COVID 19 - Dashboard
R Shiny

1. DATA CLEANING

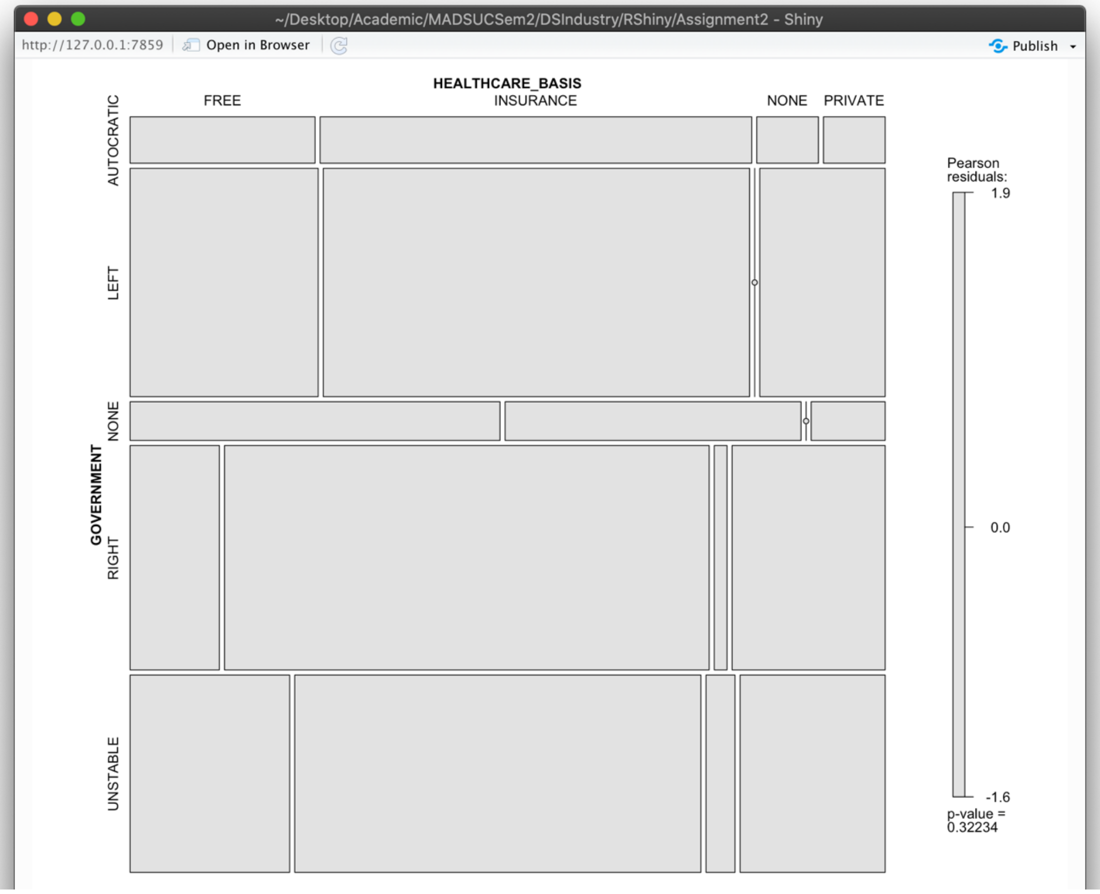
- All the **missing values** in the dataset are **replaced with NA's** of the corresponding data type.The healthcare cost per person has been found missing in all the instances where the healthcare system is **"FREE".** Logically this is incorrect. Hence the corresponding healthcare cost for all such instances are **replaced by "0"** (Numeric) as those are funded by the government.

- The dataset comprises **missing of categorical** variables – Government Type and Healthcare System. The values are **replaced by "NONE"** since the percentage of missing of these variables are low.

No	Variable	Stats / Values	Freqs (% of Valid)	Graph	Valid	Missing
2	GOVERNMENT [factor]	1. AUTOCRATIC 2. LEFT 3. NONE 4. RIGHT 5. UNSTABLE	12 (6.3%) 59 (31.1%) 10 (5.3%) 58 (30.5%) 51 (26.8%)	I IIIIII I IIIIII IIIII	190 (100%)	0 (0%)
12	HEALTHCARE_BASIS [factor]	1. FREE 2. INSURANCE 3. NONE 4. PRIVATE	41 (21.6%) 111 (58.4%) 4 (2.1%) 34 (17.9%)	IIII IIIIIIIIIII III	190 (100%)	0 (0%)

2. EXPLORATORY ANALYSIS

- **Mosaic Plot**



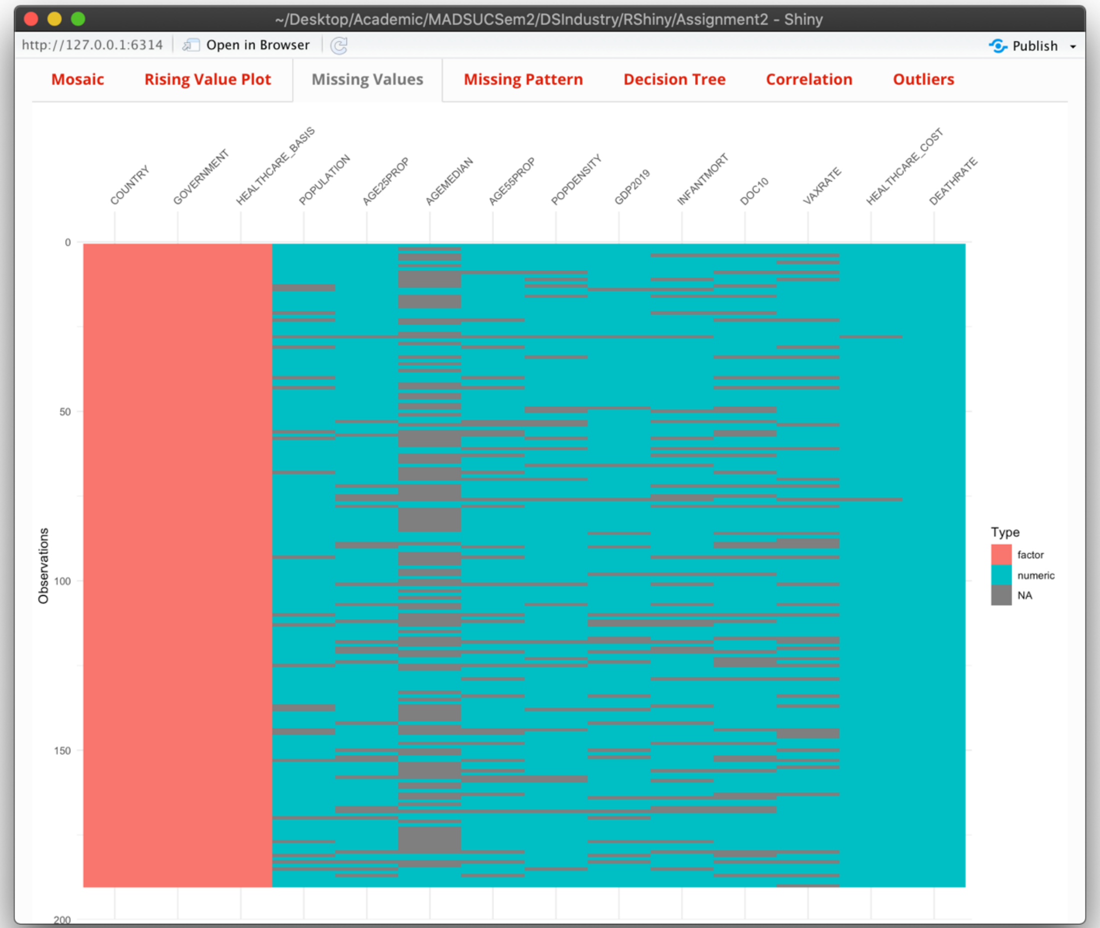
From the plot it can be seen that the proportion of **insurance payers is most** for all the country types.

- **Rising Value Plot**

The fields containing numeric values are plotted in a rising chart to observe their continuity. There is clear **discontinuity in GDP and Healthcare cost** both of which are **associated to finance**.

CONTINUOUS FIELDS	DISCONTINUOUS FIELDS
POPULATION	GDP2019
AGEMEDIAN	VAXRATE
AGE25PROP	HEALTHCARE_COST
AGE55PROP	
POPDENSITY	
INFANTMORT	
DOC10	

- **Missingness**

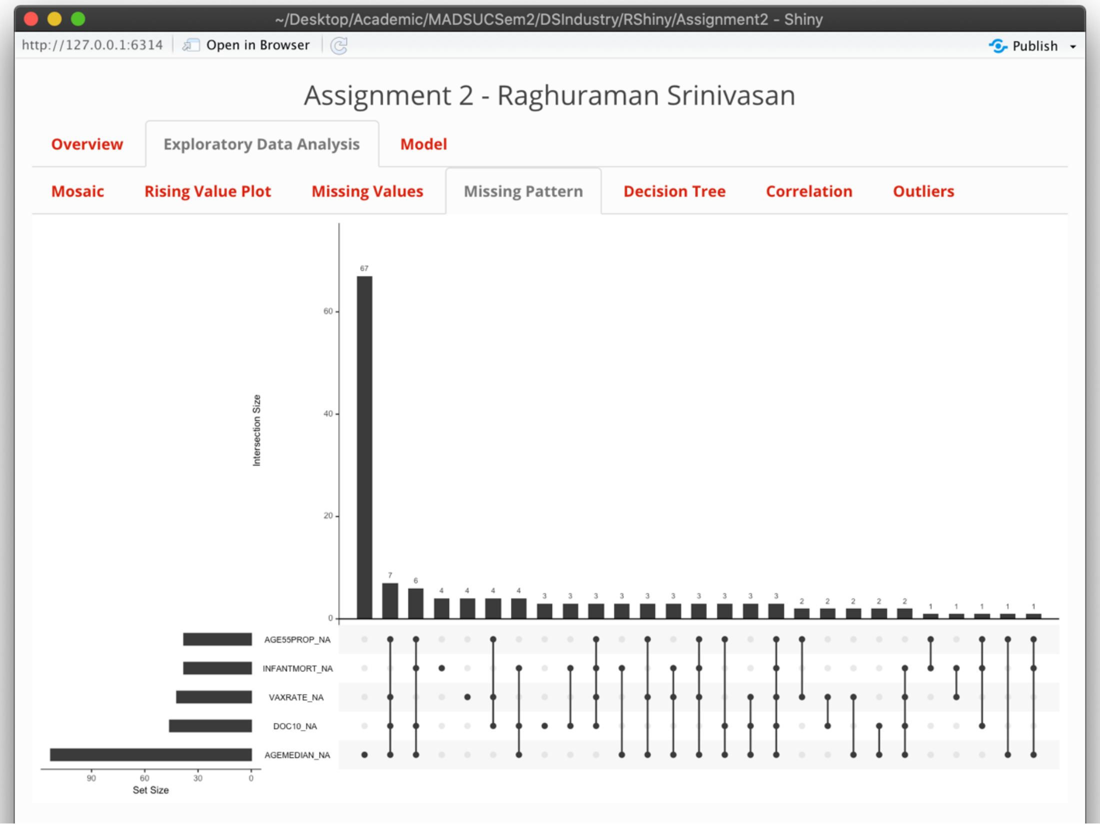



```
No   Variable           Stats / Values                Freqs (% of Valid)   Graph                                  Valid      Missing
```

```
5    AGEMEDIAN          Mean (sd) : 34.4 (2)         77 distinct values            :                          77          113
     [numeric]          min < med < max:                                         . :                          (40.53%)    (59.47%)
                        29.5 < 34.6 < 39.4                                     : : :
                        IQR (CV) : 2.6 (0.1)                                 : : : : : :
                                                                         :   : : : : : :   :
```

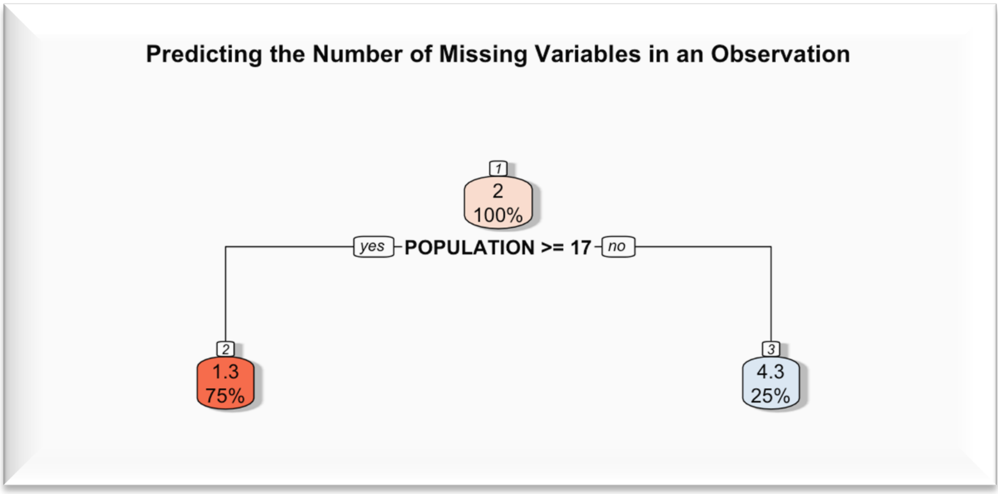
From the missing value plot, it can be seen that numeric variables are missing with **"AGEMEDIAN" missing the most (59.47%)** among them.

There **isn't ample number of observations** and the **range is small** with **median of range approximately same as the mean** of the observations. So, it is better to keep AGEMEDIAN and impute the missing values rather than discarding these, considering in mind that the **glmnet model (Regression) is tolerant with missing values**.



From the missing pattern plot it is evident that the values are not missing completely at random (MCAR). Also, there is not any strong evidence to say that the values are missing not at random (MNAR). Hence it is safe to assume that the **majority of this dataset is missing at random (MAR)**. The most missing field is AGEMEDIAN with 60% of its values are MCAR and 40% are MAR.

Since most of the missing can be classified as MAR and MCAR in that order, **partial deletion** is not the best preprocessing solution as it **leads to a biased model**. Hence **imputation can be applied** to the dataset as it **leads to unbiased model** for both missing types MCAR and MAR considering the previous analysis outcomes in mind.

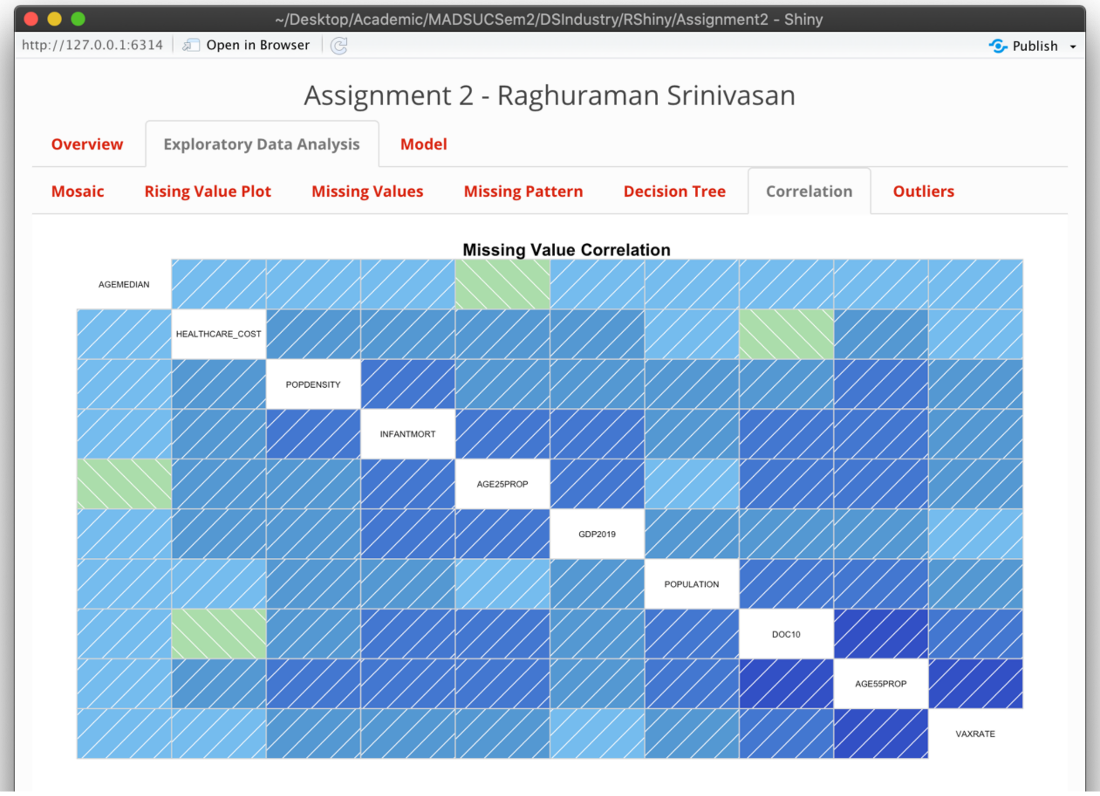


From the decision tree output for predicting missing variables in an observation, the tree has identified the **missingness explained well around "POPULATION".**

Whenever POPULATION is 17 or more, it comprises 75% of observation with just approximately 1 missing value in every observation. On the other hand, whenever POPULATION is below 17, it comprises 25% of the total observation with approximately 4 missing values in every observation.

Hence **POPULATION can be chosen as the best parameter to define an optimal split.**

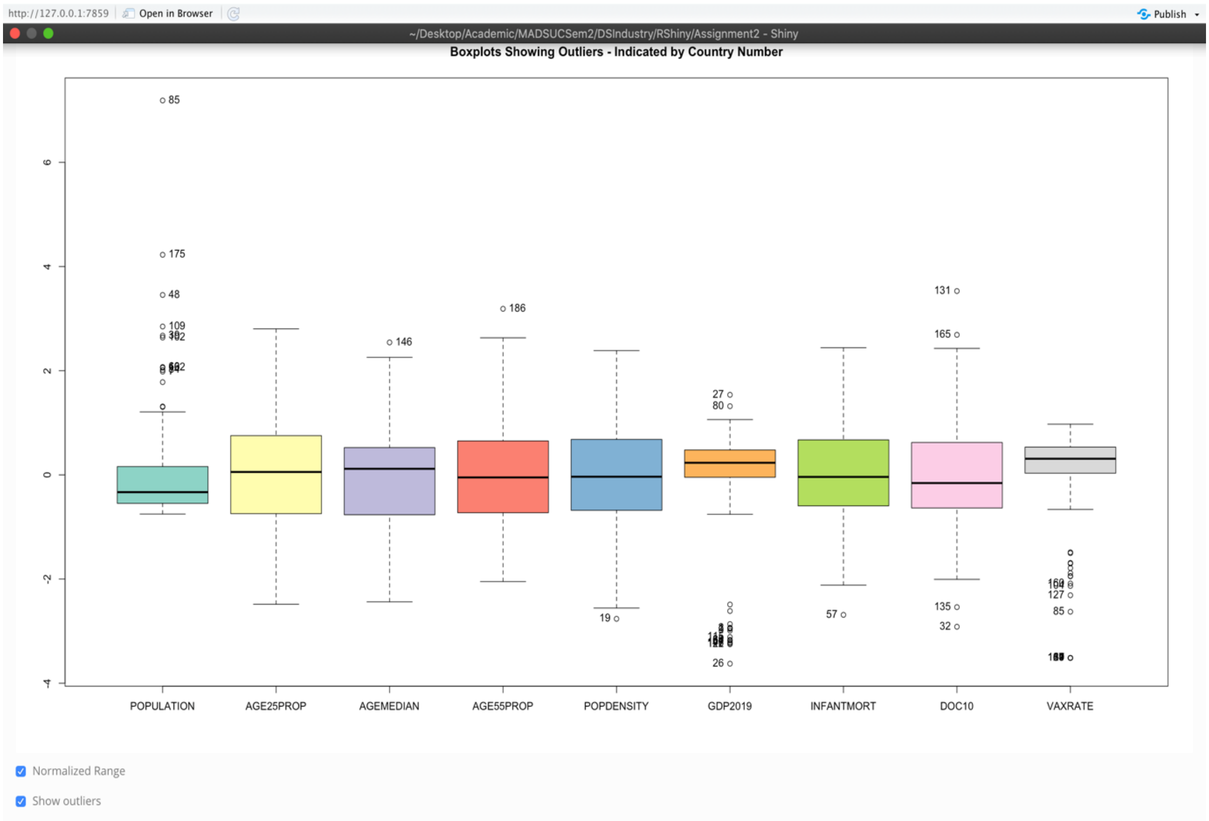
- **Correlation Chart**



The vaccination rate is correlated with doctors per 10K people which is logically supports the fact that more the number of doctors more people can be vaccinated. Also, vaccination rate correlates with people aged above 55 as they are more probable to have got vaccinated considering the longevity of their life.

Surprisingly healthcare cost is not correlated with number of **doctors per 10K** people which needs to be **followed up with a domain expert**. Finally, Age Median is not so strongly correlated with any other field explained due their missingness in majority of the observations.

- **Outliers**



Clearly **vaccination rate and GDP have similar cluster of outliers** outside the lower quartile along with few matchings from doctors available for every 10K people indicating a correlation. This very well **matches the results from the correlation chart**.

And for most other variables, there is little or no outliers indicating that **imputing** the values will be a **wise choice than partial deletion**.

3. PREPROCESSING

- **Near Zero Variance Table**

	freqRatio	percentUnique	zeroVar	nzv
COUNTRY	1.000000	100.000000	FALSE	FALSE
GOVERNMENT	1.017241	2.631579	FALSE	FALSE
POPULATION	1.000000	86.842105	FALSE	FALSE
AGE25PROP	1.000000	85.263158	FALSE	FALSE
AGEMEDIAN	1.000000	40.526316	FALSE	FALSE
AGE55PROP	1.000000	80.000000	FALSE	FALSE
POPDENSITY	1.000000	86.315789	FALSE	FALSE
GDP2019	1.000000	86.315789	FALSE	FALSE
INFANTMORT	1.000000	80.000000	FALSE	FALSE
DOC10	1.000000	75.789474	FALSE	FALSE
VAXRATE	6.000000	75.263158	FALSE	FALSE
HEALTHCARE_BASIS	2.707317	2.105263	FALSE	FALSE
HEALTHCARE_COST	41.000000	77.894737	FALSE	FALSE

The **"nzv" value for all the predictors is "FALSE"** and a high unique value percentage for most of the predictors (Except categorical variables with low cardinality which seems to be fine). Hence all the predictors can be utilized in imputing and modeling and there is no **need to discard them**.

- **Recipe Based Pipeline**

Name	Type	Value
rec	list [6] (S3: recipe)	List of length 6
var_info	list [14 x 4] (S3: tbl_df, tbl, dε	A tibble with 14 rows and 4 columns
term_info	list [14 x 4] (S3: tbl_df, tbl, dε	A tibble with 14 rows and 4 columns
steps	list [4]	List of length 4
template	list [119 x 14] (S3: tbl_df, tbl,	A tibble with 119 rows and 14 columns
levels	NULL	Pairlist of length 0
retained	logical [1]	NA

The dataset has **been split into train and test based on the "POPULATION"** condition from decision tree. The train data is used to develop a recipe-based processing pipeline (centered and scaled) to use in glmnet model.

4. MODELING

- **GLMNET Model**

Glmnet is a method based on the generalized linear model and it is used to **perform Ridge, Lasso and Elastic Net Regression** in R. Glment can **be applied to both linear regression and logistic regression**. The elastic net combines lasso regression penalty lambda1 with the ridge regression penalty lambda. But glmnet model has two parameters lambda and alpha.

The range of alpha is from 0 to 1. When alpha = 0 lasso penalty becomes 0 and goes away. When alpha = 1 ridge penalty becomes 0 and goes away. When alpha is neither 0 nor 1 we get a mixture of both the penalties that does a better job shrinking correlated variables compared to either lasso or ridge on their own.

Lambda controls how much of the penalty should be applied to the regression. When lambda is 0, both lasso and ridge penalty go away. It means glmnet only performs either standard least square for linear regression or maximum likelihood for logistic regression.

When performing glmnet for regression **we test for different values for lambda and alpha** ie. **Tuning** to **avoid over-fitting** of the training data which is its **biggest advantage**.

- **Tuning Parameters**

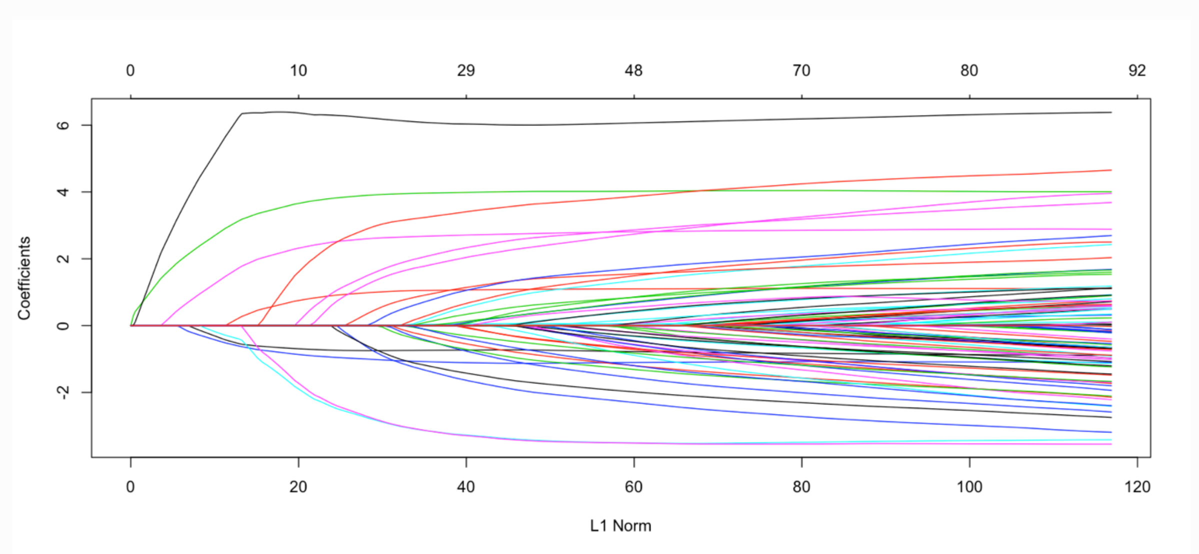
NZV Predictors | GLMNET Train | GLMNET Test

Best Tuning Parameters:

alpha	lambda
0.55	0.29

The best tuning parameters alpha and lambda for optimizing glmnet model. This is obtained by **cross validating the train data for 10 folds.**

- **Glmnet Train Plot**

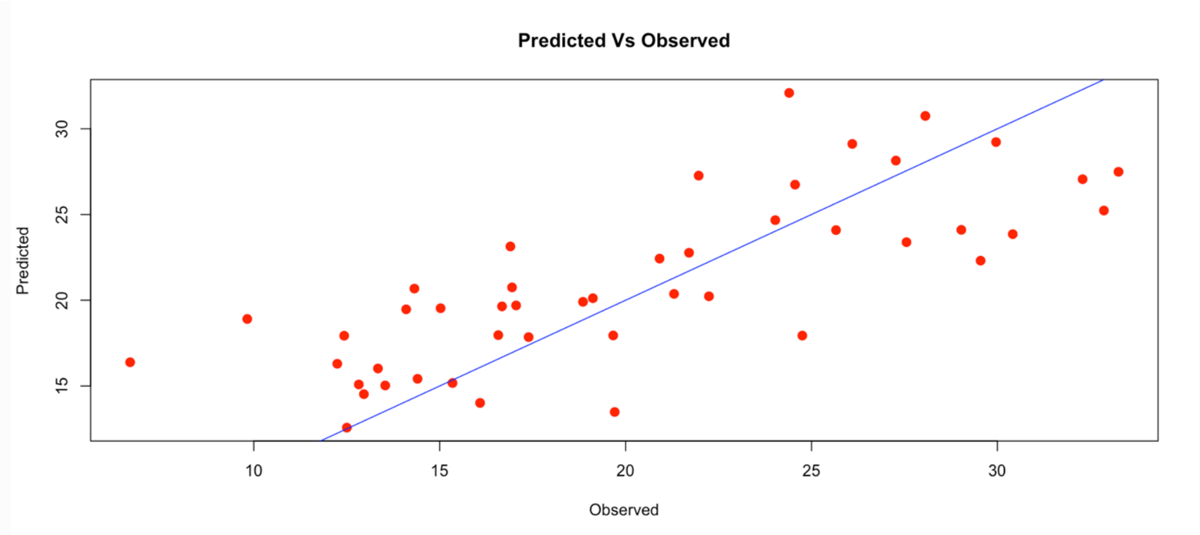


The best model uses **L1 Norms regularization ie. Lasso Regressio**n. Lasso has trained the model by **shrinking coefficient of less important features to zero** as seen from the converging lines to "0" in the plot.

- **Glmnet Train Summary**

```
glmnet

119 samples
 13 predictor

Recipe steps: knnimpute, center, scale, dummy
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 107, 108, 107, 107, 107, 107, ...
Resampling results across tuning parameters:

  alpha  lambda     RMSE      Rsquared   MAE
  0.10   0.2891590  2.505623  0.9428839  2.109598
  0.10   0.9144011  2.626130  0.9410719  2.200919
  0.10   2.8915902  3.238355  0.9324987  2.675774
  0.55   0.2891590  1.857308  0.9486019  1.480643
  0.55   0.9144011  2.392638  0.9346689  1.942559
  0.55   2.8915902  4.592915  0.8457414  3.834655
  1.00   0.2891590  1.995116  0.9393243  1.570372
  1.00   0.9144011  3.039025  0.8971834  2.476123
  1.00   2.8915902  6.055669  0.6373946  5.033972

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were alpha = 0.55 and lambda = 0.289159.
```

- **Glmnet Test Plot**



The data split is done in such a way using decision trees so that the testing set (25% observations) has more missing observations (Average of 4 values missing in every observation) and the training set (75% observations) has less missing observations (Average of only 1 value missing in every observation).

The training dataset has been imputed with missing values using KNN impute in a recipe-based pipeline. In prediction value summary it is seen that there are no absolute outliers.

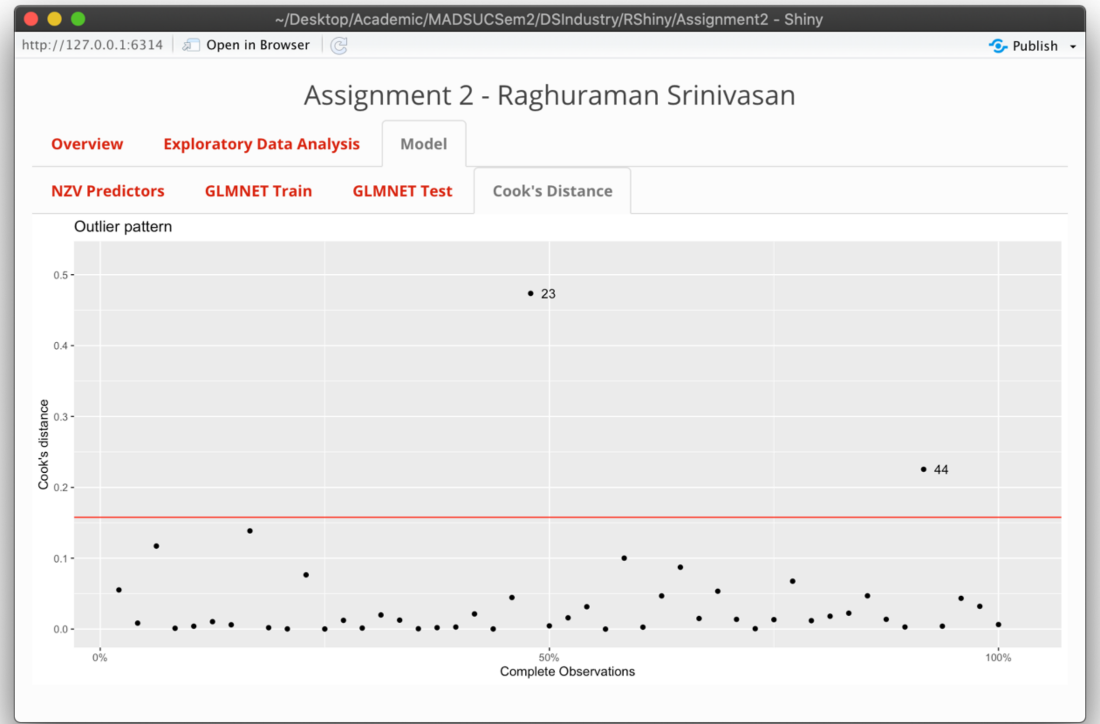
```
Preparing recipe
+ Fold01: alpha=0.10, lambda=2.892
- Fold01: alpha=0.10, lambda=2.892
+ Fold01: alpha=0.55, lambda=2.892
- Fold01: alpha=0.55, lambda=2.892
+ Fold01: alpha=1.00, lambda=2.892
- Fold01: alpha=1.00, lambda=2.892
+ Fold02: alpha=0.10, lambda=2.892
- Fold02: alpha=0.10, lambda=2.892
+ Fold02: alpha=0.55, lambda=2.892
- Fold02: alpha=0.55, lambda=2.892
+ Fold02: alpha=1.00, lambda=2.892
- Fold02: alpha=1.00, lambda=2.892
+ Fold03: alpha=0.10, lambda=2.892
- Fold03: alpha=0.10, lambda=2.892
+ Fold03: alpha=0.55, lambda=2.892
- Fold03: alpha=0.55, lambda=2.892
+ Fold03: alpha=1.00, lambda=2.892
```

```
- Fold03: alpha=1.00, lambda=2.892
+ Fold04: alpha=0.10, lambda=2.892
- Fold04: alpha=0.10, lambda=2.892
+ Fold04: alpha=0.55, lambda=2.892
- Fold04: alpha=0.55, lambda=2.892
+ Fold04: alpha=1.00, lambda=2.892
- Fold04: alpha=1.00, lambda=2.892
+ Fold05: alpha=0.10, lambda=2.892
- Fold05: alpha=0.10, lambda=2.892
+ Fold05: alpha=0.55, lambda=2.892
- Fold05: alpha=0.55, lambda=2.892
+ Fold05: alpha=1.00, lambda=2.892
- Fold05: alpha=1.00, lambda=2.892
+ Fold06: alpha=0.10, lambda=2.892
- Fold06: alpha=0.10, lambda=2.892
+ Fold06: alpha=0.55, lambda=2.892
- Fold06: alpha=0.55, lambda=2.892
+ Fold06: alpha=1.00, lambda=2.892
- Fold06: alpha=1.00, lambda=2.892
+ Fold07: alpha=0.10, lambda=2.892
- Fold07: alpha=0.10, lambda=2.892
+ Fold07: alpha=0.55, lambda=2.892
- Fold07: alpha=0.55, lambda=2.892
+ Fold07: alpha=1.00, lambda=2.892
- Fold07: alpha=1.00, lambda=2.892
+ Fold08: alpha=0.10, lambda=2.892
- Fold08: alpha=0.10, lambda=2.892
+ Fold08: alpha=0.55, lambda=2.892
- Fold08: alpha=0.55, lambda=2.892
+ Fold08: alpha=1.00, lambda=2.892
- Fold08: alpha=1.00, lambda=2.892
+ Fold09: alpha=0.10, lambda=2.892
- Fold09: alpha=0.10, lambda=2.892
+ Fold09: alpha=0.55, lambda=2.892
- Fold09: alpha=0.55, lambda=2.892
+ Fold09: alpha=1.00, lambda=2.892
- Fold09: alpha=1.00, lambda=2.892
+ Fold10: alpha=0.10, lambda=2.892
- Fold10: alpha=0.10, lambda=2.892
+ Fold10: alpha=0.55, lambda=2.892
```

```
- Fold10: alpha=0.55, lambda=2.892
+ Fold10: alpha=1.00, lambda=2.892
- Fold10: alpha=1.00, lambda=2.892
Aggregating results
Selecting tuning parameters
Fitting alpha = 0.55, lambda = 0.289 on full training set
    Observed Predicted
1  16.675170  19.64306
2  22.242820  20.23031
3  12.501690  12.56787
4  24.753643  17.93770
5  27.555380  23.38678
6  19.665768  17.94672
7  13.338310  16.02107
8  18.854293  19.90733
9  21.303688  20.37099
10 16.946015  20.75006
11 32.867513  25.23355
12 17.054077  19.69720
13 16.900950  23.13870
14 28.063069  30.75092
15 12.243724  16.29618
16 30.414610  23.85546
17 20.913728  22.43074
18  9.822761  18.90808
19 12.823278  15.08781
20 15.344297  15.17234
21 12.429191  17.93062
22 15.023312  19.53365
23 25.660262  24.09134
24 14.318566  20.68039
25 24.558096  26.74194
26 27.270671  28.14492
27 24.027002  24.67136
28 29.964555  29.22737
29 32.294027  27.05903
30 13.533629  15.02938
31 19.707101  13.48238
32 16.085410  14.01002
33 12.960282  14.52264
```

```
34 14.403706  15.41727
35 14.097057  19.47015
36 17.391203  17.85027
37 21.706630  22.77005
38  6.671895  16.38343
39 26.097741  29.11745
40 33.258590  27.49279
41 29.031149  24.10565
42 16.575425  17.96448
43 29.548626  22.30892
44 21.965865  27.26674
45 24.399416  32.09219
46 19.119807  20.11723
```

- Residual Outliers



Cook's distance is used to identify the residual outliers. It is a measure of how much a linear regression is affected by each observation. The input is the KNN imputed train data set generated using recipe-based pipeline. The prerequisite for cook's distance to spot residual outliers are dataset contain only numeric variables. It does not contain any missing values. All the variables are non NZV's as evident from the NZV table. Strongly correlated variables are removed.

Two residual outliers are found (23,46) which have strong influence over linear regression.