

# Logistic Regression Models & Results

Jonathan Olds

11/9/2020

## Loading and Prepping Data

```
batters = read_csv("BattersData.csv")

## Parsed with column specification:
## cols(
##   .default = col_double(),
##   Name = col_character(),
##   Tm = col_character(),
##   Lg = col_character(),
##   PosSummary_Field = col_character()
## )

## See spec(...) for full column specifications.

pitchers = read_csv("PitchersData.csv")

## Parsed with column specification:
## cols(
##   .default = col_double(),
##   Name = col_character(),
##   Tm = col_character(),
##   Lg = col_character()
## )

## See spec(...) for full column specifications.

# Removing 2008 & 2009 seasons - no list
batters = batters %>% filter(Season >= 2010)
pitchers = pitchers %>% filter(Season >= 2010)

# Creating Test & Train data for batters
set.seed(2020)
batters$Top100 = as.factor(batters$Top100)
batters_pi = unique(batters$playerid)
batters_piselections = batters_pi[createDataPartition(y = batters_pi, p = 0.7, list = FALSE)]
batters_train = batters[which(batters$playerid %in% batters_piselections),]
batters_test = batters[which(batters$playerid %!in% batters_piselections),]
which(batters$Name %in% batters_train & batters$Name %in% batters_test)

## integer(0)

# Create Test & Train data for pitchers
set.seed(2020)
pitchers$Top100 = factor(pitchers$Top100)
pitchers_pi = unique(pitchers$playerid)
```

```

pitchers_piselections = pitchers_pi[createDataPartition(y = pitchers_pi, p = 0.7, list = FALSE)]
pitchers_train = pitchers[which(pitchers$playerid %in% pitchers_piselections),]
pitchers_test = pitchers[which(pitchers$playerid %!in% pitchers_piselections),]
which(pitchers$Name %in% pitchers_train & pitchers$Name %in% pitchers_test)

```

```
## integer(0)
```

```

# Dataframe of only pca-eligible variables - batters
batters_active = batters[,c(88:167)]
batters_active = batters_active[,which(apply(batters_active, 2, sd) != 0)]

# Dataframe of only pca-eligible variables - pitchers
pitchers_active = pitchers[,74:140]
pitchers_active = pitchers_active[,which(apply(pitchers_active, 2, sd) != 0)]

# Function to calculate test accuracy
calc_accuracy_function = function(actual, predicted) {
  mean(actual == predicted)
}

```

## Traditional PCA & LR - Batters

Variables Included - Hits, Walks, Strikeouts, Doubles, Triples, Home Runs, Total Bases, Runs Batted In, Batting Average, On Base Percentage, Slugging Percentage, On Base + Slugging, Stolen Bases, Caught Stealing, Runs Created per Game, Putouts, Assists, Errors, Double Plays, Fielding Percentage, Range Factor/9 Innings, All Star, Gold Glove, Silver Slugger, MVP Rank & Vote Points, Rookie of the Year Rank & Vote Points

```

# Selecting variables
batters_traditional = batters_active[,c(4:15,46:47,31,57:62,68:72,75:76)]
# PCA
pca_batters_traditional = prcomp(batters_traditional, scale. = TRUE)
summary(pca_batters_traditional)

```

```
## Importance of components:
```

```

##          PC1      PC2      PC3      PC4      PC5      PC6      PC7
## Standard deviation  2.9388 1.7904 1.62347 1.29731 1.26462 1.21972 1.06107
## Proportion of Variance 0.3085 0.1145 0.09413 0.06011 0.05712 0.05313 0.04021
## Cumulative Proportion 0.3085 0.4229 0.51706 0.57717 0.63428 0.68742 0.72763
##          PC8      PC9      PC10     PC11     PC12     PC13     PC14
## Standard deviation  1.00601 0.96358 0.92043 0.86406 0.82590 0.80536 0.76788
## Proportion of Variance 0.03615 0.03316 0.03026 0.02666 0.02436 0.02316 0.02106
## Cumulative Proportion 0.76377 0.79693 0.82719 0.85385 0.87821 0.90138 0.92244
##          PC15     PC16     PC17     PC18     PC19     PC20     PC21
## Standard deviation  0.67243 0.65646 0.59478 0.48479 0.40753 0.38659 0.35311
## Proportion of Variance 0.01615 0.01539 0.01263 0.00839 0.00593 0.00534 0.00445
## Cumulative Proportion 0.93859 0.95398 0.96661 0.97500 0.98094 0.98627 0.99073
##          PC22     PC23     PC24     PC25     PC26     PC27
## Standard deviation  0.32744 0.24389 0.21275 0.17091 0.13589 0.005121
## Proportion of Variance 0.00383 0.00212 0.00162 0.00104 0.00066 0.000000
## Cumulative Proportion 0.99456 0.99668 0.99830 0.99934 1.00000 1.000000
##          PC28
## Standard deviation  1.414e-05
## Proportion of Variance 0.000e+00
## Cumulative Proportion 1.000e+00

```

```
cbind(batters[1:6,1],pca_batters_traditional$x[1:6,1:6])
```

```
##           Name      PC1      PC2      PC3      PC4      PC5
## 1   A.J. Ellis -1.7817308 2.250087 -0.67723277 -1.253646 -1.2095306
## 2   A.J. Ellis -2.1023261 2.633659 0.38138262 -1.357371 0.2617816
## 3   A.J. Ellis -3.4135491 2.817530 -0.06135366 -1.577257 -0.2328134
## 4 A.J. Pierzynski -0.4035745 1.433503 0.62050661 -1.218516 -0.6651692
## 5 A.J. Pierzynski -0.7206019 1.437979 0.37298023 -1.464506 -1.0104501
## 6 A.J. Pierzynski -0.1896964 1.632686 0.27890080 -1.097944 -0.6942622
##           PC6
## 1 0.8593889
## 2 1.2425418
## 3 0.9081588
## 4 0.6067756
## 5 0.5233870
## 6 0.6924411
```

```
head(pca_batters_traditional$scale^2, n = 6)
```

```
##   H_Bat_3yravg BB_Bat_3yravg SO_Bat_3yravg Doubles_3yravg Triples_3yravg
## 1111.443108    333.499422    918.949093    56.514469    3.732683
##   HR_Bat_3yravg
##    68.971448
```

```
# Assigning two elements of variability to vectors
```

```
PVE = summary(pca_batters_traditional)$importance[2,]
```

```
CVE = summary(pca_batters_traditional)$importance[3,]
```

```
# Graph of variability explained
```

```
PVEplot <- qplot(c(1:20), PVE[1:20]) +
```

```
  geom_bar(stat = "Identity") +
```

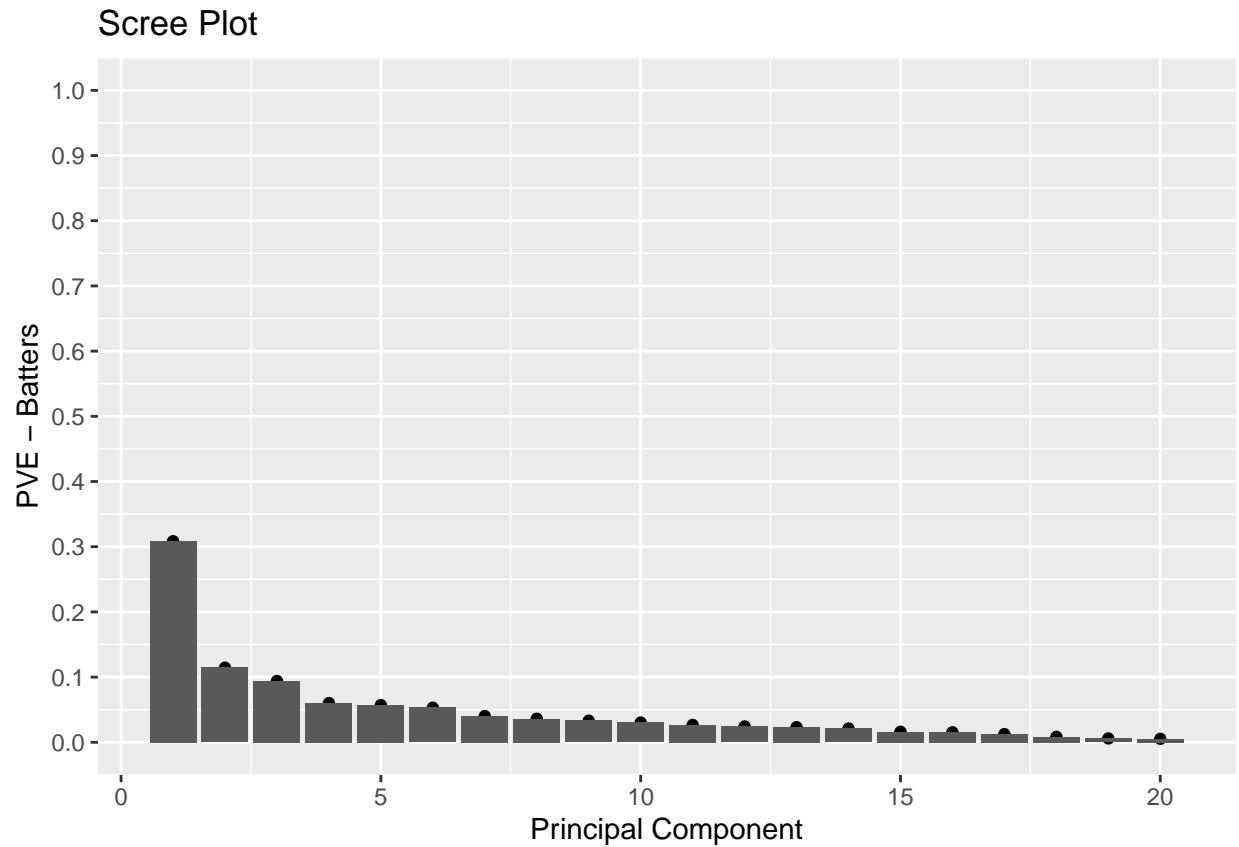
```
  xlab("Principal Component") +
```

```
  ylab("PVE - Batters") +
```

```
  ggtitle("Scree Plot") +
```

```
  scale_y_continuous(limits = c(0,1), breaks = seq(0,1,0.1))
```

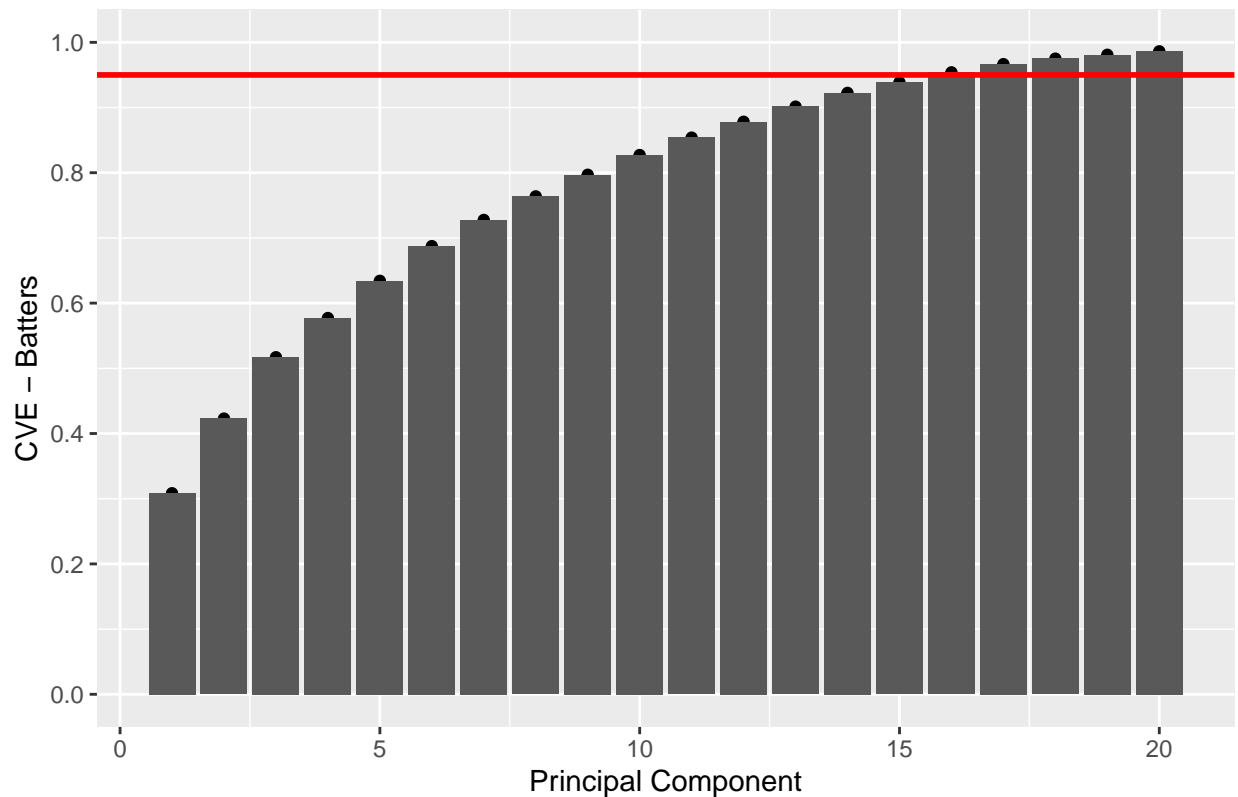
```
PVEplot
```



```
# Flattens out after the 17th PCA component

#Graph of cumulative variance explained
CVEplot <- qplot(c(1:20), CVE[1:20]) +
  geom_bar(stat = "Identity") +
  xlab("Principal Component") +
  ylab("CVE - Batters") +
  ggtitle("Scree Plot") +
  scale_y_continuous(limits = c(0,1), breaks = seq(0,1,0.2)) +
  geom_hline(yintercept = 0.95, color = "red", lwd = 1)
CVEplot
```

Scree Plot



```
# Reaches 95% cumulative variability after 16th component
```

```
# Adding pca components and descriptive values to dataset
```

```
batters_traditional = cbind(batters_traditional, pca_batters_traditional$x[,1:16])
batters_traditional = cbind(batters[,c(1:6,190:193)], batters_traditional)
```

```
# Dividing into test and train
```

```
batters_train = batters_traditional[which(batters_traditional$playerid %in% batters_piselections),]
batters_test = batters_traditional[which(batters_traditional$playerid %!in% batters_piselections),]
```

```
# Logistic regression model
```

```
set.seed(2020)
```

```
batters_glm_traditional = train(
  form = Top100 ~ PC1 + PC2 + PC3 + PC4 + PC5 + PC6 + PC7 + PC8 + PC9 + PC10 + PC11 +
    PC12 + PC13 + PC14 + PC15 + PC16,
  data = batters_train,
  trControl = trainControl(method = "cv", number = 10),
  method = "glm",
  family = "binomial"
)
```

```
# Model summary
```

```
batters_glm_traditional
```

```
## Generalized Linear Model
```

```
##
```

```
## 2075 samples
## 16 predictor
## 2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1868, 1868, 1867, 1868, 1867, 1867, ...
## Resampling results:
##
## Accuracy Kappa
## 0.8751503 0.6284962
```

#### *# Model values*

```
batters_glm_traditional$finalModel
```

```
##
## Call: NULL
##
## Coefficients:
## (Intercept) PC1 PC2 PC3 PC4 PC5
## -1.820592 0.871360 -0.134018 -0.178487 -0.046128 -0.306205
## PC6 PC7 PC8 PC9 PC10 PC11
## -0.075178 -0.263394 0.514182 0.162792 -0.421786 0.067456
## PC12 PC13 PC14 PC15 PC16
## -0.093055 0.007351 0.459836 -0.253259 0.045947
##
## Degrees of Freedom: 2074 Total (i.e. Null); 2058 Residual
## Null Deviance: 2285
## Residual Deviance: 1229 AIC: 1263
```

#### *# Test data accuracy*

```
calc_accuracy_function(batters_test$Top100,
  predict(batters_glm_traditional, newdata = batters_test))
```

```
## [1] 0.8792711
```

#### *# Dataset of predictions v actual*

```
batterscomp_traditional = cbind(batters_train$Name, predict(batters_glm_traditional,
  newdata = batters_train,
  type = "prob"), batters_train$Top100)
head(batterscomp_traditional)
```

```
## batters_train$Name 0 1 batters_train$Top100
## 1 A.J. Ellis 0.9711324 0.028867608 0
## 2 A.J. Ellis 0.9894933 0.010506690 0
## 3 A.J. Ellis 0.9935247 0.006475291 0
## 4 A.J. Pierzynski 0.9558149 0.044185071 0
## 5 A.J. Pierzynski 0.9624850 0.037515043 0
## 6 A.J. Pierzynski 0.9403715 0.059628537 1
```

#### *# Confusion matrix*

```
batters_cm_trad = confusionMatrix(predict(batters_glm_traditional, newdata = batters_test), batters_test$Top100)
batters_cm_trad$table
```

```
## Reference
## Prediction 0 1
## 0 654 81
```

```
## 1 25 118
```

## Traditional PCA & LR - Pitchers

Variables Included - Wins, Losses, Complete Games, Shutouts, Saves, Hits, Runs, Earned Runs, Earned Run Average, Walks + Hits/Innings Pitched, Run Support Per 9, Home Runs Per 9, Walks Per 9, Strikeouts Per 9, Runs Against Per 9, All Star, Gold Glove, Silver Slugger, MVP Rank and Vote Points, Cy Young Rank and Vote Points, Rookie of the Year Rank and Vote Points

```
# Selecting variables
pitchers_traditional = pitchers_active[,c(1:2,5:7,13:18,20:23,50:54,57:58,61:62)]
# PCA
pca_pitchers_traditional = prcomp(pitchers_traditional, scale. = TRUE)
summary(pca_pitchers_traditional)
```

```
## Importance of components:
```

```
##          PC1      PC2      PC3      PC4      PC5      PC6      PC7
## Standard deviation  2.4743 2.0615 1.3744 1.18180 1.13564 1.09429 0.98493
## Proportion of Variance 0.2551 0.1771 0.0787 0.05819 0.05374 0.04989 0.04042
## Cumulative Proportion 0.2551 0.4322 0.5109 0.56907 0.62280 0.67270 0.71312
##          PC8      PC9      PC10     PC11     PC12     PC13     PC14
## Standard deviation  0.98323 0.93767 0.89762 0.85965 0.81155 0.78867 0.74267
## Proportion of Variance 0.04028 0.03663 0.03357 0.03079 0.02744 0.02592 0.02298
## Cumulative Proportion 0.75340 0.79003 0.82360 0.85440 0.88184 0.90775 0.93074
##          PC15     PC16     PC17     PC18     PC19     PC20     PC21
## Standard deviation  0.66339 0.60247 0.57686 0.41680 0.36798 0.32320 0.27975
## Proportion of Variance 0.01834 0.01512 0.01387 0.00724 0.00564 0.00435 0.00326
## Cumulative Proportion 0.94907 0.96420 0.97806 0.98530 0.99094 0.99529 0.99856
##          PC22     PC23     PC24
## Standard deviation  0.14572 0.10866 0.04022
## Proportion of Variance 0.00088 0.00049 0.00007
## Cumulative Proportion 0.99944 0.99993 1.00000
```

```
cbind(pitchers[1:6,1],pca_pitchers_traditional$x[1:6,1:6])
```

```
##          Name      PC1      PC2      PC3      PC4      PC5      PC6
## 1 A.J. Burnett -4.292265 -1.6865132 -1.304909 -1.3708570 -0.01060507 -0.4932591
## 2 A.J. Burnett -3.986755 -2.3172783 -1.185097 -1.4124966 -0.09406051 -0.5868490
## 3 A.J. Burnett -4.303699 -1.7869879 -1.263816 -1.2252483  0.15905185 -0.5520055
## 4 A.J. Burnett -3.607454 -0.6621535 -1.354841 -0.9805361  0.20482561 -0.4475592
## 5 A.J. Burnett -3.935994 -0.4447429 -1.457616 -1.2737249  0.34575563 -0.5661478
## 6 A.J. Burnett -3.028110 -0.2458442 -1.184939 -1.4300465  0.05722687 -0.8335369
```

```
head(pca_pitchers_traditional$scale^2, n = 6)
```

```
##          W_3yravg      L_3yravg      CG_3yravg      SHO_3yravg      SV_3yravg
## 1.371911e+01  8.998276e+00  4.124620e-01  9.723845e-02  5.121191e+01
## H_Pitch_3yravg
## 2.439816e+03
```

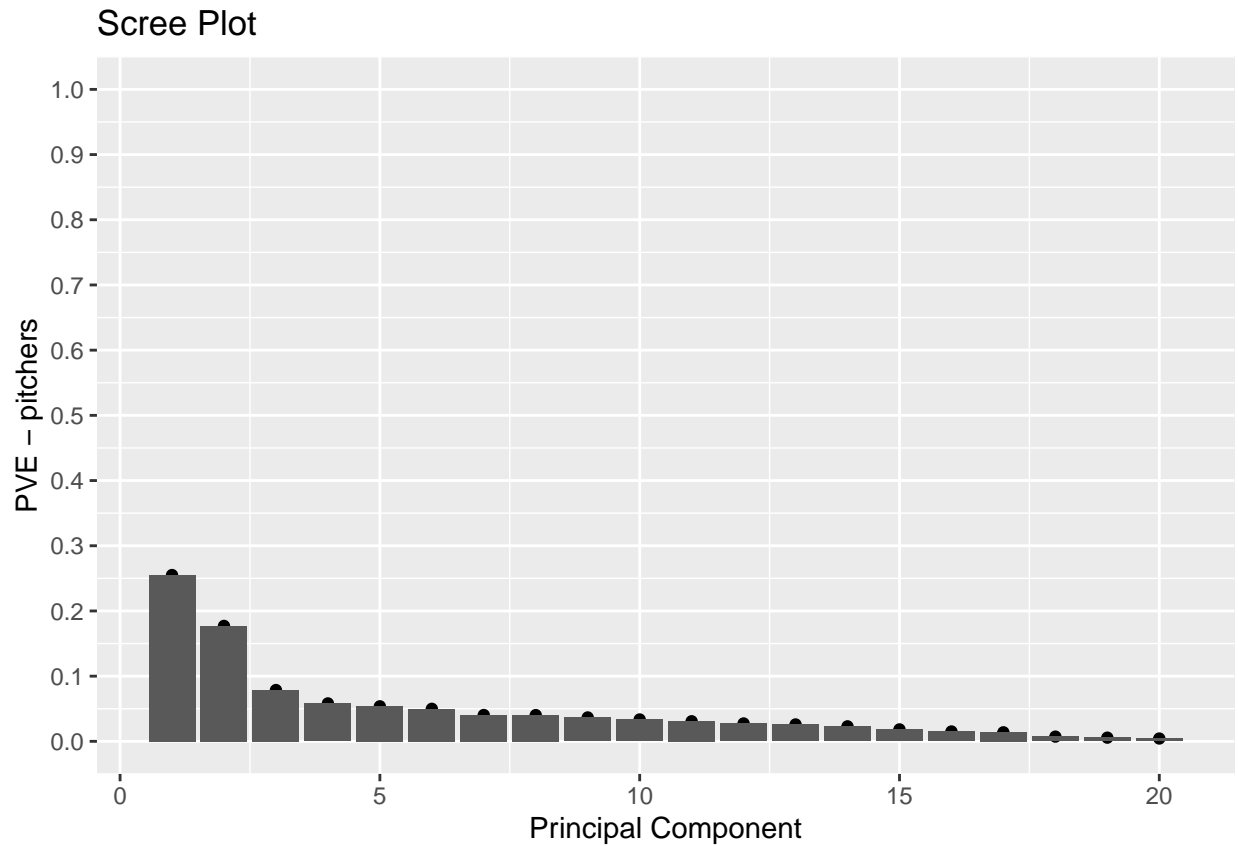
```
# Assigning elements of variability to vectors
PVE = summary(pca_pitchers_traditional)$importance[2,]
CVE = summary(pca_pitchers_traditional)$importance[3,]
```

```
# Graph of variability explained
PVEplot <- qplot(c(1:20), PVE[1:20]) +
  geom_bar(stat = "Identity") +
```

```

xlab("Principal Component") +
ylab("PVE - pitchers") +
ggtitle("Scree Plot") +
scale_y_continuous(limits = c(0,1), breaks = seq(0,1,0.1))
PVEplot

```



```

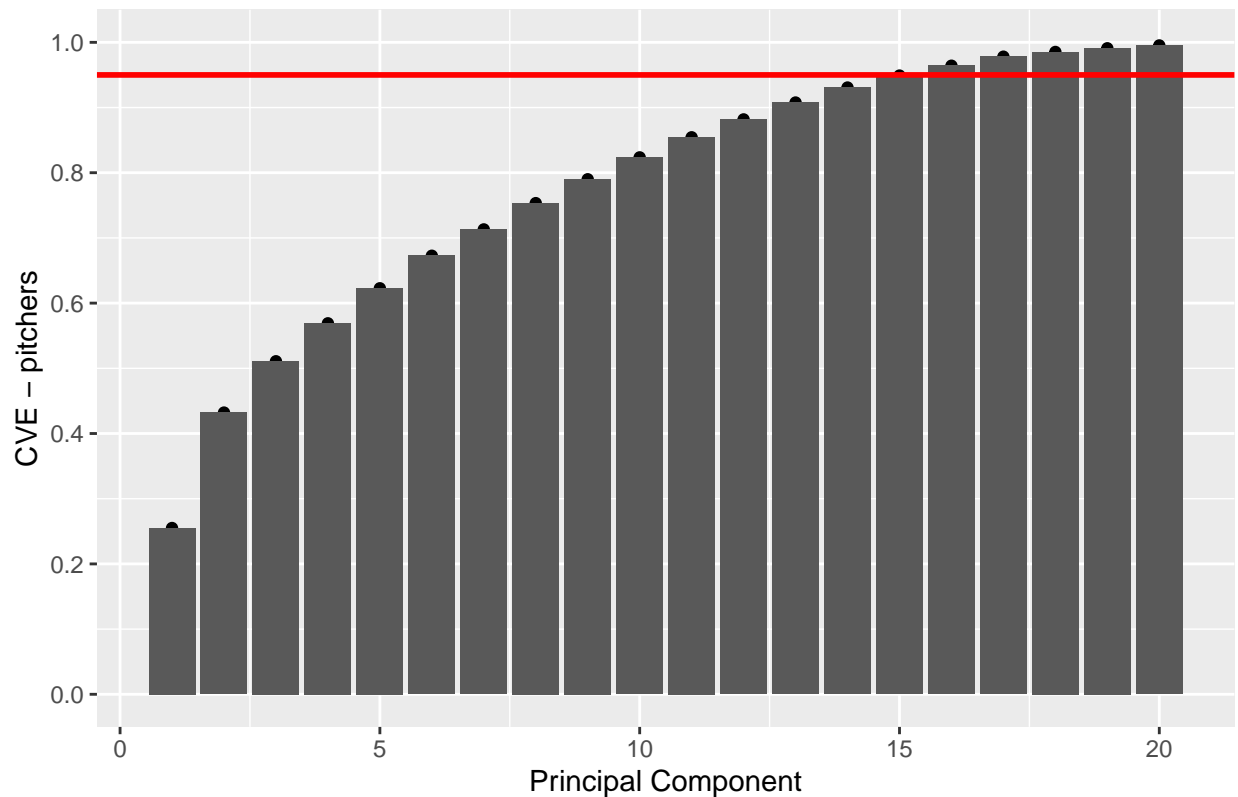
# Flattens out after the 17th PCA component

# Graph of cumulative variance explained
CVEplot <- qplot(c(1:20), CVE[1:20]) +
  geom_bar(stat = "Identity") +
  xlab("Principal Component") +
  ylab("CVE - pitchers") +
  ggtitle("Scree Plot") +
  scale_y_continuous(limits = c(0,1), breaks = seq(0,1,0.2)) +
  geom_hline(yintercept = 0.95, color = "red", lwd = 1)
CVEplot

```



Scree Plot



```
# Reaches 95% cumulative variability after 16th component
```

```
# Adding PCA components and descriptive statistics to dataset
```

```
pitchers_traditional = cbind(pitchers_traditional, pca_pitchers_traditional$x[,1:16])
pitchers_traditional = cbind(pitchers[,c(1:6,163:166)], pitchers_traditional)
```

```
# Dividing into test and train
```

```
pitchers_train = pitchers_traditional[which(pitchers_traditional$playerid %in% pitchers_piselections)]
pitchers_test = pitchers_traditional[which(pitchers_traditional$playerid %!in% pitchers_piselections)]
```

```
# Logistic Regression Model
```

```
set.seed(2020)
pitchers_glm_traditional = train(
  form = Top100 ~ PC1 + PC2,
  data = pitchers_train,
  trControl = trainControl(method = "cv", number = 10),
  method = "glm",
  family = "binomial"
)
```

```
# Model summary
```

```
pitchers_glm_traditional
```

```
## Generalized Linear Model
```

```
##
```

```
## 2789 samples
```

```
##      2 predictor
##      2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2510, 2510, 2510, 2510, 2510, 2510, ...
## Resampling results:
##
##      Accuracy   Kappa
##      0.9598424  0.6196805
```

```
# Model values
pitchers_glm_traditional$finalModel
```

```
##
## Call:  NULL
##
## Coefficients:
## (Intercept)          PC1          PC2
##      -4.640       -0.500        1.278
##
## Degrees of Freedom: 2788 Total (i.e. Null);  2786 Residual
## Null Deviance:      1382
## Residual Deviance: 624.7      AIC: 630.7
```

```
# Test accuracy
calc_accuracy_function(pitchers_test$Top100,
  predict(pitchers_glm_traditional, newdata = pitchers_test))
```

```
## [1] 0.9498767
```

```
# Dataset measuring predictions vs accuracy
pitcherscomp_traditional = cbind(pitchers_train$Name, predict(pitchers_glm_traditional,
  newdata = pitchers_train,
  type = "prob"), pitchers_train$Top100)

head(pitcherscomp_traditional)
```

```
##      pitchers_train$Name          0          1 pitchers_train$Top100
## 1      A.J. Burnett 0.9905195 0.009480454          0
## 2      A.J. Burnett 0.9963435 0.003656476          0
## 3      A.J. Burnett 0.9916046 0.008395372          0
## 4      A.J. Burnett 0.9754622 0.024537811          0
## 5      A.J. Burnett 0.9623366 0.037663379          0
## 6      A.J. Burnett 0.9689463 0.031053709          0
```

```
# Confusion Matrix
pitchers_cm_trad = confusionMatrix(predict(pitchers_glm_traditional, newdata = pitchers_test),pitchers_test$Name)
pitchers_cm_trad$table
```

```
##           Reference
## Prediction    0    1
##           0 1087   46
##           1   15   69
```

## Advanced PCA & LR - Batters

Variables Included - Batting Average on Balls in Play, Line Drive Percentage, Ground Ball Percentage, Fly Ball Percentage, Home Runs per Fly Ball, Pull Percent, Center Percent, Opposite Field Percent, Soft Contact Percent, Medium Contact Percent, Hard Contact Percent, Walks per Strikeout, weighted On Base Average, On Base Plus Slugging Plus, weighted Runs Created Plus, Batting Wins (BR), Wins Above Average, WAR (BR), oWAR (BR), WAR (FG), Runs Expectancy Wins, Win Probability Added, Player Leverage Index, Win Probability Added Per Leverage, Clutch, Offensive Winning Percentage, Batting Wins (FG), oWAR (FG), Ultimate Base Running, weighted Grounded into Double Plays, weighted Stolen Bases, Base Running Runs, Defensive Runs Saved (BR), Runs Saved on Good Plays, dWAR (BR), DRS (FG), dWAR (FG), All Star, Gold Glove, Silver Slugger, MVP Rank & Voting Points, Rookie of the Year Rank & Voting Points

```
# Selecting Variables
batters_advanced = batters_active[,c(16:29,32:45,48:51,63:72,75:76)]
# PCA
pca_batters_advanced = prcomp(batters_advanced, scale. = TRUE)
summary(pca_batters_advanced)
```

```
## Importance of components:
##              PC1      PC2      PC3      PC4      PC5      PC6      PC7
## Standard deviation  3.6229 2.3010 1.90395 1.62592 1.30766 1.2356 1.18013
## Proportion of Variance 0.2983 0.1203 0.08239 0.06008 0.03886 0.0347 0.03165
## Cumulative Proportion 0.2983 0.4186 0.50102 0.56110 0.59996 0.6347 0.66631
##              PC8      PC9      PC10     PC11     PC12     PC13     PC14
## Standard deviation  1.12971 1.06371 1.01225 0.98767 0.97367 0.94487 0.90239
## Proportion of Variance 0.02901 0.02572 0.02329 0.02217 0.02155 0.02029 0.01851
## Cumulative Proportion 0.69532 0.72103 0.74432 0.76649 0.78804 0.80833 0.82683
##              PC15     PC16     PC17     PC18     PC19     PC20     PC21
## Standard deviation  0.88157 0.85104 0.82331 0.80363 0.78623 0.74636 0.73071
## Proportion of Variance 0.01766 0.01646 0.01541 0.01468 0.01405 0.01266 0.01214
## Cumulative Proportion 0.84450 0.86096 0.87636 0.89104 0.90509 0.91775 0.92988
##              PC22     PC23     PC24     PC25     PC26     PC27     PC28
## Standard deviation  0.70874 0.66353 0.65521 0.61988 0.60918 0.53185 0.4297
## Proportion of Variance 0.01142 0.01001 0.00976 0.00873 0.00843 0.00643 0.0042
## Cumulative Proportion 0.94130 0.95131 0.96106 0.96980 0.97823 0.98466 0.9889
##              PC29     PC30     PC31     PC32     PC33     PC34     PC35
## Standard deviation  0.35842 0.33353 0.29418 0.24480 0.20504 0.15913 0.12652
## Proportion of Variance 0.00292 0.00253 0.00197 0.00136 0.00096 0.00058 0.00036
## Cumulative Proportion 0.99178 0.99430 0.99627 0.99763 0.99859 0.99916 0.99953
##              PC36     PC37     PC38     PC39     PC40     PC41     PC42
## Standard deviation  0.09220 0.08320 0.04465 0.03993 0.03104 0.02377 0.01565
## Proportion of Variance 0.00019 0.00016 0.00005 0.00004 0.00002 0.00001 0.00001
## Cumulative Proportion 0.99972 0.99988 0.99992 0.99996 0.99998 0.99999 1.00000
##              PC43     PC44
## Standard deviation  0.003612 0.001772
## Proportion of Variance 0.000000 0.000000
## Cumulative Proportion 1.000000 1.000000
```

```
cbind(batters[1:6,1],pca_batters_advanced$x[1:6,1:6])
```

```
##           Name      PC1      PC2      PC3      PC4      PC5
## 1  A.J. Ellis 0.2062065 -0.1002327 -1.4889009 0.4853071 0.343873751
## 2  A.J. Ellis 1.8376461 -0.4808878 -1.6188972 1.8248646 -0.190853413
## 3  A.J. Ellis 2.2421148 -0.9227380 -1.0371102 1.6539492 -0.826363864
## 4 A.J. Pierzynski 3.6595899 -1.0111134 -0.6137027 1.4369921 0.367040748
## 5 A.J. Pierzynski 3.4761050 -0.3749331 -1.0914236 1.8576937 -0.023311120
```

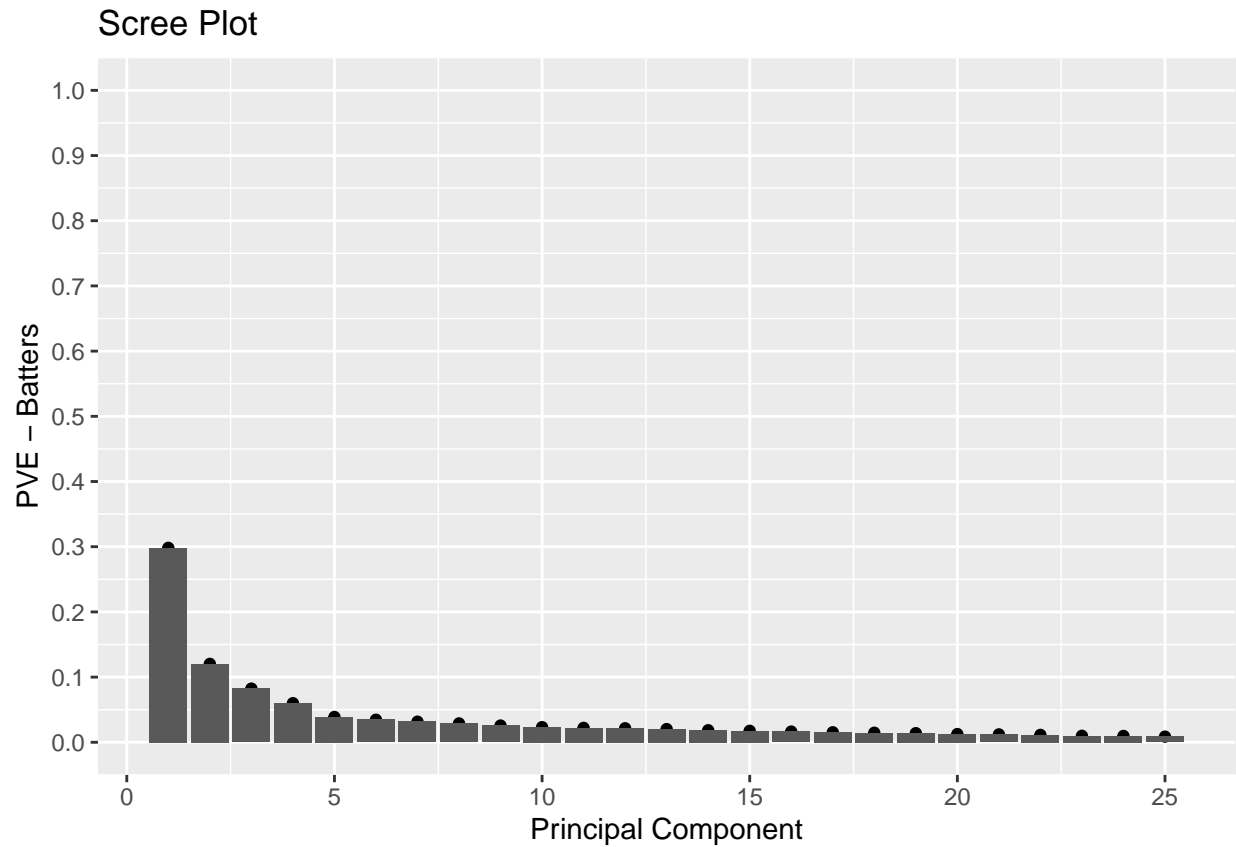
```
## 6 A.J. Pierzynski 2.1192048 -0.8670503 -0.6787158 1.2259140 -0.001398269
##          PC6
## 1 -0.61895130
## 2 -0.30155604
## 3  0.01638817
## 4 -0.41074673
## 5 -0.34912123
## 6 -0.33819364
```

```
head(pca_batters_advanced$scale^2, n = 6)
```

```
##          BA bip_3yravg  LDPercent_Bat_3yravg  GBPercent_Bat_3yravg
##          0.00114843      7.30450385      41.57701196
##  FBPercent_Bat_3yravg  HRPerFB_3yravg  PullPercent_Bat_3yravg
##          43.84569385      31.89357589      31.84198988
```

```
# Assigning elements of variability to vectors
PVE = summary(pca_batters_advanced)$importance[2,]
CVE = summary(pca_batters_advanced)$importance[3,]
```

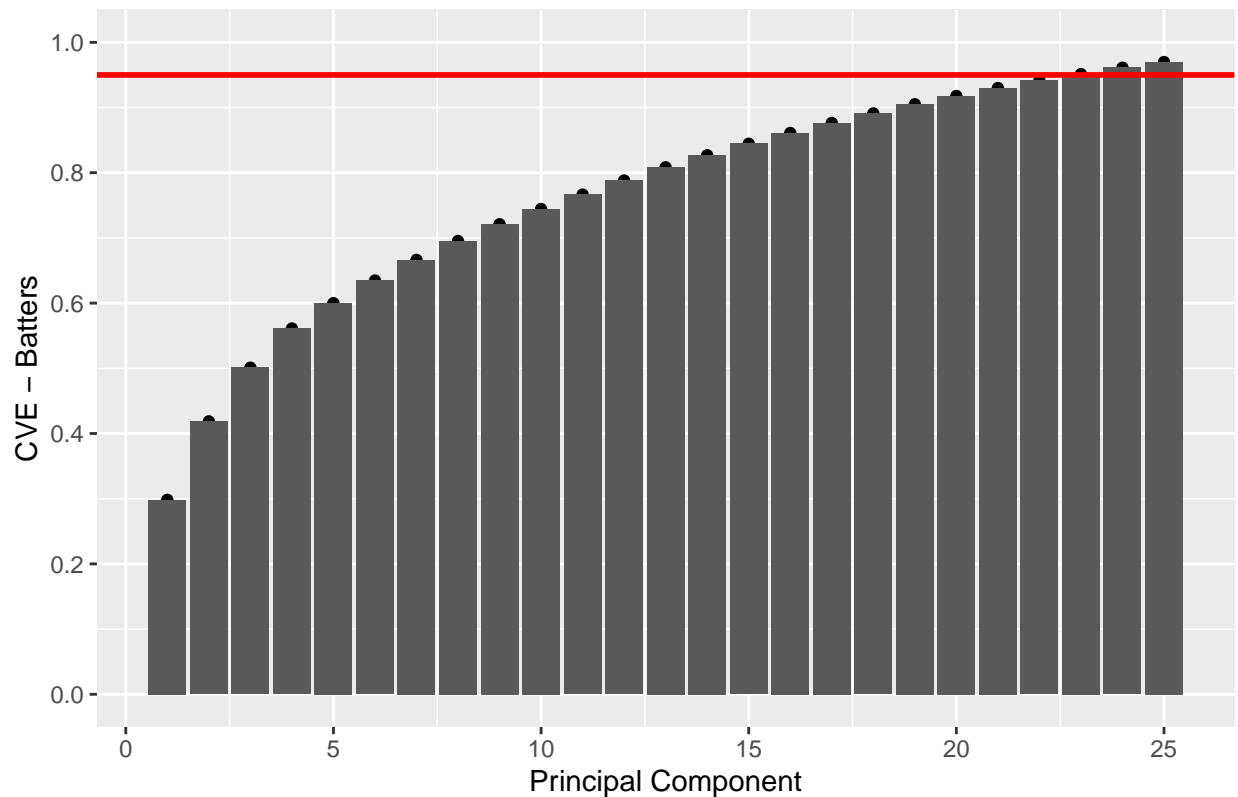
```
# Graph of variability explained
PVEplot <- qplot(c(1:25), PVE[1:25]) +
  geom_bar(stat = "Identity") +
  xlab("Principal Component") +
  ylab("PVE - Batters") +
  ggtitle("Scree Plot") +
  scale_y_continuous(limits = c(0,1), breaks = seq(0,1,0.1))
PVEplot
```



```
# Flattens out after the 18th PCA component

# Graph of cumulative variability explained
CVEplot <- qplot(c(1:25), CVE[1:25]) +
  geom_bar(stat = "Identity") +
  xlab("Principal Component") +
  ylab("CVE - Batters") +
  ggtitle("Scree Plot") +
  scale_y_continuous(limits = c(0,1), breaks = seq(0,1,0.2)) +
  geom_hline(yintercept = 0.95, color = "red", lwd = 1)
CVEplot
```

### Scree Plot



```
# Reaches 95% cumulative variability after 23rd component

# Adding PCA components and descriptive statistics back to dataset
batters_advanced = cbind(batters_advanced, pca_batters_advanced$x[,1:23])
batters_advanced = cbind(batters[,c(1:6,190:193)], batters_advanced)

# Dividing into test and train
batters_train = batters_advanced[which(batters_advanced$playerid %in% batters_piselections),]
batters_test = batters_advanced[which(batters_advanced$playerid %!in% batters_piselections),]

# Logistic regression model
set.seed(2020)
batters_glm_advanced = train(
  form = Top100 ~ PC1 + PC2 + PC3 + PC4 + PC5 + PC6 + PC7 + PC8 + PC9 + PC10 +
    PC11 + PC12 + PC13 + PC14,
  data = batters_train,
  trControl = trainControl(method = "cv", number = 10),
  method = "glm",
  family = "binomial"
)

# Model summary
batters_glm_advanced
```

```
## Generalized Linear Model
##
```

```
## 2075 samples
## 14 predictor
## 2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1868, 1868, 1867, 1868, 1867, 1867, ...
## Resampling results:
##
## Accuracy Kappa
## 0.8954124 0.6967778
```

```
# Model values
batters_glm_advanced$finalModel
```

```
##
## Call: NULL
##
## Coefficients:
## (Intercept) PC1 PC2 PC3 PC4 PC5
## -2.19094 -0.91008 0.31294 0.12142 -0.01471 0.10522
## PC6 PC7 PC8 PC9 PC10 PC11
## 0.47299 -0.32217 0.28606 -0.17598 0.06734 -0.07803
## PC12 PC13 PC14
## 0.03779 0.15950 -0.27304
##
## Degrees of Freedom: 2074 Total (i.e. Null); 2060 Residual
## Null Deviance: 2285
## Residual Deviance: 966.1 AIC: 996.1
```

```
# Test accuracy
calc_accuracy_function(batters_test$Top100,
  predict(batters_glm_advanced, newdata = batters_test))
```

```
## [1] 0.88041
```

```
# Dataset measuring predictions vs accuracy
batterscomp_advanced = cbind(batters_train$Name, predict(batters_glm_advanced,
  newdata = batters_train,
  type = "prob"), batters_train$Top100)

head(batterscomp_advanced)
```

```
## batters_train$Name 0 1 batters_train$Top100
## 1 A.J. Ellis 0.8998968 0.100103187 0
## 2 A.J. Ellis 0.9759615 0.024038482 0
## 3 A.J. Ellis 0.9849687 0.015031260 0
## 4 A.J. Pierzynski 0.9903079 0.009692068 0
## 5 A.J. Pierzynski 0.9868904 0.013109571 0
## 6 A.J. Pierzynski 0.9702758 0.029724209 1
```

```
# Confusion Matrix
batters_cm_adv = confusionMatrix(predict(batters_glm_advanced, newdata = batters_test),batters_test$Top100)
batters_cm_adv$table
```

```
## Reference
## Prediction 0 1
## 0 638 64
```

```
## 1 41 135
```

## Advanced PCA & LR - Pitchers

Variables Included - Left on Base Percentage, Line Drive Percentage, Fly Ball Percentage, Ground Ball Percentage, Soft Contact Percentage, Medium Contact Percentage, Hard Contact Percentage, Batting Average on Balls in Play, Earned Run Average Plus, Fielding Independent Pitching, Expected Fielding Independent Pitching, Skill Interactive Earned Run Average, WAR (FG), game Leverage, Wins Above Average, Wins Above Average adjusted, Wins Above Average Win-Loss %, Full Season Win-Loss Percent, WAR (BR), Win Probability Added, Runs Expectancy Wins, player Leverage Index, Win Probability Added per Leverage, Clutch, Shutdowns, Meltdowns, All Star, Gold Glove, Silver Slugger, MVP Rank & Vote Points, Cy Young Rank & Vote Points, Rookie of the Year Rank & Vote Points

```
# Selecting variables
pitchers_advanced = pitchers_active[,c(24:49,50:54,57:58,61:62)]
# PCA
pca_pitchers_advanced = prcomp(pitchers_advanced, scale. = TRUE)
summary(pca_pitchers_advanced)
```

```
## Importance of components:
##          PC1    PC2    PC3    PC4    PC5    PC6    PC7
## Standard deviation  2.999 2.117 1.54217 1.28241 1.25847 1.20316 1.14822
## Proportion of Variance 0.257 0.128 0.06795 0.04699 0.04525 0.04136 0.03767
## Cumulative Proportion 0.257 0.385 0.45296 0.49994 0.54519 0.58655 0.62422
##          PC8    PC9    PC10    PC11    PC12    PC13    PC14
## Standard deviation  1.10615 1.08014 1.0399 1.00804 0.98038 0.95609 0.93521
## Proportion of Variance 0.03496 0.03333 0.0309 0.02903 0.02746 0.02612 0.02499
## Cumulative Proportion 0.65918 0.69252 0.7234 0.75245 0.77991 0.80603 0.83101
##          PC15    PC16    PC17    PC18    PC19    PC20    PC21
## Standard deviation  0.87629 0.86084 0.80349 0.79440 0.75871 0.72141 0.63302
## Proportion of Variance 0.02194 0.02117 0.01845 0.01803 0.01645 0.01487 0.01145
## Cumulative Proportion 0.85295 0.87413 0.89257 0.91060 0.92705 0.94192 0.95337
##          PC22    PC23    PC24    PC25    PC26    PC27    PC28
## Standard deviation  0.58637 0.5551 0.5122 0.42935 0.39756 0.37545 0.29918
## Proportion of Variance 0.00982 0.0088 0.0075 0.00527 0.00452 0.00403 0.00256
## Cumulative Proportion 0.96319 0.9720 0.9795 0.98476 0.98928 0.99330 0.99586
##          PC29    PC30    PC31    PC32    PC33    PC34    PC35
## Standard deviation  0.2136 0.18124 0.17342 0.1180 0.10599 0.09061 0.05463
## Proportion of Variance 0.0013 0.00094 0.00086 0.0004 0.00032 0.00023 0.00009
## Cumulative Proportion 0.9972 0.99810 0.99896 0.9994 0.99968 0.99991 1.00000
```

```
cbind(pitchers[1:6,1],pca_pitchers_advanced$x[1:6,1:6])
```

```
##          Name    PC1    PC2    PC3    PC4    PC5    PC6
## 1 A.J. Burnett -0.7073864 -1.5006793 0.3892210 0.57042553 0.3149952 0.3802275
## 2 A.J. Burnett -2.5807611 -1.2578221 0.1097051 0.21058627 0.5107917 0.3346171
## 3 A.J. Burnett -2.5268710 -0.8552628 1.0823492 0.01536254 0.6343113 0.5469413
## 4 A.J. Burnett -0.1597402 -1.1210468 2.1716029 -0.13135818 0.8162257 0.5965805
## 5 A.J. Burnett -0.3927564 -1.1856439 2.8797992 -0.15370215 0.7560155 0.4588448
## 6 A.J. Burnett -0.1958948 -1.4623919 2.0846858 -0.73233606 -0.1666128 1.0206273
```

```
head(pca_pitchers_advanced$scale^2, n = 6)
```

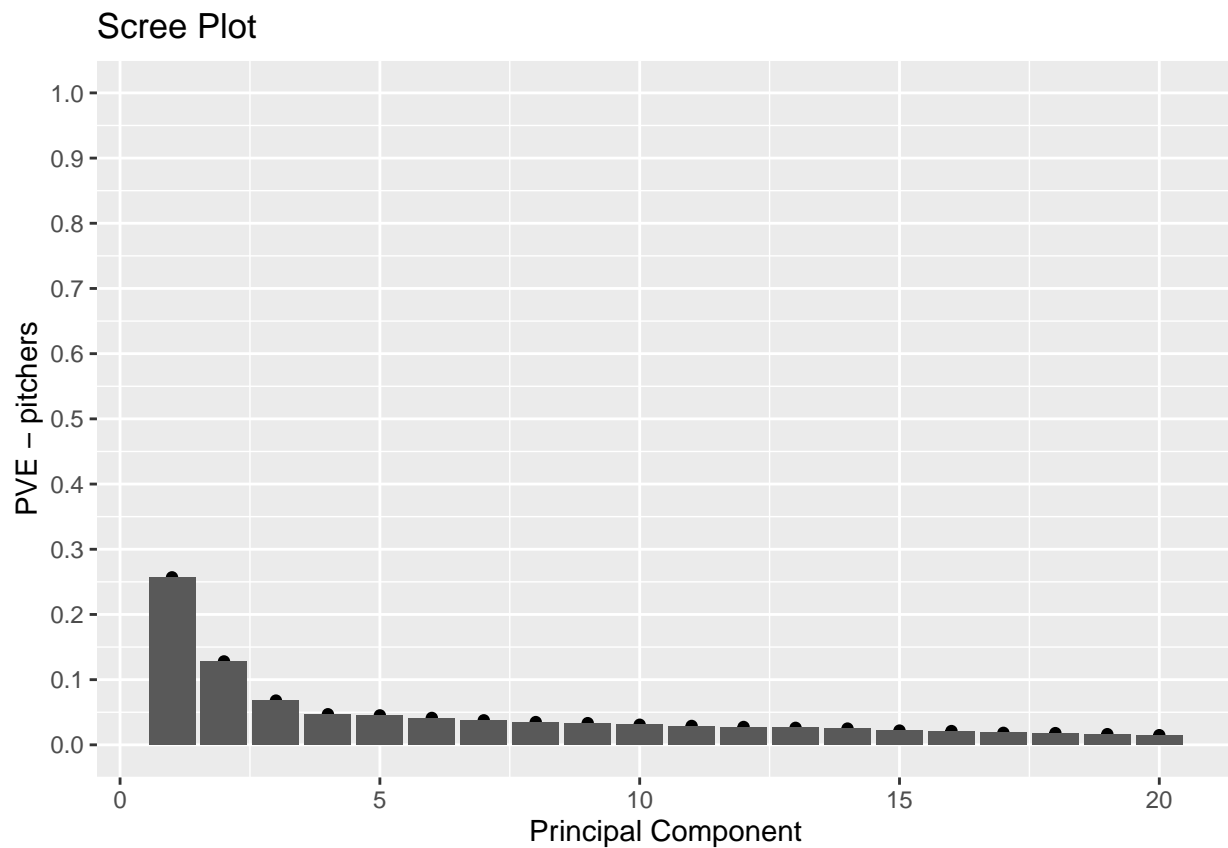
```
##          LOBPercent_3yravg  LDPercent_Pitch_3yravg  GBPercent_Pitch_3yravg
##          35.206658          7.274295          59.219351
##          FBPercent_Pitch_3yravg  SoftPercent_Pitch_3yravg  MedPercent_Pitch_3yravg
```



```
##                                56.146732                9.193243                18.871100
```

```
# Assigning elements of variability to vectors
PVE = summary(pca_pitchers_advanced)$importance[2,]
CVE = summary(pca_pitchers_advanced)$importance[3,]

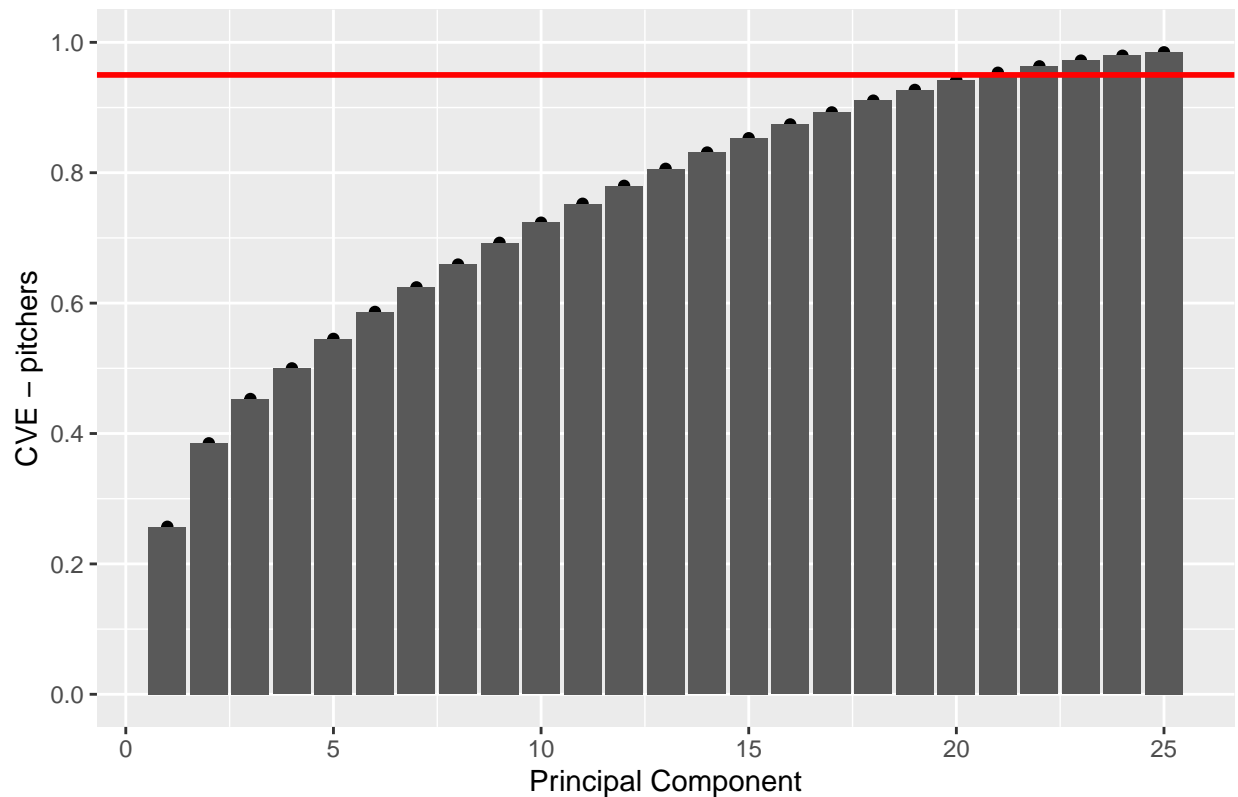
# Graph of variability explained
PVEplot <- qplot(c(1:20), PVE[1:20]) +
  geom_bar(stat = "Identity") +
  xlab("Principal Component") +
  ylab("PVE - pitchers") +
  ggtitle("Scree Plot") +
  scale_y_continuous(limits = c(0,1), breaks = seq(0,1,0.1))
PVEplot
```



```
# Flattens out after the 16th PCA component

CVEplot <- qplot(c(1:25), CVE[1:25]) +
  geom_bar(stat = "Identity") +
  xlab("Principal Component") +
  ylab("CVE - pitchers") +
  ggtitle("Scree Plot") +
  scale_y_continuous(limits = c(0,1), breaks = seq(0,1,0.2)) +
  geom_hline(yintercept = 0.95, color = "red", lwd = 1)
CVEplot
```

Scree Plot



```
# Reaches 95% cumulative variability after 21th component
```

```
# Adding PCA Components and descriptive values back to dataset
```

```
pitchers_advanced = cbind(pitchers_advanced, pca_pitchers_advanced$x[,1:21])
pitchers_advanced = cbind(pitchers[,c(1:6,163:166)], pitchers_advanced)
```

```
# Dividing into test and train
```

```
pitchers_train = pitchers_advanced[which(pitchers_advanced$playerid %in% pitchers_piselections),]
pitchers_test = pitchers_advanced[which(pitchers_advanced$playerid %!in% pitchers_piselections),]
```

```
# Logistic Regression Model
```

```
set.seed(2020)
pitchers_glm_advanced = train(
  form = Top100 ~ PC1 + PC2 + PC3 + PC4 + PC5 + PC6 + PC7 + PC8 + PC9 + PC10 + PC11,
  data = pitchers_train,
  trControl = trainControl(method = "cv", number = 10),
  method = "glm",
  family = "binomial"
)
```

```
# Model summary
```

```
pitchers_glm_advanced
```

```
## Generalized Linear Model
```

```
##
```

```
## 2789 samples
```

```
## 11 predictor
## 2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2510, 2510, 2510, 2510, 2510, 2510, ...
## Resampling results:
##
## Accuracy Kappa
## 0.9623514 0.6661509

# Model values
pitchers_glm_advanced$finalModel

##
## Call: NULL
##
## Coefficients:
## (Intercept) PC1 PC2 PC3 PC4 PC5
## -5.20764 0.89189 -0.39039 0.18459 -0.46000 0.38412
## PC6 PC7 PC8 PC9 PC10 PC11
## 0.30572 -0.11527 -0.06169 -0.29985 -0.13992 0.23161
##
## Degrees of Freedom: 2788 Total (i.e. Null); 2777 Residual
## Null Deviance: 1382
## Residual Deviance: 512.6 AIC: 536.6

# Test accuracy
calc_accuracy_function(pitchers_test$Top100,
  predict(pitchers_glm_advanced, newdata = pitchers_test))

## [1] 0.9605588

# Dataset to compare predictions vs actual
pitcherscomp_advanced = cbind(pitchers_train$Name, predict(pitchers_glm_advanced,
  newdata = pitchers_train,
  type = "prob"), pitchers_train$Top100)

head(pitcherscomp_advanced)

## pitchers_train$Name 0 1 pitchers_train$Top100
## 1 A.J. Burnett 0.9906324 0.009367640 0
## 2 A.J. Burnett 0.9981627 0.001837288 0
## 3 A.J. Burnett 0.9972427 0.002757340 0
## 4 A.J. Burnett 0.9768119 0.023188099 0
## 5 A.J. Burnett 0.9806553 0.019344713 0
## 6 A.J. Burnett 0.9717015 0.028298516 0

# Confusion Matrix
pitchers_cm_adv = confusionMatrix(predict(pitchers_glm_advanced, newdata = pitchers_test), pitchers_t
pitchers_cm_adv$table

##
## Reference
## Prediction 0 1
## 0 1085 31
## 1 17 84
```

## Combining PCA Elements and then LR - Batters

Variables Included - all PCA elements from traditional and advanced logistic regression models

```
# Merging PCA elements to one dataset
colnames(batters_traditional)[39:54] = gsub("PC", "PC_T", colnames(batters_traditional)[39:54])
colnames(batters_advanced)[55:77] = gsub("PC", "PC_A", colnames(batters_advanced)[55:77])
batters_trad_adv = cbind(batters_traditional, batters_advanced)
# Getting rid of non-PCA columns
batters_trad_adv = batters_trad_adv[,-c(32:38,55:64)]
batters_trad_adv = batters_trad_adv[,-c(11:31,48:91)]

# Dividing into test and train
batters_train = batters_trad_adv[which(batters_trad_adv$playerid %in% batters_piselections),]
batters_test = batters_trad_adv[which(batters_trad_adv$playerid %!in% batters_piselections),]

# Logistic Regression
set.seed(2020)
batters_glm_trad_adv = train(
  form = Top100 ~ PC_T1 + PC_T2 + PC_T3 + PC_T4 + PC_T5 + PC_T6 + PC_T7 + PC_T8 + PC_T9 +
    PC_T10 + PC_T11 + PC_T12 + PC_T13 + PC_T14 + PC_T15 + PC_T16 +
    PC_A1 + PC_A2 + PC_A3 + PC_A4 + PC_A5 + PC_A6 + PC_A7 + PC_A8 + PC_A9 + PC_A10 +
    PC_A11 + PC_A12 + PC_A13 + PC_A14 + PC_A15 + PC_A16 + PC_A17 + PC_A18 + PC_A19 + PC_A20 +
    PC_A21 + PC_A22 + PC_A23,
  data = batters_train,
  trControl = trainControl(method = "cv", number = 5),
  method = "glm",
  family = "binomial"
)

# Model summary
batters_glm_trad_adv
```

```
## Generalized Linear Model
##
## 2075 samples
## 39 predictor
## 2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 1659, 1661, 1659, 1660, 1661
## Resampling results:
##
## Accuracy Kappa
## 0.9002521 0.7170391
```

```
# Model values
batters_glm_trad_adv$finalModel
```

```
##
## Call: NULL
##
## Coefficients:
## (Intercept) PC_T1 PC_T2 PC_T3 PC_T4 PC_T5
## -2.35334 0.84792 -0.05273 -0.72067 0.99056 -1.41780
```

```
##      PC_T6      PC_T7      PC_T8      PC_T9      PC_T10      PC_T11
##      2.42415     -8.65299     -4.21852      3.22315     -6.04481      8.63330
##      PC_T12      PC_T13      PC_T14      PC_T15      PC_T16      PC_A1
##      3.69122     -3.38510     -4.23712      1.23822      0.23763     -0.17952
##      PC_A2      PC_A3      PC_A4      PC_A5      PC_A6      PC_A7
##      0.22872     -0.08555      0.99455     -2.49855     -0.14509      1.54285
##      PC_A8      PC_A9      PC_A10      PC_A11      PC_A12      PC_A13
##     -0.60673     -6.42919     -6.97509     -1.16171      0.96132      1.06363
##      PC_A14      PC_A15      PC_A16      PC_A17      PC_A18      PC_A19
##      4.58548      9.31489     -4.31973     -1.24214     -1.99687     -2.92949
##      PC_A20      PC_A21      PC_A22      PC_A23
##     -4.37549     -1.50067     -1.36792      0.51909
```

```
##
## Degrees of Freedom: 2074 Total (i.e. Null); 2035 Residual
## Null Deviance: 2285
## Residual Deviance: 883.1 AIC: 963.1
```

```
# Test accuracy
calc_accuracy_function(batters_test$Top100,
  predict(batters_glm_trad_adv, newdata = batters_test))
```

```
## [1] 0.8792711
```

```
# Dataset to compare predictions v accuracy
batterscomp_trad_adv = cbind(batters_train$Name, predict(batters_glm_trad_adv,
  newdata = batters_train,
  type = "prob"), batters_train$Top100)

head(batterscomp_trad_adv)
```

```
##      batters_train$Name      0      1 batters_train$Top100
## 1      A.J. Ellis 0.9349065 0.06509355      0
## 2      A.J. Ellis 0.9841825 0.01581747      0
## 3      A.J. Ellis 0.9890651 0.01093494      0
## 4      A.J. Pierzynski 0.9795149 0.02048509      0
## 5      A.J. Pierzynski 0.9821338 0.01786624      0
## 6      A.J. Pierzynski 0.9405821 0.05941792      1
```

```
# Confusion Matrix
batters_cm_trad_adv = confusionMatrix( predict(batters_glm_trad_adv, newdata = batters_test), batters.
batters_cm_trad_adv$table
```

```
##      Reference
## Prediction  0  1
##      0 637  64
##      1  42 135
```

## Combining PCA and then LR - Pitchers

Variables Included - All PCA components from traditional and advanced logistic regression models

```
# Combining PCA columns to one dataset
colnames(pitchers_traditional)[35:50] = gsub("PC", "PC_T", colnames(pitchers_traditional)[35:50])
colnames(pitchers_advanced)[46:66] = gsub("PC", "PC_A", colnames(pitchers_advanced)[46:66])
pitchers_trad_adv = cbind(pitchers_traditional, pitchers_advanced)
pitchers_trad_adv = pitchers_trad_adv[, -c(26:34, 51:60)]
```

```

pitchers_trad_adv = pitchers_trad_adv[,-c(11:25,42:76)]

# Dividing into test and train
pitchers_train = pitchers_trad_adv[which(pitchers_trad_adv$playerid %in% pitchers_piselections),]
pitchers_test = pitchers_trad_adv[which(pitchers_trad_adv$playerid %!in% pitchers_piselections),]

# Logistic Regression
set.seed(2020)
pitchers_glm_trad_adv = train(
  form = Top100 ~ PC_T1 + PC_T2 + PC_T3 + PC_T4 + PC_T5 + PC_T6 + PC_T7 + PC_T8 +
    PC_T9 + PC_T10 + PC_T11 + PC_T12 + PC_T13 + PC_T14 + PC_T15 +
    PC_A1 + PC_A2 + PC_A3 + PC_A4 + PC_A5 + PC_A6 + PC_A7 + PC_A8 + PC_A9 + PC_A10 + PC_A11 +
    PC_A12 + PC_A13 + PC_A14 + PC_A15,
  data = pitchers_train,
  trControl = trainControl(method = "cv", number = 5),
  method = "glm",
  family = "binomial"
)

# Model summary
pitchers_glm_trad_adv

```

```

## Generalized Linear Model
##
## 2789 samples
## 30 predictor
## 2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 2231, 2231, 2232, 2231, 2231
## Resampling results:
##
## Accuracy Kappa
## 0.9645007 0.6976765

```

```

# Model values
pitchers_glm_trad_adv$finalModel

```

```

##
## Call: NULL
##
## Coefficients:
## (Intercept)      PC_T1      PC_T2      PC_T3      PC_T4      PC_T5
## -6.81233    -1.09911     1.74010     6.54442     0.52452     7.05563
##      PC_T6      PC_T7      PC_T8      PC_T9      PC_T10      PC_T11
## -19.92486   -11.75328    -1.14180    -3.28097   -10.80626     0.51202
##      PC_T12      PC_T13      PC_T14      PC_T15      PC_A1      PC_A2
##  0.05635    -0.62108    -0.48841     1.08743    -0.24633    -0.07544
##      PC_A3      PC_A4      PC_A5      PC_A6      PC_A7      PC_A8
##  1.05638     1.79793     9.82820    12.04613    -1.40493     1.84088
##      PC_A9      PC_A10      PC_A11      PC_A12      PC_A13      PC_A14
##  7.50453    14.54923    33.98642   -12.32252   -22.28343    -1.62340
##      PC_A15

```

```
##      -3.62574
##
## Degrees of Freedom: 2788 Total (i.e. Null);  2758 Residual
## Null Deviance:      1382
## Residual Deviance: 446.7      AIC: 508.7

# Test accuracy
calc_accuracy_function(pitchers_test$Top100,
                       predict(pitchers_glm_trad_adv, newdata = pitchers_test))

## [1] 0.9416598

# Dataset to compare predictions v accuracy
pitcherscomp_trad_adv = cbind(pitchers_train$Name, predict(pitchers_glm_trad_adv,
                                                           newdata = pitchers_train,
                                                           type = "prob"), pitchers_train$Top100)

# Confusion Matrix
pitchers_cm_trad_adv = confusionMatrix(predict(pitchers_glm_trad_adv, newdata = pitchers_test),pitchers_train$Top100)
pitchers_cm_trad_adv

## Confusion Matrix and Statistics
##
##              Reference
## Prediction    0    1
##              0 1080   49
##              1   22   66
##
##              Accuracy : 0.9417
##              95% CI : (0.927, 0.9542)
##              No Information Rate : 0.9055
##              P-Value [Acc > NIR] : 2.823e-06
##
##              Kappa : 0.619
##
##  Mcnemar's Test P-Value : 0.002031
##
##              Sensitivity : 0.57391
##              Specificity : 0.98004
##              Pos Pred Value : 0.75000
##              Neg Pred Value : 0.95660
##              Prevalence : 0.09449
##              Detection Rate : 0.05423
##              Detection Prevalence : 0.07231
##              Balanced Accuracy : 0.77697
##
##              'Positive' Class : 1
##
```

## Performing PCA on Traditional and Advanced then LR - Batters

Variables Included - All traditional and advanced variables used in PCA

```
# Merging datasets
batters_both = cbind(batters_traditional, batters_advanced)
# Removing name, descriptive values and PCA
```

```
batters_both = batters_both[,-(55:64)]
batters_both = batters_both[, -c(1:10,39:54,92:121)]

# PCA
pca_batters_both= prcomp(batters_both, scale. = TRUE)
summary(pca_batters_both)
```

```
## Importance of components:
##          PC1      PC2      PC3      PC4      PC5      PC6      PC7
## Standard deviation  4.4717 2.6949 2.03837 1.98285 1.68254 1.44387 1.37389
## Proportion of Variance 0.3076 0.1117 0.06392 0.06049 0.04355 0.03207 0.02904
## Cumulative Proportion 0.3076 0.4194 0.48329 0.54377 0.58733 0.61940 0.64844
##          PC8      PC9      PC10     PC11     PC12     PC13     PC14
## Standard deviation  1.31548 1.24101 1.19368 1.1229 1.0635 1.01313 0.99994
## Proportion of Variance 0.02662 0.02369 0.02192 0.0194 0.0174 0.01579 0.01538
## Cumulative Proportion 0.67506 0.69876 0.72068 0.7401 0.7575 0.77327 0.78865
##          PC15     PC16     PC17     PC18     PC19     PC20     PC21
## Standard deviation  0.98560 0.96801 0.95540 0.94574 0.9157 0.90903 0.85256
## Proportion of Variance 0.01494 0.01442 0.01404 0.01376 0.0129 0.01271 0.01118
## Cumulative Proportion 0.80360 0.81801 0.83205 0.84581 0.8587 0.87143 0.88261
##          PC22     PC23     PC24     PC25     PC26     PC27     PC28
## Standard deviation  0.81674 0.78901 0.7733 0.75678 0.72760 0.69629 0.69017
## Proportion of Variance 0.01026 0.00958 0.0092 0.00881 0.00814 0.00746 0.00733
## Cumulative Proportion 0.89287 0.90245 0.9116 0.92046 0.92861 0.93607 0.94339
##          PC29     PC30     PC31     PC32     PC33     PC34     PC35
## Standard deviation  0.67659 0.65125 0.6347 0.6193 0.59093 0.56542 0.48600
## Proportion of Variance 0.00704 0.00652 0.0062 0.0059 0.00537 0.00492 0.00363
## Cumulative Proportion 0.95044 0.95696 0.9632 0.9691 0.97443 0.97935 0.98298
##          PC36     PC37     PC38     PC39     PC40     PC41     PC42
## Standard deviation  0.41758 0.39648 0.34701 0.3421 0.29531 0.27157 0.25973
## Proportion of Variance 0.00268 0.00242 0.00185 0.0018 0.00134 0.00113 0.00104
## Cumulative Proportion 0.98567 0.98809 0.98994 0.9917 0.99308 0.99422 0.99525
##          PC43     PC44     PC45     PC46     PC47     PC48     PC49
## Standard deviation  0.23863 0.22077 0.19892 0.19378 0.17167 0.15858 0.13799
## Proportion of Variance 0.00088 0.00075 0.00061 0.00058 0.00045 0.00039 0.00029
## Cumulative Proportion 0.99613 0.99688 0.99749 0.99807 0.99852 0.99891 0.99920
##          PC50     PC51     PC52     PC53     PC54     PC55     PC56
## Standard deviation  0.11907 0.10507 0.08933 0.08813 0.05879 0.05182 0.03897
## Proportion of Variance 0.00022 0.00017 0.00012 0.00012 0.00005 0.00004 0.00002
## Cumulative Proportion 0.99942 0.99959 0.99971 0.99983 0.99988 0.99992 0.99995
##          PC57     PC58     PC59     PC60     PC61     PC62
## Standard deviation  0.03409 0.03070 0.02611 0.01985 0.01488 0.005099
## Proportion of Variance 0.00002 0.00001 0.00001 0.00001 0.00000 0.000000
## Cumulative Proportion 0.99996 0.99998 0.99999 1.00000 1.00000 1.000000
##          PC63     PC64     PC65
## Standard deviation  0.003586 0.001737 1.408e-05
## Proportion of Variance 0.000000 0.000000 0.000e+00
## Cumulative Proportion 1.000000 1.000000 1.000e+00
```

```
cbind(batters[1:6,1],pca_batters_both$x[1:6,1:6])
```

```
##          Name      PC1      PC2      PC3      PC4      PC5
## 1      A.J. Ellis -0.9649128 0.9403676 1.8469182 -1.312280 -1.9560787
## 2      A.J. Ellis -2.5493732 1.6796143 1.2690923 -2.728097 -1.0313977
```



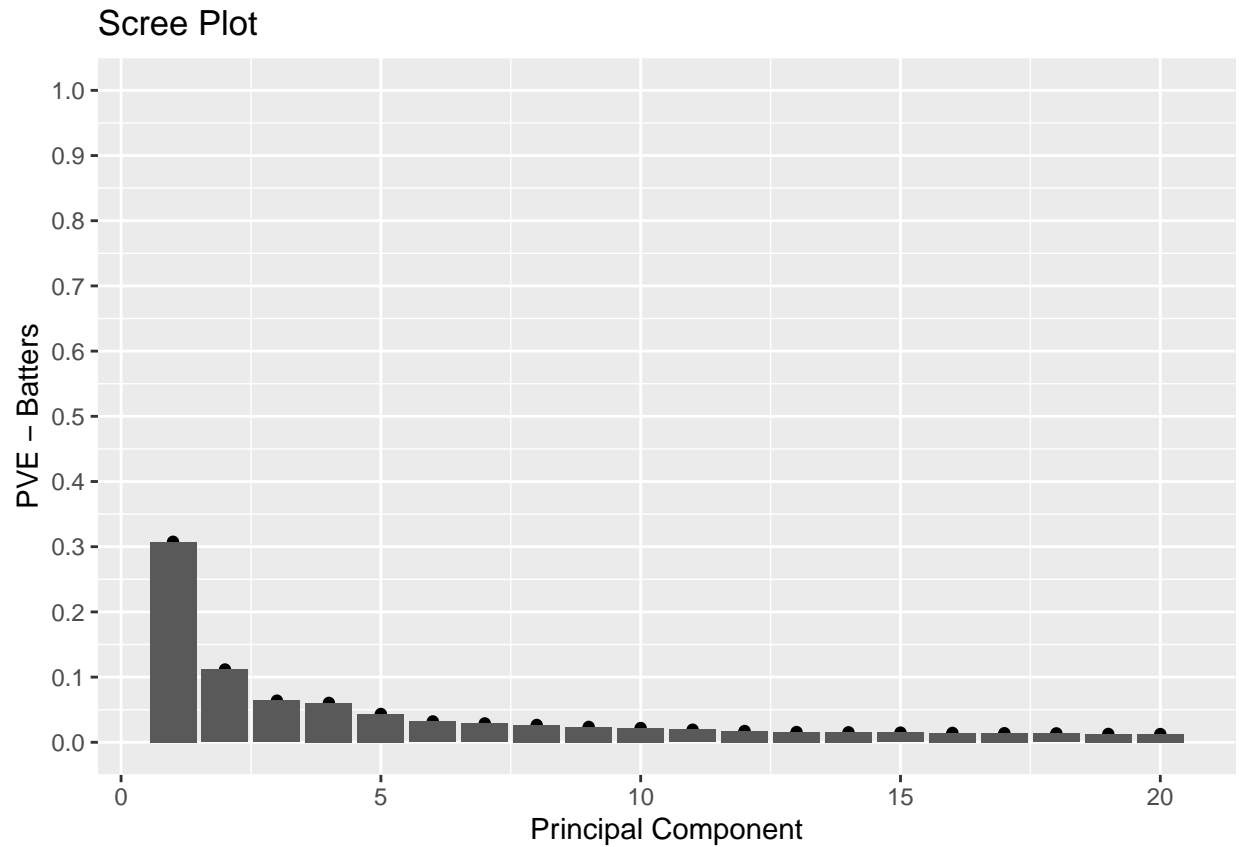
```
## 3      A.J. Ellis -3.6648016 2.2319071 1.0583864 -2.191876 -2.0484016
## 4 A.J. Pierzynski -3.0237490 1.4183354 0.9358940 -2.761832 0.8938653
## 5 A.J. Pierzynski -3.0503865 0.8581999 1.3844129 -3.257367 0.2661191
## 6 A.J. Pierzynski -1.6154950 1.3718762 0.9321839 -2.389565 0.4030617
##      PC6
## 1 0.05887774
## 2 1.05550755
## 3 1.74727064
## 4 -0.26363602
## 5 0.07664955
## 6 -0.07889585
```

```
head(pca_batters_both$scale^2, n = 6)
```

```
##      H_Bat_3yravg BB_Bat_3yravg SO_Bat_3yravg Doubles_3yravg Triples_3yravg
##      1111.443108      333.499422      918.949093      56.514469      3.732683
##      HR_Bat_3yravg
##      68.971448
```

```
PVE = summary(pca_batters_both)$importance[2,]
CVE = summary(pca_batters_both)$importance[3,]

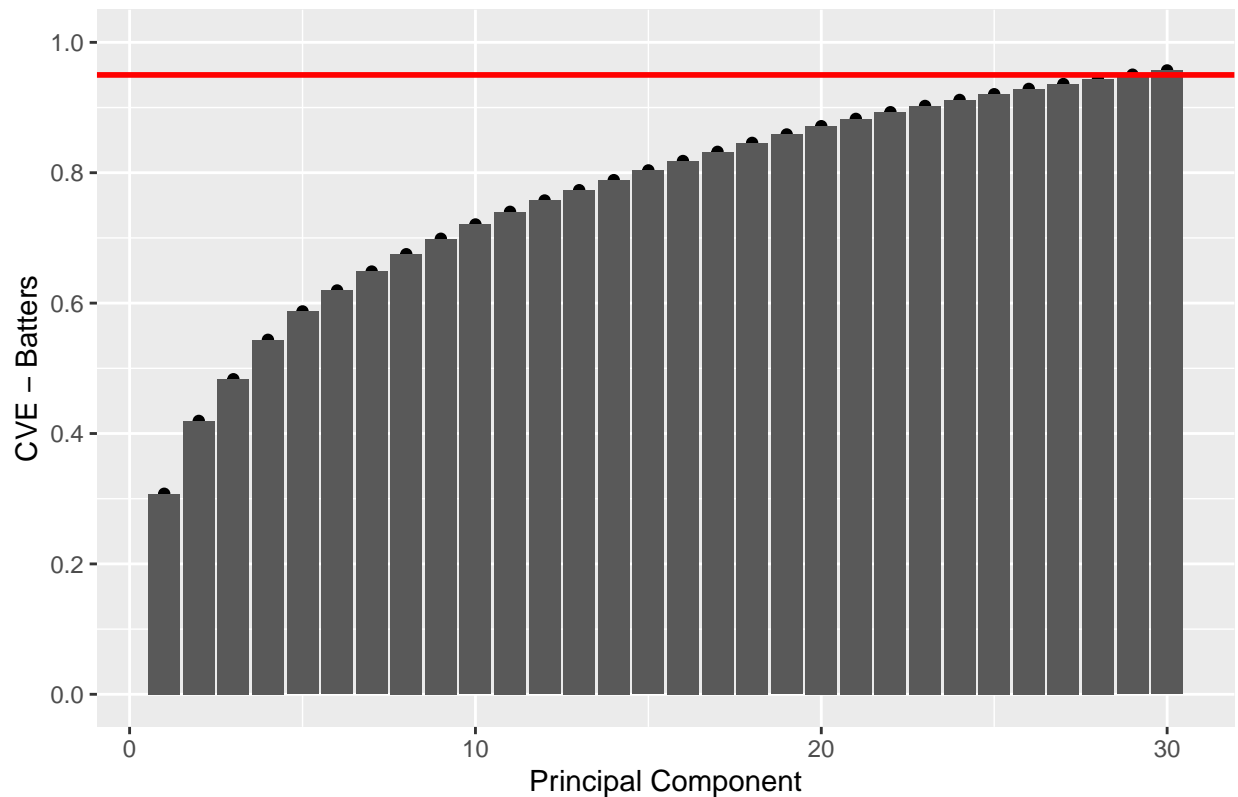
# Graph of variability explained
PVEplot <- qplot(c(1:20), PVE[1:20]) +
  geom_bar(stat = "Identity") +
  xlab("Principal Component") +
  ylab("PVE - Batters") +
  ggtitle("Scree Plot") +
  scale_y_continuous(limits = c(0,1), breaks = seq(0,1,0.1))
PVEplot
```



```
# Flattens out after the 15th PCA component

# Graph of cumulative variability explained
CVEplot <- qplot(c(1:30), CVE[1:30]) +
  geom_bar(stat = "Identity") +
  xlab("Principal Component") +
  ylab("CVE - Batters") +
  ggtitle("Scree Plot") +
  scale_y_continuous(limits = c(0,1), breaks = seq(0,1,0.2)) +
  geom_hline(yintercept = 0.95, color = "red", lwd = 1)
CVEplot
```

## Scree Plot



```
# Reaches 95% cumulative variability after 29th component

# Readding pca and descriptive values back to dataset
batters_both = cbind(batters_both, pca_batters_both$x[,1:29])
batters_both = cbind(batters[,c(1:6,190:193)], batters_both)

# Dividing into test and train
batters_train = batters_both[which(batters_both$playerid %in% batters_piselections),]
batters_test = batters_both[which(batters_both$playerid %!in% batters_piselections),]

# Logistic Regression
set.seed(2020)
batters_glm_both = train(
  form = Top100 ~ PC1 + PC2 + PC3 + PC4 + PC5 + PC6 + PC7 + PC8 + PC9 + PC10 +
    PC11 + PC12 + PC13 + PC14 + PC15 + PC16 + PC17 + PC18 + PC19 + PC20 +
    PC21 + PC22 + PC23 + PC24 + PC25 + PC26 + PC27 + PC28 + PC29,
  data = batters_train,
  trControl = trainControl(method = "cv", number = 5),
  method = "glm",
  family = "binomial"
)

# Model summary
batters_glm_both
```

```
## Generalized Linear Model
```

```
##
## 2075 samples
## 29 predictor
## 2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 1659, 1661, 1659, 1660, 1661
## Resampling results:
##
## Accuracy Kappa
## 0.8973617 0.707637

# Model values
batters_glm_both$finalModel

##
## Call: NULL
##
## Coefficients:
## (Intercept) PC1 PC2 PC3 PC4 PC5
## -2.314084 0.753649 -0.250169 -0.300000 -0.004515 -0.190674
## PC6 PC7 PC8 PC9 PC10 PC11
## -0.091182 -0.012162 0.445395 0.627264 -0.243505 0.203970
## PC12 PC13 PC14 PC15 PC16 PC17
## -0.166898 -0.087139 0.156209 0.066097 0.228376 0.383731
## PC18 PC19 PC20 PC21 PC22 PC23
## -0.078768 0.004037 -0.252510 -0.293881 0.203482 -0.109316
## PC24 PC25 PC26 PC27 PC28 PC29
## 0.179604 0.269187 0.152504 -0.221730 -0.170037 -0.144052
##
## Degrees of Freedom: 2074 Total (i.e. Null); 2045 Residual
## Null Deviance: 2285
## Residual Deviance: 910.1 AIC: 970.1

# Test accuracy
calc_accuracy_function(batters_test$Top100,
  predict(batters_glm_both, newdata = batters_test))

## [1] 0.8849658

# Dataset to compare predictions with actual
batterscomp_both = cbind(batters_train$Name, predict(batters_glm_both,
  newdata = batters_train,
  type = "prob"), batters_train$Top100)

head(batterscomp_both)

## batters_train$Name 0 1 batters_train$Top100
## 1 A.J. Ellis 0.9074176 0.09258239 0
## 2 A.J. Ellis 0.9740629 0.02593714 0
## 3 A.J. Ellis 0.9828705 0.01712955 0
## 4 A.J. Pierzynski 0.9786075 0.02139251 0
## 5 A.J. Pierzynski 0.9819264 0.01807362 0
## 6 A.J. Pierzynski 0.9557364 0.04426357 1

# Confusion Matrix
batters_cm_both = confusionMatrix(predict(batters_glm_both, newdata = batters_test),batters_test$Top100)
```

```
batters_cm_both$table
```

```
##           Reference
## Prediction    0    1
##           0 638  60
##           1  41 139
```

## Performing PCA on Traditional and Advanced, then LR - Pitchers

Variables Included - All traditional and advanced variables from PCA

```
# Combining dataset
pitchers_both = cbind(pitchers_traditional, pitchers_advanced)
# Removing unnecessary variables
pitchers_both = pitchers_both[,-(51:60)]
pitchers_both = pitchers_both[, -c(1:10, 35:50, 77:106)]

# PCA
pca_pitchers_both= prcomp(pitchers_both, scale. = TRUE)
summary(pca_pitchers_both)
```

```
## Importance of components:
##              PC1      PC2      PC3      PC4      PC5      PC6      PC7
## Standard deviation    3.449 3.0177 1.67387 1.55090 1.42606 1.33177 1.27954
## Proportion of Variance 0.238 0.1821 0.05604 0.04811 0.04067 0.03547 0.03274
## Cumulative Proportion 0.238 0.4201 0.47612 0.52423 0.56490 0.60037 0.63312
##              PC8      PC9      PC10     PC11     PC12     PC13     PC14
## Standard deviation    1.19724 1.13627 1.11144 1.05628 1.01704 1.00159 0.98281
## Proportion of Variance 0.02867 0.02582 0.02471 0.02231 0.02069 0.02006 0.01932
## Cumulative Proportion 0.66179 0.68761 0.71231 0.73463 0.75532 0.77538 0.79470
##              PC15     PC16     PC17     PC18     PC19     PC20     PC21
## Standard deviation    0.95765 0.94384 0.89864 0.88489 0.87518 0.85123 0.81223
## Proportion of Variance 0.01834 0.01782 0.01615 0.01566 0.01532 0.01449 0.01319
## Cumulative Proportion 0.81304 0.83086 0.84701 0.86267 0.87799 0.89248 0.90567
##              PC22     PC23     PC24     PC25     PC26     PC27     PC28
## Standard deviation    0.79527 0.78160 0.72295 0.65276 0.60031 0.58680 0.56465
## Proportion of Variance 0.01265 0.01222 0.01045 0.00852 0.00721 0.00689 0.00638
## Cumulative Proportion 0.91832 0.93054 0.94099 0.94952 0.95672 0.96361 0.96999
##              PC29     PC30     PC31     PC32     PC33     PC34     PC35
## Standard deviation    0.51914 0.43301 0.41927 0.40082 0.37033 0.36379 0.30868
## Proportion of Variance 0.00539 0.00375 0.00352 0.00321 0.00274 0.00265 0.00191
## Cumulative Proportion 0.97538 0.97913 0.98264 0.98586 0.98860 0.99124 0.99315
##              PC36     PC37     PC38     PC39     PC40     PC41     PC42
## Standard deviation    0.26868 0.25287 0.22058 0.19044 0.17956 0.16909 0.14406
## Proportion of Variance 0.00144 0.00128 0.00097 0.00073 0.00064 0.00057 0.00042
## Cumulative Proportion 0.99459 0.99587 0.99685 0.99757 0.99822 0.99879 0.99920
##              PC43     PC44     PC45     PC46     PC47     PC48     PC49
## Standard deviation    0.09886 0.09079 0.08608 0.07644 0.05488 0.05224 0.04188
## Proportion of Variance 0.00020 0.00016 0.00015 0.00012 0.00006 0.00005 0.00004
## Cumulative Proportion 0.99940 0.99956 0.99971 0.99983 0.99989 0.99994 0.99998
##              PC50
## Standard deviation    0.03259
## Proportion of Variance 0.00002
## Cumulative Proportion 1.00000
```

```
cbind(pitchers[1:6,1],pca_pitchers_both$x[1:6,1:6])
```

```
##           Name      PC1      PC2      PC3      PC4      PC5      PC6
## 1 A.J. Burnett 1.2778064 -4.637432 -0.6869763 -0.3462696 2.077797 -1.5628780
## 2 A.J. Burnett 3.2350875 -4.018696 -0.4089534 -0.8549426 2.410232 -1.5479390
## 3 A.J. Burnett 2.8455141 -4.046194 -1.4637225 -0.8959427 2.725041 -1.4092599
## 4 A.J. Burnett 0.2328886 -3.932835 -2.2719223 -0.6152020 1.927585 -0.4288666
## 5 A.J. Burnett 0.2974220 -4.234683 -3.1595617 -0.5027763 2.366608 -0.4087474
## 6 A.J. Burnett 0.2418791 -3.513682 -2.2255082 -0.2099395 2.388875 0.2187822
```

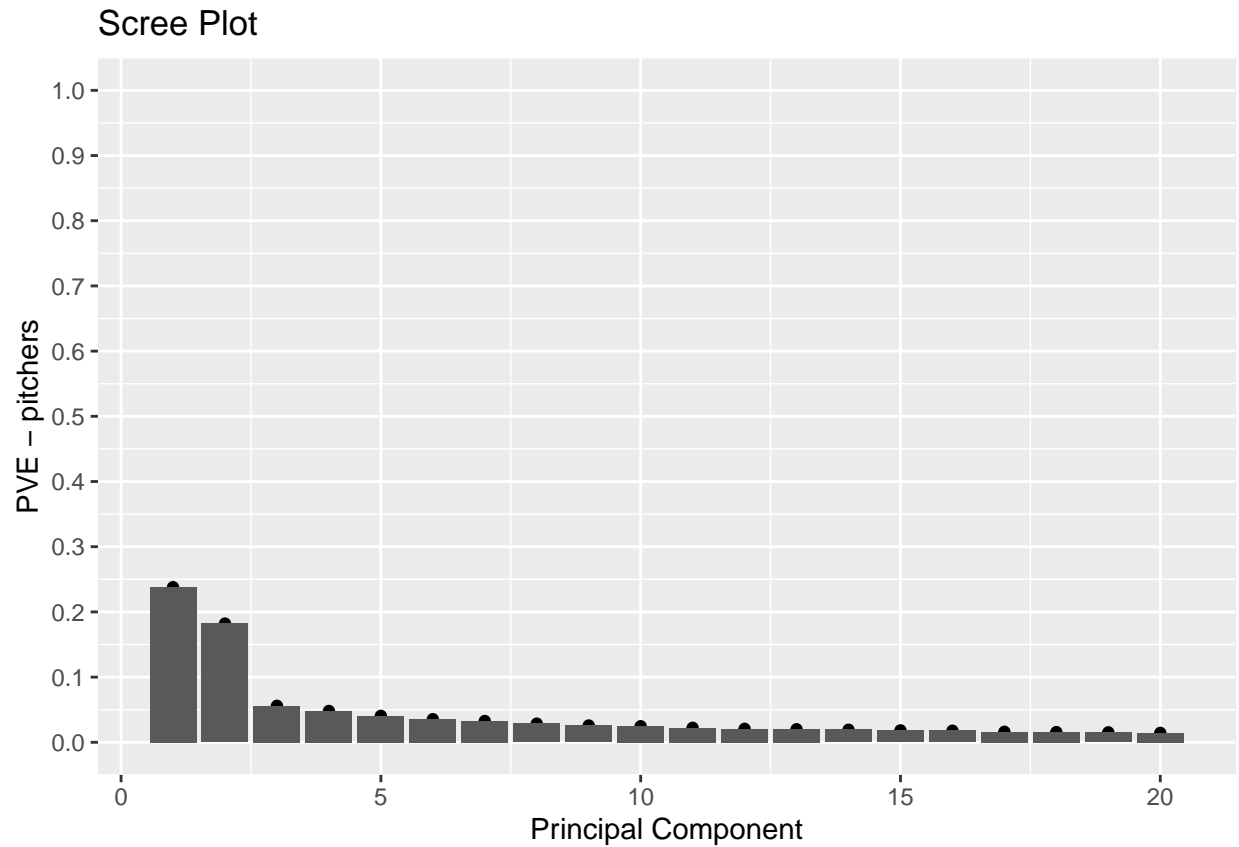
```
head(pca_pitchers_both$scale^2, n = 6)
```

```
##           W_3yavg      L_3yavg      CG_3yavg      SHO_3yavg      SV_3yavg
## 1.371911e+01 8.998276e+00 4.124620e-01 9.723845e-02 5.121191e+01
## H_Pitch_3yavg
## 2.439816e+03
```

```
PVE = summary(pca_pitchers_both)$importance[2,]
CVE = summary(pca_pitchers_both)$importance[3,]
```

```
# Graph of variability explained
```

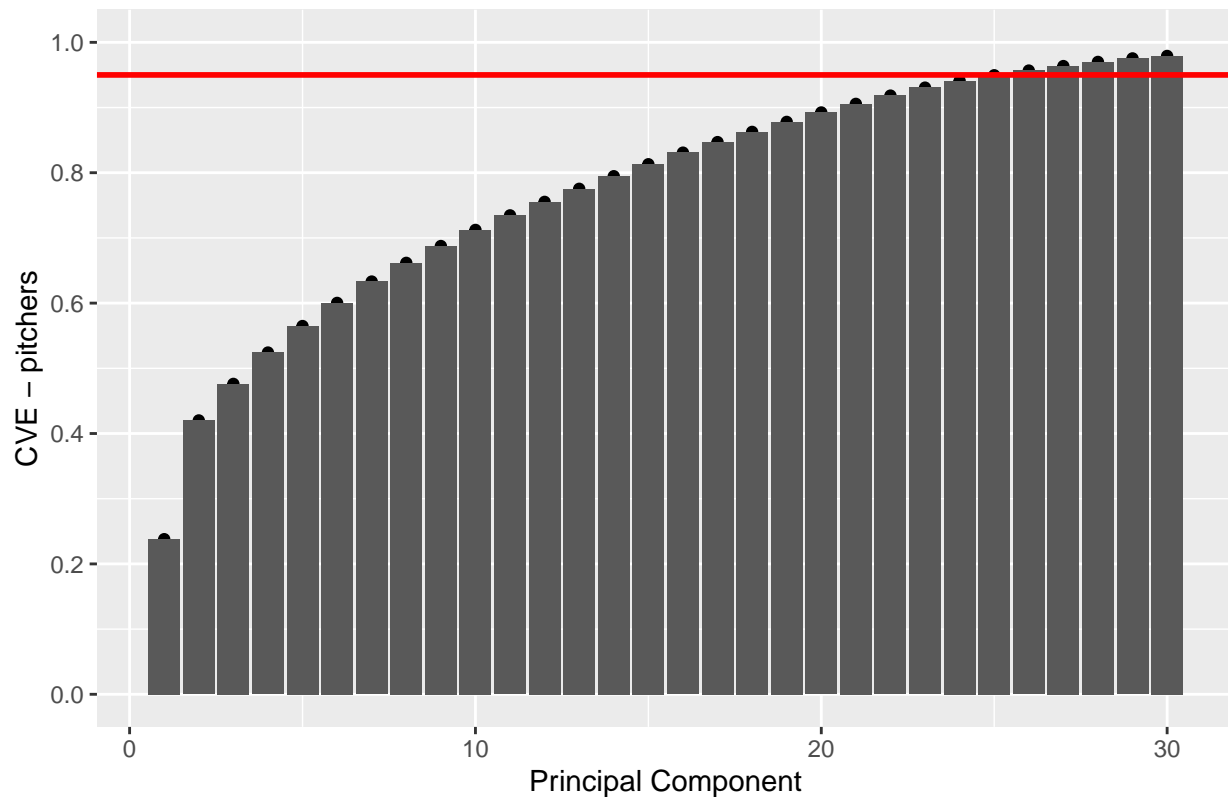
```
PVEplot <- qplot(c(1:20), PVE[1:20]) +
  geom_bar(stat = "Identity") +
  xlab("Principal Component") +
  ylab("PVE - pitchers") +
  ggtitle("Scree Plot") +
  scale_y_continuous(limits = c(0,1), breaks = seq(0,1,0.1))
PVEplot
```



```
# Flattens out after the 15th PCA component

# Graph of cumulative variability explained
CVEplot <- qplot(c(1:30), CVE[1:30]) +
  geom_bar(stat = "Identity") +
  xlab("Principal Component") +
  ylab("CVE - pitchers") +
  ggtitle("Scree Plot") +
  scale_y_continuous(limits = c(0,1), breaks = seq(0,1,0.2)) +
  geom_hline(yintercept = 0.95, color = "red", lwd = 1)
CVEplot
```

### Scree Plot



```
# Reaches 95% cumulative variability after 26th component

# Adding PCA components and descriptive statistics back to dataset
pitchers_both = cbind(pitchers_both, pca_pitchers_both$x[,1:26])
pitchers_both = cbind(pitchers[,c(1:6,163:166)], pitchers_both)

# Dividing into test and train
pitchers_train = pitchers_both[which(pitchers_both$playerid %in% pitchers_piselections),]
pitchers_test = pitchers_both[which(pitchers_both$playerid %!in% pitchers_piselections),]

# Logistic Regression
set.seed(2020)
pitchers_glm_both = train(
  form = Top100 ~ PC1 + PC2 + PC3 + PC4 + PC5 + PC6 + PC7 + PC8 + PC9 + PC10 +
    PC11 + PC12 + PC13 + PC14 + PC15 + PC16 + PC17 + PC18 + PC19 + PC20 + PC21,
  data = pitchers_train,
  trControl = trainControl(method = "cv", number = 5),
  method = "glm",
  family = "binomial"
)

# Model summary
pitchers_glm_both
```

```
## Generalized Linear Model
##
```



```

## 2789 samples
## 21 predictor
## 2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 2231, 2231, 2232, 2231, 2231
## Resampling results:
##
## Accuracy Kappa
## 0.9601977 0.6610727

# Model values
pitchers_glm_both$finalModel

##
## Call: NULL
##
## Coefficients:
## (Intercept) PC1 PC2 PC3 PC4 PC5
## -5.558541 -0.747519 -0.342441 -0.033681 0.211137 0.002337
## PC6 PC7 PC8 PC9 PC10 PC11
## 0.284047 -0.244511 0.893644 -0.603370 -0.623575 0.172424
## PC12 PC13 PC14 PC15 PC16 PC17
## 2.051121 -4.862809 -1.710655 0.586242 0.547346 0.495950
## PC18 PC19 PC20 PC21
## -0.239995 -0.014010 0.276579 0.090890
##
## Degrees of Freedom: 2788 Total (i.e. Null); 2767 Residual
## Null Deviance: 1382
## Residual Deviance: 478.4 AIC: 522.4

# Test accuracy
calc_accuracy_function(pitchers_test$Top100,
  predict(pitchers_glm_both, newdata = pitchers_test))

## [1] 0.9498767

# Dataset to compare predictions with accuracy
pitcherscomp_both = cbind(pitchers_train$Name, predict(pitchers_glm_both,
  newdata = pitchers_train,
  type = "prob"), pitchers_train$Top100)

head(pitcherscomp_both)

## pitchers_train$Name 0 1 pitchers_train$Top100
## 1 A.J. Burnett 0.9825922 0.017407784 0
## 2 A.J. Burnett 0.9965490 0.003450963 0
## 3 A.J. Burnett 0.9936781 0.006321916 0
## 4 A.J. Burnett 0.9671231 0.032876937 0
## 5 A.J. Burnett 0.9765291 0.023470896 0
## 6 A.J. Burnett 0.9621132 0.037886839 0

# Confusion Matrix
pitchers_cm_both = confusionMatrix(predict(pitchers_glm_both, newdata = pitchers_test), pitchers_test$
pitchers_cm_both$table

## Reference

```

```
## Prediction    0    1
##             0 1085   44
##             1   17   71
```

## PCA Results

```
cmlist = list(batters_cm_trad$table, pitchers_cm_trad$table,
             batters_cm_adv$table, pitchers_cm_adv$table,
             batters_cm_trad_adv$table, pitchers_cm_trad_adv$table,
             batters_cm_both$table, pitchers_cm_both$table)

get_confusion_matrix_function = function(cmlist){
  tablelist = list()
  for(i in 1:length(cmlist)){
    tablelist[[i]] = as.data.frame(cmlist[[i]])
    tablelist[[i]] = tablelist[[i]] %>%
      mutate(goodbad = ifelse(tablelist[[i]]$Prediction == tablelist[[i]]$Reference, "correct", "incorrect"))
    group_by(Reference) %>%
    mutate(prop = Freq/sum(Freq))
  }
  tablelist
}

cmlist = get_confusion_matrix_function(cmlist)

batters_cm_trad_plot = ggplot(data = cmlist[[1]], mapping = aes(x = Reference, y = Prediction, fill = goodbad)) +
  geom_tile() +
  geom_text(aes(label = Freq), vjust = .5, fontface = "bold", alpha = 1) +
  scale_fill_manual(values = c(correct = "green", incorrect = "red")) +
  theme_bw() +
  labs(fill = "Quality of Prediction", alpha = "Proportion") +
  ggtitle("CM for Traditional Variables - Batters") +
  theme(plot.title = element_text(hjust = 0.5))

pitchers_cm_trad_plot = ggplot(data = cmlist[[2]], mapping = aes(x = Reference, y = Prediction, fill = goodbad)) +
  geom_tile() +
  geom_text(aes(label = Freq), vjust = .5, fontface = "bold", alpha = 1) +
  scale_fill_manual(values = c(correct = "green", incorrect = "red")) +
  theme_bw() +
  labs(fill = "Quality of Prediction", alpha = "Proportion") +
  ggtitle("CM for Traditional Variables - Pitchers") +
  theme(plot.title = element_text(hjust = 0.5))

batters_cm_adv_plot = ggplot(data = cmlist[[3]], mapping = aes(x = Reference, y = Prediction, fill = goodbad)) +
  geom_tile() +
  geom_text(aes(label = Freq), vjust = .5, fontface = "bold", alpha = 1) +
  scale_fill_manual(values = c(correct = "green", incorrect = "red")) +
  theme_bw() +
  labs(fill = "Quality of Prediction", alpha = "Proportion") +
  ggtitle("CM for Advanced Variables - Batters") +
  theme(plot.title = element_text(hjust = 0.5))

pitchers_cm_adv_plot = ggplot(data = cmlist[[4]], mapping = aes(x = Reference, y = Prediction, fill = goodbad)) +
  geom_tile() +
```

```

geom_text(aes(label = Freq), vjust = .5, fontface = "bold", alpha = 1) +
scale_fill_manual(values = c(correct = "green", incorrect = "red")) +
theme_bw() +
labs(fill = "Quality of Prediction", alpha = "Proportion") +
ggtitle("CM for Advanced Variables - Pitchers") +
theme(plot.title = element_text(hjust = 0.5))

batters_cm_trad_adv_plot = ggplot(data = cmlist[[5]], mapping = aes(x = Reference, y = Prediction, fill =
geom_tile() +
geom_text(aes(label = Freq), vjust = .5, fontface = "bold", alpha = 1) +
scale_fill_manual(values = c(correct = "green", incorrect = "red")) +
theme_bw() +
labs(fill = "Quality of Prediction", alpha = "Proportion") +
ggtitle("CM for Traditional & Advanced PCA - Batters") +
theme(plot.title = element_text(hjust = 0.5))

pitchers_cm_trad_adv_plot = ggplot(data = cmlist[[6]], mapping = aes(x = Reference, y = Prediction, fill =
geom_tile() +
geom_text(aes(label = Freq), vjust = .5, fontface = "bold", alpha = 1) +
scale_fill_manual(values = c(correct = "green", incorrect = "red")) +
theme_bw() +
labs(fill = "Quality of Prediction", alpha = "Proportion") +
ggtitle("CM for Traditional & Advanced PCA - Pitchers") +
theme(plot.title = element_text(hjust = 0.5))

batters_cm_both_plot = ggplot(data = cmlist[[7]], mapping = aes(x = Reference, y = Prediction, fill = g
geom_tile() +
geom_text(aes(label = Freq), vjust = .5, fontface = "bold", alpha = 1) +
scale_fill_manual(values = c(correct = "green", incorrect = "red")) +
theme_bw() +
labs(fill = "Quality of Prediction", alpha = "Proportion") +
ggtitle("CM for Traditional & Advanced Variables - Batters") +
theme(plot.title = element_text(hjust = 0.5))

pitchers_cm_both_plot = ggplot(data = cmlist[[8]], mapping = aes(x = Reference, y = Prediction, fill = g
geom_tile() +
geom_text(aes(label = Freq), vjust = .5, fontface = "bold", alpha = 1) +
scale_fill_manual(values = c(correct = "green", incorrect = "red")) +
theme_bw() +
labs(fill = "Quality of Prediction", alpha = "Proportion") +
ggtitle("CM for Traditional & Advanced Variables - Pitchers") +
theme(plot.title = element_text(hjust = 0.5))

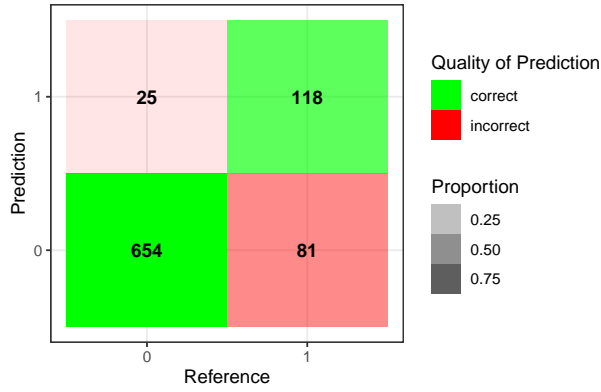
library(cowplot)

## Warning: package 'cowplot' was built under R version 4.0.3

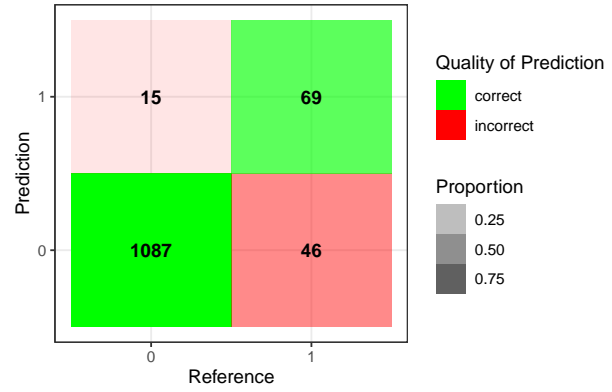
plot_grid(batters_cm_trad_plot,
pitchers_cm_trad_plot,
batters_cm_adv_plot,
pitchers_cm_adv_plot,
batters_cm_trad_adv_plot,
pitchers_cm_trad_adv_plot,
batters_cm_both_plot,
pitchers_cm_both_plot, nrow = 4, ncol = 2)

```

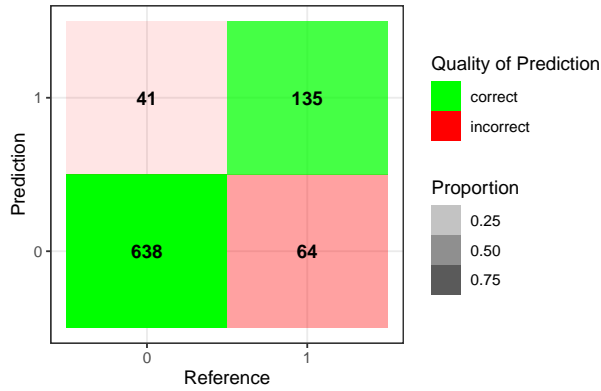
CM for Traditional Variables – Batters



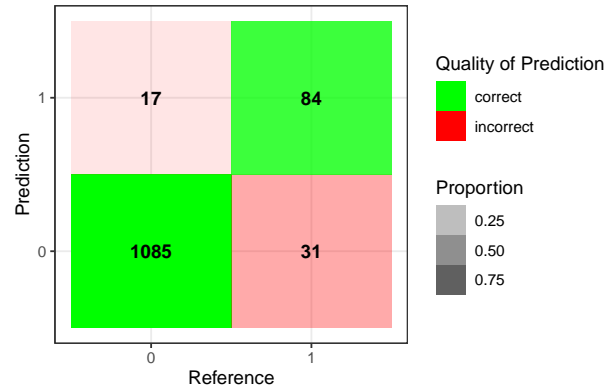
CM for Traditional Variables – Pitchers



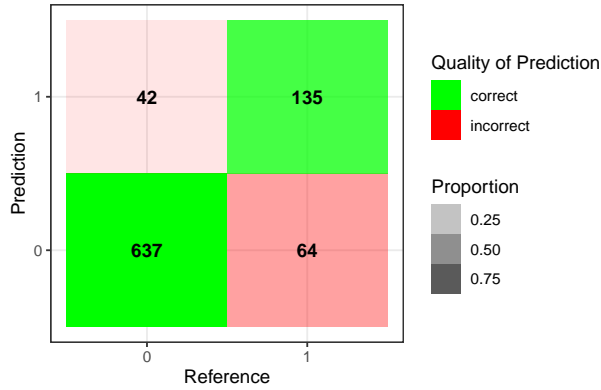
CM for Advanced Variables – Batters



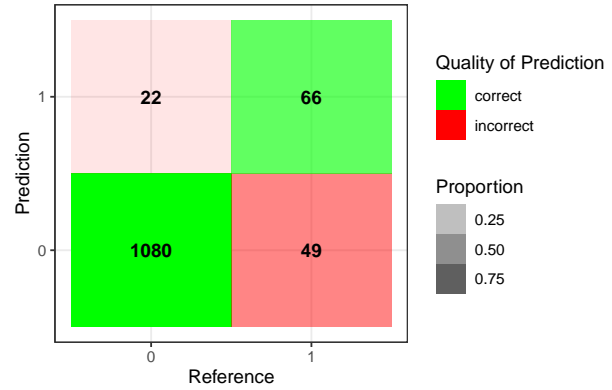
CM for Advanced Variables – Pitchers



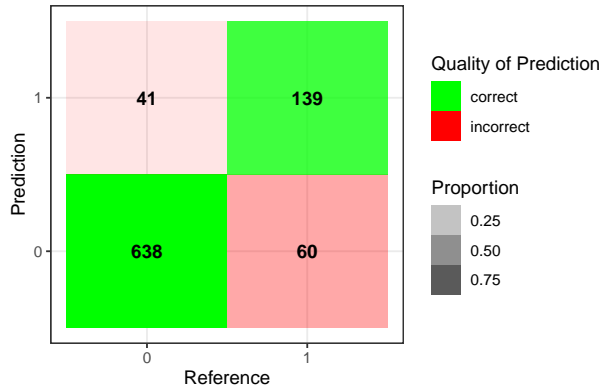
CM for Traditional & Advanced PCA – Batters



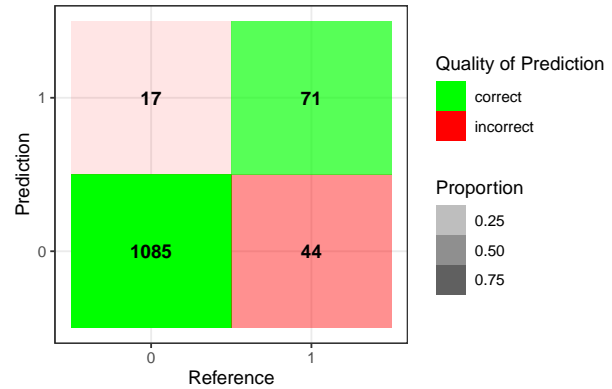
CM for Traditional & Advanced PCA – Pitchers



CM for Traditional & Advanced Variables – Batters



CM for Traditional & Advanced Variables – Pitchers



```

tablelist = list(batters_cm_trad, batters_cm_adv, batters_cm_trad_adv, batters_cm_both,
                 pitchers_cm_trad, pitchers_cm_adv, pitchers_cm_trad_adv, pitchers_cm_both)

cm_accuracy = rep(NA,8)
cm_sensitivity = rep(NA,8)
cm_specificity = rep(NA,8)
cm_pospredvalue = rep(NA,8)
cm_negpredvalue = rep(NA,8)

for(i in 1:8){
  cm_accuracy[i] = round(tablelist[[i]]$overall['Accuracy'],3)
  cm_sensitivity[i] = round(tablelist[[i]]$byClass['Sensitivity'],3)
  cm_specificity[i] = round(tablelist[[i]]$byClass['Specificity'],3)
  cm_pospredvalue[i] = round(tablelist[[i]]$byClass['Pos Pred Value'],3)
  cm_negpredvalue[i] = round(tablelist[[i]]$byClass['Neg Pred Value'],3)
}

cm_names = c("Batters Traditional PCA", "Batters Advanced PCA", "Mix of Batters Trad/Adv PCA",
             "Batters Trad/Adv then PCA", "Pitchers Traditional PCA", "Pitchers Advanced PCA", "Mix of P")

cm_table = as.data.frame(cbind(cm_names, cm_accuracy, cm_sensitivity, cm_specificity, cm_pospredvalue, cm_negpredvalue))
colnames(cm_table) = c("Method", "Accuracy", "Sensitivity", "Specificity", "Positive PV", "Negative PV")
library(kableExtra)

## Warning: package 'kableExtra' was built under R version 4.0.3
##
## Attaching package: 'kableExtra'
## The following object is masked from 'package:dplyr':
##
##      group_rows
kable(cm_table)

```

Method	Accuracy	Sensitivity	Specificity	Positive PV	Negative PV
Batters Traditional PCA	0.879	0.593	0.963	0.825	0.89
Batters Advanced PCA	0.88	0.678	0.94	0.767	0.909
Mix of Batters Trad/Adv PCA	0.879	0.678	0.938	0.763	0.909
Batters Trad/Adv then PCA	0.885	0.698	0.94	0.772	0.914
Pitchers Traditional PCA	0.95	0.6	0.986	0.821	0.959
Pitchers Advanced PCA	0.961	0.73	0.985	0.832	0.972
Mix of Pitchers Trad/Adv PCA	0.942	0.574	0.98	0.75	0.957
Pitchers Trad/Adv then PCA	0.95	0.617	0.985	0.807	0.961

## Penalized Regression Models

```

library(glmnet)
# Selecting variables
batters_traditional = batters_active[,c(4:15,46:47,31,57:62,68:72,75:76)]
batters_traditional = cbind(batters[,c(1:6,191:193,190)],batters_traditional)

# Dividing into test and train
batters_train = batters_traditional[which(batters_traditional$playerid %in% batters_piselections),]
batters_test = batters_traditional[which(batters_traditional$playerid %!in% batters_piselections),]

```

```

# Removing variables not in model
train_dv = batters_train[,c(1:9)]
test_dv = batters_test[,c(1:9)]
batters_train = batters_train[,-c(1:9)]
batters_test = batters_test[,-c(1:9)]

# Penalized logistic regression model
set.seed(2020)
cv_10 = trainControl(method = "cv", number = 10)

batters_traditional_plr_model <- train(form = Top100 ~ ., data = batters_train,
                                     method = "glmnet",
                                     family = "binomial",
                                     trControl = cv_10,
                                     tuneGrid = expand.grid(alpha = 1,
                                                           lambda = seq(0.001,0.1,by = 0.001)))

# Best lambda value
batters_traditional_plr_model$bestTune

##   alpha lambda
## 2      1 0.002

# Coefficients at that lambda value
round(coef(batters_traditional_plr_model$finalModel, batters_traditional_plr_model$bestTune$lambda),3)

## 29 x 1 sparse Matrix of class "dgCMatrix"
##              1
## (Intercept)  -16.837
## H_Bat_3yravg      .
## BB_Bat_3yravg    -0.017
## SO_Bat_3yravg    -0.010
## Doubles_3yravg    0.023
## Triples_3yravg    0.068
## HR_Bat_3yravg     0.155
## TB_3yravg         0.001
## RBI_3yravg        -0.003
## BA_3yravg         .
## OBP_3yravg        30.008
## SLG_3yravg        .
## OPS_3yravg        1.473
## SB_3yravg         0.053
## CS_3yravg         .
## RCPeG_3yravg      0.225
## PO_3yravg         0.001
## A_3yravg          0.006
## E_3yravg          0.002
## DP_3yravg        -0.020
## FldPercent_3yravg .
## RFPe9_3yravg      0.040
## AllStar_total     0.127
## GoldGlove_total   0.061
## SilverSlugger_total 0.121
## MVPRank_3yravg    -0.006

```

```
## MVPVotePoints_3yravg    0.013
## ROYRank_3yravg          0.012
## ROYVotePoints_3yravg    0.020
```

```
# Test data accuracy
calc_accuracy_function(batters_test$Top100,
  predict(batters_traditional_plr_model, newdata = batters_test,
    s = batters_traditional_plr_model$bestTune$lambda))
```

```
## [1] 0.88041
```

```
batters_train = cbind(train_dv[,1:9], batters_train[,1], batters_train[,2:29])
colnames(batters_train)[10] = "Top100"

# Dataset of predictions v actual
batterscomp_plr_traditional = cbind(batters_train$Name, batters_train$Season,
  predict(batters_traditional_plr_model,
    newdata = batters_train,
    s = batters_traditional_plr_model$bestTune$lambda,
    type = "prob"), batters_train$Top100)

head(batterscomp_plr_traditional)
```

```
##   batters_train$Name batters_train$Season      0      1
## 1      A.J. Ellis      2013 0.9019496 0.09805044
## 2      A.J. Ellis      2014 0.9574791 0.04252091
## 3      A.J. Ellis      2015 0.9712042 0.02879580
## 4    A.J. Pierzynski      2010 0.8926171 0.10738286
## 5    A.J. Pierzynski      2011 0.8991142 0.10088584
## 6    A.J. Pierzynski      2012 0.8083628 0.19163722
##   batters_train$Top100
## 1                    0
## 2                    0
## 3                    0
## 4                    0
## 5                    0
## 6                    1
```

```
# Confusion matrix
batters_cm_plr_trad = confusionMatrix(predict(batters_traditional_plr_model,
  newdata = batters_test,
  s = batters_traditional_plr_model$bestTune$lambda),
  batters_test$Top100, positive='1')

batters_cm_plr_trad$table
```

```
##           Reference
## Prediction  0    1
##           0 652  78
##           1  27 121
```

```
library(glmnet)
```

```
# Selecting variables
```

```
pitchers_traditional = pitchers_active[,c(1:2,5:7,13:18,20:23,50:54,57:58,61:62)]
pitchers_traditional = cbind(pitchers[,c(1:6,164:166,163)], pitchers_traditional)
```

```
# Dividing into test and train
```

```
pitchers_train = pitchers_traditional[which(pitchers_traditional$playerid %in% pitchers_piselections)]
pitchers_test = pitchers_traditional[which(pitchers_traditional$playerid %!in% pitchers_piselections)]
```

```

# Removing variables not in model
train_dv = pitchers_train[,c(1:9)]
test_dv = pitchers_test[,c(1:9)]
pitchers_train = pitchers_train[,-c(1:9)]
pitchers_test = pitchers_test[,-c(1:9)]

# Penalized logistic regression model
set.seed(2020)
cv_10 = trainControl(method = "cv", number = 10)

pitchers_traditional_plr_model <- train(form = Top100 ~ ., data = pitchers_train,
                                       method = "glmnet",
                                       family = "binomial",
                                       trControl = cv_10,
                                       tuneGrid = expand.grid(alpha = 1,
                                                             lambda = seq(0.001,0.1,by = 0.001)))

# Best lambda value
pitchers_traditional_plr_model$bestTune

##   alpha lambda
## 2      1 0.002

# Coefficients at that lambda value
round(coef(pitchers_traditional_plr_model$finalModel, pitchers_traditional_plr_model$bestTune$lambda))

## 25 x 1 sparse Matrix of class "dgCMatrix"
##              1
## (Intercept)  -0.389
## W_3yravg      0.192
## L_3yravg      -0.003
## CG_3yravg     0.321
## SH0_3yravg    0.246
## SV_3yravg     0.015
## H_Pitch_3yravg .
## R_Pitch_3yravg .
## ER_3yravg     .
## ERA_3yravg     .
## WHIP_3yravg   -3.441
## RSPer9_3yravg .
## HR9_3yravg    -1.723
## BB9_3yravg     .
## S09_3yravg     0.366
## RA9_3yravg    -0.368
## AllStar_total  0.152
## GoldGlove_total -1.486
## SilverSlugger_total .
## MVPRank_3yravg -1.007
## MVPVotePoints_3yravg .
## CYRank_3yravg  0.233
## CYVotePoints_3yravg 0.017
## ROYRank_3yravg  0.192
## ROYVotePoints_3yravg 0.007

```



```

# Test data accuracy
calc_accuracy_function(pitchers_test$Top100,
  predict(pitchers_traditional_plr_model, newdata = pitchers_test,
    s = pitchers_traditional_plr_model$bestTune$lambda))

## [1] 0.94659

pitchers_train = cbind(train_dv[,1:9], pitchers_train[,1], pitchers_train[,2:25])
colnames(pitchers_train)[10] = "Top100"

# Dataset of predictions v actual
pitcherscomp_plr_traditional = cbind(pitchers_train$Name, pitchers_train$Season,
  predict(pitchers_traditional_plr_model,
    newdata = pitchers_train,
    s = pitchers_traditional_plr_model$bestTune$lambda,
    type = "prob"), pitchers_train$Top100)

head(pitcherscomp_plr_traditional)

##   pitchers_train$Name pitchers_train$Season      0      1
## 1      A.J. Burnett      2010 0.9797638 0.02023617
## 2      A.J. Burnett      2011 0.9945918 0.00540825
## 3      A.J. Burnett      2012 0.9899217 0.01007830
## 4      A.J. Burnett      2013 0.9630357 0.03696435
## 5      A.J. Burnett      2014 0.9481550 0.05184496
## 6      A.J. Burnett      2015 0.9675416 0.03245842
##   pitchers_train$Top100
## 1                      0
## 2                      0
## 3                      0
## 4                      0
## 5                      0
## 6                      0

# Confusion matrix
pitchers_cm_plr_trad = confusionMatrix(predict(pitchers_traditional_plr_model,
  newdata = pitchers_test,
  s = pitchers_traditional_plr_model$bestTune$lambda),
  pitchers_test$Top100, positive='1')

pitchers_cm_plr_trad$table

##           Reference
## Prediction    0    1
##           0 1085   48
##           1   17   67

# Selecting variables
batters_advanced = batters_active[,c(16:29,32:45,48:51,63:72,75:76)]
batters_advanced = cbind(batters[,c(1:6,191:193,190)],batters_advanced)

# Dividing into test and train
batters_train = batters_advanced[which(batters_advanced$playerid %in% batters_piselections),]
batters_test = batters_advanced[which(batters_advanced$playerid %!in% batters_piselections),]

# Removing variables not in model
train_dv = batters_train[,c(1:9)]
test_dv = batters_test[,c(1:9)]

```

```

batters_train = batters_train[,-c(1:9)]
batters_test = batters_test[,-c(1:9)]

# Penalized logistic regression model
set.seed(2020)
cv_10 = trainControl(method = "cv", number = 10)

batters_advanced_plr_model <- train(form = Top100 ~ ., data = batters_train,
                                   method = "glmnet",
                                   family = "binomial",
                                   trControl = cv_10,
                                   tuneGrid = expand.grid(alpha = 1,
                                                         lambda = seq(0.001,0.1,by = 0.001)))

# Best lambda value
batters_advanced_plr_model$bestTune

##      alpha lambda
## 6         1 0.006

# Coefficients at that lambda value
round(coef(batters_advanced_plr_model$finalModel, batters_advanced_plr_model$bestTune$lambda),3)

## 45 x 1 sparse Matrix of class "dgCMatrix"
##              1
## (Intercept)    0.669
## BAbip_3yravg    .
## LDPercent_Bat_3yravg -0.056
## GBPercent_Bat_3yravg  0.018
## FBPercent_Bat_3yravg  .
## HRPerFB_3yravg    0.021
## PullPercent_Bat_3yravg .
## CentPercent_Bat_3yravg .
## OppoPercent_Bat_3yravg .
## SoftPercent_Bat_3yravg -0.002
## MedPercent_Bat_3yravg -0.021
## HardPercent_Bat_3yravg 0.010
## BBPerK_3yravg    .
## wOBA_3yravg      .
## OPSPlus_3yravg   .
## wRCPlus_3yravg   .
## BattingWins_BR_3yravg .
## WAA_Bat_3yravg    0.533
## WAR_Pos_BR_3yravg .
## oWAR_BR_3yravg    0.621
## WAR_Pos_FG_3yravg 0.111
## REW_Bat_3yravg    .
## WPA_Bat_3yravg    .
## pLI_Bat_3yravg    -4.048
## WPAPerLI_Bat_3yravg .
## Clutch_Bat_3yravg .
## OWinPercent_3yravg .
## BattingWins_FG_3yravg 0.940
## oWAR_FG_3yravg    .

```

```
## UBR_3yravg          -0.015
## wGDP_3yravg         -0.020
## wSB_3yravg          0.107
## BsR_3yravg          .
## DRS_BR_3yravg       .
## Rgood_3yravg        .
## dWAR_BR_3yravg      .
## DRS_FG_3yravg       .
## dWAR_FG_3yravg      0.141
## AllStar_total       0.013
## GoldGlove_total     0.146
## SilverSlugger_total 0.315
## MVPRank_3yravg      .
## MVPVotePoints_3yravg 0.005
## ROYRank_3yravg      .
## ROYVotePoints_3yravg 0.006
```

```
# Test data accuracy
calc_accuracy_function(batters_test$Top100,
  predict(batters_advanced_plr_model, newdata = batters_test,
    s = batters_advanced_plr_model$bestTune$lambda))
```

```
## [1] 0.8838269
```

```
batters_train = cbind(train_dv[,1:9], batters_train[,1], batters_train[,2:45])
colnames(batters_train)[10] = "Top100"

# Dataset of predictions v actual
batterscomp_plr_advanced = cbind(batters_train$Name, batters_train$Season,
  predict(batters_advanced_plr_model,
    newdata = batters_train,
    s = batters_advanced_plr_model$bestTune$lambda,
    type = "prob"), batters_train$Top100)

head(batterscomp_plr_advanced)
```

```
##   batters_train$Name batters_train$Season      0      1
## 1      A.J. Ellis      2013 0.8735693 0.12643071
## 2      A.J. Ellis      2014 0.9451556 0.05484439
## 3      A.J. Ellis      2015 0.9657759 0.03422411
## 4      A.J. Pierzynski      2010 0.9875076 0.01249242
## 5      A.J. Pierzynski      2011 0.9869686 0.01303135
## 6      A.J. Pierzynski      2012 0.9580321 0.04196788
##   batters_train$Top100
## 1                    0
## 2                    0
## 3                    0
## 4                    0
## 5                    0
## 6                    1
```

```
# Confusion matrix
batters_cm_plr_adv = confusionMatrix(predict(batters_advanced_plr_model,
  newdata = batters_test,
  s = batters_advanced_plr_model$bestTune$lambda),
  batters_test$Top100, positive='1')

batters_cm_plr_adv$table
```

```
##           Reference
## Prediction    0    1
##           0 638  61
##           1  41 138
```

```
library(glmnet)
# Selecting variables
pitchers_advanced = pitchers_active[,c(24:49,50:54,57:58,61:62)]
pitchers_advanced = cbind(pitchers[,c(1:6,164:166,163)], pitchers_advanced)

# Dividing into test and train
pitchers_train = pitchers_advanced[which(pitchers_advanced$playerid %in% pitchers_piselections),]
pitchers_test = pitchers_advanced[which(pitchers_advanced$playerid %!in% pitchers_piselections),]

# Removing variables not in model
train_dv = pitchers_train[,c(1:9)]
test_dv = pitchers_test[,c(1:9)]
pitchers_train = pitchers_train[,-c(1:9)]
pitchers_test = pitchers_test[,-c(1:9)]

# Penalized logistic regression model
set.seed(2020)
cv_10 = trainControl(method = "cv", number = 10)

pitchers_advanced_plr_model <- train(form = Top100 ~ ., data = pitchers_train,
                                   method = "glmnet",
                                   family = "binomial",
                                   trControl = cv_10,
                                   tuneGrid = expand.grid(alpha = 1,
                                                         lambda = seq(0.001,0.1,by = 0.001)))

# Best lambda value
pitchers_advanced_plr_model$bestTune

##   alpha lambda
## 2      1 0.002

# Coefficients at that lambda value
round(coef(pitchers_advanced_plr_model$finalModel, pitchers_advanced_plr_model$bestTune$lambda),3)

## 36 x 1 sparse Matrix of class "dgCMatrix"
##              1
## (Intercept)    3.614
## LOBPercent_3yravg      .
## LDPercent_Pitch_3yravg -0.058
## GBPercent_Pitch_3yravg      .
## FBPercent_Pitch_3yravg      .
## SoftPercent_Pitch_3yravg     .
## MedPercent_Pitch_3yravg     .
## HardPercent_Pitch_3yravg     .
## BABIP_3yravg           .
## ERAPlus_3yravg          .
## FIP_3yravg             -0.304
## xFIP_3yravg            -1.393
## SIERA_3yravg            .
```

```
## WAR_Pitch_FG_3yravg      0.101
## gmLI_3yravg              0.077
## WAA_Pitch_3yravg         .
## WAAadj_3yravg            0.879
## waaWLPercent_3yravg     .
## FullSeasonWLPercent_3yravg .
## WAR_Pitch_BR_3yravg     .
## WPA_Pitch_3yravg        0.787
## REW_Pitch_3yravg        0.586
## pLI_Pitch_3yravg        .
## WPAPerLI_Pitch_3yravg   .
## Clutch_Pitch_3yravg     0.104
## SD_3yravg               .
## MD_3yravg               -0.231
## AllStar_total           0.116
## GoldGlove_total         -0.001
## SilverSlugger_total     0.004
## MVPRank_3yravg          -0.433
## MVPVotePoints_3yravg    .
## CYRank_3yravg           0.155
## CYVotePoints_3yravg     0.014
## ROYRank_3yravg          0.126
## ROYVotePoints_3yravg    0.007
```

```
# Test data accuracy
```

```
calc_accuracy_function(pitchers_test$Top100,
                        predict(pitchers_advanced_plr_model, newdata = pitchers_test,
                                s = pitchers_advanced_plr_model$bestTune$lambda))
```

```
## [1] 0.9539852
```

```
pitchers_train = cbind(train_dv[,1:9], pitchers_train[,1], pitchers_train[,2:36])
colnames(pitchers_train)[10] = "Top100"
```

```
# Dataset of predictions v actual
```

```
pitcherscomp_plr_advanced = cbind(pitchers_train$Name, pitchers_train$Season,
                                   predict(pitchers_advanced_plr_model,
                                             newdata = pitchers_train,
                                             s = pitchers_advanced_plr_model$bestTune$lambda,
                                             type = "prob"), pitchers_train$Top100)

head(pitcherscomp_plr_advanced)
```

```
##   pitchers_train$Name pitchers_train$Season      0      1
## 1      A.J. Burnett      2010 0.9905722 0.009427836
## 2      A.J. Burnett      2011 0.9982258 0.001774177
## 3      A.J. Burnett      2012 0.9972094 0.002790577
## 4      A.J. Burnett      2013 0.9790621 0.020937943
## 5      A.J. Burnett      2014 0.9877541 0.012245892
## 6      A.J. Burnett      2015 0.9812868 0.018713213
##   pitchers_train$Top100
## 1                      0
## 2                      0
## 3                      0
## 4                      0
## 5                      0
```

```
## 6                                0

# Confusion matrix
pitchers_cm_plr_adv = confusionMatrix(predict(pitchers_advanced_plr_model,
                                              newdata = pitchers_test,
                                              s = pitchers_advanced_plr_model$bestTune$lambda),
                                      pitchers_test$Top100, positive='1')

pitchers_cm_plr_adv$table

##           Reference
## Prediction    0    1
##           0 1081   35
##           1   21   80

# Merging datasets
batters_both = cbind(batters_traditional, batters_advanced)
# Removing name, descriptive values and duplicates
batters_both = batters_both[,-c(39:48,86:92)]

# Dividing into test and train
batters_train = batters_both[which(batters_both$playerid %in% batters_piselections),]
batters_test = batters_both[which(batters_both$playerid %!in% batters_piselections),]

# Removing variables not in model
train_dv = batters_train[,c(1:9)]
test_dv = batters_test[,c(1:9)]
batters_train = batters_train[,-c(1:9)]
batters_test = batters_test[,-c(1:9)]

# Penalized logistic regression model
set.seed(2020)
cv_10 = trainControl(method = "cv", number = 10)

batters_both_plr_model <- train(form = Top100 ~ ., data = batters_train,
                               method = "glmnet",
                               family = "binomial",
                               trControl = cv_10,
                               tuneGrid = expand.grid(alpha = 1,
                                                       lambda = seq(0.001,0.1,by = 0.001)))

# Best lambda value
batters_both_plr_model$bestTune

##   alpha lambda
## 8      1 0.008

# Coefficients at that lambda value
round(coef(batters_both_plr_model$finalModel, batters_both_plr_model$bestTune$lambda),3)

## 66 x 1 sparse Matrix of class "dgCMatrix"
##           1
## (Intercept)      0.265
## H_Bat_3yravg      .
## BB_Bat_3yravg    -0.009
## SO_Bat_3yravg      0.000
## Doubles_3yravg      .
```

## Triples_3yravg	.
## HR_Bat_3yravg	0.002
## TB_3yravg	.
## RBI_3yravg	.
## BA_3yravg	.
## OBP_3yravg	.
## SLG_3yravg	.
## OPS_3yravg	.
## SB_3yravg	.
## CS_3yravg	.
## RCPeG_3yravg	0.018
## PO_3yravg	.
## A_3yravg	.
## E_3yravg	0.024
## DP_3yravg	.
## FldPercent_3yravg	.
## RFPer9_3yravg	.
## AllStar_total	0.026
## GoldGlove_total	0.137
## SilverSlugger_total	0.268
## MVPRank_3yravg	.
## MVPVotePoints_3yravg	0.004
## ROYRank_3yravg	.
## ROYVotePoints_3yravg	0.005
## BABip_3yravg	.
## LDPercent_Bat_3yravg	-0.046
## GBPercent_Bat_3yravg	0.013
## FBPercent_Bat_3yravg	.
## HRPerFB_3yravg	0.017
## PullPercent_Bat_3yravg	.
## CentPercent_Bat_3yravg	.
## OppoPercent_Bat_3yravg	.
## SoftPercent_Bat_3yravg	-0.001
## MedPercent_Bat_3yravg	-0.016
## HardPercent_Bat_3yravg	0.007
## BBPerK_3yravg	.
## wOBA_3yravg	.
## OPSPlus_3yravg	.
## wRCPlus_3yravg	.
## BattingWins_BR_3yravg	.
## WAA_Bat_3yravg	0.489
## WAR_Pos_BR_3yravg	.
## oWAR_BR_3yravg	0.478
## WAR_Pos_FG_3yravg	0.276
## REW_Bat_3yravg	.
## WPA_Bat_3yravg	.
## pLI_Bat_3yravg	-3.634
## WPAPerLI_Bat_3yravg	.
## Clutch_Bat_3yravg	.
## OwnPercent_3yravg	.
## BattingWins_FG_3yravg	0.964
## oWAR_FG_3yravg	.
## UBR_3yravg	-0.005
## wGDP_3yravg	.

```
## wSB_3yavg          0.068
## BsR_3yavg          .
## DRS_BR_3yavg       .
## Rgood_3yavg        .
## dWAR_BR_3yavg      .
## DRS_FG_3yavg       .
## dWAR_FG_3yavg      .
```

```
# Test data accuracy
```

```
calc_accuracy_function(batters_test$Top100,
  predict(batters_both_plr_model, newdata = batters_test,
    s = batters_both_plr_model$bestTune$lambda))
```

```
## [1] 0.8826879
```

```
batters_train = cbind(train_dv[,1:9], batters_train[,1], batters_train[,2:66])
colnames(batters_train)[10] = "Top100"
```

```
# Dataset of predictions v actual
```

```
batterscomp_plr_both = cbind(batters_train$Name, batters_train$Season,
  predict(batters_both_plr_model,
    newdata = batters_train,
    s = batters_both_plr_model$bestTune$lambda,
    type = "prob"), batters_train$Top100)

head(batterscomp_plr_both)
```

```
##   batters_train$Name batters_train$Season      0      1
## 1      A.J. Ellis      2013 0.8905597 0.10944032
## 2      A.J. Ellis      2014 0.9600852 0.03991476
## 3      A.J. Ellis      2015 0.9751091 0.02489095
## 4  A.J. Pierzynski      2010 0.9865358 0.01346417
## 5  A.J. Pierzynski      2011 0.9859001 0.01409990
## 6  A.J. Pierzynski      2012 0.9538396 0.04616040
##   batters_train$Top100
## 1                    0
## 2                    0
## 3                    0
## 4                    0
## 5                    0
## 6                    1
```

```
# Confusion matrix
```

```
batters_cm_plr_both = confusionMatrix(predict(batters_both_plr_model,
  newdata = batters_test,
  s = batters_both_plr_model$bestTune$lambda),
  batters_test$Top100, positive='1')

batters_cm_plr_both$table
```

```
##           Reference
## Prediction  0  1
##           0 637  61
##           1  42 138
```

```
# Combining dataset
```

```
pitchers_both = cbind(pitchers_traditional, pitchers_advanced)
# Removing unnecessary variables
pitchers_both = pitchers_both[, -c(35:44, 71:79)]
```



```

# Dividing into test and train
pitchers_train = pitchers_both[which(pitchers_both$playerid %in% pitchers_piselections),]
pitchers_test = pitchers_both[which(pitchers_both$playerid %!in% pitchers_piselections),]

# Removing variables not in model
train_dv = pitchers_train[,c(1:9)]
test_dv = pitchers_test[,c(1:9)]
pitchers_train = pitchers_train[,-c(1:9)]
pitchers_test = pitchers_test[,-c(1:9)]

# Penalized logistic regression model
set.seed(2020)
cv_10 = trainControl(method = "cv", number = 10)

pitchers_both_plr_model <- train(form = Top100 ~ ., data = pitchers_train,
                                method = "glmnet",
                                family = "binomial",
                                trControl = cv_10,
                                tuneGrid = expand.grid(alpha = 1,
                                                         lambda = seq(0.001,0.1,by = 0.001)))

# Best lambda value
pitchers_both_plr_model$bestTune

##   alpha lambda
## 1      1 0.001

# Coefficients at that lambda value
round(coef(pitchers_both_plr_model$finalModel, pitchers_both_plr_model$bestTune$lambda),3)

## 51 x 1 sparse Matrix of class "dgCMatrix"
##              1
## (Intercept)    3.491
## W_3yravg       0.030
## L_3yravg       .
## CG_3yravg      0.033
## SHO_3yravg     0.353
## SV_3yravg      0.064
## H_Pitch_3yravg .
## R_Pitch_3yravg .
## ER_3yravg      .
## ERA_3yravg     .
## WHIP_3yravg    .
## RSPer9_3yravg  .
## HR9_3yravg     -0.672
## BB9_3yravg     0.440
## SO9_3yravg     0.115
## RA9_3yravg     .
## AllStar_total  .
## GoldGlove_total -0.021
## SilverSlugger_total .
## MVPRank_3yravg -0.412
## MVPVotePoints_3yravg -0.010
## CYRank_3yravg  0.143

```

```
## CYVotePoints_3yravg      0.015
## ROYRank_3yravg           0.145
## ROYVotePoints_3yravg     0.007
## LOBPercent_3yravg        -0.024
## LDPercent_Pitch_3yravg   -0.047
## GBPercent_Pitch_3yravg    .
## FBPercent_Pitch_3yravg    .
## SoftPercent_Pitch_3yravg  .
## MedPercent_Pitch_3yravg   .
## HardPercent_Pitch_3yravg  .
## BABIP_3yravg             .
## ERAPlus_3yravg           .
## FIP_3yravg               .
## xFIP_3yravg              -1.640
## SIERA_3yravg             -0.120
## WAR_Pitch_FG_3yravg       0.003
## gmLI_3yravg              0.422
## WAA_Pitch_3yravg          .
## WAAadj_3yravg            1.586
## waaWLPercent_3yravg       .
## FullSeasonWLPercent_3yravg .
## WAR_Pitch_BR_3yravg       .
## WPA_Pitch_3yravg          0.558
## REW_Pitch_3yravg          1.048
## pLI_Pitch_3yravg          .
## WPAperLI_Pitch_3yravg     .
## Clutch_Pitch_3yravg       0.285
## SD_3yravg                -0.058
## MD_3yravg                -0.299
```

```
# Test data accuracy
```

```
calc_accuracy_function(pitchers_test$Top100,
                        predict(pitchers_both_plr_model, newdata = pitchers_test,
                                s = pitchers_both_plr_model$bestTune$lambda))
```

```
## [1] 0.9506984
```

```
pitchers_train = cbind(train_dv[,1:9], pitchers_train[,1], pitchers_train[,2:51])
colnames(pitchers_train)[10] = "Top100"
```

```
# Dataset of predictions v actual
```

```
pitcherscomp_plr_both = cbind(pitchers_train$Name, pitchers_train$Season,
                              predict(pitchers_both_plr_model,
                                      newdata = pitchers_train,
                                      s = pitchers_both_plr_model$bestTune$lambda,
                                      type = "prob"), pitchers_train$Top100)

head(pitcherscomp_plr_both)
```

```
##   pitchers_train$Name pitchers_train$Season      0      1
## 1      A.J. Burnett      2010 0.9839259 0.016074115
## 2      A.J. Burnett      2011 0.9980669 0.001933126
## 3      A.J. Burnett      2012 0.9972141 0.002785873
## 4      A.J. Burnett      2013 0.9755006 0.024499404
## 5      A.J. Burnett      2014 0.9872433 0.012756732
## 6      A.J. Burnett      2015 0.9832952 0.016704791
```

```
## pitchers_train$Top100
## 1 0
## 2 0
## 3 0
## 4 0
## 5 0
## 6 0

# Confusion matrix
pitchers_cm_plr_both = confusionMatrix(predict(pitchers_both_plr_model,
                                              newdata = pitchers_test,
                                              s = pitchers_both_plr_model$bestTune$lambda),
                                      pitchers_test$Top100, positive='1')

pitchers_cm_plr_both$table

##           Reference
## Prediction    0    1
##           0 1080   38
##           1   22   77
```

## PLR Results

```
cmlist = list(batters_cm_plr_trad$table, pitchers_cm_plr_trad$table,
              batters_cm_plr_adv$table, pitchers_cm_plr_adv$table,
              batters_cm_plr_both$table, pitchers_cm_plr_both$table)

get_confusion_matrix_function = function(cmlist){
  tablelist = list()
  for(i in 1:length(cmlist)){
    tablelist[[i]] = as.data.frame(cmlist[[i]])
    tablelist[[i]] = tablelist[[i]] %>%
    mutate(goodbad = ifelse(tablelist[[i]]$Prediction == tablelist[[i]]$Reference, "correct", "incorrect"),
           group_by(Reference) %>%
           mutate(prop = Freq/sum(Freq))
  }
  tablelist
}

cmlist = get_confusion_matrix_function(cmlist)

batters_cm_plr_trad_plot = ggplot(data = cmlist[[1]], mapping = aes(x = Reference, y = Prediction, fill =
  geom_tile() +
  geom_text(aes(label = Freq), vjust = .5, fontface = "bold", alpha = 1) +
  scale_fill_manual(values = c(correct = "green", incorrect = "red")) +
  theme_bw() +
  labs(fill = "Quality of Prediction", alpha = "Proportion") +
  ggtitle("CM for Traditional Variables - Batters") +
  theme(plot.title = element_text(hjust = 0.5))

pitchers_cm_plr_trad_plot = ggplot(data = cmlist[[2]], mapping = aes(x = Reference, y = Prediction, fill =
  geom_tile() +
  geom_text(aes(label = Freq), vjust = .5, fontface = "bold", alpha = 1) +
  scale_fill_manual(values = c(correct = "green", incorrect = "red")) +
  theme_bw() +
  labs(fill = "Quality of Prediction", alpha = "Proportion") +
```

```

ggtitle("CM for Traditional Variables - Pitchers") +
theme(plot.title = element_text(hjust = 0.5))

batters_cm_plr_adv_plot = ggplot(data = cmlist[[3]], mapping = aes(x = Reference, y = Prediction, fill =
  geom_tile() +
  geom_text(aes(label = Freq), vjust = .5, fontface = "bold", alpha = 1) +
  scale_fill_manual(values = c(correct = "green", incorrect = "red")) +
  theme_bw() +
  labs(fill = "Quality of Prediction", alpha = "Proportion") +
  ggtitle("CM for Advanced Variables - Batters") +
  theme(plot.title = element_text(hjust = 0.5))

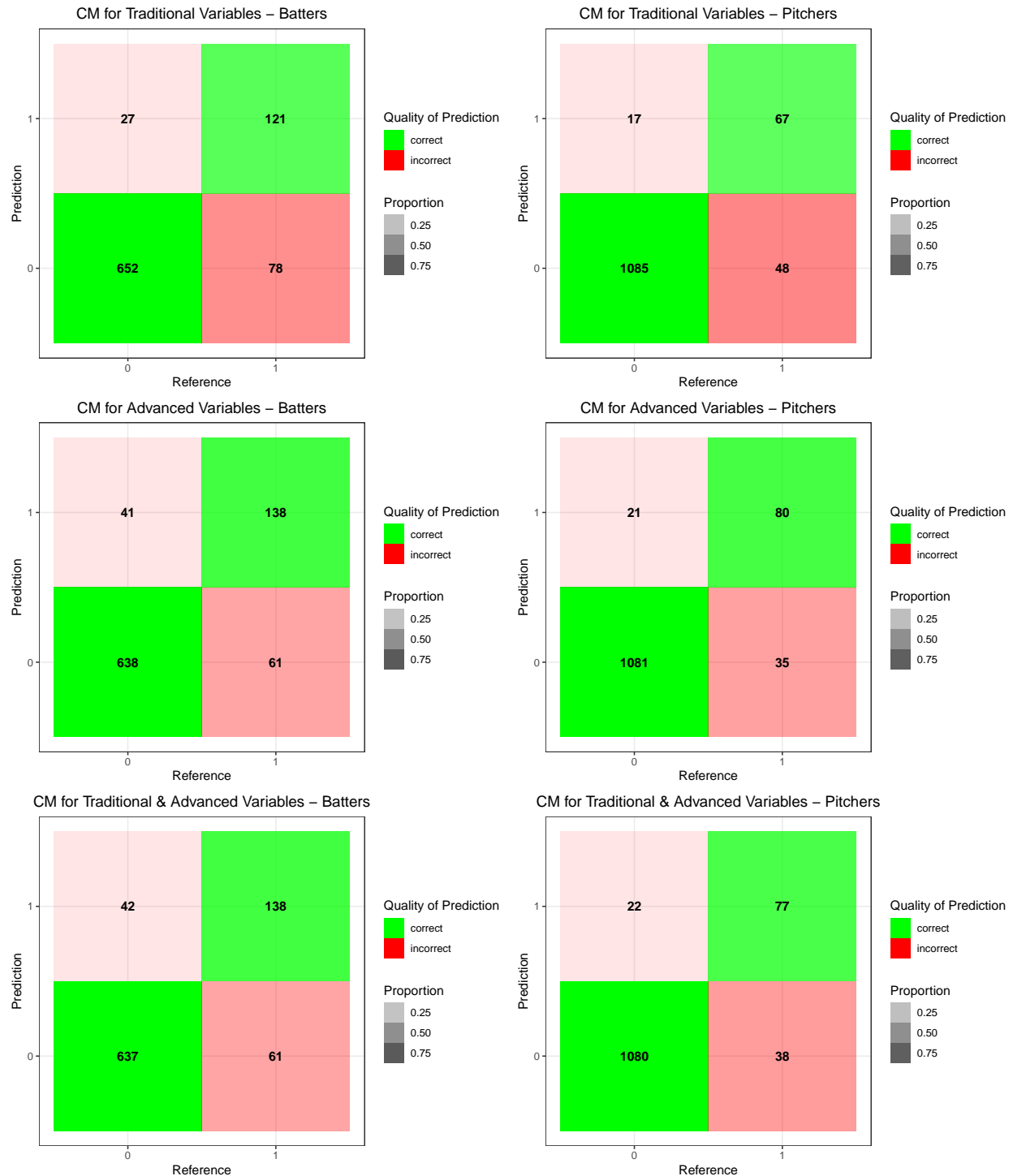
pitchers_cm_plr_adv_plot = ggplot(data = cmlist[[4]], mapping = aes(x = Reference, y = Prediction, fill =
  geom_tile() +
  geom_text(aes(label = Freq), vjust = .5, fontface = "bold", alpha = 1) +
  scale_fill_manual(values = c(correct = "green", incorrect = "red")) +
  theme_bw() +
  labs(fill = "Quality of Prediction", alpha = "Proportion") +
  ggtitle("CM for Advanced Variables - Pitchers") +
  theme(plot.title = element_text(hjust = 0.5))

batters_cm_plr_both_plot = ggplot(data = cmlist[[5]], mapping = aes(x = Reference, y = Prediction, fill =
  geom_tile() +
  geom_text(aes(label = Freq), vjust = .5, fontface = "bold", alpha = 1) +
  scale_fill_manual(values = c(correct = "green", incorrect = "red")) +
  theme_bw() +
  labs(fill = "Quality of Prediction", alpha = "Proportion") +
  ggtitle("CM for Traditional & Advanced Variables - Batters") +
  theme(plot.title = element_text(hjust = 0.5))

pitchers_cm_plr_both_plot = ggplot(data = cmlist[[6]], mapping = aes(x = Reference, y = Prediction, fill =
  geom_tile() +
  geom_text(aes(label = Freq), vjust = .5, fontface = "bold", alpha = 1) +
  scale_fill_manual(values = c(correct = "green", incorrect = "red")) +
  theme_bw() +
  labs(fill = "Quality of Prediction", alpha = "Proportion") +
  ggtitle("CM for Traditional & Advanced Variables - Pitchers") +
  theme(plot.title = element_text(hjust = 0.5))

library(cowplot)
plot_grid(batters_cm_plr_trad_plot,
pitchers_cm_plr_trad_plot,
batters_cm_plr_adv_plot,
pitchers_cm_plr_adv_plot,
batters_cm_plr_both_plot,
pitchers_cm_plr_both_plot, nrow = 3, ncol = 2)

```



```
tablelist = list(batters_cm_plr_trad, batters_cm_plr_adv, batters_cm_plr_both,
                 pitchers_cm_plr_trad, pitchers_cm_plr_adv, pitchers_cm_plr_both)

cm_accuracy = rep(NA,6)
cm_sensitivity = rep(NA,6)
cm_specificity = rep(NA,6)
cm_pospredvalue = rep(NA,6)
```

```

cm_negpredvalue = rep(NA,6)

for(i in 1:6){
  cm_accuracy[i] = round(tablelist[[i]]$overall['Accuracy'],3)
  cm_sensitivity[i] = round(tablelist[[i]]$byClass['Sensitivity'],3)
  cm_specificity[i] = round(tablelist[[i]]$byClass['Specificity'],3)
  cm_pospredvalue[i] = round(tablelist[[i]]$byClass['Pos Pred Value'],3)
  cm_negpredvalue[i] = round(tablelist[[i]]$byClass['Neg Pred Value'],3)
}

cm_names = c("Batters Traditional PLR", "Batters Advanced PLR",
             "Batters Trad/Adv then PLR", "Pitchers Traditional PLR", "Pitchers Advanced PLR", "Pitchers Trad/Adv then PLR")

cm_plr_table = as.data.frame(cbind(cm_names, cm_accuracy, cm_sensitivity, cm_specificity, cm_pospredvalue, cm_negpredvalue))
colnames(cm_plr_table) = c("Method", "Accuracy", "Sensitivity", "Specificity", "Positive PV", "Negative PV")
kable(cm_plr_table)

```

Method	Accuracy	Sensitivity	Specificity	Positive PV	Negative PV
Batters Traditional PLR	0.88	0.608	0.96	0.818	0.893
Batters Advanced PLR	0.884	0.693	0.94	0.771	0.913
Batters Trad/Adv then PLR	0.883	0.693	0.938	0.767	0.913
Pitchers Traditional PLR	0.947	0.583	0.985	0.798	0.958
Pitchers Advanced PLR	0.954	0.696	0.981	0.792	0.969
Pitchers Trad/Adv then PLR	0.951	0.67	0.98	0.778	0.966

```

pca_sensitivity = cm_table[c(1:2,4:6,8),c(1,3)]
plr_sensitivity = cm_plr_table[c(1:6),c(1,3)]

vc_comparison_table = rbind(pca_sensitivity, plr_sensitivity)
vc_comparison_table$Type = rep(c("Traditional Bat", "Advanced Bat", "Both Bat", "Traditional Pitch", "Advanced Pitch", "Both Pitch"), 6)

vc_comparison_table$Method <- factor(vc_comparison_table$Method, levels = c("Batters Traditional PCA", "Batters Advanced PCA", "Batters Trad/Adv PCA", "Pitchers Traditional PCA", "Pitchers Advanced PCA", "Pitchers Trad/Adv PCA"))

ggplot(vc_comparison_table, aes(x = Method, y = Sensitivity, fill = Type)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  ggtitle("Comparison of Sensitivity Across PCA and PLR Methods") +
  theme(plot.title = element_text(hjust = 0.5))

```

