Project 02: Team Assignment

2025-05-05

Table of contents

Due: May 18, 2025 11:59 PM

Assignment

This is role playing. I am your new boss. I am in charge of production at ABC Beverage and you are a team of data scientists reporting to me. My leadership has told me that new

regulations are requiring us to understand our manufacturing process, the predictive factors and be able to report to them our predictive model of "PH".

Please use the historical data set I am providing. Build and report the factors in BOTH a technical and non-technical report. I like to use Word and Excel. Please provide your non-technical report in a business friendly readable document and your predictions in an Excel readable format. The technical report should show clearly the models you tested and how you selected your final approach.

Please submit both Rpubs links and .rmd files or other readable formats for technical and non-technical reports. Also submit the excel file showing the prediction of your models for pH.

Setup

```
library(caret)
library(corrplot)
library(earth)
library(Formula)
library(glue)
library(knitr)
library(lattice)
library(plotmo)
library(plotrix)
library(readxl)
library(tidyverse)
```

Load the Data

```
trainData <- read_excel(
  path = "./data/TrainingData.xlsx"
)
testData <- read_excel(
  path = "./data/TestingData.xlsx"
)
```

EDA

We are going to do some exploratory data analysis (EDA) on the data. We will look at the data, the missing values, the correlation, and the visualizations.

Summary Information

Let's get a taste of our data. We shall examine our columns and their types, the number of rows and columns, and the first few rows of the data. Perhaps most importantly, we will look at the summary of the data: the mean, median, min, max, quartiles, and standard deviation of each column.

```
data.frame(
  train_nrow = nrow(trainData),
  train_ncol = ncol(trainData),
  test_nrow = nrow(testData),
  test_ncol = ncol(testData)
) |>
  kable()
```

train_nrow	train_ncol	test_nrow	test_ncol
2571	33	267	33

```
sapply(trainData, class) |>
  kable()
```

	x
Brand Code	character
Carb Volume	numeric
Fill Ounces	numeric
PC Volume	numeric
Carb Pressure	numeric
Carb Temp	numeric
PSC	numeric
PSC Fill	numeric
PSC CO2	numeric
Mnf Flow	numeric
Carb Pressure1	numeric
Fill Pressure	numeric
Hyd Pressure1	numeric
Hyd Pressure2	numeric
Hyd Pressure3	numeric
Hyd Pressure4	numeric
Filler Level	numeric

	x
Filler Speed	numeric
Temperature	numeric
Usage cont	numeric
Carb Flow	numeric
Density	numeric
MFR	numeric
Balling	numeric
Pressure Vacuum	numeric
PH	numeric
Oxygen Filler	numeric
Bowl Setpoint	numeric
Pressure Setpoint	numeric
Air Pressurer	numeric
Alch Rel	numeric
Carb Rel	numeric
Balling Lvl	numeric

```
# These make a mess of a PDF
# head(trainData)
# head(testData)
# summary(trainData)
```

Missing Values (`NA`)

Let's quantify how much data is missing from our data.

```
count_missing <- function(data) {
  data |>
    summarise(across(everything(), ~ sum(is.na(.)))) |>
    pivot_longer(
      everything(),
      names_to = "variable",
      values_to = "missing"
    ) |>
    filter(missing > 0) |>
    mutate(
      ratio = missing / nrow(data)
    ) |>
    arrange(desc(missing))
```

```
}

count_missing(trainData) |>
  kable()
```

variable	missing	ratio
MFR	212	0.0824582
Brand Code	120	0.0466744
Filler Speed	57	0.0221704
PC Volume	39	0.0151692
PSC CO2	39	0.0151692
Fill Ounces	38	0.0147802
PSC	33	0.0128355
Carb Pressure1	32	0.0124465
Hyd Pressure4	30	0.0116686
Carb Pressure	27	0.0105018
Carb Temp	26	0.0101128
PSC Fill	23	0.0089459
Fill Pressure	22	0.0085570
Filler Level	20	0.0077791
Hyd Pressure2	15	0.0058343
Hyd Pressure3	15	0.0058343
Temperature	14	0.0054454
Oxygen Filler	12	0.0046674
Pressure Setpoint	12	0.0046674
Hyd Pressure1	11	0.0042785
Carb Volume	10	0.0038895
Carb Rel	10	0.0038895
Alch Rel	9	0.0035006
Usage cont	5	0.0019448
PH	4	0.0015558
Mnf Flow	2	0.0007779
Carb Flow	2	0.0007779
Bowl Setpoint	2	0.0007779
Density	1	0.0003890
Balling	1	0.0003890
Balling Lvl	1	0.0003890

```
count_missing(testData) |>
  kable()
```

variable	missing	ratio
PH	267	1.0000000
MFR	31	0.1161049
Filler Speed	10	0.0374532
Brand Code	8	0.0299625
Fill Ounces	6	0.0224719
PSC	5	0.0187266
PSC CO2	5	0.0187266
PC Volume	4	0.0149813
Carb Pressure1	4	0.0149813
Hyd Pressure4	4	0.0149813
PSC Fill	3	0.0112360
Oxygen Filler	3	0.0112360
Alch Rel	3	0.0112360
Fill Pressure	2	0.0074906
Filler Level	2	0.0074906
Temperature	2	0.0074906
Usage cont	2	0.0074906
Pressure Setpoint	2	0.0074906
Carb Rel	2	0.0074906
Carb Volume	1	0.0037453
Carb Temp	1	0.0037453
Hyd Pressure2	1	0.0037453
Hyd Pressure3	1	0.0037453
Density	1	0.0037453
Balling	1	0.0037453
Pressure Vacuum	1	0.0037453
Bowl Setpoint	1	0.0037453
Air Pressurer	1	0.0037453

Correlation List

The list is long, so we shall consider correlation that is $\rho >= 0.50$ to be interesting. There is a full pairwise plot just below.

```
cor(trainData[-1], use = "pairwise.complete.obs") |>
  as.data.frame() |>
  rownames_to_column("variable") |>
  pivot_longer(
    -variable,
    names_to = "correlation",
```

```
  values_to = "value"
) |>
mutate(
  abs_value = abs(round(value, 2))
) |>
filter(
  variable != correlation & abs_value >= 0.5
) |>
arrange(desc(abs_value)) |>
kable()
```

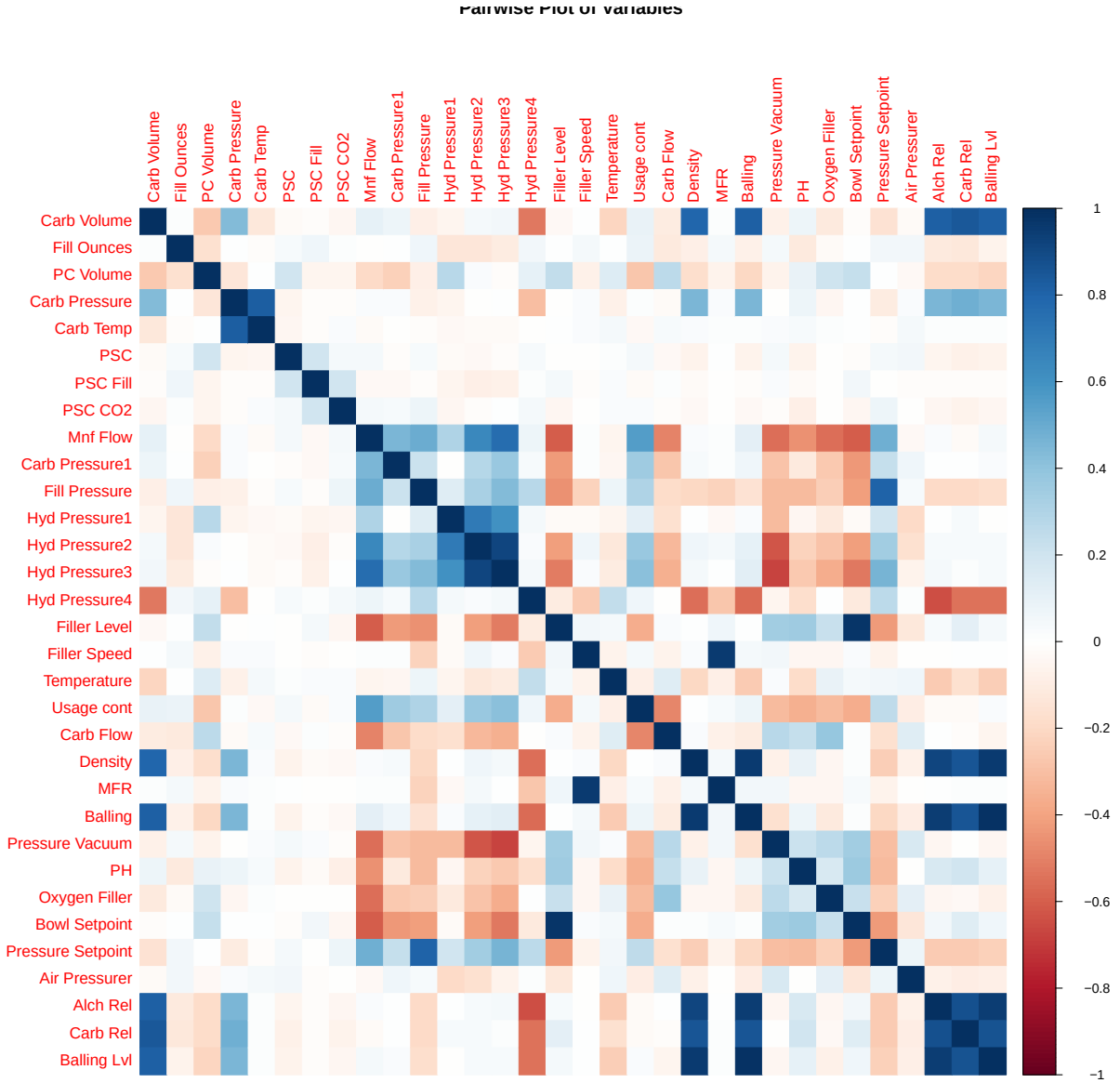
variable	correlation	value	abs_value
Balling	Balling Lvl	0.9782085	0.98
Balling Lvl	Balling	0.9782085	0.98
Density	Balling	0.9551553	0.96
Balling	Density	0.9551553	0.96
Filler Level	Bowl Setpoint	0.9489345	0.95
Density	Balling Lvl	0.9479195	0.95
Bowl Setpoint	Filler Level	0.9489345	0.95
Balling Lvl	Density	0.9479195	0.95
Filler Speed	MFR	0.9297119	0.93
MFR	Filler Speed	0.9297119	0.93
Alch Rel	Balling Lvl	0.9258951	0.93
Balling Lvl	Alch Rel	0.9258951	0.93
Hyd Pressure2	Hyd Pressure3	0.9248823	0.92
Hyd Pressure3	Hyd Pressure2	0.9248823	0.92
Balling	Alch Rel	0.9247934	0.92
Alch Rel	Balling	0.9247934	0.92
Density	Alch Rel	0.9027398	0.90
Alch Rel	Density	0.9027398	0.90
Alch Rel	Carb Rel	0.8435849	0.84
Carb Rel	Alch Rel	0.8435849	0.84
Carb Rel	Balling Lvl	0.8447733	0.84
Balling Lvl	Carb Rel	0.8447733	0.84
Density	Carb Rel	0.8243548	0.82
Balling	Carb Rel	0.8224442	0.82
Carb Rel	Density	0.8243548	0.82
Carb Rel	Balling	0.8224442	0.82
Carb Pressure	Carb Temp	0.8141717	0.81
Carb Temp	Carb Pressure	0.8141717	0.81
Carb Volume	Carb Rel	0.7933040	0.79

variable	correlation	value	abs_value
Carb Rel	Carb Volume	0.7933040	0.79
Carb Volume	Balling	0.7827032	0.78
Carb Volume	Alch Rel	0.7807840	0.78
Carb Volume	Balling Lvl	0.7794982	0.78
Balling	Carb Volume	0.7827032	0.78
Alch Rel	Carb Volume	0.7807840	0.78
Balling Lvl	Carb Volume	0.7794982	0.78
Carb Volume	Density	0.7616051	0.76
Mnf Flow	Hyd Pressure3	0.7590961	0.76
Hyd Pressure3	Mnf Flow	0.7590961	0.76
Density	Carb Volume	0.7616051	0.76
Hyd Pressure1	Hyd Pressure2	0.7233047	0.72
Hyd Pressure2	Hyd Pressure1	0.7233047	0.72
Fill Pressure	Pressure Setpoint	0.6771748	0.68
Pressure Setpoint	Fill Pressure	0.6771748	0.68
Mnf Flow	Hyd Pressure2	0.6561663	0.66
Hyd Pressure2	Mnf Flow	0.6561663	0.66
Hyd Pressure1	Hyd Pressure3	0.6341679	0.63
Hyd Pressure3	Hyd Pressure1	0.6341679	0.63
Hyd Pressure3	Pressure Vacuum	-0.6028821	0.60
Pressure Vacuum	Hyd Pressure3	-0.6028821	0.60
Mnf Flow	Filler Level	-0.5785108	0.58
Mnf Flow	Bowl Setpoint	-0.5806429	0.58
Filler Level	Mnf Flow	-0.5785108	0.58
Bowl Setpoint	Mnf Flow	-0.5806429	0.58
Hyd Pressure2	Pressure Vacuum	-0.5647077	0.56
Pressure Vacuum	Hyd Pressure2	-0.5647077	0.56
Mnf Flow	Pressure Vacuum	-0.5277447	0.53
Mnf Flow	Oxygen Filler	-0.5302731	0.53
Pressure Vacuum	Mnf Flow	-0.5277447	0.53
Oxygen Filler	Mnf Flow	-0.5302731	0.53
Mnf Flow	Usage cont	0.5204366	0.52
Usage cont	Mnf Flow	0.5204366	0.52
Hyd Pressure3	Filler Level	-0.4992295	0.50
Hyd Pressure4	Alch Rel	-0.4997986	0.50
Filler Level	Hyd Pressure3	-0.4992295	0.50
Alch Rel	Hyd Pressure4	-0.4997986	0.50

Pairwise Plot

Correlation plot of our training data:

```
corrplot(
  cor(trainData[-1], use = "complete.obs"),
  method = "color",
  title = "Pairwise Plot of Variables"
)
```

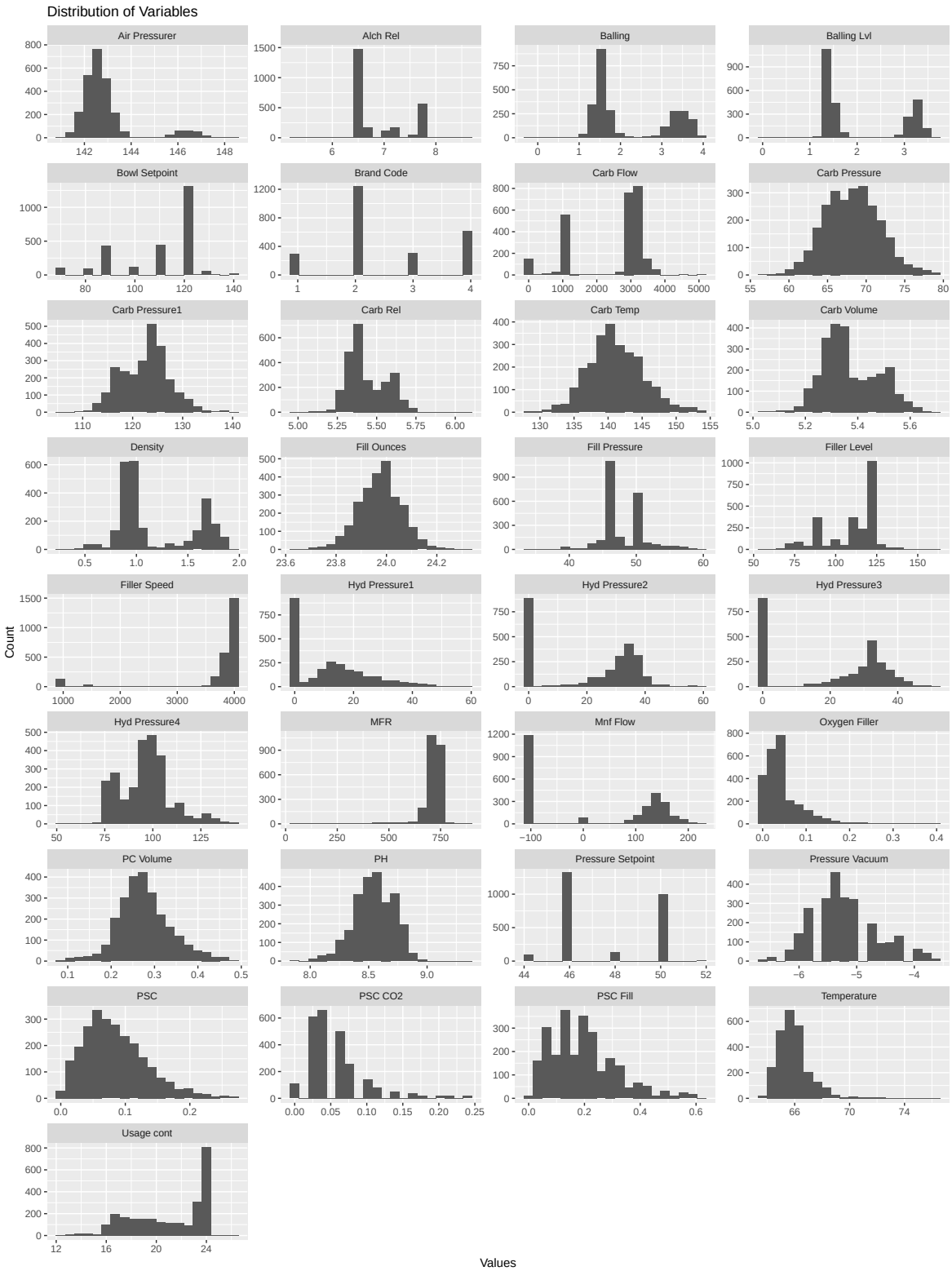


Visualize

Histogram

Histogram of our original data. We are recoding `Brand Code` to be numeric so that we can plot it. We will also use `pivot_longer()` to reshape the data so that we can plot it all in one fell swoop. And we have a handle on `NA`s, so we are filtering them out.

```
trainData |>
  mutate(
    `Brand Code` = recode(
      `Brand Code`,
      "A" = 1,
      "B" = 2,
      "C" = 3,
      "D" = 4
    )
  ) |>
  pivot_longer(
    cols = everything(),
    names_to = "variable",
    values_to = "values"
  ) |>
  filter(!is.na(values)) |>
  ggplot(aes(x = values)) +
  geom_histogram(bins = 20) +
  facet_wrap(~ variable, ncol = 4, scales = "free") +
  labs(
    title = "Distribution of Variables",
    x = "Values",
    y = "Count"
  )
```

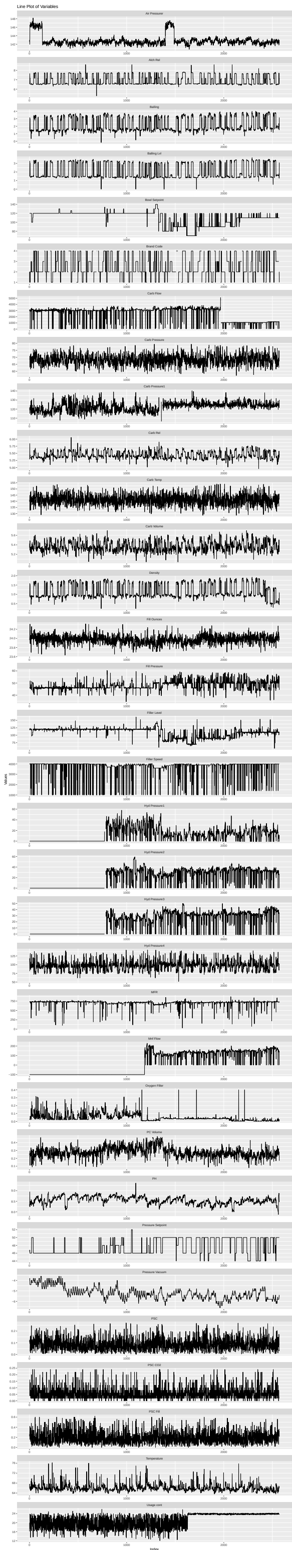

Distribution of Variables
Air Pressurer
Alch Rel
Balling
Balling Lvl
Bowl Setpoint
Brand Code
Carb Flow
Carb Pressure
Carb Pressure1
Carb Rel
Carb Temp
Carb Volume
Density
Fill Ounces
Fill Pressure
Filler Level
Filler Speed
Hyd Pressure1
Hyd Pressure2
Hyd Pressure3
Hyd Pressure4
MFR
Mnf Flow
Oxygen Filler
PC Volume
PH
Pressure Setpoint
Pressure Vacuum
PSC
PSC CO2
PSC Fill
Temperature
Usage cont
Count
Values

Line Plot

We would like to better understand the data. A graphic can be invaluable. But our data is not a time series. Nonetheless, they are observations taken at different times (we think it's safe to say). And with some preliminary plotting, the results are interesting if nothing else. So, we will try plotting it with a line plot and see what surfaces. We would like to emphasize that in no way are we suggesting that this should be interpreted as a time series. We are simply trying to get a better understanding of the data.

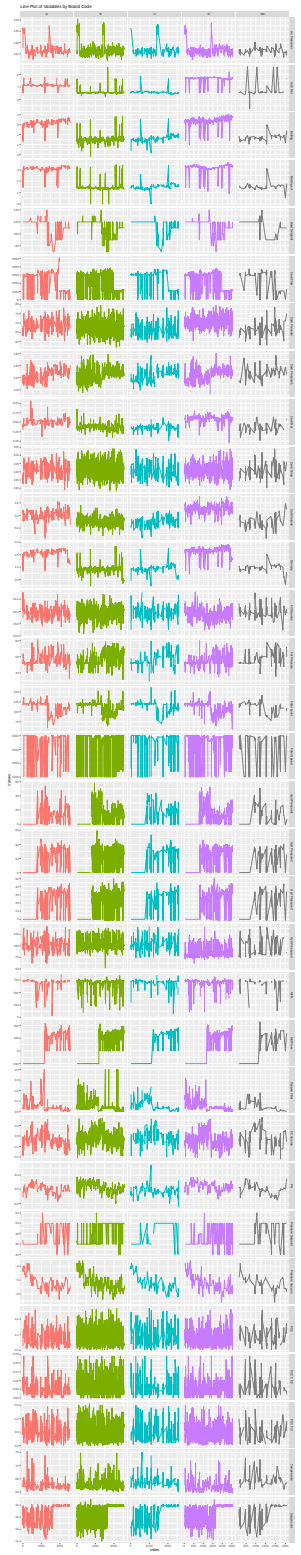
We will use `pivot_longer()` to reshape the data so that we can plot it one variable stacked on top of another. We will also use `row_number()` to create an index for the x-axis. We will use `facet_wrap()` to create a separate plot for each variable.

```
trainData |>
  mutate(
    index = row_number(),
    `Brand Code` = recode(
      `Brand Code`,
      "A" = 1,
      "B" = 2,
      "C" = 3,
      "D" = 4
    )
  ) |>
  relocate(
    index,
    .before = everything()
  ) |>
  pivot_longer(
    cols = -1,
    names_to = "variable",
    values_to = "values"
  ) |>
  ggplot(aes(x = index, y = values)) +
  geom_line() +
  facet_wrap(~ variable, ncol = 1, scales = "free") +
  labs(
    title = "Line Plot of Variables",
    x = "Index",
    y = "Values"
  )
```


Line Plot of Variables

We are curious about the role of `Brand Code`. Do patterns emerge when the data is separated by `Brand Code`? Our first plot was mayhem. We shall facet such that each plot is a single predictor intersected with a single brand (`A`, `B`, `C`, `D`, `NA`). We will use `facet_wrap()` to create a separate plot for each variable and `Brand Code`. This will create a grid of plots, with one row for each variable and one column for each `Brand Code`.

```
trainData |>
  mutate(
    index = row_number()
  ) |>
  relocate(
    index,
    .before = everything()
  ) |>
  pivot_longer(
    # preserve the `Brand Code` and `index` columns
    cols = -c(1, 2),
    names_to = "variable",
    values_to = "values"
  ) |>
  ggplot(aes(x = index, y = values, color = `Brand Code`)) +
  geom_line(show.legend = FALSE) +
  facet_grid(variable ~ `Brand Code`, scales = "free") +
  labs(
    title = "Line Plot of Variables by Brand Code",
    x = "Index",
    y = "Values"
  )
```


Line Plot of Variables by Board Code

Preprocessing Round 1

We have some work to do:

- We have a lot of variables.
- We have a lot of correlation.
- We have `NA`'s in our target variable.
- We have predictors with varying amounts of missing data.
- We have a categorical variable.
- And we have some data that is not `NA`, but looks as if it should be.

We shall solve all of these problems. We are going to save most of it to a new dataframe called `trainDataPP`. But we will remove our `NA` target rows from `trainData` because we don't want to train our model to predict `NA`s. We will use the `dummyVars()` function to one-hot encode our categorical variable.

```
# first we will remove the `NA` predictors from our data
trainData <- trainData |>
  filter(!is.na(`PH`))

# Miscellaneous preprocessing
trainDataPP <- trainData |>
  mutate(
    # Make the NAs encoded as numbers into NAs.
    # For those with legitimate 0 values, this is flawed.
    `Hyd Pressure1` = if_else(`Hyd Pressure1` == 0, NA, `Hyd Pressure1`),
    `Hyd Pressure2` = if_else(`Hyd Pressure2` == 0, NA, `Hyd Pressure2`),
    `Hyd Pressure3` = if_else(`Hyd Pressure3` == 0, NA, `Hyd Pressure3`),
    `Mnf Flow` = if_else(`Mnf Flow` < 0, NA, `Mnf Flow`),
    # We will use `Brand Code`'s NAs as "Unknowns"'
    `Brand Code` = if_else(is.na(`Brand Code`), "Unknown", `Brand Code`)
  ) |>
  rename(
    # Rename `Brand Code` otherwise `dummyVars()` makes a mess of it.
    BrandCode. = `Brand Code`
  )

# Apply one-hot encoding to `BrandCode.`
trainDataPP <- dummyVars(~ ., data = trainDataPP) |>
  predict(newdata = trainDataPP) |>
  as.data.frame()
```

Now we shall reduce the number of variables by removing those that are highly correlated. We will consider correlation that is $\rho >= 0.80$ to be significant. We will remove one of each pair

of variables that are correlated. And we will be careful when remove variables that we do not remove both variables of a relationship. Our highly correlated variable are:

Variable 1	Variable 2	Correlation	Remove
Balling	Balling Lvl	0.98	Balling Lvl
Density	Balling	0.96	Density
Filler Level	Bowl Setpoint	0.95	Filler Level
Density	Balling Lvl	0.95	Density
Filler Speed	MFR	0.93	MFR
Alch Rel	Balling Lvl	0.93	Balling Lvl
Balling	Alch Rel	0.92	Alch Rel
Density	Alch Rel	0.90	Density
BrandCode.D	Alch Rel	0.89	Alch Rel
Alch Rel	Carb Rel	0.84	Alch Rel
Carb Rel	Balling Lvl	0.84	Balling Lvl
Density	Carb Rel	0.82	Density
Balling	Carb Rel	0.82	-
Carb Pressure	Carb Temp	0.81	Carb Pressure
Hyd Pressure2	Filler Speed	0.80	Hyd Pressure2

```
trainDataPP <- trainDataPP |>
  select(
    -c(
      "`Balling Lvl`",
      Density,
      "`Filler Level`",
      MFR,
      "`Alch Rel`",
      "`Carb Pressure`",
      "`Hyd Pressure2`"
    )
  )
```

Explore Partially Preprocessed Data

Taste

```
data.frame(
  train_nrow = nrow(trainDataPP),
```

```
  train_ncol = ncol(trainDataPP),
  test_nrow = nrow(trainDataPP),
  test_ncol = ncol(trainDataPP)
) |>
  kable()
```

train_nrow	train_ncol	test_nrow	test_ncol
2567	30	2567	30

```
sapply(trainDataPP, class) |>
  kable()
```

	x
BrandCode.A	numeric
BrandCode.B	numeric
BrandCode.C	numeric
BrandCode.D	numeric
BrandCode.Unknown	numeric
Carb Volume	numeric
Fill Ounces	numeric
PC Volume	numeric
Carb Temp	numeric
PSC	numeric
PSC Fill	numeric
PSC CO2	numeric
Mnf Flow	numeric
Carb Pressure1	numeric
Fill Pressure	numeric
Hyd Pressure1	numeric
Hyd Pressure3	numeric
Hyd Pressure4	numeric
Filler Speed	numeric
Temperature	numeric
Usage cont	numeric
Carb Flow	numeric
Balling	numeric
Pressure Vacuum	numeric
PH	numeric
Oxygen Filler	numeric

	x
Bowl Setpoint	numeric
Pressure Setpoint	numeric
Air Pressurer	numeric
Carb Rel	numeric

```
# These make a mess of a PDF
# head(trainDataPP)
# summary(trainDataPP)
```

Missing Values

```
trainDataPP |>
  summarise(across(everything(), ~ sum(is.na(.)))) |>
  pivot_longer(
    everything(),
    names_to = "variable",
    values_to = "missing"
  ) |>
  filter(missing > 0) |>
  mutate(
    ratio = missing / nrow(trainData)
  ) |>
  arrange(desc(missing)) |>
  kable()
```

variable	missing	ratio
Mnf Flow	1183	0.4608492
Hyd Pressure1	849	0.3307363
Hyd Pressure3	827	0.3221660
Filler Speed	54	0.0210362
PC Volume	39	0.0151928
PSC CO2	39	0.0151928
Fill Ounces	38	0.0148033
PSC	33	0.0128555
Carb Pressure1	32	0.0124659
Hyd Pressure4	28	0.0109077
Carb Temp	26	0.0101286

variable	missing	ratio
PSC Fill	23	0.0089599
Fill Pressure	18	0.0070121
Temperature	12	0.0046747
Pressure Setpoint	12	0.0046747
Oxygen Filler	11	0.0042852
Carb Volume	10	0.0038956
Carb Rel	8	0.0031165
Usage cont	5	0.0019478
Carb Flow	2	0.0007791
Bowl Setpoint	2	0.0007791

Correlation List

Let's see what has a significant amount of correlation (>=0.80). There is a full pairwise plot below. I am going to keep both `Carb Rel` and `Balling` in the list, because I removed variables that were correlated with each of them. So I don't feel comfortable removing them.

```
cor(trainDataPP, use = "pairwise.complete.obs") |>
  as.data.frame() |>
  rownames_to_column("variable") |>
  pivot_longer(
    -variable,
    names_to = "correlation",
    values_to = "value"
  ) |>
  mutate(
    abs_value = abs(round(value, 2))
  ) |>
  filter(
    variable != correlation & abs_value >= 0.8
  ) |>
  arrange(desc(abs_value)) |>
  kable()
```

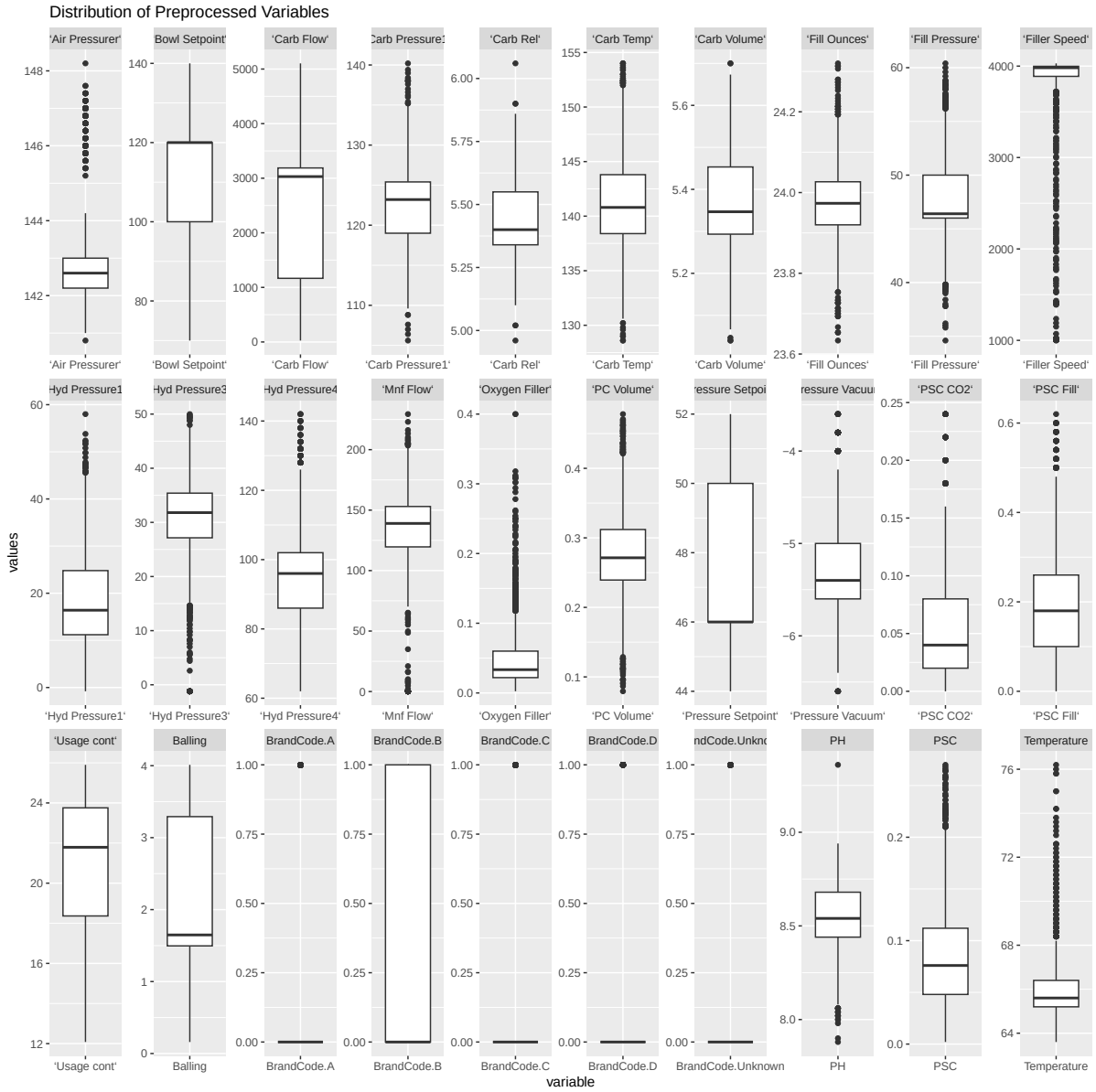
variable	correlation	value	abs_value
Balling	Carb Rel	0.8231274	0.82
Carb Rel	Balling	0.8231274	0.82

Visualize

Box Plot

Finally (for histogram'ish stuff) we would like to see a box plot of each variable to get a better idea of the fringes (outliers).

```
trainDataPP |>
  pivot_longer(
    cols = everything(),
    names_to = "variable",
    values_to = "values"
  ) |>
  filter(!is.na(values)) |>
  ggplot(aes(x = variable, y = values)) +
  geom_boxplot() +
  labs(
    title = "Distribution of Preprocessed Variables"
  ) +
  facet_wrap(~variable, ncol = 10, scale="free")
```

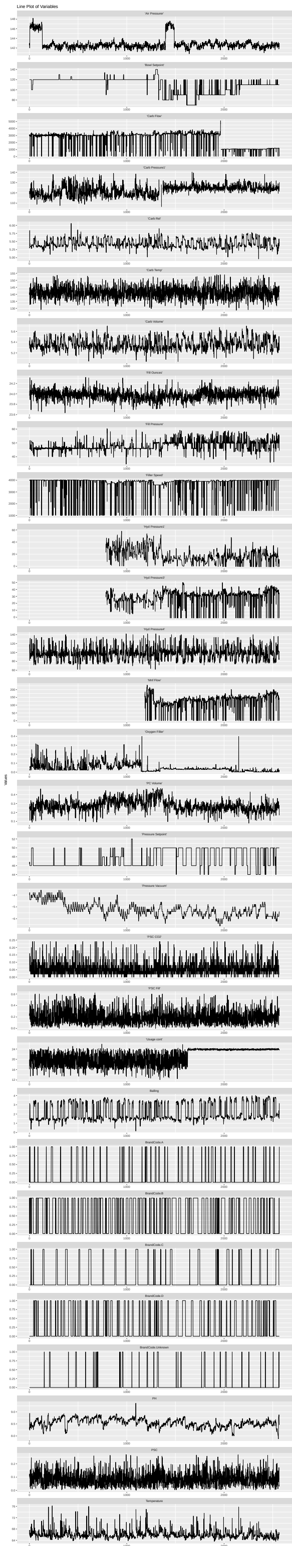


Line Plot

Let's see what our line plots look like now. We shall use the same method as before.

```
trainDataPP |>
  mutate(
    index = row_number()
  ) |>
```

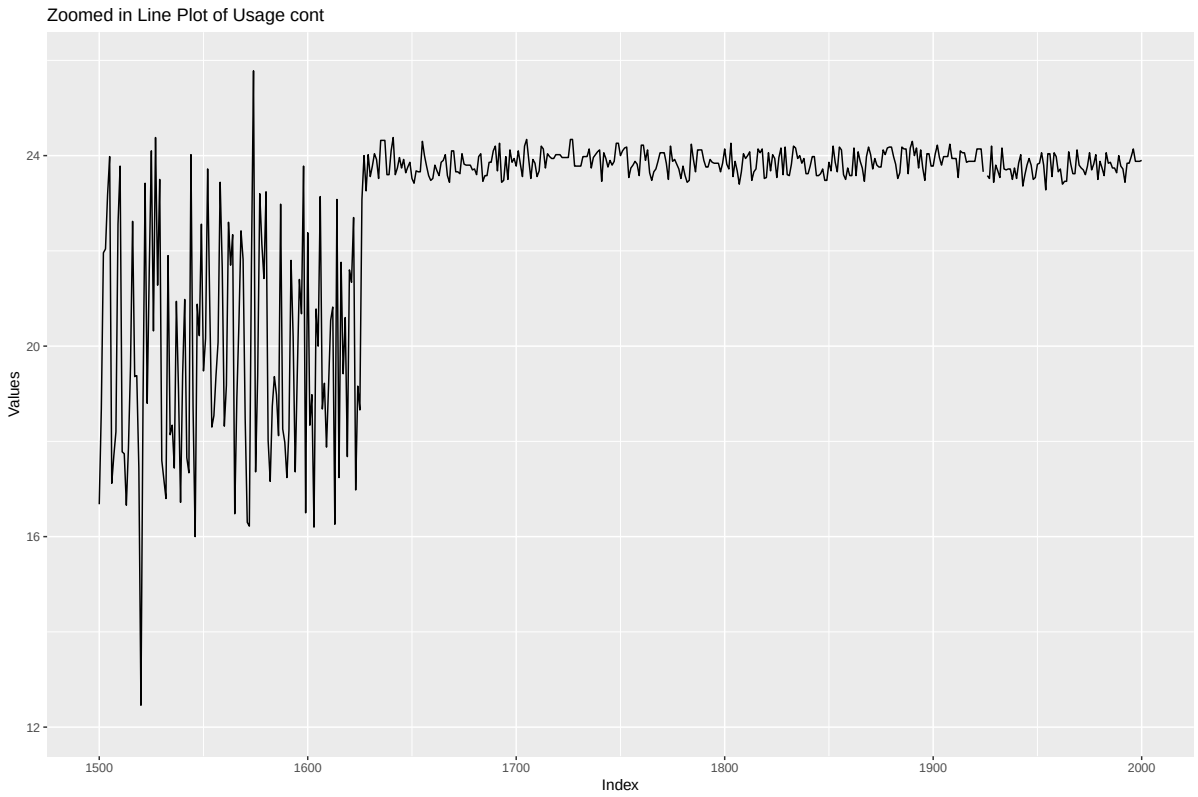
```
  relocate(
    index,
    .before = everything()
  ) |>
  pivot_longer(
    cols = -1,
    names_to = "variable",
    values_to = "values"
  ) |>
  ggplot(aes(x = index, y = values)) +
  geom_line() +
  facet_wrap(~ variable, ncol = 1, scales = "free") +
  labs(
    title = "Line Plot of Variables",
    x = "Index",
    y = "Values"
  )
```


Line Plot of Variables

We are inclined to do something about `Hyd Pressure1`, `Hyd Pressure3`, and `Mnf Flow`. They are are `NA` for a long time. The other odd ball is `Usage cont`. It's value goes high at ~1600. Let's zoom in on it and see if we can see anything.

```
trainDataPP |>
  mutate(
    index = row_number()
  ) |>
  relocate(
    index,
    .before = everything()
  ) |>
  pivot_longer(
    cols = -1,
    names_to = "variable",
    values_to = "values"
  ) |>
  filter(variable == "`Usage cont`") |>
  ggplot(aes(x = index, y = values)) +
  geom_line() +
  scale_x_continuous(
    limits = c(1500, 2000)
  ) +
  labs(
    title = "Zoomed in Line Plot of Usage cont",
    x = "Index",
    y = "Values"
  )
```

```
Warning: Removed 2066 rows containing missing values or values outside the scale range
(`geom_line()`).
```



Egads, it looks as if the noisy behavior that led up to ~1625 shrunk down to a much smaller scale. I'm tempted to explode it and center it around 0. Let's see if there is correlation between [0, 1600], if so then we can eliminate this guy and use the other one.

```
cor(trainDataPP[1:1600,], use = "pairwise.complete.obs") |>
  as.data.frame() |>
  rownames_to_column("variable") |>
  pivot_longer(
    -variable,
    names_to = "correlation",
    values_to = "value"
  ) |>
  mutate(
    abs_value = abs(round(value, 2))
  ) |>
  filter(
    variable == "`Usage cont`" & value > 0.1
  ) |>
  arrange(desc(abs_value)) |>
  kable()
```

variable	correlation	value	abs_value
Usage cont	Usage cont	1.0000000	1.0
Usage cont	Carb Flow	0.1036177	0.1

Not even close. Lastly, let's do the same for `Hyd Pressure1`, `Hyd Pressure3`, and `Mnf Flow`.

```
cor(trainDataPP[1200:dim(trainDataPP)[1],], use = "pairwise.complete.obs") |>
  as.data.frame() |>
  rownames_to_column("variable") |>
  pivot_longer(
    -variable,
    names_to = "correlation",
    values_to = "value"
  ) |>
  mutate(
    abs_value = abs(round(value, 2))
  ) |>
  filter(
    variable != correlation & value > 0.7
  ) |>
  arrange(desc(abs_value)) |>
  kable()
```

variable	correlation	value	abs_value
Balling	Carb Rel	0.8234461	0.82
Carb Rel	Balling	0.8234461	0.82
Carb Volume	Carb Rel	0.8068913	0.81
Carb Rel	Carb Volume	0.8068913	0.81
Carb Volume	Balling	0.7835841	0.78
Mnf Flow	Filler Speed	0.7757626	0.78
Filler Speed	Mnf Flow	0.7757626	0.78
Balling	Carb Volume	0.7835841	0.78
BrandCode.D	Balling	0.7653092	0.77
Hyd Pressure3	Filler Speed	0.7710100	0.77
Filler Speed	Hyd Pressure3	0.7710100	0.77
Balling	BrandCode.D	0.7653092	0.77
Mnf Flow	Hyd Pressure3	0.7251646	0.73
Hyd Pressure3	Mnf Flow	0.7251646	0.73
BrandCode.D	Carb Volume	0.7203591	0.72
BrandCode.D	Carb Rel	0.7205871	0.72

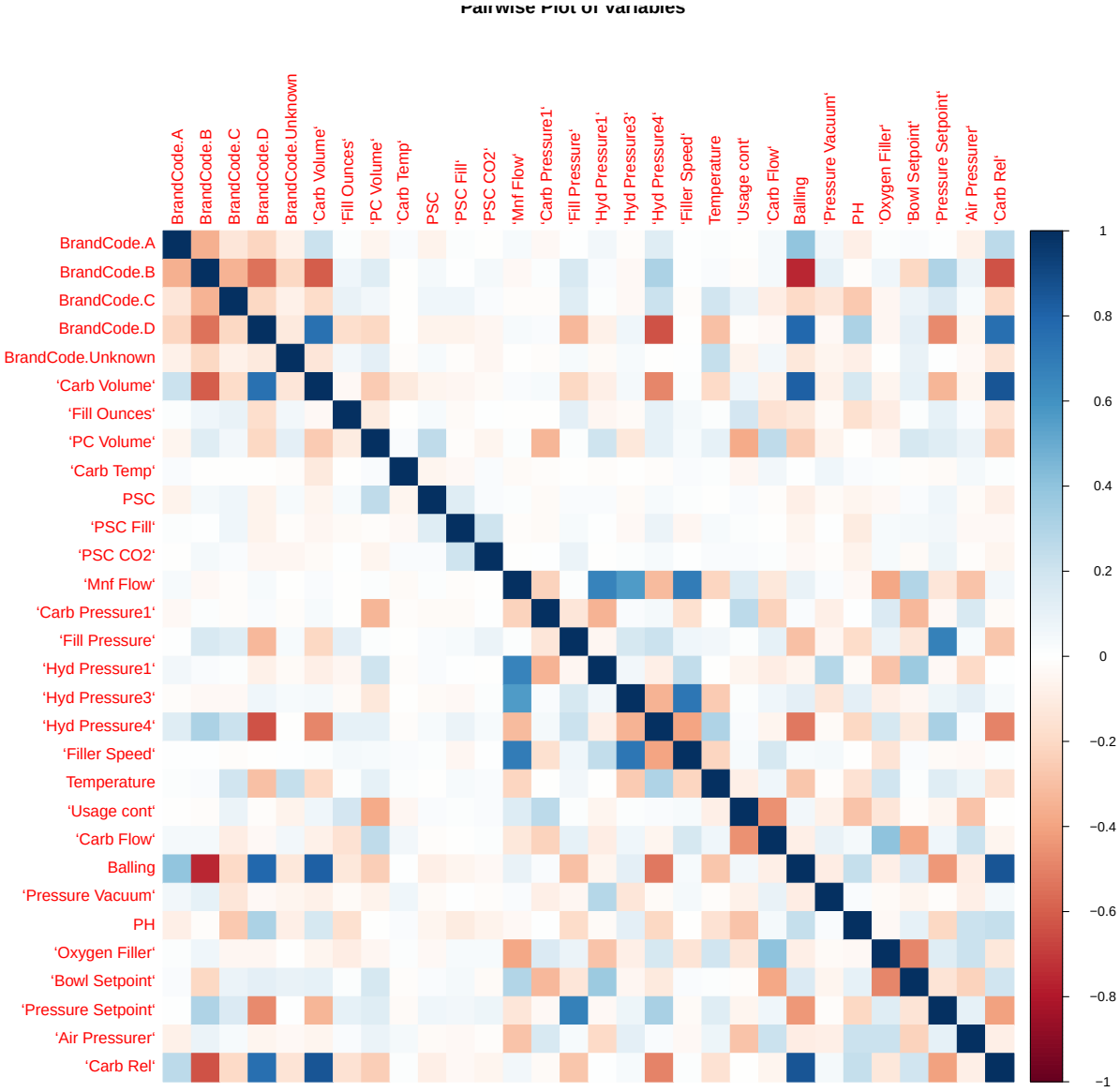
variable	correlation	value	abs_value
`Carb Volume`	BrandCode.D	0.7203591	0.72
`Carb Rel`	BrandCode.D	0.7205871	0.72

Oh, what light through yonder window breaks? It looks like `Hyd Pressure3` is highly correlated with `Filler Speed`. We believe that is justification for removing `Hyd Pressure3` from the list.

Pairwise Plot

Correlation plot of our preprocessed training data:

```
corrplot(
  cor(trainDataPP, use = "complete.obs"),
  method = "color",
  title = "Pairwise Plot of Variables"
)
```



Preprocessing Round 2

> **Caution**
>
> We are dropping `Hyd Pressure1` and `Mnf Flow` because of the large volume of `NA`s. But we are going to keep `Usage cont`. It has some strange values, nonetheless in our preliminary model evaluation we found that `Usage cont` is a highly important variable.
> Note: we are also dropping `Hyd Pressure3` because it is highly correlated with `Filler`

> Speed.

We are going to drop `Hyd Pressure3` because of what values we have for him, he's pretty correlated `Filler Speed`. And right now I don't know what to do with `Hyd Pressure1` and `Mnf Flow`, nor do I know what to do with `Usage cont`. We need to get on with the show. So, I'm going to drop them all and prepare our dataset for non-linear regression.

```
trainDataPP <- trainDataPP |>
  select(
    -c(
      "`Hyd Pressure1`",
      "`Hyd Pressure3`",
      "`Mnf Flow`"
    )
  )
```

Non-Linear Regression

Create a little workshop of utilities.

```
processAndPartition <- function(data, methods) {
  # separate the data into training and test sets before we preprocess the data
  X <- select(data, -PH)
  y <- data$PH

  # preprocess the data
  model <- preProcess(X, method = methods)
  X <- predict(model, X)

  set.seed(31415)
  trainIndex <- createDataPartition(
    y,
    p = .8,
    list = FALSE
  )
  return(list(
    train_X = X[trainIndex, ],
    train_y = y[trainIndex],
    test_X = X[-trainIndex, ],
    test_y = y[-trainIndex]
  ))
```

```
}

report <- function(model, data) {
  predictions <- predict(model, data$test_X)
  RMSE <- RMSE(predictions, data$test_y)
  R2 <- R2(predictions, data$test_y)
  glue("Model: {model$method}, RMSE: {round(RMSE, 2)}, R2: {round(R2, 2)}") |>
    print()
  glue("Best tuned parameters: {colnames(model$bestTune)} = {model$bestTune}") |>
    print()
  print(varImp(model))
}
```

Prepare the Data

It may make sense to tailor it for some models. We'll see. For now I'm going to assume that scaling, centering and imputing is good for everybody.

Our first data set for training - `dataA` - is derived from `trainDataPP`. We are going to scale, center, impute and do an 80% train data partition on it.

```
dataA <- processAndPartition(trainDataPP, c("center", "scale", "medianImpute", "pca"))
```

Our performance has not been so good. We want to see whether it improves when we use our unadulterated dataA. Actually, we will adulterate it a little bit. We will use `dummyVars()` to one-hot encode our categorical variables.

```
trainDataF <- dummyVars(~ ., data = trainData) |>
  predict(newdata = trainData) |>
  as.data.frame()

# Note: I experimented with "pca" on this training set and it all around did worse
# that without it.
dataB <- processAndPartition(trainDataF, c("center", "scale", "medianImpute"))
```

KNN

k-Nearest Neighbors is a non-parametric method used for classification and regression. It works by finding the k-nearest neighbors to a given point and using their values to predict the value of that point.

Dataset A

We shall use our refined dataset for this fitting.

```
model <- train(
  x = dataA$train_X,
  y = dataA$train_y,
  method = "knn",
  tuneLength = 20,
  trControl = trainControl(
    method = "cv",
    number = 10
  )
)
report(model, dataA)
```

```
Model: knn, RMSE: 0.13, R2: 0.44
Best tuned parameters: k = 9
loess r-squared variable importance

     Overall
PC2  100.000
PC4   38.874
PC9   25.462
PC3   21.056
PC18  20.220
PC1   18.452
PC6   11.013
PC16   9.018
PC12   7.658
PC8    6.714
PC10   6.045
PC5    5.377
PC15   5.171
PC11   4.305
PC14   2.246
PC17   1.545
PC19   1.433
PC7    1.350
PC13   0.000
```

Dataset B

And we will repeat fitting KNN with the same hyperparameters but this time we will use our full model.

```
model <- train(
  x = dataB$train_X,
  y = dataB$train_y,
  method = "knn",
  tuneLength = 20,
  trControl = trainControl(
    method = "cv",
    number = 10
  )
)
report(model, dataB)
```

```
Model: knn, RMSE: 0.12, R2: 0.51
Best tuned parameters: k = 5
loess r-squared variable importance

  only 20 most important variables shown (out of 35)

                    Overall
`Mnf Flow`          100.000
`Usage cont`         75.213
`Bowl Setpoint`      57.596
`Filler Level`       48.134
`Pressure Setpoint`  42.313
`Carb Flow`          40.527
`Brand Code`C        32.227
`Hyd Pressure3`      28.083
`Pressure Vacuum`    23.499
`Fill Pressure`      22.177
`Hyd Pressure2`      19.735
`Brand Code`D        16.597
`Oxygen Filler`      15.200
`Carb Rel`           12.476
Temperature          11.797
`Alch Rel`           11.470
`Hyd Pressure4`       8.423
`Balling Lvl`         5.306
`PC Volume`           4.745
`Brand Code`A         4.152
```

SVM

Support Vector Machines (SVM) is a supervised learning algorithm that can be used for classification or regression. It works by finding the hyperplane that best separates the data into different classes. In this case, we are using it for regression.

Linear

Dataset A

```
model <- train(
  x = dataA$train_X,
  y = dataA$train_y,
  method = "svmLinear",
  tuneLength = 10,
  trControl = trainControl(
    method = "cv",
    number = 10
  )
)
report(model, dataA)
```

```
Model: svmLinear, RMSE: 0.15, R2: 0.24
Best tuned parameters: C = 1
loess r-squared variable importance

     Overall
PC2  100.000
PC4   38.874
PC9   25.462
PC3   21.056
PC18  20.220
PC1   18.452
PC6   11.013
PC16   9.018
PC12   7.658
PC8    6.714
PC10   6.045
PC5    5.377
PC15   5.171
PC11   4.305
```

```
PC14   2.246
PC17   1.545
PC19   1.433
PC7    1.350
PC13   0.000
```

Dataset B

```r
model <- train(
  x = dataB$train_X,
  y = dataB$train_y,
  method = "svmLinear",
  tuneLength = 10,
  trControl = trainControl(
    method = "cv",
    number = 10
  )
)
report(model, dataB)
```

```
Model: svmLinear, RMSE: 0.14, R2: 0.34
Best tuned parameters: C = 1
loess r-squared variable importance

  only 20 most important variables shown (out of 35)

                    Overall
`Mnf Flow`          100.000
`Usage cont`         75.213
`Bowl Setpoint`      57.596
`Filler Level`       48.134
`Pressure Setpoint`  42.313
`Carb Flow`          40.527
`Brand Code`C        32.227
`Hyd Pressure3`      28.083
`Pressure Vacuum`    23.499
`Fill Pressure`      22.177
`Hyd Pressure2`      19.735
`Brand Code`D        16.597
`Oxygen Filler`      15.200
`Carb Rel`           12.476
```

```
Temperature          11.797
`Alch Rel`           11.470
`Hyd Pressure4`       8.423
`Balling Lvl`         5.306
`PC Volume`           4.745
`Brand Code`A         4.152
```

Polynomial

Dataset A

```
model <- train(
  x = dataA$train_X,
  y = dataA$train_y,
  method = "svmPoly",
  tuneLength = 3,
  trControl = trainControl(
    method = "cv",
    number = 6
  )
)
report(model, dataA)
```

```
Model: svmPoly, RMSE: 0.14, R2: 0.33
Best tuned parameters: degree = 3
Best tuned parameters: scale = 0.01
Best tuned parameters: C = 1
loess r-squared variable importance

     Overall
PC2  100.000
PC4   38.874
PC9   25.462
PC3   21.056
PC18  20.220
PC1   18.452
PC6   11.013
PC16   9.018
PC12   7.658
PC8    6.714
PC10   6.045
```

```
PC5    5.377
PC15   5.171
PC11   4.305
PC14   2.246
PC17   1.545
PC19   1.433
PC7    1.350
PC13   0.000
```

Dataset B

```r
model <- train(
  x = dataB$train_X,
  y = dataB$train_y,
  method = "svmPoly",
  tuneLength = 3,
  trControl = trainControl(
    method = "cv",
    number = 6
  )
)
report(model, dataB)
```

```
Model: svmPoly, RMSE: 0.12, R2: 0.46
Best tuned parameters: degree = 3
Best tuned parameters: scale = 0.01
Best tuned parameters: C = 1
loess r-squared variable importance

  only 20 most important variables shown (out of 35)

                    Overall
`Mnf Flow`          100.000
`Usage cont`         75.213
`Bowl Setpoint`      57.596
`Filler Level`       48.134
`Pressure Setpoint`  42.313
`Carb Flow`          40.527
`Brand Code`C        32.227
`Hyd Pressure3`      28.083
`Pressure Vacuum`    23.499
```

```
`Fill Pressure`      22.177
`Hyd Pressure2`      19.735
`Brand Code`D        16.597
`Oxygen Filler`      15.200
`Carb Rel`           12.476
Temperature          11.797
`Alch Rel`           11.470
`Hyd Pressure4`       8.423
`Balling Lvl`         5.306
`PC Volume`           4.745
`Brand Code`A         4.152
```

Radial

Dataset A

```
model <- train(
  x = dataA$train_X,
  y = dataA$train_y,
  method = "svmRadial",
  tuneLength = 5,
  trControl = trainControl(
    method = "cv",
    number = 10
  )
)
report(model, dataA)
```

```
Model: svmRadial, RMSE: 0.13, R2: 0.42
Best tuned parameters: sigma = 0.0349324581481468
Best tuned parameters: C = 2
loess r-squared variable importance

     Overall
PC2  100.000
PC4   38.874
PC9   25.462
PC3   21.056
PC18  20.220
PC1   18.452
PC6   11.013
```

```
PC16   9.018
PC12   7.658
PC8    6.714
PC10   6.045
PC5    5.377
PC15   5.171
PC11   4.305
PC14   2.246
PC17   1.545
PC19   1.433
PC7    1.350
PC13   0.000
```

Dataset B

```r
model <- train(
  x = dataB$train_X,
  y = dataB$train_y,
  method = "svmRadial",
  tuneLength = 5,
  trControl = trainControl(
    method = "cv",
    number = 10
  )
)
report(model, dataB)
```

```
Model: svmRadial, RMSE: 0.11, R2: 0.54
Best tuned parameters: sigma = 0.0203106799787865
Best tuned parameters: C = 4
loess r-squared variable importance

  only 20 most important variables shown (out of 35)

                    Overall
`Mnf Flow`          100.000
`Usage cont`         75.213
`Bowl Setpoint`      57.596
`Filler Level`       48.134
`Pressure Setpoint`  42.313
`Carb Flow`          40.527
```

```
`Brand Code`C        32.227
`Hyd Pressure3`      28.083
`Pressure Vacuum`    23.499
`Fill Pressure`      22.177
`Hyd Pressure2`      19.735
`Brand Code`D        16.597
`Oxygen Filler`      15.200
`Carb Rel`           12.476
Temperature          11.797
`Alch Rel`           11.470
`Hyd Pressure4`       8.423
`Balling Lvl`         5.306
`PC Volume`           4.745
`Brand Code`A         4.152
```

MARS

Multivariate Adaptive Regression Splines (MARS) is a non-parametric regression technique that can be used for both linear and non-linear regression. It works by fitting piecewise linear functions to the data.

Dataset A

```r
grid = expand.grid(
  degree = 1:2,
  nprune = 5:10
)
model <- train(
  x = dataA$train_X,
  y = dataA$train_y,
  method = "earth",
  tuneGrid = grid,
  trControl = trainControl(
    method = "cv",
    number = 10
  )
)
report(model, dataA)
```

```
Model: earth, RMSE: 0.14, R2: 0.27
Best tuned parameters: nprune = 10
```

```
Best tuned parameters: degree = 2
earth variable importance

     Overall
PC2   100.00
PC4    65.58
PC1    47.44
PC16   38.83
PC18   31.64
PC11   26.15
PC12   25.31
PC7    10.28
PC19    0.00
```

Dataset B

```r
grid = expand.grid(
  degree = 1:2,
  nprune = 5:10
)
model <- train(
  x = dataB$train_X,
  y = dataB$train_y,
  method = "earth",
  tuneGrid = grid,
  trControl = trainControl(
    method = "cv",
    number = 10
  )
)
report(model, dataB)
```

```
Model: earth, RMSE: 0.14, R2: 0.34
Best tuned parameters: nprune = 10
Best tuned parameters: degree = 1
earth variable importance

                        Overall
`\\`Mnf Flow\\``        100.000
`\\`Brand Code\\`C`      61.570
Temperature              44.185
```

```
`\\`Usage cont\\``       38.343
`\\`Pressure Vacuum\\``  29.961
Balling                  23.656
`\\`Alch Rel\\``         18.238
`\\`Carb Pressure1\\``    7.678
`\\`Bowl Setpoint\\``     0.000
```

Neural Network

Neural Networks are a class of models that are inspired by the way the human brain works. They are used for both classification and regression tasks. In this case, we are using it for regression.

Dataset A

```r
grid <- expand.grid(
  size = c(2, 4, 6, 8, 10),
  decay = c(0, 0.05, 0.1, 0.15)
)
model <- train(
  x = dataA$train_X,
  y = dataA$train_y,
  method = "nnet",
  linout = TRUE,
  trace = FALSE,
  maxit = 1000,
  tuneGrid = grid,
  trControl = trainControl(
    method="cv",
    number=5
  )
)
report(model, dataA)
```

```
Model: nnet, RMSE: 0.13, R2: 0.39
Best tuned parameters: size = 6
Best tuned parameters: decay = 0.1
nnet variable importance

     Overall
PC8  100.000
```

```
PC17  93.232
PC2   78.737
PC9   74.722
PC19  74.567
PC4   61.135
PC6   57.581
PC13  37.083
PC5   35.818
PC12  30.678
PC16  24.214
PC15  21.450
PC14   9.324
PC11   8.351
PC10   6.762
PC1    6.432
PC7    3.555
PC18   3.366
PC3    0.000
```

Dataset B

```r
grid <- expand.grid(
  size = c(2, 4, 6, 8, 10),
  decay = c(0, 0.05, 0.1, 0.15)
)
model <- train(
  x = dataB$train_X,
  y = dataB$train_y,
  method = "nnet",
  linout = TRUE,
  trace = FALSE,
  maxit = 1000,
  tuneGrid = grid,
  trControl = trainControl(
    method="cv",
    number=5
  )
)
report(model, dataB)
```

```
Model: nnet, RMSE: 0.12, R2: 0.5
```

```
Best tuned parameters: size = 8
Best tuned parameters: decay = 0.15
nnet variable importance

  only 20 most important variables shown (out of 35)

                    Overall
`Carb Rel`           100.00
Density               99.56
`Hyd Pressure2`       96.35
`Hyd Pressure1`       94.80
`Pressure Vacuum`     87.58
`Carb Flow`           84.68
`Pressure Setpoint`   83.08
`Alch Rel`            77.01
`Bowl Setpoint`       76.51
`Mnf Flow`            75.06
`Brand Code`C         67.49
Temperature           51.90
`Hyd Pressure3`       51.87
`Air Pressurer`       47.70
`Usage cont`          46.77
`Balling Lvl`         43.45
Balling               42.83
`Oxygen Filler`       40.67
`Filler Level`        39.36
`Brand Code`D         36.85
```