Predicting the Critical Temperature of a Superconductor ¶

Monica

monica110394@gmail.com (mailto:monica110394@gmail.com)

Programming Language: R 3.5.1 in Jupyter Notebook

R Libraries used:

- psych
- purrr
- tidyr
- ggplot2
- reshape2
- gridExtra
- cowplot
- ggpubr
- mlbench
- caret
- FSelector
- mRMRe
- leaps
- car
- glmnet
- randomForest
- xgboost
- readr
- stringr

Table of Contents

1. Introduction

Superconductors have been in great use in every industry today. However the wide spread applications of the superconductors have been held back by two major issues, it can conduct current with zero resistance only at or below it superconducting critical temperature and the second issue is that the scientific model and theory that predicts critical temperature is still an open problem. There are not many theory based prediction studies on this yet. Hence we have been provided with a research paper by Kam Hamidieh, which is one of the few studies on this topic. Hence, for this assignment, we will take an entirely data driven approach to create a statistical model that predicts the critical temperature.

For this assignment we will first do an EDA, after whichwe will filter some of the variables out of the 81 variables, and then we can use the filtered set of features with 3 types of feature selection methods i.e. first, linear regression with stepwise feature selection process, second, embedded feature selection models and the third is Gradient Boosting Feature selection models.

2. Exploratory Data Analysis:

Before the start of Exploratory Data Analysis, it is very important to identify the possible errors in the dataset, for example, outliers. After removing the obvious errors like missing values and duplicates, performing EDA can help to us to get rid of the extra errors that cannot be seen by just looking at the dataset and help us to perform statistical analysis on the given dataset and hence visualisation comes into play. Also, EDA helps us to answer main questions like:

- What is the type of variation occuring within the variables
- What type of covariation is occuring between the variables

2.1 Overview of the Training Dataset:

Reading the dataset $train.csv$ in R. $header = T$ is by default.

In [1]:

```
# loading the training dataset
trainingData = read.csv("train.csv", header = T)
```

Let us find out the dimensions of the dataset, i.e. number of records and number of attributes.

In [2]:

```
# displaying the dimension of the dataset
cat("The training dataset has", dim(trainingData)[1], "records, each with", dim(trainingDat
    "attributes.")
```

```
The training dataset has 21263 records, each with 82 attributes.
```

To find the what our dataset looks like we can use the $str()$ function. It is basically displays the internal structure of any R object. It can be used as an alternative to the function $summary()$ to some extent.

In [3]:

```
# displaying the structure of the dataset
cat("The structure of the dataset is->\n\n")
str(trainingData)
```

```
The structure of the dataset is->

'data.frame':   21263 obs. of  82 variables:
 $ number_of_elements            : int  4 5 4 4 4 4 4 4 4 4 ...
 $ mean_atomic_mass              : num  88.9 92.7 88.9 88.9 88.9 ...
 $ wtd_mean_atomic_mass          : num  57.9 58.5 57.9 57.9 57.8 ...
 $ gmean_atomic_mass             : num  66.4 73.1 66.4 66.4 66.4 ...
 $ wtd_gmean_atomic_mass         : num  36.1 36.4 36.1 36.1 36.1 ...
 $ entropy_atomic_mass           : num  1.18 1.45 1.18 1.18 1.18 ...
 $ wtd_entropy_atomic_mass       : num  1.062 1.058 0.976 1.022 1.129 ...
 $ range_atomic_mass             : num  123 123 123 123 123 ...
 $ wtd_range_atomic_mass         : num  31.8 36.2 35.7 33.8 27.8 ...
 $ std_atomic_mass               : num  52 47.1 52 52 52 ...
 $ wtd_std_atomic_mass           : num  53.6 54 53.7 53.6 53.6 ...
 $ mean_fie                      : num  775 766 775 775 775 ...
 $ wtd_mean_fie                  : num  1010 1011 1011 1011 1010 ...
 $ gmean_fie                     : num  718 721 718 718 718 ...
 $ wtd_gmean_fie                 : num  938 939 939 939 937 ...
 $ entropy_fie                   : num  1.31 1.54 1.31 1.31 1.31 ...
 $ wtd_entropy_fie               : num  0.791 0.807 0.774 0.783 0.805 ...
 $ range_fie                     : num  811 811 811 811 811 ...
 $ wtd_range_fie                 : num  736 743 743 740 729 ...
 $ std_fie                       : num  324 290 324 324 324 ...
 $ wtd_std_fie                   : num  356 355 355 355 356 ...
 $ mean_atomic_radius            : num  160 161 160 160 160 ...
 $ wtd_mean_atomic_radius        : num  106 105 105 105 106 ...
 $ gmean_atomic_radius           : num  136 141 136 136 136 ...
 $ wtd_gmean_atomic_radius       : num  84.5 84.4 84.2 84.4 84.8 ...
 $ entropy_atomic_radius         : num  1.26 1.51 1.26 1.26 1.26 ...
 $ wtd_entropy_atomic_radius     : num  1.21 1.2 1.13 1.17 1.26 ...
 $ range_atomic_radius           : int  205 205 205 205 205 205 205 171 171
171 ...
 $ wtd_range_atomic_radius       : num  42.9 50.6 49.3 46.1 36.5 ...
 $ std_atomic_radius             : num  75.2 67.3 75.2 75.2 75.2 ...
 $ wtd_std_atomic_radius         : num  69.2 68 67.8 68.5 70.6 ...
 $ mean_Density                  : num  4654 5821 4654 4654 4654 ...
 $ wtd_mean_Density              : num  2962 3021 2999 2980 2924 ...
 $ gmean_Density                 : num  725 1237 725 725 725 ...
 $ wtd_gmean_Density             : num  53.5 54.1 54 53.8 53.1 ...
 $ entropy_Density               : num  1.03 1.31 1.03 1.03 1.03 ...
 $ wtd_entropy_Density           : num  0.815 0.915 0.76 0.789 0.86 ...
 $ range_Density                 : num  8959 10489 8959 8959 8959 ...
 $ wtd_range_Density             : num  1580 1667 1667 1623 1492 ...
 $ std_Density                   : num  3306 3767 3306 3306 3306 ...
 $ wtd_std_Density               : num  3573 3633 3592 3582 3553 ...
 $ mean_ElectronAffinity         : num  81.8 90.9 81.8 81.8 81.8 ...
 $ wtd_mean_ElectronAffinity     : num  112 112 112 112 111 ...
 $ gmean_ElectronAffinity        : num  60.1 69.8 60.1 60.1 60.1 ...
 $ wtd_gmean_ElectronAffinity    : num  99.4 101.2 101.1 100.2 97.8 ...
 $ entropy_ElectronAffinity      : num  1.16 1.43 1.16 1.16 1.16 ...
 $ wtd_entropy_ElectronAffinity  : num  0.787 0.839 0.786 0.787 0.787 ...
 $ range_ElectronAffinity        : num  127 127 127 127 127 ...
```

```
 $ wtd_range_ElectronAffinity     : num  81 81.2 81.2 81.1 80.8 ...
 $ std_ElectronAffinity           : num  51.4 49.4 51.4 51.4 51.4 ...
 $ wtd_std_ElectronAffinity       : num  42.6 41.7 41.6 42.1 43.5 ...
 $ mean_FusionHeat                : num  6.91 7.78 6.91 6.91 6.91 ...
 $ wtd_mean_FusionHeat            : num  3.85 3.8 3.82 3.83 3.87 ...
 $ gmean_FusionHeat               : num  3.48 4.4 3.48 3.48 3.48 ...
 $ wtd_gmean_FusionHeat           : num  1.04 1.04 1.04 1.04 1.04 ...
 $ entropy_FusionHeat             : num  1.09 1.37 1.09 1.09 1.09 ...
 $ wtd_entropy_FusionHeat         : num  0.995 1.073 0.927 0.964 1.045 ...
 $ range_FusionHeat               : num  12.9 12.9 12.9 12.9 12.9 ...
 $ wtd_range_FusionHeat           : num  1.74 1.6 1.76 1.74 1.74 ...
 $ std_FusionHeat                 : num  4.6 4.47 4.6 4.6 4.6 ...
 $ wtd_std_FusionHeat             : num  4.67 4.6 4.65 4.66 4.68 ...
 $ mean_ThermalConductivity       : num  108 172 108 108 108 ...
 $ wtd_mean_ThermalConductivity   : num  61 61.4 60.9 61 61.1 ...
 $ gmean_ThermalConductivity      : num  7.06 16.06 7.06 7.06 7.06 ...
 $ wtd_gmean_ThermalConductivity  : num  0.622 0.62 0.619 0.621 0.625 ...
 $ entropy_ThermalConductivity    : num  0.308 0.847 0.308 0.308 0.308 ...
 $ wtd_entropy_ThermalConductivity: num  0.263 0.568 0.25 0.257 0.273 ...
 $ range_ThermalConductivity      : num  400 430 400 400 400 ...
 $ wtd_range_ThermalConductivity  : num  57.1 51.4 57.1 57.1 57.1 ...
 $ std_ThermalConductivity        : num  169 199 169 169 169 ...
 $ wtd_std_ThermalConductivity    : num  139 140 139 139 138 ...
 $ mean_Valence                   : num  2.25 2 2.25 2.25 2.25 2.25 2.25 2.2
5 2.25 2.25 ...
 $ wtd_mean_Valence               : num  2.26 2.26 2.27 2.26 2.24 ...
 $ gmean_Valence                  : num  2.21 1.89 2.21 2.21 2.21 ...
 $ wtd_gmean_Valence              : num  2.22 2.21 2.23 2.23 2.21 ...
 $ entropy_Valence                : num  1.37 1.56 1.37 1.37 1.37 ...
 $ wtd_entropy_Valence            : num  1.07 1.05 1.03 1.05 1.1 ...
 $ range_Valence                  : int  1 2 1 1 1 1 1 1 1 1 ...
 $ wtd_range_Valence              : num  1.09 1.13 1.11 1.1 1.06 ...
 $ std_Valence                    : num  0.433 0.632 0.433 0.433 0.433 ...
 $ wtd_std_Valence                : num  0.437 0.469 0.445 0.441 0.429 ...
 $ critical_temp                  : num  29 26 19 22 23 23 11 33 36 31 ...
```

Looking at the attributes above, we can say that all the attributes are numerical and hence it will be better for us to perform a $multiple\ linear\ regression\ model$ rather than a logistic model. A few of our attributes is of datatype $integer$ and most of it is of $numeric$ datatype.

A glimpse of the $train.csv$ data is displayed below.

In [4]:

```
# displaying the first few records of the dataset
cat("Displaying the first few records of the training dataset->")
head(trainingData,5)
```

```
Displaying the first few records of the training dataset->
```

A data.frame: 5 × 82

	number_of_elements	mean_atomic_mass	wtd_mean_atomic_mass	gmean_atomic_mass	wtd_
	<int>	<dbl>	<dbl>	<dbl>	
1	4	88.94447	57.86269	66.36159	
2	5	92.72921	58.51842	73.13279	
3	4	88.94447	57.88524	66.36159	
4	4	88.94447	57.87397	66.36159	
5	4	88.94447	57.84014	66.36159	

2.2 Statistical Summary:

Now, let us check the $Statistical\ Summary$ of our dataset. The best way to do this is by using $describe()$. This function accepts a dataframe and can recognise the datatype of a variable, be it character, factor, numeric etc in the dataframe and gives us a brief statistical summary for each one of them. It gives information such as mean, median, min, max, range and so on.

In [6]:

```
# advanced descriptive statistics from the psych library or Statistical summary
# install.packages("psych")
library(psych)

# rounding the statistics to 3 decimal places and storing it as a dataframe
descDF<-round(describe(trainingData), 3)

# adding a column "Feature" to the dataframe so that it makes it easier for plotting
descDF$Feature<-rownames(descDF)

# printing the first few rows of the description dataset
descDF
```

```
Installing package into 'C:/Users/Monica/Documents/R/win-library/4.0'
(as 'lib' is unspecified)

also installing the dependencies 'tmvnsim', 'mnormt'



package 'tmvnsim' successfully unpacked and MD5 sums checked
package 'mnormt' successfully unpacked and MD5 sums checked
package 'psych' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\Monica\AppData\Local\Temp\RtmpqS3qvD\downloaded_packages
```

A psych: 82 × 14

	vars	n	mean	sd	median	trimmed	m
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dl
number_of_elements	1	21263	4.115	1.439	4.000	4.106	1.4
mean_atomic_mass	2	21263	87.558	29.676	84.923	85.819	19.9
wtd_mean_atomic_mass	3	21263	72.988	33.490	60.697	68.406	18.4
gmean_atomic_mass	4	21263	71.291	31.030	66.362	67.623	15.2
wtd_gmean_atomic_mass	5	21263	58.540	36.651	39.918	52.085	15.2
entropy_atomic_mass	6	21263	1.166	0.365	1.200	1.192	0.3
wtd_entropy_atomic_mass	7	21263	1.064	0.401	1.147	1.093	0.3
range_atomic_mass	8	21263	115.601	54.627	122.906	118.755	60.8
wtd_range_atomic_mass	9	21263	33.225	26.968	26.636	28.696	16.0
std_atomic_mass	10	21263	44.392	20.035	45.124	45.335	19.2
wtd_std_atomic_mass	11	21263	41.448	19.984	44.286	42.074	17.2
mean_fie	12	21263	769.615	87.489	764.900	763.041	53.6
wtd_mean_fie	13	21263	870.442	143.278	889.967	879.073	174.4
gmean_fie	14	21263	737.475	78.327	727.961	730.173	53.5
wtd_gmean_fie	15	21263	832.770	119.773	856.203	839.543	128.1
entropy_fie	16	21263	1.299	0.382	1.356	1.325	0.3

	vars	n	mean	sd	median	trimmed	m
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dl
wtd_entropy_fie	17	21263	0.927	0.334	0.917	0.924	0.2
range_fie	18	21263	572.223	309.614	764.100	592.974	68.9
wtd_range_fie	19	21263	483.517	224.043	510.440	494.860	278.3
std_fie	20	21263	215.631	109.967	266.374	221.657	78.4
wtd_std_fie	21	21263	224.050	127.927	258.450	233.091	136.5
mean_atomic_radius	22	21263	157.983	20.147	160.250	159.018	15.4
wtd_mean_atomic_radius	23	21263	134.720	28.802	125.970	132.333	28.8
gmean_atomic_radius	24	21263	144.449	22.091	142.808	144.246	15.8
wtd_gmean_atomic_radius	25	21263	120.989	35.838	113.181	117.557	39.6
entropy_atomic_radius	26	21263	1.268	0.375	1.331	1.292	0.3
wtd_entropy_atomic_radius	27	21263	1.131	0.407	1.243	1.167	0.3
range_atomic_radius	28	21263	139.325	67.272	171.000	146.801	50.4
wtd_range_atomic_radius	29	21263	51.370	35.019	43.000	45.729	22.6
std_atomic_radius	30	21263	51.601	22.898	58.663	53.766	18.7
...	...	...	...	...	...	...	
wtd_mean_FusionHeat	53	21263	13.848	14.279	8.331	10.956	5.6
gmean_FusionHeat	54	21263	10.137	10.066	5.253	8.161	2.6
wtd_gmean_FusionHeat	55	21263	10.141	13.134	4.930	7.631	5.4
entropy_FusionHeat	56	21263	1.093	0.376	1.112	1.114	0.3
wtd_entropy_FusionHeat	57	21263	0.914	0.370	0.995	0.938	0.3
range_FusionHeat	58	21263	21.139	20.371	12.878	16.902	5.2
wtd_range_FusionHeat	59	21263	8.219	11.414	3.436	5.913	2.6
std_FusionHeat	60	21263	8.323	8.672	4.948	6.386	1.6
wtd_std_FusionHeat	61	21263	7.718	7.288	5.501	6.232	1.8
mean_ThermalConductivity	62	21263	89.707	38.517	96.504	89.978	29.5
wtd_mean_ThermalConductivity	63	21263	81.549	45.519	73.333	76.951	35.9
gmean_ThermalConductivity	64	21263	29.842	34.060	14.288	23.034	10.8
wtd_gmean_ThermalConductivity	65	21263	27.308	40.191	6.096	19.120	8.1
entropy_ThermalConductivity	66	21263	0.728	0.326	0.739	0.731	0.3
wtd_entropy_ThermalConductivity	67	21263	0.540	0.318	0.546	0.524	0.3
range_ThermalConductivity	68	21263	250.893	158.704	399.795	261.322	4.1
wtd_range_ThermalConductivity	69	21263	62.033	43.123	56.556	57.775	45.7
std_ThermalConductivity	70	21263	98.944	60.143	135.762	101.303	50.0
wtd_std_ThermalConductivity	71	21263	96.234	63.710	113.557	97.354	80.2
mean_Valence	72	21263	3.198	1.045	2.833	3.061	0.8
wtd_mean_Valence	73	21263	3.153	1.191	2.618	2.993	0.8
gmean_Valence	74	21263	3.057	1.046	2.615	2.902	0.6
wtd_gmean_Valence	75	21263	3.056	1.175	2.434	2.881	0.5

	vars	n	mean	sd	median	trimmed	m
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dl
entropy_Valence	76	21263	1.296	0.393	1.369	1.323	0.4
wtd_entropy_Valence	77	21263	1.053	0.380	1.167	1.084	0.3
range_Valence	78	21263	2.041	1.242	2.000	1.968	1.4
wtd_range_Valence	79	21263	1.483	0.978	1.063	1.334	0.3
std_Valence	80	21263	0.839	0.485	0.800	0.815	0.5
wtd_std_Valence	81	21263	0.674	0.456	0.500	0.640	0.3
critical_temp	82	21263	34.421	34.254	20.000	30.314	25.4

Let us look at some of the statistics from the table above in detail and lets see what we can find out. Of all the statistics listed above, the ones which are most important to us are *mean* and *standard deviation.*

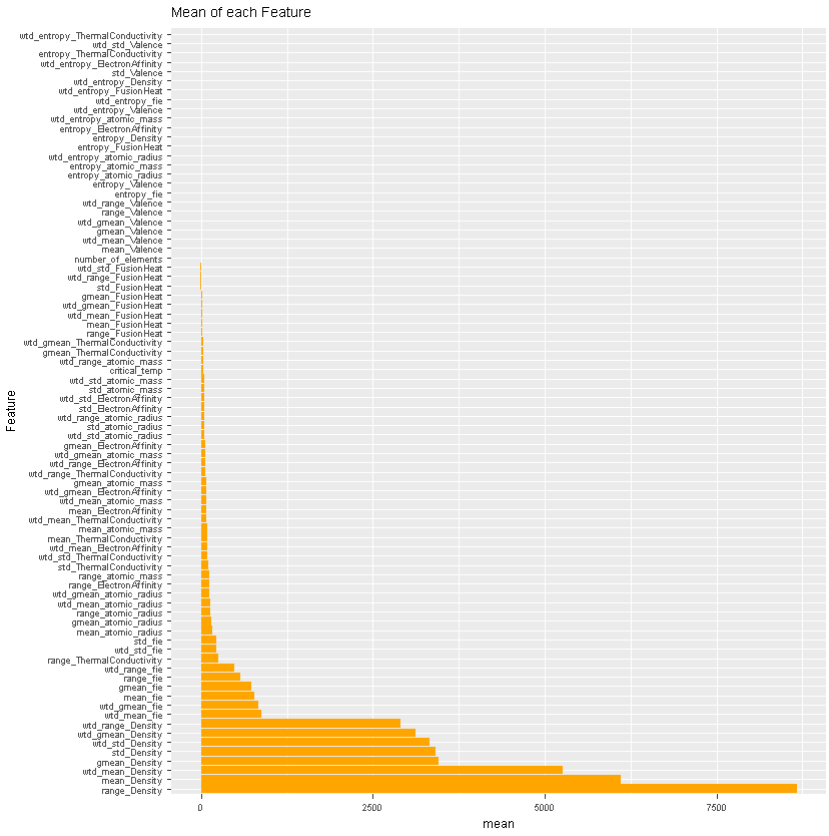
Mean and Standard Deviation:

Mean, as we all know is the average of all the values for a given feature.

The standard deviation is a measurement that estimates the spread of a dataset with respect to its mean which is calculated as square root of variance. The more the data is spread out from the mean or the average value the higher the standard deviation. So intuitively, I understand that the features with a higher standard deviation is of more use to us in building a model rather than the ones with a low standard deviation. Data with low standard deviation will show little/no variation in the data and hence will not contribute to the changes related to the target variable. A low standard deviation is not always preferrable.

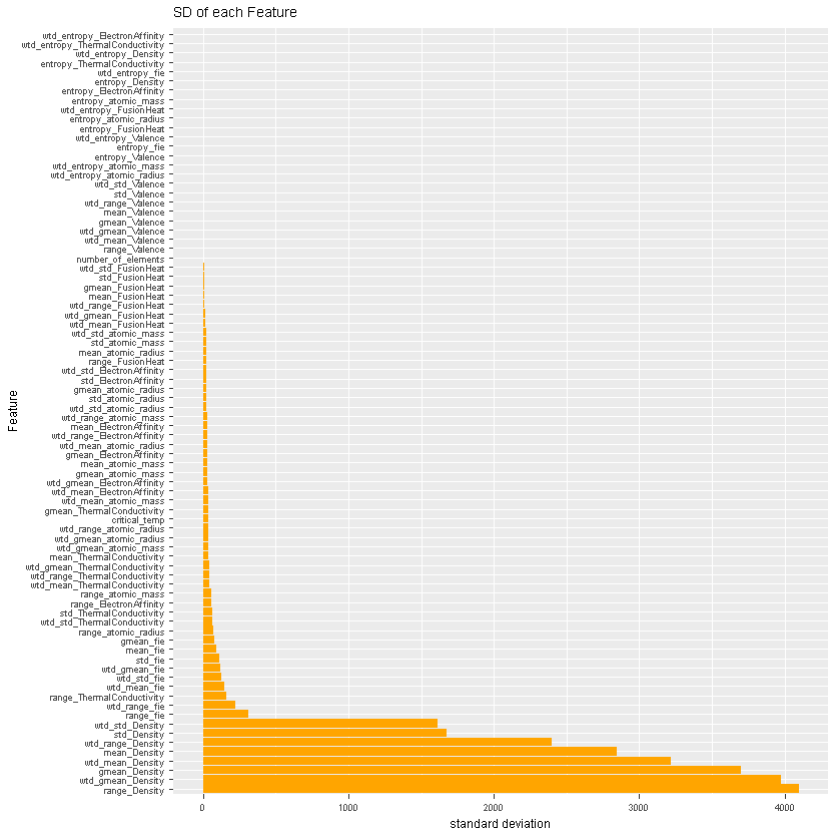
In [20]:

```
# plotting a graph of the mean of the variables
# install.packages("ggplot2")
library(ggplot2)
ggplot(data=descDF, aes(x=reorder(Feature, -mean), y=mean)) + geom_bar(stat="identity",fil
theme(text = element_text(size=7), axis.text.y = element_text(hjust=1))+ labs(title="Mean
                                                                              y = "mean", x
```



In [15]:

```
# plotting a graph of the standard deviation of the variables
ggplot(data=descDF, aes(x=reorder(Feature, -sd), y=sd)) + geom_bar(stat="identity",fill = '
theme(text = element_text(size=7), axis.text.y = element_text(hjust=1))+labs(title="SD of e
                                                                              y = "standard d
```



Description of the above Observations from the Graphs:

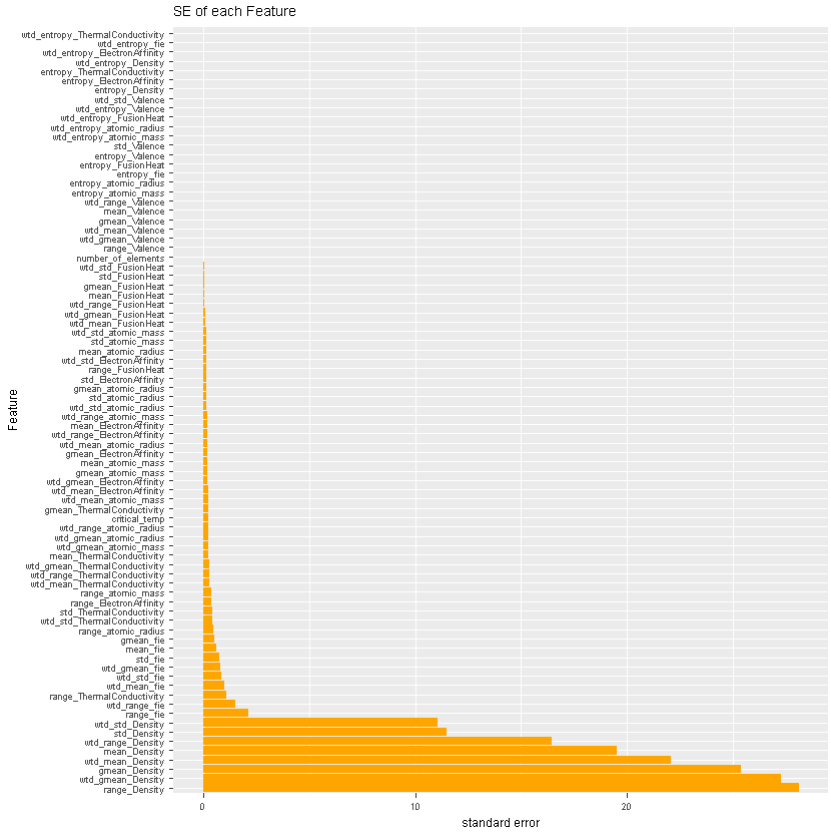
By looking at the above plot we can say that all the features relating Density i.e "gmean_Density", "wtd_gmean_Density" and "range_Density" has the highest standard deviation around the mean. So we can intuitively say that these could be variables of significance, although we do not know at this point. Also, looking at the graph, "wtd_entropy_ElectronAffinity", "wtd_entropy_ThermalConductivity" and many more does not have a significant standard deviation and hence may not be contributing to the changes in the target.

Standard Error:

Standard error, the way I understand it is the standard deviation of the means in a dataset. Even this shows us the variation in the dataset. The more data points involved in the calculations of the mean, the smaller the standard error tends to be. When the standard error is small, the data is said to be more representative of the true mean. In cases where the standard error is large, the data may have some notable irregularities. Standard errors function more as a way to determine the accuracy of the sample or the accuracy of multiple samples by analyzing deviation within the means.

In [16]:

```
# plotting a graph of the standard error of the variables
ggplot(data=descDF, aes(x=reorder(Feature, -se), y=se)) + geom_bar(stat="identity",fill = '
theme(text = element_text(size=7), axis.text.y = element_text(hjust=1))+labs(title="SE of e
                                                                              y = "standard e
```



Description of the above Observations from the Graph:

Again variables like "gmean_Density", "wtd_gmean_Density" and "range_Density" has the highest standard error. On the other hand "wtd_entropy_Thermal_Conductivity", "wtd_entropy_fie", "wtd_entropy_ThermalConductivity" and many more, has relatively no standard error. Hence here we can say that variables like "wtd_entropy_Thermal_Conductivity", "wtd_entropy_fie", "wtd_entropy_ThermalConductivity" set a better representation of the true mean of the actual population.

Skewness:

It is the measure of distortion from the symmetrical bell curve or the normal distribution. It measures the lack of symmetry in data distribution. It differentiates extreme values in one versus the other tail. A symmetrical distribution will have a skewness of 0. If skew is negative it is left skewed and if the skew is positive is it right skewed. Skewness of 0 will give us a normal distribution.

In [17]:

```
# plotting a graph of the skewness of the variables
ggplot(data=descDF, aes(x=reorder(Feature, -skew), y=skew)) + geom_bar(stat="identity",fil
theme(text = element_text(size=7), axis.text.y = element_text(hjust=1))+labs(title="Skewnes
                                                                    y = "skew", x="
```

Description of the above Observations from the Graph:

So, by looking at the above plot it is understood that almost all the variables have skewness in them except "wtd_entropy_fie" and "number of elements". Hence we can say that these two variables are almost normally distributed, whereas for other than these two variables, all the variables are either left skewed or right skewed. Therefore, for our model building we know that that data is not normally distributed and hence skewness is also taken into consideration.

2.3 Investigating Each Variable Distribution:

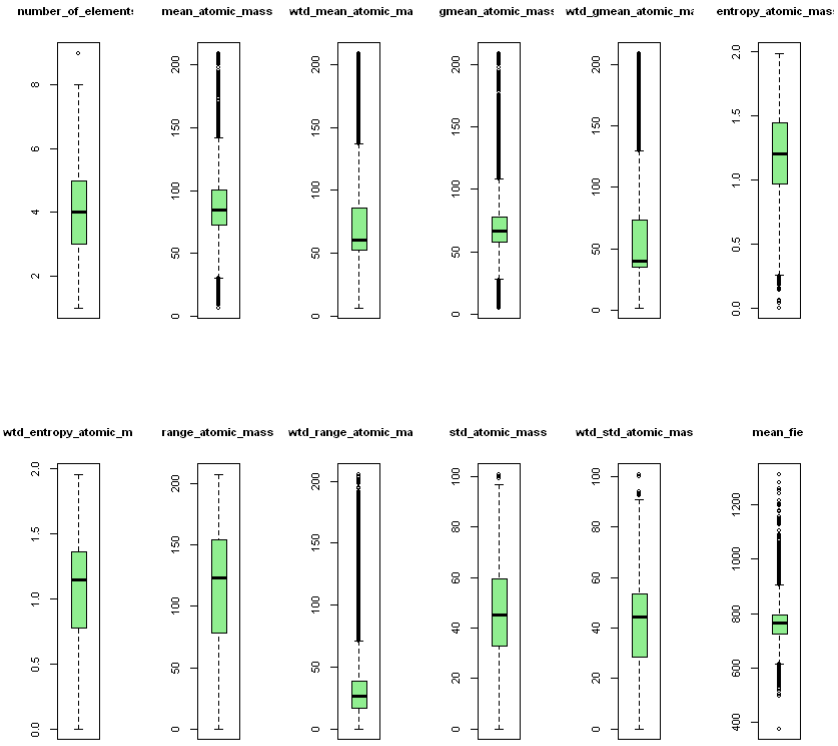
Now that we have seen the bar charts comparing the statistics of every features, we can now investigate and observe how each variable's distribution is.

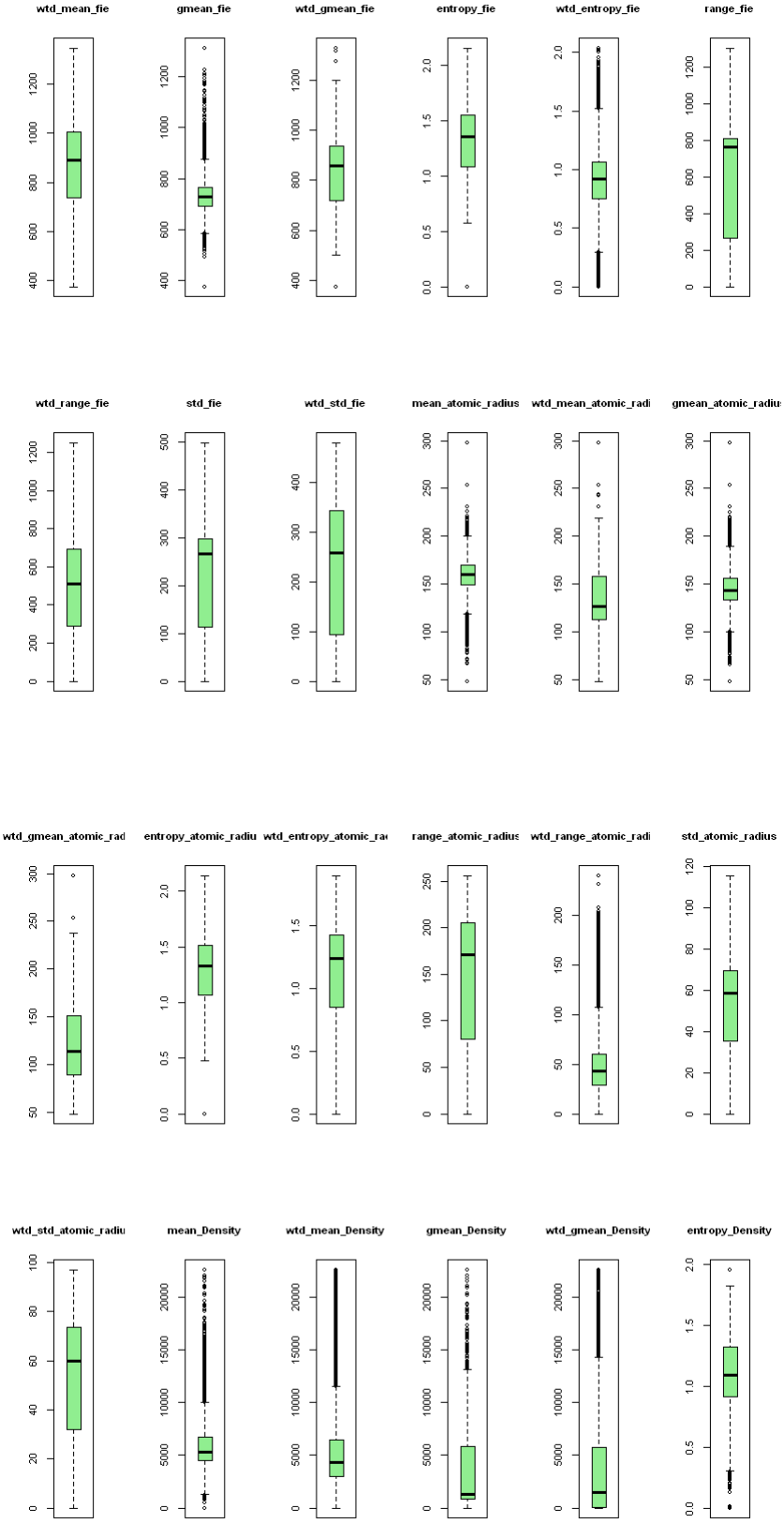
2.3.1 Boxplots:

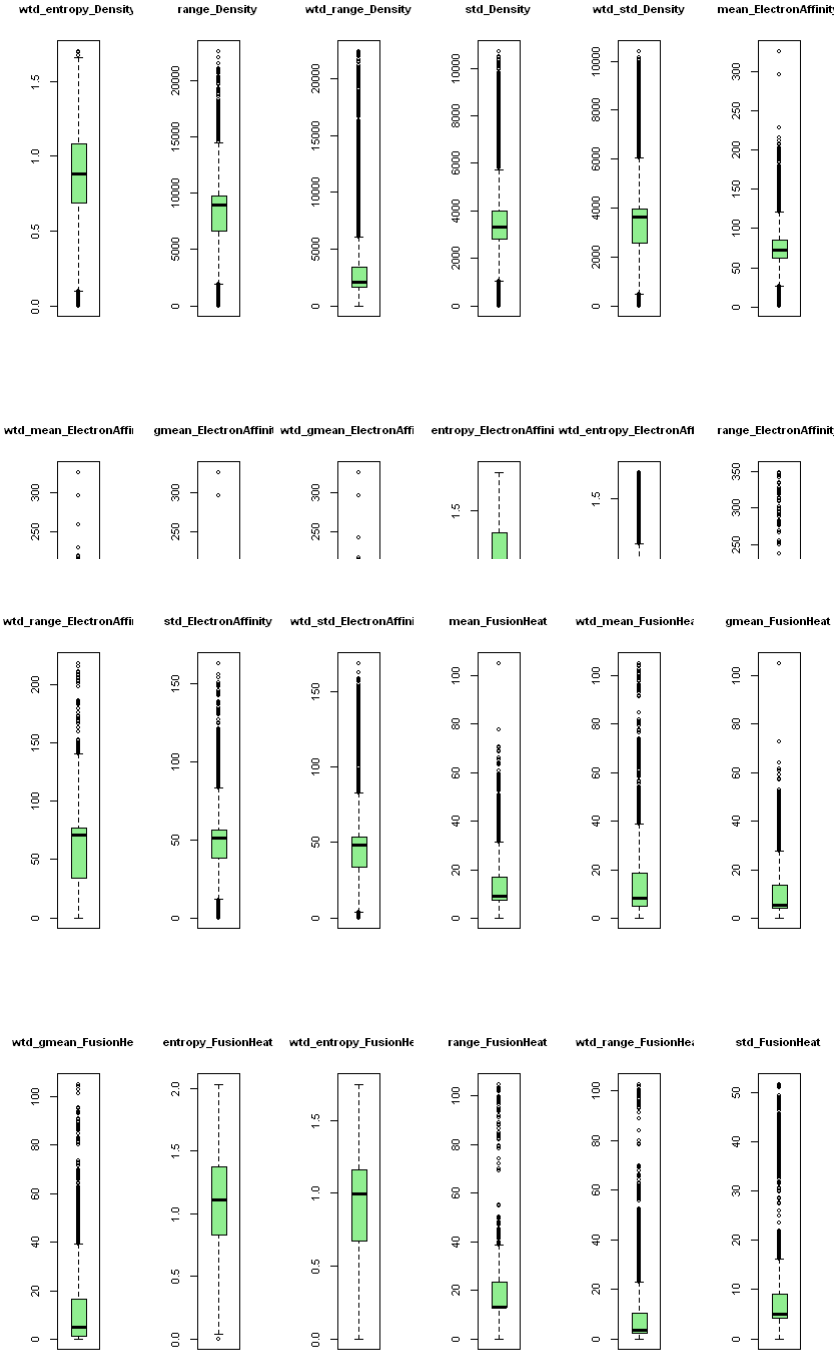
Boxplots are a very good way to explore the individual distribution of each variable based on median, first quartile, third quartile, outliers, maximum and minimum. Since all variables we have are numerical, we can do boxplots for every one of them to check the distribution.

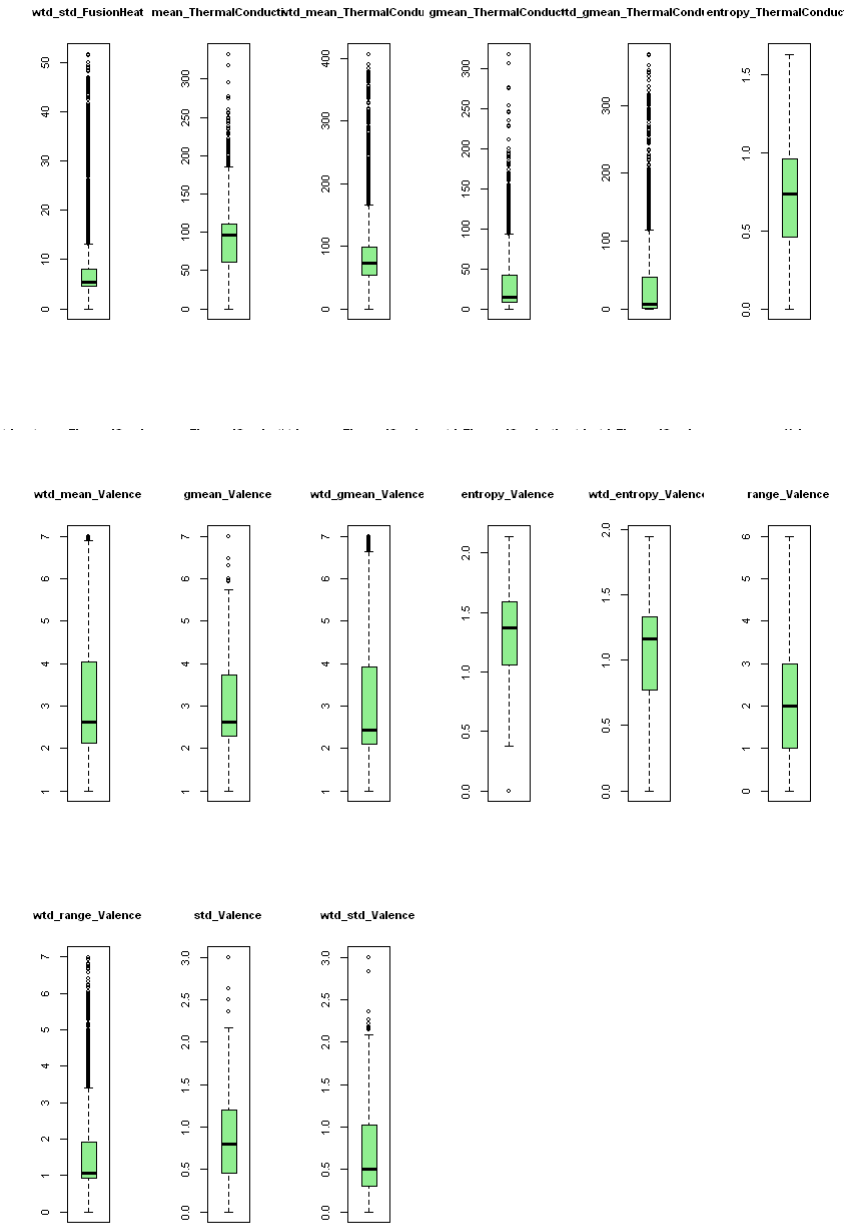
In [18]:

```
# plotting them all in a loop, this gives us neat boxplots
par(mfrow = c(2.5,6))
for (i in 1:(length(trainingData) - 1)) {
        boxplot(trainingData[,i], main = names(trainingData[i]), type="l", col = 'lightgree
}
```




wtd_mean_fie
gmean_fie
wtd_gmean_fie
entropy_fie
wtd_entropy_fie
range_fie
wtd_range_fie
std_fie
wtd_std_fie
mean_atomic_radius
wtd_mean_atomic_radi
gmean_atomic_radiu
wtd_gmean_atomic_rad
entropy_atomic_radiu
wtd_entropy_atomic_ra
range_atomic_radius
wtd_range_atomic_radi
std_atomic_radius
wtd_std_atomic_radiu
mean_Density
wtd_mean_Density
gmean_Density
wtd_gmean_Density
entropy_Density


wtd_entropy_Density
range_Density
wtd_range_Density
std_Density
wtd_std_Density
mean_ElectronAffinity
wtd_mean_ElectronAffi
gmean_ElectronAffinit
wtd_gmean_ElectronAffi
entropy_ElectronAffini
wtd_entropy_ElectronAffi
range_ElectronAffinit
wtd_range_ElectronAffi
std_ElectronAffinity
wtd_std_ElectronAffini
mean_FusionHeat
wtd_mean_FusionHea
gmean_FusionHeat
wtd_gmean_FusionHe
entropy_FusionHeat
wtd_entropy_FusionHe
range_FusionHeat
wtd_range_FusionHea
std_FusionHeat



We know that boxplots are one of the best methods to detect outliers. The reasons we are detect them, is that, the outliers have a significant impact on the model prediction, which may not be good. Also, it is not a good idea to drop all the outliers as we might be loosing important information, specially in the case when we have a large number of outliers, which is very clear by looking at the above boxplots. Log scale transformation to reduce the variability of the data, can be quite tedious in a scenario like this when we have 81 variables. Well, as to my knowledge, instead of using linear models, we can make use of tree based models like Random Forest Method as these models are less impacted by outliers. We will compare the linear and the random forest models later in the assignment.

2.3.2 Histograms:

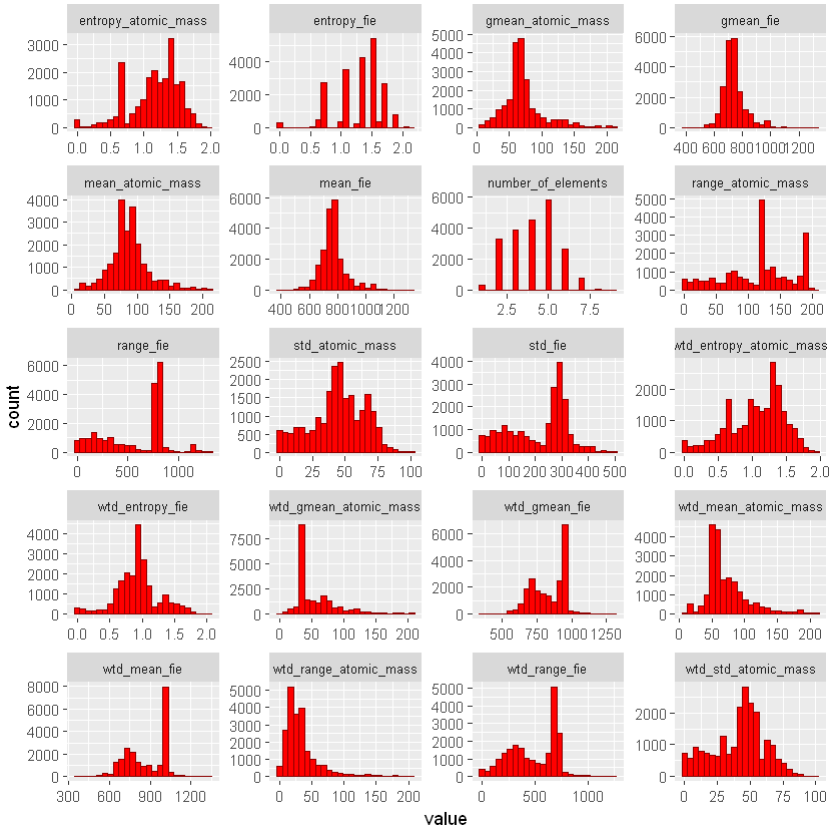
Histograms are a common way to show the distribution of the numerical data. From histograms we can know that either the data is left skewed or right skewed or normal. In the case where the data is skewed, it is not a good idea to take mean into consideration. We should rather consider median instead of mean for evaluation purposes. The reason being, when the data is skewed, mean may not be a good estimate of the centre of the data because a lot of data is falling there. In this case we should go for median to evaluate the cetre of the data. In positive skewness the mean is greater than the median and in negative skewness the mean is less than the median.

In [27]:

```
# we use ggplot2 library for better graphs
# install.packages("ggplot2")
# install.packages("reshape2")
# install.packages("purrr")
# install.packages("tidyr")

# plotting histogram of each variable in the dataframe
library(purrr)
library(tidyr)
library(ggplot2)
```

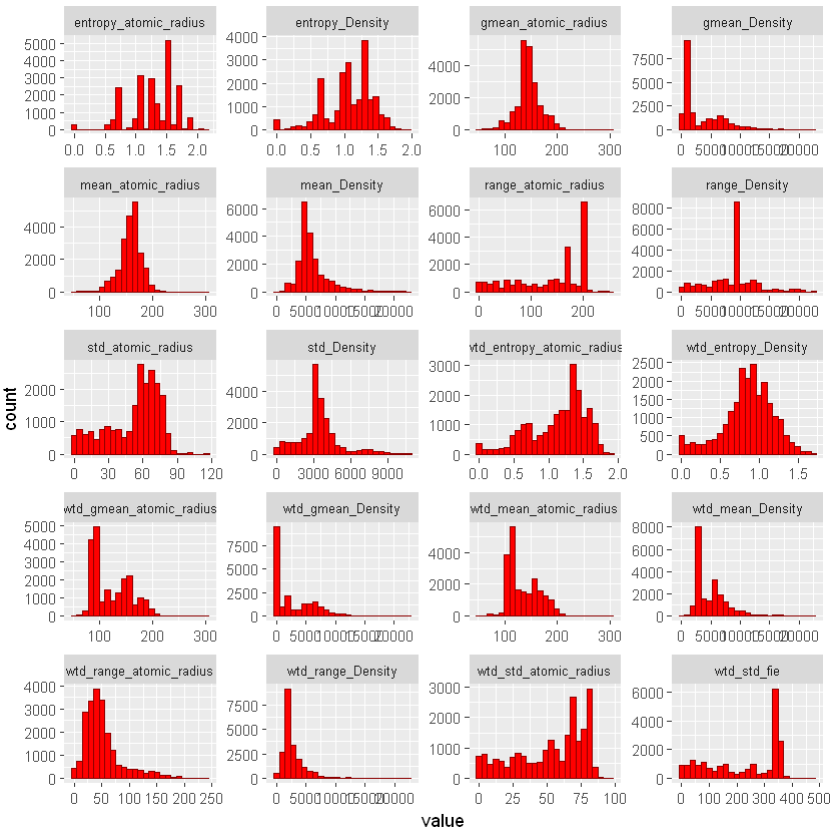
In [28]:

```
# plotting for the first 20 variables
trainingData[0:20] %>%
  keep(is.numeric) %>%
  gather() %>%
ggplot(aes(value)) +
    facet_wrap(~ key, scales = "free", ncol=4) +
    geom_histogram(fill="red",color="darkred",bins=25) +
theme(strip.text = element_text(size = 8))
```



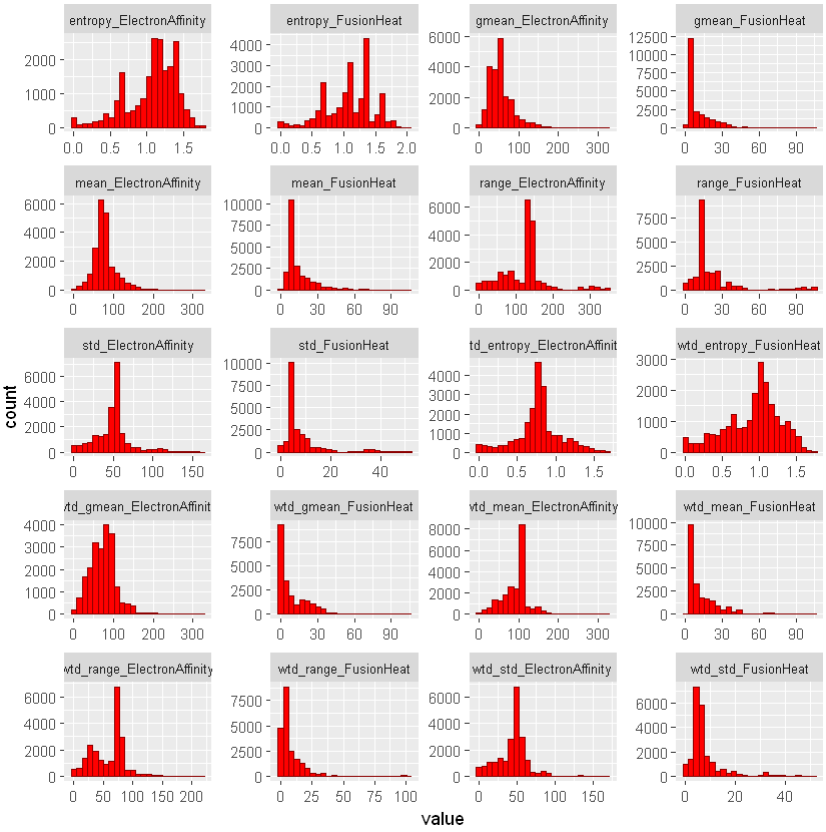
In [29]:

```
# plotting for the next 20 variables
trainingData[21:40] %>%
  keep(is.numeric) %>%
  gather() %>%
ggplot(aes(value)) +
    facet_wrap(~ key, scales = "free", ncol=4) +
    geom_histogram(fill="red",color="darkred",bins=25) +
theme(strip.text = element_text(size = 8))
```



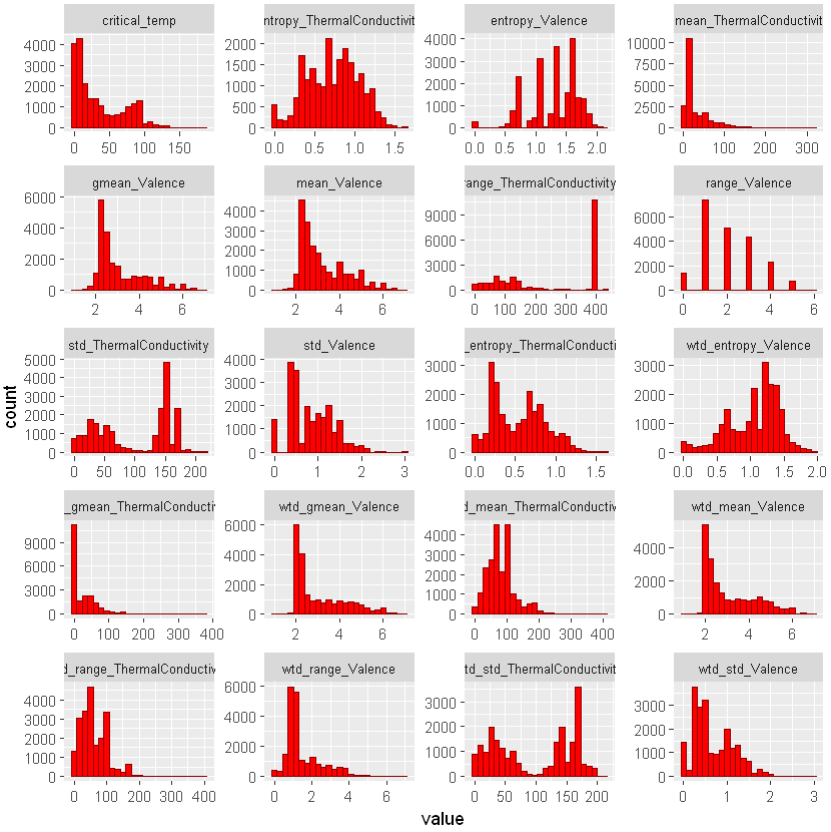
In [30]:

```
# plotting for the next 20 variables
trainingData[42:61] %>%
  keep(is.numeric) %>%
  gather() %>%
ggplot(aes(value)) +
    facet_wrap(~ key, scales = "free", ncol=4) +
    geom_histogram(fill="red",color="darkred",bins=25) +
theme(strip.text = element_text(size = 8))
```



In [31]:

```
# plotting for the last 20 variables
trainingData[63:82] %>%
  keep(is.numeric) %>%
  gather() %>%
ggplot(aes(value)) +
    facet_wrap(~ key, scales = "free", ncol=4) +
    geom_histogram(fill="red",color="darkred",bins=25)+
    theme(strip.text = element_text(size = 8))
```



So by looking at the histograms above, we can find many types of histogram like skewed distribution, normal distribution, unimodal distribution, bimodal and multimodal distribution and peaked distribution. For example, "wtd_std_ThermalConductivity" is a bimodal histogram, "entropy_ThermalConductivity" is a unimodal histogram, "wtd_range_atomic_radius" is right skewed and so on. These histograms say that the data has a lot of variation in it which intuitively can also be favourable for our model. We can also see from many of the histograms like the one for "mean_FusionHeat" and "range_Density" that there are many outliers as it has many extremely low or high values

2.4 Checking Correlation Between the Variables:

Correlation is the degree of association between two variables. It is always said that out of two very highly correlated values, one of them should be removed. The reason being, correlated values does not always decrease the accuracy of the model but they will not improve it either. We remove correlated variables as it makes the learning algorithm faster. In modelling, less features usually mean high improvement of the model in terms of speed. It also helps to decrease harmful bias. But if the highly correlated values also relate to the target to a good level, we might want to keep it. For some algorithms like Random Forest and Naive Bayes, may actually directly or indirectly benefit from correlated variables. Also, removing the highly correlated values will work best with wrapper feature selection methods like stepwise forward/backward/hybrid selection. Hence, we will perform paiwise correlation to filter out the highly correlated values after which it will be easier for us to build a linear model.

Here our threshold for highly correlated values is 0.80 or -0.80.

2.4.1 DIY Pearson Correlation Matrix Function (heatmap):

In [35]:

```
# http://www.sthda.com/english/wiki/ggplot2-quick-correlation-matrix-heatmap-r-software-anc
# install.packages("reshape2")

library(reshape2)
heatMap <- function(x,y) {
cormat <- round(cor(trainingData[x:y]),2)
melted_cormat <- melt(cormat)

# Get lower triangle of the correlation matrix
  get_lower_tri<-function(cormat){
    cormat[upper.tri(cormat)] <- NA
    return(cormat)
  }
  # Get upper triangle of the correlation matrix
  get_upper_tri <- function(cormat){
    cormat[lower.tri(cormat)]<- NA
    return(cormat)
  }

upper_tri <- get_upper_tri(cormat)
melted_cormat <- melt(upper_tri, na.rm = TRUE)

reorder_cormat <- function(cormat){
# Use correlation between variables as distance
dd <- as.dist((1-cormat)/2)
hc <- hclust(dd)
cormat <-cormat[hc$order, hc$order]
}

ggheatmap <- ggplot(melted_cormat, aes(Var2, Var1, fill = value))+
 geom_tile(color = "white")+
 scale_fill_gradient2(low = "blue", high = "red", mid = "white",
   midpoint = 0, limit = c(-1,1), space = "Lab",
    name="Pearson\nCorrelation") +
  theme_minimal()+ # minimal theme
 theme(axis.text.x = element_text(angle = 45, vjust = 1,
    size = 12, hjust = 1))+
 coord_fixed()

ggheatmap +
geom_text(aes(Var2, Var1, label = value), color = "black", size = 4) +
theme(
  axis.title.x = element_blank(),
  axis.title.y = element_blank(),
  panel.grid.major = element_blank(),
  panel.border = element_blank(),
  panel.background = element_blank(),
  axis.ticks = element_blank(),
  legend.justification = c(1, 0),
  legend.position = c(0.6, 0.7),
  legend.direction = "horizontal")+
  guides(fill = guide_colorbar(barwidth = 7, barheight = 1,
                title.position = "top", title.hjust = 0.5))
}
```

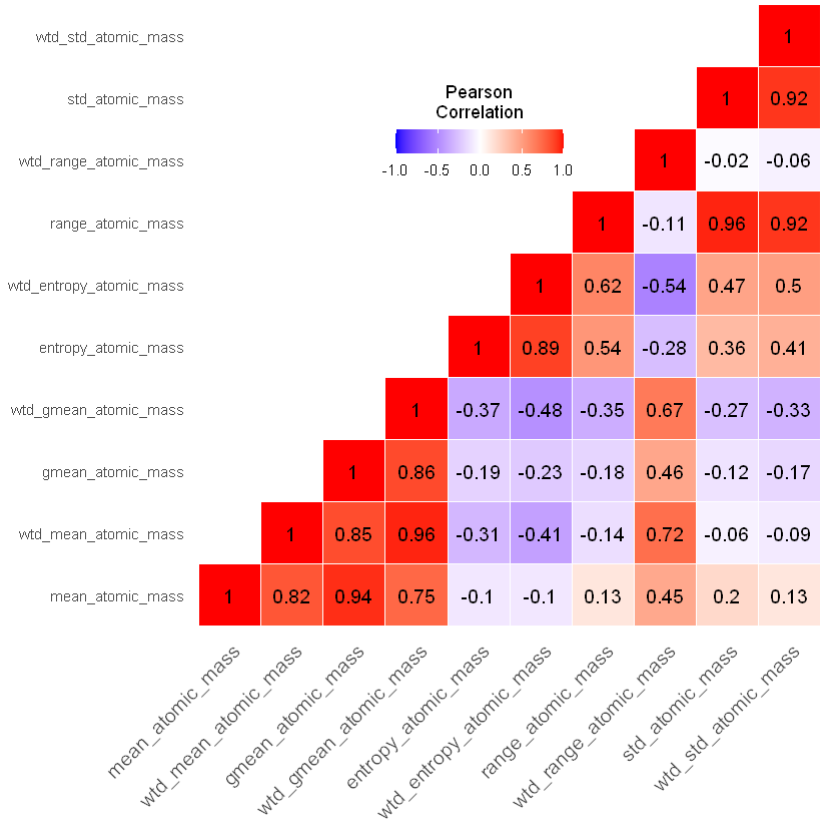
I would also like to mention here that the correlation heatmap was not being displayed properly for all variables at once and hence I have broken down the matrix to 8 parts, i.e. the big 8 categories. It will help to read the

correlation coefficient clearly.

Atomic Mass:

In [36]:

```
heatMap(2,11)
```



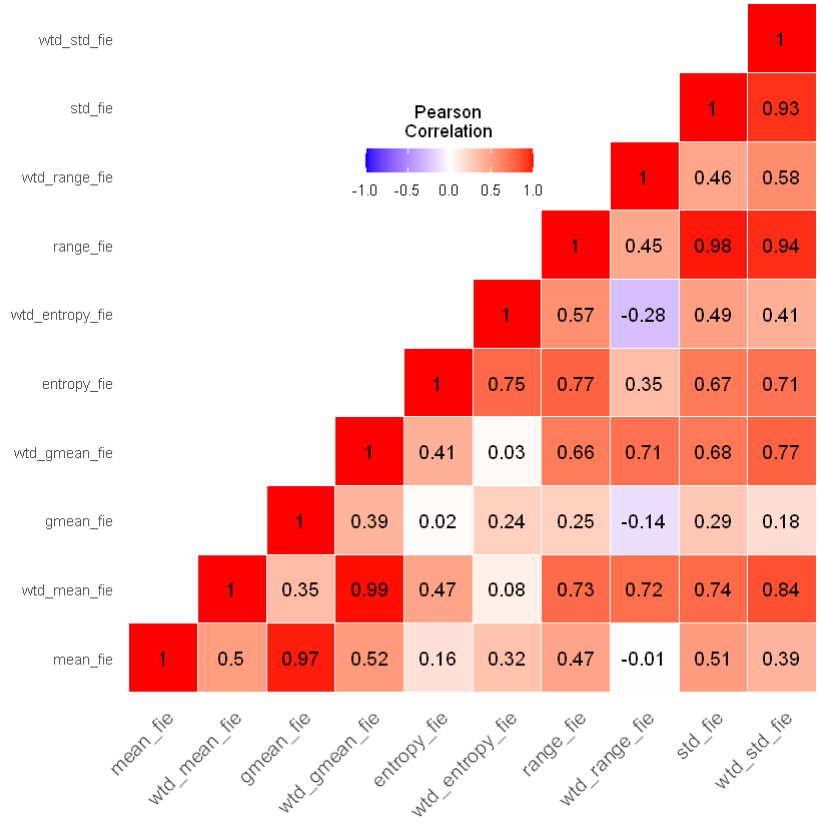
Here are some of the observations from the above plot(highly correlated variables):

- mean_atomic_mass and wtd_mean_atomic_mass -> 0.82
- mean_atomic_mass and gmean_atomic_mass -> 0.94
- wtd_mean_atomic_mass and gmean_atomic_mass -> 0.85
- wtd_mean_atomic_mass and wtd_gmean_atomic_mass -> 0.96
- gmean_atomic_mass and wtd_mean_atomic_mass -> 0.86
- entropy_atomic_mass and wtd_entropy_atomic_mass ->0.89
- range_atomic_mass and std_atomic_mass -> 0.96
- range_atomic_mass and wtd_std_atomic_mass -> 0.92
- std_atomic_mass and wtd_std_atomic_mass -> 0.92

Fie(First Inonisation Energy):

In [37]:

```
heatMap(12,21)
```



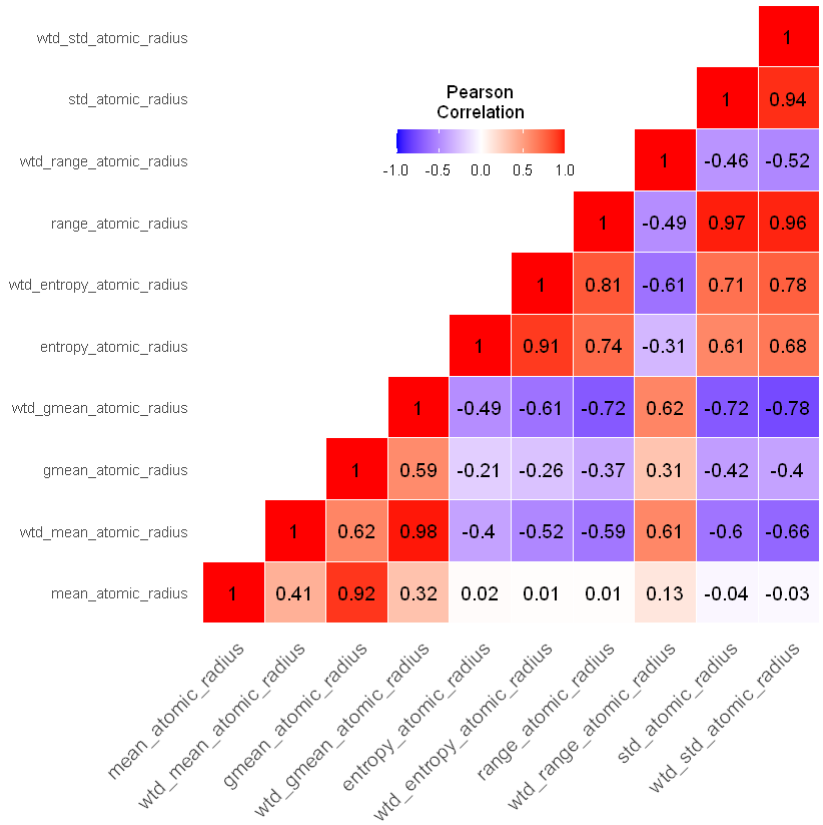
Here are some of the observations from the above plot(highly correlated variables):

- mean_fie and gmean_fie -> 0.97
- wtd_mean_fie and wtd_gmean_fie -> 0.99
- wtd_mean_fie and wtd_std_fie -> 0.83
- range_fie and std_fie -> 0.98
- range_fie and wtd_std_fie -> 0.94
- std_fie and wtd_std_fie -> 0.93

Atomic Radius:

In [38]:

```
heatMap(22,31)
```



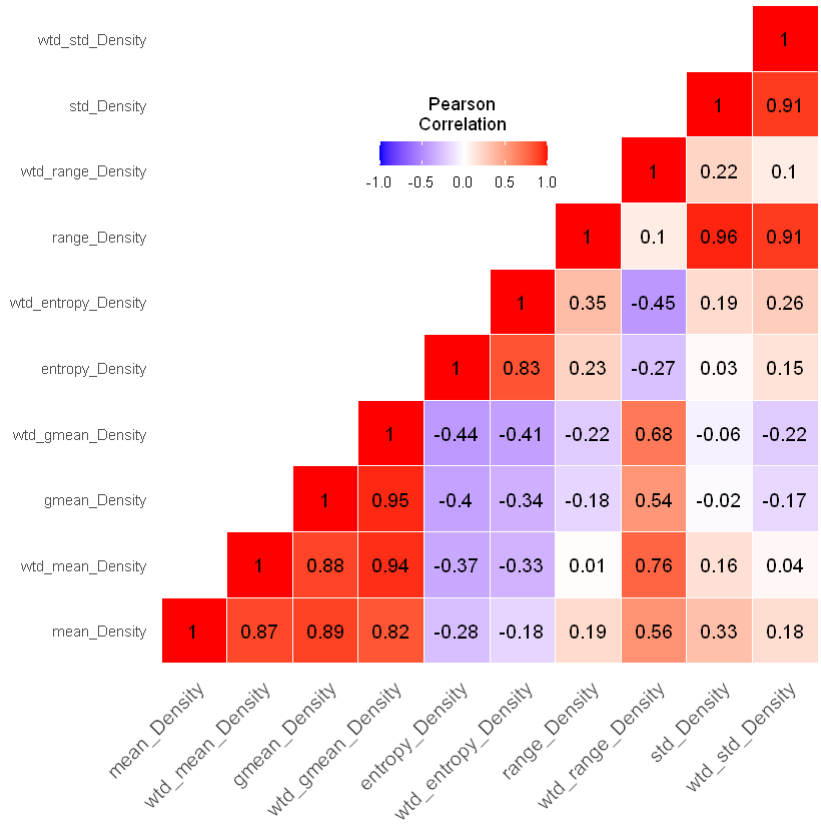
Here are some of the observations from the above plot(highly correlated variables):

- mean_atomic_radius and gmean_atomic_radius -> 0.92
- wtd_mean_atomic_radius and wtd_gmean_atomic_radius -> 0.98
- entropy_atomic_radius and wtd_entropy_atomic_radius -> 0.91
- wtd_entropy_atomic_radius and range_atomic_radius -> 0.81
- range_atomic_radius and std_atomic_radius -> 0.97
- range_atomic_radius and wtd_std_atomic_radius -> 0.96
- std_atomic_radius and wtd_std_atomic_radius -> 0.94

Density:

In [39]:

```
heatMap(32,41)
```



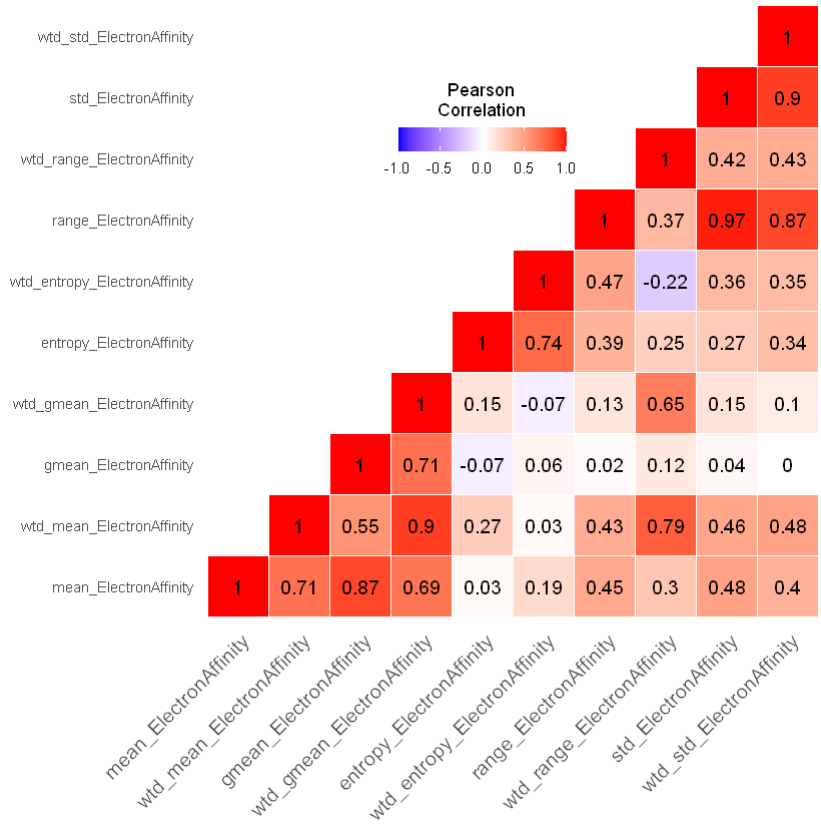
Here are some of the observations from the above plot(highly correlated variables):

- mean_Density and wtd_mean_Density -> 0.87
- mean_Density and gmean_Density -> 0.89
- mean_Density and wtd_gmean_Density -> 0.82
- wtd_mean_Density and gmean_Density ->0.88
- wtd_mean_Density and wtd_gmean_Density -> 0.94
- gmean_Density and wtd_gmean_Density -> 0.95
- entropy_Density and wtd_entropy_Density -> 0.83
- range_Density and std_Density -> 0.96
- range_Density and wtd_std_Density -> 0.91
- std_Density and wtd_std_Density -> 0.91

Electron Affinity:

In [40]:

```
heatMap(42,51)
```



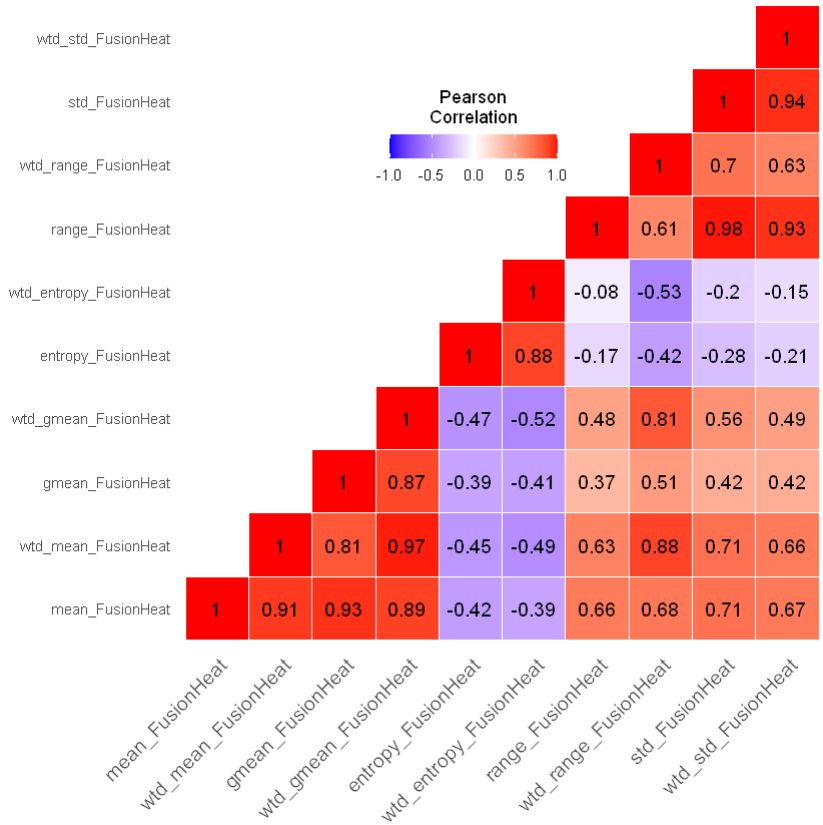
Here are some of the observations from the above plot(highly correlated variables):

- mean_ElectronAffinity and gmean_ElectronAffinity -> 0.87
- wtd_mean_ElectronAffinity and wtd_gmean_ElectronAffinity -> 0.90
- range_ElectronAffinity and std_ElectronAffinity -> 0.97
- range_ElectronAffinity and wtd_std_ElectronAffinity -> 0.87
- std_ElectronAffinity and wtd_std_ElectronAffinity -> 0.90

Fusion Heat:

In [41]:

```
heatMap(52,61)
```



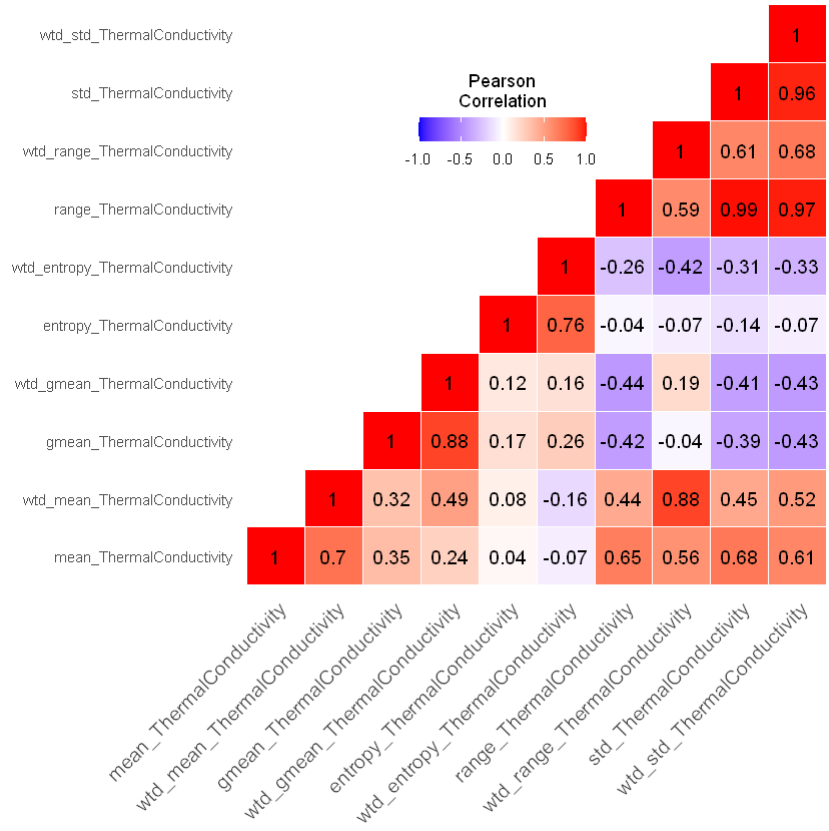
Here are some of the observations from the above plot(highly correlated variables):

- mean_FusionHeat and wtd_mean_FusionHeat -> 0.91
- mean_FusionHeat and gmean_FusionHeat -> 0.93
- mean_FusionHeat and wtd_gmean_FusionHeat -> 0.89
- wtd_mean_FusionHeat and gmean_FusionHeat -> 0.81
- wtd_mean_FusionHeat and wtd_gmean_FusionHeat -> 0.97
- wtd_mean_FusionHeat and wtd_range_FusionHeat -> 0.88
- gmean_FusionHeat and wtd_gmean_FusionHeat -> 0.87
- wtd_gmean_FusionHeat and wtd_range_FusionHeat -> 0.81
- entropy_FusionHeat and wtd_entropy_FusionHeat -> 0.88
- range_FusionHeat and std_FusionHeat -> 0.98
- range_FusionHeat and wtd_std_FusionHeat -> 0.93
- std_FusionHeat and wtd_std_FusionHeat -> 0.94

Thermal Conductivity:

In [42]:

```
heatMap(62,71)
```



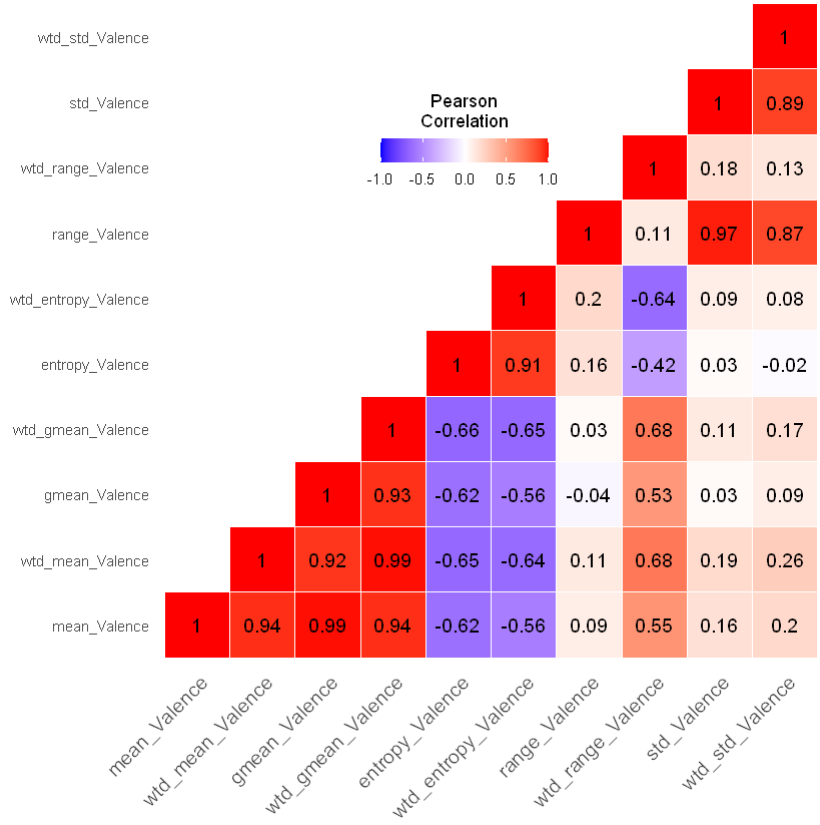
Here are some of the observations from the above plot(highly correlated variables):

- wtd_mean_ThermalConductivity and wtd_range_ThermalConductivity -> 0.88
- gmean_ThermalConductivity and wtd_gmean_ThermalConductivity -> 0.88
- range_ThermalConductivity and range_ThermalConductivity -> 0.99
- range_ThermalConductivity and wtd_std_ThermalConductivity -> 0.99
- std_ThermalConductivity and wtd_std_ThermalConductivity -> 0.96

Valence:

In [43]:

```
heatMap(72,81)
```



Here are some of the observations from the above plot(highly correlated variables):

- mean_Valence and wtd_mean_Valence -> 0.94
- mean_Valence and gmean_Valence -> 0.99
- mean_Valence and wtd_gmean_Valence -> 0.94
- wtd_mean_Valence and gmean_Valence -> 0.92
- wtd_mean_Valence and wtd_gmean_Valence -> 0.99
- gmean_Valence and wtd_gmean_Valence -> 0.93
- entropy_Valence and wtd_entropy_Valence -> 0.91
- range_Valence and std_Valence -> 0.97
- range_Valence and wtd_range_Valence -> 0.87
- std_Valence and wtd_std_Valence -> 0.89

Also, it is quite interesting to note that there are no variables which are highly negatively correlated.

2.4.2 Pairwise Correlation Coefficients Matrix of all Variables:

In [44]:

```
# printing the correlation matrix for all 82 variables at once
traningCor=round(cor(trainingData),4)
traningCor
```

A matrix: 82 × 82 of type dbl

	number_of_elements	mean_atomic_mass	wtd_mean_atomic_m
number_of_elements	1.0000	-0.1419	-0.3
mean_atomic_mass	-0.1419	1.0000	0.8
wtd_mean_atomic_mass	-0.3531	0.8160	1.0
gmean_atomic_mass	-0.2930	0.9403	0.8
wtd_gmean_atomic_mass	-0.4545	0.7458	0.9
entropy_atomic_mass	0.9393	-0.1040	-0.3
wtd_entropy_atomic_mass	0.8818	-0.0976	-0.4
range_atomic_mass	0.6828	0.1257	-0.1
wtd_range_atomic_mass	-0.3203	0.4462	0.7
std_atomic_mass	0.5140	0.1965	-0.0
wtd_std_atomic_mass	0.5464	0.1307	-0.0
mean_fie	0.1675	-0.2858	-0.2
wtd_mean_fie	0.4844	-0.2221	-0.5
gmean_fie	0.0242	-0.2406	-0.1
wtd_gmean_fie	0.4242	-0.2194	-0.5
entropy_fie	0.9732	-0.1669	-0.3
wtd_entropy_fie	0.7192	-0.1636	-0.1
range_fie	0.7812	-0.2556	-0.4
wtd_range_fie	0.3296	-0.0805	-0.4
std_fie	0.6740	-0.2766	-0.4
wtd_std_fie	0.7178	-0.2228	-0.4
mean_atomic_radius	-0.0014	0.4977	0.2
wtd_mean_atomic_radius	-0.4221	0.3768	0.6
gmean_atomic_radius	-0.2404	0.5611	0.4
wtd_gmean_atomic_radius	-0.5183	0.3599	0.6
entropy_atomic_radius	0.9722	-0.1400	-0.3
wtd_entropy_atomic_radius	0.9041	-0.1476	-0.4
range_atomic_radius	0.7681	-0.2707	-0.5
wtd_range_atomic_radius	-0.3713	0.1411	0.3
std_atomic_radius	0.6248	-0.3264	-0.5
...	...	...	
wtd_mean_FusionHeat	-0.4493	-0.1354	0.0

	number_of_elements	mean_atomic_mass	wtd_mean_atomic_m
gmean_FusionHeat	-0.5143	0.0148	0.1
wtd_gmean_FusionHeat	-0.5191	-0.0430	0.1
entropy_FusionHeat	0.9008	-0.0085	-0.2
wtd_entropy_FusionHeat	0.8605	-0.0285	-0.2
range_FusionHeat	0.0057	-0.3476	-0.2
wtd_range_FusionHeat	-0.3718	-0.1675	-0.0
std_FusionHeat	-0.1134	-0.3380	-0.2
wtd_std_FusionHeat	-0.0748	-0.3358	-0.2
mean_ThermalConductivity	0.2277	-0.1583	-0.2
wtd_mean_ThermalConductivity	0.2061	-0.0660	-0.0
gmean_ThermalConductivity	-0.4853	0.0060	0.1
wtd_gmean_ThermalConductivity	-0.4692	0.0564	0.2
entropy_ThermalConductivity	0.5019	-0.1001	-0.0
wtd_entropy_ThermalConductivity	0.2071	-0.0982	0.0
range_ThermalConductivity	0.6961	-0.1145	-0.3
wtd_range_ThermalConductivity	0.3168	-0.0278	-0.1
std_ThermalConductivity	0.6020	-0.1107	-0.3
wtd_std_ThermalConductivity	0.6656	-0.1109	-0.3
mean_Valence	-0.6094	0.3741	0.5
wtd_mean_Valence	-0.6486	0.3047	0.5
gmean_Valence	-0.6185	0.3922	0.5
wtd_gmean_Valence	-0.6593	0.3214	0.5
entropy_Valence	0.9678	-0.1568	-0.3
wtd_entropy_Valence	0.8926	-0.1456	-0.3
range_Valence	0.2319	-0.1074	-0.0
wtd_range_Valence	-0.4478	0.1686	0.3
std_Valence	0.1054	-0.0803	-0.0
wtd_std_Valence	0.0352	-0.0813	0.0
critical_temp	0.6011	-0.1135	-0.3

2.5 How every group of Features is affecting Critical Temperature of Superconductor:

Now that we have seen the correlation among the variables itself, intuitively, it will also be better for us to check how the features are affecting the target variable as well. Below is the function that is creating a scatter plot along with a regression line for each individual variable with respect to the target variable.

In [50]:

```
# install.packages("cowplot")
# install.packages("ggpubr")
# install.packages("gridExtra")
library(gridExtra)
library(cowplot)
library(ggpubr)

test <-colnames(trainingData)
# list of all the scatter plots for each and every variable against the target variable
plot_lst <- list()

for (i in 1:82){
  g<-ggplot(trainingData, aes_string(y = 'critical_temp', x =test[i]))+
    geom_point(size=0.6)+
    geom_smooth(method = "lm",size=0.5)
  plot_lst[[i]] <- g
}
```

```
Attaching package: 'ggpubr'


The following object is masked from 'package:cowplot':

    get_legend
```

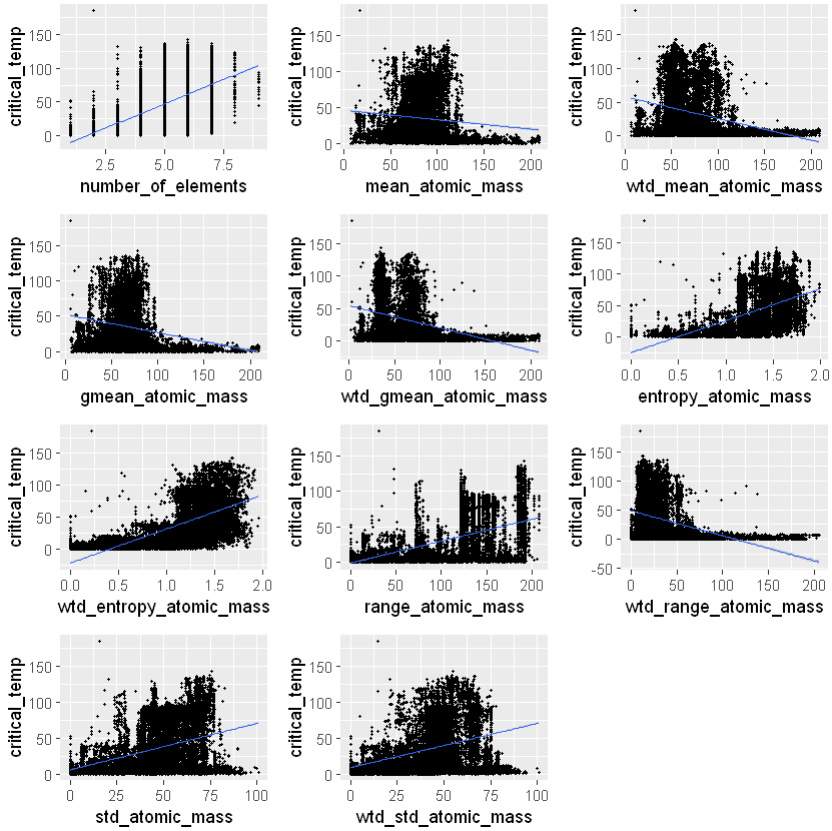
The plots with a regression line having positive slope tells us that the particular variable is positively related with the target and vice versa. Although the value of the slope determines the degree of relation of the variable withe the target.

Atomic Mass:

In [51]:

```
plot_grid(plot_lst[[1]],plot_lst[[2]],plot_lst[[3]],plot_lst[[4]],
          plot_lst[[5]],plot_lst[[6]],plot_lst[[7]],plot_lst[[8]],
          plot_lst[[9]],plot_lst[[10]],plot_lst[[11]], nrow = 4, ncol=3)
```

```
`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'
```



Here are some of the observations from the above plots:

Positively related to critical_temp:

- number_of_elements
- entropy_atomic_mass
- wtd_entropy_atomic_mass
- range_atomic_mass
- std_atomic_mass
- wtd_std_atomic_mass

Negatively related to critical_temp:

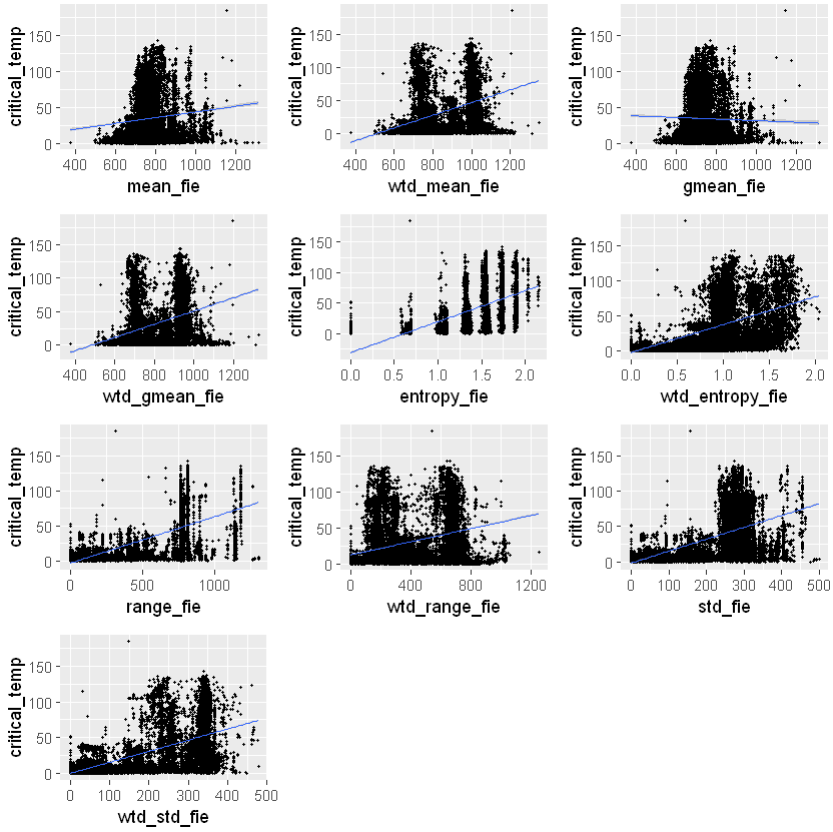
- mean_atomic_mass
- wtd_mean_atomic_mass
- gmean_atomic_mass
- wtd_gmean_atomic_mass
- wtd_range_atomic_mass

Fie:

In [52]:

```
plot_grid(plot_lst[[12]],plot_lst[[13]],plot_lst[[14]],plot_lst[[15]],
          plot_lst[[16]],plot_lst[[17]],plot_lst[[18]],plot_lst[[19]],
          plot_lst[[20]],plot_lst[[21]], nrow = 4, ncol=3)
```

```
`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'
```



Here are some of the observations from the above plots:

Positively related to critical_temp:

- mean_fie
- wtd_mean_fie
- wtd_gmean_fie
- entropy_fie
- wtd_entropy_fie
- range_fie
- wtd_range_fie
- std_fie
- wtd_std_fie

Negatively related to critical_temp:

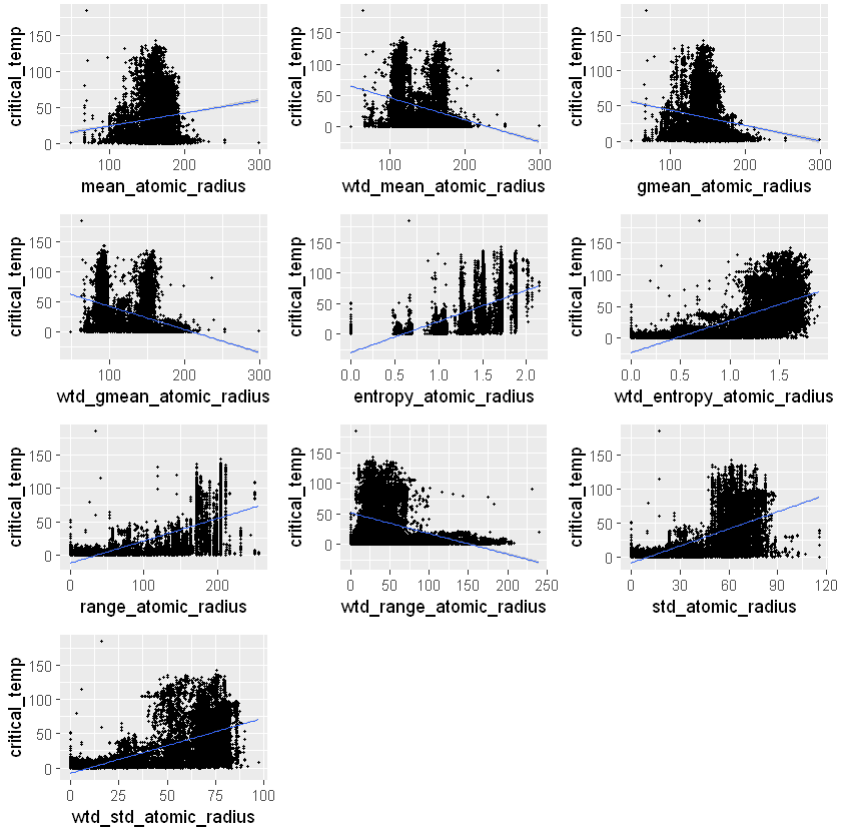
- gmean_fie

Atomic Radius:

In [53]:

```
plot_grid(plot_lst[[22]],plot_lst[[23]],plot_lst[[24]],plot_lst[[25]],
          plot_lst[[26]],plot_lst[[27]],plot_lst[[28]],plot_lst[[29]],
          plot_lst[[30]],plot_lst[[31]], nrow = 4, ncol=3)
```

```
`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'
```



Here are some of the observations from the above plots:

Positively related to critical_temp:

- mean_atomic_radius
- entropy_atomic_radius
- wtd_entropy_atomic_radius
- range_atomic_radius
- std_atomic_radius
- wtd_std_atomic_radius

Negatively related to critical_temp:

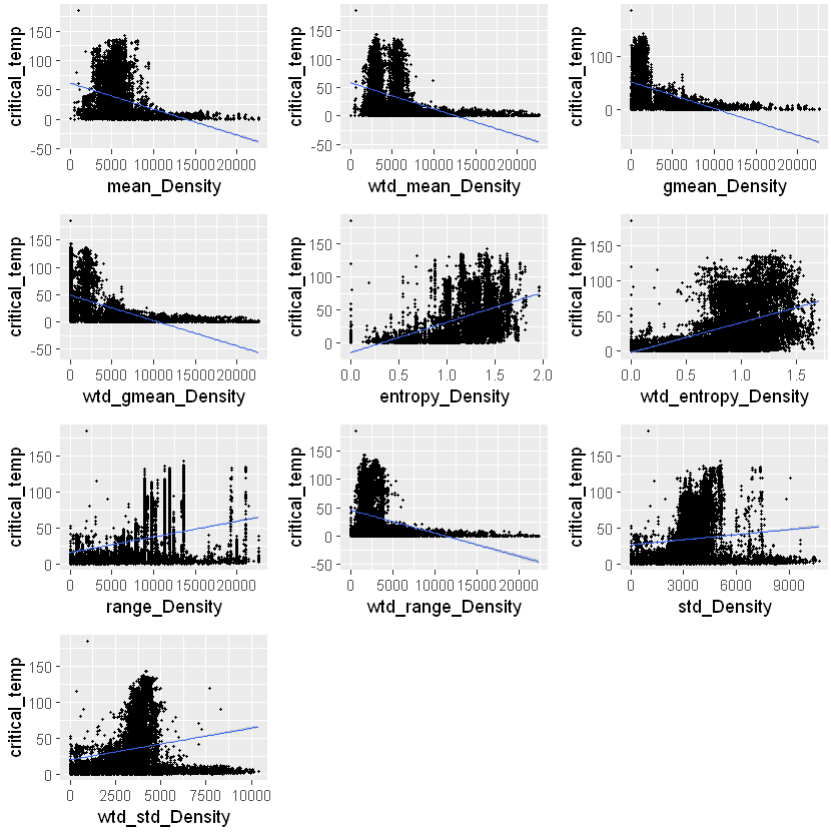
- wtd_mean_atomic_radius
- gmean_atomic_radius
- wtd_gmean_atomic_radius
- wtd_range_atomic_radius

Density:

In [54]:

```
plot_grid(plot_lst[[32]],plot_lst[[33]],plot_lst[[34]],plot_lst[[35]],
          plot_lst[[36]],plot_lst[[37]],plot_lst[[38]],plot_lst[[39]],
          plot_lst[[40]],plot_lst[[41]], nrow = 4, ncol=3)
```

```
`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'
```



Here are some of the observations from the above plots:

Positively related to critical_temp:

- entropy_Density
- wtd_entropy_Density
- range_Density
- std_Density
- wtd_std_Density

Negatively related to critical_temp:

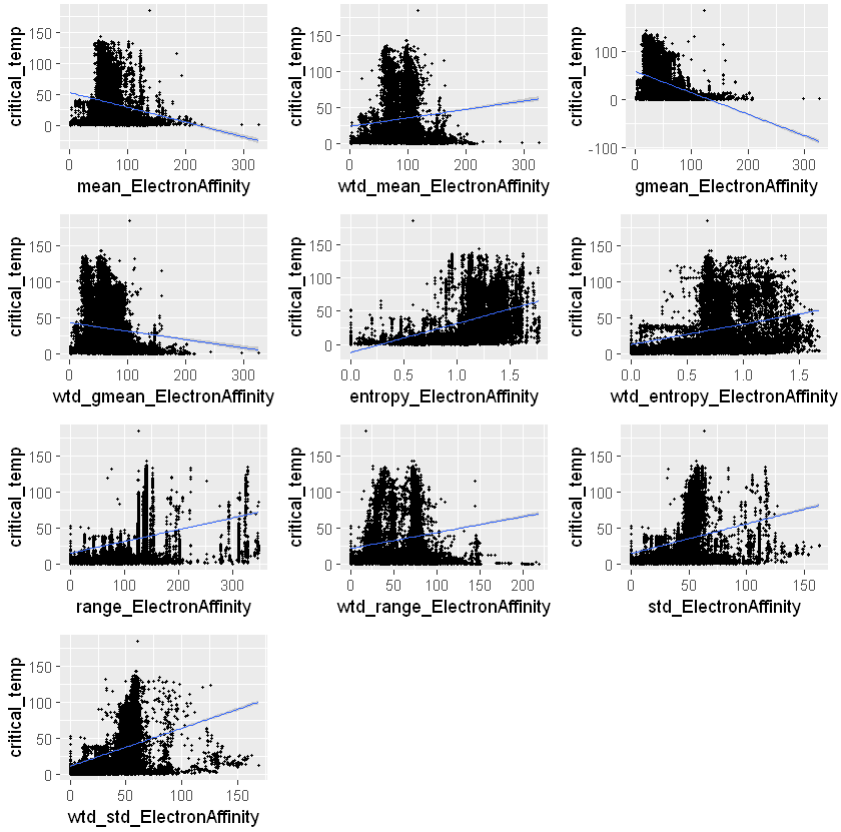
- mean_Density
- wtd_mean_Density
- gmean_Density
- wtd_gmean_Density
- wtd_range_Density

Electron Affinity:

In [55]:

```
plot_grid(plot_lst[[42]],plot_lst[[43]],plot_lst[[44]],plot_lst[[45]],
          plot_lst[[46]],plot_lst[[47]],plot_lst[[48]],plot_lst[[49]],
          plot_lst[[50]],plot_lst[[51]], nrow = 4, ncol=3)
```

```
`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'
```



Here are some of the observations from the above plots:

Positively related to critical_temp:

- wtd_mean_ElectronAffinity
- entropy_ElectronAffinity
- wtd_entropy_ElectronAffinity
- range_ElectronAffinity
- wtd_range_ElectronAffinity
- std_ElectronAffinity
- wtd_std_ElectronAffinity

Negatively related to critical_temp:

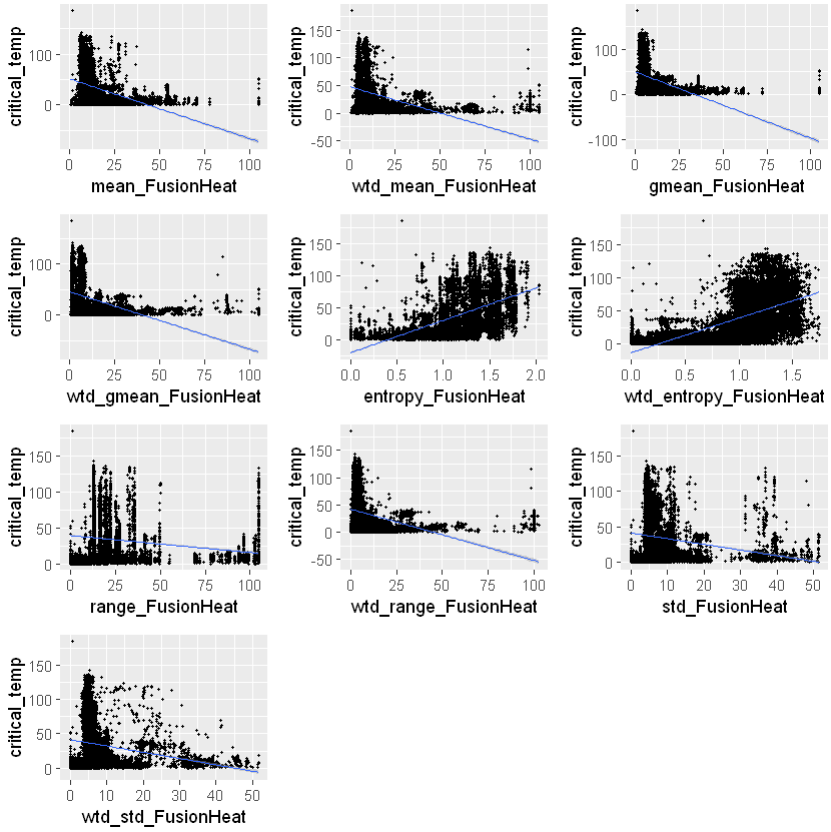
- mean_ElectronAffinity
- gmean_ElectronAffinity
- wtd_gmean_ElectronAffinity

Fusion Heat

In [56]:

```
plot_grid(plot_lst[[52]],plot_lst[[53]],plot_lst[[54]],plot_lst[[55]],
          plot_lst[[56]],plot_lst[[57]],plot_lst[[58]],plot_lst[[59]],
          plot_lst[[60]],plot_lst[[61]], nrow = 4, ncol=3)
```

```
`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'
```



Here are some of the observations from the above plots:

Positively related to critical_temp:

- entropy_FusionHeat
- wtd_entropy_FusionHeat

Negatively related to critical_temp:

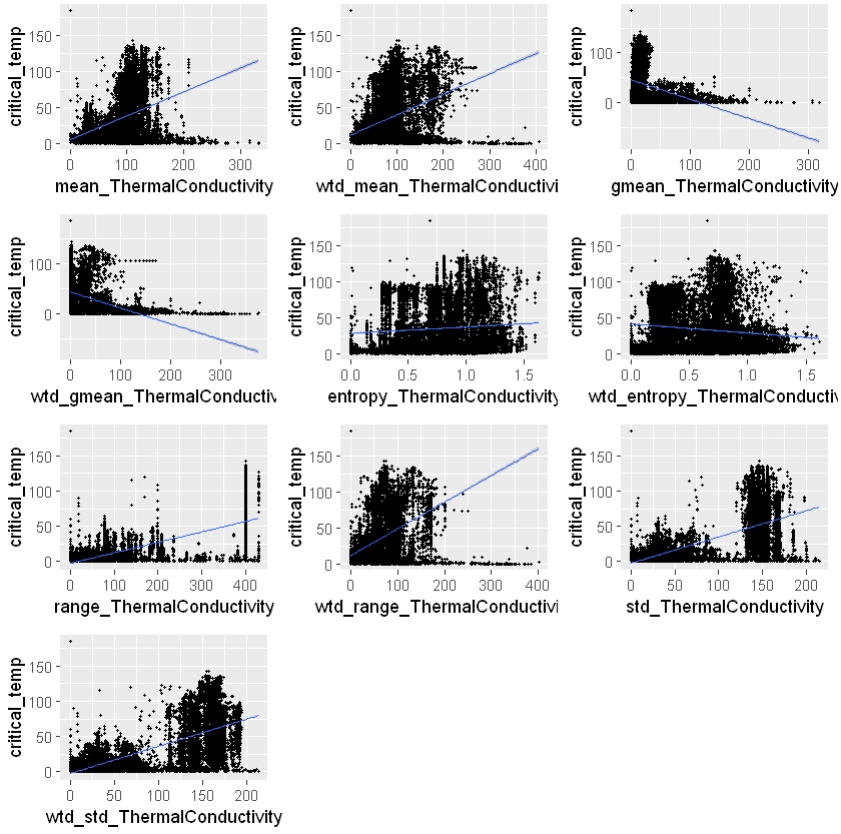
- mean_FusionHeat
- wtd_mean_FusionHeat
- gmean_FusionHeat
- wtd_gmean_FusionHeat
- range_FusionHeat
- wtd_range_FusionHeat
- std_FusionHeat
- wtd_std_FusionHeat

Thermal Conductivity:

In [57]:

```
plot_grid(plot_lst[[62]],plot_lst[[63]],plot_lst[[64]],plot_lst[[65]],
          plot_lst[[66]],plot_lst[[67]],plot_lst[[68]],plot_lst[[69]],
          plot_lst[[70]],plot_lst[[71]], nrow = 4, ncol=3)
```

```
`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'
```



Here are some of the observations from the above plots:

Positively related to critical_temp:

- mean_ThermalConductivity
- wtd_mean_ThermalConductivity
- entropy_ThermalConductivity
- range_ThermalConductivity
- wtd_range_ThermalConductivity
- std_ThermalConductivity
- wtd_std_ThermalConductivity

Negatively related to critical_temp:

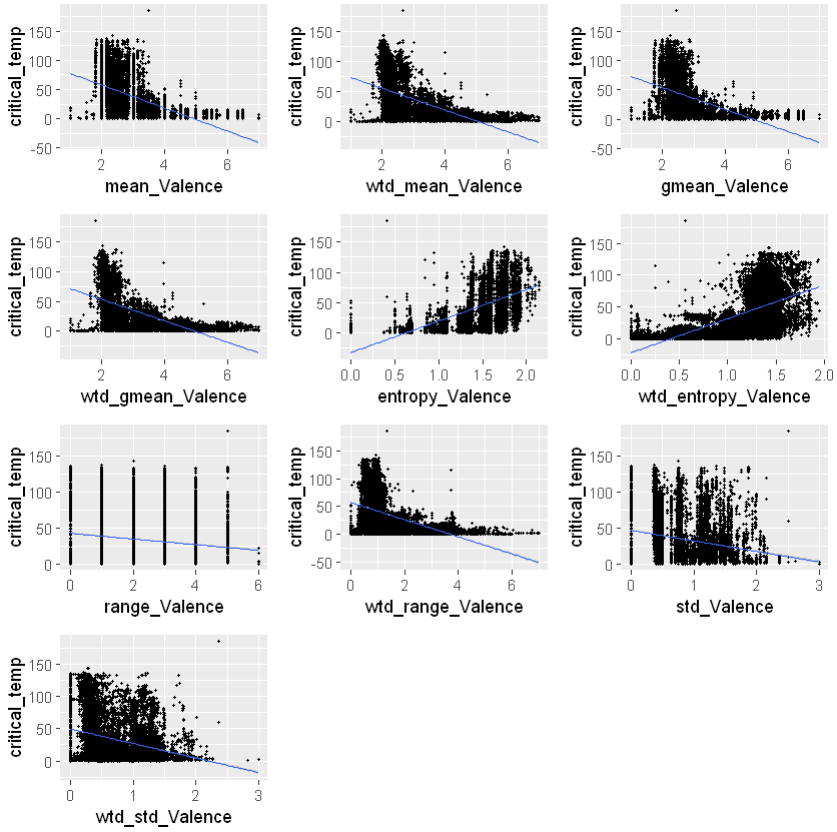
- gmean_ThermalConductivity
- wtd_gmean_ThermalConductivity
- wtd_entropy_ThermalConductivity

Valence:

In [58]:

```
plot_grid(plot_lst[[72]],plot_lst[[73]],plot_lst[[74]],plot_lst[[75]],
          plot_lst[[76]],plot_lst[[77]],plot_lst[[78]],plot_lst[[79]],
          plot_lst[[80]],plot_lst[[81]], nrow = 4, ncol=3)
```

```
`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'
```



Here are some of the observations from the above plots:

Positively related to critical_temp:

- entropy_Valence
- wtd_entropy_Valence

Negatively related to critical_temp:

- mean_Valence
- wtd_mean_Valence
- gmean_Valence
- wtd_gmean_Valence
- range_Valence
- wtd_range_Valence
- std_Valence
- wtd_std_Valence

Intuition for quantifying the correlation of Predictors with Target Variables:

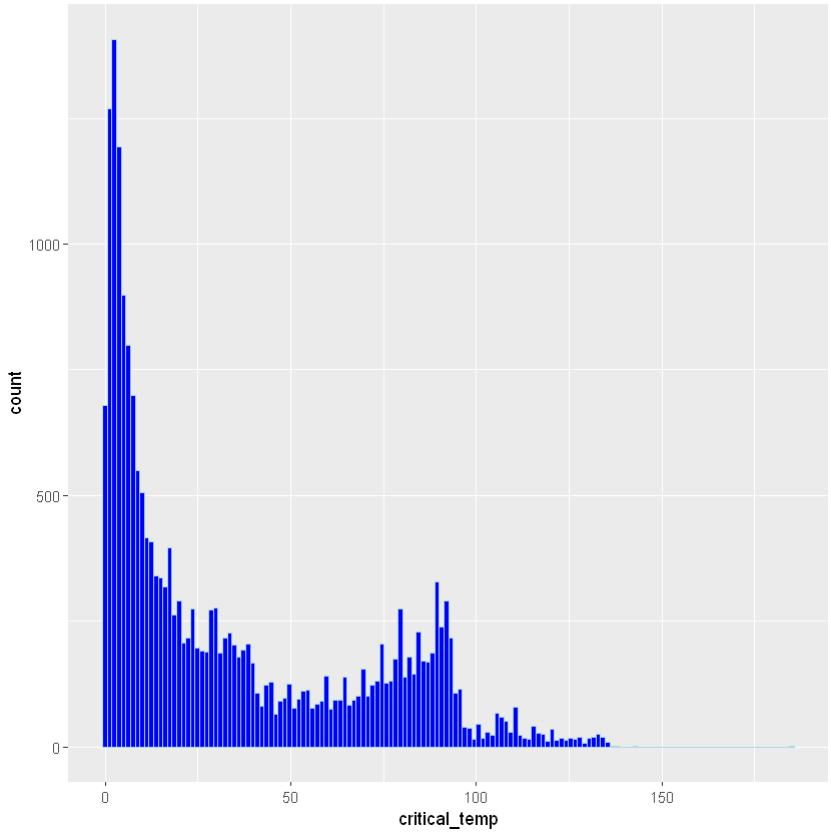
At this point, we only know that if a predictor is positively or negatively correlation by the usage of regression line. But we cannot quantify how much a predicor is related to the target i.e. upto how much extent a predictor can explain the target. This is the whole point of modelling, so that we can predict the coefficient of a variable. If the coefficient of a predictor is large then it can explain the target more as compared to the predictor with a low coefficient. Also, if a coefficient of a predictor is positive, that means the predictor and the target has a positive relationship(directly proportional). With a negative coefficient, the predictor and the target will have a negative relationship, i.e. if the predictor value will increase the target value will decrease(inversely proportional).

2.6 Distribution of the Superconducting Critical Temperatures:

The plot demonstrates the distribution of the critical temperatures. As we can see it is right skewed with a bump around 80K. The count of the superconducting materials that has a critical temperature between 0K and 20K is the greatest. Whereas most of the superconductor materials have critical temperature between 25K and 90K.

In [59]:

```
trainingData %>%
#   keep(is.numeric) %>%
#   gather() %>%
ggplot(aes(x=critical_temp)) +
#     facet_wrap(~ key, scales = "free", ncol=4) +
    geom_histogram(fill="blue",color="lightblue",bins=150) +
theme(strip.text = element_text(size = 8))
```



3. First Linear Model with ALL the Features:

Explaning the statistics in the summary:

1. **Residuals:** This section of the output gives the distance from the actual data to the fitted line. Or in simple words we can say that it gives the error between the predicted and the actual values.
   $Residual\ ei = yi\bar{}yi$
   Residuals should be symmetrically distributed about the fitted line. It gives us an idea how well our model fits our data. The lesse it is the better fit we have and a better model. Here, it gives us specific details like Minimum, 1st Quartile, Median, 3rd Quartile and Maximum.

2. **Intercept**: The co-efficient Intercept here gives us an estimate of the target variable, if no other predictor variable is taken into account to predict the target variable.

3. **Estimate**: The co-efficient Estimate here tells us how good our predictors x can estimate the target variable y.

4. **Std. Error**: The standard deviation of an estimate is called the standard error. The standard error of the coefficient measures how precisely the model estimates the coefficient's unknown value. The standard error of the coefficient is always positive. Lower the value of the std. error, better will be our analysis of the target variable.

5. **t value**: "The coefficient t-value is a measure of how many standard deviations our coefficient estimate is far away from 0. We want it to be far away from zero as this would indicate we could reject the null hypothesis. In general, t-values are also used to compute p-values."

6. **Pr(>|t|)**: "This number is the p-value, defined as the probability of observing any value equal or larger than t-value if H0 is true. The larger the t statistic, the smaller the p-value. This p-value is associated with the F statistic, and is used to interpret the significance for the whole model fit to our data. Generally, we use 0.05 as the good cutoff for significance; when p-values are smaller than 0.05, we reject H0. In our model, some of the p values are very close to 0. Three stars (or asterisks) represent a highly significant p-value. Consequently, a small p-value for the intercept and the slope indicates that we can reject the null hypothesis. The p-value in the table is calculated using the formula $(n-2)(1-P(T<=ti))$."

7. **Residual standard error**: "Residual Standard Error is measure of the quality of a linear regression fit. Theoretically, every linear model is assumed to contain an error term E. Due to the presence of this error term, we are not capable of perfectly predicting our target variable from the predictor one. The Residual Standard Error is the average amount that the target will deviate from the true regression line. It's also worth noting that the Residual Standard Error was calculated with 384 degrees of freedom. Simplistically, degrees of freedom are the number of data points that went into the estimation of the parameters used after taking into account these parameters (restriction). DoF are related to, but not the same as, the number of measurements."

8. **Multiple R-squared**: "The R-squared (R2) statistic provides a measure of how well the model is fitting the actual data. It takes the form of a proportion of variance. R2 is a measure of the linear relationship between our predictor variable and our target variable. It always lies between 0 and 1 (i.e.: a number near 0 represents a regression that does not explain the variance in the response variable well and a number close to 1 does explain the observed variance in the response variable). In multiple regression, the R2 will always increase as more variables are included in the model. That's why the adjusted R2 is the preferred measure as it adjusts for the number of variables considered."

9. **Adjusted R-squared**: It is multiple R-squared plus the penalty of the features. It is more right to use Adjusted R2 rather than Multiple R2.

10. **F-Statistic**: In simple words F-statistic will tell if a group of variables is jointly significant. F-statistic must always be used in combination with a p value. F statistic will only give significant results for the group of variables jointly. If the results are significant we cannot say that all our variables is significant and therefore to find out the significant variables p value must also be considered with the F statistic. Also the F test compares the model with the null model, to decide whether the added predictors are significant or not.

In [60]:

```
# building the model
model1.all.features.lm = lm(critical_temp~., data = trainingData)
```

In [61]:

```
# printing the summary of the model
summary(model1.all.features.lm)
```

```
Call:
lm(formula = critical_temp ~ ., data = trainingData)

Residuals:
    Min      1Q  Median      3Q     Max 
-84.987  -9.370   0.595  10.976 171.246 

Coefficients:
                              Estimate Std. Error t value Pr(>|t|)    
(Intercept)                 -2.081e+01  4.991e+00  -4.169 3.07e-05 ***
number_of_elements          -3.496e+00  7.480e-01  -4.674 2.97e-06 ***
mean_atomic_mass             8.480e-01  8.274e-02  10.249  < 2e-16 ***
wtd_mean_atomic_mass        -9.041e-01  1.031e-01  -8.773  < 2e-16 ***
gmean_atomic_mass           -5.102e-01  8.194e-02  -6.226 4.86e-10 ***
wtd_gmean_atomic_mass        6.468e-01  9.764e-02   6.625 3.57e-11 ***
entropy_atomic_mass         -3.596e+01  4.599e+00  -7.819 5.58e-15 ***
wtd_entropy_atomic_mass      4.555e+00  3.638e+00   1.252 0.210617    
range_atomic_mass            2.142e-01  1.653e-02  12.957  < 2e-16 ***
wtd_range_atomic_mass        2.598e-02  2.208e-02   1.177 0.239203    
std_atomic_mass             -5.608e-01  6.276e-02  -8.936  < 2e-16 ***
wtd_std_atomic_mass          9.048e-02  5.465e-02   1.656 0.097782 .  
mean_fie                     1.601e-01  6.339e-02   2.526 0.011553 *  
wtd_mean_fie                -1.787e-01  7.733e-02  -2.311 0.020843 *  
gmean_fie                   -1.524e-01  6.265e-02  -2.432 0.015012 *  
wtd_gmean_fie                1.984e-01  7.637e-02   2.598 0.009379 ** 
entropy_fie                 -1.187e+02  2.019e+01  -5.879 4.19e-09 ***
wtd_entropy_fie              4.414e+01  4.776e+00   9.243  < 2e-16 ***
range_fie                    6.821e-02  6.411e-03  10.640  < 2e-16 ***
wtd_range_fie                2.147e-02  3.628e-03   5.917 3.32e-09 ***
std_fie                     -1.986e-01  2.240e-02  -8.866  < 2e-16 ***
wtd_std_fie                 -2.323e-02  2.061e-02  -1.127 0.259721    
mean_atomic_radius          -5.351e-01  1.798e-01  -2.976 0.002926 ** 
wtd_mean_atomic_radius       3.257e+00  2.411e-01  13.507  < 2e-16 ***
gmean_atomic_radius          2.009e-01  1.801e-01   1.115 0.264757    
wtd_gmean_atomic_radius     -2.862e+00  2.361e-01 -12.124  < 2e-16 ***
entropy_atomic_radius        7.932e+01  1.760e+01   4.506 6.64e-06 ***
wtd_entropy_atomic_radius    4.482e+01  5.308e+00   8.445  < 2e-16 ***
range_atomic_radius          1.952e-01  2.217e-02   8.806  < 2e-16 ***
wtd_range_atomic_radius     -9.148e-02  1.595e-02  -5.735 9.91e-09 ***
std_atomic_radius           -3.558e-01  9.808e-02  -3.628 0.000286 ***
wtd_std_atomic_radius       -3.318e-01  8.782e-02  -3.778 0.000159 ***
mean_Density                -4.752e-03  5.008e-04  -9.490  < 2e-16 ***
wtd_mean_Density            -1.664e-04  6.188e-04  -0.269 0.788063    
gmean_Density                1.167e-03  4.736e-04   2.463 0.013769 *  
wtd_gmean_Density            2.328e-03  5.904e-04   3.943 8.07e-05 ***
entropy_Density              1.607e+01  3.508e+00   4.582 4.63e-06 ***
wtd_entropy_Density         -1.979e+01  2.695e+00  -7.343 2.16e-13 ***
range_Density               -1.582e-03  2.159e-04  -7.325 2.47e-13 ***
wtd_range_Density           -5.202e-05  2.611e-04  -0.199 0.842109    
std_Density                  6.088e-03  6.973e-04   8.730  < 2e-16 ***
wtd_std_Density             -1.584e-03  5.164e-04  -3.067 0.002165 ** 
mean_ElectronAffinity       -1.096e-01  4.654e-02  -2.355 0.018511 *  
wtd_mean_ElectronAffinity    5.172e-01  5.052e-02  10.237  < 2e-16 ***
```

```
gmean_ElectronAffinity           1.801e-01  4.034e-02   4.465 8.06e-06 ***
wtd_gmean_ElectronAffinity      -5.732e-01  4.456e-02 -12.864  < 2e-16 ***
entropy_ElectronAffinity         4.487e+00  2.588e+00   1.734 0.082958 .
wtd_entropy_ElectronAffinity    -2.122e+01  2.195e+00  -9.666  < 2e-16 ***
range_ElectronAffinity          -3.677e-01  1.736e-02 -21.180  < 2e-16 ***
wtd_range_ElectronAffinity      -1.389e-01  2.074e-02  -6.699 2.15e-11 ***
std_ElectronAffinity             1.243e+00  5.816e-02  21.375  < 2e-16 ***
wtd_std_ElectronAffinity        -5.428e-01  3.878e-02 -13.997  < 2e-16 ***
mean_FusionHeat                  1.733e+00  1.899e-01   9.125  < 2e-16 ***
wtd_mean_FusionHeat             -1.943e+00  1.946e-01  -9.984  < 2e-16 ***
gmean_FusionHeat                -1.558e+00  1.753e-01  -8.892  < 2e-16 ***
wtd_gmean_FusionHeat             1.620e+00  1.814e-01   8.933  < 2e-16 ***
entropy_FusionHeat              -1.946e+01  2.711e+00  -7.178 7.29e-13 ***
wtd_entropy_FusionHeat           2.544e+01  1.924e+00  13.219  < 2e-16 ***
range_FusionHeat                -3.708e-01  6.711e-02  -5.525 3.33e-08 ***
wtd_range_FusionHeat             6.024e-01  6.771e-02   8.897  < 2e-16 ***
std_FusionHeat                  -5.615e-01  2.625e-01  -2.139 0.032448 *
wtd_std_FusionHeat               7.597e-01  1.575e-01   4.823 1.42e-06 ***
mean_ThermalConductivity        -6.763e-02  2.474e-02  -2.734 0.006271 **
wtd_mean_ThermalConductivity     5.256e-01  2.813e-02  18.687  < 2e-16 ***
gmean_ThermalConductivity       -5.952e-02  2.349e-02  -2.534 0.011286 *
wtd_gmean_ThermalConductivity   -3.238e-01  2.722e-02 -11.895  < 2e-16 ***
entropy_ThermalConductivity      1.119e+01  1.995e+00   5.608 2.07e-08 ***
wtd_entropy_ThermalConductivity  1.483e+00  1.598e+00   0.928 0.353482
range_ThermalConductivity       -8.705e-02  1.336e-02  -6.517 7.34e-11 ***
wtd_range_ThermalConductivity   -2.286e-01  1.623e-02 -14.082  < 2e-16 ***
std_ThermalConductivity          2.818e-01  4.145e-02   6.797 1.10e-11 ***
wtd_std_ThermalConductivity     -7.988e-03  2.337e-02  -0.342 0.732470
mean_Valence                    -1.717e+01  6.046e+00  -2.840 0.004519 **
wtd_mean_Valence                 2.443e+01  7.204e+00   3.391 0.000698 ***
gmean_Valence                    2.126e+01  5.707e+00   3.725 0.000196 ***
wtd_gmean_Valence               -2.840e+01  6.755e+00  -4.204 2.64e-05 ***
entropy_Valence                  7.441e+01  1.229e+01   6.054 1.43e-09 ***
wtd_entropy_Valence             -7.037e+01  5.561e+00 -12.654  < 2e-16 ***
range_Valence                    4.790e+00  7.316e-01   6.547 6.02e-11 ***
wtd_range_Valence               -7.827e-01  6.307e-01  -1.241 0.214625
std_Valence                      6.943e+00  2.449e+00   2.835 0.004590 **
wtd_std_Valence                 -2.502e+01  1.906e+00 -13.127  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 17.59 on 21181 degrees of freedom
Multiple R-squared:  0.7373,    Adjusted R-squared:  0.7363
F-statistic: 733.8 on 81 and 21181 DF,  p-value: < 2.2e-16
```

Observations from the Model Summary above:

- **Degrees of Freedom**: 21181
- **Model Parameters**: 81 plus intercept
- **Residual standard error**: 17.59
- **Adjusted R-squared**: 0.7363
- **F-statistic**: 733.8 on 81 and 21181 DF

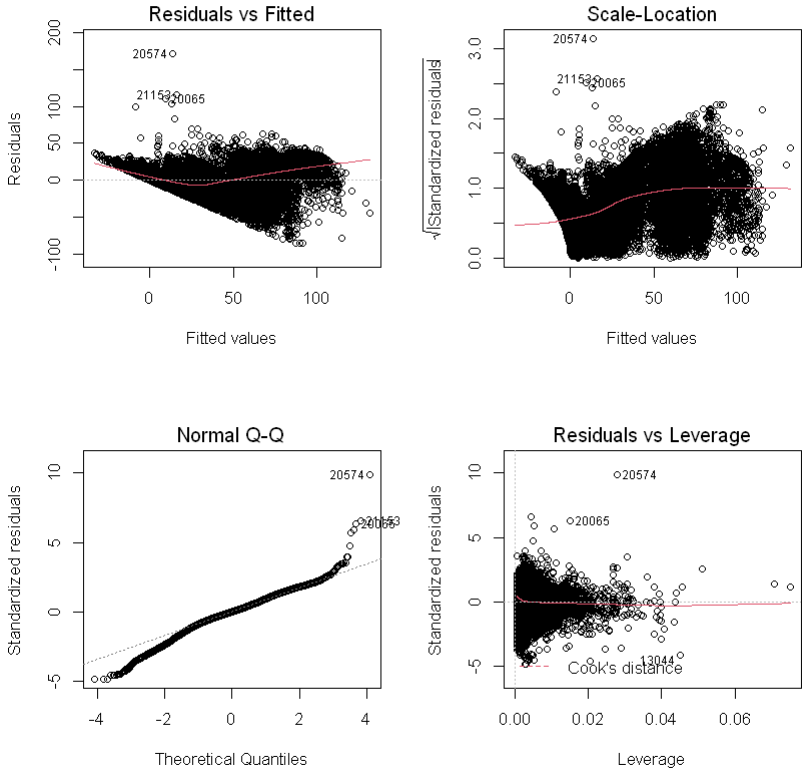
From these statistics the model looks fine but there are 81 variables which is quite a lot. Also,not all variables are significant as we can see in the summary. Many variables have one * or < >. Variables with 3 or 2 *'s can be considered as significant features. Also, it can be very expensive in terms of run-time to the linear regression model. Hence next we perform feature selection process so that our model's accuracy is fine with less number of variables.

The diagnostic plots below shows residuals in four different ways.

1. The **residual vs fitted plot**: This plot is used to check the linear assumption. It shows if residuals have non-linear patterns. If you find equally spread residuals around a horizontal line without distinct patterns, that is a good indication you have linear relationships. However, if the relationship between predictors and an response variable is non-linear, an obvious pattern could show up in this plot if the model cannot capture the non-linearity.
2. The normal **Q-Q plot**: The Q-Q plot (i.e., quantile-quantile plot) is a graphical tool to help us assess if a set of data plausibly came from some theoretical distribution such as a Normal. For example, if we run a statistical analysis that assumes our dependent variable is Normally distributed, we can use a Normal Q-Q plot to check that assumption. In the case of linear regression analysis, we assume that residual is normally distributed with constant variance and mean equal to zero. The normal Q-Q plot shows if residuals are normally distributed. Do residuals follow a straight line well or do they deviate severely? It is good if residuals are lined well on the straight dashed line.
3. The **scale-location plot**: It is used to check the assumption of equal variance by showing if residuals are spread equally along the ranges of predictors. It is good if we can see a horizontal line with equally (randomly) spread points.
4. The **residual-leverage plot**: it helps us identify influential data samples. Not all outliers are influential in linear regression analysis. Here we care about the samples that are influential to determine the regression line. These samples can be very influential even if they look to be within a reasonable range of the values. They can alter the results if we exclude them from analysis. In the residual-leverage plot, we look for outlying values at the upper right corner or at the lower right corner. Samples located in those places can be influential against a regression line. We usually use Cook’s distance, indicated by a red dash line. When samples are outside of the Cook’s distance (i.e, they have high Cook’s distance scores, Cook’s distance measures how much the entire regression function changes when the i-th case is deleted.), the samples are influential to the regression results. The regression results will be altered if we exclude those samples.

In [62]:

```
# plotting the model parameters
par(mfcol=c(2,2))
plot(model1.all.features.lm)
```



Observations from the above plots:

- **Residuals v/s Fitted**: It shows that the residuals have non linear pattern as they are not equally spread around the horizontal line and hence a distinct pattern can be seen in the plot.
- **Normal Q-Q plot**: It shows that most of the residuals are not lined up well on the straight dashed line and hence we cannot say that the residuals are normally distributed.
- **Scale- Location**: The residuals seem to be randomly spread along the line, also the line is not horizontal and it seems to be a curve more than a line. Hence the assumption of equal variance does not apply here.
- **Residuals vs Leverage**: Looking at the plot, we cannot find any outlaying values in the upper right corner or at the lower right corner, therefore no influential points can be seen that can influence the regression results. We can barely see Cook's distance lines because all cases are well inside of the Cook's distance lines. Therefore no influential cases are observed.

4. Applying Filters on Features:

4.1 Finding Redundant Features Using Pairwise Correlation Matrix:

Earlier we mentioned that when two variables are highly correlated we can remove them from the variable list. We also showed with correlation plots that which variables are highly correlated. Hence, in this section we are going to find those highly correlated variables(cutoff=0.80) and remove them from our list of variables.

In [159]:

```
# install.packages("mlbench")
# install.packages("caret")
library(mlbench)
library(caret)
```

The $findCorrelation()$ function from the $caret$ package finds which pairs of variables are highly correlated. It then checks which one of the variables in the pair is more correlated to the target, after which it decides which variable should be kept in the list of redundant variables and which variable should be kept in the model.

In [160]:

```
# ensure the results are repeatable
set.seed(7)
x=findCorrelation(cor(trainingData), cutoff=0.80, verbose = FALSE, names = FALSE, exact = 
paste("Number of features that are highly correlated-> ",length(x))
paste("The features which are Highly Correlated and need to be Filtered are: ->")
x
```

'Number of features that are highly correlated-> 52'

'The features which are Highly Correlated and need to be Filtered are: ->'

18 · 21 · 27 · 28 · 31 · 76 · 77 · 16 · 20 · 7 · 1 · 35 · 26 · 34 · 68 · 75 ·
73 · 71 · 6 · 57 · 25 · 72 · 46 · 56 · 13 · 33 · 8 · 23 · 5 · 36 · 17 · 55 ·
54 · 11 · 64 · 48 · 51 · 53 · 3 · 4 · 24 · 9 · 38 · 43 · 12 · 40 · 61 · 42 ·
63 · 58 · 81 · 80

In [161]:

```
# removing the redundant features
corTrainingData=subset(trainingData, select=-c(18, 21, 27, 28, 31, 76, 77, 16, 20, 7, 1, 3
                                               71, 6, 57, 25, 72, 46, 56, 13, 33, 8, 23, 5
                                               64, 48, 51, 53, 3, 4, 24, 9, 38, 43, 12, 40
paste("Number of Rows and Columns of the New Filtered Dataset->")
paste(dim(corTrainingData))
paste("Variables left are removing highly correlated values are->")
sapply(corTrainingData,class)
```

'Number of Rows and Columns of the New Filtered Dataset->'

'21263' · '30'

'Variables left are removing highly correlated values are->'

mean_atomic_mass: 'numeric' **std_atomic_mass:** 'numeric' **gmean_fie:** 'numeric'
wtd_gmean_fie: 'numeric' **wtd_range_fie:** 'numeric' **mean_atomic_radius:** 'numeric'
wtd_range_atomic_radius: 'numeric' **std_atomic_radius:** 'numeric' **mean_Density:**
'numeric' **wtd_entropy_Density:** 'numeric' **wtd_range_Density:** 'numeric' **wtd_std_Density:**
'numeric' **gmean_ElectronAffinity:** 'numeric' **wtd_gmean_ElectronAffinity:** 'numeric'
wtd_entropy_ElectronAffinity: 'numeric' **wtd_range_ElectronAffinity:** 'numeric'
std_ElectronAffinity: 'numeric' **mean_FusionHeat:** 'numeric' **wtd_range_FusionHeat:**
'numeric' **std_FusionHeat:** 'numeric' **mean_ThermalConductivity:** 'numeric'
wtd_gmean_ThermalConductivity: 'numeric' **entropy_ThermalConductivity:** 'numeric'
wtd_entropy_ThermalConductivity: 'numeric' **wtd_range_ThermalConductivity:** 'numeric'
std_ThermalConductivity: 'numeric' **gmean_Valence:** 'numeric' **range_Valence:** 'integer'
wtd_range_Valence: 'numeric' **critical_temp:** 'numeric'

4.2 Information Gain for Variable Selection:

Information gain tells us how much information is given by the independent variable about the dependent variable. Information gain is helpful in case of both categorical and numerical dependent variable. For numeric dependent variables, bins are created.

Although there are many functions, we are using $information.gain()$ function from $FSelector$ package.

In [162]:

```
# install.packages("FSelector")
library(FSelector)

# applying information gain method
infoGain<-information.gain(critical_temp~., corTrainingData)

# displaying importance of variables
paste("Number of Important Features ->")
paste(dim(infoGain))
infoGain
```

'Number of Important Features ->'

'29' · '1'

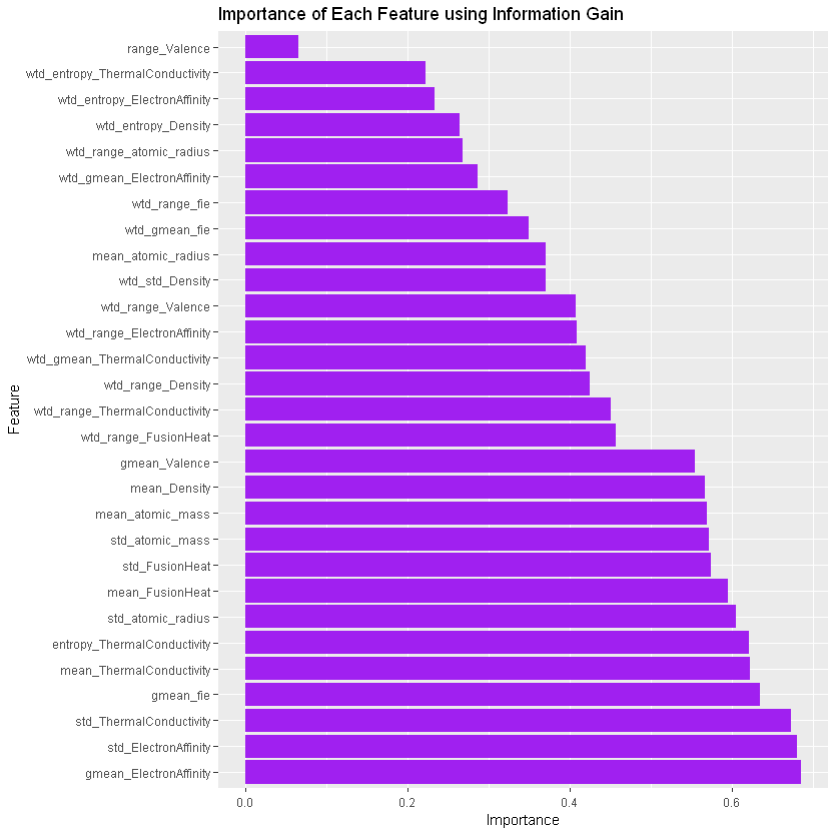
A data.frame: 29 × 1

	attr_importance
	<dbl>
mean_atomic_mass	0.56859295
std_atomic_mass	0.57218930
gmean_fie	0.63430508
wtd_gmean_fie	0.34862158
wtd_range_fie	0.32377864
mean_atomic_radius	0.37045159
wtd_range_atomic_radius	0.26717154
std_atomic_radius	0.60528294
mean_Density	0.56666024
wtd_entropy_Density	0.26417252
wtd_range_Density	0.42469479
wtd_std_Density	0.37052099
gmean_ElectronAffinity	0.68496669
wtd_gmean_ElectronAffinity	0.28630627
wtd_entropy_ElectronAffinity	0.23339773
wtd_range_ElectronAffinity	0.40865558
std_ElectronAffinity	0.68000836
mean_FusionHeat	0.59495886
wtd_range_FusionHeat	0.45699779
std_FusionHeat	0.57403247
mean_ThermalConductivity	0.62242575
wtd_gmean_ThermalConductivity	0.41949232
entropy_ThermalConductivity	0.62096041

	attr_importance
	<dbl>
wtd_entropy_ThermalConductivity	0.22175868
wtd_range_ThermalConductivity	0.45027941
std_ThermalConductivity	0.67253541
gmean_Valence	0.55428754
range_Valence	0.06489444
wtd_range_Valence	0.40767350

In [74]:

```
# adding a column "Feature" to the dataframe so that it makes it easier for plotting
infoGain$Variables1<-rownames(infoGain)

# plotting a graph of the importance of the variables using information gain
ggplot(data=infoGain, aes(x=reorder(Variables1, -attr_importance), y=attr_importance)) +
        geom_bar(stat="identity",fill = "purple")+ coord_flip()+
        theme(text = element_text(size=9), axis.text.y = element_text(hjust=1))+
        labs(title="Importance of Each Feature using Information Gain",y = "Importance", x=
```



From the above plot we can say that "std_ElectronAffinity", "std_ThermalConductivity" and "gmean_electronAffinity" are one of the most important variables but "range_Valence" is the least important as it gives very less information about the target and therefore we will remove it from the list of appropriate features for the model.

In [75]:

```
# removing range_Valence
corTrainingData=subset(corTrainingData, select=-c(28))

# displaying the number of rows after removing range_Valence
paste("Number of Rows and Columns of the New Filtered Dataset After Information Gain Methoc
paste(dim(corTrainingData))

# displaying the variables now left
paste("Variables left are removing variables with very less information values are->")
sapply(corTrainingData,class)
```

'Number of Rows and Columns of the New Filtered Dataset After Information Gain Method->'

'21263' · '29'

'Variables left are removing variables with very less information values are->'

mean_atomic_mass: 'numeric' **std_atomic_mass:** 'numeric' **gmean_fie:** 'numeric'
wtd_gmean_fie: 'numeric' **wtd_range_fie:** 'numeric' **mean_atomic_radius:** 'numeric'
wtd_range_atomic_radius: 'numeric' **std_atomic_radius:** 'numeric' **mean_Density:**
'numeric' **wtd_entropy_Density:** 'numeric' **wtd_range_Density:** 'numeric' **wtd_std_Density:**
'numeric' **gmean_ElectronAffinity:** 'numeric' **wtd_gmean_ElectronAffinity:** 'numeric'
wtd_entropy_ElectronAffinity: 'numeric' **wtd_range_ElectronAffinity:** 'numeric'
std_ElectronAffinity: 'numeric' **mean_FusionHeat:** 'numeric' **wtd_range_FusionHeat:**
'numeric' **std_FusionHeat:** 'numeric' **mean_ThermalConductivity:** 'numeric'
wtd_gmean_ThermalConductivity: 'numeric' **entropy_ThermalConductivity:** 'numeric'
wtd_entropy_ThermalConductivity: 'numeric' **wtd_range_ThermalConductivity:** 'numeric'
std_ThermalConductivity: 'numeric' **gmean_Valence:** 'numeric' **wtd_range_Valence:**
'numeric' **critical_temp:** 'numeric'

4.3 Top 20 Features Using mRMRe:

mRMRe means Parallelized **Minimum Redundancy, Maximum Relevance** Ensemble Feature Selection.
Using the function **mRMR.data()** from the **mRMRe** package, we will just show the 20 most important variables.
This step is just for the purpose of showing top 20 features and is not used to filter any variables.

In [78]:

```
# install.packages("mRMRe")
library(mRMRe)

myData <- mRMR.data(data = data.frame(corTrainingData))
tempMrMr<-mRMR.ensemble(data = myData, target_indices = 29,
              feature_count = 20, solution_count = 1)
```

```
Loading required package: survival

Loading required package: igraph


Attaching package: 'igraph'


The following object is masked from 'package:tidyr':

    crossing


The following objects are masked from 'package:purrr':

    compose, simplify


The following objects are masked from 'package:stats':

    decompose, spectrum


The following object is masked from 'package:base':

    union
```

In [79]:

```
# displaying the indices of the top 20 features using mRMRe
paste("The indices and scores of 20 Most Important Features Using mRMRe->")
paste(tempMrMr@filters)
tempMrMr
```

'The indices and scores of 20 Most Important Features Using mRMRe->'

'c(4, 12, 23, 6, 7, 19, 9, 5, 17, 22, 21, 2, 27, 18, 28, 25, 10, 8, 13, 26)'

```
Formal class 'mRMRe.Filter' [package "mRMRe"] with 8 slots
  ..@ filters       :List of 1
  .. ..$ 29: int [1:20, 1] 4 12 23 6 7 19 9 5 17 22 ...
  ..@ scores        :List of 1
  .. ..$ 29: num [1:20, 1] -0.0216 -0.01583 -0.01125 -0.00791 -0.00666 ...
  ..@ mi_matrix     : num [1:29, 1:29] NA 0.1965 NA NA -0.0805 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:29] "mean_atomic_mass" "std_atomic_mass" "gmean_fie" "w
td_gmean_fie" ...
  .. .. ..$ : chr [1:29] "mean_atomic_mass" "std_atomic_mass" "gmean_fie" "w
td_gmean_fie" ...
  ..@ causality_list:List of 1
  .. ..$ 29: num [1:29] NaN -0.00348 NaN -0.0142 -0.02441 ...
  ..@ sample_names  : chr [1:21263] "1" "2" "3" "4" ...
  ..@ feature_names : chr [1:29] "mean_atomic_mass" "std_atomic_mass" "gmean
_fie" "wtd_gmean_fie" ...
  ..@ target_indices: int 29
  ..@ levels        : int [1:20] 1 1 1 1 1 1 1 1 1 ...
```

In [80]:

```
# selecting those top 20 features which we found using mRMRe
mRMReTrainingData=subset(corTrainingData, select=c(4, 12, 23, 6, 7, 19, 9, 5, 17, 22, 21, 
paste("Top 20 most Important Features using mRMRe are ->")
sapply(mRMReTrainingData,class)
```

'Top 20 most Important Features using mRMRe are ->'

wtd_gmean_fie: 'numeric' **wtd_std_Density:** 'numeric' **entropy_ThermalConductivity:** 'numeric' **mean_atomic_radius:** 'numeric' **wtd_range_atomic_radius:** 'numeric'
wtd_range_FusionHeat: 'numeric' **mean_Density:** 'numeric' **wtd_range_fie:** 'numeric'
std_ElectronAffinity: 'numeric' **wtd_gmean_ThermalConductivity:** 'numeric'
mean_ThermalConductivity: 'numeric' **std_atomic_mass:** 'numeric' **gmean_Valence:** 'numeric' **mean_FusionHeat:** 'numeric' **wtd_range_Valence:** 'numeric'
wtd_range_ThermalConductivity: 'numeric' **wtd_entropy_Density:** 'numeric'
std_atomic_radius: 'numeric' **gmean_ElectronAffinity:** 'numeric'
std_ThermalConductivity: 'numeric'

5. Second Linear Model with Filtered Features:

In [81]:

```
# building second linear model with the filtered variables
model2.filtered.features.lm = lm(critical_temp~., data = corTrainingData)
```

In [82]:

```
# displaying the summary of the second model
summary(model2.filtered.features.lm)
```

```
Call:
lm(formula = critical_temp ~ ., data = corTrainingData)

Residuals:
     Min       1Q   Median       3Q      Max 
-127.188  -13.005   -0.474   13.376  170.158 

Coefficients:
                                Estimate Std. Error t value Pr(>|t|)    
(Intercept)                   -9.424e+01  4.788e+00 -19.683  < 2e-16 ***
mean_atomic_mass               1.438e-01  1.228e-02  11.713  < 2e-16 ***
std_atomic_mass                2.625e-01  1.295e-02  20.262  < 2e-16 ***
gmean_fie                      1.107e-01  5.346e-03  20.713  < 2e-16 ***
wtd_gmean_fie                 -3.817e-02  5.015e-03  -7.611 2.82e-14 ***
wtd_range_fie                  1.070e-02  2.131e-03   5.021 5.18e-07 ***
mean_atomic_radius             2.779e-01  1.619e-02  17.161  < 2e-16 ***
wtd_range_atomic_radius       -1.514e-01  9.913e-03 -15.270  < 2e-16 ***
std_atomic_radius              3.329e-01  1.661e-02  20.047  < 2e-16 ***
mean_Density                  -3.280e-03  1.733e-04 -18.932  < 2e-16 ***
wtd_entropy_Density            6.850e+00  1.187e+00   5.772 7.92e-09 ***
wtd_range_Density              1.659e-03  1.236e-04  13.425  < 2e-16 ***
wtd_std_Density               -2.909e-03  1.664e-04 -17.484  < 2e-16 ***
gmean_ElectronAffinity         2.616e-02  1.314e-02   1.991   0.0465 *  
wtd_gmean_ElectronAffinity    -7.464e-02  1.489e-02  -5.011 5.45e-07 ***
wtd_entropy_ElectronAffinity  -3.864e+01  1.383e+00 -27.936  < 2e-16 ***
wtd_range_ElectronAffinity    -2.381e-01  1.594e-02 -14.933  < 2e-16 ***
std_ElectronAffinity           1.060e-01  1.103e-02   9.608  < 2e-16 ***
mean_FusionHeat                2.122e-01  2.983e-02   7.114 1.16e-12 ***
wtd_range_FusionHeat          -2.293e-02  2.513e-02  -0.912   0.3615    
std_FusionHeat                -5.554e-01  3.568e-02 -15.565  < 2e-16 ***
mean_ThermalConductivity       1.658e-01  1.185e-02  13.997  < 2e-16 ***
wtd_gmean_ThermalConductivity -2.923e-01  1.013e-02 -28.841  < 2e-16 ***
entropy_ThermalConductivity    2.136e+01  1.218e+00  17.537  < 2e-16 ***
wtd_entropy_ThermalConductivity  9.265e+00  1.248e+00   7.424 1.18e-13 ***
wtd_range_ThermalConductivity  2.635e-01  9.364e-03  28.142  < 2e-16 ***
std_ThermalConductivity        1.046e-01  1.257e-02   8.320  < 2e-16 ***
gmean_Valence                  2.738e+00  3.535e-01   7.747 9.83e-15 ***
wtd_range_Valence             -1.976e+00  3.582e-01  -5.516 3.50e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 20.66 on 21234 degrees of freedom
Multiple R-squared:  0.6367,    Adjusted R-squared:  0.6362 
F-statistic:  1329 on 28 and 21234 DF,  p-value: < 2.2e-16
```

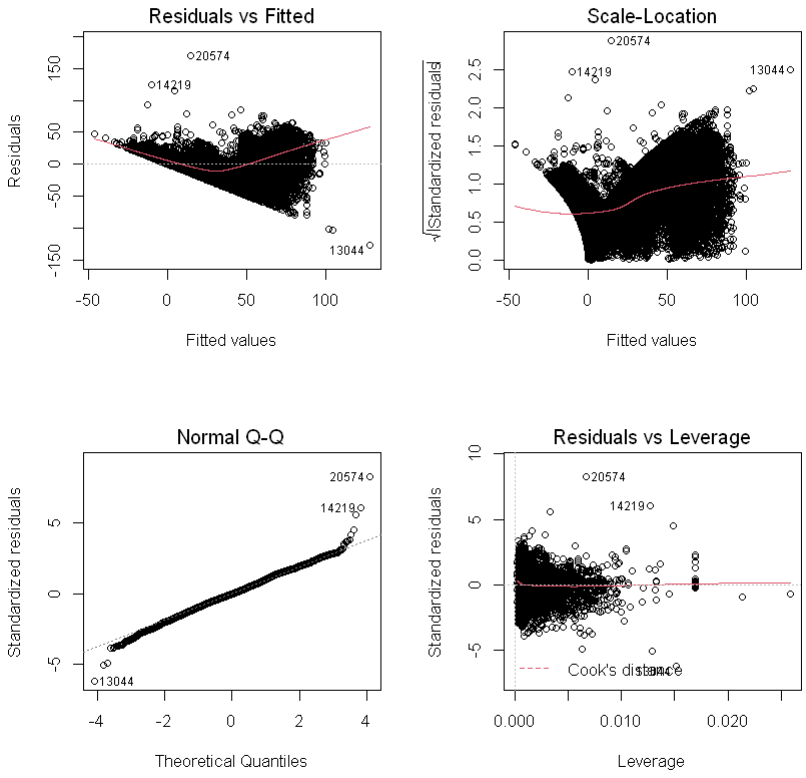
Observations from the Model Summary above:

- **Degrees of Freedom**: 21168
- **Model Parameters**: 28 plus intercept
- **Residual standard error**: 20.67
- **Adjusted R-squared**: 0.6363
- **F-statistic**: 1325 on 28 and 21168 DF

Here we notice that "Residual standard error" is more as compared to our first model. Also, "Adjusted R-squared" is less than our first model, which is unexpected. This model also has a couple of variables which are not significant. From this point onwards we will try some more feature selection using the latest set of variables.

In [83]:

```
par(mfcol=c(2,2))
plot(model2.filtered.features.lm)
```



Observations from the above plots:

- **Residuals v/s Fitted**: It still shows that the residuals have non linear pattern as they are not equally spread around the horizontal line and hence a distinct pattern can be seen in the plot.
- **Normal Q-Q plot**: Unlike our previous model, here we can see that most of the residuals are lined up well on the straight dashed line and hence we can say that the residuals are normally distributed.
- **Scale- Location**: Again, the residuals seem to be randomly spread along the line, also the line is not horizontal and it seems to be a curve more than a line. Hence the assumption of equal variance does not apply here.
- **Residuals vs Leverage**: Looking at the plot, we cannot find any outlaying values in the upper right corner or at the lower right corner, therefore no influential points can be seen that can influence the regression results. We can barely see Cook's distance lines because all cases are well inside of the Cook's distance lines. Therefore no influential cases are observed.

6. Stepwise Feature Selection:

Subset selection is an approach of identifying a subset of predictors (or attributes) that we believe have a strong association with the response variable, stepwise. We keep on adding and subtracting the variables in stepwise approach until we get the desired model.

1. **Mallows C_p**: Malllow calulated C_p by the below formula,

   C_p = SSres/MSres - N + 2p

   where

   SSres is the residual sum of squares for the model with p-1 variables,
   MSres is the residual mean square when using all available variables,
   N is the number of observations, and
   p is the number of variables used for the model plus one.

   The general procedure to find an adequate model by means of the Cp statistic is to calculate Cp for all possible combinations of variables and the Cp values against p. The model with the lowest Cp value approximately equal to p is the most "adequate" model.

2. **BIC**: When fitting models, it is possible to increase the likelihood by adding parameters, but doing so may result in overfitting. The BIC resolves this problem by introducing a penalty term for the number of parameters in the model.

   k log(n)- 2log(L(θ̂))

   where

   n is the number of data points
   k is the number of parameters which your model estimates
   θ is the set of all parameters.
   L(θ̂) is the maximised value of the likelihood function of the model

   The penalty term is larger in BIC than in AIC. Though these two measures are derived from a different perspective, they are closely related. Apparently, the only difference is BIC considers the number of observations in the formula, which AIC does not. Though BIC is always higher than AIC, lower the value of these two measures, better the model.

6.1 Best Subset Selection:

For best subset selection we will use $regsubsets()$ function the best model containing the latest set of predictors. And then we will decide the number of variables our best model should have based on graphs of C_p, BIC, Adjusted R2 and RSS. Best subset selection method is the best method as compared to forward/backward stepwise selection but in reality we do not use much of this method as it is quite expensive in terms of time complexity(O(2^p), where p is the number of predictors).

In [88]:

```
# building the model
# install.packages("leaps")
library(leaps)
regfit.full <- regsubsets(critical_temp~., data = corTrainingData, nvmax = 30)
reg.summary.full <- summary(regfit.full)
reg.summary.full
```

```
Subset selection object
Call: regsubsets.formula(critical_temp ~ ., data = corTrainingData,
    nvmax = 30)
28 Variables  (and intercept)
                               Forced in Forced out
mean_atomic_mass                   FALSE      FALSE
std_atomic_mass                    FALSE      FALSE
gmean_fie                          FALSE      FALSE
wtd_gmean_fie                      FALSE      FALSE
wtd_range_fie                      FALSE      FALSE
mean_atomic_radius                 FALSE      FALSE
wtd_range_atomic_radius            FALSE      FALSE
std_atomic_radius                  FALSE      FALSE
mean_Density                       FALSE      FALSE
wtd_entropy_Density                FALSE      FALSE
wtd_range_Density                  FALSE      FALSE
wtd_std_Density                    FALSE      FALSE
gmean_ElectronAffinity             FALSE      FALSE
wtd_gmean_ElectronAffinity         FALSE      FALSE
wtd_entropy_ElectronAffinity       FALSE      FALSE
wtd_range_ElectronAffinity         FALSE      FALSE
std_ElectronAffinity               FALSE      FALSE
mean_FusionHeat                    FALSE      FALSE
wtd_range_FusionHeat               FALSE      FALSE
std_FusionHeat                     FALSE      FALSE
mean_ThermalConductivity           FALSE      FALSE
wtd_gmean_ThermalConductivity      FALSE      FALSE
entropy_ThermalConductivity        FALSE      FALSE
wtd_entropy_ThermalConductivity    FALSE      FALSE
wtd_range_ThermalConductivity      FALSE      FALSE
std_ThermalConductivity            FALSE      FALSE
gmean_Valence                      FALSE      FALSE
wtd_range_Valence                  FALSE      FALSE
1 subsets of each size up to 28
Selection Algorithm: exhaustive
          mean_atomic_mass std_atomic_mass gmean_fie wtd_gmean_fie
1  ( 1 )  " "              " "             " "       " "
2  ( 1 )  " "              " "             " "       " "
3  ( 1 )  " "              " "             " "       " "
4  ( 1 )  " "              " "             " "       " "
5  ( 1 )  " "              " "             " "       " "
6  ( 1 )  " "              " "             " "       " "
7  ( 1 )  " "              " "             " "       " "
8  ( 1 )  " "              " "             " "       " "
9  ( 1 )  " "              " "             " "       " "
10  ( 1 ) " "              " "             " "       " "
11  ( 1 ) " "              "*"             " "       " "
12  ( 1 ) " "              "*"             " "       " "
13  ( 1 ) " "              "*"             " "       " "
14  ( 1 ) " "              "*"             " "       " "
```

```
15  ( 1 ) " "               "*"               "*"         " "
16  ( 1 ) " "               "*"               "*"         " "
17  ( 1 ) " "               "*"               "*"         " "
18  ( 1 ) " "               "*"               "*"         " "
19  ( 1 ) "*"               "*"               "*"         " "
20  ( 1 ) "*"               "*"               "*"         " "
21  ( 1 ) "*"               "*"               "*"         " "
22  ( 1 ) "*"               "*"               "*"         "*"
23  ( 1 ) "*"               "*"               "*"         "*"
24  ( 1 ) "*"               "*"               "*"         "*"
25  ( 1 ) "*"               "*"               "*"         "*"
26  ( 1 ) "*"               "*"               "*"         "*"
27  ( 1 ) "*"               "*"               "*"         "*"
28  ( 1 ) "*"               "*"               "*"         "*"
          wtd_range_fie mean_atomic_radius wtd_range_atomic_radius
1   ( 1 )  " "            " "                " "
2   ( 1 )  " "            " "                " "
3   ( 1 )  " "            " "                " "
4   ( 1 )  " "            " "                " "
5   ( 1 )  " "            " "                " "
6   ( 1 )  " "            " "                "*"
7   ( 1 )  " "            " "                "*"
8   ( 1 )  " "            " "                "*"
9   ( 1 )  " "            " "                " "
10  ( 1 ) " "             " "                " "
11  ( 1 ) " "             " "                " "
12  ( 1 ) " "             " "                " "
13  ( 1 ) " "             " "                "*"
14  ( 1 ) " "             " "                "*"
15  ( 1 ) " "             "*"                "*"
16  ( 1 ) " "             "*"                "*"
17  ( 1 ) " "             "*"                "*"
18  ( 1 ) " "             "*"                "*"
19  ( 1 ) " "             "*"                "*"
20  ( 1 ) " "             "*"                "*"
21  ( 1 ) " "             "*"                "*"
22  ( 1 ) " "             "*"                "*"
23  ( 1 ) " "             "*"                "*"
24  ( 1 ) " "             "*"                "*"
25  ( 1 ) " "             "*"                "*"
26  ( 1 ) "*"             "*"                "*"
27  ( 1 ) "*"             "*"                "*"
28  ( 1 ) "*"             "*"                "*"
          std_atomic_radius mean_Density wtd_entropy_Density wtd_range_D
ensity
1   ( 1 )  " "               " "          " "                 " "
2   ( 1 )  " "               " "          " "                 " "
3   ( 1 )  "*"               " "          " "                 " "
4   ( 1 )  " "               " "          " "                 " "
5   ( 1 )  "*"               " "          " "                 " "
6   ( 1 )  "*"               " "          " "                 " "
7   ( 1 )  "*"               " "          " "                 " "
8   ( 1 )  "*"               " "          " "                 " "
9   ( 1 )  "*"               " "          "*"                 " "
10  ( 1 ) "*"                " "          "*"                 " "
11  ( 1 ) "*"                " "          "*"                 " "
12  ( 1 ) "*"                " "          "*"                 " "
13  ( 1 ) "*"                " "          " "                 " "
14  ( 1 ) "*"                " "          "*"                 "*"
15  ( 1 ) "*"                " "          " "                 " "
16  ( 1 ) "*"                " "          " "                 " "
```

```
17  ( 1 ) "*"                 "*"             "*"                   "*"
18  ( 1 ) "*"                 "*"             "*"                   "*"
19  ( 1 ) "*"                 "*"             "*"                   "*"
20  ( 1 ) "*"                 "*"             "*"                   "*"
21  ( 1 ) "*"                 "*"             "*"                   "*"
22  ( 1 ) "*"                 "*"             "*"                   "*"
23  ( 1 ) "*"                 "*"             " "                   "*"
24  ( 1 ) "*"                 "*"             "*"                   "*"
25  ( 1 ) "*"                 "*"             "*"                   "*"
26  ( 1 ) "*"                 "*"             "*"                   "*"
27  ( 1 ) "*"                 "*"             "*"                   "*"
28  ( 1 ) "*"                 "*"             "*"                   "*"
          wtd_std_Density gmean_ElectronAffinity wtd_gmean_ElectronAffin
ity
1  ( 1 )  " "               " "                  " "
2  ( 1 )  " "               "*"                  " "
3  ( 1 )  " "               " "                  "*"
4  ( 1 )  " "               "*"                  " "
5  ( 1 )  " "               " "                  "*"
6  ( 1 )  " "               " "                  "*"
7  ( 1 )  " "               " "                  "*"
8  ( 1 )  " "               " "                  "*"
9  ( 1 )  " "               " "                  " "
10  ( 1 ) " "               " "                  " "
11  ( 1 ) " "               " "                  " "
12  ( 1 ) "*"               " "                  " "
13  ( 1 ) "*"               " "                  " "
14  ( 1 ) "*"               " "                  " "
15  ( 1 ) "*"               " "                  " "
16  ( 1 ) "*"               " "                  " "
17  ( 1 ) "*"               " "                  " "
18  ( 1 ) "*"               " "                  " "
19  ( 1 ) "*"               " "                  " "
20  ( 1 ) "*"               " "                  " "
21  ( 1 ) "*"               " "                  " "
22  ( 1 ) "*"               " "                  " "
23  ( 1 ) "*"               " "                  "*"
24  ( 1 ) "*"               " "                  "*"
25  ( 1 ) "*"               " "                  "*"
26  ( 1 ) "*"               " "                  "*"
27  ( 1 ) "*"               "*"                  "*"
28  ( 1 ) "*"               "*"                  "*"
          wtd_entropy_ElectronAffinity wtd_range_ElectronAffinity
1  ( 1 )  " "                          " "
2  ( 1 )  " "                          " "
3  ( 1 )  " "                          " "
4  ( 1 )  " "                          " "
5  ( 1 )  " "                          " "
6  ( 1 )  " "                          " "
7  ( 1 )  " "                          " "
8  ( 1 )  " "                          " "
9  ( 1 )  "*"                          "*"
10  ( 1 ) "*"                          "*"
11  ( 1 ) "*"                          "*"
12  ( 1 ) "*"                          "*"
13  ( 1 ) "*"                          "*"
14  ( 1 ) "*"                          "*"
15  ( 1 ) "*"                          "*"
16  ( 1 ) "*"                          "*"
17  ( 1 ) "*"                          "*"
18  ( 1 ) "*"                          "*"
```

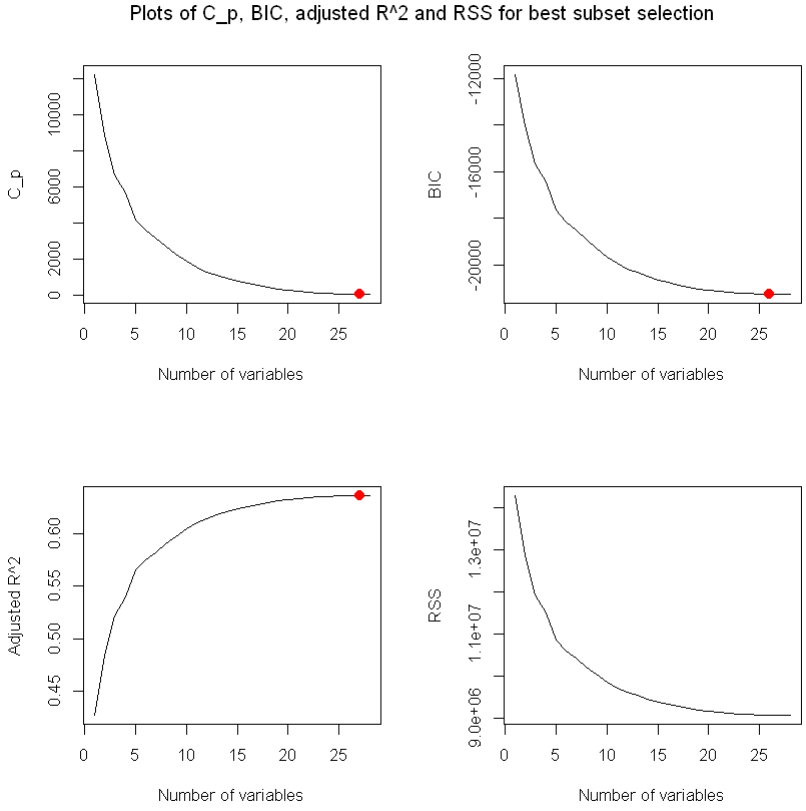
```
19  ( 1 ) "*"                          "*"
20  ( 1 ) "*"                          "*"
21  ( 1 ) "*"                          "*"
22  ( 1 ) "*"                          "*"
23  ( 1 ) "*"                          "*"
24  ( 1 ) "*"                          "*"
25  ( 1 ) "*"                          "*"
26  ( 1 ) "*"                          "*"
27  ( 1 ) "*"                          "*"
28  ( 1 ) "*"                          "*"
          std_ElectronAffinity mean_FusionHeat wtd_range_FusionHeat
1   ( 1 )  " "                  " "            " "
2   ( 1 )  " "                  " "            " "
3   ( 1 )  " "                  " "            " "
4   ( 1 )  " "                  " "            " "
5   ( 1 )  " "                  " "            " "
6   ( 1 )  " "                  " "            " "
7   ( 1 )  " "                  " "            " "
8   ( 1 )  " "                  " "            " "
9   ( 1 )  " "                  " "            " "
10  ( 1 ) "*"                   " "            " "
11  ( 1 ) "*"                   " "            " "
12  ( 1 ) "*"                   " "            " "
13  ( 1 ) "*"                   " "            " "
14  ( 1 ) "*"                   " "            " "
15  ( 1 ) "*"                   " "            " "
16  ( 1 ) "*"                   " "            " "
17  ( 1 ) "*"                   " "            " "
18  ( 1 ) "*"                   " "            " "
19  ( 1 ) "*"                   " "            " "
20  ( 1 ) "*"                   "*"            " "
21  ( 1 ) "*"                   " "            " "
22  ( 1 ) "*"                   " "            " "
23  ( 1 ) "*"                   "*"            " "
24  ( 1 ) "*"                   "*"            " "
25  ( 1 ) "*"                   "*"            " "
26  ( 1 ) "*"                   "*"            " "
27  ( 1 ) "*"                   "*"            " "
28  ( 1 ) "*"                   "*"            "*"
          std_FusionHeat mean_ThermalConductivity wtd_gmean_ThermalCondu
ctivity
1   ( 1 )  " "            " "                     " "
2   ( 1 )  " "            " "                     " "
3   ( 1 )  " "            " "                     " "
4   ( 1 )  " "            "*"                     "*"
5   ( 1 )  " "            "*"                     "*"
6   ( 1 )  " "            "*"                     "*"
7   ( 1 )  " "            "*"                     "*"
8   ( 1 )  "*"            "*"                     "*"
9   ( 1 )  "*"            "*"                     "*"
10  ( 1 ) "*"             "*"                     "*"
11  ( 1 ) "*"             "*"                     "*"
12  ( 1 ) "*"             "*"                     "*"
13  ( 1 ) "*"             "*"                     "*"
14  ( 1 ) "*"             "*"                     "*"
15  ( 1 ) "*"             " "                     "*"
16  ( 1 ) "*"             "*"                     "*"
17  ( 1 ) "*"             "*"                     "*"
18  ( 1 ) "*"             "*"                     "*"
19  ( 1 ) "*"             "*"                     "*"
20  ( 1 ) "*"             "*"                     "*"
```

```
21  ( 1 ) "*"                 "*"                       "*"
22  ( 1 ) "*"                 "*"                       "*"
23  ( 1 ) "*"                 "*"                       "*"
24  ( 1 ) "*"                 "*"                       "*"
25  ( 1 ) "*"                 "*"                       "*"
26  ( 1 ) "*"                 "*"                       "*"
27  ( 1 ) "*"                 "*"                       "*"
28  ( 1 ) "*"                 "*"                       "*"
          entropy_ThermalConductivity wtd_entropy_ThermalConductivity
1   ( 1 )  " "                        " "
2   ( 1 )  " "                        " "
3   ( 1 )  " "                        " "
4   ( 1 )  " "                        " "
5   ( 1 )  " "                        " "
6   ( 1 )  " "                        " "
7   ( 1 )  "*"                        " "
8   ( 1 )  "*"                        " "
9   ( 1 )  " "                        "*"
10  ( 1 ) "*"                         " "
11  ( 1 ) "*"                         " "
12  ( 1 ) "*"                         " "
13  ( 1 ) "*"                         "*"
14  ( 1 ) "*"                         " "
15  ( 1 ) "*"                         "*"
16  ( 1 ) "*"                         "*"
17  ( 1 ) "*"                         " "
18  ( 1 ) "*"                         "*"
19  ( 1 ) "*"                         "*"
20  ( 1 ) "*"                         "*"
21  ( 1 ) "*"                         "*"
22  ( 1 ) "*"                         "*"
23  ( 1 ) "*"                         "*"
24  ( 1 ) "*"                         "*"
25  ( 1 ) "*"                         "*"
26  ( 1 ) "*"                         "*"
27  ( 1 ) "*"                         "*"
28  ( 1 ) "*"                         "*"
          wtd_range_ThermalConductivity std_ThermalConductivity gmean_Va
lence
1   ( 1 )  " "                          "*"                     " "
2   ( 1 )  " "                          "*"                     " "
3   ( 1 )  " "                          "*"                     " "
4   ( 1 )  "*"                          " "                     " "
5   ( 1 )  "*"                          " "                     " "
6   ( 1 )  "*"                          " "                     " "
7   ( 1 )  "*"                          " "                     " "
8   ( 1 )  "*"                          " "                     " "
9   ( 1 )  "*"                          " "                     " "
10  ( 1 ) "*"                           " "                     " "
11  ( 1 ) "*"                           " "                     " "
12  ( 1 ) "*"                           " "                     " "
13  ( 1 ) "*"                           " "                     " "
14  ( 1 ) "*"                           " "                     " "
15  ( 1 ) "*"                           "*"                     " "
16  ( 1 ) "*"                           "*"                     " "
17  ( 1 ) "*"                           " "                     " "
18  ( 1 ) "*"                           " "                     " "
19  ( 1 ) "*"                           " "                     " "
20  ( 1 ) "*"                           " "                     " "
21  ( 1 ) "*"                           "*"                     "*"
22  ( 1 ) "*"                           "*"                     "*"
```

```
23  ( 1 ) "*"                                 "*"                                 "*"
24  ( 1 ) "*"                                 "*"                                 "*"
25  ( 1 ) "*"                                 "*"                                 "*"
26  ( 1 ) "*"                                 "*"                                 "*"
27  ( 1 ) "*"                                 "*"                                 "*"
28  ( 1 ) "*"                                 "*"                                 "*"
          wtd_range_Valence
1   ( 1 )  " "
2   ( 1 )  " "
3   ( 1 )  " "
4   ( 1 )  " "
5   ( 1 )  " "
6   ( 1 )  " "
7   ( 1 )  " "
8   ( 1 )  " "
9   ( 1 )  " "
10  ( 1 ) " "
11  ( 1 ) " "
12  ( 1 ) " "
13  ( 1 ) " "
14  ( 1 ) " "
15  ( 1 ) " "
16  ( 1 ) " "
17  ( 1 ) " "
18  ( 1 ) " "
19  ( 1 ) " "
20  ( 1 ) " "
21  ( 1 ) " "
22  ( 1 ) " "
23  ( 1 ) " "
24  ( 1 ) " "
25  ( 1 ) "*"
26  ( 1 ) "*"
27  ( 1 ) "*"
28  ( 1 ) "*"
```

In [89]:

```
par(mfrow = c(2, 2))
plot(reg.summary.full$cp, xlab = "Number of variables", ylab = "C_p", type = "l")
points(which.min(reg.summary.full$cp), reg.summary.full$cp[which.min(reg.summary.full$cp)],
plot(reg.summary.full$bic, xlab = "Number of variables", ylab = "BIC", type = "l")
points(which.min(reg.summary.full$bic), reg.summary.full$bic[which.min(reg.summary.full$bi
plot(reg.summary.full$adjr2, xlab = "Number of variables", ylab = "Adjusted R^2", type = "
points(which.max(reg.summary.full$adjr2), reg.summary.full$adjr2[which.max(reg.summary.ful
       cex = 2, pch = 20)
plot(reg.summary.full$rss, xlab = "Number of variables", ylab = "RSS", type = "l")
mtext("Plots of C_p, BIC, adjusted R^2 and RSS for best subset selection", side = 3, line =
```



Observations from the above plots:

- C_p is lowest for 28 variables.
- BIC is lowest for 27 variables.
- Adjusted R2 is maximum for 28 variables.
- RSS is lowest for 28 variables.

  Examining the above plots, we can say that the number of predictors that will be best for our model is 28 variables.

6.2 Forward Stepwise Selection:

In forward selection type of feature selection method, it first begins with an empty model and then one by one, it keeps on adding variables until we get a single perfect model. It comprises of an absolute accuracy as it cannot remove a variable if the combination is not giving the best model. Although it is computationally cheaper(O(p^2)), it is unlikely to get the best subset as compared to the Best Subset Selection Method.

In [90]:

```
# Building the model
regfit.fwd <- regsubsets(critical_temp~., data = corTrainingData, nvmax = 30, method="forwa
reg.summary.fwd <- summary(regfit.fwd)
reg.summary.fwd
```

```
Subset selection object
Call: regsubsets.formula(critical_temp ~ ., data = corTrainingData,
    nvmax = 30, method = "forward")
28 Variables  (and intercept)
                              Forced in Forced out
mean_atomic_mass                  FALSE      FALSE
std_atomic_mass                   FALSE      FALSE
gmean_fie                         FALSE      FALSE
wtd_gmean_fie                     FALSE      FALSE
wtd_range_fie                     FALSE      FALSE
mean_atomic_radius                FALSE      FALSE
wtd_range_atomic_radius           FALSE      FALSE
std_atomic_radius                 FALSE      FALSE
mean_Density                      FALSE      FALSE
wtd_entropy_Density               FALSE      FALSE
wtd_range_Density                 FALSE      FALSE
wtd_std_Density                   FALSE      FALSE
gmean_ElectronAffinity            FALSE      FALSE
wtd_gmean_ElectronAffinity        FALSE      FALSE
wtd_entropy_ElectronAffinity      FALSE      FALSE
wtd_range_ElectronAffinity        FALSE      FALSE
std_ElectronAffinity              FALSE      FALSE
mean_FusionHeat                   FALSE      FALSE
wtd_range_FusionHeat              FALSE      FALSE
std_FusionHeat                    FALSE      FALSE
mean_ThermalConductivity          FALSE      FALSE
wtd_gmean_ThermalConductivity     FALSE      FALSE
entropy_ThermalConductivity       FALSE      FALSE
wtd_entropy_ThermalConductivity   FALSE      FALSE
wtd_range_ThermalConductivity     FALSE      FALSE
std_ThermalConductivity           FALSE      FALSE
gmean_Valence                     FALSE      FALSE
wtd_range_Valence                 FALSE      FALSE
1 subsets of each size up to 28
Selection Algorithm: forward
          mean_atomic_mass std_atomic_mass gmean_fie wtd_gmean_fie
1  ( 1 )  " "              " "             " "       " "
2  ( 1 )  " "              " "             " "       " "
3  ( 1 )  " "              " "             " "       " "
4  ( 1 )  " "              " "             " "       " "
5  ( 1 )  " "              " "             " "       " "
6  ( 1 )  " "              " "             " "       " "
7  ( 1 )  " "              " "             " "       " "
8  ( 1 )  " "              " "             "*"       " "
9  ( 1 )  " "              " "             "*"       " "
10  ( 1 ) " "              " "             "*"       " "
11  ( 1 ) " "              " "             "*"       " "
12  ( 1 ) " "              " "             "*"       " "
13  ( 1 ) " "              "*"             "*"       " "
14  ( 1 ) " "              "*"             "*"       " "
15  ( 1 ) " "              "*"             "*"       " "
16  ( 1 ) " "              "*"             "*"       " "
```

```
17  ( 1 ) " "               "*"               "*"         " "
18  ( 1 ) " "               "*"               "*"         " "
19  ( 1 ) " "               "*"               "*"         " "
20  ( 1 ) " "               "*"               "*"         " "
21  ( 1 ) " "               "*"               "*"         " "
22  ( 1 ) "*"               "*"               "*"         " "
23  ( 1 ) "*"               "*"               "*"         " "
24  ( 1 ) "*"               "*"               "*"         "*"
25  ( 1 ) "*"               "*"               "*"         "*"
26  ( 1 ) "*"               "*"               "*"         "*"
27  ( 1 ) "*"               "*"               "*"         "*"
28  ( 1 ) "*"               "*"               "*"         "*"
          wtd_range_fie mean_atomic_radius wtd_range_atomic_radius
1   ( 1 )  " "            " "                " "
2   ( 1 )  " "            " "                " "
3   ( 1 )  " "            " "                " "
4   ( 1 )  "*"            " "                " "
5   ( 1 )  "*"            " "                " "
6   ( 1 )  "*"            " "                " "
7   ( 1 )  "*"            " "                " "
8   ( 1 )  "*"            " "                " "
9   ( 1 )  "*"            " "                " "
10  ( 1 ) "*"             " "                " "
11  ( 1 ) "*"             " "                " "
12  ( 1 ) "*"             " "                " "
13  ( 1 ) "*"             " "                " "
14  ( 1 ) "*"             " "                " "
15  ( 1 ) "*"             " "                " "
16  ( 1 ) "*"             " "                " "
17  ( 1 ) "*"             " "                "*"
18  ( 1 ) "*"             "*"                "*"
19  ( 1 ) "*"             "*"                "*"
20  ( 1 ) "*"             "*"                "*"
21  ( 1 ) "*"             "*"                "*"
22  ( 1 ) "*"             "*"                "*"
23  ( 1 ) "*"             "*"                "*"
24  ( 1 ) "*"             "*"                "*"
25  ( 1 ) "*"             "*"                "*"
26  ( 1 ) "*"             "*"                "*"
27  ( 1 ) "*"             "*"                "*"
28  ( 1 ) "*"             "*"                "*"
          std_atomic_radius mean_Density wtd_entropy_Density wtd_range_D
ensity
1   ( 1 )  " "                " "          " "                 " "
2   ( 1 )  " "                " "          " "                 " "
3   ( 1 )  "*"                " "          " "                 " "
4   ( 1 )  "*"                " "          " "                 " "
5   ( 1 )  "*"                " "          " "                 " "
6   ( 1 )  "*"                " "          " "                 " "
7   ( 1 )  "*"                " "          " "                 " "
8   ( 1 )  "*"                " "          " "                 " "
9   ( 1 )  "*"                " "          " "                 " "
10  ( 1 ) "*"                 " "          " "                 " "
11  ( 1 ) "*"                 " "          " "                 " "
12  ( 1 ) "*"                 " "          "*"                 " "
13  ( 1 ) "*"                 " "          "*"                 " "
14  ( 1 ) "*"                 " "          "*"                 " "
15  ( 1 ) "*"                 " "          "*"                 " "
16  ( 1 ) "*"                 " "          "*"                 " "
17  ( 1 ) "*"                 " "          "*"                 " "
18  ( 1 ) "*"                 " "          "*"                 " "
```

```
19  ( 1 ) "*"                 " "                 "*"                 "*"
20  ( 1 ) "*"                 "*"                 "*"                 "*"
21  ( 1 ) "*"                 "*"                 "*"                 "*"
22  ( 1 ) "*"                 "*"                 "*"                 "*"
23  ( 1 ) "*"                 "*"                 "*"                 "*"
24  ( 1 ) "*"                 "*"                 "*"                 "*"
25  ( 1 ) "*"                 "*"                 "*"                 "*"
26  ( 1 ) "*"                 "*"                 "*"                 "*"
27  ( 1 ) "*"                 "*"                 "*"                 "*"
28  ( 1 ) "*"                 "*"                 "*"                 "*"
          wtd_std_Density gmean_ElectronAffinity wtd_gmean_ElectronAffin
ity
1   ( 1 )  " "              " "                   " "
2   ( 1 )  " "              "*"                   " "
3   ( 1 )  " "              "*"                   " "
4   ( 1 )  " "              "*"                   " "
5   ( 1 )  " "              "*"                   " "
6   ( 1 )  " "              "*"                   " "
7   ( 1 )  " "              "*"                   " "
8   ( 1 )  " "              "*"                   " "
9   ( 1 )  " "              "*"                   " "
10  ( 1 ) " "               "*"                   " "
11  ( 1 ) " "               "*"                   " "
12  ( 1 ) " "               "*"                   " "
13  ( 1 ) " "               "*"                   " "
14  ( 1 ) "*"               "*"                   " "
15  ( 1 ) "*"               "*"                   " "
16  ( 1 ) "*"               "*"                   " "
17  ( 1 ) "*"               "*"                   " "
18  ( 1 ) "*"               "*"                   " "
19  ( 1 ) "*"               "*"                   " "
20  ( 1 ) "*"               "*"                   " "
21  ( 1 ) "*"               "*"                   " "
22  ( 1 ) "*"               "*"                   " "
23  ( 1 ) "*"               "*"                   " "
24  ( 1 ) "*"               "*"                   " "
25  ( 1 ) "*"               "*"                   " "
26  ( 1 ) "*"               "*"                   " "
27  ( 1 ) "*"               "*"                   "*"
28  ( 1 ) "*"               "*"                   "*"
          wtd_entropy_ElectronAffinity wtd_range_ElectronAffinity
1   ( 1 )  " "                        " "
2   ( 1 )  " "                        " "
3   ( 1 )  " "                        " "
4   ( 1 )  " "                        " "
5   ( 1 )  " "                        " "
6   ( 1 )  " "                        " "
7   ( 1 )  "*"                        " "
8   ( 1 )  "*"                        " "
9   ( 1 )  "*"                        " "
10  ( 1 ) "*"                         " "
11  ( 1 ) "*"                         "*"
12  ( 1 ) "*"                         "*"
13  ( 1 ) "*"                         "*"
14  ( 1 ) "*"                         "*"
15  ( 1 ) "*"                         "*"
16  ( 1 ) "*"                         "*"
17  ( 1 ) "*"                         "*"
18  ( 1 ) "*"                         "*"
19  ( 1 ) "*"                         "*"
20  ( 1 ) "*"                         "*"
```

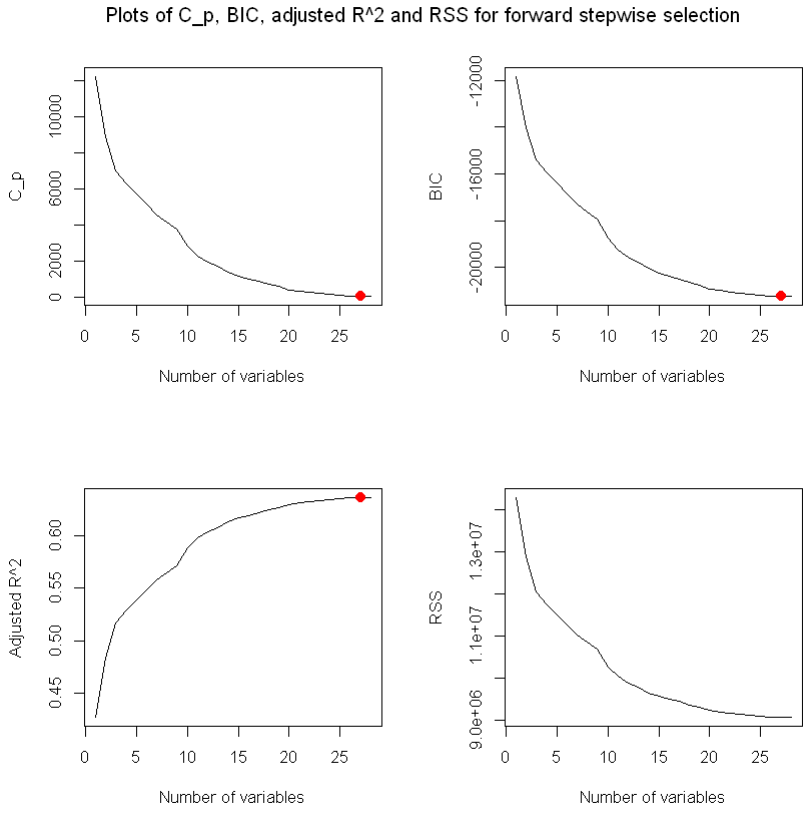
```
21  ( 1 ) "*"                          "*"
22  ( 1 ) "*"                          "*"
23  ( 1 ) "*"                          "*"
24  ( 1 ) "*"                          "*"
25  ( 1 ) "*"                          "*"
26  ( 1 ) "*"                          "*"
27  ( 1 ) "*"                          "*"
28  ( 1 ) "*"                          "*"
          std_ElectronAffinity mean_FusionHeat wtd_range_FusionHeat
1   ( 1 )  " "                 " "             " "
2   ( 1 )  " "                 " "             " "
3   ( 1 )  " "                 " "             " "
4   ( 1 )  " "                 " "             " "
5   ( 1 )  " "                 " "             " "
6   ( 1 )  " "                 " "             " "
7   ( 1 )  " "                 " "             " "
8   ( 1 )  " "                 " "             " "
9   ( 1 )  " "                 " "             " "
10  ( 1 ) " "                  " "             " "
11  ( 1 ) " "                  " "             " "
12  ( 1 ) " "                  " "             " "
13  ( 1 ) " "                  " "             " "
14  ( 1 ) " "                  " "             " "
15  ( 1 ) " "                  " "             " "
16  ( 1 ) "*"                  " "             " "
17  ( 1 ) "*"                  " "             " "
18  ( 1 ) "*"                  " "             " "
19  ( 1 ) "*"                  " "             " "
20  ( 1 ) "*"                  " "             " "
21  ( 1 ) "*"                  " "             " "
22  ( 1 ) "*"                  " "             " "
23  ( 1 ) "*"                  " "             " "
24  ( 1 ) "*"                  " "             " "
25  ( 1 ) "*"                  " "             " "
26  ( 1 ) "*"                  "*"             " "
27  ( 1 ) "*"                  "*"             " "
28  ( 1 ) "*"                  "*"             "*"
          std_FusionHeat mean_ThermalConductivity wtd_gmean_ThermalCondu
ctivity
1   ( 1 )  " "             " "                    " "
2   ( 1 )  " "             " "                    " "
3   ( 1 )  " "             " "                    " "
4   ( 1 )  " "             " "                    " "
5   ( 1 )  "*"             " "                    " "
6   ( 1 )  "*"             " "                    " "
7   ( 1 )  "*"             " "                    " "
8   ( 1 )  "*"             " "                    " "
9   ( 1 )  "*"             " "                    " "
10  ( 1 ) "*"              " "                    "*"
11  ( 1 ) "*"              " "                    "*"
12  ( 1 ) "*"              " "                    "*"
13  ( 1 ) "*"              " "                    "*"
14  ( 1 ) "*"              " "                    "*"
15  ( 1 ) "*"              "*"                    "*"
16  ( 1 ) "*"              "*"                    "*"
17  ( 1 ) "*"              "*"                    "*"
18  ( 1 ) "*"              "*"                    "*"
19  ( 1 ) "*"              "*"                    "*"
20  ( 1 ) "*"              "*"                    "*"
21  ( 1 ) "*"              "*"                    "*"
22  ( 1 ) "*"              "*"                    "*"
```

```
23  ( 1 ) "*"            "*"                  "*"
24  ( 1 ) "*"            "*"                  "*"
25  ( 1 ) "*"            "*"                  "*"
26  ( 1 ) "*"            "*"                  "*"
27  ( 1 ) "*"            "*"                  "*"
28  ( 1 ) "*"            "*"                  "*"
          entropy_ThermalConductivity wtd_entropy_ThermalConductivity
1   ( 1 )  " "                        " "
2   ( 1 )  " "                        " "
3   ( 1 )  " "                        " "
4   ( 1 )  " "                        " "
5   ( 1 )  " "                        " "
6   ( 1 )  "*"                        " "
7   ( 1 )  "*"                        " "
8   ( 1 )  "*"                        " "
9   ( 1 )  "*"                        " "
10  ( 1 ) "*"                         " "
11  ( 1 ) "*"                         " "
12  ( 1 ) "*"                         " "
13  ( 1 ) "*"                         " "
14  ( 1 ) "*"                         " "
15  ( 1 ) "*"                         " "
16  ( 1 ) "*"                         " "
17  ( 1 ) "*"                         " "
18  ( 1 ) "*"                         " "
19  ( 1 ) "*"                         " "
20  ( 1 ) "*"                         " "
21  ( 1 ) "*"                         "*"
22  ( 1 ) "*"                         "*"
23  ( 1 ) "*"                         "*"
24  ( 1 ) "*"                         "*"
25  ( 1 ) "*"                         "*"
26  ( 1 ) "*"                         "*"
27  ( 1 ) "*"                         "*"
28  ( 1 ) "*"                         "*"
          wtd_range_ThermalConductivity std_ThermalConductivity gmean_Va
lence
1   ( 1 )  " "                          "*"                     " "
2   ( 1 )  " "                          "*"                     " "
3   ( 1 )  " "                          "*"                     " "
4   ( 1 )  " "                          "*"                     " "
5   ( 1 )  " "                          "*"                     " "
6   ( 1 )  " "                          "*"                     " "
7   ( 1 )  " "                          "*"                     " "
8   ( 1 )  " "                          "*"                     " "
9   ( 1 )  "*"                          "*"                     " "
10  ( 1 ) "*"                           "*"                     " "
11  ( 1 ) "*"                           "*"                     " "
12  ( 1 ) "*"                           "*"                     " "
13  ( 1 ) "*"                           "*"                     " "
14  ( 1 ) "*"                           "*"                     " "
15  ( 1 ) "*"                           "*"                     " "
16  ( 1 ) "*"                           "*"                     " "
17  ( 1 ) "*"                           "*"                     " "
18  ( 1 ) "*"                           "*"                     " "
19  ( 1 ) "*"                           "*"                     " "
20  ( 1 ) "*"                           "*"                     " "
21  ( 1 ) "*"                           "*"                     " "
22  ( 1 ) "*"                           "*"                     " "
23  ( 1 ) "*"                           "*"                     "*"
24  ( 1 ) "*"                           "*"                     "*"
```

```
25  ( 1 ) "*"                                 "*"                                 "*"
26  ( 1 ) "*"                                 "*"                                 "*"
27  ( 1 ) "*"                                 "*"                                 "*"
28  ( 1 ) "*"                                 "*"                                 "*"
          wtd_range_Valence
1   ( 1 )  " "
2   ( 1 )  " "
3   ( 1 )  " "
4   ( 1 )  " "
5   ( 1 )  " "
6   ( 1 )  " "
7   ( 1 )  " "
8   ( 1 )  " "
9   ( 1 )  " "
10  ( 1 ) " "
11  ( 1 ) " "
12  ( 1 ) " "
13  ( 1 ) " "
14  ( 1 ) " "
15  ( 1 ) " "
16  ( 1 ) " "
17  ( 1 ) " "
18  ( 1 ) " "
19  ( 1 ) " "
20  ( 1 ) " "
21  ( 1 ) " "
22  ( 1 ) " "
23  ( 1 ) " "
24  ( 1 ) " "
25  ( 1 ) "*"
26  ( 1 ) "*"
27  ( 1 ) "*"
28  ( 1 ) "*"
```

In [91]:

```
par(mfrow = c(2, 2))
plot(reg.summary.fwd$cp, xlab = "Number of variables", ylab = "C_p", type = "l")
points(which.min(reg.summary.fwd$cp), reg.summary.fwd$cp[which.min(reg.summary.fwd$cp)], c
plot(reg.summary.fwd$bic, xlab = "Number of variables", ylab = "BIC", type = "l")
points(which.min(reg.summary.fwd$bic), reg.summary.fwd$bic[which.min(reg.summary.fwd$bic)]
plot(reg.summary.fwd$adjr2, xlab = "Number of variables", ylab = "Adjusted R^2", type = "l
points(which.max(reg.summary.fwd$adjr2), reg.summary.fwd$adjr2[which.max(reg.summary.fwd$a
       cex = 2, pch = 20)
plot(reg.summary.fwd$rss, xlab = "Number of variables", ylab = "RSS", type = "l")
mtext("Plots of C_p, BIC, adjusted R^2 and RSS for forward stepwise selection", side = 3,
```



Observations from the above plots:

- C_p is lowest for 28 variables.
- BIC is lowest for 27 variables.
- Adjusted R2 is maximum for 28 variables.
- RSS is lowest for 28 variables.

Examining the above plots, we can say that the number of predictors that will be best for our model is 28 variables.

In [92]:

```
# displaying the coefficients of the predictors using forward stepwise selection
coef(regfit.fwd, 28)
```

(Intercept): -94.2350224207943 **mean_atomic_mass:** 0.143807411042115
std_atomic_mass: 0.262468989633525 **gmean_fie:** 0.110729569941097 **wtd_gmean_fie:** -0.0381693050568154 **wtd_range_fie:** 0.0106981789269697 **mean_atomic_radius:** 0.277873444125613 **wtd_range_atomic_radius:** -0.151380406277973 **std_atomic_radius:** 0.332923466110136 **mean_Density:** -0.00328020517070794 **wtd_entropy_Density:** 6.84994688411662 **wtd_range_Density:** 0.00165887523934412 **wtd_std_Density:** -0.00290920716256944 **gmean_ElectronAffinity:** 0.0261571272206187
wtd_gmean_ElectronAffinity: -0.0746354571870936 **wtd_entropy_ElectronAffinity:** -38.6443655103925 **wtd_range_ElectronAffinity:** -0.238063428384714
std_ElectronAffinity: 0.105990273522442 **mean_FusionHeat:** 0.212241260386319
wtd_range_FusionHeat: -0.0229284401633827 **std_FusionHeat:** -0.555443818251619
mean_ThermalConductivity: 0.165846207152527 **wtd_gmean_ThermalConductivity:** -0.292305820498875 **entropy_ThermalConductivity:** 21.3580477936154
wtd_entropy_ThermalConductivity: 9.26486698266744 **wtd_range_ThermalConductivity:** 0.26351635875217 **std_ThermalConductivity:** 0.104598552407887 **gmean_Valence:** 2.73834890423946 **wtd_range_Valence:** -1.97589168368328

6.3 Backward Selection:

In backward selection type of feature selection method, it first begins with a model with all the features and then one by one, it keeps on subtracting variables until we get a single perfect model. It comprises of an absolute accuracy as it cannot add back a variable if the combination is not giving the best model. Although it is computationally cheaper(O(p^2)), it is unlikely to get the best subset as compared to the Best Subset Selection Method.

In [93]:

```
regfit.bwd <- regsubsets(critical_temp~., data = corTrainingData, nvmax = 30, method="backı
reg.summary.bwd <- summary(regfit.bwd)
reg.summary.bwd
```

```
Subset selection object
Call: regsubsets.formula(critical_temp ~ ., data = corTrainingData,
    nvmax = 30, method = "backward")
28 Variables  (and intercept)
                               Forced in Forced out
mean_atomic_mass                   FALSE      FALSE
std_atomic_mass                    FALSE      FALSE
gmean_fie                          FALSE      FALSE
wtd_gmean_fie                      FALSE      FALSE
wtd_range_fie                      FALSE      FALSE
mean_atomic_radius                 FALSE      FALSE
wtd_range_atomic_radius            FALSE      FALSE
std_atomic_radius                  FALSE      FALSE
mean_Density                       FALSE      FALSE
wtd_entropy_Density                FALSE      FALSE
wtd_range_Density                  FALSE      FALSE
wtd_std_Density                    FALSE      FALSE
gmean_ElectronAffinity             FALSE      FALSE
wtd_gmean_ElectronAffinity         FALSE      FALSE
wtd_entropy_ElectronAffinity       FALSE      FALSE
wtd_range_ElectronAffinity         FALSE      FALSE
std_ElectronAffinity               FALSE      FALSE
mean_FusionHeat                    FALSE      FALSE
wtd_range_FusionHeat               FALSE      FALSE
std_FusionHeat                     FALSE      FALSE
mean_ThermalConductivity           FALSE      FALSE
wtd_gmean_ThermalConductivity      FALSE      FALSE
entropy_ThermalConductivity        FALSE      FALSE
wtd_entropy_ThermalConductivity    FALSE      FALSE
wtd_range_ThermalConductivity      FALSE      FALSE
std_ThermalConductivity            FALSE      FALSE
gmean_Valence                      FALSE      FALSE
wtd_range_Valence                  FALSE      FALSE
1 subsets of each size up to 28
Selection Algorithm: backward
          mean_atomic_mass std_atomic_mass gmean_fie wtd_gmean_fie
1  ( 1 )  " "              " "             " "       " "
2  ( 1 )  " "              " "             " "       " "
3  ( 1 )  " "              " "             " "       " "
4  ( 1 )  " "              " "             " "       " "
5  ( 1 )  " "              " "             " "       " "
6  ( 1 )  " "              " "             " "       " "
7  ( 1 )  " "              " "             " "       " "
8  ( 1 )  " "              " "             " "       " "
9  ( 1 )  " "              " "             " "       " "
10  ( 1 ) " "              " "             " "       " "
11  ( 1 ) " "              "*"             " "       " "
12  ( 1 ) " "              "*"             " "       " "
13  ( 1 ) " "              "*"             "*"       " "
14  ( 1 ) " "              "*"             "*"       " "
15  ( 1 ) " "              "*"             "*"       " "
16  ( 1 ) " "              "*"             "*"       " "
17  ( 1 ) " "              "*"             "*"       " "
```

```
18  ( 1 ) " "               "*"               "*"         " "
19  ( 1 ) "*"               "*"               "*"         " "
20  ( 1 ) "*"               "*"               "*"         " "
21  ( 1 ) "*"               "*"               "*"         " "
22  ( 1 ) "*"               "*"               "*"         " "
23  ( 1 ) "*"               "*"               "*"         "*"
24  ( 1 ) "*"               "*"               "*"         "*"
25  ( 1 ) "*"               "*"               "*"         "*"
26  ( 1 ) "*"               "*"               "*"         "*"
27  ( 1 ) "*"               "*"               "*"         "*"
28  ( 1 ) "*"               "*"               "*"         "*"
          wtd_range_fie mean_atomic_radius wtd_range_atomic_radius
1   ( 1 )  " "           " "                " "
2   ( 1 )  " "           " "                " "
3   ( 1 )  " "           " "                " "
4   ( 1 )  " "           " "                " "
5   ( 1 )  " "           " "                " "
6   ( 1 )  " "           " "                " "
7   ( 1 )  " "           " "                " "
8   ( 1 )  " "           " "                " "
9   ( 1 )  " "           " "                " "
10  ( 1 ) " "            " "                "*"
11  ( 1 ) " "            " "                "*"
12  ( 1 ) " "            " "                "*"
13  ( 1 ) " "            " "                "*"
14  ( 1 ) " "            "*"                "*"
15  ( 1 ) " "            "*"                "*"
16  ( 1 ) " "            "*"                "*"
17  ( 1 ) " "            "*"                "*"
18  ( 1 ) " "            "*"                "*"
19  ( 1 ) " "            "*"                "*"
20  ( 1 ) " "            "*"                "*"
21  ( 1 ) " "            "*"                "*"
22  ( 1 ) " "            "*"                "*"
23  ( 1 ) " "            "*"                "*"
24  ( 1 ) " "            "*"                "*"
25  ( 1 ) " "            "*"                "*"
26  ( 1 ) "*"            "*"                "*"
27  ( 1 ) "*"            "*"                "*"
28  ( 1 ) "*"            "*"                "*"
          std_atomic_radius mean_Density wtd_entropy_Density wtd_range_Densi
ty
1   ( 1 )  " "                " "          " "                " "
2   ( 1 )  " "                " "          " "                " "
3   ( 1 )  " "                " "          " "                " "
4   ( 1 )  " "                " "          " "                " "
5   ( 1 )  " "                " "          " "                " "
6   ( 1 )  " "                " "          " "                " "
7   ( 1 )  "*"                " "          " "                " "
8   ( 1 )  "*"                " "          " "                " "
9   ( 1 )  "*"                " "          " "                " "
10  ( 1 ) "*"                 " "          " "                " "
11  ( 1 ) "*"                 " "          " "                " "
12  ( 1 ) "*"                 " "          " "                " "
13  ( 1 ) "*"                 " "          " "                " "
14  ( 1 ) "*"                 " "          " "                " "
15  ( 1 ) "*"                 " "          " "                " "
16  ( 1 ) "*"                 " "          " "                "*"
17  ( 1 ) "*"                 "*"          " "                "*"
18  ( 1 ) "*"                 "*"          " "                "*"
19  ( 1 ) "*"                 "*"          " "                "*"
```

```
20  ( 1 ) "*"                 "*"             " "                     "*"
21  ( 1 ) "*"                 "*"             " "                     "*"
22  ( 1 ) "*"                 "*"             " "                     "*"
23  ( 1 ) "*"                 "*"             " "                     "*"
24  ( 1 ) "*"                 "*"             "*"                     "*"
25  ( 1 ) "*"                 "*"             "*"                     "*"
26  ( 1 ) "*"                 "*"             "*"                     "*"
27  ( 1 ) "*"                 "*"             "*"                     "*"
28  ( 1 ) "*"                 "*"             "*"                     "*"
          wtd_std_Density gmean_ElectronAffinity wtd_gmean_ElectronAffinity
1   ( 1 )  " "              " "                  " "
2   ( 1 )  " "              " "                  " "
3   ( 1 )  " "              " "                  " "
4   ( 1 )  " "              " "                  " "
5   ( 1 )  " "              " "                  " "
6   ( 1 )  " "              " "                  " "
7   ( 1 )  " "              " "                  " "
8   ( 1 )  " "              " "                  " "
9   ( 1 )  " "              " "                  " "
10  ( 1 ) " "               " "                  " "
11  ( 1 ) " "               " "                  " "
12  ( 1 ) "*"               " "                  " "
13  ( 1 ) "*"               " "                  " "
14  ( 1 ) "*"               " "                  " "
15  ( 1 ) "*"               " "                  " "
16  ( 1 ) "*"               " "                  " "
17  ( 1 ) "*"               " "                  " "
18  ( 1 ) "*"               " "                  " "
19  ( 1 ) "*"               " "                  " "
20  ( 1 ) "*"               " "                  " "
21  ( 1 ) "*"               " "                  "*"
22  ( 1 ) "*"               " "                  "*"
23  ( 1 ) "*"               " "                  "*"
24  ( 1 ) "*"               " "                  "*"
25  ( 1 ) "*"               " "                  "*"
26  ( 1 ) "*"               " "                  "*"
27  ( 1 ) "*"               "*"                  "*"
28  ( 1 ) "*"               "*"                  "*"
          wtd_entropy_ElectronAffinity wtd_range_ElectronAffinity
1   ( 1 )  " "                         " "
2   ( 1 )  " "                         " "
3   ( 1 )  " "                         " "
4   ( 1 )  " "                         " "
5   ( 1 )  " "                         "*"
6   ( 1 )  "*"                         "*"
7   ( 1 )  "*"                         "*"
8   ( 1 )  "*"                         "*"
9   ( 1 )  "*"                         "*"
10  ( 1 ) "*"                          "*"
11  ( 1 ) "*"                          "*"
12  ( 1 ) "*"                          "*"
13  ( 1 ) "*"                          "*"
14  ( 1 ) "*"                          "*"
15  ( 1 ) "*"                          "*"
16  ( 1 ) "*"                          "*"
17  ( 1 ) "*"                          "*"
18  ( 1 ) "*"                          "*"
19  ( 1 ) "*"                          "*"
20  ( 1 ) "*"                          "*"
21  ( 1 ) "*"                          "*"
22  ( 1 ) "*"                          "*"
```

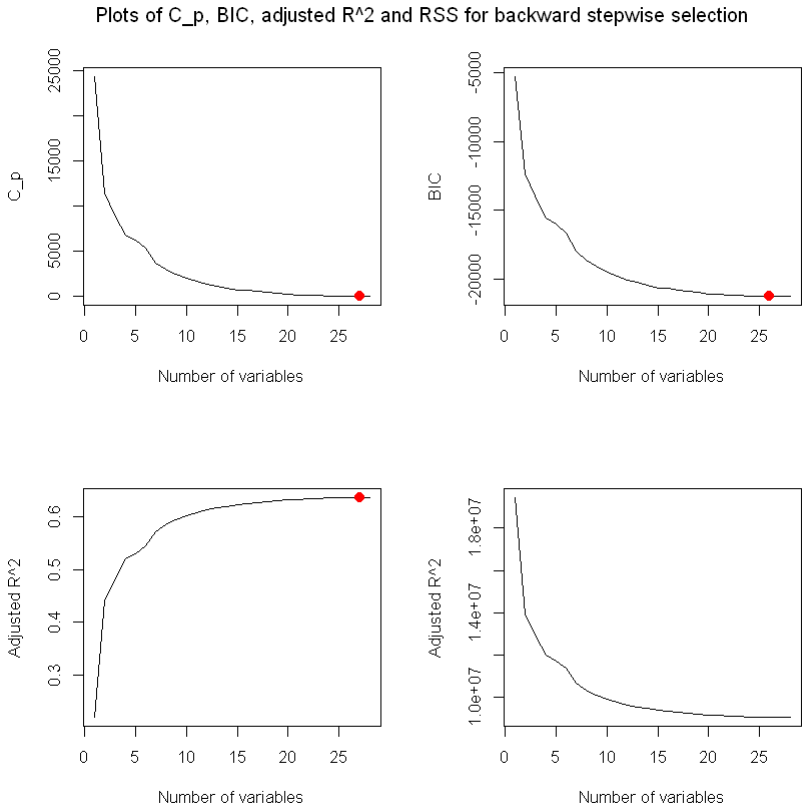
```
23  ( 1 ) "*"                          "*"
24  ( 1 ) "*"                          "*"
25  ( 1 ) "*"                          "*"
26  ( 1 ) "*"                          "*"
27  ( 1 ) "*"                          "*"
28  ( 1 ) "*"                          "*"
          std_ElectronAffinity mean_FusionHeat wtd_range_FusionHeat
1   ( 1 )  " "                  " "            " "
2   ( 1 )  " "                  " "            " "
3   ( 1 )  " "                  " "            " "
4   ( 1 )  " "                  " "            " "
5   ( 1 )  " "                  " "            " "
6   ( 1 )  " "                  " "            " "
7   ( 1 )  " "                  " "            " "
8   ( 1 )  "*"                  " "            " "
9   ( 1 )  "*"                  " "            " "
10  ( 1 ) "*"                   " "            " "
11  ( 1 ) "*"                   " "            " "
12  ( 1 ) "*"                   " "            " "
13  ( 1 ) "*"                   " "            " "
14  ( 1 ) "*"                   " "            " "
15  ( 1 ) "*"                   " "            " "
16  ( 1 ) "*"                   " "            " "
17  ( 1 ) "*"                   " "            " "
18  ( 1 ) "*"                   " "            " "
19  ( 1 ) "*"                   " "            " "
20  ( 1 ) "*"                   " "            " "
21  ( 1 ) "*"                   " "            " "
22  ( 1 ) "*"                   "*"            " "
23  ( 1 ) "*"                   "*"            " "
24  ( 1 ) "*"                   "*"            " "
25  ( 1 ) "*"                   "*"            " "
26  ( 1 ) "*"                   "*"            " "
27  ( 1 ) "*"                   "*"            " "
28  ( 1 ) "*"                   "*"            "*"
          std_FusionHeat mean_ThermalConductivity wtd_gmean_ThermalConductiv
ity
1   ( 1 )  " "            " "                      " "
2   ( 1 )  " "            " "                      "*"
3   ( 1 )  " "            " "                      "*"
4   ( 1 )  " "            " "                      "*"
5   ( 1 )  " "            " "                      "*"
6   ( 1 )  " "            " "                      "*"
7   ( 1 )  " "            " "                      "*"
8   ( 1 )  " "            " "                      "*"
9   ( 1 )  "*"            " "                      "*"
10  ( 1 ) "*"             " "                      "*"
11  ( 1 ) "*"             " "                      "*"
12  ( 1 ) "*"             " "                      "*"
13  ( 1 ) "*"             " "                      "*"
14  ( 1 ) "*"             " "                      "*"
15  ( 1 ) "*"             " "                      "*"
16  ( 1 ) "*"             " "                      "*"
17  ( 1 ) "*"             " "                      "*"
18  ( 1 ) "*"             " "                      "*"
19  ( 1 ) "*"             " "                      "*"
20  ( 1 ) "*"             "*"                      "*"
21  ( 1 ) "*"             "*"                      "*"
22  ( 1 ) "*"             "*"                      "*"
23  ( 1 ) "*"             "*"                      "*"
24  ( 1 ) "*"             "*"                      "*"
```

```
25  ( 1 ) "*"            "*"                      "*"
26  ( 1 ) "*"            "*"                      "*"
27  ( 1 ) "*"            "*"                      "*"
28  ( 1 ) "*"            "*"                      "*"
          entropy_ThermalConductivity wtd_entropy_ThermalConductivity
1   ( 1 )  " "                        " "
2   ( 1 )  " "                        " "
3   ( 1 )  " "                        " "
4   ( 1 )  "*"                        " "
5   ( 1 )  "*"                        " "
6   ( 1 )  "*"                        " "
7   ( 1 )  "*"                        " "
8   ( 1 )  "*"                        " "
9   ( 1 )  "*"                        " "
10  ( 1 ) "*"                         " "
11  ( 1 ) "*"                         " "
12  ( 1 ) "*"                         " "
13  ( 1 ) "*"                         " "
14  ( 1 ) "*"                         " "
15  ( 1 ) "*"                         "*"
16  ( 1 ) "*"                         "*"
17  ( 1 ) "*"                         "*"
18  ( 1 ) "*"                         "*"
19  ( 1 ) "*"                         "*"
20  ( 1 ) "*"                         "*"
21  ( 1 ) "*"                         "*"
22  ( 1 ) "*"                         "*"
23  ( 1 ) "*"                         "*"
24  ( 1 ) "*"                         "*"
25  ( 1 ) "*"                         "*"
26  ( 1 ) "*"                         "*"
27  ( 1 ) "*"                         "*"
28  ( 1 ) "*"                         "*"
          wtd_range_ThermalConductivity std_ThermalConductivity gmean_Valenc
e
1   ( 1 )  "*"                          " "                     " "
2   ( 1 )  "*"                          " "                     " "
3   ( 1 )  "*"                          "*"                     " "
4   ( 1 )  "*"                          "*"                     " "
5   ( 1 )  "*"                          "*"                     " "
6   ( 1 )  "*"                          "*"                     " "
7   ( 1 )  "*"                          "*"                     " "
8   ( 1 )  "*"                          "*"                     " "
9   ( 1 )  "*"                          "*"                     " "
10  ( 1 ) "*"                           "*"                     " "
11  ( 1 ) "*"                           "*"                     " "
12  ( 1 ) "*"                           "*"                     " "
13  ( 1 ) "*"                           "*"                     " "
14  ( 1 ) "*"                           "*"                     " "
15  ( 1 ) "*"                           "*"                     " "
16  ( 1 ) "*"                           "*"                     " "
17  ( 1 ) "*"                           "*"                     " "
18  ( 1 ) "*"                           "*"                     "*"
19  ( 1 ) "*"                           "*"                     "*"
20  ( 1 ) "*"                           "*"                     "*"
21  ( 1 ) "*"                           "*"                     "*"
22  ( 1 ) "*"                           "*"                     "*"
23  ( 1 ) "*"                           "*"                     "*"
24  ( 1 ) "*"                           "*"                     "*"
25  ( 1 ) "*"                           "*"                     "*"
26  ( 1 ) "*"                           "*"                     "*"
```

```
27  ( 1 ) "*"                                 "*"                                 "*"
28  ( 1 ) "*"                                 "*"                                 "*"
          wtd_range_Valence
1   ( 1 )  " "
2   ( 1 )  " "
3   ( 1 )  " "
4   ( 1 )  " "
5   ( 1 )  " "
6   ( 1 )  " "
7   ( 1 )  " "
8   ( 1 )  " "
9   ( 1 )  " "
10  ( 1 ) " "
11  ( 1 ) " "
12  ( 1 ) " "
13  ( 1 ) " "
14  ( 1 ) " "
15  ( 1 ) " "
16  ( 1 ) " "
17  ( 1 ) " "
18  ( 1 ) " "
19  ( 1 ) " "
20  ( 1 ) " "
21  ( 1 ) " "
22  ( 1 ) " "
23  ( 1 ) " "
24  ( 1 ) " "
25  ( 1 ) "*"
26  ( 1 ) "*"
27  ( 1 ) "*"
28  ( 1 ) "*"
```

In [94]:

```
par(mfrow = c(2, 2))
plot(reg.summary.bwd$cp, xlab = "Number of variables", ylab = "C_p", type = "l")
points(which.min(reg.summary.bwd$cp), reg.summary.bwd$cp[which.min(reg.summary.bwd$cp)], c
plot(reg.summary.bwd$bic, xlab = "Number of variables", ylab = "BIC", type = "l")
points(which.min(reg.summary.bwd$bic), reg.summary.bwd$bic[which.min(reg.summary.bwd$bic)]
plot(reg.summary.bwd$adjr2, xlab = "Number of variables", ylab = "Adjusted R^2", type = "l
points(which.max(reg.summary.bwd$adjr2), reg.summary.bwd$adjr2[which.max(reg.summary.bwd$a
       cex = 2, pch = 20)
plot(reg.summary.bwd$rss, xlab = "Number of variables", ylab = "Adjusted R^2", type = "l")
mtext("Plots of C_p, BIC, adjusted R^2 and RSS for backward stepwise selection", side = 3,
```



Observations from the above plots:

- C_p is lowest for 28 variables.
- BIC is lowest for 27 variables.
- Adjusted R2 is maximum for 28 variables.
- RSS is lowest for 28 variables.

Examining the above plots, we can say that the number of predictors that will be best for our model is 28 variables.

In [95]:

```
coef(regfit.bwd, 28)
```

(Intercept): -94.2350224207944 **mean_atomic_mass:** 0.143807411042115
std_atomic_mass: 0.262468989633525 **gmean_fie:** 0.110729569941097 **wtd_gmean_fie:** -0.0381693050568154 **wtd_range_fie:** 0.0106981789269697 **mean_atomic_radius:** 0.277873444125613 **wtd_range_atomic_radius:** -0.151380406277973 **std_atomic_radius:** 0.332923466110136 **mean_Density:** -0.00328020517070794 **wtd_entropy_Density:** 6.8499468841166 **wtd_range_Density:** 0.00165887523934412 **wtd_std_Density:** -0.00290920716256944 **gmean_ElectronAffinity:** 0.0261571272206187
wtd_gmean_ElectronAffinity: -0.0746354571870937 **wtd_entropy_ElectronAffinity:** -38.6443655103925 **wtd_range_ElectronAffinity:** -0.238063428384714
std_ElectronAffinity: 0.105990273522442 **mean_FusionHeat:** 0.212241260386319
wtd_range_FusionHeat: -0.0229284401633826 **std_FusionHeat:** -0.555443818251619
mean_ThermalConductivity: 0.165846207152527 **wtd_gmean_ThermalConductivity:** -0.292305820498875 **entropy_ThermalConductivity:** 21.3580477936154
wtd_entropy_ThermalConductivity: 9.26486698266745 **wtd_range_ThermalConductivity:** 0.26351635875217 **std_ThermalConductivity:** 0.104598552407887 **gmean_Valence:** 2.73834890423945 **wtd_range_Valence:** -1.97589168368328

6.4 Hybrid Selection:

In hybrid selection type of feature selection method, unlike forward and backward, it can pick or drop a variable at the same time until we get a single perfect model. It is not the best stepwise selection method but it can closely replicate the Best Subset Selection Method. Also it is computationally cheaper(O(p^2)).

In [96]:

```
regfit.both <- regsubsets(critical_temp~., data = corTrainingData, nvmax = 30, method="seq
reg.summary.both <- summary(regfit.both)
reg.summary.both
```

```
Subset selection object
Call: regsubsets.formula(critical_temp ~ ., data = corTrainingData,
    nvmax = 30, method = "seqrep")
28 Variables  (and intercept)
                               Forced in Forced out
mean_atomic_mass                   FALSE      FALSE
std_atomic_mass                    FALSE      FALSE
gmean_fie                          FALSE      FALSE
wtd_gmean_fie                      FALSE      FALSE
wtd_range_fie                      FALSE      FALSE
mean_atomic_radius                 FALSE      FALSE
wtd_range_atomic_radius            FALSE      FALSE
std_atomic_radius                  FALSE      FALSE
mean_Density                       FALSE      FALSE
wtd_entropy_Density                FALSE      FALSE
wtd_range_Density                  FALSE      FALSE
wtd_std_Density                    FALSE      FALSE
gmean_ElectronAffinity             FALSE      FALSE
wtd_gmean_ElectronAffinity         FALSE      FALSE
wtd_entropy_ElectronAffinity       FALSE      FALSE
wtd_range_ElectronAffinity         FALSE      FALSE
std_ElectronAffinity               FALSE      FALSE
mean_FusionHeat                    FALSE      FALSE
wtd_range_FusionHeat               FALSE      FALSE
std_FusionHeat                     FALSE      FALSE
mean_ThermalConductivity           FALSE      FALSE
wtd_gmean_ThermalConductivity      FALSE      FALSE
entropy_ThermalConductivity        FALSE      FALSE
wtd_entropy_ThermalConductivity    FALSE      FALSE
wtd_range_ThermalConductivity      FALSE      FALSE
std_ThermalConductivity            FALSE      FALSE
gmean_Valence                      FALSE      FALSE
wtd_range_Valence                  FALSE      FALSE
1 subsets of each size up to 28
Selection Algorithm: 'sequential replacement'
          mean_atomic_mass std_atomic_mass gmean_fie wtd_gmean_fie
1  ( 1 )  " "              " "             " "       " "
2  ( 1 )  " "              " "             " "       " "
3  ( 1 )  " "              " "             " "       " "
4  ( 1 )  " "              " "             " "       " "
5  ( 1 )  " "              " "             " "       " "
6  ( 1 )  " "              " "             " "       " "
7  ( 1 )  "*"              "*"             "*"       "*"
8  ( 1 )  " "              " "             " "       " "
9  ( 1 )  "*"              "*"             "*"       "*"
10  ( 1 ) " "              "*"             " "       " "
11  ( 1 ) "*"              "*"             "*"       "*"
12  ( 1 ) " "              "*"             " "       " "
13  ( 1 ) " "              "*"             "*"       " "
14  ( 1 ) " "              "*"             "*"       "*"
15  ( 1 ) "*"              "*"             "*"       "*"
16  ( 1 ) " "              "*"             "*"       " "
17  ( 1 ) " "              "*"             "*"       "*"
```

```
18  ( 1 ) " "               "*"               "*"         " "
19  ( 1 ) "*"               "*"               "*"         " "
20  ( 1 ) "*"               "*"               "*"         " "
21  ( 1 ) "*"               "*"               "*"         " "
22  ( 1 ) "*"               "*"               "*"         "*"
23  ( 1 ) "*"               "*"               "*"         "*"
24  ( 1 ) "*"               "*"               "*"         "*"
25  ( 1 ) "*"               "*"               "*"         "*"
26  ( 1 ) "*"               "*"               "*"         "*"
27  ( 1 ) "*"               "*"               "*"         "*"
28  ( 1 ) "*"               "*"               "*"         "*"
          wtd_range_fie mean_atomic_radius wtd_range_atomic_radius
1   ( 1 )  " "           " "               " "
2   ( 1 )  " "           " "               " "
3   ( 1 )  " "           " "               " "
4   ( 1 )  " "           " "               " "
5   ( 1 )  " "           " "               " "
6   ( 1 )  " "           " "               "*"
7   ( 1 )  "*"           "*"               "*"
8   ( 1 )  " "           " "               "*"
9   ( 1 )  "*"           "*"               "*"
10  ( 1 ) " "            " "               "*"
11  ( 1 ) "*"            "*"               "*"
12  ( 1 ) " "            " "               "*"
13  ( 1 ) " "            "*"               "*"
14  ( 1 ) " "            "*"               "*"
15  ( 1 ) "*"            "*"               "*"
16  ( 1 ) " "            "*"               "*"
17  ( 1 ) " "            "*"               "*"
18  ( 1 ) " "            "*"               "*"
19  ( 1 ) " "            "*"               "*"
20  ( 1 ) " "            "*"               "*"
21  ( 1 ) " "            "*"               "*"
22  ( 1 ) " "            "*"               "*"
23  ( 1 ) " "            "*"               "*"
24  ( 1 ) "*"            "*"               "*"
25  ( 1 ) " "            "*"               "*"
26  ( 1 ) "*"            "*"               "*"
27  ( 1 ) "*"            "*"               "*"
28  ( 1 ) "*"            "*"               "*"
          std_atomic_radius mean_Density wtd_entropy_Density wtd_range_Densi
ty
1   ( 1 )  " "               " "          " "                " "
2   ( 1 )  " "               " "          " "                " "
3   ( 1 )  "*"               " "          " "                " "
4   ( 1 )  "*"               " "          " "                " "
5   ( 1 )  "*"               " "          " "                " "
6   ( 1 )  "*"               " "          " "                " "
7   ( 1 )  " "               " "          " "                " "
8   ( 1 )  "*"               " "          " "                " "
9   ( 1 )  "*"               "*"          " "                " "
10  ( 1 ) "*"                " "          " "                " "
11  ( 1 ) "*"                "*"          "*"                "*"
12  ( 1 ) "*"                " "          " "                " "
13  ( 1 ) "*"                " "          " "                " "
14  ( 1 ) "*"                " "          " "                " "
15  ( 1 ) "*"                "*"          "*"                "*"
16  ( 1 ) "*"                " "          " "                " "
17  ( 1 ) "*"                "*"          " "                "*"
18  ( 1 ) "*"                "*"          "*"                "*"
19  ( 1 ) "*"                "*"          "*"                "*"
```

```
20  ( 1 ) "*"                 "*"             "*"                     "*"
21  ( 1 ) "*"                 "*"             "*"                     "*"
22  ( 1 ) "*"                 "*"             "*"                     "*"
23  ( 1 ) "*"                 "*"             " "                     "*"
24  ( 1 ) "*"                 "*"             "*"                     "*"
25  ( 1 ) "*"                 "*"             "*"                     "*"
26  ( 1 ) "*"                 "*"             "*"                     "*"
27  ( 1 ) "*"                 "*"             "*"                     "*"
28  ( 1 ) "*"                 "*"             "*"                     "*"
          wtd_std_Density gmean_ElectronAffinity wtd_gmean_ElectronAffinity
1   ( 1 )  " "                " "                    " "
2   ( 1 )  " "                "*"                    " "
3   ( 1 )  " "                " "                    "*"
4   ( 1 )  " "                " "                    "*"
5   ( 1 )  " "                " "                    "*"
6   ( 1 )  " "                " "                    "*"
7   ( 1 )  " "                " "                    " "
8   ( 1 )  " "                " "                    "*"
9   ( 1 )  " "                " "                    " "
10  ( 1 ) "*"                 " "                    "*"
11  ( 1 ) " "                 " "                    " "
12  ( 1 ) "*"                 " "                    "*"
13  ( 1 ) "*"                 " "                    "*"
14  ( 1 ) "*"                 " "                    "*"
15  ( 1 ) "*"                 "*"                    "*"
16  ( 1 ) "*"                 " "                    " "
17  ( 1 ) "*"                 " "                    " "
18  ( 1 ) "*"                 " "                    " "
19  ( 1 ) "*"                 " "                    " "
20  ( 1 ) "*"                 " "                    " "
21  ( 1 ) "*"                 " "                    " "
22  ( 1 ) "*"                 " "                    " "
23  ( 1 ) "*"                 " "                    "*"
24  ( 1 ) "*"                 "*"                    "*"
25  ( 1 ) "*"                 " "                    "*"
26  ( 1 ) "*"                 " "                    "*"
27  ( 1 ) "*"                 "*"                    "*"
28  ( 1 ) "*"                 "*"                    "*"
          wtd_entropy_ElectronAffinity wtd_range_ElectronAffinity
1   ( 1 )  " "                        " "
2   ( 1 )  " "                        " "
3   ( 1 )  " "                        " "
4   ( 1 )  " "                        " "
5   ( 1 )  " "                        " "
6   ( 1 )  " "                        " "
7   ( 1 )  " "                        " "
8   ( 1 )  " "                        " "
9   ( 1 )  " "                        " "
10  ( 1 ) " "                         " "
11  ( 1 ) " "                         " "
12  ( 1 ) "*"                         " "
13  ( 1 ) "*"                         " "
14  ( 1 ) "*"                         " "
15  ( 1 ) "*"                         " "
16  ( 1 ) "*"                         "*"
17  ( 1 ) "*"                         "*"
18  ( 1 ) "*"                         "*"
19  ( 1 ) "*"                         "*"
20  ( 1 ) "*"                         "*"
21  ( 1 ) "*"                         "*"
22  ( 1 ) "*"                         "*"
```

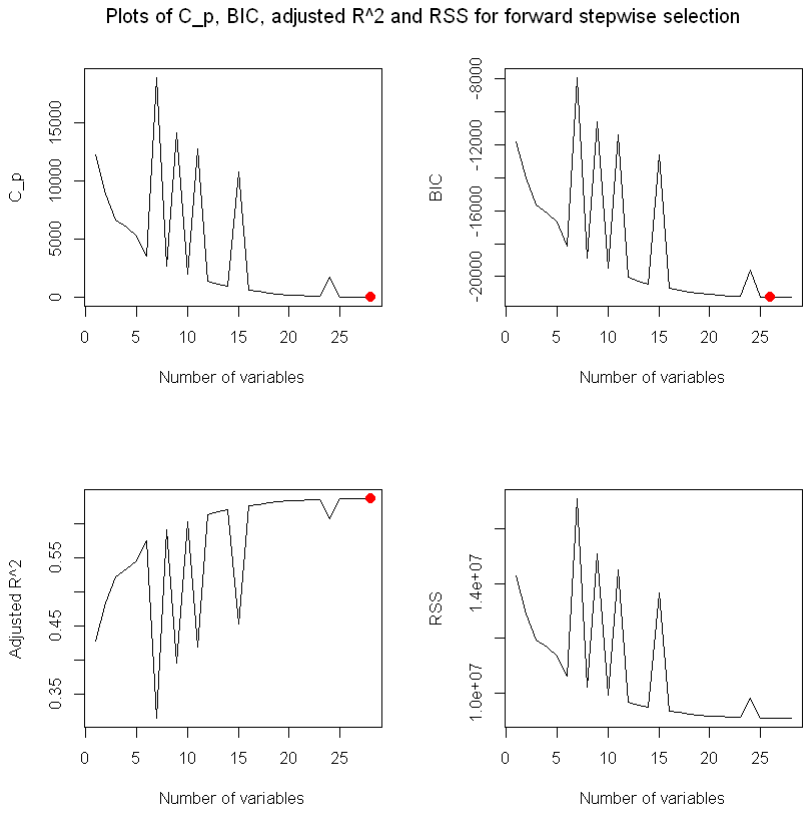
```
23  ( 1 ) "*"                          "*"
24  ( 1 ) "*"                          "*"
25  ( 1 ) "*"                          "*"
26  ( 1 ) "*"                          "*"
27  ( 1 ) "*"                          "*"
28  ( 1 ) "*"                          "*"
          std_ElectronAffinity mean_FusionHeat wtd_range_FusionHeat
1   ( 1 )  " "                  " "           " "
2   ( 1 )  " "                  " "           " "
3   ( 1 )  " "                  " "           " "
4   ( 1 )  " "                  " "           " "
5   ( 1 )  " "                  " "           " "
6   ( 1 )  " "                  " "           " "
7   ( 1 )  " "                  " "           " "
8   ( 1 )  " "                  " "           " "
9   ( 1 )  " "                  " "           " "
10  ( 1 ) " "                   " "           " "
11  ( 1 ) " "                   " "           " "
12  ( 1 ) " "                   "*"           " "
13  ( 1 ) " "                   " "           " "
14  ( 1 ) " "                   " "           " "
15  ( 1 ) " "                   " "           " "
16  ( 1 ) "*"                   " "           " "
17  ( 1 ) "*"                   " "           " "
18  ( 1 ) "*"                   " "           " "
19  ( 1 ) "*"                   " "           " "
20  ( 1 ) "*"                   "*"           " "
21  ( 1 ) "*"                   " "           " "
22  ( 1 ) "*"                   " "           " "
23  ( 1 ) "*"                   "*"           " "
24  ( 1 ) "*"                   "*"           "*"
25  ( 1 ) "*"                   "*"           " "
26  ( 1 ) "*"                   "*"           " "
27  ( 1 ) "*"                   "*"           "*"
28  ( 1 ) "*"                   "*"           "*"
          std_FusionHeat mean_ThermalConductivity wtd_gmean_ThermalConductiv
ity
1   ( 1 )  " "            " "                     " "
2   ( 1 )  " "            " "                     " "
3   ( 1 )  " "            " "                     " "
4   ( 1 )  "*"            " "                     " "
5   ( 1 )  "*"            " "                     " "
6   ( 1 )  " "            "*"                     "*"
7   ( 1 )  " "            " "                     " "
8   ( 1 )  "*"            "*"                     "*"
9   ( 1 )  " "            " "                     " "
10  ( 1 ) "*"             "*"                     "*"
11  ( 1 ) " "             " "                     " "
12  ( 1 ) "*"             "*"                     "*"
13  ( 1 ) "*"             "*"                     "*"
14  ( 1 ) "*"             "*"                     "*"
15  ( 1 ) " "             " "                     " "
16  ( 1 ) "*"             "*"                     "*"
17  ( 1 ) "*"             " "                     "*"
18  ( 1 ) "*"             "*"                     "*"
19  ( 1 ) "*"             "*"                     "*"
20  ( 1 ) "*"             "*"                     "*"
21  ( 1 ) "*"             "*"                     "*"
22  ( 1 ) "*"             "*"                     "*"
23  ( 1 ) "*"             "*"                     "*"
24  ( 1 ) "*"             "*"                     "*"
```

```
25  ( 1 ) "*"            "*"                       "*"
26  ( 1 ) "*"            "*"                       "*"
27  ( 1 ) "*"            "*"                       "*"
28  ( 1 ) "*"            "*"                       "*"
          entropy_ThermalConductivity wtd_entropy_ThermalConductivity
1   ( 1 )  " "                        " "
2   ( 1 )  " "                        " "
3   ( 1 )  " "                        " "
4   ( 1 )  " "                        " "
5   ( 1 )  " "                        " "
6   ( 1 )  " "                        " "
7   ( 1 )  " "                        " "
8   ( 1 )  "*"                        " "
9   ( 1 )  " "                        " "
10  ( 1 ) "*"                         " "
11  ( 1 ) " "                         " "
12  ( 1 ) "*"                         " "
13  ( 1 ) "*"                         " "
14  ( 1 ) "*"                         " "
15  ( 1 ) " "                         " "
16  ( 1 ) "*"                         "*"
17  ( 1 ) "*"                         " "
18  ( 1 ) "*"                         "*"
19  ( 1 ) "*"                         "*"
20  ( 1 ) "*"                         "*"
21  ( 1 ) "*"                         "*"
22  ( 1 ) "*"                         "*"
23  ( 1 ) "*"                         "*"
24  ( 1 ) "*"                         "*"
25  ( 1 ) "*"                         "*"
26  ( 1 ) "*"                         "*"
27  ( 1 ) "*"                         "*"
28  ( 1 ) "*"                         "*"
          wtd_range_ThermalConductivity std_ThermalConductivity gmean_Valenc
e
1   ( 1 )  " "                          "*"                     " "
2   ( 1 )  " "                          "*"                     " "
3   ( 1 )  " "                          "*"                     " "
4   ( 1 )  " "                          "*"                     " "
5   ( 1 )  "*"                          "*"                     " "
6   ( 1 )  "*"                          " "                     " "
7   ( 1 )  " "                          " "                     " "
8   ( 1 )  "*"                          " "                     " "
9   ( 1 )  " "                          " "                     " "
10  ( 1 ) "*"                           " "                     " "
11  ( 1 ) " "                           " "                     " "
12  ( 1 ) "*"                           " "                     " "
13  ( 1 ) "*"                           " "                     " "
14  ( 1 ) "*"                           " "                     " "
15  ( 1 ) " "                           " "                     " "
16  ( 1 ) "*"                           "*"                     " "
17  ( 1 ) "*"                           "*"                     " "
18  ( 1 ) "*"                           " "                     " "
19  ( 1 ) "*"                           " "                     " "
20  ( 1 ) "*"                           " "                     " "
21  ( 1 ) "*"                           "*"                     "*"
22  ( 1 ) "*"                           "*"                     "*"
23  ( 1 ) "*"                           "*"                     "*"
24  ( 1 ) " "                           " "                     " "
25  ( 1 ) "*"                           "*"                     "*"
26  ( 1 ) "*"                           "*"                     "*"
```

```
27  ( 1 ) "*"                                 "*"                                 "*"
28  ( 1 ) "*"                                 "*"                                 "*"
          wtd_range_Valence
1   ( 1 )  " "
2   ( 1 )  " "
3   ( 1 )  " "
4   ( 1 )  " "
5   ( 1 )  " "
6   ( 1 )  " "
7   ( 1 )  " "
8   ( 1 )  " "
9   ( 1 )  " "
10  ( 1 ) " "
11  ( 1 ) " "
12  ( 1 ) " "
13  ( 1 ) " "
14  ( 1 ) " "
15  ( 1 ) " "
16  ( 1 ) " "
17  ( 1 ) " "
18  ( 1 ) " "
19  ( 1 ) " "
20  ( 1 ) " "
21  ( 1 ) " "
22  ( 1 ) " "
23  ( 1 ) " "
24  ( 1 ) " "
25  ( 1 ) "*"
26  ( 1 ) "*"
27  ( 1 ) " "
28  ( 1 ) "*"
```

In [97]:

```
par(mfrow = c(2, 2))
plot(reg.summary.both$cp, xlab = "Number of variables", ylab = "C_p", type = "l")
points(which.min(reg.summary.both$cp), reg.summary.both$cp[which.min(reg.summary.both$cp)]
plot(reg.summary.both$bic, xlab = "Number of variables", ylab = "BIC", type = "l")
points(which.min(reg.summary.both$bic), reg.summary.both$bic[which.min(reg.summary.both$bi
plot(reg.summary.both$adjr2, xlab = "Number of variables", ylab = "Adjusted R^2", type = "
points(which.max(reg.summary.both$adjr2), reg.summary.both$adjr2[which.max(reg.summary.both
       cex = 2, pch = 20)
plot(reg.summary.both$rss, xlab = "Number of variables", ylab = "RSS", type = "l")
mtext("Plots of C_p, BIC, adjusted R^2 and RSS for forward stepwise selection", side = 3,
```



Observations from the above plots:

- C_p is lowest for 28 variables.
- BIC is lowest for 26 variables.
- Adjusted R2 is maximum for 28 variables.
- RSS is lowest for 28 variables.

Examining the above plots, we can say that the number of predictors that will be best for our model is 28 variables.

In [98]:

```
# displaying the coefficients of the predictors using hybrid selecion method
coef(regfit.both, 28)
```

(Intercept): -94.2350224207959 **mean_atomic_mass:** 0.143807411042115
std_atomic_mass: 0.262468989633535 **gmean_fie:** 0.110729569941097 **wtd_gmean_fie:** -0.0381693050568142 **wtd_range_fie:** 0.0106981789269697 **mean_atomic_radius:** 0.277873444125612 **wtd_range_atomic_radius:** -0.151380406277972 **std_atomic_radius:** 0.332923466110127 **mean_Density:** -0.00328020517070798 **wtd_entropy_Density:** 6.84994688411627 **wtd_range_Density:** 0.00165887523934407 **wtd_std_Density:** -0.00290920716256939 **gmean_ElectronAffinity:** 0.0261571272206173
wtd_gmean_ElectronAffinity: -0.0746354571870964 **wtd_entropy_ElectronAffinity:** -38.6443655103919 **wtd_range_ElectronAffinity:** -0.238063428384718
std_ElectronAffinity: 0.105990273522438 **mean_FusionHeat:** 0.212241260386326
wtd_range_FusionHeat: -0.0229284401633827 **std_FusionHeat:** -0.555443818251626
mean_ThermalConductivity: 0.16584620715253 **wtd_gmean_ThermalConductivity:** -0.292305820498877 **entropy_ThermalConductivity:** 21.3580477936157
wtd_entropy_ThermalConductivity: 9.26486698266764 **wtd_range_ThermalConductivity:** 0.263516358752178 **std_ThermalConductivity:** 0.104598552407885 **gmean_Valence:** 2.73834890423956 **wtd_range_Valence:** -1.97589168368307

So, at the end of the Stepwise Selection Method of feature selection, we can say that neither of the methods removed any variables. Hence the model accuracy remains the same as model2. We can try doing the Multicollinearity test to see we can filter out any non-significant variables.

7. Third Linear Model After Stepwise:

7.1 Checking For Multicollinearity:

Multicollinearity refers to the association between two variables. Multicollinearity can sometimes make a variable less important when those variables should actually be significant. The $Variance\ Inflation\ Factor\ (VIF)$ quantifies the severity of multicollinearity in an ordinary least squares regression analysis:

$$VIF = 1/(1 - r^2)$$

Where r is the correlation between two independent variables. We usually say there's collinearity if $VIF \geq 10$.

Now, both the processes i.e. VIF and correlation coefficient are used to test multicollinearity. But there a slight difference. Both cannot depend on one another. It can happen that the variables with very high correlation coefficient can have very low VIF. Also it can happen that variables with very high VIF can have vey low correlation coefficient. Hence, we should always be careful what variables we are choosing for our model.

Therefore, in spite of performing correlation coefficient method at the very start, it would be better for us to check VIF values too. VIF cutoff taken here is 10. Anything below 10 can be taken into the model as significant variables and anything above 10 can be discarded as non-significant. Also it should be noted that the process of removing the variables should be one by one. As VIF changes for other variables left in every step. For

example, suppose "wtd_gmean_fie" and "mean_Density" has high VIF in the very first iteration. Removing first "wtd_gmean_fie" can alter the VIF of "mean_Density" in the secong iteration and it may or may not have high VIF.

In [99]:

```
# install.packages("car")
library(car)

stepTrainingData<-corTrainingData
# building a model with the latest dataset to check for initial VIFs
model4.afterStep.lm = lm(critical_temp~., data = stepTrainingData)
print(vif(model4.afterStep.lm))

# removing std_ThermalConductivity
stepTrainingData=subset(stepTrainingData, select=-c(26))
paste("Please note that the actual number of predictors is 28 minus critical_temp. And so t
paste("**********************************************Iteration 1**************************
paste("Number of Rows and Columns of the New Dataset After Stepwise Method->")
paste(dim(stepTrainingData))
# building linear model with the latest dataset
fit.stepTrainData.lm = lm(critical_temp~., data = stepTrainingData)
paste("removing -> std_ThermalConductivity")
print(vif(fit.stepTrainData.lm))

# removing wtd_gmean_fie
stepTrainingData=subset(stepTrainingData, select=-c(4))
paste("**********************************************Iteration 2**************************
paste("Number of Rows and Columns of the New Dataset After checking For Multicollinearity-
paste(dim(stepTrainingData))
# building linear model with the latest dataset
fit.stepTrainData.lm = lm(critical_temp~., data = stepTrainingData)
paste("removing -> wtd_gmean_fie")
print(vif(fit.stepTrainData.lm))

# removing mean_Density
stepTrainingData=subset(stepTrainingData, select=-c(8))
paste("**********************************************Iteration 3**************************
paste("Number of Rows and Columns of the New Dataset After checking For Multicollinearity-
paste(dim(stepTrainingData))
# building linear model with the latest dataset
fit.stepTrainData.lm = lm(critical_temp~., data = stepTrainingData)
paste("removing -> mean_Density")
print(vif(fit.stepTrainData.lm))

# removing wtd_gmean_ElectronAffinity
stepTrainingData=subset(stepTrainingData, select=-c(12))
paste("**********************************************Iteration 4**************************
paste("Number of Rows and Columns of the New Dataset After checking For Multicollinearity-
paste(dim(stepTrainingData))
# building linear model with the latest dataset
fit.stepTrainData.lm = lm(critical_temp~., data = stepTrainingData)
paste("removing -> wtd_gmean_ElectronAffinity")
print(vif(fit.stepTrainData.lm))
```

```
Loading required package: carData


Attaching package: 'car'


The following object is masked from 'package:purrr':
```

```
    some


The following object is masked from 'package:psych':

    logit
```

```
                mean_atomic_mass                  std_atomic_mass
                        6.613040                        3.355458
                       gmean_fie                    wtd_gmean_fie
                        8.734330                       17.970432
                   wtd_range_fie                mean_atomic_radius
                       11.350557                        5.301486
          wtd_range_atomic_radius                 std_atomic_radius
                        6.003443                        7.203703
                    mean_Density               wtd_entropy_Density
                       12.118902                        7.172007
                wtd_range_Density                  wtd_std_Density
                        4.374996                        3.582797
           gmean_ElectronAffinity        wtd_gmean_ElectronAffinity
                        7.236018                       11.066850
     wtd_entropy_ElectronAffinity        wtd_range_ElectronAffinity
                        7.796122                       10.370607
            std_ElectronAffinity                  mean_FusionHeat
                        2.865337                        5.661344
            wtd_range_FusionHeat                   std_FusionHeat
                        4.097840                        4.769893
        mean_ThermalConductivity    wtd_gmean_ThermalConductivity
                       10.375524                        8.265140
     entropy_ThermalConductivity wtd_entropy_ThermalConductivity
                        7.851006                        7.857392
   wtd_range_ThermalConductivity           std_ThermalConductivity
                        8.122304                       28.479507
                   gmean_Valence                wtd_range_Valence
                        6.812860                        6.114786
```

'Please note that the actual number of predictors is 28 minus critical_temp. And so for the rest of the iterations.'

'**********Iteration 1**********'

'Number of Rows and Columns of the New Dataset After Stepwise Method->'

'21263' '28'

'removing -> std_ThermalConductivity'

```
                mean_atomic_mass                  std_atomic_mass
                        6.587395                        3.353294
                       gmean_fie                    wtd_gmean_fie
                        8.427880                       17.786556
                   wtd_range_fie                mean_atomic_radius
                       11.227972                        5.274566
          wtd_range_atomic_radius                 std_atomic_radius
```

```
                      5.592804                      7.194361
                mean_Density            wtd_entropy_Density
                     12.115392                      6.014026
             wtd_range_Density                wtd_std_Density
                      4.373726                      3.505944
       gmean_ElectronAffinity     wtd_gmean_ElectronAffinity
                      7.033009                     11.034815
 wtd_entropy_ElectronAffinity     wtd_range_ElectronAffinity
                      7.421578                     10.320347
         std_ElectronAffinity               mean_FusionHeat
                      2.855846                      5.562535
         wtd_range_FusionHeat                std_FusionHeat
                      4.090652                      4.750699
    mean_ThermalConductivity  wtd_gmean_ThermalConductivity
                      2.993813                      4.010119
 entropy_ThermalConductivity wtd_entropy_ThermalConductivity
                      4.729819                      7.678788
wtd_range_ThermalConductivity                  gmean_Valence
                      5.712163                      6.453588
            wtd_range_Valence
                      6.074176
```

'**********Iteration 2**********'

'Number of Rows and Columns of the New Dataset After checking For Multicollinearity->'

'21263' · '27'

'removing -> wtd_gmean_fie'

```
               mean_atomic_mass                std_atomic_mass
                      6.586968                      3.351735
                     gmean_fie                  wtd_range_fie
                      4.377182                      4.247048
            mean_atomic_radius         wtd_range_atomic_radius
                      5.141824                      3.638897
             std_atomic_radius                   mean_Density
                      7.104950                     12.022633
          wtd_entropy_Density              wtd_range_Density
                      5.992374                      4.208934
              wtd_std_Density          gmean_ElectronAffinity
                      3.504126                      6.946684
    wtd_gmean_ElectronAffinity    wtd_entropy_ElectronAffinity
                     10.730034                      7.414617
    wtd_range_ElectronAffinity            std_ElectronAffinity
                     10.243722                      2.841250
             mean_FusionHeat             wtd_range_FusionHeat
                      5.552191                      4.075274
              std_FusionHeat         mean_ThermalConductivity
                      4.711453                      2.938315
 wtd_gmean_ThermalConductivity     entropy_ThermalConductivity
                      4.007875                      4.671449
wtd_entropy_ThermalConductivity   wtd_range_ThermalConductivity
                      7.636648                      5.333435
                gmean_Valence               wtd_range_Valence
                      6.412497                      5.602152
```

'**********Iteration 3**********'

'Number of Rows and Columns of the New Dataset After checking For Multicollinearity->'

'21263' · '26'

'removing -> mean_Density'

mean_atomic_mass	3.310109
std_atomic_mass	3.214097
gmean_fie	3.467382
wtd_range_fie	4.152227
mean_atomic_radius	4.507540
wtd_range_atomic_radius	3.638576
std_atomic_radius	6.622145
wtd_entropy_Density	5.954912
wtd_range_Density	4.108551
wtd_std_Density	3.027585
gmean_ElectronAffinity	6.937990
wtd_gmean_ElectronAffinity	10.628438
wtd_entropy_ElectronAffinity	7.403188
wtd_range_ElectronAffinity	10.238290
std_ElectronAffinity	2.812995
mean_FusionHeat	5.483863
wtd_range_FusionHeat	4.069493
std_FusionHeat	4.705734
mean_ThermalConductivity	2.682999
wtd_gmean_ThermalConductivity	3.874083
entropy_ThermalConductivity	4.671138
wtd_entropy_ThermalConductivity	7.635346
wtd_range_ThermalConductivity	5.268743
gmean_Valence	5.614948
wtd_range_Valence	5.593907

'**********Iteration 4**********'

'Number of Rows and Columns of the New Dataset After checking For Multicollinearity->'

'21263' · '25'

'removing -> wtd_gmean_ElectronAffinity'

mean_atomic_mass	3.261550
std_atomic_mass	3.206387
gmean_fie	3.464691
wtd_range_fie	4.107467
mean_atomic_radius	4.506195
wtd_range_atomic_radius	3.378147
std_atomic_radius	6.610355
wtd_entropy_Density	5.553628
wtd_range_Density	4.033710
wtd_std_Density	3.025989

```
          gmean_ElectronAffinity    wtd_entropy_ElectronAffinity
                        3.159975                        6.463434
      wtd_range_ElectronAffinity           std_ElectronAffinity
                        5.851116                        2.525863
                 mean_FusionHeat           wtd_range_FusionHeat
                        5.267362                        4.067791
                  std_FusionHeat       mean_ThermalConductivity
                        4.616325                        2.645927
    wtd_gmean_ThermalConductivity     entropy_ThermalConductivity
                        3.867924                        4.393732
  wtd_entropy_ThermalConductivity   wtd_range_ThermalConductivity
                        6.938747                        5.192339
                    gmean_Valence              wtd_range_Valence
                        5.601524                        5.586287
```

After performing VIF(), we are left with 24 predictors. Now let us display the summary of the model.

In [100]:

```
# displaying the summary of the model
summary(fit.stepTrainData.lm)
```

```
Call:
lm(formula = critical_temp ~ ., data = stepTrainingData)

Residuals:
    Min      1Q  Median      3Q     Max
-142.24  -13.34   -0.55   13.69  176.17

Coefficients:
                                 Estimate Std. Error t value Pr(>|t|)
(Intercept)                    -4.887e+01  4.124e+00 -11.850  < 2e-16 ***
mean_atomic_mass               -1.370e-02  8.718e-03  -1.572  0.11604
std_atomic_mass                 3.090e-01  1.280e-02  24.134  < 2e-16 ***
gmean_fie                       4.599e-02  3.404e-03  13.510  < 2e-16 ***
wtd_range_fie                  -2.977e-03  1.296e-03  -2.297  0.02161 *
mean_atomic_radius              1.480e-01  1.509e-02   9.803  < 2e-16 ***
wtd_range_atomic_radius        -8.173e-02  7.519e-03 -10.871  < 2e-16 ***
std_atomic_radius               3.978e-01  1.608e-02  24.732  < 2e-16 ***
wtd_entropy_Density             1.079e+01  1.056e+00  10.221  < 2e-16 ***
wtd_range_Density               1.195e-03  1.200e-04   9.966  < 2e-16 ***
wtd_std_Density                -3.821e-03  1.546e-04 -24.713  < 2e-16 ***
gmean_ElectronAffinity         -2.877e-02  8.779e-03  -3.277  0.00105 **
wtd_entropy_ElectronAffinity   -3.930e+01  1.273e+00 -30.864  < 2e-16 ***
wtd_range_ElectronAffinity     -2.728e-01  1.211e-02 -22.533  < 2e-16 ***
std_ElectronAffinity            1.403e-01  1.047e-02  13.402  < 2e-16 ***
mean_FusionHeat                 1.522e-01  2.909e-02   5.233 1.69e-07 ***
wtd_range_FusionHeat           -5.762e-02  2.531e-02  -2.276  0.02283 *
std_FusionHeat                 -5.098e-01  3.549e-02 -14.364  < 2e-16 ***
mean_ThermalConductivity        2.058e-01  6.050e-03  34.017  < 2e-16 ***
wtd_gmean_ThermalConductivity  -3.767e-01  7.010e-03 -53.739  < 2e-16 ***
entropy_ThermalConductivity     1.320e+01  9.211e-01  14.328  < 2e-16 ***
wtd_entropy_ThermalConductivity 1.323e+01  1.186e+00  11.162  < 2e-16 ***
wtd_range_ThermalConductivity   3.388e-01  7.569e-03  44.765  < 2e-16 ***
gmean_Valence                  -3.048e-01  3.241e-01  -0.941  0.34697
wtd_range_Valence              -1.836e+00  3.461e-01  -5.305 1.14e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 20.89 on 21238 degrees of freedom
Multiple R-squared:  0.6286,    Adjusted R-squared:  0.6281
F-statistic:  1498 on 24 and 21238 DF,  p-value: < 2.2e-16
```

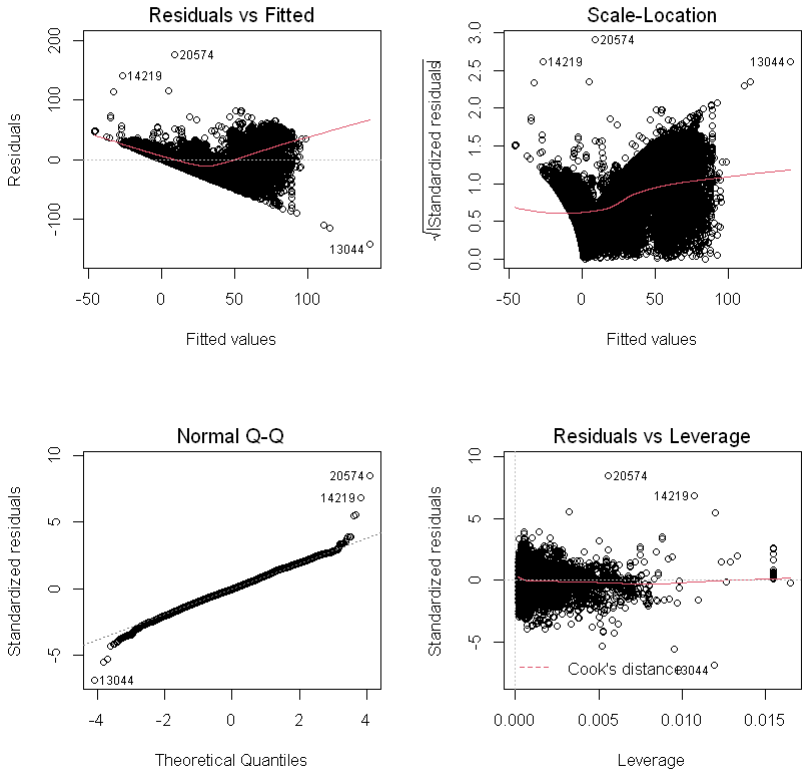
Observations from the Model Summary above:

- **Degrees of Freedom**: 21238
- **Model Parameters**: 24 plus intercept
- **Residual standard error**: 20.89
- **Adjusted R-squared**: 0.6281
- **F-statistic**: 1498 on 24 and 21238 DF

Although the Residual standard error is almost the same as the second model, we can see that Adjusted R2 has decreased by 0.1. Which should be fine as we have fairly less number of more significant variables for our model.

In [106]:

```
# plotting residual plots
par(mfcol=c(2,2))
# plot(model3.multicollinear.features.lm)
plot(fit.stepTrainData.lm)
```



Observations from the above plots:

- **Residuals v/s Fitted**: It still shows that the residuals have non linear pattern as they are not equally spread around the horizontal line and hence a distinct pattern can be seen in the plot.
- **Normal Q-Q plot**: Unlike our previous model, here we can see that most of the residuals are lined up well on the straight dashed line and hence we can say that the residuals are normally distributed.
- **Scale- Location**: Again, the residuals seem to be randomly spread along the line, also the line is not horizontal and it seems to be a curve more than a line. Hence the assumption of equal variance does not apply here.
- **Residuals vs Leverage**: Looking at the plot, we cannot find any outlaying values in the upper right corner or at the lower right corner, therefore no influential points can be seen that can influence the regression results. We can barely see Cook's distance lines because all cases are well inside of the Cook's distance lines. Therefore no influential cases are observed.

7.2 Checking p-values of predictors:

p-value is the probability value for a given statistical model, that if the null hypothesis is true, the statistical summary is greater than or equal in magnitude to the observed results. This probability is used to accept or reject the null hypothesis.

Removal of different features from the dataset wil have different effects on the p value of the dataset. We can remove different features and measure the p-value in each case. The measured p-value can be used to decide whether to keep that feature or not.

Now that we got our latest set of predictors in the previous linear model, we can see that still it consists of some predictors that are not as significant as others. For example, "mean_atomic_mass" and gmean_Valence have very high p-values of 0.11 and 0.3 respectively. Here threshold for p-value is 0.01. The smaller the p-value is, more significant the predictor. Therefore, in this step we will try to remove the predictors one by one by looking at their p-values which should be less than 0.01.

In [107]:

```
pValueTrainData <- stepTrainingData

# removing mean_atomic_mass
pValueTrainData<-subset(pValueTrainData, select=-c(1))
# building linear model with the latest dataset
fit.pValueTrainData.lm = lm(critical_temp~., data = pValueTrainData)

# removing gmean_Valence
pValueTrainData<-subset(pValueTrainData, select=-c(22))
# building linear model with the latest dataset
fit.pValueTrainData.lm = lm(critical_temp~., data = pValueTrainData)

# removing wtd_range_FusionHeat
pValueTrainData<-subset(pValueTrainData, select=-c(15))
# building linear model with the latest dataset
fit.pValueTrainData.lm = lm(critical_temp~., data = pValueTrainData)

# removing wtd_range_fie
pValueTrainData<-subset(pValueTrainData, select=-c(3))
# building linear model with the latest dataset
fit.pValueTrainData.lm = lm(critical_temp~., data = pValueTrainData)

summary(fit.pValueTrainData.lm)
```

```
Call:
lm(formula = critical_temp ~ ., data = pValueTrainData)

Residuals:
     Min       1Q   Median       3Q      Max 
-142.263  -13.397   -0.561   13.711  175.353 

Coefficients:
                               Estimate Std. Error t value Pr(>|t|)
(Intercept)                  -4.719e+01  4.055e+00 -11.638  < 2e-16 *
**
std_atomic_mass               2.967e-01  1.145e-02  25.921  < 2e-16 *
**
gmean_fie                     4.395e-02  3.330e-03  13.196  < 2e-16 *
**
mean_atomic_radius            1.293e-01  1.296e-02   9.980  < 2e-16 *
**
wtd_range_atomic_radius      -8.253e-02  7.379e-03 -11.185  < 2e-16 *
**
std_atomic_radius             4.014e-01  1.298e-02  30.927  < 2e-16 *
**
wtd_entropy_Density           1.029e+01  9.898e-01  10.397  < 2e-16 *
**
wtd_range_Density             1.200e-03  1.120e-04  10.707  < 2e-16 *
**
wtd_std_Density              -3.782e-03  1.497e-04 -25.264  < 2e-16 *
**
gmean_ElectronAffinity       -3.040e-02  7.926e-03  -3.835 0.000126 *
**
wtd_entropy_ElectronAffinity -3.877e+01  1.257e+00 -30.851  < 2e-16 *
**
wtd_range_ElectronAffinity   -2.838e-01  1.071e-02 -26.513  < 2e-16 *
```

```
**
std_ElectronAffinity             1.465e-01  1.003e-02  14.603  < 2e-16 *
**
mean_FusionHeat                  1.425e-01  2.591e-02   5.497 3.90e-08 *
**
std_FusionHeat                  -5.468e-01  3.118e-02 -17.536  < 2e-16 *
**
mean_ThermalConductivity         2.047e-01  5.345e-03  38.306  < 2e-16 *
**
wtd_gmean_ThermalConductivity   -3.776e-01  6.824e-03 -55.333  < 2e-16 *
**
entropy_ThermalConductivity      1.295e+01  8.927e-01  14.510  < 2e-16 *
**
wtd_entropy_ThermalConductivity  1.426e+01  1.124e+00  12.689  < 2e-16 *
**
wtd_range_ThermalConductivity    3.429e-01  7.079e-03  48.433  < 2e-16 *
**
wtd_range_Valence               -2.255e+00  2.995e-01  -7.531 5.22e-14 *
**
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 20.89 on 21242 degrees of freedom
Multiple R-squared:  0.6283,    Adjusted R-squared:  0.628
F-statistic:  1795 on 20 and 21242 DF,  p-value: < 2.2e-16
```

Observations from the Model Summary above:

- **Degrees of Freedom**: 21242
- **Model Parameters**: 20 plus intercept
- **Residual standard error**: 20.89
- **Adjusted R-squared**: 0.628
- **F-statistic**: 1795 on 20 and 21242 DF

Comparing with the previous model, we can see that all the statistics of the model remains the same with less number of features. Hence we will keep this list of features to build our final model using Stepwise Feature Selection Methods.

Building the Final Model 1 using Stepwise, Multicollinearity and p-value feature selection methods:

In [108]:

```
# list of features which would be used to build this final model 1 except critical_temp bec
# it is what we are going to predict
colnames(pValueTrainData)
```

'std_atomic_mass' · 'gmean_fie' · 'mean_atomic_radius' · 'wtd_range_atomic_radius' · 'std_atomic_radius' · 'wtd_entropy_Density' · 'wtd_range_Density' · 'wtd_std_Density' · 'gmean_ElectronAffinity' · 'wtd_entropy_ElectronAffinity' · 'wtd_range_ElectronAffinity' · 'std_ElectronAffinity' · 'mean_FusionHeat' · 'std_FusionHeat' · 'mean_ThermalConductivity' · 'wtd_gmean_ThermalConductivity' · 'entropy_ThermalConductivity' · 'wtd_entropy_ThermalConductivity' · 'wtd_range_ThermalConductivity' · 'wtd_range_Valence' · 'critical_temp'

Splitting the data into 80:20 ratio:

In [109]:

```
n = nrow(pValueTrainData)
trainIndex = sample(1:n, size = round(0.8*n), replace=FALSE)
train = pValueTrainData[trainIndex ,]
test = pValueTrainData[-trainIndex ,]
paste("Number of rows and columns in Training Data->")
paste(dim(train))
paste("Number of rows and columns in Testing Data->")
paste(dim(test))
```

'Number of rows and columns in Training Data->'

'17010' · '21'

'Number of rows and columns in Testing Data->'

'4253' · '21'

In [110]:

```
# building the final model
final.model.1 = lm(critical_temp~., data = train)

# predicting critical_temp using the final model
predict.final.model.1 <- predict(final.model.1, test)

# calculating MSE for final model 1
paste("MSE for Final Model 1->",mean((predict.final.model.1 - test$critical_temp)^2))

# displaying the summary of final model 1
summary(final.model.1)
```

'MSE for Final Model 1-> 428.926180223764'

```
Call:
lm(formula = critical_temp ~ ., data = train)

Residuals:
     Min       1Q   Median       3Q      Max 
-142.182  -13.369   -0.589   13.626  176.201 

Coefficients:
                                Estimate Std. Error t value Pr(>|t|)    
(Intercept)                   -5.258e+01  4.552e+00 -11.550  < 2e-16 ***
std_atomic_mass                2.902e-01  1.282e-02  22.633  < 2e-16 ***
gmean_fie                      4.709e-02  3.735e-03  12.606  < 2e-16 ***
mean_atomic_radius             1.451e-01  1.457e-02   9.963  < 2e-16 ***
wtd_range_atomic_radius       -8.244e-02  8.276e-03  -9.962  < 2e-16 ***
std_atomic_radius              4.105e-01  1.458e-02  28.162  < 2e-16 ***
wtd_entropy_Density            1.109e+01  1.109e+00  10.000  < 2e-16 ***
wtd_range_Density              1.185e-03  1.257e-04   9.433  < 2e-16 ***
wtd_std_Density               -3.733e-03  1.669e-04 -22.358  < 2e-16 ***
gmean_ElectronAffinity        -2.912e-02  8.933e-03  -3.260  0.00112 **  
wtd_entropy_ElectronAffinity  -3.900e+01  1.416e+00 -27.549  < 2e-16 ***
wtd_range_ElectronAffinity    -2.876e-01  1.202e-02 -23.920  < 2e-16 ***
std_ElectronAffinity           1.467e-01  1.126e-02  13.027  < 2e-16 ***
mean_FusionHeat                1.207e-01  2.963e-02   4.076 4.61e-05 ***
std_FusionHeat                -5.215e-01  3.544e-02 -14.713  < 2e-16 ***
mean_ThermalConductivity       2.058e-01  5.996e-03  34.329  < 2e-16 ***
wtd_gmean_ThermalConductivity -3.694e-01  7.646e-03 -48.310  < 2e-16 ***
entropy_ThermalConductivity    1.244e+01  9.960e-01  12.487  < 2e-16 ***
wtd_entropy_ThermalConductivity 1.410e+01  1.255e+00  11.238  < 2e-16 ***
wtd_range_ThermalConductivity  3.404e-01  7.908e-03  43.038  < 2e-16 ***
wtd_range_Valence             -2.052e+00  3.364e-01  -6.101 1.08e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 20.94 on 16989 degrees of freedom
Multiple R-squared:  0.6264,    Adjusted R-squared:  0.626 
F-statistic:  1424 on 20 and 16989 DF,  p-value: < 2.2e-16
```

8. Embedded Methods Feature Selection:

In embedded feature selection techniques, the feature selection algorithm is integrated as a part of the learning algorithm. Decision tree algorithm is the most common embedded technique. These algorithms select a feature in the recursive step of the tree growth process and divide the sample set into smaller subsets. The more child nodes in a subset are in the same class, the more informative the features are.

Splitting the data into 80:20 ratio:

In [111]:

```
n = nrow(corTrainingData)
trainIndex = sample(1:n, size = round(0.8*n), replace=FALSE)
train = corTrainingData[trainIndex ,]
test = corTrainingData[-trainIndex ,]
paste("Number of rows and columns in Training Data->")
paste(dim(train))
paste("Number of rows and columns in Testing Data->")
paste(dim(test))
```

'Number of rows and columns in Training Data->'

'17010' · '29'

'Number of rows and columns in Testing Data->'

'4253' · '29'

8.1 Lasso Model:

Lasso is one approach that we often used in machine learning to perform feature selection automatically. Depending on the shrinkage parameter, Lasso regularizes the coefficient in a way such that the estimated coefficients can be shrunk toward zero. The R library that we are going to use is the $glmnet$ and the function we are going to use is $cv.glmnet()$.

In [127]:

```
# install.packages("glmnet")
library(glmnet)
```

In [128]:

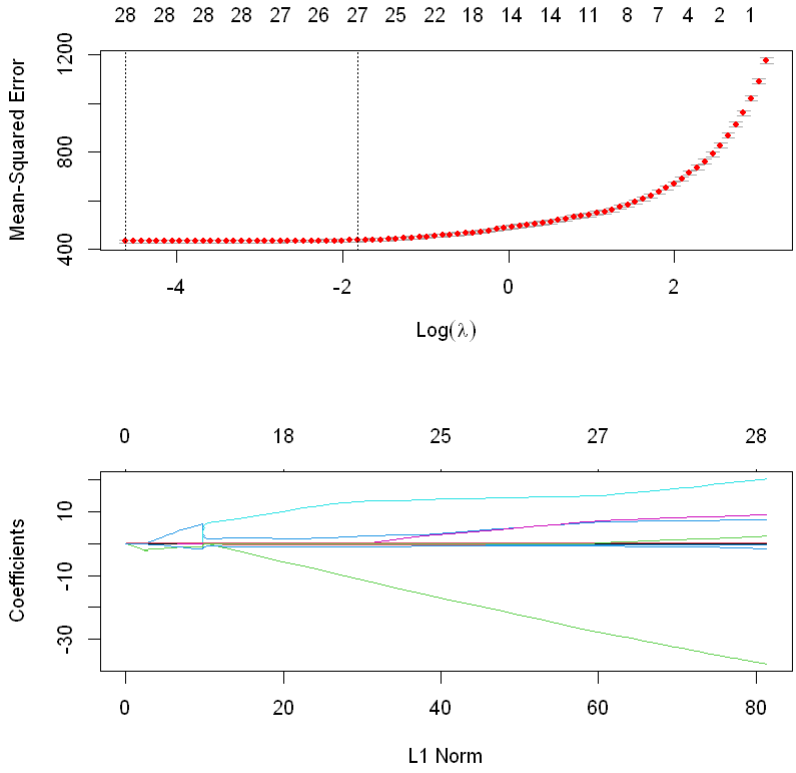
```
# install.packages("glmnet")
library(glmnet)

# model.matrix here is used to create a data matrix with predicted variables which converts
# dummy variables. cv.glmnet is used to fit lasso model with cross-validation.
# creating data matrix corresponding to variables, also transforming any qualitative variab
x_vars_lasso <- model.matrix(critical_temp~., data = train)
x_vars_lasso <- x_vars_lasso[, -1]
y_var_lasso <- train$critical_temp

# the purpose of fixing the seed of the random number generator is to make the result repea
set.seed(1)
# fitting a lasso model with cross-validation using the cv.glmnet() function
cv.lasso <- cv.glmnet(x_vars_lasso, y_var_lasso, alpha = 1, thresh = 1e-12)

# finding the cross-validated lambda value
bestlam.lasso <- cv.lasso$lambda.min
paste("Best Lambda for Lasso is->",bestlam.lasso)

# refitting lasso model using lambda chosen by cross-validation
fit.lasso <- glmnet(x_vars_lasso, y_var_lasso, alpha = 1, thresh = 1e-12)

# predicting the cofficients, no efficients are 0, hence all 28 variables are important
predict.lasso <- predict(fit.lasso, s = bestlam.lasso, type = "coefficients")[0:29, ]

par(mfrow=c(2,1))
# plotting the MSE as a function of the logarithm of  λ  with error bars
plot(cv.lasso)
# plotting the lasso model
plot(fit.lasso)
predict.lasso
```

```
Your code contains a unicode char which cannot be displayed in your
current locale and R will silently convert it to an escaped form when the
R kernel executes this code. This can lead to subtle errors if you use
such chars to do comparisons. For more information, please see
https://github.com/IRkernel/repr/wiki/Problems-with-unicode-on-windows (http
s://github.com/IRkernel/repr/wiki/Problems-with-unicode-on-windows)
```

'Best Lambda for Lasso is-> 0.00994439475669191'

(Intercept): -95.0874972025225 **mean_atomic_mass:** 0.133851648963002
std_atomic_mass: 0.270029831676017 **gmean_fie:** 0.108410306540115
wtd_gmean_fie: -0.0334432636448957 **wtd_range_fie:** 0.00805318122364968
mean_atomic_radius: 0.287093636772178 **wtd_range_atomic_radius:**
-0.147777848808228 **std_atomic_radius:** 0.316261847413995 **mean_Density:**
-0.00314311636061448 **wtd_entropy_Density:** 7.28166659316232
wtd_range_Density: 0.00162034208177168 **wtd_std_Density:** -0.00299541162442615
gmean_ElectronAffinity: 0.0296166961801033 **wtd_gmean_ElectronAffinity:**
-0.0815986957950492 **wtd_entropy_ElectronAffinity:** -37.6254876328881
wtd_range_ElectronAffinity: -0.226010407534296 **std_ElectronAffinity:**
0.0949059129253662 **mean_FusionHeat:** 0.207754108173646
wtd_range_FusionHeat: -0.0364020218687476 **std_FusionHeat:** -0.515641291267848
mean_ThermalConductivity: 0.17096833480693 **wtd_gmean_ThermalConductivity:**

-0.295905707183017 **entropy_ThermalConductivity:** 20.1670729912501
wtd_entropy_ThermalConductivity: 9.01181934031936
wtd_range_ThermalConductivity: 0.262634282980597 **std_ThermalConductivity:**
0.101322579808322 **gmean_Valence:** 2.29534157232859 **wtd_range_Valence:**
-1.66435365239147



Observations from the above Output:

- We can see that the best lambda for lasso is 0.0099
- By looking at the predicted coefficents by lasso, it is evident that none of the coefficients is shrunken to zero, which we unexpected. Probably all the features that went in were important.
- The first plot is a plot of cross validation. It shows the CV MSE as a function of log(lambda). At the start when log(lambda) is approximately 2.5, lamda is big, coefficients are restricted to be very small, MSE is big. Later towards the end, when lambda is very small and the coefficients are big, MSE becomes small and stays flat, indicating that the full model is probably the best. At the top of the model the number 28 indicates the presence of 28 predictors in the model.
- In the second graph we can say that the predictors go into the model in the order of the magnitude of their coefficients. So in this plot, all the predictors go into the model almost at the same time. Also, the coefficient path of each predictor is linear unless a new feature enters in. For example, in the plot above,

the predictors with the color light blue, light green, and dark blue has a bent in their linearity and hence their slope changes, as they were affected by entrance of other predictors. I would like to mention here that the slight bump in the line is created due to correlation with other variables. The coeffient path of all other predictors remains linear. To sum up, we can say that all the predictors are more or less important as they go into the model at the same time.

8.2 Ridge Regression Model:

In machine learning, we also call **Lasso as L1 regularizition** and **Ridge as L2 regularization**. The difference between the two regularization methods lies in how they penalize the estimated parameters of your model. The Ridge regularization will shrink all the estimated parameters towards zero, but never equal to zero. In contrast, the Lasso regularization will force some of the estimated parameters to be zero.

Lasso: aplpha = 1
Ridge: alpha = 0

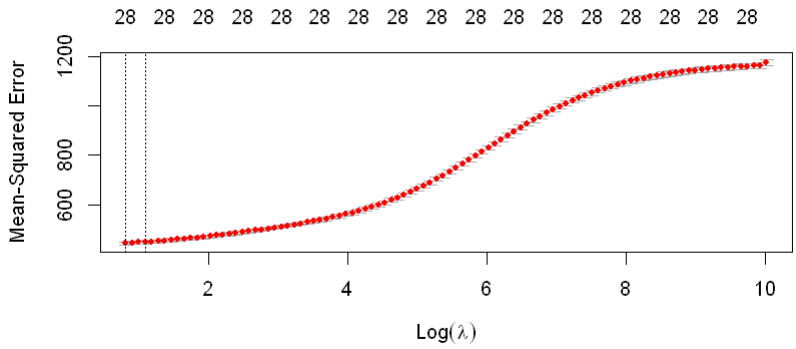
In [129]:

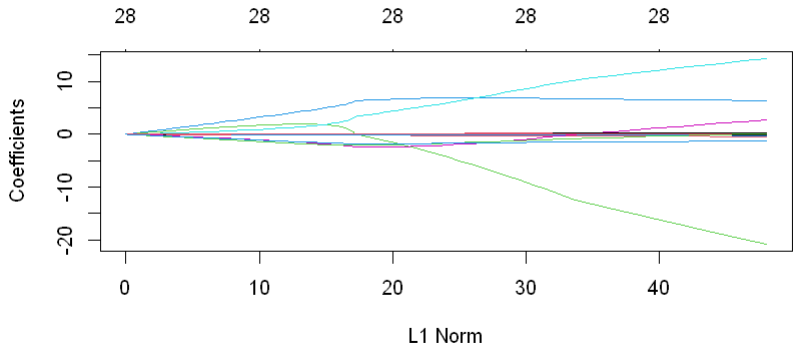
```
# install.packages("glmnet")
library(glmnet)

# creating data matrix corresponding to variables, also transforming any qualitative variab
x_vars_ridge <- model.matrix(critical_temp~., data = train)
x_vars_ridge <- x_vars_ridge[, -1]
y_var_ridge <- train$critical_temp

# the purpose of fixing the seed of the random number generator is to make the result repea
set.seed(1)
# fitting a ridge model with cross-validation using the cv.glmnet() function
cv.ridge <- cv.glmnet(x_vars_ridge, y_var_ridge, alpha = 0, thresh = 1e-12)

# finding the cross-validated lambda value
bestlam.ridge <- cv.ridge$lambda.min
paste("Best Lambda for Ridge is->",bestlam.ridge)

# refitting lasso model using lambda chosen by cross-validation
fit.ridge <- glmnet(x_vars_ridge, y_var_ridge, alpha = 0, thresh = 1e-12)

# predicting the cofficients, no efficients are 0, hence all 28 variables are important
predict.ridge <- predict(fit.ridge, s = bestlam.ridge, type = "coefficients")[0:29, ]

par(mfrow=c(2,1))
# plotting the MSE as a function of the logarithm of  λ  with error bars
plot(cv.ridge)
plot(fit.ridge)
predict.ridge
```

```
Your code contains a unicode char which cannot be displayed in your
current locale and R will silently convert it to an escaped form when the
R kernel executes this code. This can lead to subtle errors if you use
such chars to do comparisons. For more information, please see
https://github.com/IRkernel/repr/wiki/Problems-with-unicode-on-windows (http
s://github.com/IRkernel/repr/wiki/Problems-with-unicode-on-windows)
```

'Best Lambda for Ridge is-> 2.24446950967932'

(Intercept): -61.2792394811303 **mean_atomic_mass:** 0.0691544704196705
std_atomic_mass: 0.228492933267121 **gmean_fie:** 0.0601751833149839 **wtd_gmean_fie:**
-0.00344204699541602 **wtd_range_fie:** -0.00223433092772159 **mean_atomic_radius:**
0.198909285220299 **wtd_range_atomic_radius:** -0.0867036586992824 **std_atomic_radius:**
0.252975655389963 **mean_Density:** -0.00165733784951247 **wtd_entropy_Density:**
6.38359555685445 **wtd_range_Density:** 0.000779191222411637 **wtd_std_Density:**
-0.00233434827515355 **gmean_ElectronAffinity:** -0.0263578073852773
wtd_gmean_ElectronAffinity: -0.109911440159582 **wtd_entropy_ElectronAffinity:**
-20.7322474318167 **wtd_range_ElectronAffinity:** -0.105465602682036
std_ElectronAffinity: 0.0695262339087055 **mean_FusionHeat:** 0.14908576911536
wtd_range_FusionHeat: -0.0530620847247198 **std_FusionHeat:** -0.420856621338628
mean_ThermalConductivity: 0.133991879223793 **wtd_gmean_ThermalConductivity:**
-0.216376614375514 **entropy_ThermalConductivity:** 14.2470511908783
wtd_entropy_ThermalConductivity: 2.72495966642824 **wtd_range_ThermalConductivity:**
0.211805458415014 **std_ThermalConductivity:** 0.0982969234650962 **gmean_Valence:**
0.133983984567691 **wtd_range_Valence:** -1.30778379139291





Observations from the above Output:

- We can see that the best lambda for lasso is 2.238
- By looking at the predicted coefficents by ridge, it is evident that all the coefficients are approximately zero but not zero. Probably all the features that went in were important.
- The first plot is a plot of cross validation. It shows the CV MSE as a function of log(lambda). At the start when log(lambda) is approximately 10, lamda is big, coefficients are restricted to be very small, MSE is big. Later towards the end, when lambda is very small and the coefficients are big, MSE becomes small and stays flat, indicating that the full model is probably the best. At the top of the model the number 28 indicates the presence of 28 predictors in the model.
- Same as Lasso, in the second graph we can say that the predictors go into the model in the order of the magnitude of their coefficients. So in this plot, all the predictors go into the model almost at the same time. Also, the coefficient path of each predictor is linear unless a new feature enters in. For example, in the plot above, the predictors with the color light blue, light green, and dark blue has a bent in their linearity and hence their slope changes, as they were affected by entrance of other predictors. I would like to mention here that the slight bump in the line is created due to correlation with other variables. The coeffient path of all other predictors remains linear. To sum up, we can say that all the predictors are more or less important as they go into the model at the same time.

8.3 Random Forest Model:

Random Forest is one of the popular machine learning algorithms. As we know, random forest works on decison trees. This algorithm has an ensemble approach, in other words nearest neighbour predictor. Ensembles use divide and conquer approach to improve performance. As with other models, this algoritmn also has some advantages and disadvantages. Random forest model enables relatively fast runtimes. The are also able to deal with unbalanced data, missing data and other data based inconsistencies and umknowns. On the other hand, random forests models cannot predict beyonf the training data rance, specifically when used for regression. Additionally they may overfit the data sets that are particular "noisy".

In [153]:

```
# install.packages("randomForest")
library('randomForest')

# Using random forest for variable selection
randomForModel <-randomForest(critical_temp ~ ., data = train)

# Getting the list of important variables
importance(randomForModel)
```

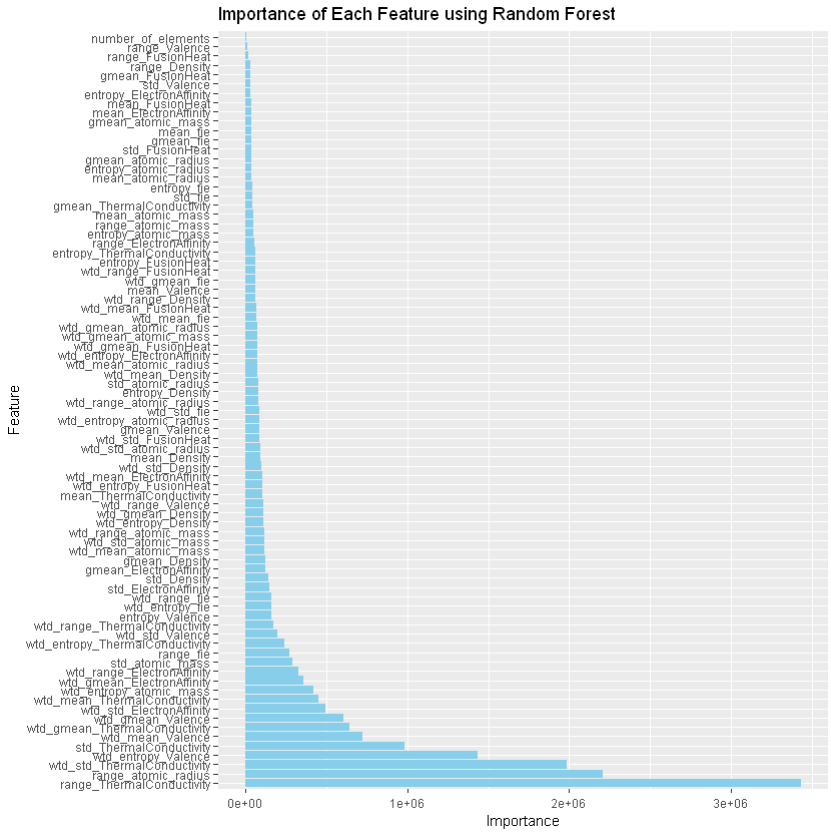
A matrix: 81 × 1 of type dbl

	IncNodePurity
number_of_elements	1567.404
mean_atomic_mass	43967.746
wtd_mean_atomic_mass	114014.258
gmean_atomic_mass	31834.493
wtd_gmean_atomic_mass	69937.272
entropy_atomic_mass	46842.275
wtd_entropy_atomic_mass	416010.298
range_atomic_mass	46015.195
wtd_range_atomic_mass	112169.116
std_atomic_mass	290228.627
wtd_std_atomic_mass	112313.445
mean_fie	33180.608
wtd_mean_fie	64629.749
gmean_fie	33889.268
wtd_gmean_fie	60991.274
entropy_fie	38664.874
wtd_entropy_fie	157670.665
range_fie	271613.495
wtd_range_fie	155235.243
std_fie	41397.335
wtd_std_fie	81284.462
mean_atomic_radius	36429.670
wtd_mean_atomic_radius	70812.728
gmean_atomic_radius	34613.221
wtd_gmean_atomic_radius	69534.590
entropy_atomic_radius	35119.287
wtd_entropy_atomic_radius	81948.420
range_atomic_radius	2205788.443
wtd_range_atomic_radius	77458.798

	IncNodePurity
std_atomic_radius	75478.532
...	...
mean_FusionHeat	31297.809
wtd_mean_FusionHeat	64095.986
gmean_FusionHeat	28565.315
wtd_gmean_FusionHeat	70042.097
entropy_FusionHeat	57291.244
wtd_entropy_FusionHeat	103594.758
range_FusionHeat	15405.430
wtd_range_FusionHeat	57550.782
std_FusionHeat	33935.948
wtd_std_FusionHeat	86058.749
mean_ThermalConductivity	103963.998
wtd_mean_ThermalConductivity	452259.338
gmean_ThermalConductivity	41940.254
wtd_gmean_ThermalConductivity	638863.840
entropy_ThermalConductivity	56078.571
wtd_entropy_ThermalConductivity	239187.469
range_ThermalConductivity	3432315.932
wtd_range_ThermalConductivity	172607.197
std_ThermalConductivity	983098.847
wtd_std_ThermalConductivity	1983084.121
mean_Valence	61025.016
wtd_mean_Valence	724625.450
gmean_Valence	82788.107
wtd_gmean_Valence	602377.242
entropy_Valence	160993.797
wtd_entropy_Valence	1434641.007
range_Valence	9057.766
wtd_range_Valence	108496.545
std_Valence	30036.990
wtd_std_Valence	196319.277

In [154]:

```
randomDF<-importance(randomForModel)

# adding a column "Feature" to the dataframe so that it makes it easier for plotting
randomDF <-as.data.frame(randomDF)
randomDF$Feature<-rownames(randomDF)

# plotting a graph of the importance of the variables using random forest
ggplot(data=randomDF, aes(x=reorder(Feature, -IncNodePurity), y=IncNodePurity)) +
        geom_bar(stat="identity",fill = "sky blue")+ coord_flip()+
        theme(text = element_text(size=9), axis.text.y = element_text(hjust=1))+
        labs(title="Importance of Each Feature using Random Forest",y = "Importance", x="Fe
```



Looking at the above plot we can say that 'wtd_range_ThermalConductivity', 'std_ThermalConductivity', 'gmean_Valence' and 'wtd_range_Valence' are most important variables and the least important variables are 'gmean_fie' and 'std_FusionHeat'. Based on our intuition and looking at the plot, we can take top 20 features from the random forest model.

Hence the columns we are selecting here are:

'std_atomic_mass', 'wtd_gmean_fie', 'wtd_range_fie', 'std_atomic_radius', 'wtd_entropy_Density',

'wtd_std_Density', 'gmean_ElectronAffinity', 'wtd_gmean_ElectronAffinity', 'wtd_entropy_ElectronAffinity', 'wtd_range_ElectronAffinity', 'std_ElectronAffinity', 'mean_FusionHeat', 'wtd_range_FusionHeat', 'mean_ThermalConductivity', 'wtd_gmean_ThermalConductivity', 'wtd_entropy_ThermalConductivity', 'wtd_range_ThermalConductivity', 'std_ThermalConductivity', 'gmean_Valence', 'wtd_range_Valence'

In [155]:

```
# displaying the summary of Random Forest Model
print(randomForModel)
```

```
Call:
 randomForest(formula = critical_temp ~ ., data = train)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 27

          Mean of squared residuals: 83.07631
                    % Var explained: 92.92
```

8.4 Comparing the 3 Emdebbed Feature Selection Models By MSE:

In [156]:

```
# predicting critical_temp using all the 3 methods above
predict.lasso <- predict(fit.lasso, s = bestlam.lasso, newx = x_vars_lasso)
predic.ridge <- predict(fit.ridge, s = bestlam.ridge, newx = x_vars_ridge)
predict.random <- predict(randomForModel, test)
```

In [131]:

```
paste("MSE for Lasso->",mean((predict.lasso - test$critical_temp)^2))
paste("MSE for Ridge->",mean((predic.ridge - test$critical_temp)^2))
paste("MSE for Random Forest->",mean((predict.random - test$critical_temp)^2))
```

```
Warning message in predict.lasso - test$critical_temp:
"longer object length is not a multiple of shorter object length"
```

'MSE for Lasso-> 1900.42152625157'

```
Warning message in predic.ridge - test$critical_temp:
"longer object length is not a multiple of shorter object length"
```

'MSE for Ridge-> 1830.24643617511'

'MSE for Random Forest-> 87.8297201432046'

Building the Final Model 2 after Embedded feature selection methods:

After performing embedded feature selection, i.e. Lasso, Ridge and Random Forest, we calculated the corresponding MSEs as we can see above.

In terms of MSEs: **Random Forest < Ridge < Lasso**

Hence we select Random Forest Model as our Final Model 2 for comparison with Final Model 1.

In [132]:

```
# calculating R squared
rSquared_random<-1-(sum((test$critical_temp-predict.random)^2)/sum((test$critical_temp-mean
cat("R2 value for a Random Forest Model is ->",rSquared_random)
```

```
R2 value for a Random Forest Model is -> 0.9243475
```

In [133]:

```
# calculating residual sum of squares
rse_random<-sqrt(sum((predict.random-test$critical_temp)^2)/(nrow(train)-2))
cat("Residual Standard Error RSE for a Random Forest Model is ->",rse_random)
```

```
Residual Standard Error RSE for a Random Forest Model is -> 4.686427
```

9. XGBoost Model with All the Features (Final Model 3):

This is a type of gradient Boosting models to create an ensemble of trees to predict a response. The trees are added in a sequential manner to improve the model by accounting for the points which are difficult to predict. Once a gradient boosted model is fitted, the weighted average of all the trees is used to give a final prediction. Gradient Boosted models predict well because they are able to account for the complex interactions and correlations among the features.

Coverting into Numeric Datatypes:

In [134]:

```
trainingXGB <- trainingData
# converting to numeric type datatype
trainingXGB[["number_of_elements"]] <- as.numeric(trainingXGB[["number_of_elements"]])
trainingXGB[["range_atomic_radius"]] <- as.numeric(trainingXGB[["range_atomic_radius"]])
trainingXGB[["range_Valence"]] <- as.numeric(trainingXGB[["range_Valence"]])

# displaying the number of rows and columns
dim(trainingXGB)

# displaying all the variables with class
sapply(trainingXGB, class)
```

21263 · 82

number_of_elements: 'numeric' **mean_atomic_mass:** 'numeric' **wtd_mean_atomic_mass:** 'numeric' **gmean_atomic_mass:** 'numeric' **wtd_gmean_atomic_mass:** 'numeric' **entropy_atomic_mass:** 'numeric' **wtd_entropy_atomic_mass:** 'numeric' **range_atomic_mass:** 'numeric' **wtd_range_atomic_mass:** 'numeric' **std_atomic_mass:** 'numeric' **wtd_std_atomic_mass:** 'numeric' **mean_fie:** 'numeric' **wtd_mean_fie:** 'numeric' **gmean_fie:** 'numeric' **wtd_gmean_fie:** 'numeric' **entropy_fie:** 'numeric' **wtd_entropy_fie:** 'numeric' **range_fie:** 'numeric' **wtd_range_fie:** 'numeric' **std_fie:** 'numeric' **wtd_std_fie:** 'numeric' **mean_atomic_radius:** 'numeric' **wtd_mean_atomic_radius:** 'numeric' **gmean_atomic_radius:** 'numeric' **wtd_gmean_atomic_radius:** 'numeric' **entropy_atomic_radius:** 'numeric' **wtd_entropy_atomic_radius:** 'numeric' **range_atomic_radius:** 'numeric' **wtd_range_atomic_radius:** 'numeric' **std_atomic_radius:** 'numeric' **wtd_std_atomic_radius:** 'numeric' **mean_Density:** 'numeric' **wtd_mean_Density:** 'numeric' **gmean_Density:** 'numeric' **wtd_gmean_Density:** 'numeric' **entropy_Density:** 'numeric' **wtd_entropy_Density:** 'numeric' **range_Density:** 'numeric' **wtd_range_Density:** 'numeric' **std_Density:** 'numeric' **wtd_std_Density:** 'numeric' **mean_ElectronAffinity:** 'numeric' **wtd_mean_ElectronAffinity:** 'numeric' **gmean_ElectronAffinity:** 'numeric' **wtd_gmean_ElectronAffinity:** 'numeric' **entropy_ElectronAffinity:** 'numeric' **wtd_entropy_ElectronAffinity:** 'numeric' **range_ElectronAffinity:** 'numeric' **wtd_range_ElectronAffinity:** 'numeric' **std_ElectronAffinity:** 'numeric' **wtd_std_ElectronAffinity:** 'numeric' **mean_FusionHeat:** 'numeric' **wtd_mean_FusionHeat:** 'numeric' **gmean_FusionHeat:** 'numeric' **wtd_gmean_FusionHeat:** 'numeric' **entropy_FusionHeat:** 'numeric' **wtd_entropy_FusionHeat:** 'numeric' **range_FusionHeat:** 'numeric' **wtd_range_FusionHeat:** 'numeric' **std_FusionHeat:** 'numeric' **wtd_std_FusionHeat:** 'numeric' **mean_ThermalConductivity:** 'numeric' **wtd_mean_ThermalConductivity:** 'numeric' **gmean_ThermalConductivity:** 'numeric' **wtd_gmean_ThermalConductivity:** 'numeric' **entropy_ThermalConductivity:** 'numeric' **wtd_entropy_ThermalConductivity:** 'numeric' **range_ThermalConductivity:** 'numeric' **wtd_range_ThermalConductivity:** 'numeric' **std_ThermalConductivity:** 'numeric' **wtd_std_ThermalConductivity:** 'numeric' **mean_Valence:** 'numeric' **wtd_mean_Valence:** 'numeric' **gmean_Valence:** 'numeric' **wtd_gmean_Valence:** 'numeric' **entropy_Valence:** 'numeric' **wtd_entropy_Valence:** 'numeric' **range_Valence:** 'numeric' **wtd_range_Valence:** 'numeric' **std_Valence:** 'numeric' **wtd_std_Valence:** 'numeric' **critical_temp:** 'numeric'

Loading the libraries:

In [139]:

```
# options(repos=c(CRAN="https://cran.rstudio.com/"))
# install.packages("xgboost")
# install.packages("caret")
# install.packages("readr")
# install.packages("stringr")
# install.packages("car")

library(xgboost)
library(readr)
library(stringr)
library(caret)
library(car)
```

Splitting the data into 80:20 ratio:

In [140]:

```
n = nrow(trainingXGB)
trainIndex = sample(1:n, size = round(0.8*n), replace=FALSE)
train = trainingXGB[trainIndex ,]
test = trainingXGB[-trainIndex ,]
paste("Number of rows and columns in Training Data->")
paste(dim(train))
paste("Number of rows and columns in Testing Data->")
paste(dim(test))
```

'Number of rows and columns in Training Data->'

'17010' · '82'

'Number of rows and columns in Testing Data->'

'4253' · '82'

Tuning and Running the Model

In [141]:

```
train_target <- train$critical_temp
test_target <- test$critical_temp
new_train <- model.matrix(~.+0,data = train[,-c(82)],with=F)
new_test <- model.matrix(~.+0,data = test[,-c(82)],with=F)

#convert factor to numeric
train_target <- as.numeric(train_target)-1
test_target <- as.numeric(test_target)-1
```

In [142]:

```
#preparing matrix
dtrain <- xgb.DMatrix(data = new_train,label = train_target)
dtest <- xgb.DMatrix(data = new_test, label = test_target)
```

In [143]:

```
#default parameters
params <- list(booster = "gbtree", objective = "reg:linear", eta=0.2, gamma=0, max_depth=1(
               min_child_weight=1, subsample=1, colsample_bytree=1)
```

In [144]:

```
xgbcv <- xgb.cv( params = params, data = dtrain, nrounds = 100, nfold = 10, showsd = T, st
                print.every.n = 10, early.stop.round = 20, maximize = F)
```

```
Warning message:
"'print.every.n' is deprecated.
Use 'print_every_n' instead.
See help("Deprecated") and help("xgboost-deprecated")."
Warning message:
"'early.stop.round' is deprecated.
Use 'early_stopping_rounds' instead.
See help("Deprecated") and help("xgboost-deprecated")."


[18:46:38] WARNING: amalgamation/../src/objective/regression_obj.cu:174: re
g:linear is now deprecated in favor of reg:squarederror.
[18:46:38] WARNING: amalgamation/../src/objective/regression_obj.cu:174: re
g:linear is now deprecated in favor of reg:squarederror.
[18:46:38] WARNING: amalgamation/../src/objective/regression_obj.cu:174: re
g:linear is now deprecated in favor of reg:squarederror.
[18:46:38] WARNING: amalgamation/../src/objective/regression_obj.cu:174: re
g:linear is now deprecated in favor of reg:squarederror.
[18:46:38] WARNING: amalgamation/../src/objective/regression_obj.cu:174: re
g:linear is now deprecated in favor of reg:squarederror.
[18:46:39] WARNING: amalgamation/../src/objective/regression_obj.cu:174: re
g:linear is now deprecated in favor of reg:squarederror.
[18:46:39] WARNING: amalgamation/../src/objective/regression_obj.cu:174: re
g:linear is now deprecated in favor of reg:squarederror.
[18:46:39] WARNING: amalgamation/../src/objective/regression_obj.cu:174: re
g:linear is now deprecated in favor of reg:squarederror.
[18:46:39] WARNING: amalgamation/../src/objective/regression_obj.cu:174: re
g:linear is now deprecated in favor of reg:squarederror.
[18:46:39] WARNING: amalgamation/../src/objective/regression_obj.cu:174: re
g:linear is now deprecated in favor of reg:squarederror.
[1]	train-rmse:38.510390+0.086272	test-rmse:38.771918+0.865163
Multiple eval metrics are present. Will use test_rmse for early stopping.
Will train until test_rmse hasn't improved in 20 rounds.

[11]	train-rmse:6.662020+0.030965	test-rmse:10.810217+0.481549
[21]	train-rmse:4.245216+0.040129	test-rmse:9.678281+0.443447
[31]	train-rmse:4.066344+0.037338	test-rmse:9.655507+0.440439
[41]	train-rmse:4.024365+0.036674	test-rmse:9.665246+0.441557
Stopping. Best iteration:
[30]	train-rmse:4.073392+0.036918	test-rmse:9.654905+0.441031
```

In [145]:

```
# building xgboost model
xgb1 <- xgb.train (params = params, data = dtrain, nrounds = 100, watchlist = list(val=dtes
                   print.every.n = 10, early.stop.round = 10, maximize = F , eval_metric =
```

```
Warning message:
"'print.every.n' is deprecated.
Use 'print_every_n' instead.
See help("Deprecated") and help("xgboost-deprecated")."
Warning message:
"'early.stop.round' is deprecated.
Use 'early_stopping_rounds' instead.
See help("Deprecated") and help("xgboost-deprecated")."

[18:47:41] WARNING: amalgamation/../src/objective/regression_obj.cu:174: re
g:linear is now deprecated in favor of reg:squarederror.
[18:47:41] WARNING: amalgamation/../src/learner.cc:516:
Parameters: { early_stop_round, print_every_n } might not be used.

  This may not be accurate due to some parameters are only used in language
bindings but
  passed down to XGBoost core.  Or some parameters are not used but slip thr
ough this
  verification. Please open an issue if you find above cases.


[1]     val-rmse:38.806194      train-rmse:38.503002
Multiple eval metrics are present. Will use train_rmse for early stopping.
Will train until train_rmse hasn't improved in 10 rounds.

[11]    val-rmse:11.135551      train-rmse:6.714009
[21]    val-rmse:10.147174      train-rmse:4.325082
[31]    val-rmse:10.171157      train-rmse:4.148352
[41]    val-rmse:10.195969      train-rmse:4.105887
[51]    val-rmse:10.206573      train-rmse:4.089798
[61]    val-rmse:10.217288      train-rmse:4.077995
[71]    val-rmse:10.225932      train-rmse:4.071424
[81]    val-rmse:10.235764      train-rmse:4.066199
[91]    val-rmse:10.240043      train-rmse:4.063998
[100]   val-rmse:10.242629      train-rmse:4.062681
```

In [146]:

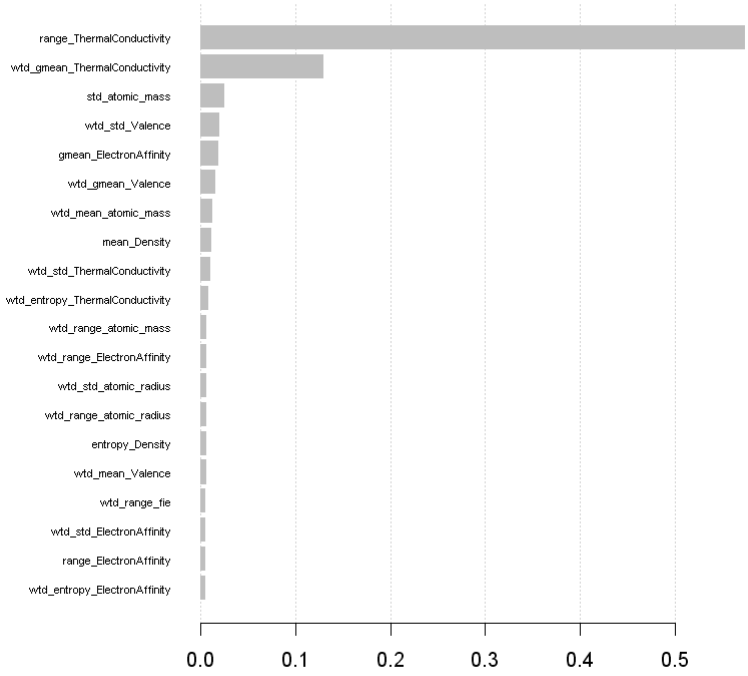
```
# predict critical_temp using xgboost model
predict.xgb <- predict (xgb1,dtest)
```

In [147]:

```
# calculating importance of features
mat <- xgb.importance (feature_names = colnames(new_train),model = xgb1)

# displaying MSE for xgboost model
paste("MSE for XGBoost->",mean((predict.xgb - test$critical_temp)^2))
paste("Below is the Feature Importance by XGBoost Method:")

# displaying the plot of importance of variables
xgb.plot.importance (importance_matrix = mat[1:20])
```

'MSE for XGBoost-> 106.292904039029'

'Below is the Feature Importance by XGBoost Method:'



In [148]:

```
# calculating R squared
rSquared_xgboost<-1-(sum((test$critical_temp-predict.xgb)^2)/sum((test$critical_temp-mean(t
cat("R2 value for a XGBoost Model is ->",rSquared_xgboost)
```

```
R2 value for a XGBoost Model is -> 0.909163
```

In [149]:

```
# calculating residual standard error
rse_xgboost<-sqrt(sum((predict.xgb-test$critical_temp)^2)/(nrow(train)-2))
cat("Residual Standard Error RSE for a XGBoost Model is ->",rse_xgboost)
```

```
Residual Standard Error RSE for a XGBoost Model is -> 5.155529
```

10. Model Comparison and Explaination:

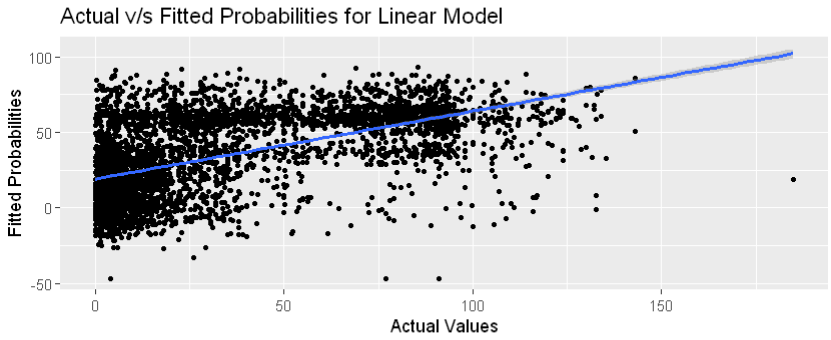
Final Model 1 | Linear Model

- **Model Parameters**: 20 plus intercept
- **Residual standard error**: 20.89
- **Adjusted R-squared**: 0.6278
- **MSE**: 425.115626189918
- **Features Selected**: std_atomic_mass, gmean_fie, mean_atomic_radius, wtd_range_atomic_radius, std_atomic_radius, wtd_entropy_Density, wtd_range_Density, wtd_std_Density, gmean_ElectronAffinity, wtd_entropy_ElectronAffinity, wtd_range_ElectronAffinity, std_ElectronAffinity, mean_FusionHeat, std_FusionHeat, mean_ThermalConductivity, wtd_gmean_ThermalConductivity, entropy_ThermalConductivity, wtd_entropy_ThermalConductivity, wtd_range_ThermalConductivity, wtd_range_Valence

In [150]:

```
library(ggplot2)

DF1<-as.data.frame(predict.final.model.1)
DF1$actual<-test$critical_temp

# plotting of actual v/s fitted probabilities of linear model
ggplot(DF1, aes(y=predict.final.model.1,x=actual)) + geom_point(size=1) + geom_smooth(metho
        labs(title="Actual v/s Fitted Probabilities for Linear Model", x="Actual Values", y
        coord_fixed(ratio=0.4)
```

```
`geom_smooth()` using formula 'y ~ x'
```



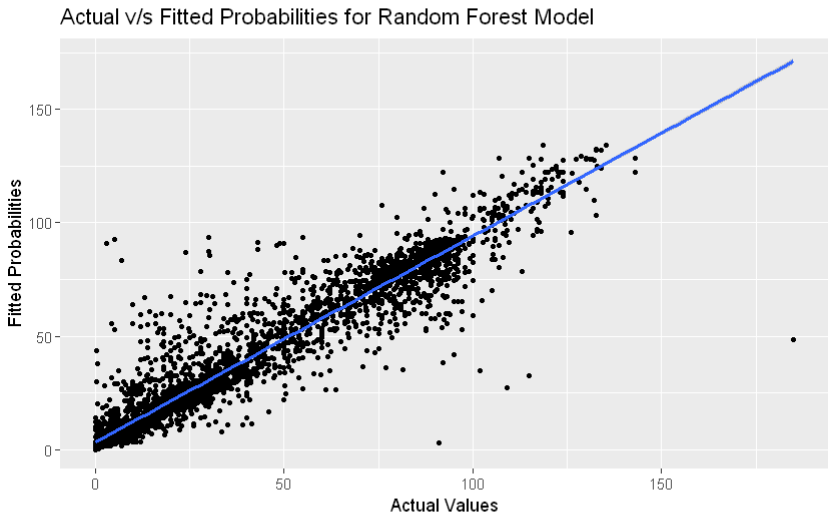
Final Model 2 | Random Forest

- **Model Parameters**: 20
- **Residual standard error**: 4.521203
- **R-squared**: 0.9301894
- **MSE**: 81.7458667311886
- **Features Selected**: std_atomic_mass, wtd_gmean_fie, wtd_range_fie, std_atomic_radius, wtd_entropy_Density, wtd_std_Density, gmean_ElectronAffinity, wtd_gmean_ElectronAffinity, wtd_entropy_ElectronAffinity, wtd_range_ElectronAffinity', std_ElectronAffinity, 'mean_FusionHeat, wtd_range_FusionHeat, mean_ThermalConductivity, wtd_gmean_ThermalConductivity, wtd_entropy_ThermalConductivity, wtd_range_ThermalConductivity, std_ThermalConductivity, gmean_Valence, wtd_range_Valence

In [157]:

```
library(ggplot2)

DF2<-as.data.frame(predict.random)
DF2$actual<-test$critical_temp

# plotting of actual v/s fitted probabilities of random forest model
ggplot(DF2, aes(y=predict.random,x=actual)) + geom_point(size=1) + geom_smooth(method = "lr
        labs(title="Actual v/s Fitted Probabilities for Random Forest Model", x="Actual Va
        coord_fixed(ratio=0.6)

# predict.random
```

```
`geom_smooth()` using formula 'y ~ x'
```



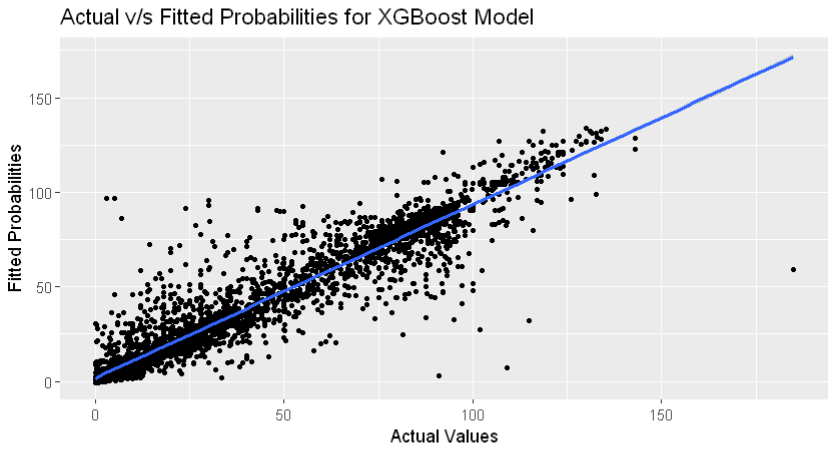
Final Model 3 | XGBoost

- **Model Parameters**: 20
- **Residual standard error**: 5.028832
- **R-squared**: 0.9139599
- **MSE**: 101.132810184913
- **Features Selected**: range_ThermalConductivity, wtd_gmean_ThermalConductivity, std_Density, wtd_mean_Valence, gmean_Density, range_atomic_radius, wtd_range_atomic_mass, entropy_ThermalConductivity, wtd_gmean_Valence, mean_Density, entropy_Density, wtd_range_fie, wtd_mean_atomic_mass, wtd_entropy_Valence, wtd_std_ElectronAffinity, wtd_std_ThermalConductivity, std_atomic_mass, wtd_entropy_atomic_mass, wtd_entropy_FusionHeat, wtd_Entropy_ThermalConductivity

In [152]:

```
library(ggplot2)

DF3<-as.data.frame(predict.xgb)
DF3$actual<-test$critical_temp

# plotting of actual v/s fitted probabilities of xgboost model
ggplot(DF3, aes(y=predict.xgb,x=actual)) + geom_point(size=1) + geom_smooth(method = "lm",s
        labs(title="Actual v/s Fitted Probabilities for XGBoost Model", x="Actual Values",
        coord_fixed(ratio=0.5)
```

```
`geom_smooth()` using formula 'y ~ x'
```



Comparing the above three final model, the best model chosen here is obviously Random Forest Model. The parameters passed in this model was 30 variables which we got after filtering out the highly correlated variables and also the variables with less information gain. In this list of 30 variables, we had taken the top 20 most important features as suggested by this model.

- Also, comparing the R squared value, we have the highest R squared value for Random Forest which is 0.93.
- Looking at the RSE, we have the lowest RSE for Random Forest Model which is 4.52.
- Taking MSE into consideration, again Random Forest model has the lowest MSE which is 81.75.

  The second best model is XGBoost with a slight less accuracy as Random Forest. According to Random Forest Model, the predictor that can explain the Superconductivity Critical Temperature the most, is std_ThermalConductivity.

As compared to Random Forest and XGBoost, our linear model is definitely not a good model as it gives very low R squared and very high MSE and RSE.

The probable reason why linear regression model did work here, could be because of the high correlation cutoff 0.80. It may have given us a better model if the cutoff was a bit high like 0.90. Also, in EDA we looked and talked over the impact of outliers on models. Well, to my knowledge linear models do not work that well if there are lot of outliers. We were told that the data is already wrangled, and hence it could be such that the outliers could actually hold very important information. In such cases tree based models like Random Forest and Gradient Boosting Models work well with data with so many so called "outliers". Tree based methods are very robust to outliers.

Also, looking again at the histogram of the target variable "critical_temp", we can see the values are right skewed. In such cases log transformations work better than getting rid of outliers. Log transformations were portrayed in the reccommended research paper written by Kam Hamidieh given to us. The relationship of SD Critical Temp and Mean Critical Temp per element shows a non linear relationship, whereas relationship of SD Critical Temp and log(Mean Critical Temperature) per element showed a linear relationship. But in this assignment we did not try log transformations. Therefore, we should always try to transform the data rather than first remove it. Hence this can also be counted as one of the probable reasons for a bad linear model and a good random forest model.

Therefore, from the above discussions, we can conclude that the transformation techniques generally works better than dropping for improving the predictive accuracy of both linear and tree based models. It is very important to treat outliers by either dropping them or transforming them, if multiple linear regression method is used. Random Forests and Gradient Boosting Models are anyway better than linear models most of the time for prediction as they work on decision trees.

11. Variable Identification

After getting our best model, it is very easy to identify the key properties for a superconductor. In other words, the properties that contribute the most to model's performance and highly is listed below:

- std_atomic_mass
- wtd_gmean_fie
- wtd_range_fie
- std_atomic_radius
- wtd_entropy_Density
- wtd_std_Density
- gmean_ElectronAffinity
- wtd_gmean_ElectronAffinity
- wtd_entropy_ElectronAffinity
- wtd_range_ElectronAffinity
- std_ElectronAffinity
- mean_FusionHeat
- wtd_range_FusionHeat
- mean_ThermalConductivity
- wtd_gmean_ThermalConductivity
- wtd_entropy_ThermalConductivity
- wtd_range_ThermalConductivity
- std_ThermalConductivity

- gmean_Valence
- wtd_range_Valence

At every step of model development we can see that the variables related with ThermalConductivity played an important role in predicting critical_temp. Therefore there was an intuition for this feature being selected as the best predictor. According to random forest model, the best feature is std_ThermalConductivity, gmean_Valence and wtd_range_Valence and according to XGBoost model we can say that range_ThermalConductivity and wtd_gmean_ThermalConductivity is the best feature for predicting critical_temp.

12. Conclusion

To conclude we can say that we successfully built 3 models with each type of feature selection method, that is stepwise, embedded and gradient boosting type of feature selection process. As discussed previously random forest and XGBoost models were quite robust with our set of features, whereas the linear model was not. One of the probable reasons could be extreme values for every feature and the other probable reason could be amount of skewness present in the target variable. Also, while filtering the features using correlation, there could be a mistake of removing a highly correlated variable out of the two that can be highly correlated with the target also.

13. References

https://github.com/bhklab/mRMRe/issues/16 (https://github.com/bhklab/mRMRe/issues/16) -> mRMRe

https://machinelearningmastery.com/an-introduction-to-feature-selection/ (https://machinelearningmastery.com/an-introduction-to-feature-selection/) -> feature selection methods

https://machinelearningmastery.com/feature-selection-with-the-caret-r-package/ (https://machinelearningmastery.com/feature-selection-with-the-caret-r-package/) -> feature selection with caret package

https://stackoverflow.com/questions/42932663/findcorrelation-function-in-r (https://stackoverflow.com/questions/42932663/findcorrelation-function-in-r) -> findCorrelation()

https://stackoverflow.com/questions/5234117/how-to-drop-columns-by-name-in-a-data-frame (https://stackoverflow.com/questions/5234117/how-to-drop-columns-by-name-in-a-data-frame) -> remove unwanted columns

https://stackoverflow.com/questions/16194212/how-to-suppress-warnings-globally-in-an-r-script (https://stackoverflow.com/questions/16194212/how-to-suppress-warnings-globally-in-an-r-script) -> suppress warnings in R

http://www.sthda.com/english/articles/37-model-selection-essentials-in-r/154-stepwise-regression-essentials-in-r/ (http://www.sthda.com/english/articles/37-model-selection-essentials-in-r/154-stepwise-regression-essentials-in-r/) -> hybrid stepwise feature selecion

https://datasciencebeginners.com/2018/11/26/functions-and-packages-for-feature-selection-in-r/#Example_3_Using_information_gain_for_variable_selection (https://datasciencebeginners.com/2018/11/26/functions-and-packages-for-feature-selection-in-r/#Example_3_Using_information_gain_for_variable_selection) -> feature selection methods

https://stackoverflow.com/questions/17200114/how-to-split-data-into-training-testing-sets-using-sample-function (https://stackoverflow.com/questions/17200114/how-to-split-data-into-training-testing-sets-using-sample-function) -> splitting data into train and test

https://support.rstudio.com/hc/en-us/community/posts/205198138-packages-installation-problem (https://support.rstudio.com/hc/en-us/community/posts/205198138-packages-installation-problem) -> R package installation problems

https://www.hackerearth.com/practice/machine-learning/machine-learning-algorithms/beginners-tutorial-on-xgboost-parameter-tuning-r/tutorial/ (https://www.hackerearth.com/practice/machine-learning/machine-learning-algorithms/beginners-tutorial-on-xgboost-parameter-tuning-r/tutorial/) -> xgboost

https://www.analyticsvidhya.com/blog/2016/01/xgboost-algorithm-easy-steps/ (https://www.analyticsvidhya.com/blog/2016/01/xgboost-algorithm-easy-steps/) -> xgboost

https://www.kaggle.com/mburakergenc/predictions-with-xgboost-and-linear-regression (https://www.kaggle.com/mburakergenc/predictions-with-xgboost-and-linear-regression) -> xgboost trouble shooting

https://stats.stackexchange.com/questions/278747/how-does-xgboost-do-regression-using-trees/278752 (https://stats.stackexchange.com/questions/278747/how-does-xgboost-do-regression-using-trees/278752) -> xgboost troubleshooting

https://drsimonj.svbtle.com/quick-plot-of-all-variables (https://drsimonj.svbtle.com/quick-plot-of-all-variables) -> plotting histograms

https://stackoverflow.com/questions/45618181/change-font-size-of-titles-from-facet-wrap/45618363 (https://stackoverflow.com/questions/45618181/change-font-size-of-titles-from-facet-wrap/45618363) -> change font size of facet_wrap title

http://www.sthda.com/english/wiki/print.php?id=187 (http://www.sthda.com/english/wiki/print.php?id=187) -> how to color histograms

http://www.sthda.com/english/wiki/ggplot2-quick-correlation-matrix-heatmap-r-software-and-data-visualization (http://www.sthda.com/english/wiki/ggplot2-quick-correlation-matrix-heatmap-r-software-and-data-visualization) -> how to draw heatmap

https://homepage.divms.uiowa.edu/~luke/classes/STAT4580/twonum.html (https://homepage.divms.uiowa.edu/~luke/classes/STAT4580/twonum.html) -> scatterplot-ggplot2

https://stackoverflow.com/questions/52436487/show-multiple-plots-from-ggplot-on-one-page-in-r (https://stackoverflow.com/questions/52436487/show-multiple-plots-from-ggplot-on-one-page-in-r) -> multiple plots in a grid https://r4ds.had.co.nz/exploratory-data-analysis.html (https://r4ds.had.co.nz/exploratory-data-analysis.html) -> explanation of EDA

https://www.investopedia.com/terms/s/standard-error.asp (https://www.investopedia.com/terms/s/standard-error.asp) -> standard error

https://codeburst.io/2-important-statistics-terms-you-need-to-know-in-data-science-skewness-and-kurtosis-388fef94eeaa (https://codeburst.io/2-important-statistics-terms-you-need-to-know-in-data-science-skewness-and-kurtosis-388fef94eeaa) -> what is skewness

https://heartbeat.fritz.ai/how-to-make-your-machine-learning-models-robust-to-outliers-44d404067d07 (https://heartbeat.fritz.ai/how-to-make-your-machine-learning-models-robust-to-outliers-44d404067d07) -> boxplots and outliers and model prediction

https://datascience.stackexchange.com/questions/24452/in-supervised-learning-why-is-it-bad-to-have-correlated-features (https://datascience.stackexchange.com/questions/24452/in-supervised-learning-why-is-it-bad-to-have-correlated-features) -> why we have to remove highly correlated variables

https://www.statisticshowto.datasciencecentral.com/probability-and-statistics/f-statistic-value-test/ (https://www.statisticshowto.datasciencecentral.com/probability-and-statistics/f-statistic-value-test/) -> F statistic

http://www.statistics4u.info/fundstat_eng/cc_varsel_mallowscp.html (http://www.statistics4u.info/fundstat_eng/cc_varsel_mallowscp.html) -> Mallow's C_p

https://www.statisticshowto.datasciencecentral.com/bayesian-information-criterion/ (https://www.statisticshowto.datasciencecentral.com/bayesian-information-criterion/) and https://medium.com/@analyttica/what-is-bayesian-information-criterion-bic-b3396a894be6 (https://medium.com/@analyttica/what-is-bayesian-information-criterion-bic-b3396a894be6) -> BIC

https://stats.stackexchange.com/questions/80157/relationship-between-correlation-and-multicollinearity (https://stats.stackexchange.com/questions/80157/relationship-between-correlation-and-multicollinearity) -> VIF()

https://www.researchgate.net/post/Whats_the_difference_between_correlation_and_VIF (https://www.researchgate.net/post/Whats_the_difference_between_correlation_and_VIF) -> difference between correlation coefficient and multicollinearity

https://towardsdatascience.com/feature-selection-correlation-and-p-value-da8921bfb3cf (https://towardsdatascience.com/feature-selection-correlation-and-p-value-da8921bfb3cf)

https://medium.com/@cxu24/common-methods-for-feature-selection-you-should-know-2346847fdf31 (https://medium.com/@cxu24/common-methods-for-feature-selection-you-should-know-2346847fdf31)

https://stats.stackexchange.com/questions/154825/what-to-conclude-from-this-lasso-plot-glmnet (https://stats.stackexchange.com/questions/154825/what-to-conclude-from-this-lasso-plot-glmnet)

https://rstudio-pubs-static.s3.amazonaws.com/54576_142b58255f8944c990c5663290d28517.html (https://rstudio-pubs-static.s3.amazonaws.com/54576_142b58255f8944c990c5663290d28517.html)

https://www.distilnetworks.com/glossary/term/random-forest-model/ (https://www.distilnetworks.com/glossary/term/random-forest-model/)

https://heartbeat.fritz.ai/how-to-make-your-machine-learning-models-robust-to-outliers-44d404067d07 (https://heartbeat.fritz.ai/how-to-make-your-machine-learning-models-robust-to-outliers-44d404067d07)

Also, some of the materials from the tutorials, reccommended research paper and previous semester modelling assignment was also included in this assignment.

In [ ]: