Statistical study of Alcohol By Volume (ABV) in Craft Beer

STAT 420, Summer 2018 - Final Project

03 August 2018

---

1. Group Information

This study is conducted by the following members from **Group #29**:

- Apoorva H Srinivasa (NetID: **apoorva6**)
- Madhukar K P (NetID: **mk30**)
- Nicholas Reinicke (NetID: **ndr3**)
- Raymond Ordona (NetID: **rordona2**)

---

2. Introduction

There are many brew enthusiasts who like to brew beer at home. To make a good beer or to improve a recipe over many tries, it is essential to capture readings for several attributes during and after the brewing process which requires one to buy costly sensors or measuring gadgets. Using *Applied Statistics with R* we wanted to check if we can analyze and predict any of the attributes of the beer and hopefully suggest cost savings by eliminating any sensors or measuring gadgets for it. Or even predict any attribute of the final product in the early stages of brewing process so that any tweaks can be made to the recipe based on the knowledge of the predictions of the final product.

Alcohol By Volume (ABV) is an important attribute of the final output from brewing process. For this project, we have taken as our main objective, to study the attributes in brewing process that affect the alcohol content in craft beers and make best **predictions** for it, essentially eliminating any sensors or measuring gadgets for same and also to get a measure of it in the early stages of brewing process without having to wait for weeks for the brewing to complete.

For this project, we are using dataset sourced from Brewer's Friend Beer Recipes available under Kaggle platform. This dataset contains information on approximately 75,000 different recipes of craft beer along with the readings of various sensors captured during the brewing process.

---

3. Methods

In this section we'll first load the raw data, perform necessary preprocessing and analyze each attributes in the dataset. We'll use visualizations like histogram and boxplots to explore and understand the domain and distribution of each attributes. We'll then define the methods that will be used to build the models and metrics that shall be used to evaluate the different models and to pick the best model. And finally we'll implement different models, refine them as needed and gather metrics for them.

3.1 Data Definition and Preparation

Data Definition:

The raw dataset has a total of 23 attributes. The response variable - `ABV` is a *real* variable. Below table lists all the other attributes (predictors) and their definitions. Attributes that are of no importance for statistical study and which will not be used are also indicated here.

Now, in below table it's intuitive to notice that attributes like IDs and free-form text are straight away ignored for this study. One particular attribute which we have decided not to use and to which we would like to call upon attention is - `FG`. This attribute is captured after the brewing process and our response variable `ABV` is derived directly from it and to be inline with the objective stated in the introduction section to predict `ABV` before the completion of brewing process we have decided to not use it in our prediction model.

Sl#	Attribute	Domain	Used for Study?	Description
01	`SugarScale`	Categorical	Yes	Scale to determine the concentration of dissolved solids in wort
02	`BrewMethod`	Categorical	Yes	Various techniques for brewing (Ex: All Grain)
03	`Size(L)`	Real	Yes	Amount brewed for recipe listed
04	`OG`	Real	Yes	Specific gravity of wort before fermentation
05	`IBU`	Real	Yes	International Bittering Units
06	`Color`	Real	Yes	Color using Reference Method - light to dark ex. 40 = black
07	`BoilSize`	Real	Yes	Fluid at beginning of boil
08	`BoilTime`	Real	Yes	Time wort is boiled
09	`BoilGravity`	Real	Yes	Specific gravity of wort before the boil
10	`Efficiency`	Real	Yes	Beer mash extraction efficiency - extracting sugars from the grain during mash
11	`FG`	Real	No	Specific gravity of wort after fermentation
12	`Name`	String	No	Descriptive Name of Beer
13	`URL`	String	No	Location of recipe webpage
14	`Style`	String	No	Type of brew
15	`PitchRate`	String	No	Yeast added to fermentor (Data is not complete, has many non-numeric values)
16	`PrimaryTemp`	String	No	Temperature at fermenting stage (Data is not complete, has many non-numeric values)
17	`PrimingMethod`	String	No	Free form text detailing the method used for priming
18	`PrimingAmount`	String	No	Free form text indicating the amount of priming
19	`MashThickness`	String	No	Amount of water (Data is not complete, has many non-numeric values)
20	`BeerID`	Real	No	Record ID in the dataset
21	`StyleID`	Real	No	Numeric ID for type of brew
22	`UserId`	Real	No	Unique Id of user who provided or updated the receipe

Data Preparation:

Having gone through the definition of attributes and after cursory checks on the dataset we noticed that there are few records with nulls, `NaN`, `NA` or `N/A`. So as a next step we will pre-process the raw dataset by cleaning up unwanted columns and rows with unwanted values.

Below are the detailed tasks performed as part of data pre-processing:

- Read raw dataset
- Remove unwanted columns
- Remove rows with nulls/ `NaN` / `NA` / `N/A`
- Rename the column name `Size(L)` to `Size` for ease of reference.
- Coerce `SugarScale` and `BrewMethod` into factor variables.
- Remove spaces in the levels of categorical attributes `SugarScale` and `BrewMethod` for easy representation in model summary (specifically interaction models).
- Coerce `BoilGravity` into a numerical attribute

- Reorder the row numbers (names)
- Normalize OG measurements based on the method of measurements (See http://betatestbrewing.com/pages/plato_to_sg.html)

Below is the `R` code block for the tasks(pseudocode) described above,

```
# read raw dataset
beer_data_raw = read.csv('recipeData.csv')

# remove unwanted columns and rows with nulls/`NaN`/`NA`/`N/A`
beer_data = subset(beer_data_raw,
                   subset = beer_data_raw$ABV != "N/A" & beer_data_raw$BoilGravity != "N/A",
                   select = c("ABV", "SugarScale", "BrewMethod", "Size.L.", "OG", "IBU",
                              "Color", "BoilSize","BoilTime", "BoilGravity", "Efficiency"))

# rename column name
colnames(beer_data)[4] = "Size"

# coerce `SugarScale` and `BrewMethod` into factor variables
beer_data$SugarScale = as.factor(beer_data$SugarScale)
beer_data$BrewMethod = as.factor(beer_data$BrewMethod)

# remove spaces in the levels for factor variables
levels(beer_data$SugarScale) = gsub(" ", "", levels(beer_data$SugarScale), fixed = TRUE)
levels(beer_data$BrewMethod) = gsub(" ", "", levels(beer_data$BrewMethod), fixed = TRUE)

# coerce BoilGravity into a numerical attribute
beer_data$BoilGravity = as.numeric(beer_data$BoilGravity)

# reorder row numbers
rownames(beer_data) = c(1:nrow(beer_data))

# normailize OG for "Plato" measurements
invisible(ifelse(beer_data$SugarScale == "Specific Gravity", beer_data$OG, beer_data$OG <- 1 + ((beer_data$O
G)/ (258.6 - ((227.1*beer_data$OG)/258.2)))))
```

Number of records in the dataset before pre-processing (raw dataset) : \(73861\)

Number of records in the dataset after pre-processing (clean dataset): \(70871\)

Below are the first few records of the prepped dataset,

ABV	SugarScale	BrewMethod	Size	OG	IBU	Color	BoilSize	BoilTime	BoilGravity	Efficiency
5.48	SpecificGravity	AllGrain	21.77	1.004094	17.65	4.83	28.39	75	40	70
8.16	SpecificGravity	AllGrain	20.82	1.004203	60.65	15.64	24.61	60	72	70
6.48	SpecificGravity	AllGrain	50.00	1.004114	17.84	4.57	60.00	90	52	72
5.58	SpecificGravity	AllGrain	24.61	1.004094	40.12	8.00	29.34	70	49	79
5.36	SpecificGravity	AllGrain	20.82	1.004090	19.97	5.94	28.39	75	42	70
5.77	SpecificGravity	AllGrain	22.71	1.004114	31.63	34.76	30.28	75	44	73
8.22	SpecificGravity	AllGrain	20.82	1.004192	93.02	8.29	28.39	60	60	70

3.2 Data Exploration

In this section we examine the descriptive statistics like min, max, median, mean, etc., of numeric attributes and count of levels for categorical attributes to understand the domain, range and distribution for each. This can be helpful later in deciding if transformations need to be applied on repsone or any predictors.

Below `R` code computes the descriptive statistics described above.

```
# descriptive stats for response attribute
row_names = c("Min", "1st Quartile", "Median", "Mean", "3rd Quartile", "Max")
resp_num_stats = cbind(summary(beer_data$ABV))
colnames(resp_num_stats) = c("ABV")
rownames(resp_num_stats) = row_names
#resp_num_stats


# descriptive stats for numeric predictor attributes
pred_num_col = c("Size","OG","IBU","Color","BoilSize","BoilTime", "BoilGravity","Efficiency")
pred_num_stats = data.frame(matrix(NA, ncol=length(pred_num_col), nrow=6))
colnames(pred_num_stats) = pred_num_col
rownames(pred_num_stats) = row_names
for(i in 1:length(pred_num_col)){
  pred_num_stats[,i] = as.vector(summary(beer_data[,c(pred_num_col[i])]))
}
#pred_num_stats


# descriptive stats for categorical predictor attributes
SugarScale_lvls = as.data.frame(table(beer_data$SugarScale))
colnames(SugarScale_lvls) = c("Levels", "Frequency")
BrewMethod_lvls = as.data.frame(table(beer_data$BrewMethod))
colnames(BrewMethod_lvls) = c("Levels", "Frequency")
#SugarScale_lvls
#BrewMethod_lvls
```

Below is the descriptive statistics for response attribute `ABV`,

	ABV
Min	0.00
1st Quartile	5.07
Median	5.79
Mean	6.13
3rd Quartile	6.82
Max	54.72

Below is the descriptive statistics for numeric predictor attributes,

	Size	OG	IBU	Color	BoilSize	BoilTime	BoilGravity	Efficiency
Min	1.00	1.004	0.00	0.00	1.0	0.00	1.00	0.00
1st Quartile	18.93	1.004	23.90	5.15	21.0	60.00	42.00	65.00
Median	20.82	1.004	36.16	8.35	28.0	60.00	49.00	70.00
Mean	44.85	1.006	44.78	13.34	50.8	65.13	63.87	66.21
3rd Quartile	24.00	1.004	56.74	16.67	30.0	60.00	62.00	75.00
Max	9200.00	1.149	3409.30	50.00	9700.0	240.00	509.00	100.00

Below is the descriptive statistics for categorical attribute - `SugarScale`,

Levels	Frequency
Plato	1882
SpecificGravity	68989

Below is the descriptive statistics for categorical attribute - `BrewMethod`,

Levels	Frequency
AllGrain	47778
BIAB	11879
extract	8053
PartialMash	3161

Inferences:

From the above descriptive statistics we can make below inferences,

- Response attribute does not stretch over several orders of magnitude, so transformations (like log()) on it is unlikely to help and hence transformations can be ignored during model search.
- Similar to response attribute, none of the numeric attributes stretch over several orders of magnitude, so transformations (like log()) on them too is unlikely to help.
- Categorical attributes' levels look good and does not raise any concern.

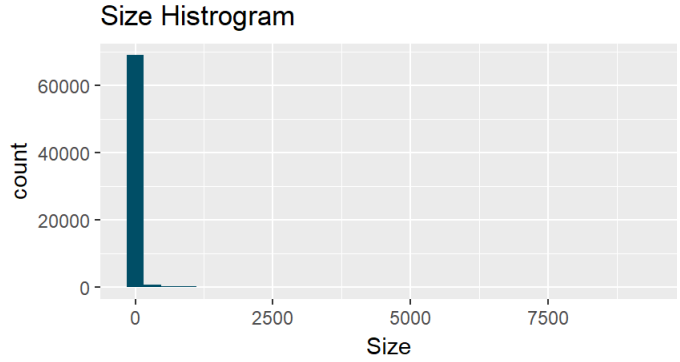
3.3 Exploratory Visualization

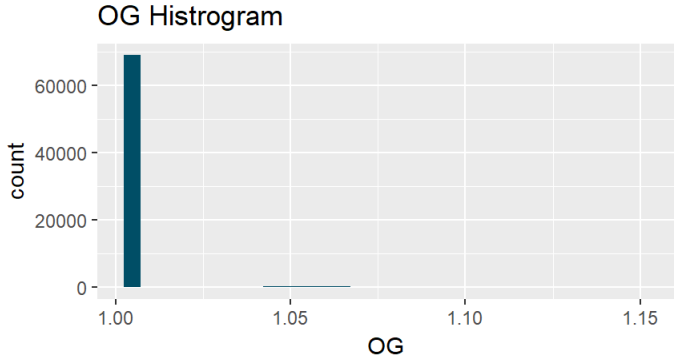
In this section we'll plot histogram for each of the attributes in the prepped data to further stress the points made in preivous section and also to see if we can catch any outliers for individual attributes and consider - if or as needed - to remove them for better prediction performance. We'll also visually check collinearity by plotting a matrix of scatterplots for every pair of attributes in the preppred dataset. This will aid in dropping some of the highly collinear attributes to reduce time during model selection procedures like backward-AIC, etc.

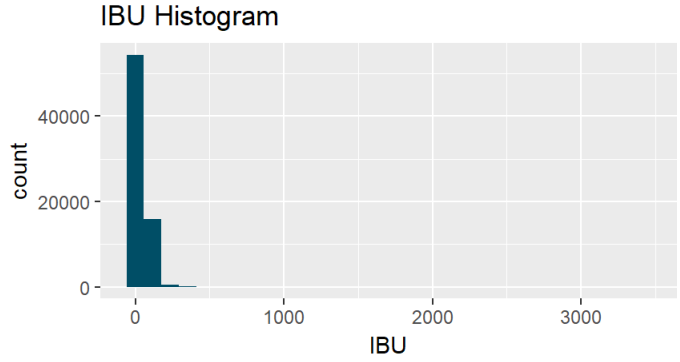
Below is the `R` code for plotting histogram for individual attributes,

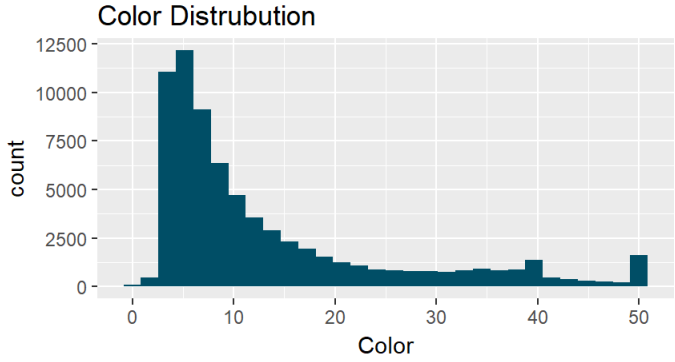
```
library(ggplot2)
library(grid)
library(gridExtra)

Size_hist = ggplot(data=beer_data, aes(beer_data$Size)) +
  geom_histogram(bins=30, fill = "#004e66") +
  labs(title = "Size Histrogram", x = "Size")

OG_hist = ggplot(data=beer_data, aes(beer_data$OG)) +
  geom_histogram(bins=30, fill = "#004e66") +
  labs(title = "OG Histrogram", x = "OG")

IBU_hist = ggplot(data=beer_data, aes(beer_data$IBU)) +
  geom_histogram(bins=30, fill = "#004e66") +
  labs(title = "IBU Histogram", x = "IBU")

Color_hist = ggplot(data=beer_data, aes(beer_data$Color)) +
  geom_histogram(bins=30, fill = "#004e66") +
  labs(title = "Color Distrubution", x = "Color")

BoilSize_hist = ggplot(data=beer_data, aes(beer_data$BoilSize)) +
  geom_histogram(bins=30, fill = "#004e66") +
  labs(title = "BoilSize Distrubution", x = "BoilSize")

BoilTime_hist = ggplot(data=beer_data, aes(beer_data$BoilTime)) +
  geom_histogram(bins=30, fill = "#004e66") +
  labs(title = "BoilTime Distrubution", x = "BoilTime")

BoilGravity_hist = ggplot(data=beer_data, aes(beer_data$BoilGravity)) +
  geom_histogram(bins=30, fill = "#004e66") +
  labs(title = "BoilGravity Distrubution", x = "BoilGravity")

Efficiency_hist = ggplot(data=beer_data, aes(beer_data$Efficiency)) +
  geom_histogram(bins=30, fill = "#004e66") +
  labs(title = "Efficiency Distrubution", x = "Efficiency")

grid.arrange(Size_hist, OG_hist, IBU_hist, Color_hist, BoilSize_hist, BoilTime_hist, BoilGravity_hist, Effic
iency_hist, nrow = 4, ncol = 2, heights = c(2.5, 2.5, 2.5, 2.5), top = textGrob("Predictor Historgrams",gp=g
par(fontsize=20,font=3)))
```

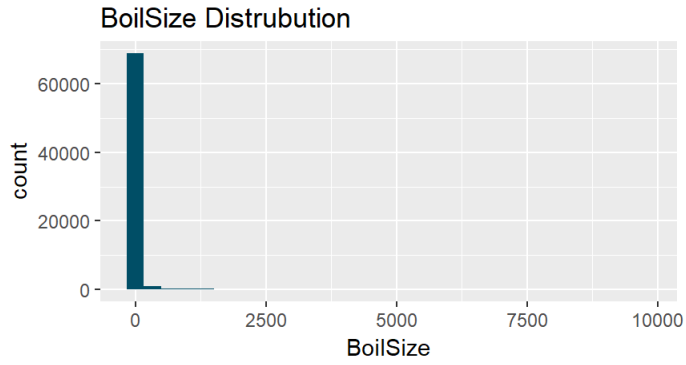
Predictor Historgrams

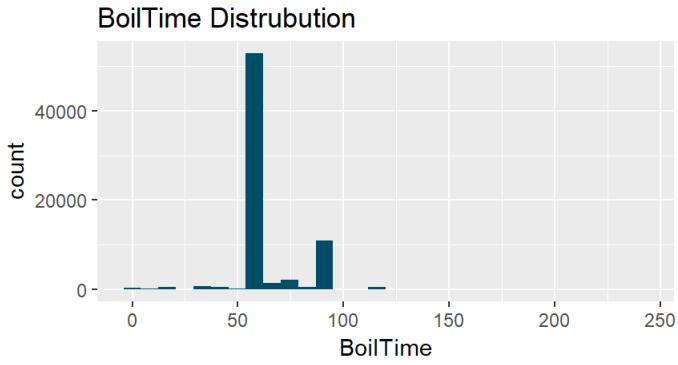


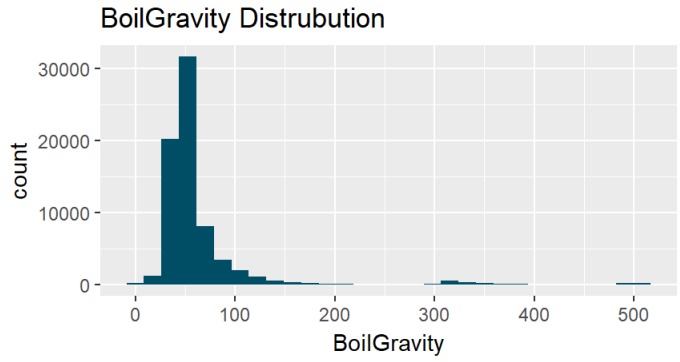


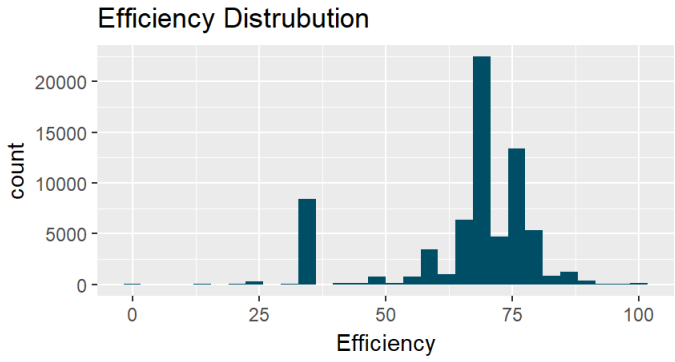




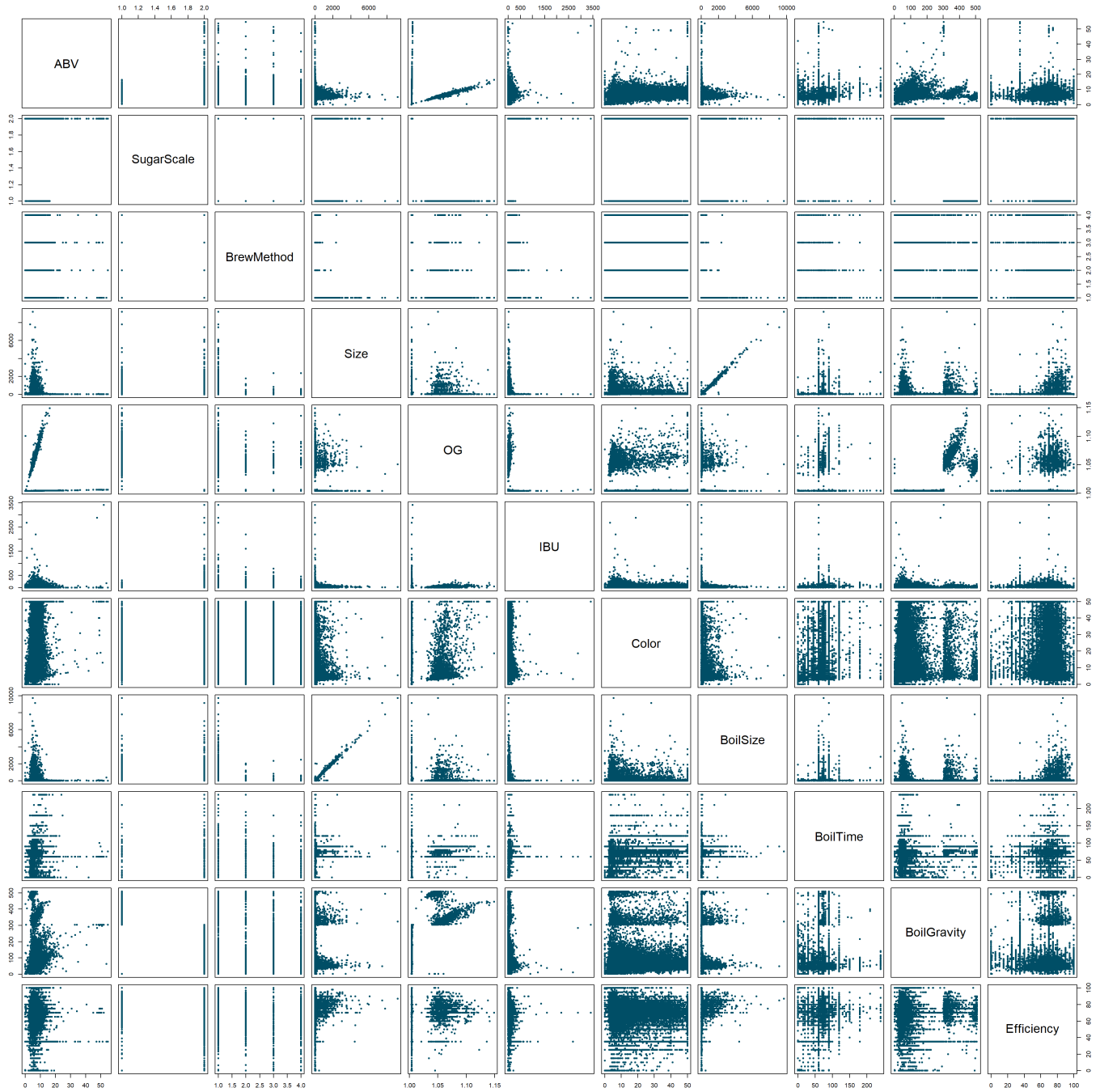









```
pairs(beer_data, col = "#004e66", pch = 20)
```



Inferences:

From the above visuals we can make below inferences,

- Looking at the histrograms, we see that certain predictors might have values we would consider as an outlier. Size has a value that is at least two orders of magnitude greater than the remainder of the data. In addition, BoilSize also has a value that appears to be at least two orders of magnitude greater than the remainder of the data. This will not likely reduce our model accuracy by a great amount but it is good to keep in mind that we may not get accurate predictions when the parameters take on those large values.
- Looking to the pairs plot, we see that BoilSize and Size are essentially correlated. This implies that we should remove at least on the values for our final predictor model. There does not appear to be any other colinearity issues implied by the pairs graph.

3.4 Methodology & Metrics

Described in this section are the high level steps that shall be taken to fit models and the metrics that shall be captured for each. Models will later be compared based on the metrics and a best one will be picked.

High Level Implemention Approach/Steps:

- Split data into train and test set; 20% of randomly picked data shall be set asside as test data and remaining 80% as test data. Models will be fit using train data and their performance on test data shall be captured. Metrics captured on the test data will mainly decide the best model.
- Fit different models using below approachs,

    - Use an additive model using all attributes as starting model and do backward search using AIC and BIC
    - Use a model with all two-way interactions along with their main effects as starting model and do backward search using AIC and BIC
    - Use a model with two-degree polynomials for all attributes along with their main effects as starting model and do backward search using AIC and BIC
- For each model capture the metrics defined in next sub-section on train and test data.
- We'll then pick the best model based on metrics and further tune or diagnose it for leverages, outliers and influencial observations.

Metrics:

Below are the metrics that shall be captured for each of the models that we fit. Argument to pick the best model will be made based on some or all of these metrics.

- RMSE on train data
- RMSE on test data
- LOOCV.RMSE
- R-squared
- Adj-R-squared
- Average percent error on train data
- Average percent error on test data
- Time taken to search the best model (in minutes)

3.5 Implementation

This section includes the detailed `R` code implementation following the approach described in previous section.

Train and Test Split:

First we'll split the dataset into train and test data. We'll set a seed and randomly pick 20% of data and set it asign as test data and will use rest 80% as the train data. Below is the `R` code for same.

```
# set seed to project group id
set.seed(29)
beer_data_trn_idx  = sample(nrow(beer_data), size = trunc(0.80 * nrow(beer_data)))
beer_data_trn = beer_data[beer_data_trn_idx, ]
beer_data_tst = beer_data[-beer_data_trn_idx, ]
```

Number of records in the train dataset: \(56696\)

Number of records in the test dataset : \(14175\)

Build generic function to fit models and compute metrics:

Now, we'll build a generic fuction which will below tasks,

- Take as inputs,
    - `strt_model` : Starting model for search
    - `dir` : Direction of search
    - `crit` : Selection metric to be used - "AIC" or "BIC"
- Then, identify the model using the search method dicated by the input parameters.
- Finally compute metrics defined in Methodology & Metrics section and return a named vector of these metrics.
    - `strt_model` : Starting model for search
    - `dir` : Direction of search
    - `crit` : Selection metric to be used - "AIC" or "BIC"

Below is the `R` code for the function just described,

```
# function that takes start model and search criteria and return the best model from search along with metri
cs for it
func_fit_get_met = function(strt_model,
                            dir = "backward",
                            crit = "AIC"){

  # identify the k value for step search
  if(crit == "AIC"){
    kk = 2
  } else if(crit == "BIC"){
    kk = log(nrow(beer_data_trn))
  }

  # Perform variable selection using the critera defined in input
  strt_time = Sys.time()
  model_bst = step(strt_model, direction = dir, trace = 0, k = kk)
  end_time  = Sys.time()

  # compute metrics and return a named vector of metrics
  return(
    list("model_bst" = model_bst,
         "metrics_v" = c("RMSE on Train Data"        = mean(resid(model_bst)^2),
                         "RMSE on Test Data"         = mean((predict(model_bst,
                                                                     newdata = beer_data_tst) - beer_data_
tst$ABV)^2),
                         "LOOCV.RMSE"                = sqrt(mean((resid(model_bst) / (1 - hatvalues(model_b
st))) ^ 2)),
                         "R-squared"                 = summary(model_bst)$r.squared,
                         "Adjusted R-squared"        = summary(model_bst)$adj.r.squared,
                         "Avg % Error on Train Data" = (1/nrow(beer_data_trn)) *
                                                        sum( abs(predict(model_bst) - beer_data_trn$ABV)
                                                             / predict(model_bst) ) * 100,
                         "Avg % Error on Test Data"  = (1/nrow(beer_data_tst)) *
                                                        sum( abs(predict(model_bst, newdata = beer_data_tst
)
                                                                 - beer_data_tst$ABV) /
                                                                predict(model_bst, newdata = beer_data_tst)
) * 100,
                         "Training Time"             = as.numeric(end_time - strt_time)
           )))
}
```

Fit different starting models and search best one for each using backward AIC and BIC:

With the generic fucntion defined, now we'll fit 3 starter models defined in the section Methodology section and search two best models using the function, one with backward AIC search and another with backward BIC search. Along they way we'll also catpure the metrics returned by the function in a dataframe for present it in results section. This will result in 6 models and metrics for each of them.

```
### Define starter models ###

# additive starter model
beer_m1_add  = lm(ABV ~ ., data = beer_data_trn)

# two-way interactions starter model
beer_m2_int2 = lm(ABV ~ (.)^2, data = beer_data_trn)

# two-degree polynomial starter model
beer_m3_ply2 = lm(ABV ~ (.) + I(Size^2) + I(OG^2) + I(IBU^2) + I(Color^2) + I(BoilSize^2) + I(BoilTime^2) +
                    I(BoilGravity^2) + I(Efficiency^2), data = beer_data_trn)
```

```
### Search best models for each using backward AIC and BIC and store the models in a list ###
beer_m1_add_bst_AIC  = func_fit_get_met(beer_m1_add, dir = "backward", crit = "AIC")
beer_m1_add_bst_BIC  = func_fit_get_met(beer_m1_add, dir = "backward", crit = "BIC")

beer_m2_int2_bst_AIC = func_fit_get_met(beer_m2_int2, dir = "backward", crit = "AIC")
beer_m2_int2_bst_BIC = func_fit_get_met(beer_m2_int2, dir = "backward", crit = "BIC")

beer_m3_ply2_bst_AIC = func_fit_get_met(beer_m3_ply2, dir = "backward", crit = "AIC")
beer_m3_ply2_bst_BIC = func_fit_get_met(beer_m3_ply2, dir = "backward", crit = "BIC")
```

```
# create list of models for easy access later
beer_models_lst = list("beer_m1_add_bst_AIC"  = beer_m1_add_bst_AIC$model_bst,
                       "beer_m1_add_bst_BIC"  = beer_m1_add_bst_BIC$model_bst,
                       "beer_m2_int2_bst_AIC" = beer_m2_int2_bst_AIC$model_bst,
                       "beer_m2_int2_bst_BIC" = beer_m2_int2_bst_BIC$model_bst,
                       "beer_m3_ply2_bst_AIC" = beer_m3_ply2_bst_AIC$model_bst,
                       "beer_m3_ply2_bst_BIC" = beer_m3_ply2_bst_BIC$model_bst)

# create a dataframe containing rows of metric results for each model for easy access later
beer_models_metrics = rbind("beer_m1_add_bst_AIC"  = beer_m1_add_bst_AIC$metrics_v,
                            "beer_m1_add_bst_BIC"  = beer_m1_add_bst_BIC$metrics_v,
                            "beer_m2_int2_bst_AIC" = beer_m2_int2_bst_AIC$metrics_v,
                            "beer_m2_int2_bst_BIC" = beer_m2_int2_bst_BIC$metrics_v,
                            "beer_m3_ply2_bst_AIC" = beer_m3_ply2_bst_AIC$metrics_v,
                            "beer_m3_ply2_bst_BIC" = beer_m3_ply2_bst_BIC$metrics_v)
```

Below is the table containing metrics for all the models found from the selection procedures.

	RMSE on Train Data	RMSE on Test Data	LOOCV.RMSE	R-squared	Adjusted R-squared	Avg % Error on Train Data	Avg % Error on Test Data	Training Time
beer_m1_add_bst_AIC	2.2579	2.0668	1.5053	0.3725	0.3723	15.503	15.633	0.2518
beer_m1_add_bst_BIC	2.2579	2.0668	1.5053	0.3725	0.3723	15.503	15.633	0.2473
beer_m2_int2_bst_AIC	0.1931	0.1894	0.4414	0.9463	0.9463	4.964	4.790	3.0780
beer_m2_int2_bst_BIC	**0.1935**	**0.1892**	**0.4409**	**0.9462**	**0.9462**	**4.968**	**4.921**	**3.6708**
beer_m3_ply2_bst_AIC	1.9603	1.7650	1.4194	0.4552	0.4550	14.015	14.185	1.9327
beer_m3_ply2_bst_BIC	1.9608	1.7723	1.4170	0.4550	0.4549	14.030	14.199	4.2443

In this table, the best model is highlighted in green, we'll revisit this again in results and discussions section.

3.6 Model Refinement

In this section we'll try to diagnorse the influential observations for the best model that was picked in previous section and see if removing the outliers or influential observations or handling highly collinear attributes makes any improvements in the model.

Outliers: In below `R` code we are computing standardized residuals and finding outliers using `3 standard deviations` as the cutt-off; that is any observations whose standardized residuals are `3 standard deviations` will be considered outliers.

```
# compute standardized residuals
outlr_cnt = sum(abs(rstandard(beer_m2_int2_bst_BIC$model_bst)) > 3)
outlr_cnt
```

```
## [1] 590
```

We see that there are \(590\) outliers. We'll now try to fit a model without the influential observations and see if the resulting model does any better; we'll comparae the metrics for the model before and after removing the influential observations.

```
# fit starter model without the influential observations
beer_m2_int2_outlr  = lm(ABV ~ (.)^2,
                         data = beer_data_trn,
                         subset = abs(rstandard(beer_m2_int2_bst_BIC$model_bst)) <= 3)

# do backward AIC search and find best model
beer_m2_int2_bst_BIC_outlr  = func_fit_get_met(beer_m2_int2_outlr, dir = "backward", crit = "BIC")

# store the before and after metrics in a dataframe
metrics_bfr_afr_outlr = rbind("before" = beer_models_metrics[4,],
                         "after"  = beer_m2_int2_bst_BIC_outlr$metrics_v)
```

Below table indicates the metrics of the model before and after removing the outlier observations. We can see that removing the outliers didn't do any better, so we'll stick to the best model found with the outliers. We'll revisit this and argue further in results and discussions

section.

	RMSE on Train Data	RMSE on Test Data	LOOCV.RMSE	R-squared	Adjusted R-squared	Avg % Error on Train Data	Avg % Error on Test Data	Training Time
before	**0.1935**	**0.1892**	**0.4409**	**0.9462**	**0.9462**	**4.968**	**4.921**	**3.671**
after	0.1341	0.1916	0.3665	0.9551	0.9551	31.552	4.926	3.612

Influential Observations:

In below `R` code we are finding the influential observations using cooks distance.

```
# compute cooks distance and find the count of records that are highly influential
influ_cnt = sum(cooks.distance(beer_m2_int2_bst_BIC$model_bst) > 4 / length(cooks.distance(beer_m2_int2_bst_
BIC$model_bst)))
```

We see that there are \(2112\) influential observations. We'll now try to fit a model without the influential observations and see if the resulting model does any better; we'll comparae the metrics for the model before and after removing the influential observations.

```
# fit starter model without the influential observations
beer_m2_int2_infl   = lm(ABV ~ (.)^2,
                         data = beer_data_trn,
                         subset = cooks.distance(beer_m2_int2_bst_BIC$model_bst) < 4 /
                                  length(cooks.distance(beer_m2_int2_bst_BIC$model_bst)))

# do backward AIC search and find best model
beer_m2_int2_bst_BIC_infl  = func_fit_get_met(beer_m2_int2_infl, dir = "backward", crit = "BIC")

# store the before and after metrics in a dataframe
metrics_bfr_afr_infl = rbind("before" = beer_models_metrics[4,],
                         "after"  = beer_m2_int2_bst_BIC_infl$metrics_v)
```

Below table indicates the metrics of the model before and after removing the influential observations. We can see that removing the influential observations didn't do any better, so we'll stick to the best model found with the influential observations. We'll revisit this again in results and discussions section.

	RMSE on Train Data	RMSE on Test Data	LOOCV.RMSE	R-squared	Adjusted R-squared	Avg % Error on Train Data	Avg % Error on Test Data	Training Time
before	**0.1935**	**0.1892**	**0.4409**	**0.9462**	**0.9462**	**4.968**	**4.921**	**3.671**
after	0.1261	0.1916	0.3552	0.9481	0.9481	31.433	4.926	3.632

Collinearity:

Under exploratory data visualization section we mentioned that `BoilSize` and `Size` are highly colinear. So now we'll remove one of these attributes - say `BoilSize` - and see if it results in a better model; we'll comparae the metrics for the model before and after removing the influential observations.

```
# fit starter model without the influential observations
beer_m2_int2_colnr   = lm(ABV ~ (.-BoilSize)^2, data = beer_data_trn)

# do backward AIC search and find best model
beer_m2_int2_bst_BIC_colnr  = func_fit_get_met(beer_m2_int2_colnr, dir = "backward", crit = "BIC")

# store the before and after metrics in a dataframe
metrics_bfr_afr_colnr = rbind("before" = beer_models_metrics[4,],
                         "after"  = beer_m2_int2_bst_BIC_colnr$metrics_v)
```

Below table indicates the metrics of the model before and after handling highly collinear attributes and we can see that this didn't result in a better model what we already had, so we'll stick to the best model found with the influential observations. We'll discuss this futher in results and discussions section.

	RMSE on Train Data	RMSE on Test Data	LOOCV.RMSE	R-squared	Adjusted R-squared	Avg % Error on Train Data	Avg % Error on Test Data	Training Time

	RMSE on Train Data	RMSE on Test Data	LOOCV.RMSE	R-squared	Adjusted R-squared	Avg % Error on Train Data	Avg % Error on Test Data	Training Time
before	**0.1935**	**0.1892**	**0.4409**	**0.9462**	**0.9462**	**4.968**	**4.921**	**3.671**
after	0.1937	0.1893	0.4410	0.9462	0.9461	4.942	4.918	1.656

Finally after above diagnosis, we'll stick to the best model found without removing outliers or influential observations or collinear attributes. Also interested readers go check Appendix - A for assumptions diagnostics for the best model.

Below is the summary of our final (best) model, it has \(35\) \(\beta\) parameters!

```
beer_model_best = beer_m2_int2_bst_BIC$model_bst
summary(beer_model_best)
```

```
##
## Call:
## lm(formula = ABV ~ SugarScale + BrewMethod + Size + OG + IBU +
##     Color + BoilSize + BoilTime + BoilGravity + Efficiency +
##     SugarScale:OG + SugarScale:IBU + SugarScale:Color + SugarScale:BoilGravity +
##     SugarScale:Efficiency + BrewMethod:BoilTime + Size:OG + Size:Efficiency +
##     OG:BoilSize + OG:BoilTime + OG:Efficiency + IBU:Color + IBU:BoilTime +
##     IBU:BoilGravity + Color:BoilSize + Color:BoilGravity + Color:Efficiency +
##     BoilSize:Efficiency + BoilTime:BoilGravity + BoilTime:Efficiency,
##     data = beer_data_trn)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -19.494  -0.226  -0.034   0.244   6.386
##
## Coefficients:
##                                      Estimate Std. Error t value
## (Intercept)                         -1.11e+02   5.81e+00  -19.09
## SugarScaleSpecificGravity           -2.62e+04   3.78e+01 -691.27
## BrewMethodBIAB                       4.46e-02   2.33e-02    1.92
## BrewMethodextract                    4.29e-01   3.46e-02   12.40
## BrewMethodPartialMash                1.13e-01   5.00e-02    2.26
## Size                                 1.04e-02   3.30e-03    3.16
## OG                                   1.09e+02   5.51e+00   19.85
## IBU                                  3.19e-03   5.54e-04    5.75
## Color                                7.76e-03   2.62e-03    2.96
## BoilSize                            -9.57e-03   3.06e-03   -3.13
## BoilTime                            -1.56e-01   2.14e-02   -7.28
## BoilGravity                          1.69e-03   2.72e-04    6.23
## Efficiency                           3.08e-01   7.83e-02    3.93
## SugarScaleSpecificGravity:OG         2.61e+04   3.77e+01  691.36
## SugarScaleSpecificGravity:IBU       -4.83e-03   5.00e-04   -9.67
## SugarScaleSpecificGravity:Color     -1.35e-02   2.04e-03   -6.63
## SugarScaleSpecificGravity:BoilGravity  6.78e-04   1.95e-04    3.47
## SugarScaleSpecificGravity:Efficiency  -1.37e-02   4.03e-03   -3.40
## BrewMethodBIAB:BoilTime              9.86e-05   3.51e-04    0.28
## BrewMethodextract:BoilTime          -7.46e-03   5.61e-04  -13.29
## BrewMethodPartialMash:BoilTime      -2.38e-03   7.98e-04   -2.99
## Size:OG                             -1.41e-02   3.34e-03   -4.22
## Size:Efficiency                      4.59e-05   9.39e-06    4.88
## OG:BoilSize                          1.31e-02   3.10e-03    4.23
## OG:BoilTime                          1.61e-01   2.14e-02    7.51
## OG:Efficiency                       -2.87e-01   7.42e-02   -3.88
## IBU:Color                            6.36e-05   3.84e-06   16.56
## IBU:BoilTime                         2.60e-05   2.93e-06    8.86
## IBU:BoilGravity                     -8.29e-06   6.19e-07  -13.39
## Color:BoilSize                       2.99e-06   8.01e-07    3.73
## Color:BoilGravity                   -3.88e-05   5.54e-06   -7.01
## Color:Efficiency                    -4.95e-05   1.19e-05   -4.15
## BoilSize:Efficiency                 -4.40e-05   8.95e-06   -4.92
## BoilTime:BoilGravity                -2.00e-05   3.48e-06   -5.76
## BoilTime:Efficiency                 -8.24e-05   1.05e-05   -7.86
##                                     Pr(>|t|)
## (Intercept)                          < 2e-16 ***
## SugarScaleSpecificGravity            < 2e-16 ***
```

```
## BrewMethodBIAB                          0.05532 .
## BrewMethodextract                       < 2e-16 ***
## BrewMethodPartialMash                   0.02404 *
## Size                                    0.00157 **
## OG                                      < 2e-16 ***
## IBU                                     8.7e-09 ***
## Color                                   0.00309 **
## BoilSize                                0.00178 **
## BoilTime                                3.4e-13 ***
## BoilGravity                             4.8e-10 ***
## Efficiency                              8.4e-05 ***
## SugarScaleSpecificGravity:OG            < 2e-16 ***
## SugarScaleSpecificGravity:IBU           < 2e-16 ***
## SugarScaleSpecificGravity:Color         3.5e-11 ***
## SugarScaleSpecificGravity:BoilGravity   0.00052 ***
## SugarScaleSpecificGravity:Efficiency    0.00067 ***
## BrewMethodBIAB:BoilTime                 0.77870
## BrewMethodextract:BoilTime              < 2e-16 ***
## BrewMethodPartialMash:BoilTime          0.00283 **
## Size:OG                                 2.4e-05 ***
## Size:Efficiency                         1.1e-06 ***
## OG:BoilSize                             2.4e-05 ***
## OG:BoilTime                             5.8e-14 ***
## OG:Efficiency                           0.00011 ***
## IBU:Color                               < 2e-16 ***
## IBU:BoilTime                            < 2e-16 ***
## IBU:BoilGravity                         < 2e-16 ***
## Color:BoilSize                          0.00019 ***
## Color:BoilGravity                       2.5e-12 ***
## Color:Efficiency                        3.3e-05 ***
## BoilSize:Efficiency                     8.7e-07 ***
## BoilTime:BoilGravity                    8.6e-09 ***
## BoilTime:Efficiency                     3.8e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.44 on 56661 degrees of freedom
## Multiple R-squared:  0.946,  Adjusted R-squared:  0.946
## F-statistic: 2.93e+04 on 34 and 56661 DF,  p-value: <2e-16
```

4. Results

Below is the table containing metrics for all the models found from the initial selection procedures.

Table-1

	RMSE on Train Data	RMSE on Test Data	LOOCV.RMSE	R-squared	Adjusted R-squared	Avg % Error on Train Data	Avg % Error on Test Data	Training Time
beer_m1_add_bst_AIC	2.2579	2.0668	1.5053	0.3725	0.3723	15.503	15.633	0.2518
beer_m1_add_bst_BIC	2.2579	2.0668	1.5053	0.3725	0.3723	15.503	15.633	0.2473
beer_m2_int2_bst_AIC	0.1931	0.1894	0.4414	0.9463	0.9463	4.964	4.790	3.0780
beer_m2_int2_bst_BIC	**0.1935**	**0.1892**	**0.4409**	**0.9462**	**0.9462**	**4.968**	**4.921**	**3.6708**
beer_m3_ply2_bst_AIC	1.9603	1.7650	1.4194	0.4552	0.4550	14.015	14.185	1.9327
beer_m3_ply2_bst_BIC	1.9608	1.7723	1.4170	0.4550	0.4549	14.030	14.199	4.2443

Below table contains metrics captured before and after removing the outliers for the best model picked from Table-1 above.

Table-2

	RMSE on Train Data	RMSE on Test Data	LOOCV.RMSE	R-squared	Adjusted R-squared	Avg % Error on Train Data	Avg % Error on Test Data	Training Time
before	**0.1935**	**0.1892**	**0.4409**	**0.9462**	**0.9462**	**4.968**	**4.921**	**3.671**
after	0.1341	0.1916	0.3665	0.9551	0.9551	31.552	4.926	3.612

Below table contains metrics captured before and after removing the influential observations for the best model picked from Table-1 above.

Table-3

	RMSE on Train Data	RMSE on Test Data	LOOCV.RMSE	R-squared	Adjusted R-squared	Avg % Error on Train Data	Avg % Error on Test Data	Training Time
before	**0.1935**	**0.1892**	**0.4409**	**0.9462**	**0.9462**	**4.968**	**4.921**	**3.671**
after	0.1261	0.1916	0.3552	0.9481	0.9481	31.433	4.926	3.632

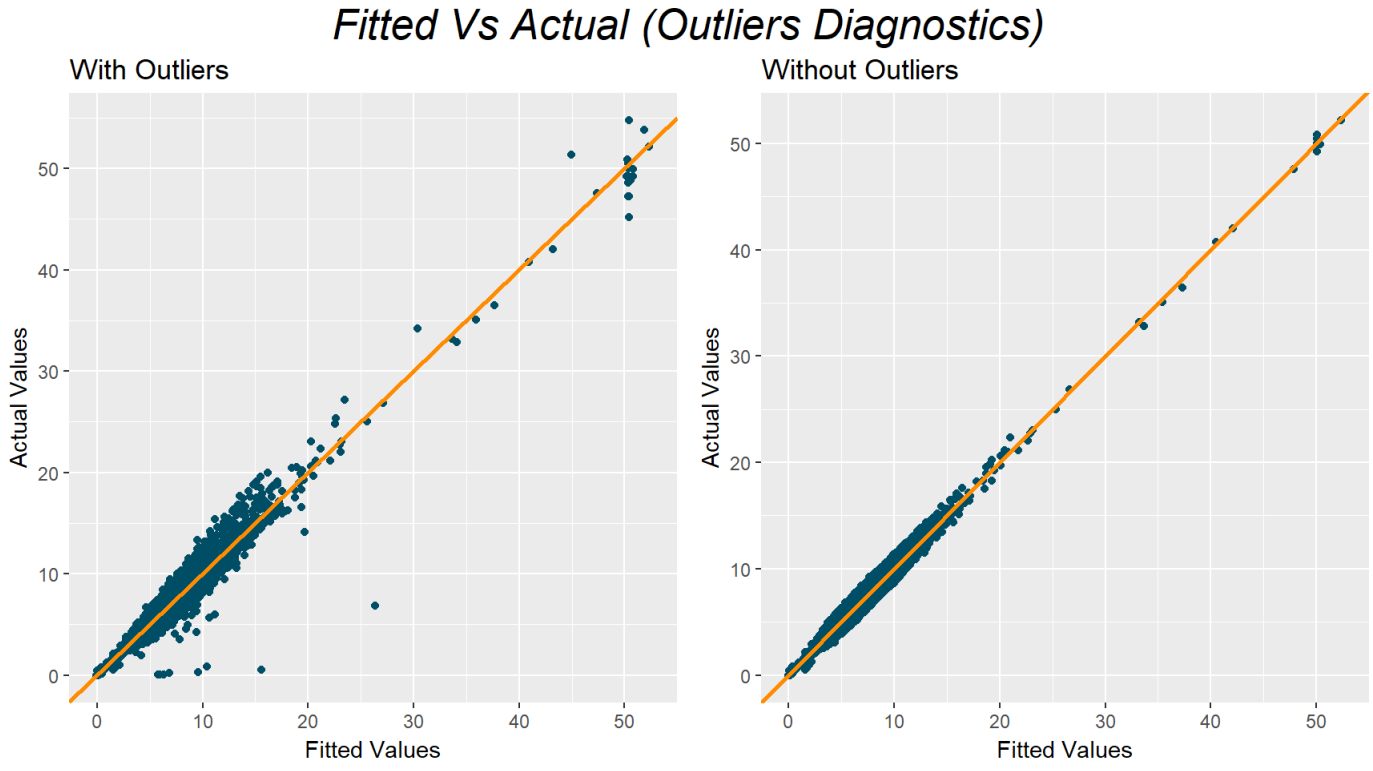
Below table contains metrics captured before and after removing collinear attributes for the best model picked from Table-1 above.

Table-4

	RMSE on Train Data	RMSE on Test Data	LOOCV.RMSE	R-squared	Adjusted R-squared	Avg % Error on Train Data	Avg % Error on Test Data	Training Time
before	**0.1935**	**0.1892**	**0.4409**	**0.9462**	**0.9462**	**4.968**	**4.921**	**3.671**
after	0.1937	0.1893	0.4410	0.9462	0.9461	4.942	4.918	1.656

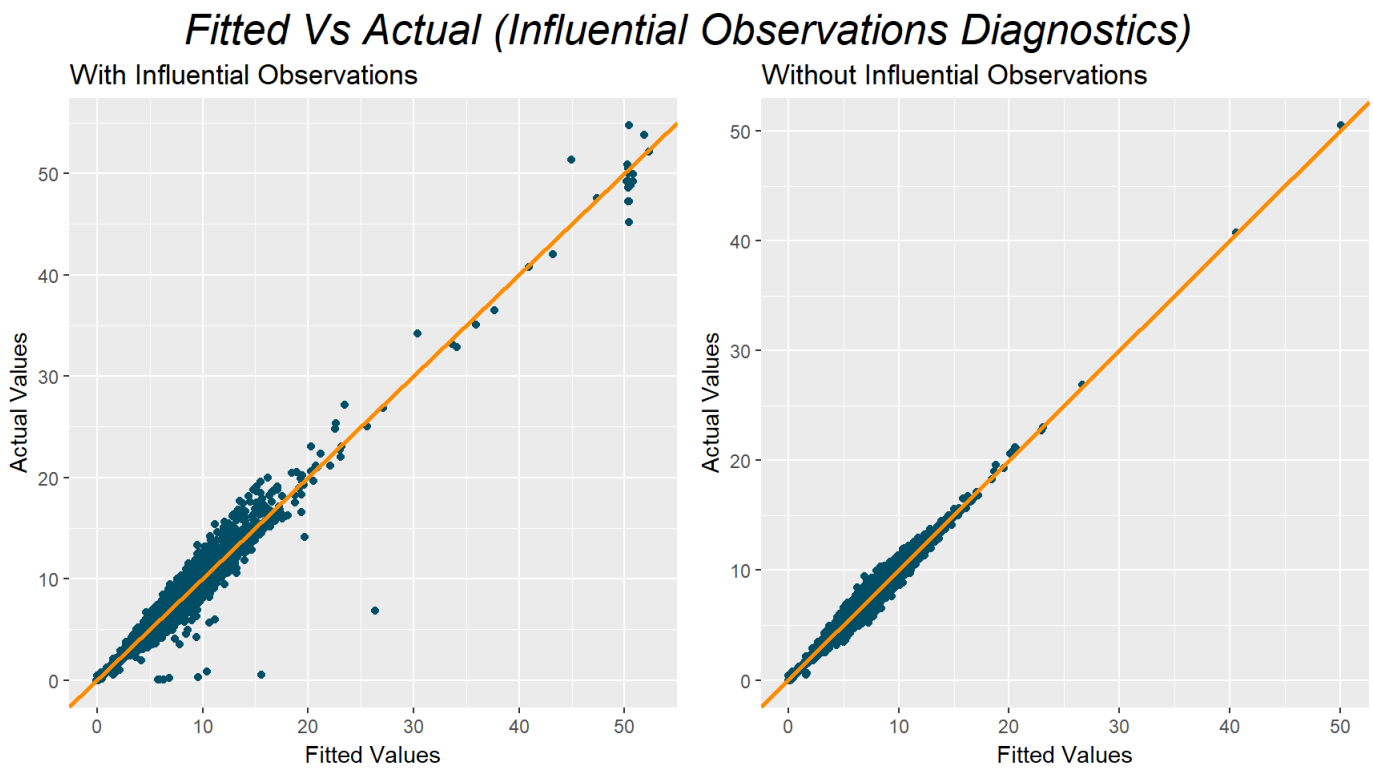
Below is the Actual Vs Predicted scatter plot for the best model fit with and wihout outlier observations,

Fig-1



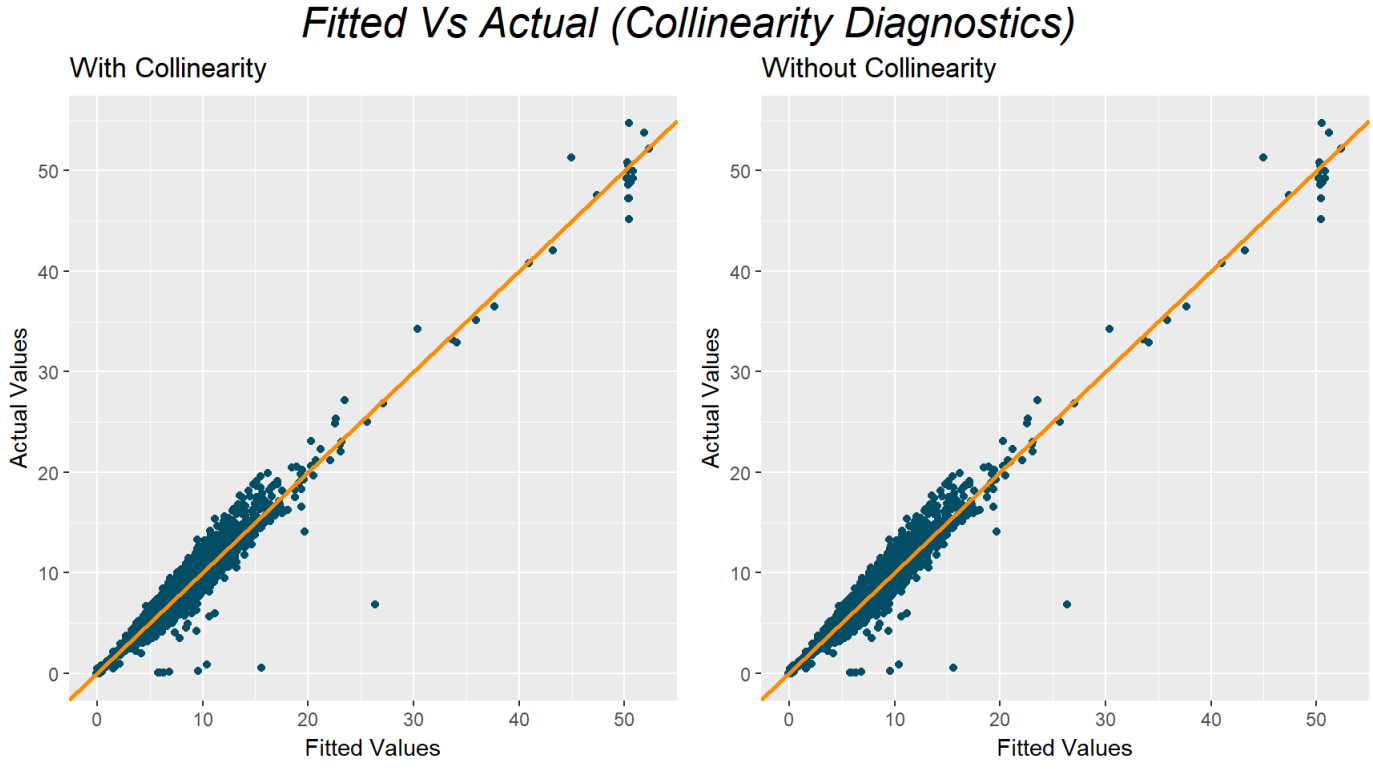
Below is the Actual Vs Predicted scatter plot for the best model fit with and wihout influential observations,

Fig-2



Below is the Actual Vs Predicted scatter plot for the best model fit with and wihout collinear attributes,

Fig-3



5. Discussion

In this section we'll reflect on the results of our study and draw inferences and conclusion from them.

5.1 Reflection on Results

Initial Model Hunt:

For this study, we started with three model structures - an additive model, a two-way interactions model and a two-dgree polynomial one.

We then did a backward search using AIC and BIC on all the three starter models and arrived at a best model for each. This search method resulted in 6 models. Table-1 under the results section indicates the various metrics for each of these models. In this table we can see that the additive models (first two rows) performed the worst. Compared to other two starter model, it produced highest RMSE on test data, highest average percent error and lowest R^2 and $Adjusted\text{ }R^2$ values. So we straight way rejected the models resulting from additve starter model AIC/BIC search. Similary, the two-degree polynomial models (last two rows) performed poorly, producing metrics close to that of additive models. Two-way interactions model (middle two rows) on the other hand performed the best, it gave very low RMSE of around 0.19 on test data indicating that it's prediction was roughly off by only 0.19 which is very low error percent for an ABV value of say 5. It achieved high R^2 and $Adjusted\text{ }R^2$ values; both the interaction models were able to explain around 95% of variations in the reponse attribute `ABV`. Other metrics of these two models were also acceptable compared to additive and polynomial models; like low LOOCV.RMSE and low average percent error on test data. The low RMSE on test data also suggested that these two interaction models doesn't suffer from over-fitting or under-fitting problems. So, the two interaction models were short listed and picking the best among them was bit hard as most of their metrics were pretty close. The average percent error of two-way interaction model resulting from AIC search (row number 3) resulted in a very high average percent error on train data, even worse than the additive and polynomial models. This raised suspicion and suggested that this model could have some drawbacks which was not exposed by the test data. Given this suspicion on row number 3 model and the fact that the model from row 4 produced least LOOCV.RMSE and is also a smaller model (because of BIC search) we decided to pick the ***interaction model from BIC search as the best model (row number 4)***

Model Tuning/Further-Diagnostics:

After our inital model search indicated that interactions clearly were significant, as a next step we did some further analsyis on our best interaction model so see if there are any oppurtunities to tune the model for improved performance. We did three kinds of diagnositcs,

- Outlier Diagnostics: First, anlaysis included check for outliers. Here we identified the outliers using standardized residuals and re-fit the model without them. Table-2 under results section indicates the metrics of the interaction model before and after removing the outliers. Here we can see that removing outliers didn't help, the RMSE on train data dropped and that on test data increased suggesting over-fit problem. The "Fitted vs Actual" plot of Fig-1 also reflects the same. In this figure the overfitting problem in the second graph is indicated by the points there being slightly more closer to the line in comparison to those in the first graph. If only the test RMSE metric also had dropped the structure of second graph would have indicated an improvement. As a result of this diagnositics we rejected the model fit without the outliers.
- Influential Observations Diagnostics: Next, we analyzed for influential observations. Here we identified the influential observations and re-fit the model without them. Table-3 under results section indicates the metrics of the interaction model before and after removing the influential observations. Similar to Outlier Diagnostics we can see that removing influential observations too didn't help and the model suffered from over-fitting problem reflected by the reduced RMSE on train data and increased RMSE on test data. "Fitted vs Actual" plot of Fig-2 is very close to that of Fig-1 and so suggests similar inferences. As a result of this diagnositics we rejected the model fit without the influential observations
- Collinearity Diagnostics: Finally, we tried to handle highly collinear attributes - `BoilSize` and `Size` - to see if it is throwing off the estimates for any β parameters and so removing any one of them would help improve the model. Here we re-fit the model without `BoilSize` and captured metrics for it. Table-4 under results section indicates the metrics of the interaction model before and after handling collinearity issue. Metrics in this table clearly indicates that there was no improvement; most of the metrics before and after were pretty close. This is further stressed by the "Fitted vs Actual" plot of Fig-3 where the two graphs look very identical to each other. This confirmed the fact that for the purpose of "prediction" collinearity is not an issue. Hence, at the end of this diagnositc as well we decided to stick with our best model picked from Table-1.

5.2 Conclusion

Our final model was able to explain almost 95% of variations in `ABV` which is impressive and also resulted in RMSE of around 0.19 on unseen data. Based on this we conclude that our model does a pretty good job in predicting `ABV` using just the attributes from initial brewing process and so beer enthusiats can use this model to get a good estimation of their final product in the initial stages of brewing process itself and also save cost by doing away with any gadgets usually used to measure `ABV`.

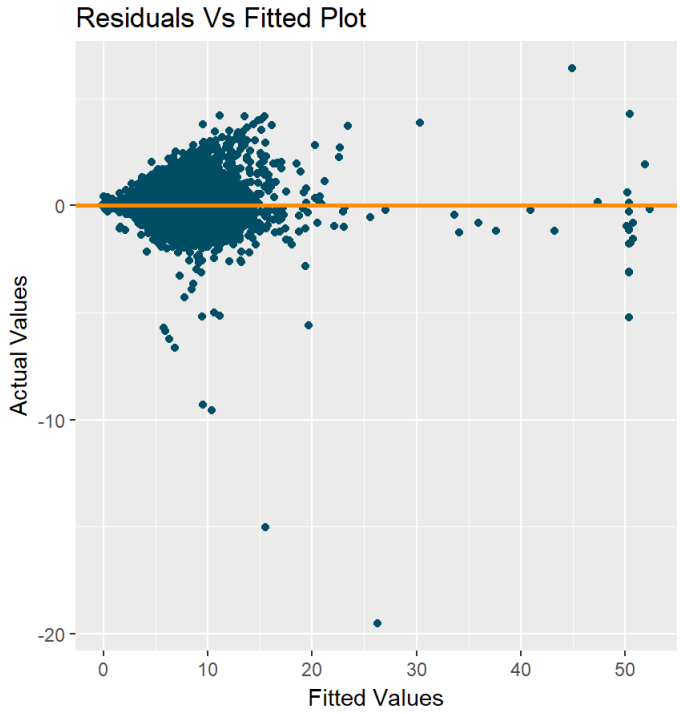
---

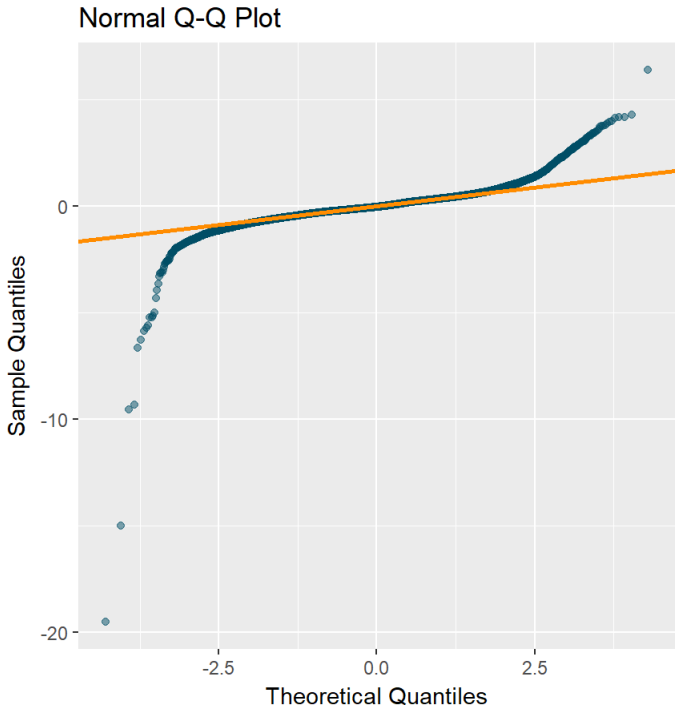
6. Appendix

6.1 Appendix - A

If the main objective of statistical study is to produce a "prediction" model then confirmity to LINE assumptions is of little importance, nevertheless in this section we have presented the assumptions diagnostics on our best model for interested users.

Below is the "Residuals Vs Fitted Plot" and "Normal Q-Q Plot",





From above plots we can say that,

- Linearity assumption is not violated as for any fitted value the mean of the residuals are roughly around 0 in the fitted vs residuals plot.
- Constant variance assumption is violated as spread of residuals across fitted lines is not constant in the fitted vs residuals plot.
- In the Q-Q plot we can see that it has thick edges but the majority of the points are on the line, so we would say that our model comes close in confirming to normality assumption.

6.2 Appendix - B

Just an extra, for fun, what if we brew our own beer at home, but want to only serve 4-6 ABV contents for occasional beer drinkers?

Let's see,

```
test_for_occasions = beer_data_tst[ which ( beer_data_tst$ABV <= 6 & beer_data_tst$ABV >=4 ),]

predicted_occasions = predict(beer_model_best, test_for_occasions, type="response")
actual_occasions = test_for_occasions$ABV

# RMSE
(rmse = sqrt(mean((predicted_occasions - actual_occasions)^2)))
```

```
## [1] 0.3287
```

That's so much better. We should be able to brew beer, predicting an expected alcohool content with an error of roughly 0.3287 if we focus our beer making around the 4-6 ABV.

6.3 Appendix - C

Can we serve drinkers with advance taste for beer?

Here, we're assuming they're looking for stronger beer above 6 ABV perhaps. Let's see:

```
test_for_hard = beer_data_tst[ which ( beer_data_tst$ABV > 6 ),]

predicted_hard = predict(beer_model_best, test_for_hard, type="response")
actual_hard = test_for_hard$ABV

# RMSE
(rmse = sqrt(mean((predicted_hard - actual_hard)^2)))
```

Probably yes! We will have an error of roughly 0.3287 which is small and acceptable.