Candy power ranking

Czuee Morey

29th July 2019

1) Business Question

The Lidl purchasing group wants to expand its candy offering, and wants to create a brand new product. Based on data from existing brand products, the goal is to find out which product characteristics drive customer sentiment and subsequently make a recommendation on a new product.

What are we trying to predict?

Which candy characteristics are correlated with the highest customer sentiment? Recommend a new candy based on these characteristics.

What type of problem is it?

This is a multivariate inference problem in which we have to find the characteristics that contribute to the highest percentage wins for candies in 269,000 matchups with another candy. A prediction is not important here, but we can make a simple regression model to recommend a new candy.

What type of data do we have?

The data is in csv format. It presents a header row with the column names. It contains binary variables and two numeric variables in percentiles (not percentage). The predictor variable is in percentage.

2) Data Cleaning

Import the dataset and check its size

```
candy <- read.csv("candy-data.csv", header = TRUE, na.strings = c("NA"," "))
dim(candy)
```

```
## [1] 85 13
```

View Data. Anything strange?

```
str(candy)
```

```
## 'data.frame':    85 obs. of  13 variables:
##  $ competitorname  : Factor w/ 85 levels "100 Grand","3 Musketeers",..: 1 2 45 46 3 4 5 6 7 8 ...
##  $ chocolate       : int  1 1 0 0 0 1 1 0 0 0 ...
##  $ fruity          : int  0 0 0 0 1 0 0 0 0 1 ...
##  $ caramel         : int  1 0 0 0 0 0 1 0 0 1 ...
##  $ peanutyalmondy  : int  0 0 0 0 0 1 1 1 0 0 ...
##  $ nougat          : int  0 1 0 0 0 0 1 0 0 0 ...
##  $ crispedricewafer: int  1 0 0 0 0 0 0 0 0 0 ...
##  $ hard            : int  0 0 0 0 0 0 0 0 0 0 ...
```

```
##  $ bar             : int  1 1 0 0 0 1 1 0 0 0 ...
##  $ pluribus        : int  0 0 0 0 0 0 0 1 1 0 ...
##  $ sugarpercent    : num  0.732 0.604 0.011 0.011 0.906 ...
##  $ pricepercent    : num  0.86 0.511 0.116 0.511 0.511 ...
##  $ winpercent      : num  67 67.6 32.3 46.1 52.3 ...
```

```
head(candy)
```

```
##   competitorname chocolate fruity caramel peanutyalmondy nougat
## 1      100 Grand         1      0       1              0      0
## 2   3 Musketeers         1      0       0              0      1
## 3       One dime         0      0       0              0      0
## 4    One quarter         0      0       0              0      0
## 5      Air Heads         0      1       0              0      0
## 6     Almond Joy         1      0       0              1      0
##   crispedricewafer hard bar pluribus sugarpercent pricepercent winpercent
## 1                1    0   1        0        0.732        0.860   66.97173
## 2                0    0   1        0        0.604        0.511   67.60294
## 3                0    0   0        0        0.011        0.116   32.26109
## 4                0    0   0        0        0.011        0.511   46.11650
## 5                0    0   0        0        0.906        0.511   52.34146
## 6                0    0   1        0        0.465        0.767   50.34755
```

Data Cleaning

1. There are no NAs or missing values in the file. So, the dataset is complete and it is not required to clean any errors or missing values.
2. Each column is "tidy", each value is placed in its own "cell", each variable in its own column, and each observation in its own row, so we don't need to transform it.
3. The variables chocolate to pluribus are binary, but are represented as integers. I will correct this if required.
4. "sugarpercent" is a percentile value for the dataset, not percentage of sugar. Same for "pricepercent". We do not have information about the underlying distribution with absolute values, but only the rank in the dataset.
5. Assumption: Competitor name is not a predictor (unless brand value drives sales) and is unique, so we can make it the row name. In any case, a brand name is not useful for Lidl to create a new candy.
6. The data seems to be complete, and no new features or dummy variables need to be added. We will explore quadratic and interaction terms while modeling. Also, there are only 12 variables, so feature reduction doesn't seem necessary.

Make row names as the Competitor name

```
row.names(candy) <- candy$competitorname
candy <- subset(candy, select =  -competitorname)
head(candy)
```

```
##              chocolate fruity caramel peanutyalmondy nougat
## 100 Grand            1      0       1              0      0
## 3 Musketeers         1      0       0              0      1
## One dime             0      0       0              0      0
## One quarter          0      0       0              0      0
```

```
## Air Heads            0      1       0              0      0
## Almond Joy           1      0       0              1      0
##              crispedricewafer hard bar pluribus sugarpercent pricepercent
## 100 Grand                    1    0   1        0        0.732        0.860
## 3 Musketeers                 0    0   1        0        0.604        0.511
## One dime                     0    0   0        0        0.011        0.116
## One quarter                  0    0   0        0        0.011        0.511
## Air Heads                    0    0   0        0        0.906        0.511
## Almond Joy                   0    0   1        0        0.465        0.767
##              winpercent
## 100 Grand      66.97173
## 3 Musketeers   67.60294
## One dime       32.26109
## One quarter    46.11650
## Air Heads      52.34146
## Almond Joy     50.34755
```

Scale winpercent to be on the same scale (0-1). I am not doing and center and scaling for other variables because they are binary or percentiles.

```
candy$winpercent <- candy$winpercent/100
```

2) Exploratory Data Analysis (EDA)

Distribution of each variable

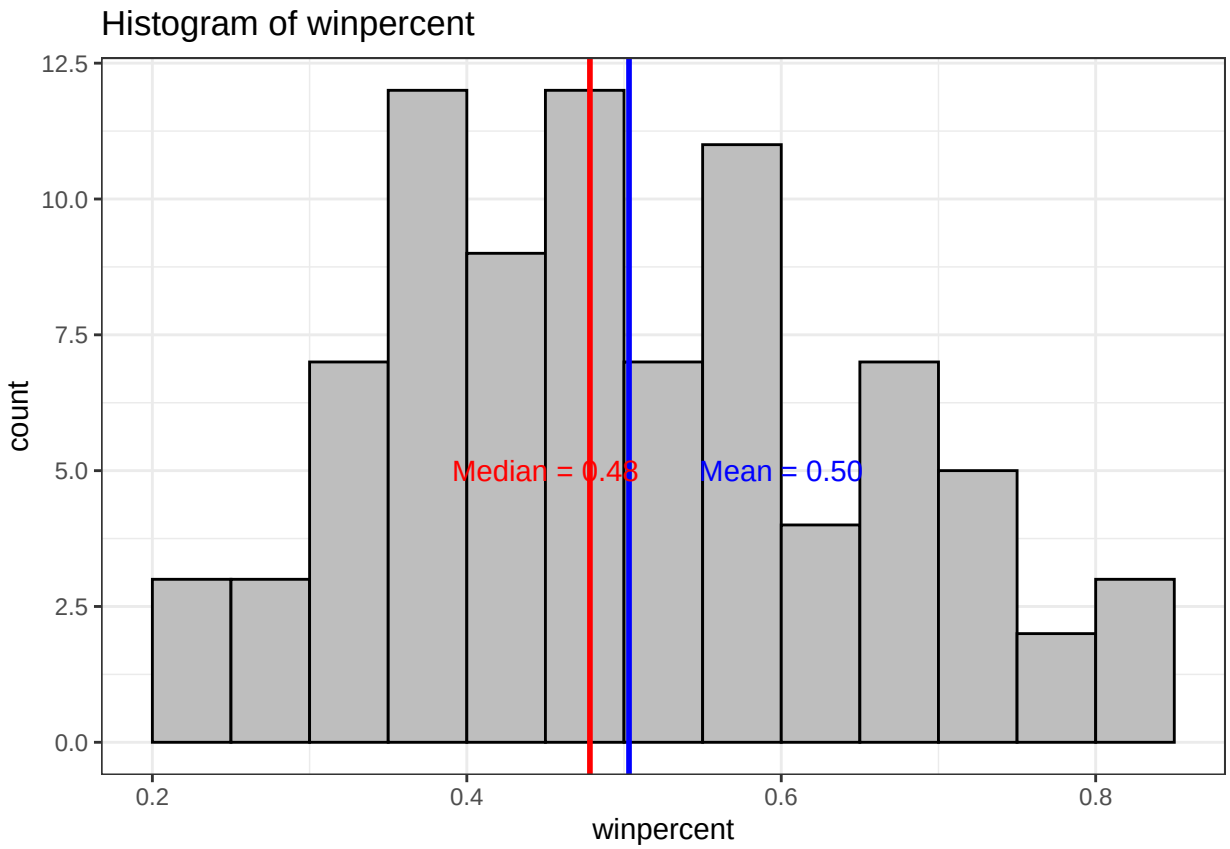
Let's start with looking at the distribution of continuous variables.

```
dlookr::describe(candy[,10:12])
```

```
## # A tibble: 3 x 26
##   variable     n    na  mean    sd se_mean   IQR skewness kurtosis   p00
##   <chr>    <dbl> <dbl> <dbl> <dbl>   <dbl> <dbl>    <dbl>    <dbl> <dbl>
## 1 sugarpe~    85     0 0.479 0.283  0.0307 0.512   0.0954   -1.13  0.011
## 2 pricepe~    85     0 0.469 0.286  0.0310 0.396   0.133    -1.15  0.011
## 3 winperc~    85     0 0.503 0.147  0.0160 0.207   0.332    -0.580 0.224
## # ... with 16 more variables: p01 <dbl>, p05 <dbl>, p10 <dbl>, p20 <dbl>,
## #   p25 <dbl>, p30 <dbl>, p40 <dbl>, p50 <dbl>, p60 <dbl>, p70 <dbl>,
## #   p75 <dbl>, p80 <dbl>, p90 <dbl>, p95 <dbl>, p99 <dbl>, p100 <dbl>
```

Response variable: winpercent

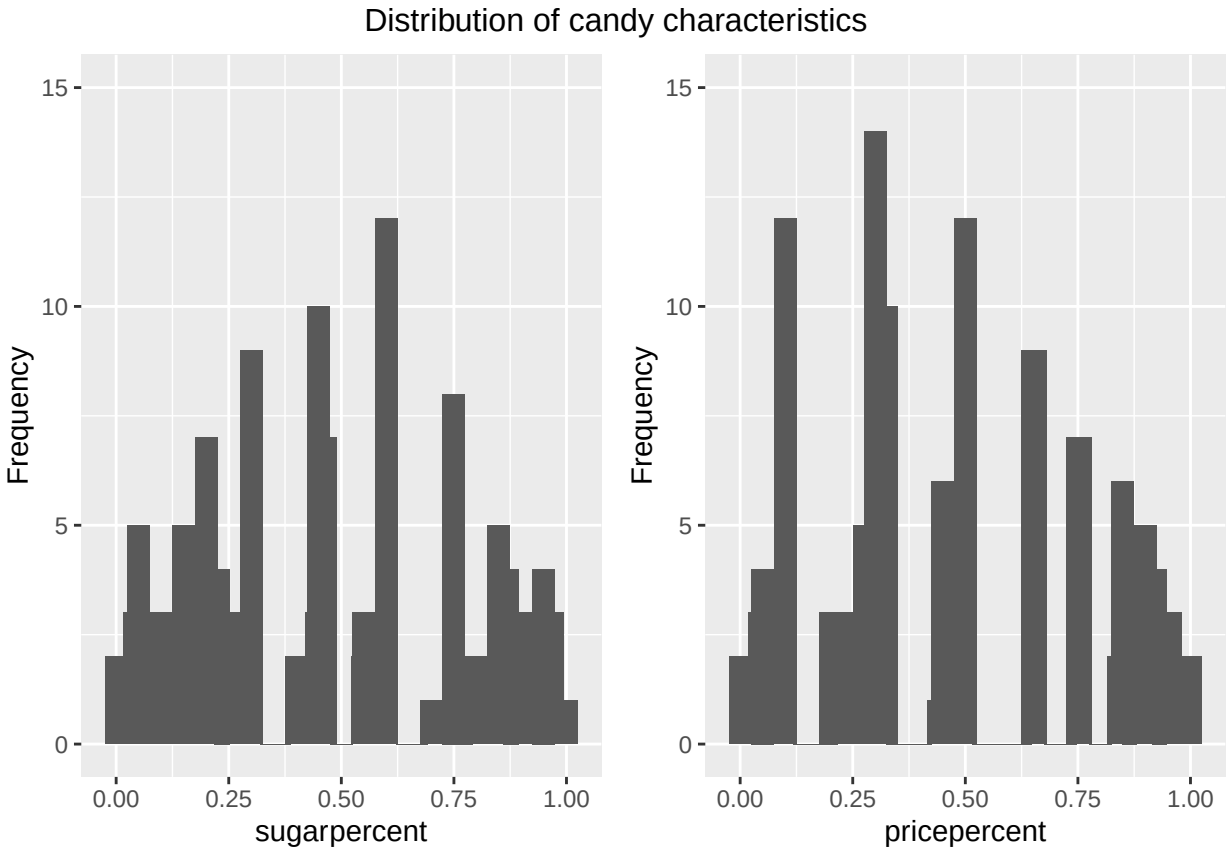
```
ggplot(data = candy) +
  geom_histogram(mapping = aes(x = winpercent), binwidth = 0.05, boundary = 0, fill = "gray", col = "bla
    geom_vline(xintercept = mean(candy$winpercent), col = "blue", size = 1) +
    geom_vline(xintercept = median(candy$winpercent), col = "red", size = 1) +
    annotate("text", label = "Median = 0.48", x = 0.45, y = 5, col = "red", size = 4) +
    annotate("text", label = "Mean = 0.50", x = 0.6, y = 5, col = "blue", size = 4) +
    ggtitle("Histogram of winpercent") +
    theme_bw()
```



For winpercent, the mean is close to the median, but the median is slightly lower indicating a slight positive skew. The kurtosis is low. The sd_mean is low, and SD is lower than the mean which means that the mean is well-defined. The distribution of winpercent is not exactly normal, but we can proceed without transforming it.

```
price.h <- qplot(pricepercent, data = candy, ylab = "Frequency") +stat_bin(binwidth = 0.05) +ylim(0,15)
sugar.h <- qplot(sugarpercent, data = candy, ylab = "Frequency") +stat_bin(binwidth = 0.05) +ylim(0,15)

grid.arrange(sugar.h, price.h, nrow = 1, top=textGrob("Distribution of candy characteristics"))
```

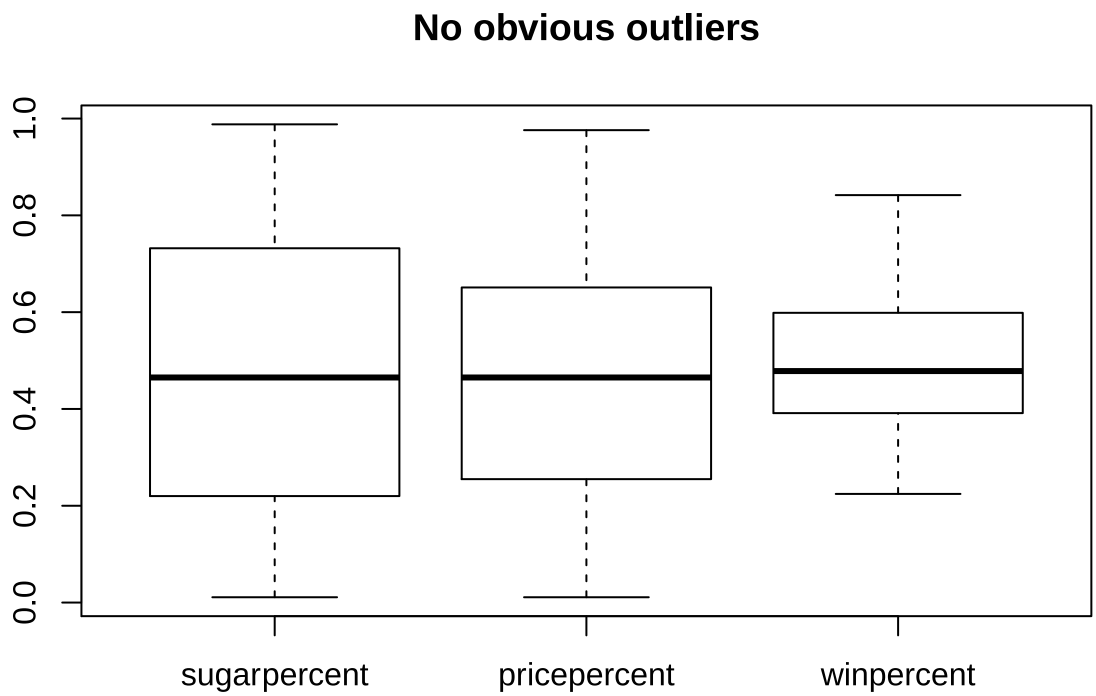
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



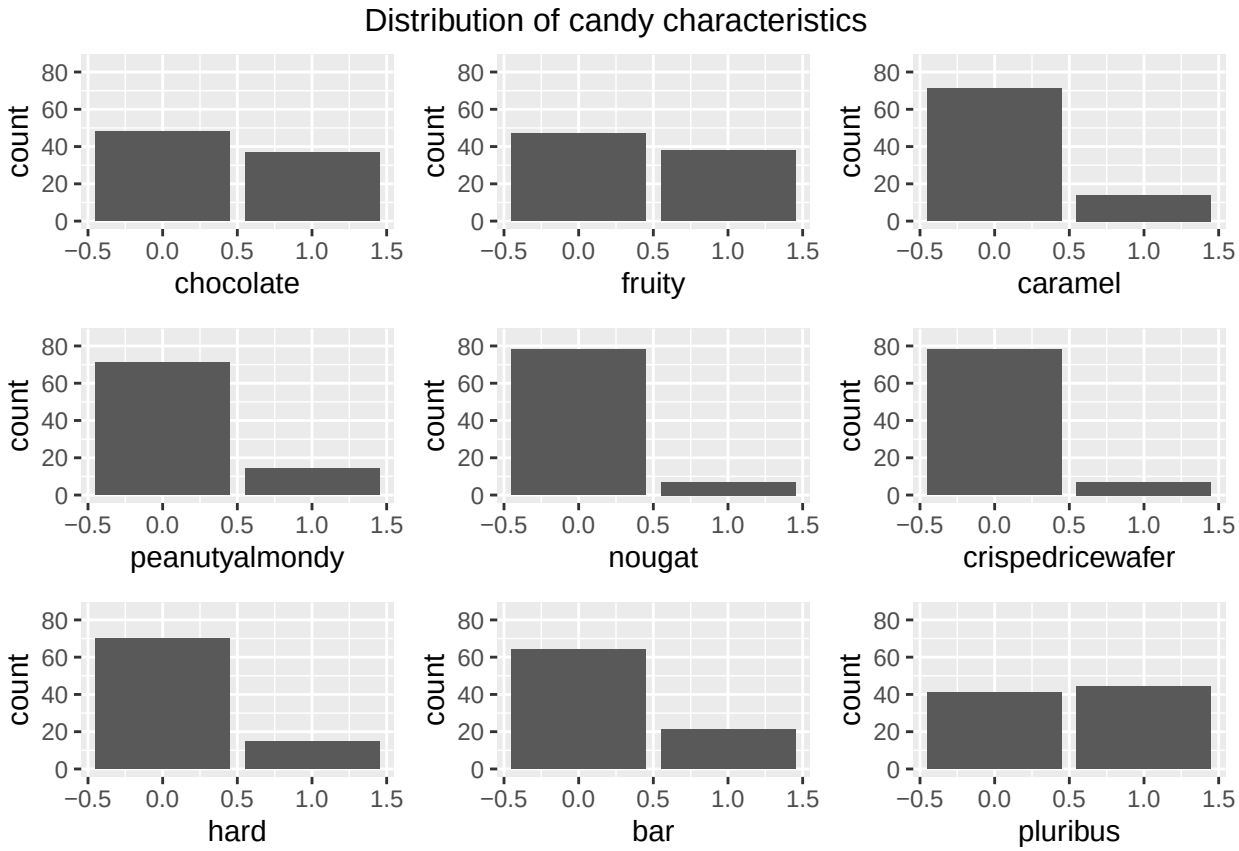
We already saw that the mean and median for the sugar and price variables is close with low skew and kurtosis. However, there are a lot of breaks in the data or no values for certain sugar and price percentiles. This could be because: a. Our data is sparse, with only 85 observations so certain sugar/price points will not be covered. b. Certain unit prices like €3.99 are more preferred than a continuous range. Same could be true for spikes in sugarpercent due to rounding, etc.

Outlier detection

```
boxplot(candy[10:12], main = "No obvious outliers")
```




```r
nm <- names(candy[1:9])
pltlist <- list()
for (i in seq_along(nm)) {
    pltlist[[i]] <- ggplot(candy, aes_string(x = nm[i]))+ geom_bar() + ylim(0, nrow(candy))
}
grid.arrange(grobs = pltlist, top=textGrob("Distribution of candy characteristics"))
```



Chocolate and fruity each are present in >35 candies while the other ingredients are in smaller quantities. More than 20% of candies are bars. Almost equal number of candies are single or present in a bag/box.

Do we have candies with both chocolate and fruity flavors?

```
xtabs(~chocolate+fruity, data = candy)
```

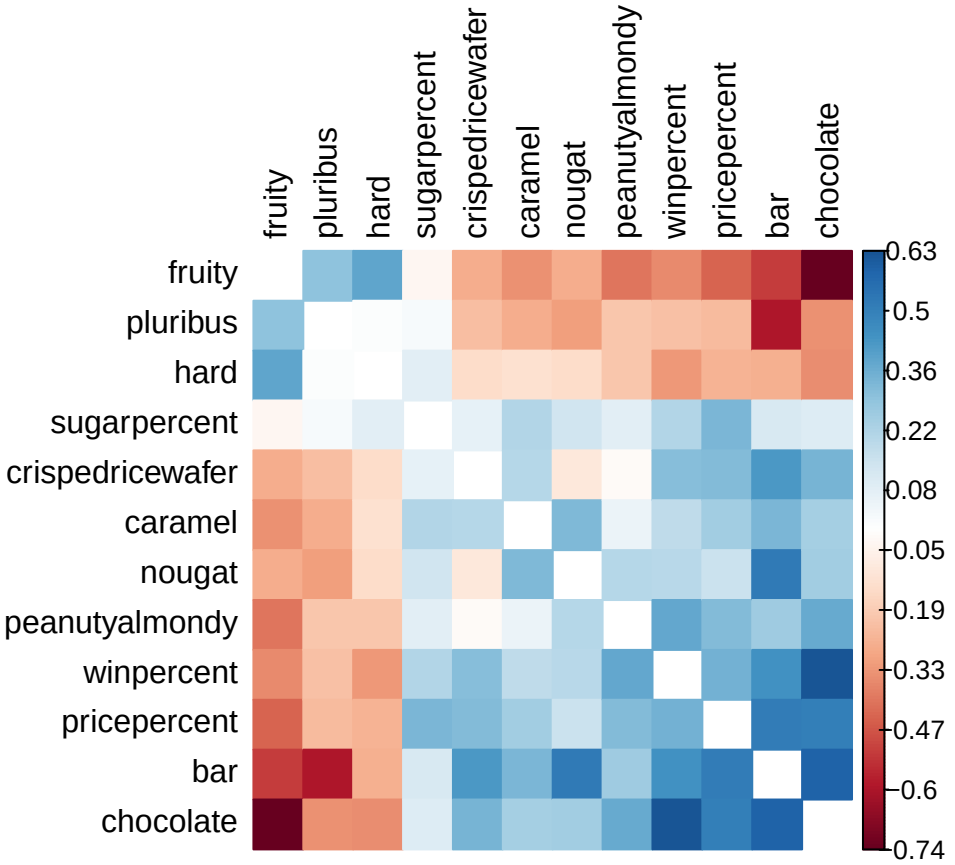
```
##           fruity
## chocolate  0  1
##         0 11 37
##         1 36  1
```

Most candies have either chocolate or fruity, not both. 11 candies have neither.

Correlation between different variables

Quick and dirty correlation between variables

```
corrplot(cor(candy, method = "spearman"),
         method = "color",
         tl.col = "black",
         order =  "FPC",
         is.corr = FALSE,
         diag = FALSE
         )
```



Caveat: Since most variables are binary, this is not a proper correlation matrix, but it gives us an idea about the trends.

1. Candies that are fruity, hard and pluribus separate out as a cluster and are negatively correlated with the other variables.
2. Chocolate candies are most preferred by customers. Other winning characteristics are peanutyalmondy, bars, crispedricewafer.
3. Chocolate candies tend to be bars and have peanutyalmondy, crispedricewafer and a few other ingredients. But they tend to be expensive.
4. Bars generally have chocolate and nougat but are not pluribus. They are also the most expensive and tend to have high sugar.
5. No two variables have an extremely high correlation that shows replicates, so no need to remove any variables.

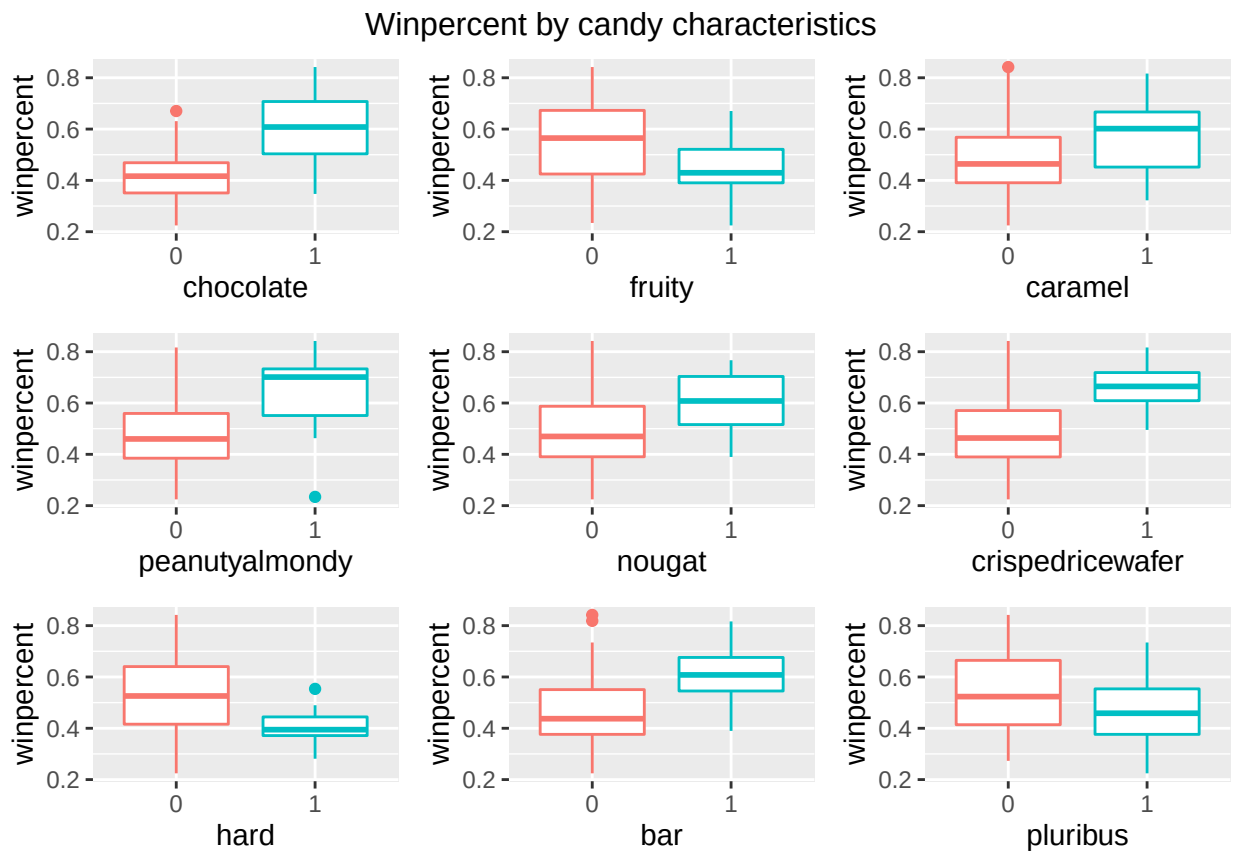
Let's drill deeper into the distribution and correlations.

```
cov(candy[ ,10:12])
```

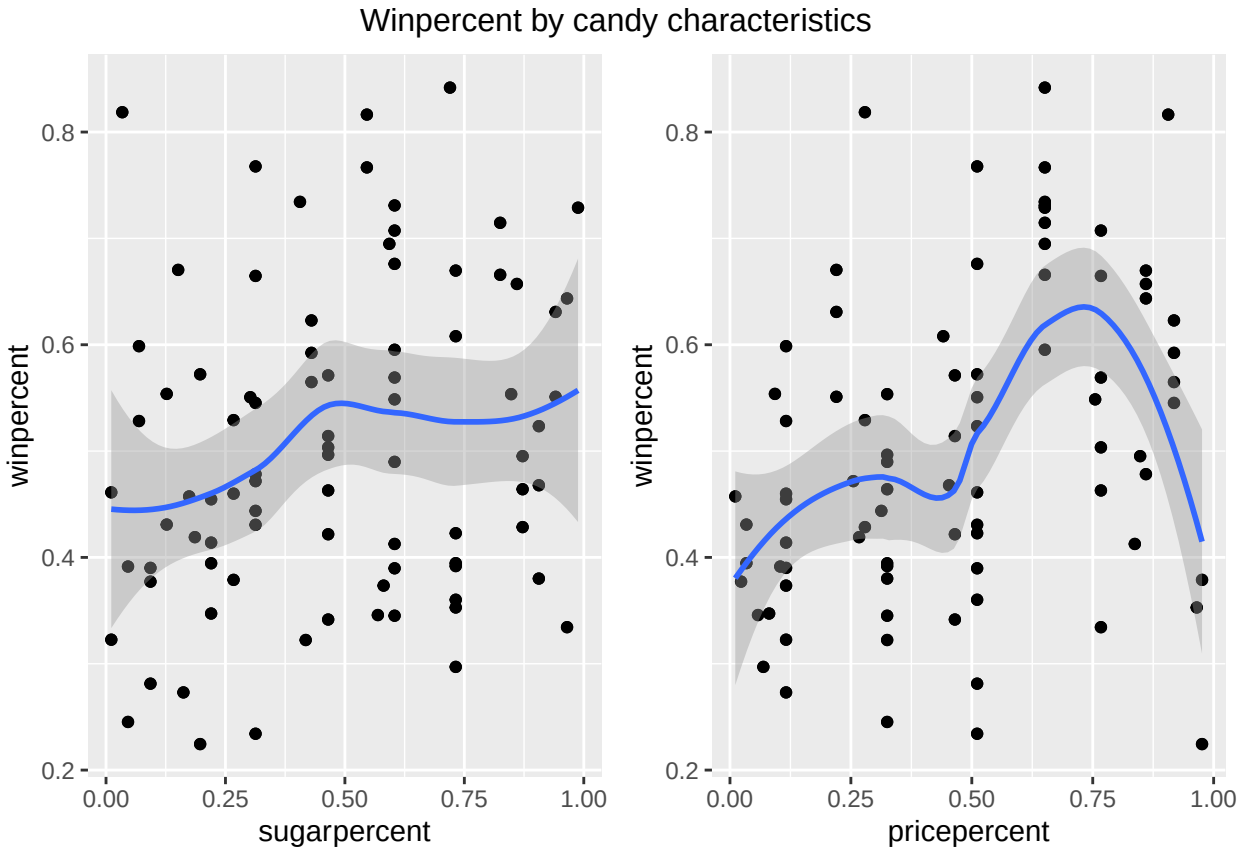
```
##              sugarpercent pricepercent  winpercent
## sugarpercent  0.079963324   0.02664055 0.009534717
## pricepercent  0.026640553   0.08164713 0.014519120
## winpercent    0.009534717   0.01451912 0.021651231
```

The correlation between winpercent and pricepercent is around 0.35, a moderate positive correlation.

```
winlist  <- list()
candy_n <- lapply(candy[ , 1:9], factor)
candy_n <- data.frame(candy_n, winpercent = candy$winpercent)
for (i in seq_along(nm)) {
    winlist[[i]] <- ggplot(candy_n, aes_string(nm[i], y ="winpercent")) +
        geom_boxplot(aes_string(color = nm[i]), show.legend = FALSE)
}


win1 <- qplot(sugarpercent, winpercent, data = candy) + geom_point() + geom_smooth(method = "loess")
win2 <- qplot(pricepercent, winpercent, data = candy) + geom_point() + geom_smooth(method = "loess")

grid.arrange(grobs = winlist, top=textGrob("Winpercent by candy characteristics"))
```




```
grid.arrange(win1, win2, top=textGrob("Winpercent by candy characteristics"), nrow = 1)
```

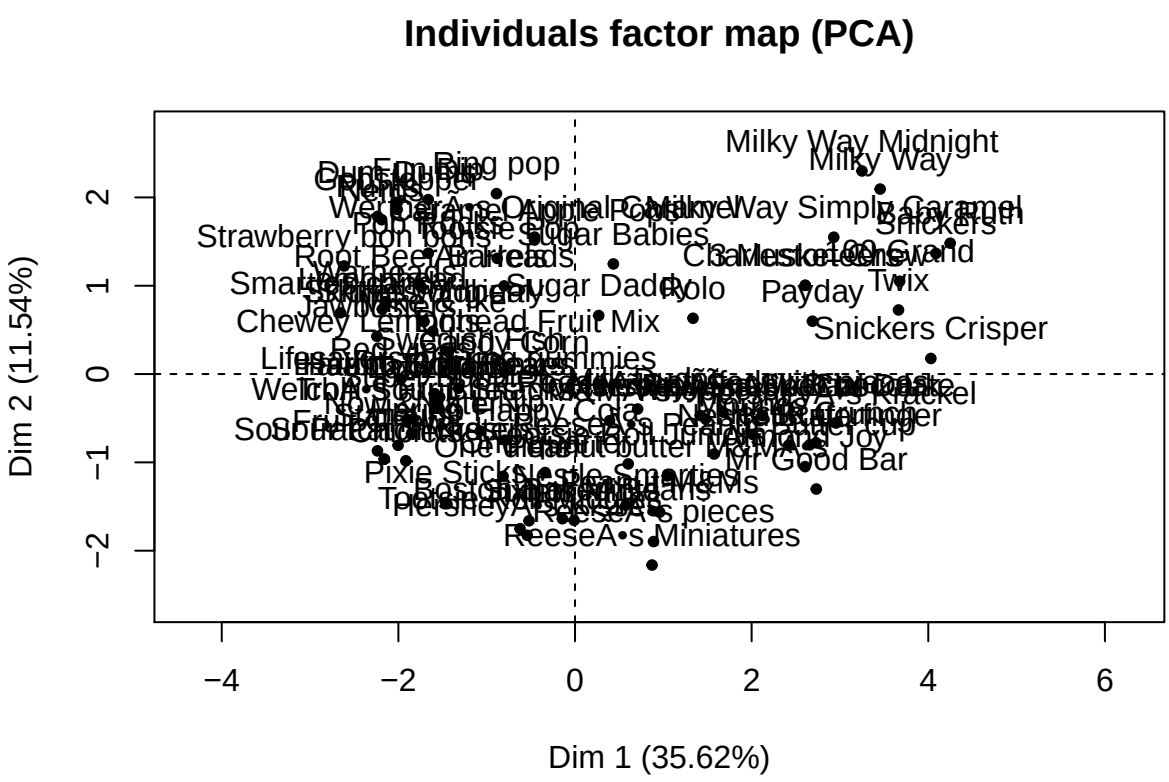


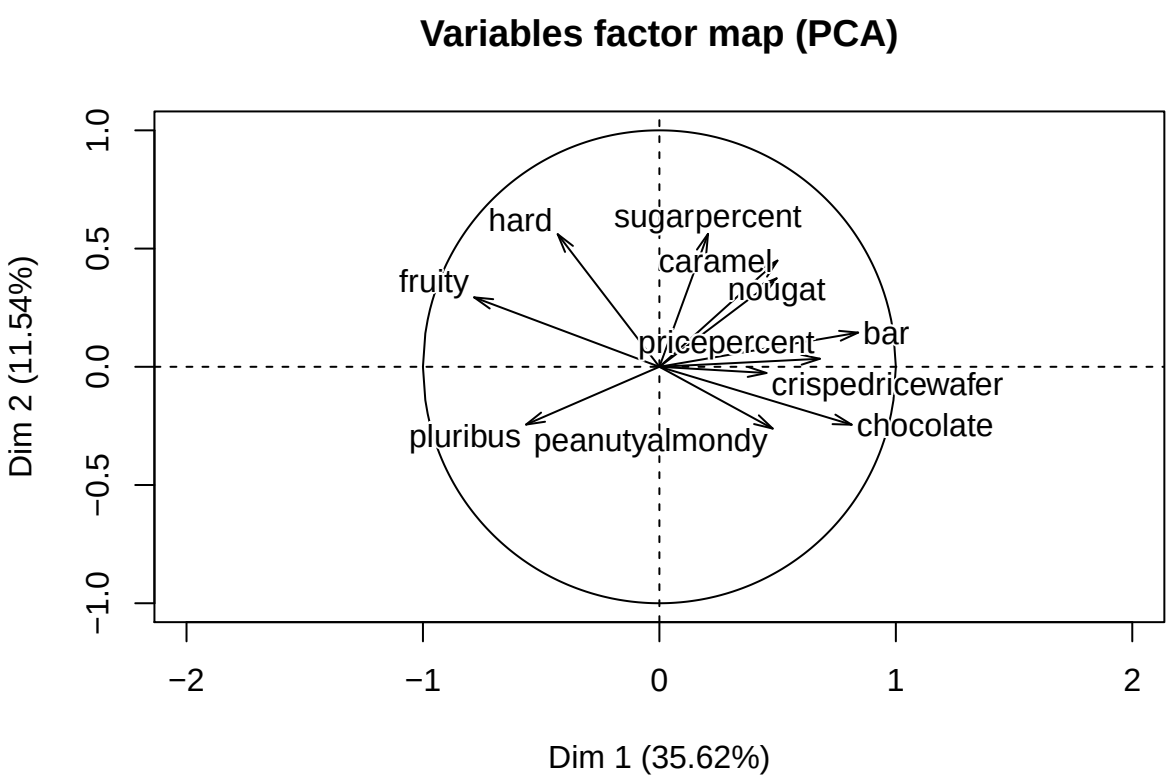
The boxplots confirm our first impressions that chocolate, bar, peanutyalmondy and crispedricewafer have high winpercent. Of course, there are some outliers like a few bar0 points with very high winpercent or peanutyalmondy1 with very low winpercent.

The correlation between winpercent and pricepercent does not follow a linear relationship - moderate price points are preferred but very high price points are not preferred. The relationship with sugarpercent is positive but weak as seen before.

I will run a categorical PCA (multiple correspondence analysis) to visualize the distance between the binary variables, and have a better picture of what contributes to varibility in the dataset.

```
res.pca <- PCA(candy[ ,1:11],
               #quanti.sup = c(10:11),
               ncp = 5, graph = TRUE)
```

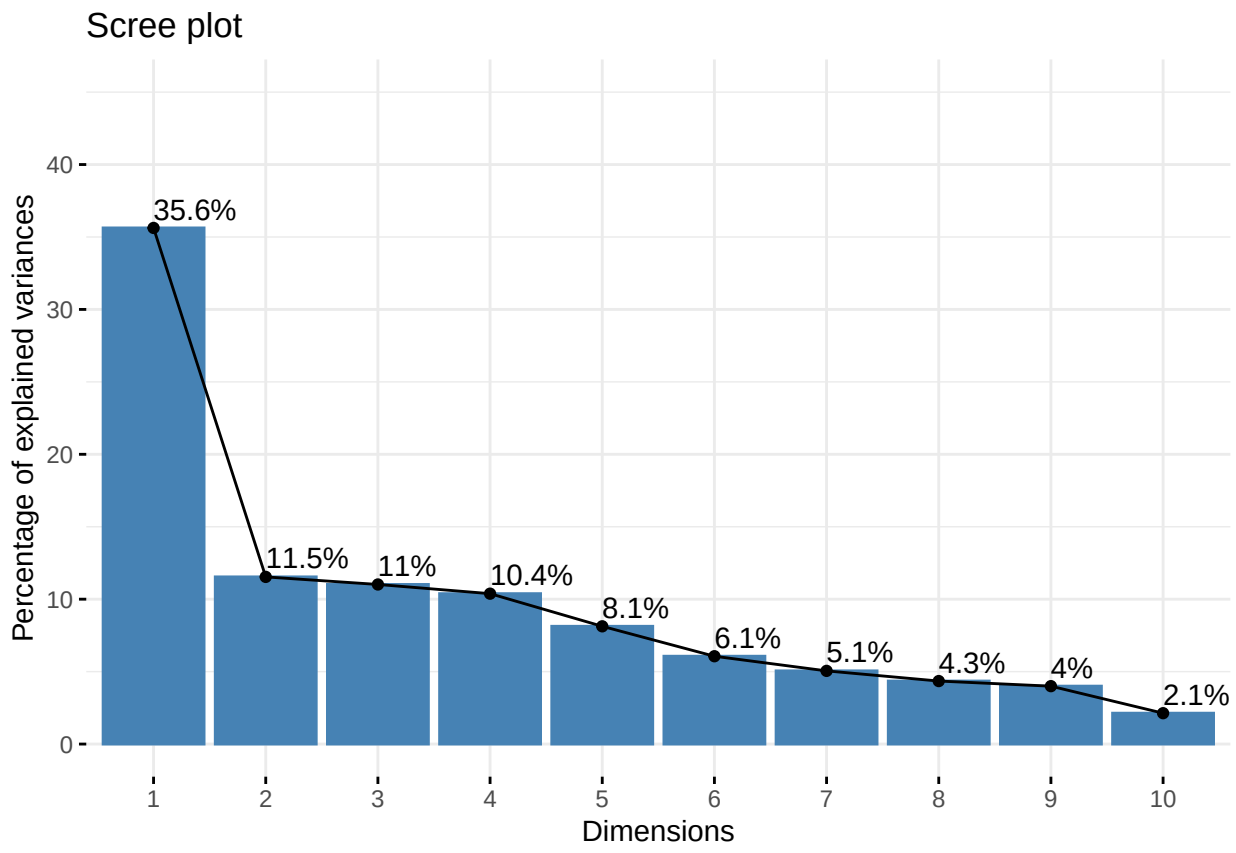

Individuals factor map (PCA)
Milky Way Midnight
Milky Way
Milky Way Simply Caramel
Snickers
Strawberry bon bons
Sugar Babies
100 Grand
Twix
Sugar Daddy
Payday
Snickers Crisper
Chewey Lemonhead Fruit Mix
Reese's Miniatures
Dim 1 (35.62%)
Dim 2 (11.54%)
−4
−2
0
2
4
6
−2
−1
0
1
2




```
res.pca$eig
```

```
##         eigenvalue percentage of variance
## comp 1   3.9187452              35.624956
## comp 2   1.2692196              11.538360
## comp 3   1.2118369              11.016699
## comp 4   1.1419752              10.381593
## comp 5   0.8941622               8.128748
## comp 6   0.6668683               6.062439
## comp 7   0.5557406               5.052188
## comp 8   0.4784604               4.349640
## comp 9   0.4397143               3.997402
## comp 10  0.2344714               2.131558
## comp 11  0.1888059               1.716417
##         cumulative percentage of variance
## comp 1                           35.62496
## comp 2                           47.16332
## comp 3                           58.18002
## comp 4                           68.56161
## comp 5                           76.69036
## comp 6                           82.75279
## comp 7                           87.80498
## comp 8                           92.15462
## comp 9                           96.15202
## comp 10                          98.28358
## comp 11                         100.00000
```

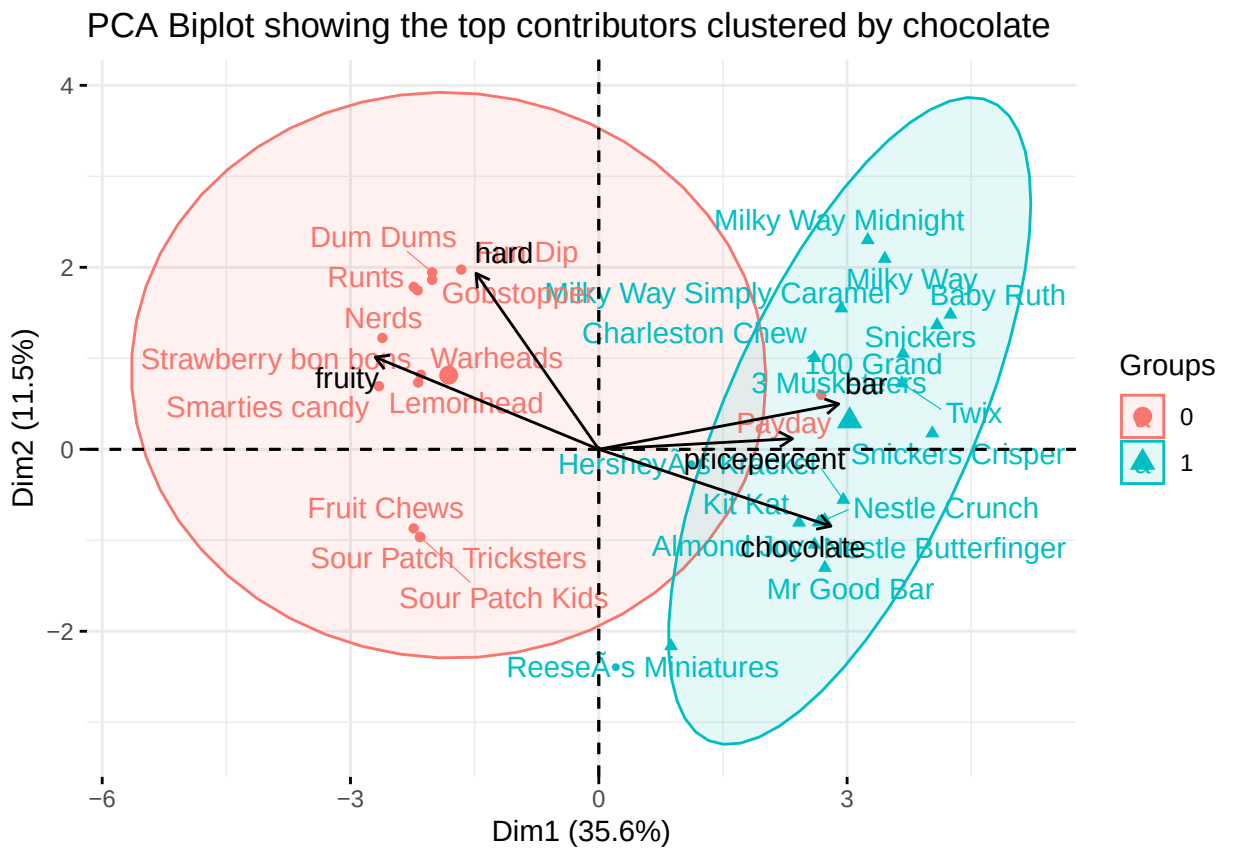
```
fviz_screeplot(res.pca, addlabels = TRUE, ylim = c(0, 45))
```



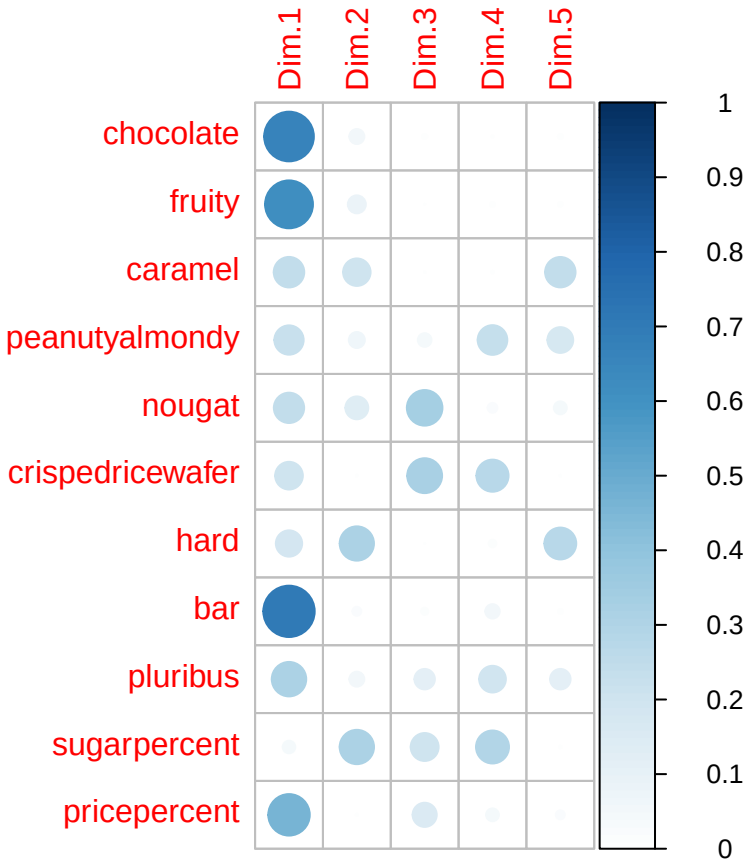
Dim1 has the highest contribution of 35%. Dim 2, 3 & 4 have almost equal contribution about 11%.

It takes 11 dimesions to cover 11% of the variability, which is equal to the number of variables so all the variables are important.

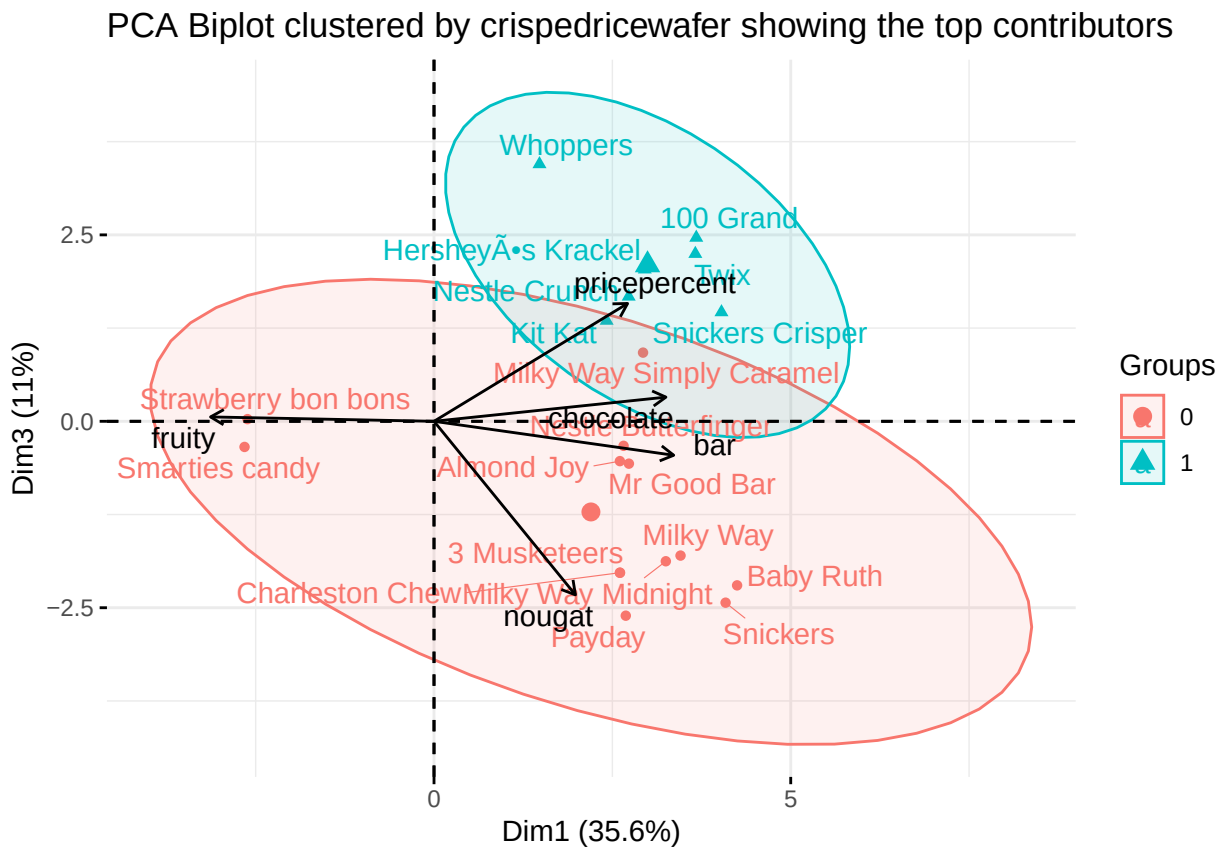
```
grp <- as.factor(candy[ , "chocolate"])
fviz_pca_biplot(res.pca,
                #geom.ind = c("point"),
                select.ind = list(contrib = 30),
                habillage = grp, addEllipses = TRUE, ellipse.level = 0.95,
                select.var = list(contrib = 5),
                repel = T, col.var = "black",
                title = "PCA Biplot showing the top contributors clustered by chocolate"
                )
```




```
corrplot::corrplot(res.pca$var$cos2, is.corr=FALSE, cl.lim = c(0, 1), cl.ratio = 0.7)
```




```
grp <- as.factor(candy[ , "crispedricewafer"])
fviz_pca_biplot(res.pca,
                axes = c(1,3),
                select.ind = list(contrib = 20),
                habillage = grp, addEllipses = TRUE, ellipse.level = 0.95,
                select.var = list(contrib = 5),
                repel = T, col.var = "black",
                title = "PCA Biplot clustered by crispedricewafer showing the top contributors"
                )
```



Nougat is seen as a top contributor here, probably due to a cluster of snickers, milkyway, etc.

Let's review our conclusions: Chocolate, bar, peanutyalmondy and crispedricewafer have high winpercent. Pricepercent also has moderate positive correlation with winpercent. Fruity, hard and pluribus are negatively correlated with winpercent.

Chocolate, bar, pricepercent, fruity, hard, and to a lesser extent crispedricewafer & nougat explain about 50% of the variability in the dataset.

4) Modeling

We are not sure the data is randomized, so we will shuffle it just in case:

```
set.seed(123)
candy_rand <- candy[sample(1:nrow(candy)), ]
dim(candy_rand)
```

```
## [1] 85 12
```

This is a small dataset with only 85 rows and considerable variation. We will divide the dataset into train and test sets and make sure we use cross validation when we train our model. In that way we ensure we are using our few observations as well as we can.

Split into training & test sets

```r
set.seed(9)
inTrain <- createDataPartition(candy_rand$winpercent, p=0.8, list =FALSE)

ctrain <- candy_rand[inTrain,]
ctest <- candy_rand[-inTrain,]

dim(ctrain); dim(ctest)
```

```
## [1] 69 12
```

```
## [1] 16 12
```

Linear Models

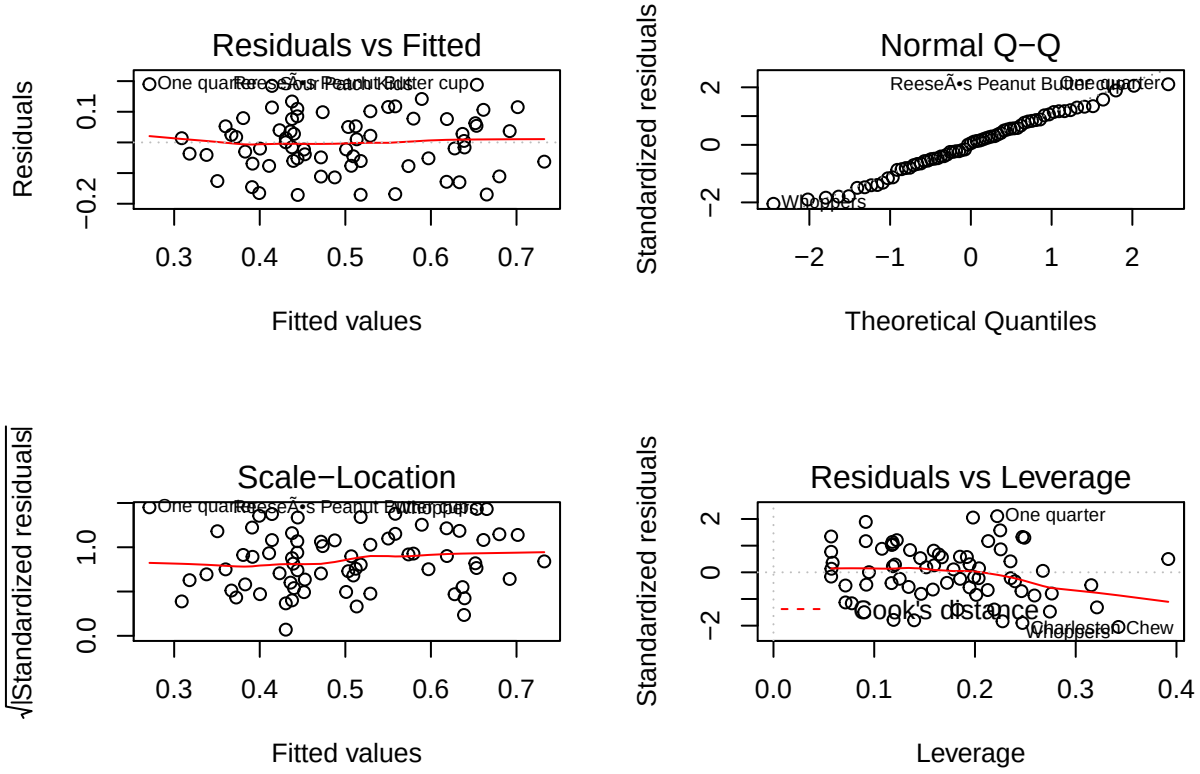
```r
fit1 <- lm(winpercent ~ ., data = ctrain)
summary(fit1)
```

```
##
## Call:
## lm(formula = winpercent ~ ., data = ctrain)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.171339 -0.060625  0.004839  0.076167  0.190022
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       0.31835    0.04784   6.654 1.20e-08 ***
## chocolate         0.19255    0.04131   4.661 1.94e-05 ***
## fruity            0.11390    0.04250   2.680  0.00961 **
## caramel           0.04437    0.04473   0.992  0.32549
## peanutyalmondy    0.10175    0.04240   2.400  0.01968 *
## nougat           -0.01761    0.05814  -0.303  0.76312
## crispedricewafer  0.12712    0.05523   2.302  0.02503 *
## hard             -0.06201    0.03716  -1.669  0.10068
## bar               0.02730    0.05190   0.526  0.60087
## pluribus         -0.01678    0.03261  -0.514  0.60891
## sugarpercent      0.14324    0.04827   2.967  0.00438 **
## pricepercent     -0.09546    0.05813  -1.642  0.10605
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1023 on 57 degrees of freedom
## Multiple R-squared:  0.5803, Adjusted R-squared:  0.4993
## F-statistic: 7.163 on 11 and 57 DF,  p-value: 1.766e-07
```

Chocolate, fruity, peanutyalmondy and crispedricewafer have high significance in the model, which was expected. Sugarpercent is also significant. Although pricepercent shows some correlation, it is not linear and is hence not picked up.

The R-squared is not very high at 0.58 and RSE is 0.102.

```
par(mfrow=c(2,2))
plot(fit1)
```



F-statistic is significant with a small p-value, which indicates that there is a relationship between the predictors and response. The Residuals vs Fitted values and Q-Q plot are close to the expected values, so it seems like a good fit. There are a few outliers, but they don't have high leverage. "One quarter" has medium leverage.

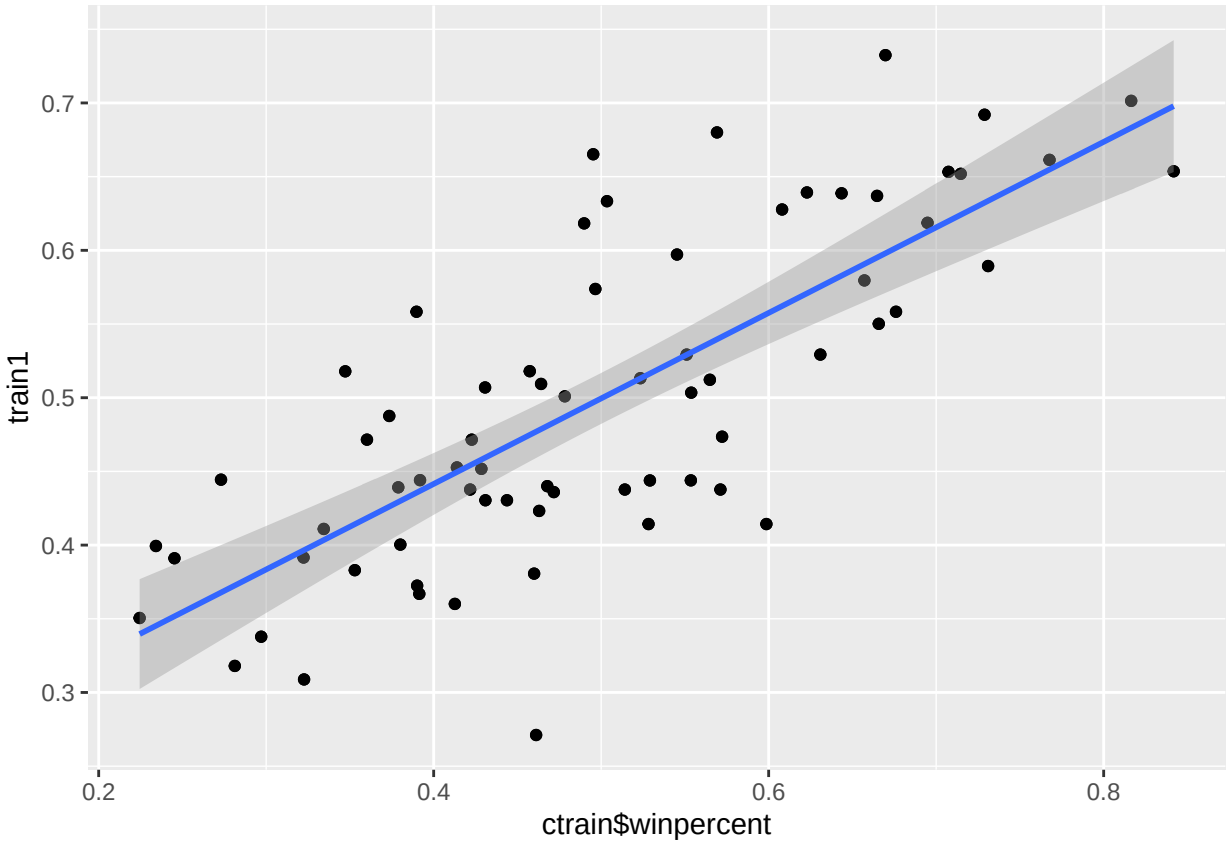
```
confint(fit1)
```

```
##                       2.5 %     97.5 %
## (Intercept)       0.22254554 0.41414487
## chocolate         0.10981886 0.27527957
## fruity            0.02879756 0.19900542
## caramel          -0.04521061 0.13394396
## peanutyalmondy    0.01685455 0.18665338
## nougat           -0.13402511 0.09881296
## crispedricewafer  0.01652636 0.23771173
## hard             -0.13641815 0.01240682
## bar              -0.07661989 0.13122550
## pluribus         -0.08206862 0.04851856
## sugarpercent      0.04657759 0.23989337
## pricepercent     -0.21185004 0.02093905
```

```
train1 <- predict(fit1, newdata = ctrain)
RMSE(train1, ctrain$winpercent)
```

```
## [1] 0.09299118
```

```
qplot(ctrain$winpercent, train1) + geom_point() + geom_smooth(method = lm)
```



Overall, the basic model seems like a good fit with RMSE 0.093.

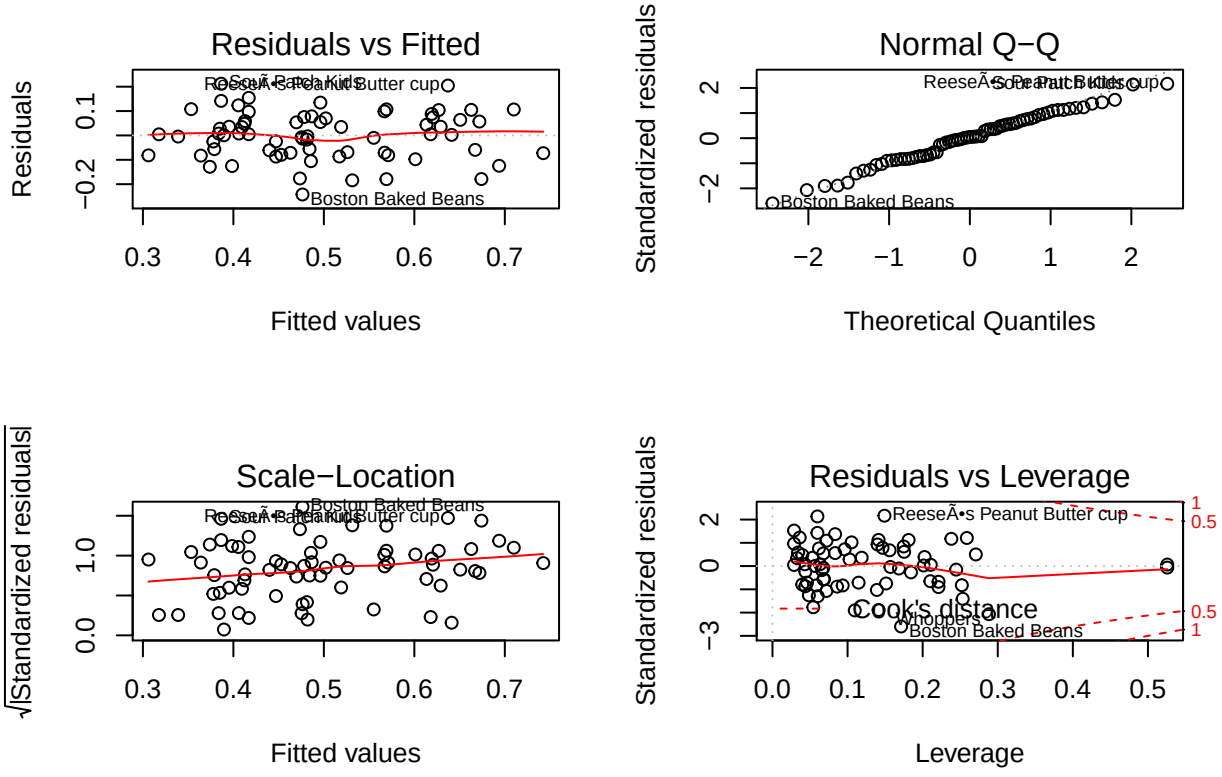
I will try some other combinations and interactions.

```
fit2 <- lm(winpercent ~ chocolate+ caramel+ bar+ peanutyalmondy+ crispedricewafer+ sugarpercent+ chocola

summary(fit2)
```

```
##
## Call:
## lm(formula = winpercent ~ chocolate + caramel + bar + peanutyalmondy +
##     crispedricewafer + sugarpercent + chocolate:caramel + I(pricepercent^2),
##     data = ctrain)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.242229 -0.071252  0.004596  0.070054  0.212175
##
```

```
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)         0.37889    0.02734  13.857  < 2e-16 ***
## chocolate           0.12461    0.03321   3.752 0.000397 ***
## caramel            -0.10459    0.07561  -1.383 0.171710
## bar                 0.01415    0.03934   0.360 0.720396
## peanutyalmondy      0.08380    0.04006   2.092 0.040672 *
## crispedricewafer    0.13154    0.05180   2.539 0.013725 *
## sugarpercent        0.12995    0.04549   2.857 0.005874 **
## I(pricepercent^2) -0.10323    0.05332  -1.936 0.057598 .
## chocolate:caramel   0.17913    0.08812   2.033 0.046504 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1025 on 60 degrees of freedom
## Multiple R-squared:  0.557,  Adjusted R-squared:  0.4979
## F-statistic: 9.429 on 8 and 60 DF,  p-value: 2.519e-08
```

```
par(mfrow=c(2,2))
plot(fit2)
```



Only 2 points seem to have high leverage.

```
train2 <- predict(fit2, newdata = ctrain)
RMSE(train2, ctrain$winpercent)
```

```
## [1] 0.09553617
```

RMSE on training set is not very different than for fit1.

Random forest

```
set.seed(18)
fit.rf <- train(winpercent ~ ., data = ctrain, method ="rf", importance = TRUE)
fit.rf
```

```
## Random Forest
##
## 69 samples
## 11 predictors
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 69, 69, 69, 69, 69, 69, ...
## Resampling results across tuning parameters:
##
##   mtry  RMSE       Rsquared   MAE
##    2    0.1151062  0.4655196  0.09344634
##    6    0.1151550  0.4467472  0.09335302
##   11    0.1192002  0.4131551  0.09715609
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 2.
```

46% of the variability is explained by this model.

Train several models

```
doParallel::registerDoParallel(4) # here I'm using 4 cores from my computer
#getDoParWorkers()

set.seed(987) # for replicability

my_control <- trainControl(method = "cv", # for "cross-validation"
                           number = 5, # number of k-folds
                           savePredictions = "final",
                           allowParallel = TRUE)
```

```
set.seed(18)

model_list <- caretList(winpercent ~ .,
                        data = ctrain,
                        trControl = my_control,
                        methodList = c("lm", "rpart", "svmLinear", "rf", "xgbTree", "glm"),
                        tuneList = NULL, # no manual hyperparameter tuning
                        continue_on_fail = FALSE # stops if something fails
                        #preProcess  = c("center","scale")
                        )
```

```
model_list$rf
```

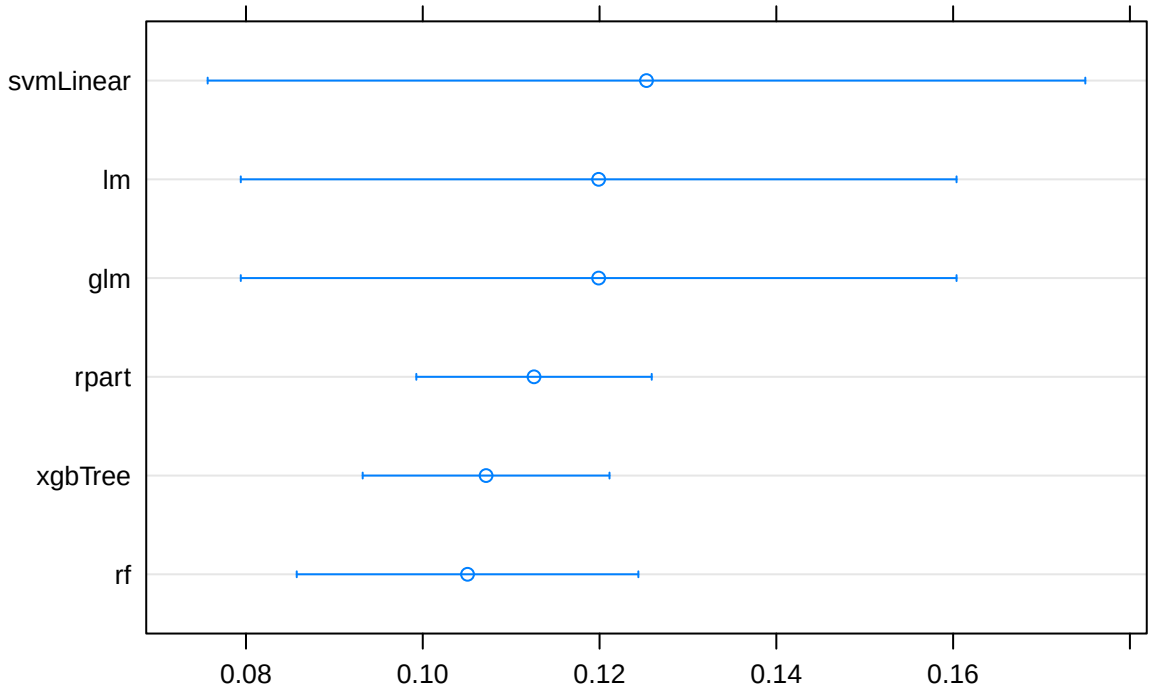
```
## Random Forest
##
## 69 samples
## 11 predictors
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 56, 55, 56, 54, 55
## Resampling results across tuning parameters:
##
##   mtry  RMSE       Rsquared   MAE
##    2    0.1069277  0.4684755  0.08587968
##    6    0.1053491  0.4866048  0.08169957
##   11    0.1050655  0.4957272  0.08216107
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 11.
```

```
options(digits = 3)

model_results <- data.frame(LM = min(model_list$lm$results$RMSE),
                            SVM = min(model_list$svmLinear$results$RMSE),
                            RPART= min(model_list$rpart$results$RMSE),
                            RF = min(model_list$rf$results$RMSE),
                            XGBT = min(model_list$xgbTree$results$RMSE),
                            GLM = min(model_list$glm$results$RMSE))


print(model_results)
```

```
##     LM   SVM RPART    RF  XGBT  GLM
## 1 0.12 0.125 0.113 0.105 0.107 0.12
```

Random forest has the lowest RMSE,but it is also difficult to interpret than a LM. There is also a possibility that random forest is overfitting to the data.

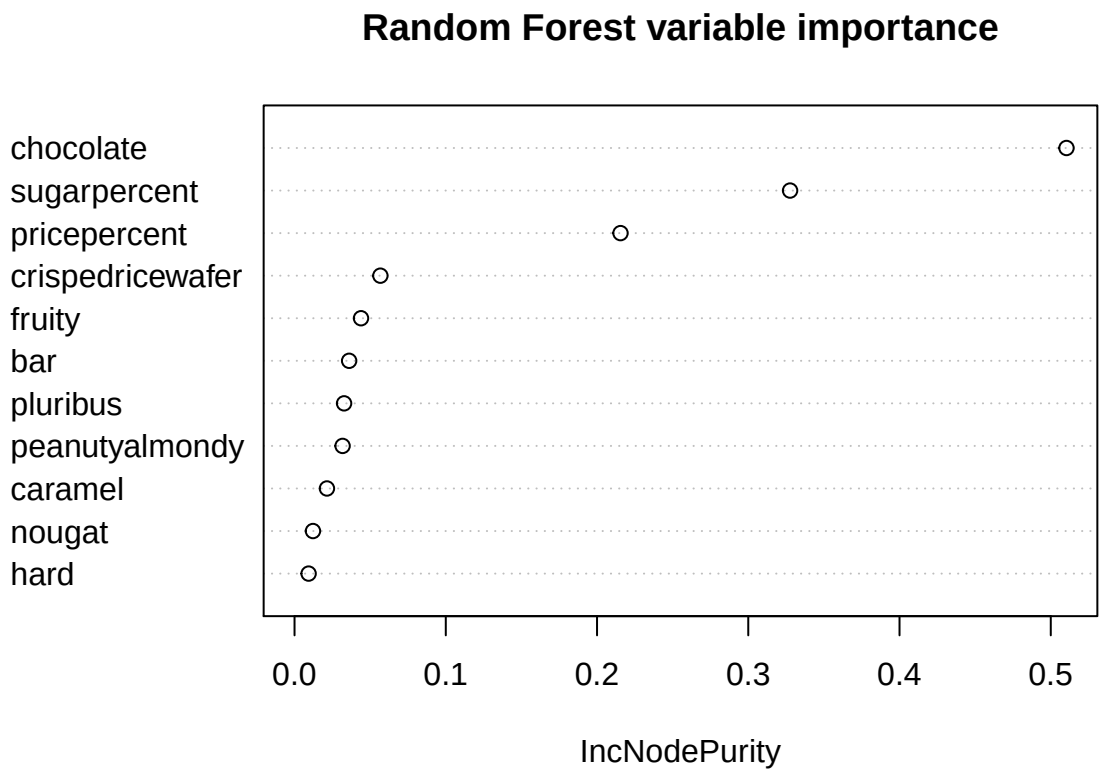
```
resamples <- resamples(model_list)
par(mfrow=c(2,2))
dotplot(resamples, metric = "RMSE", title = "Dotplot for RMSE")
```



Confidence Level: 0.95

```
#dotplot(resamples, metric = "Rsquared", title = "Dotplot for Rsquared")
```

```
randf <- model_list$rf
#head(getTree(randf$finalModel, 1))

#importance(randf$finalModel)
varImpPlot(randf$finalModel, main =  "Random Forest variable importance")
```



We will use fit1, fit2 and model_list for testing.

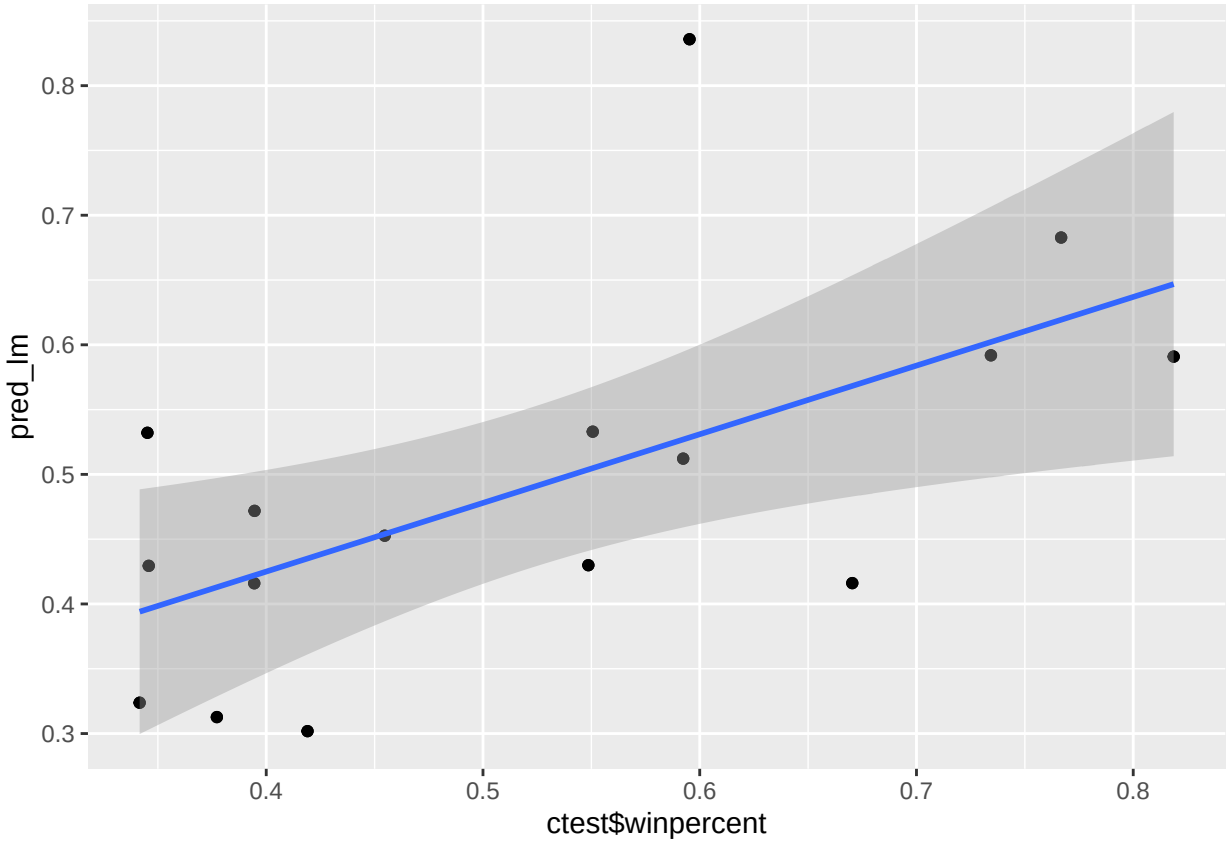
Predict on Test dataset

```
x_test <- ctest[ ,1:11]
y_test<- ctest[ ,12]

pred_lm <- predict.train(model_list$lm, newdata = x_test)
pred_svm <- predict.train(model_list$svmLinear, newdata = x_test)
pred_rf <- predict.train(model_list$rf, newdata = x_test)
pred_xgbT <- predict.train(model_list$xgbTree, newdata = x_test)

pred_fit1 <- predict(fit1, newdata = x_test)
pred_fit2<- predict(fit2, newdata = x_test)

pred_RMSE <- data.frame(LM = RMSE(pred_lm, y_test),
                        SVM = RMSE(pred_svm, y_test),
                        RF = RMSE(pred_rf, y_test),
                        XGBT = RMSE(pred_xgbT, y_test),
                        F1 = RMSE(pred_fit1, y_test),
                        F2 = RMSE(pred_fit2, y_test))

print(pred_RMSE)
```

```
##      LM   SVM    RF XGBT    F1    F2
## 1 0.134 0.124 0.125 0.12 0.134 0.135
```

The test prediction accuracy is not very different across the various models.

```
qplot(ctest$winpercent, pred_lm) + geom_point() + geom_smooth(method = lm)
```




```
print(x_test[which(pred_lm > 0.8),], y_test[which(pred_lm > 0.8)])
```

```
##                  chocolate fruity caramel peanutyalmondy nougat
## Snickers Crisper         1      0       1              1      0
##                  crispedricewafer hard bar pluribus sugarpercent
## Snickers Crisper                1    0   1        0        0.604
##                  pricepercent
## Snickers Crisper        0.651
```

The outlier Snickers Crisper has a high predicted value, but actually has a lower winpercent.