.28826   19.67842   3.064  0.00253 **
## Doubles           1.93986    2.75972   0.703  0.48304
## Triples           1.81230    2.76588   0.655  0.51318
## HomeRuns         -2.93023    9.31276  -0.315  0.75340
## Walks_AtBat      -4.45969    3.63624  -1.226  0.22167
## StrikeOuts_AtBat  0.34196    2.59876   0.132  0.89546
## BasesStolen       0.03304    0.02867   1.152  0.25071
## OutStealingBases -0.01104    0.07143  -0.155  0.87730
## HitByPitch_AtBat  0.08247    0.04960   1.663  0.09815 .
```

```
## Hits_Allowed     -1.89096    2.76095  -0.685  0.49432
## HomeRuns_Allowed  4.93043   10.50664   0.469  0.63946
## Walks_Allowed     4.51089    3.63372   1.241  0.21612
## StrikeOuts       -0.37364    2.59705  -0.144  0.88577
## Errors           -0.17204    0.04140  -4.155 5.08e-05 ***
## DoublePlays      -0.10819    0.03654  -2.961  0.00349 **
## Singles           1.91348    2.76139   0.693  0.48927
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.467 on 175 degrees of freedom
##   (2085 observations deleted due to missingness)
## Multiple R-squared:  0.5501, Adjusted R-squared:  0.5116
## F-statistic: 14.27 on 15 and 175 DF,  p-value: < 2.2e-16
```

These produce pretty identica results. Maybe this is why Singles aren't tracked!

Let's drop the least correlated feature.

```
test <- function(droplist, data=training){
  df <- data %>% select(-all_of(droplist))
  lm <- lm(Wins ~ . , df)
  #
  # print("Features Dropped")
  # print(droplist)

  print("R Squared")
  print(summary(lm)$adj.r.squared)

    # print("p value")
  # f <- summary(lm)$fstatistic
  # p <- pf(f[1],f[2],f[3],lower.tail=F)
  # attributes(p) <- NULL
  # print(p)
  return(summary(lm)$adj.r.squared)
}

droplist <- c()
Rs <- c()
for (feature in corr_coefs$feature) {
  droplist <- append(droplist, feature)
  Rs <- append(Rs, test(droplist))
}
```

```
## [1] "R Squared"
## [1] 0.511555
## [1] "R Squared"
## [1] 0.512959
```

```
## [1] "R Squared"
## [1] 0.5129123
## [1] "R Squared"
## [1] 0.5137191
## [1] "R Squared"
## [1] 0.5159577
## [1] "R Squared"
## [1] 0.4688989
## [1] "R Squared"
## [1] 0.4484293
## [1] "R Squared"
## [1] 0.4411111
## [1] "R Squared"
## [1] 0.441697
## [1] "R Squared"
## [1] 0.2794082
## [1] "R Squared"
## [1] 0.1312266
## [1] "R Squared"
## [1] 0.1343159
## [1] "R Squared"
## [1] 0.007486925
## [1] "R Squared"
## [1] -0.0006944347
## [1] "R Squared"
## [1] -0.0001635025
## [1] "R Squared"
## [1] 0
```

```
Rs
```

```
##  [1]  0.5115550332  0.5129590441  0.5129123060  0.5137190669  0.5159576721
##  [6]  0.4688988863  0.4484292674  0.4411110815  0.4416970484  0.2794081952
## [11]  0.1312266089  0.1343158517  0.0074869252 -0.0006944347 -0.0001635025
## [16]  0.0000000000
```

The 5th round is slightly better than the 1st 4, but not by much.

Can we try removing the features with the lowest p-values?

```
features2 <- summary(lm_all)$coefficients %>% data.frame %>% arrange(desc(Pr...t..))
features2
```

	Estimate	Std..Error	t.value	Pr...t..
StrikeOuts_AtBat	0.3419626	2.5987586	0.1315869	0.8954622
StrikeOuts	-0.3736449	2.5970524	-0.1438727	0.8857665
OutStealingBases	-0.0110443	0.0714310	-0.1546145	0.8773034

	Estimate	Std..Error	t.value	Pr...t..
HomeRuns	-2.9302310	9.3127561	-0.3146470	0.7534045
HomeRuns_Allowed	4.9304318	10.5066449	0.4692680	0.6394622
Triples	1.8123007	2.7658780	0.6552352	0.5131767
Hits_Allowed	-1.8909569	2.7609462	-0.6848945	0.4943166
Singles	1.9134762	2.7613936	0.6929386	0.4892666
Doubles	1.9398643	2.7597222	0.7029201	0.4830396
BasesStolen	0.0330440	0.0286728	1.1524493	0.2507083
Walks_AtBat	-4.4596914	3.6362413	-1.2264564	0.2216746
Walks_Allowed	4.5108907	3.6337201	1.2413974	0.2161203
HitByPitch_AtBat	0.0824727	0.0496001	1.6627518	0.0981521
DoublePlays	-0.1081921	0.0365408	-2.9608557	0.0034937
(Intercept)	60.2882626	19.6784170	3.0636744	0.0025324
Errors	-0.1720420	0.0414040	-4.1552001	0.0000508

```
test(c("StrikeOuts_AtBat", "StrikeOuts"))
```

```
## [1] "R Squared"
## [1] 0.4664896

## [1] 0.4664896
```

0.5 Removing Outliers

```
# removing outliers
out_ind <- c()
outliers <- c()

for (col in names(training)){

  out <- boxplot.stats(training[, col])$out
  outliers <- append(outliers, out)
  out_ind <- append(out_ind, which(training[, col] %in% c(out)))

}

# to see the outliers
# cbind(outliers, moneyball_training[out_ind,])

# removing the outliers
training_mod <- raining[-out_ind, ]
```

```r
lm2 <- lm(Wins ~ . -HomeRuns, data = training)
summary(lm2)
```

```
##
## Call:
## lm(formula = Wins ~ . - HomeRuns, data = training)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -19.8708  -5.6564  -0.0599   5.2545  22.9274
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)      60.28826   19.67842   3.064  0.00253 **
## Hits             -2.93023    9.31276  -0.315  0.75340
## Doubles           4.87010   10.50720   0.464  0.64358
## Triples           4.74253   10.51456   0.451  0.65252
## Walks_AtBat      -4.45969    3.63624  -1.226  0.22167
## StrikeOuts_AtBat  0.34196    2.59876   0.132  0.89546
## BasesStolen       0.03304    0.02867   1.152  0.25071
## OutStealingBases -0.01104    0.07143  -0.155  0.87730
## HitByPitch_AtBat  0.08247    0.04960   1.663  0.09815 .
## Hits_Allowed     -1.89096    2.76095  -0.685  0.49432
## HomeRuns_Allowed  4.93043   10.50664   0.469  0.63946
## Walks_Allowed     4.51089    3.63372   1.241  0.21612
## StrikeOuts       -0.37364    2.59705  -0.144  0.88577
## Errors           -0.17204    0.04140  -4.155 5.08e-05 ***
## DoublePlays      -0.10819    0.03654  -2.961  0.00349 **
## Singles           4.84371   10.50851   0.461  0.64542
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.467 on 175 degrees of freedom
##   (2085 observations deleted due to missingness)
## Multiple R-squared:  0.5501, Adjusted R-squared:  0.5116
## F-statistic: 14.27 on 15 and 175 DF,  p-value: < 2.2e-16
```

```r
lm3 <- lm(Wins ~ . -HomeRuns -StrikeOuts_AtBat, data = training)
summary(lm3)
```

```
##
## Call:
## lm(formula = Wins ~ . - HomeRuns - StrikeOuts_AtBat, data = training)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
```

```
## -19.8652  -5.6911  -0.0713   5.2828  22.9235
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      60.488230  19.564802   3.092  0.00231 **
## Hits             -2.754544   9.190786  -0.300  0.76475
## Doubles           4.953224  10.458873   0.474  0.63638
## Triples           4.825953  10.466084   0.461  0.64529
## Walks_AtBat      -4.461994   3.626034  -1.231  0.22014
## BasesStolen       0.032792   0.028529   1.149  0.25193
## OutStealingBases -0.010623   0.071160  -0.149  0.88150
## HitByPitch_AtBat  0.082131   0.049394   1.663  0.09814 .
## Hits_Allowed     -2.150028   1.930192  -1.114  0.26684
## HomeRuns_Allowed  5.014416  10.457923   0.479  0.63219
## Walks_Allowed     4.513224   3.623518   1.246  0.21459
## StrikeOuts       -0.031908   0.007435  -4.291 2.93e-05 ***
## Errors           -0.172188   0.041273  -4.172 4.74e-05 ***
## DoublePlays      -0.108333   0.036423  -2.974  0.00335 **
## Singles           4.927142  10.460040   0.471  0.63819
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.443 on 176 degrees of freedom
##   (2085 observations deleted due to missingness)
## Multiple R-squared:  0.5501, Adjusted R-squared:  0.5143
## F-statistic: 15.37 on 14 and 176 DF,  p-value: < 2.2e-16
```

```
lm4 <- lm(Wins ~ . -HomeRuns -StrikeOuts_AtBat -Doubles, data = training)
summary(lm4)
```

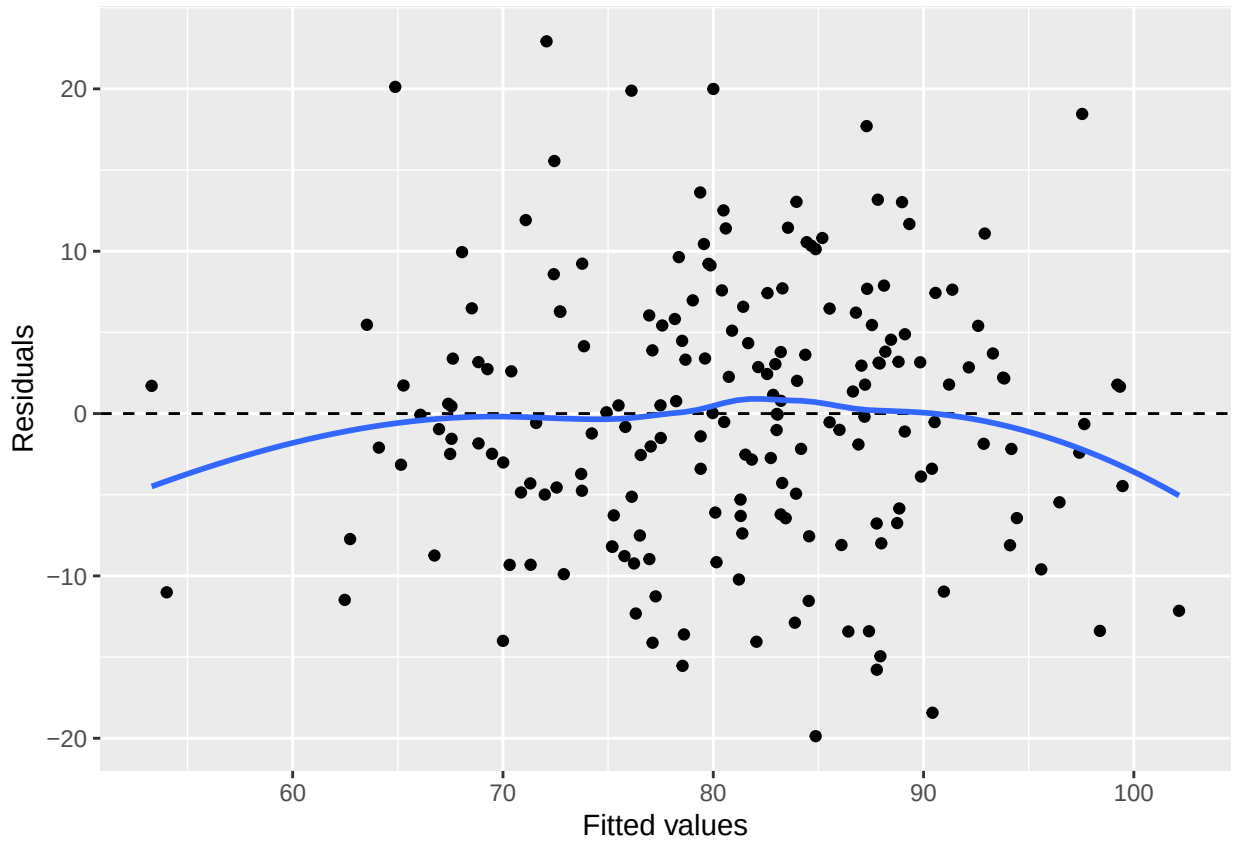
```
##
## Call:
## lm(formula = Wins ~ . - HomeRuns - StrikeOuts_AtBat - Doubles,
##     data = training)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -19.8310  -5.7335  -0.0574   5.4622  22.9967
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      60.615194  19.520050   3.105  0.00221 **
## Hits              1.550162   1.357665   1.142  0.25509
## Triples          -0.130537   0.081028  -1.611  0.10896
## Walks_AtBat      -4.328168   3.607076  -1.200  0.23178
## BasesStolen       0.035086   0.028053   1.251  0.21269
## OutStealingBases -0.011397   0.070985  -0.161  0.87262
```

```
## HitByPitch_AtBat  0.082749   0.049268   1.680  0.09481 .
## Hits_Allowed     -1.501239   1.356766  -1.106  0.27002
## HomeRuns_Allowed  0.061672   0.036975   1.668  0.09709 .
## Walks_Allowed     4.379004   3.604494   1.215  0.22603
## StrikeOuts       -0.031965   0.007418  -4.309 2.71e-05 ***
## Errors           -0.172725   0.041167  -4.196 4.30e-05 ***
## DoublePlays      -0.107670   0.036316  -2.965  0.00345 **
## Singles          -0.026614   0.030029  -0.886  0.37667
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.425 on 177 degrees of freedom
##   (2085 observations deleted due to missingness)
## Multiple R-squared:  0.5495, Adjusted R-squared:  0.5164
## F-statistic: 16.61 on 13 and 177 DF,  p-value: < 2.2e-16
```

```
# I've gone too far R-Squared has gone down
```

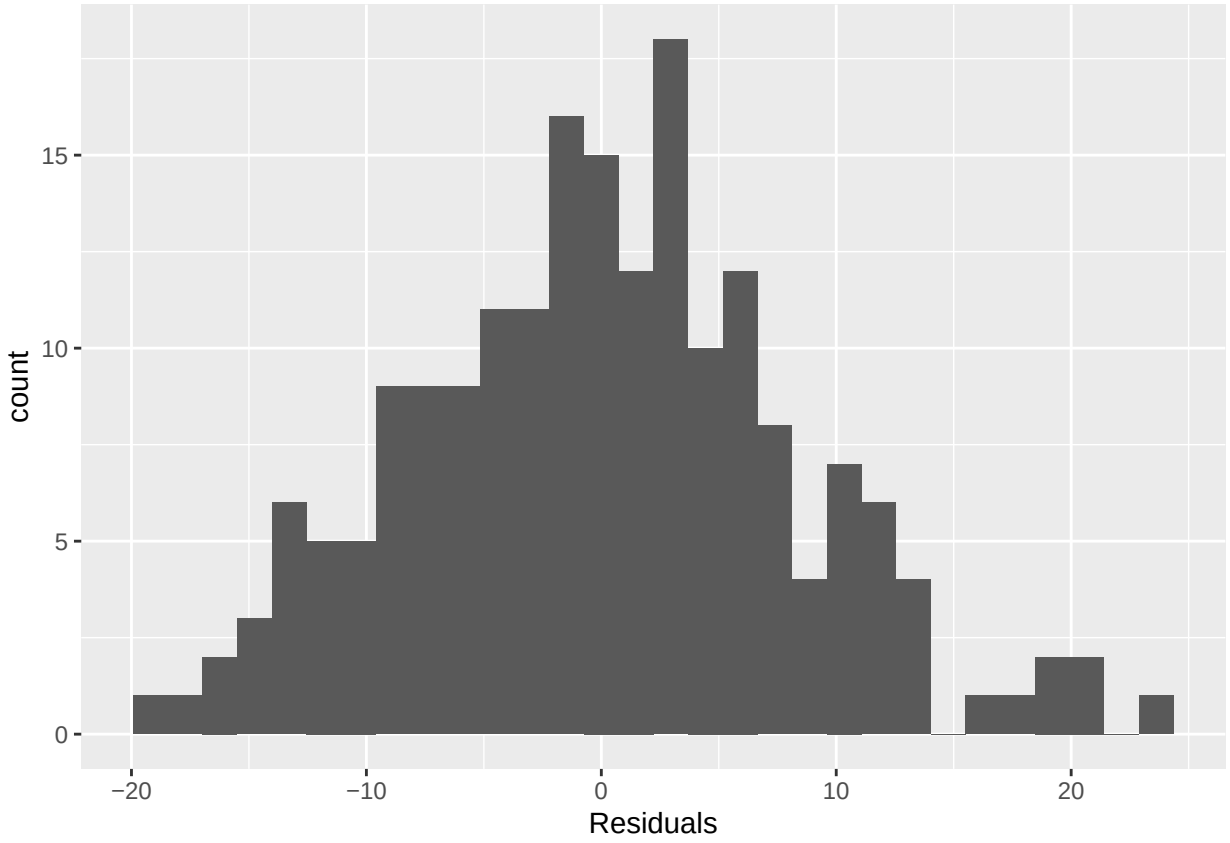
```
ggplot(data = lm2, aes(x = .fitted, y = .resid)) +
  geom_point() +
  geom_hline(yintercept = 0, linetype = "dashed") +
  geom_smooth(se = FALSE) +
  xlab("Fitted values") +
  ylab("Residuals")
```

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

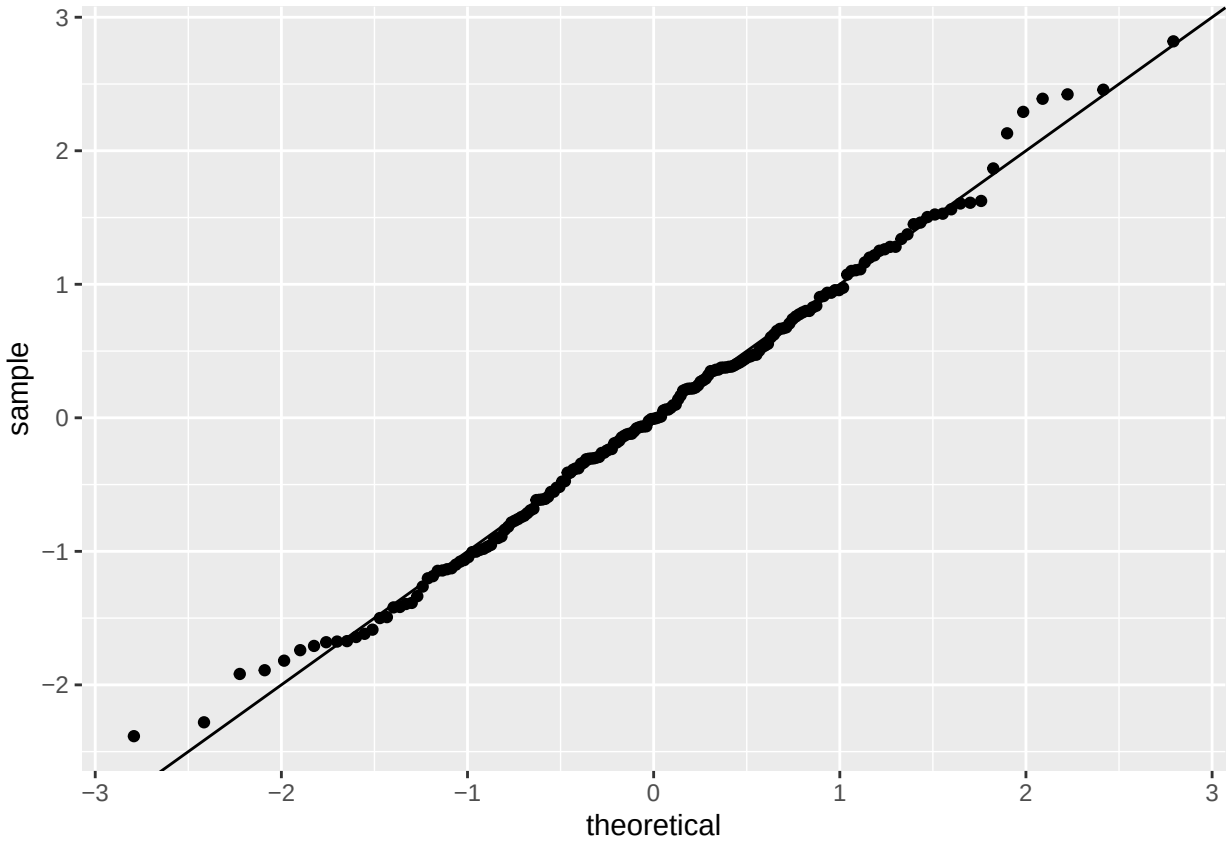



```
ggplot(data = lm2, aes(x = .resid)) +
  geom_histogram() +
  xlab("Residuals")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ggplot(data = lm2) +
  stat_qq(aes(sample = .stdresid)) +
  geom_abline()
```



0.6 References

https://www.kaggle.com/reisel/how-to-handle-correlated-features