Project 2 in R

DATA 624-01 Group 3

Z. Palmore, K. Popkin, K. Potter, C. Nan, J. Ramalingam

7/8/2021

Table of Contents

# Summary

# Data Exploration

Our data consists of 33 variables from a food and beverage manufacturing company. Our goal is to predict the hydronium ion concentration, or pH, which is a measure of acidity or alkalinity. This ‘PH’ measure, as shown in the data, is a Key Performance Indicator (KPI) and must conform to a critical range.

## Initial Observations

The data is loaded, and all packages used throughout the report are provided. We view a random sample of the data to inspect initial observations. It shows the randomly selected data as 1 -5 denoting the first, second, third, fourth, and fifth observation for each variable.

library(utils)  
library(psych)  
library(tidyverse)  
library(corrplot)  
library(ggcorrplot)  
library(ggpubr)  
library(MASS)  
library(mice)  
library(VIM)  
library(caret)  
library(zoo)  
library(naniar)  
library(partykit)  
library(flextable)  
library(bestNormalize)  
theme\_set(theme\_minimal())  
set.seed(004)  
ph <- read.csv(  
 "https://raw.githubusercontent.com/palmorezm/msds/main/624/Projects/Project2/StudentData%20-%20TO%20MODEL.csv")  
obs.sample <- as.data.frame(t(head(sample\_n(ph[1:33], 5), 5)))  
obs.sample <- as.data.frame(lapply(obs.sample[2:33,],   
function(x) round(as.numeric(as.character(x)),1)))  
obs.sample <- rbind(ph$ï..Brand.Code, obs.sample)  
obs.sample <- obs.sample %>%   
 mutate(Variable = colnames(ph)) %>%  
 dplyr::select(Variable, X1, X2, X3, X4, X5)  
flextable(obs.sample) %>%   
 theme\_vanilla() %>%   
 set\_table\_properties(layout = "autofit")

| **Variable** | **X1** | **X2** | **X3** | **X4** | **X5** |
| --- | --- | --- | --- | --- | --- |
| ï..Brand.Code | B | A | B | A | A |
| Carb.Volume | 5 | 5.4 | 5.2 | 5.3 | 5.3 |
| Fill.Ounces | 24.2 | 24.1 | 24 | 24 | 24.1 |
| PC.Volume | 0.1 | 0.3 | 0.2 | 0.2 | 0.3 |
| Carb.Pressure | 61.2 | 68.4 | 63 | 75.4 | 66.2 |
| Carb.Temp | 139.2 | 141.8 | 135.4 | 152.2 | 138.8 |
| PSC | 0 | 0.1 | 0.2 | 0.1 | 0.1 |
| PSC.Fill |  | 0.2 | 0.4 | 0.3 | 0.1 |
| PSC.CO2 | 0.1 | 0 | 0.2 | 0 | 0.1 |
| Mnf.Flow | 0.2 | -100.2 | 0.2 | -100 | -100.2 |
| Carb.Pressure1 | 122.4 | 111.8 | 125.8 | 117.6 | 114.4 |
| Fill.Pressure |  | 46 | 43.8 | 46.6 | 46.2 |
| Hyd.Pressure1 | 0 | 0 | 0.2 | 0 | 0 |
| Hyd.Pressure2 | 0.2 | 0 | 0.2 | 0 | 0 |
| Hyd.Pressure3 | -1.2 | 0 | -1.2 | 0 | 0 |
| Hyd.Pressure4 | 98 | 74 | 126 | 88 | 94 |
| Filler.Level |  | 120 | 128.2 | 119.4 | 120 |
| Filler.Speed | 1010 | 4010 | 1010 | 4010 | 3978 |
| Temperature | 67 | 64.4 | 71.6 | 66 | 65.4 |
| Usage.cont | 24.4 | 16.8 | 23.8 | 19.2 | 16.7 |
| Carb.Flow | 3014 | 2980 | 40 | 3110 | 2984 |
| Density | 0.8 | 1.7 | 0.7 | 0.9 | 0.9 |
| MFR | 466.4 | 732.6 |  | 730.6 | 726.8 |
| Balling | 1.3 | 3.4 | 0.8 | 1.4 | 1.3 |
| Pressure.Vacuum | -5 | -5 | -5.6 | -4.4 | -4.6 |
| PH | 8.5 | 8.8 | 8.5 | 8.6 | 8.8 |
| Oxygen.Filler | 0 | 0 | 0 | 0 | 0.1 |
| Bowl.Setpoint | 90 | 120 | 90 | 120 | 120 |
| Pressure.Setpoint | 48 | 46 | 50 | 46 | 46 |
| Air.Pressurer | 142.4 | 142.4 | 142.6 | 146.6 | 142.6 |
| Alch.Rel | 6.5 | 7.1 | 6.5 | 6.5 | 6.5 |
| Carb.Rel | 5.5 | 5.5 | 5.2 | 5.3 | 5.4 |
| Balling.Lvl | 1.6 | 3.3 | 1.4 | 1.4 | 1.5 |

From these initial observations, we notice that data is missing. These appear as empty spots on the table. Some data points are also zero or negative but is unclear if this is intentional. For example, variable ‘Pressure.Vacuum’ is negative as one might expect when holding pressure in vacuum, but the variables ‘Mnf.Flow,’ ‘PSC,’ ‘Hyd.Pressure1,’ ‘Hyd.Pressure2,’ ‘Hyd.Pressure3,’ and others are less intuitive.

Our table also displays differences the in scale of the variable’s raw values. This will need to be reviewed to ensure it does not harm the model’s predictive capabilities. It also appears that due to the conversion of the spreadsheet format from excel to ‘csv’ when hosting the data remotely, our first column variable name is legible but needs the “i..” symbol removed. None of the other 33 variables need to be renamed. We take a closer look at these variables with some descriptive statistics.

## Inferential Statisitcs

Our target variable, PH, appears to be on a scale between 1 and 10, as it should be, while the others vary widely between -100 (Mnf.Flow) and greater than 4000 (Filler.Speed). For this reason, we review each variable’s statistics individually. We calculate their total observations (obs), percent missing (pm), mean, median (med), standard deviation (sd), minimum value (min), maximum value (max), skewness (skew), and standard error (se) and combine them into a table by variable.

ph.desc <- ph %>%   
 describe() %>%  
 mutate("pm" = round(((2571 - n)/2571)\*100, 2),   
 obs = n,   
 med = median) %>%   
 round(digits = 1) %>%   
 mutate(var = colnames(ph)) %>%  
 dplyr::select(var, obs, pm, mean, med,  
 sd, min, max, skew, se)  
flextable::flextable(ph.desc) %>%   
 theme\_vanilla() %>%   
 set\_table\_properties(layout = "autofit")

| **var** | **obs** | **pm** | **mean** | **med** | **sd** | **min** | **max** | **skew** | **se** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ï..Brand.Code | 2,571 | 0.0 | 3.4 | 3.0 | 1.1 | 1.0 | 5.0 | 0.0 | 0.0 |
| Carb.Volume | 2,561 | 0.4 | 5.4 | 5.3 | 0.1 | 5.0 | 5.7 | 0.4 | 0.0 |
| Fill.Ounces | 2,533 | 1.5 | 24.0 | 24.0 | 0.1 | 23.6 | 24.3 | 0.0 | 0.0 |
| PC.Volume | 2,532 | 1.5 | 0.3 | 0.3 | 0.1 | 0.1 | 0.5 | 0.3 | 0.0 |
| Carb.Pressure | 2,544 | 1.0 | 68.2 | 68.2 | 3.5 | 57.0 | 79.4 | 0.2 | 0.1 |
| Carb.Temp | 2,545 | 1.0 | 141.1 | 140.8 | 4.0 | 128.6 | 154.0 | 0.2 | 0.1 |
| PSC | 2,538 | 1.3 | 0.1 | 0.1 | 0.0 | 0.0 | 0.3 | 0.8 | 0.0 |
| PSC.Fill | 2,548 | 0.9 | 0.2 | 0.2 | 0.1 | 0.0 | 0.6 | 0.9 | 0.0 |
| PSC.CO2 | 2,532 | 1.5 | 0.1 | 0.0 | 0.0 | 0.0 | 0.2 | 1.7 | 0.0 |
| Mnf.Flow | 2,569 | 0.1 | 24.6 | 65.2 | 119.5 | -100.2 | 229.4 | 0.0 | 2.4 |
| Carb.Pressure1 | 2,539 | 1.2 | 122.6 | 123.2 | 4.7 | 105.6 | 140.2 | 0.1 | 0.1 |
| Fill.Pressure | 2,549 | 0.9 | 47.9 | 46.4 | 3.2 | 34.6 | 60.4 | 0.5 | 0.1 |
| Hyd.Pressure1 | 2,560 | 0.4 | 12.4 | 11.4 | 12.4 | -0.8 | 58.0 | 0.8 | 0.2 |
| Hyd.Pressure2 | 2,556 | 0.6 | 21.0 | 28.6 | 16.4 | 0.0 | 59.4 | -0.3 | 0.3 |
| Hyd.Pressure3 | 2,556 | 0.6 | 20.5 | 27.6 | 16.0 | -1.2 | 50.0 | -0.3 | 0.3 |
| Hyd.Pressure4 | 2,541 | 1.2 | 96.3 | 96.0 | 13.1 | 52.0 | 142.0 | 0.5 | 0.3 |
| Filler.Level | 2,551 | 0.8 | 109.3 | 118.4 | 15.7 | 55.8 | 161.2 | -0.8 | 0.3 |
| Filler.Speed | 2,514 | 2.2 | 3,687.2 | 3,982.0 | 770.8 | 998.0 | 4,030.0 | -2.9 | 15.4 |
| Temperature | 2,557 | 0.5 | 66.0 | 65.6 | 1.4 | 63.6 | 76.2 | 2.4 | 0.0 |
| Usage.cont | 2,566 | 0.2 | 21.0 | 21.8 | 3.0 | 12.1 | 25.9 | -0.5 | 0.1 |
| Carb.Flow | 2,569 | 0.1 | 2,468.4 | 3,028.0 | 1,073.7 | 26.0 | 5,104.0 | -1.0 | 21.2 |
| Density | 2,570 | 0.0 | 1.2 | 1.0 | 0.4 | 0.2 | 1.9 | 0.5 | 0.0 |
| MFR | 2,359 | 8.2 | 704.0 | 724.0 | 73.9 | 31.4 | 868.6 | -5.1 | 1.5 |
| Balling | 2,570 | 0.0 | 2.2 | 1.6 | 0.9 | -0.2 | 4.0 | 0.6 | 0.0 |
| Pressure.Vacuum | 2,571 | 0.0 | -5.2 | -5.4 | 0.6 | -6.6 | -3.6 | 0.5 | 0.0 |
| PH | 2,567 | 0.2 | 8.5 | 8.5 | 0.2 | 7.9 | 9.4 | -0.3 | 0.0 |
| Oxygen.Filler | 2,559 | 0.5 | 0.0 | 0.0 | 0.0 | 0.0 | 0.4 | 2.7 | 0.0 |
| Bowl.Setpoint | 2,569 | 0.1 | 109.3 | 120.0 | 15.3 | 70.0 | 140.0 | -1.0 | 0.3 |
| Pressure.Setpoint | 2,559 | 0.5 | 47.6 | 46.0 | 2.0 | 44.0 | 52.0 | 0.2 | 0.0 |
| Air.Pressurer | 2,571 | 0.0 | 142.8 | 142.6 | 1.2 | 140.8 | 148.2 | 2.3 | 0.0 |
| Alch.Rel | 2,562 | 0.3 | 6.9 | 6.6 | 0.5 | 5.3 | 8.6 | 0.9 | 0.0 |
| Carb.Rel | 2,561 | 0.4 | 5.4 | 5.4 | 0.1 | 5.0 | 6.1 | 0.5 | 0.0 |
| Balling.Lvl | 2,570 | 0.0 | 2.1 | 1.5 | 0.9 | 0.0 | 3.7 | 0.6 | 0.0 |

No variable is missing greater than 8.2% (MFR) of its cases with 25 of the 33 variables at or under 1% missing. Our target ‘PH,’ is only missing 0.2%. A simple imputation should fix this without issue, as long as the errors are randomly dispersed.

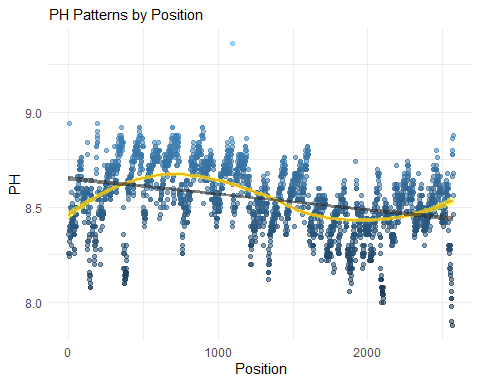
The standard deviations inform us that the spread of each variables is also wide in several cases like ‘Filler.Speed,’ and ‘Carb.Flow’ while extremely narrow or even 0 for variables, ‘Oxygen.Filler,’ ‘PSC,’ PSC.CO2,’ ‘PSC.Fill,’ ‘Carb.Volume,’ and several more. These statistics confirm that we will be working with data objects of completely different scales each centered around their own averages. Our target ‘PH,’ has very little deviation as well, but in this case, it is to be expected since by nature it follows a logarithmic pattern and we are only seeing values between 7.9 and 9.4.

There is some skewness among the variables and thereby potential outliers. This is most pronounced in the variables, ‘MFR,’ and ‘Filler.Speed,’ where the data has skewed left from its averages. Fortunately, all but two variables (Carb.Flow and Filler.Speed) have low standard errors indicating that nearly all variables will serve as good reference data sets for prediction due to low variation in spread from their means. To learn how good these variables may be at predicting, we need to see the relationship of each with our target ‘PH,’ interpret, and apply the results during modeling.

## Variable Relationships

To gain a better understanding of the relationships between our variables and the target ‘PH’ we visualize the points as scatterplots with fitted regression lines. Before we make our first plot, we create an index value for ‘PH’ in the order it was read. This shows the pattern in individual measures of acidity or alkalinity as they were taken. We call this index value the ‘Position’ given its purpose of describing measurement position for each ‘PH’ value.

ph %>%   
 dplyr::select(PH, Oxygen.Filler) %>%   
 mutate(Index = 1:length(ph$PH)) %>%   
 ggplot(aes(Index, PH, color = PH, alpha=.01)) +   
 geom\_point(aes()) +   
 geom\_smooth(method = "loess",   
 color="goldenrod2",   
 lty = "solid",   
 fill = "yellow1") +  
 geom\_smooth(method = "lm",   
 color="grey34",   
 lty = "dotted",   
 fill = "grey13") +  
 labs(subtitle = "PH Patterns by Position",  
 x = "Position", y = "PH") +  
 theme(legend.position = "none")



# Bounds estimates  
cap99 <- (length(ph$PH) - 4) / length(ph$PH)

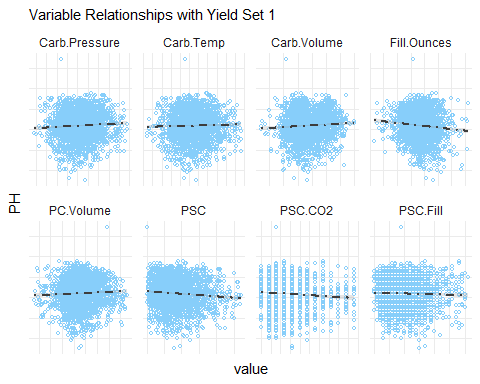
Interestingly, the pattern appears to fit an almost perfect sinusoidal curve, as demonstrated by the yellow line which is a closely fit locally weighted scatterplot smoothing (LOESS). For contrast, the black line attempts to fit a linear trend to the data. The shade of blue of each point indicates PH level and clustering. The lighter the point, the more alkaline and more solitary the measurement while darker points are more acidic and grouped together. A good example of the lightest point is the lone outlier plotted well above a pH of 9 just past position 1000.

A greater number of blue points (pH measurements) fall onto and around the yellow line than the black linear line and the pH shades do not show any big clusters or groups of pH values. From this we can be confident in two principles about this data. First, that the data measurements are random with no obvious pH clusters and secondly, that the data’s collection process was not random, it was more sinusoidal.

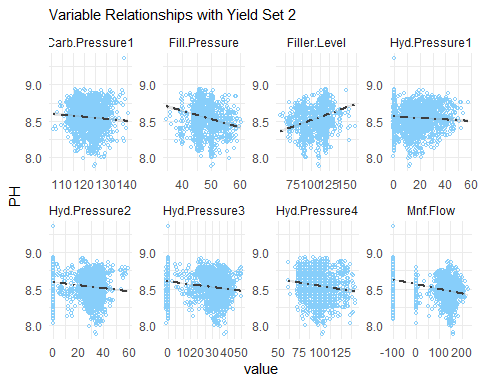
From this perspective, it is likely that there is a give and take of pH in each beverage. They are likely bounded by pH levels and the fluctuations in each beverage was accurately captured by the machine or device measuring pH. At this stage, we can also estimate the bounds as 99.84% of the measurements fall between 8.0 and 9.0 while over 75% is between 8.25 and 8.75. This should not have a tremendous effect on all models, but it would certainly affect any business decisions related to our predictions.

We continue to visualize the relationship of our target ‘PH,’ with the variables. For ease of viewing, we plot these variables in 4 sets each given a linear black line of best fit using ‘PH,’ as the response.

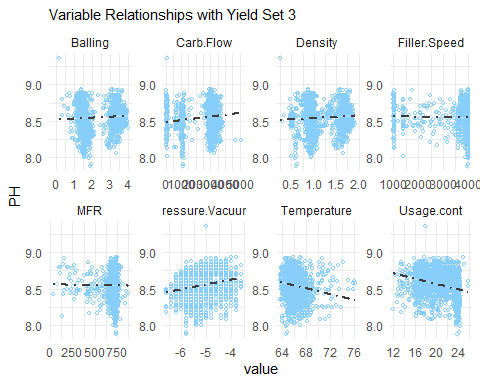
# 1st set of variables - excluding brand code (categorical)  
ph[c(2:9,26)] %>%  
 gather(variable, value, -PH) %>%   
 ggplot(., aes(value, PH)) +   
 labs(  
 subtitle =  
 "Variable Relationships with Yield Set 1") +   
 geom\_point(fill = "white",  
 size=1,   
 shape=1,   
 color="light sky blue") +   
 geom\_smooth(formula = y~x,   
 method = "lm",   
 size=1,  
 se = TRUE,  
 color = "grey24",   
 linetype = "dotdash",   
 alpha=0.25) +   
 facet\_wrap(~variable,   
 scales ="free",  
 ncol = 4) +   
 theme(axis.text.x = element\_blank(),   
 axis.text.y = element\_blank(),   
 axis.ticks = element\_blank())



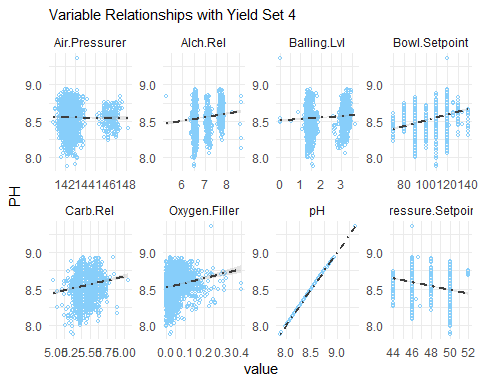
# 2nd set of variables  
ph[c(10:17,26)] %>%  
 gather(variable, value, -PH) %>%   
 ggplot(., aes(value, PH)) +   
 labs(  
 subtitle =  
 "Variable Relationships with Yield Set 2") +  
 geom\_point(fill = "white",  
 size=1,   
 shape=1,   
 color="light sky blue") +   
 geom\_smooth(formula = y~x,   
 method = "lm",   
 size=1,  
 se = TRUE,  
 color = "grey24",   
 lty = "dotdash",   
 alpha=0.25) +   
 facet\_wrap(~variable,   
 scales ="free",  
 ncol = 4)



# 3rd set of variables   
ph[c(18:26)] %>%  
 gather(variable, value, -PH) %>%   
 ggplot(., aes(value, PH)) +   
 labs(  
 subtitle =  
 "Variable Relationships with Yield Set 3") +  
 geom\_point(fill = "white",  
 size=1,   
 shape=1,   
 color="light sky blue") +   
 geom\_smooth(formula = y~x,   
 method = "lm",   
 size=1,  
 se = TRUE,  
 color = "grey24",   
 linetype = "dotdash",   
 alpha=0.25,   
 fill="white") +   
 facet\_wrap(~variable,   
 scales ="free",  
 ncol = 4)

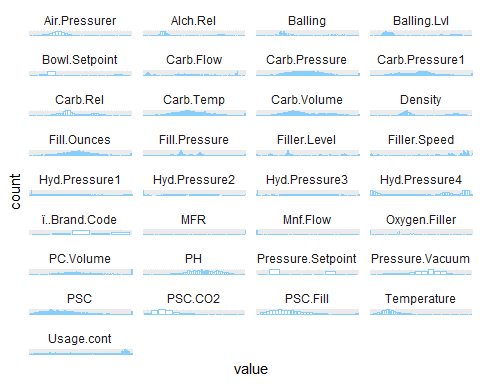


# last 8 variables  
ph[c(27:33,26)] %>%   
 mutate(pH = PH) %>%   
 gather(variable, value, -PH) %>%   
 ggplot(., aes(value, PH)) +   
 labs(  
 subtitle =  
 "Variable Relationships with Yield Set 4") +  
 labs(subtitle = ) +  
 geom\_point(fill = "white",  
 size=1,   
 shape=1,   
 color="light sky blue") +   
 geom\_smooth(formula = y~x,   
 method = "lm",   
 size=1,  
 se = TRUE,  
 color = "grey24",   
 linetype = "dotdash",   
 alpha=0.25) +   
 facet\_wrap(~variable,   
 scales ="free",  
 ncol = 4)



None of the variables appear to follow a linear relationship with ‘PH.’ If they were, it would look similar to the plot of ‘PH,’ with itself. Most variables are continuous. Excluding a handful of seemingly discrete numerical values such as ‘Pressure.Setpoint,’ ‘Bowl.Setpoint,’ and ‘Alch.Rel,’ there is an inherent randomness to all relationships with ‘PH.’ We repeat the visualization process to create histograms of each variable to review their distributions.

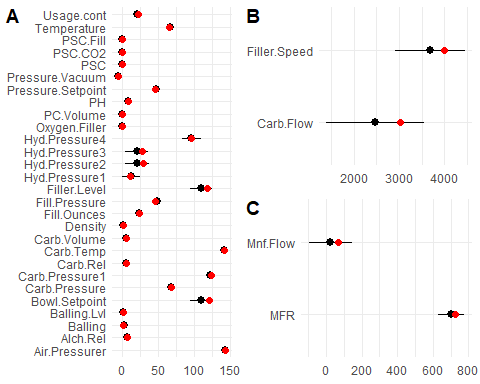
ph %>%  
 gather(variable, value) %>%   
 ggplot(., aes(value)) +   
 ggtitle("Linearity of Values") +   
 geom\_histogram(stat="count",   
 binwidth = 10,   
 fill = "white",   
 color="light sky blue") +   
 facet\_wrap(~variable,   
 scales ="free",  
 ncol = 4) +  
 theme(axis.text.x = element\_blank(),   
 axis.text.y = element\_blank(),   
 plot.title = element\_blank())



At a binwidth of 10, ‘PH’ appears normally distributed. However, many variables have two modes (biomodal) and skewed. Temperature is skewed right due to a suspected outlier. We will need to fix this before modeling.

Additionally, the discrete numeric values formed by ‘Pressure.Setpoint,’ and ‘Bowl.Setpoint,’ in our scatterplots are confirmed. They can be spotted easily this way since they look more like our categorical variable “Brand.Code” than a continuous variable like ‘Temperature.’ Given these results, we must consider skew more closely. We use a point range function and visual to explore further.

# Function to calculate and set pointrange  
xysdu <- function(x) {  
 m <- mean(x)  
 ymin <- m - sd(x)  
 ymax <- m + sd(x)  
 return(c(y = m, ymin = ymin, ymax = ymax))  
}  
# Full picture point range   
ptrng.full <- ph %>%   
 dplyr::select(where(is.numeric)) %>%   
 gather(variable, value) %>%   
 ggplot(aes(variable, value)) + coord\_flip() +  
 stat\_summary(fun.data=xysdu, geom = "Pointrange", shape=16, size=.5, color="black") +  
 stat\_summary(fun.y=median, geom="point", shape=16, size=2, color="red") +   
 theme(legend.position = "None", axis.title.x = element\_blank(), axis.title.y = element\_blank())  
# Similar point ranges (smaller)  
ptrng.small <- ph %>%   
 dplyr::select(where(is.numeric)) %>%   
 dplyr::select(-MFR, -Filler.Speed, -Carb.Flow, -Mnf.Flow) %>%   
 gather(variable, value) %>%   
 ggplot(aes(variable, value)) + coord\_flip() +  
 stat\_summary(fun.data=xysdu, geom = "Pointrange", shape=16, size=.5, color="black") +  
 stat\_summary(fun.y=median, geom="point", shape=16, size=2, color="red") +   
 theme(legend.position = "None", axis.title.x = element\_blank(), axis.title.y = element\_blank())  
# Similar point ranges (Medium)  
ptrng.med <- ph %>%   
 dplyr::select(where(is.numeric)) %>%   
 dplyr::select(MFR, Mnf.Flow) %>%   
 gather(variable, value) %>%   
 ggplot(aes(variable, value)) + coord\_flip() +  
 stat\_summary(fun.data=xysdu, geom = "Pointrange", shape=16, size=.5, color="black") +  
 stat\_summary(fun.y=median, geom="point", shape=16, size=2, color="red") +   
 theme(legend.position = "None", axis.title.x = element\_blank(), axis.title.y = element\_blank())  
# Similar point ranges (larger)  
ptrng.lrg <- ph %>%   
 dplyr::select(where(is.numeric)) %>%   
 dplyr::select(Filler.Speed, Carb.Flow) %>%   
 gather(variable, value) %>%   
 ggplot(aes(variable, value)) + coord\_flip() +  
 stat\_summary(fun.data=xysdu, geom = "Pointrange", shape=16, size=.5, color="black") +  
 stat\_summary(fun.y=median, geom="point", shape=16, size=2, color="red") +   
 theme(legend.position = "None", axis.title.x = element\_blank(), axis.title.y = element\_blank())  
ggarrange(ptrng.small,   
 ggarrange(ptrng.lrg, ptrng.med, ncol = 1, labels = c("B", "C")), nrow = 1, labels = "A"  
 )



The red dot on each variable range is its median while the black dot is its mean. The more of the black dot we see, the more skewed that distribution is. Here, we label three parts A, B, and C to denote a corresponding group of variables on similar scales with centers of their distribution closer together than other groups. It is best to simply think of these labels as A being the small-scale group, B showing the large-scale group, and C as a middle or medium scale group.

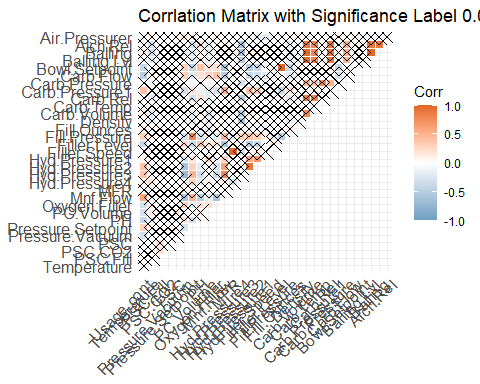
Using a point range visualization like this lets us check the magnitude of outliers by exploiting the difference in robustness between median and mean. It also helps us pick out exactly which variables are best for modeling by sighting variables with many points that influence their distribution more than other variables. Unfortunately, we are still not sure if any of the seemingly outlier points were intentional so a fare bit of intuition is used to identify what should and should not be influencing the distribution.

When considering variable relationships, we also need to consider their correlations. In this final correlation plot we examine not only the variable’s correlation with ‘PH’ but all variable’s correlations with one another.

# Correlation Plot  
order <- ph %>%   
 dplyr::select(where(is.numeric)) %>%  
 arrange() %>%   
 gather(variable, value) %>%   
 pivot\_wider(id\_cols = variable, names\_from = variable, values\_from = value) %>%  
 unnest()  
order <- order[,order(colnames(order),decreasing=TRUE)]  
cors <- cor(order, use = "complete.obs")  
p.mat <- ggcorrplot::cor\_pmat(cors, sig.level = 0.05)  
sum(p.mat > .05)

## [1] 670

ggcorrplot(cors, "square", "lower",   
 colors = c("#6D9EC1", "white", "#E46726"),   
 outline.color = "white",   
 digits = 1,   
 p.mat=p.mat,  
 hc.order = FALSE,   
 hc.method = "complete") +   
 coord\_flip() +   
 labs(title = "Corrlation Matrix with Significance Label 0.05")



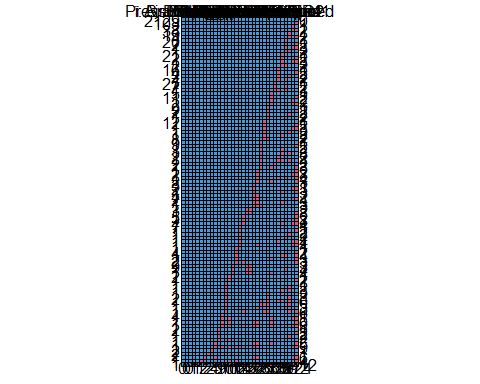
Although computed and shown for reference, we ignore insignificantly correlated variables with p-values greater than or equal to 0.05. Those we ignore have an ‘X’ drawn through its box. Variables directly measuring pressure and flow exhibit more negative control over ‘PH’ shown in darker blue. The only significant exception to this is ‘Carb.Flow.’ Meanwhile, ‘Bowl.Setpoint,’ ‘Filler.Level,’ ‘Oxygen.Filler,’ and ‘Usage.Count’ are the only remaining significantly correlated variables.

# Data Preparation

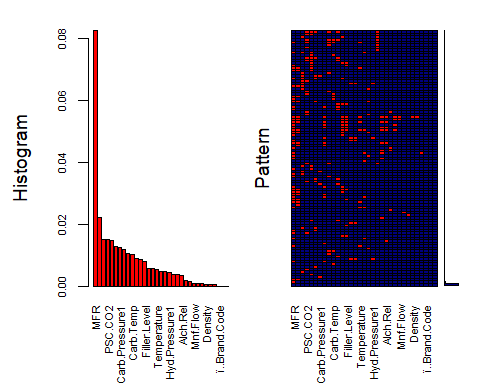
## Imputation

Before imputing, we check for patterns in the missing data. According to the mice method used in the md.pattern() function, there is no clear pattern. The errors are randomly dispersed. However, since we discovered some natural eb and flow in the collection process we double check with our own eyes.

# Check for missing data patterns with mice  
na.pattern.mice <- md.pattern(ph) # No clear pattern



# Double check with hist and plot   
aggr(ph, col=c('navyblue','red'),   
 numbers=TRUE, sortVars=TRUE,   
 labels=names(ph), cex.axis=.7,   
 gap=3, ylab=c("Histogram","Pattern"))



##   
## Variables sorted by number of missings:   
## Variable Count  
## MFR 0.0824581875  
## Filler.Speed 0.0221703617  
## PC.Volume 0.0151691949  
## PSC.CO2 0.0151691949  
## Fill.Ounces 0.0147802412  
## PSC 0.0128354726  
## Carb.Pressure1 0.0124465189  
## Hyd.Pressure4 0.0116686114  
## Carb.Pressure 0.0105017503  
## Carb.Temp 0.0101127966  
## PSC.Fill 0.0089459354  
## Fill.Pressure 0.0085569817  
## Filler.Level 0.0077790743  
## Hyd.Pressure2 0.0058343057  
## Hyd.Pressure3 0.0058343057  
## Temperature 0.0054453520  
## Oxygen.Filler 0.0046674446  
## Pressure.Setpoint 0.0046674446  
## Hyd.Pressure1 0.0042784909  
## Carb.Volume 0.0038895371  
## Carb.Rel 0.0038895371  
## Alch.Rel 0.0035005834  
## Usage.cont 0.0019447686  
## PH 0.0015558149  
## Mnf.Flow 0.0007779074  
## Carb.Flow 0.0007779074  
## Bowl.Setpoint 0.0007779074  
## Density 0.0003889537  
## Balling 0.0003889537  
## Balling.Lvl 0.0003889537  
## ï..Brand.Code 0.0000000000  
## Pressure.Vacuum 0.0000000000  
## Air.Pressurer 0.0000000000

# Small amount missing - simple median will do  
obs.missing.perc <-   
 (sum(ph.desc$obs) / (33\*2571)\*100)  
# replaces NA with median (given a removal of missing values in calculation)  
for (i in colnames(ph)) {  
 ph[[i]][is.na(ph[[i]])] <- median(ph[[i]], na.rm=TRUE)  
}  
# Confirm none are missing  
sum(is.na(ph))

## [1] 0

The histogram distributes the missing data as expected from our inferential statistics. The lion’s share of missing values (8.2%) is concentrated under variable ‘MFR’ dwarfing the next variables by comparison. Our pattern plot also randomly spreads red pixelated rectangles on a dark blue background, indicating completely random missing values. Since the quantity of missing values is quite small (less than 1%) we perform a simple imputation by the median of each variable using a for loop.

## Outlier Extraction

To extract and remove outliers, we first identify which variables contain the outliers. Variable ‘Brand.Code’ is categorical and has no missing values so we exclude it by selecting all numeric variables (which includes everything else). We then identify and replace those outliers using a formula with upper and lower bounds to extract nonoutlier data, leaving the rest as outliers. Thus, we conveniently quantify these outliers as greater than or less than 1.5 times each variable’s interquartile range.

# select numeric variables  
ph.numerics <- ph %>%   
 dplyr::select(where(is.numeric))  
# remove outliers based on IQR  
for (i in colnames(ph.numerics)) {  
 iqr <- IQR(ph.numerics[[i]])  
 q <- quantile(ph.numerics[[i]], probs = c(0.25, 0.75), na.rm = FALSE)  
 qupper <- q[2]+1.5\*iqr  
 qlower <- q[1]+1.5\*iqr  
 outlier\_free <- subset(ph.numerics, ph.numerics[[i]] > (q[1] - 1.5\*iqr) & ph.numerics[[i]] < (q[2]+1.5\*iqr) )  
}  
ph.numerics <- outlier\_free  
# join outlier free numerics with categorical   
Brand.Code <- ph$ï..Brand.Code  
df <- cbind(Brand.Code, ph.numerics)  
df.summary <- summary(df)

After identifying and replacing outliers by looping through each variable in the data, we join the outlier free numeric data with the excluded ‘Brand.Code’ variable. This retains the data types as read in and is complete with a binding function for data frames. We then check for missing data in ‘Brand.Code’ to ensure it all transferred appropriately and review the results to see that very few outliers were present.

## Transformations

Although the data did not exhibit linear trends, nor fit a typically gaussian pattern in its isolation, we consider transformations that would best normalize each variable’s distribution. Using a function in the bestNormalize package we store the chosen transformations of each variable in a data frame named ‘best.norms’ along with corresponding metrics to transform them.

# Produce recommended transformations  
df.nums <- df %>%   
 dplyr::select(where(is.numeric))  
best.norms <- df.nums[1:11,1:10]  
for (i in colnames(df.nums)) {  
 best.norms[[i]] <- bestNormalize(df.nums[[i]],  
 allow\_orderNorm = FALSE,  
 out\_of\_sample =FALSE)  
}  
best.norms$Carb.Volume$chosen\_transform

## Standardized Yeo-Johnson Transformation with 2571 nonmissing obs.:  
## Estimated statistics:  
## - lambda = -4.99994   
## - mean (before standardization) = 0.1999832   
## - sd (before standardization) = 1.568536e-06

From this data frame, we can call upon the desired variables to know which transformation is best at a specific time. These will be used in conjunction with model importance when building models.

## Split and Selections

We split the data set giving 70% to training and 30% testing data frames. Using a forward and backward traveling stepwise regression that considered Akaike information criterion (AIC) to evaluate model fit, we determined the best variables. Quite intuitively, any coefficient in the model that had a p-value lower than 0.05, was used. This indicated significant coefficient to us and was the basis for our selection-based models.

# Split 70-30 training/test  
set.seed(1102)  
index <- createDataPartition(  
 data$PH, p = .7,   
 list = FALSE, times = 1)  
train <- data[index,]  
test <- data[-index,]

# StepAIC Model Selectors  
df.selected <- df %>%   
 dplyr::select(Brand.Code,   
 Carb.Volume,  
 Fill.Ounces,  
 PC.Volume,  
 Carb.Temp,  
 PSC,   
 PSC.Fill,   
 PSC.CO2,  
 PH,  
 Mnf.Flow,   
 Carb.Pressure1,   
 Fill.Pressure,   
 Hyd.Pressure2,   
 Hyd.Pressure3,   
 Filler.Level,  
 Temperature,   
 Usage.cont,  
 Carb.Flow,  
 Density,  
 Balling,  
 Pressure.Vacuum,   
 Oxygen.Filler,   
 Bowl.Setpoint,   
 Pressure.Setpoint,   
 Alch.Rel,  
 Balling.Lvl)

# Model Building

## Parametric Models

Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here.

## Nonparametric Models

Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here.

model.pm <- lm(PH~.,df)  
pm <- stepAIC(model.pm, trace = F, direction = "both")  
p <- summary(pm)$call  
pm <- lm(p[2], df)  
summary(pm)

Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here.

model.ks <- lm(PH~., df)  
summary(model.ks)

Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here.

nnetGrid <- expand.grid(.decay=c(0, 0.01, 0.1, 0.5, 0.9),  
 .size=c(1, 10, 15, 20),  
 .bag=FALSE)  
nnetModel <- train(x = train,  
 y = train,  
 method = "avNNet",  
 tuneGrid = nnetGrid,  
 preProc = c("center", "scale"),  
 trace=FALSE,  
 linout=TRUE,  
 maxit=500)  
# Neural net may take several minutes  
nnetPred <- predict(nnetModel, newdata = testData$x)

Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here.

library(naniar)  
miss\_var\_summary(data)  
vis\_miss(data, sort\_miss = TRUE)

Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here.

library(zoo)  
library(naniar)  
data<-na.locf(data,fromLast = TRUE)  
is.na(data)  
  
  
vis\_miss(data, sort\_miss = TRUE)   
str(data)

Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here.

# Split 70-30 training test  
set.seed(1102)  
index <- createDataPartition(data$PH, p = .7,   
 list = FALSE, times = 1)  
train <- data[index,]  
test <- data[-index,]

Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here.

Ridge Regression

library(elasticnet)  
trctrl<- trainControl(method="repeatedcv", number=3, repeats=2)  
  
ridge <- caret::train(PH~., data=train, method="ridge",   
 trControl=trctrl)  
ridgePred <- predict(ridge, newdata = test)  
ridge\_test<-data.frame(postResample(pred = ridgePred, obs = test$PH))   
ridge\_test

Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here.

Random Forest

library(party)  
library(caret)  
forest <- train(PH~., data=train, method="cforest",   
 trControl=trctrl,  
 tuneLength =2)  
forestPred <- predict(forest, newdata = test)  
forest\_test<-data.frame(postResample(pred = forestPred, obs = test$PH))   
forest\_test

Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here.

Support Vector

library(caret)  
library(kernlab)  
svm\_grid <- expand.grid(C = c(1,1000))  
svm<-train(PH~., data = train,   
 method = 'svmRadialCost',   
 trControl = trctrl,   
 tuneGrid = svm\_grid)  
svmPred <- predict(svm, newdata = test)  
svm\_test<-data.frame(postResample(pred = svmPred, obs = test$PH))   
svm\_test

Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here.

# Model Selection

Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here.

## Evaluation Criteria

Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here.

## Predictions

Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here.

## Selection

Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here.

# Conclusion

Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here.

## Table of Results

Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here.

## Discussion

Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here. Some words here.