29595916_FIT5149_Ass1

September 16, 2019

1 FIT5149 S2 2019 Assessment 1: Choosing and Explaining Likely Chemical properties for semiconductors

Student information - Family Name: Alchetti

- Given Name: Pradnya
- Student ID: 29595916
- Student email: palc0001@student.monash.edu

Programming Language: R 3.6.0 in Jupyter Notebook
R Libraries used: - caret

- tidyverse
- RColorBrewer
- scales
- grid
- gridExtra
- leaps
- glmnet
- xgboost
- broom

1.1 Table of Contents

1.2 1. Introduction

Superconducting materials are the materials that conduct current with zero electrical resistance which occurs at absolute low temperatures. This low temperature is known as the critical temperature. At this temperature the transition of the material into a superconductor is so sudden that it appears to be a transition into a different phase of matter. Superconductors have many prominent applications such as superconducting coils and Magnetic Resonance Imaging (MRI) used in the healthcare domain for internal body imaging.

Predicting the critical temperature of a superconductor is still an open issue. In this notebook, we will analyze the superconductor data from the Superconducting Material Database maintained by Japan's National Institute for Materials Science (NIMS) and predict the critical temperature of the material given the chemial properties. This analysis will help the researchers in synthesizing superconducting materials.

In the first part we analyze the data through Exploratory Data Analysis wherein we will understand the nature of the data, distribution of different variables affecting the critical temperature and the correlation between the variables.

After EDA, we will implement various variable selection methods to identify and select important variables. At the end, we will build different statistical models for predicting critical temperature given the chemical properties of the material and compare the models based on the accuracy parameters and select the most appropriate model.

The dataset consists of:

- train.csv
- unique m.csv

1.2.1 1.1 Libraries

```
In [1]: require(leaps)
        require(caret)
        require(leaps)
        require(broom)
        library(caret)
        library(tidyverse)
        library(RColorBrewer)
        library(scales)
        library(grid)
        library(gridExtra)
        library(leaps)
        library(glmnet)
        library(xgboost)
        library(broom)
```

```
Loading required package: leaps
Loading required package: caret
Loading required package: lattice
Loading required package: ggplot2
Registered S3 methods overwritten by 'ggplot2':
  method         from
```

```
  [.quosures     rlang
  c.quosures     rlang
  print.quosures rlang
Loading required package: broom
 Attaching packages  tidyverse 1.2.1
 tibble  2.1.2      purrr   0.3.2
 tidyr   0.8.3      dplyr   0.8.1
 readr   1.3.1      stringr 1.4.0
 tibble  2.1.2      forcats 0.4.0
 Conflicts  tidyverse_conflicts()
 dplyr::filter() masks stats::filter()
 dplyr::lag()    masks stats::lag()
 purrr::lift()   masks caret::lift()

Attaching package: scales

The following object is masked from package:purrr:

    discard

The following object is masked from package:readr:

    col_factor


Attaching package: gridExtra

The following object is masked from package:dplyr:

    combine

Loading required package: Matrix

Attaching package: Matrix

The following object is masked from package:tidyr:

    expand

Loading required package: foreach

Attaching package: foreach

The following objects are masked from package:purrr:

    accumulate, when

Loaded glmnet 2.0-18
```

```
Attaching package: xgboost

The following object is masked from package:dplyr:

    slice
```

1.2.2 1.2 Loading the dataset

```
In [2]: df_train <- read.csv("train.csv")
```

1.3 2. Data Exploration

Dimensions of the datsset

```
In [3]: dim(df_train)
```

1. 21263 2. 82

- The superconductor dataset consists of 82 columns including the target variable and 21263 rows.

The overall structure of the data is as follows:

```
In [4]: str(df_train)
```

```
'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 2.25 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 ...
```

All the variables are numerial continuous except the variables i.e., number_of_elements, range_atomic_radius and range_Valence which appear to be numerical discrete. The target variable "critical_temp" is numerical continuous which indicates that we will be predicting continuous variable using statistical models.

```
In [5]: head(df_train)
```

A data.frame: 6 Œ 82

number_of_elements <int>	mean_atomic_mass <dbl>	wtd_mean_atomic_mass <dbl>	gmean_atomi <dbl>
4	88.94447	57.86269	66.36159
5	92.72921	58.51842	73.13279
4	88.94447	57.88524	66.36159
4	88.94447	57.87397	66.36159
4	88.94447	57.84014	66.36159
4	88.94447	57.79504	66.36159

Each row in the given superconductor dataset indicates a separate superconducting material with different number of elements and chemical properties

```
In [6]: summary(df_train)
```

```
 number_of_elements mean_atomic_mass  wtd_mean_atomic_mass gmean_atomic_mass
 Min.   :1.000      Min.   :  6.941   Min.   :  6.423      Min.   :  5.321
 1st Qu.:3.000      1st Qu.: 72.458   1st Qu.: 52.144      1st Qu.: 58.041
 Median :4.000      Median : 84.923   Median : 60.697      Median : 66.362
 Mean   :4.115      Mean   : 87.558   Mean   : 72.988      Mean   : 71.291
 3rd Qu.:5.000      3rd Qu.:100.404   3rd Qu.: 86.104      3rd Qu.: 78.117
 Max.   :9.000      Max.   :208.980   Max.   :208.980      Max.   :208.980
 wtd_gmean_atomic_mass entropy_atomic_mass wtd_entropy_atomic_mass
 Min.   :  1.961       Min.   :0.0000      Min.   :0.0000
 1st Qu.: 35.249       1st Qu.:0.9667      1st Qu.:0.7754
 Median : 39.918       Median :1.1995      Median :1.1468
 Mean   : 58.540       Mean   :1.1656      Mean   :1.0639
```

```
3rd Qu.: 73.113       3rd Qu.:1.4445      3rd Qu.:1.3594
Max.   :208.980       Max.   :1.9838      Max.   :1.9582
range_atomic_mass wtd_range_atomic_mass std_atomic_mass  wtd_std_atomic_mass
Min.   :  0.00    Min.   :  0.00        Min.   :  0.00   Min.   :  0.00
1st Qu.: 78.51    1st Qu.: 16.82        1st Qu.: 32.89   1st Qu.: 28.54
Median :122.91    Median : 26.64        Median : 45.12   Median : 44.29
Mean   :115.60    Mean   : 33.23        Mean   : 44.39   Mean   : 41.45
3rd Qu.:154.12    3rd Qu.: 38.36        3rd Qu.: 59.32   3rd Qu.: 53.63
Max.   :207.97    Max.   :205.59        Max.   :101.02   Max.   :101.02
   mean_fie       wtd_mean_fie      gmean_fie      wtd_gmean_fie
Min.   : 375.5   Min.   : 375.5   Min.   : 375.5   Min.   : 375.5
1st Qu.: 723.7   1st Qu.: 738.9   1st Qu.: 692.5   1st Qu.: 720.1
Median : 764.9   Median : 890.0   Median : 728.0   Median : 856.2
Mean   : 769.6   Mean   : 870.4   Mean   : 737.5   Mean   : 832.8
3rd Qu.: 796.3   3rd Qu.:1004.1   3rd Qu.: 765.7   3rd Qu.: 937.6
Max.   :1313.1   Max.   :1348.0   Max.   :1313.1   Max.   :1327.6
 entropy_fie    wtd_entropy_fie    range_fie      wtd_range_fie
Min.   :0.000   Min.   :0.0000   Min.   :   0.0   Min.   :   0.0
1st Qu.:1.086   1st Qu.:0.7538   1st Qu.: 262.4   1st Qu.: 291.1
Median :1.356   Median :0.9168   Median : 764.1   Median : 510.4
Mean   :1.299   Mean   :0.9267   Mean   : 572.2   Mean   : 483.5
3rd Qu.:1.551   3rd Qu.:1.0618   3rd Qu.: 810.6   3rd Qu.: 690.7
Max.   :2.158   Max.   :2.0386   Max.   :1304.5   Max.   :1251.9
   std_fie       wtd_std_fie     mean_atomic_radius wtd_mean_atomic_radius
Min.   :  0.0   Min.   :  0.00   Min.   : 48.0      Min.   : 48.0
1st Qu.:114.1   1st Qu.: 92.99   1st Qu.:149.3      1st Qu.:112.1
Median :266.4   Median :258.45   Median :160.2      Median :126.0
Mean   :215.6   Mean   :224.05   Mean   :158.0      Mean   :134.7
3rd Qu.:297.7   3rd Qu.:342.66   3rd Qu.:169.9      3rd Qu.:158.3
Max.   :499.7   Max.   :479.16   Max.   :298.0      Max.   :298.0
gmean_atomic_radius wtd_gmean_atomic_radius entropy_atomic_radius
Min.   : 48.0       Min.   : 48.00          Min.   :0.000
1st Qu.:133.5       1st Qu.: 89.21          1st Qu.:1.066
Median :142.8       Median :113.18          Median :1.331
Mean   :144.4       Mean   :120.99          Mean   :1.268
3rd Qu.:155.9       3rd Qu.:150.99          3rd Qu.:1.512
Max.   :298.0       Max.   :298.00          Max.   :2.142
wtd_entropy_atomic_radius range_atomic_radius wtd_range_atomic_radius
Min.   :0.0000            Min.   :  0.0       Min.   :  0.00
1st Qu.:0.8522            1st Qu.: 80.0       1st Qu.: 28.60
Median :1.2429            Median :171.0       Median : 43.00
Mean   :1.1311            Mean   :139.3       Mean   : 51.37
3rd Qu.:1.4257            3rd Qu.:205.0       3rd Qu.: 60.22
Max.   :1.9037            Max.   :256.0       Max.   :240.16
std_atomic_radius wtd_std_atomic_radius  mean_Density
Min.   :  0.00    Min.   : 0.00         Min.   :    1.429
1st Qu.: 35.11    1st Qu.:32.02         1st Qu.: 4513.500
Median : 58.66    Median :59.93         Median : 5329.086
```

```
Mean   : 51.60    Mean   :52.34         Mean   : 6111.465
3rd Qu.: 69.42    3rd Qu.:73.78         3rd Qu.: 6728.000
Max.   :115.50    Max.   :97.14         Max.   :22590.000
wtd_mean_Density    gmean_Density       wtd_gmean_Density   entropy_Density
Min.   :    1.429   Min.   :    1.429   Min.   :    0.686   Min.   :0.000
1st Qu.: 2999.158   1st Qu.:  883.117   1st Qu.:   66.747   1st Qu.:0.914
Median : 4303.422   Median : 1339.975   Median : 1515.365   Median :1.091
Mean   : 5267.189   Mean   : 3460.692   Mean   : 3117.241   Mean   :1.072
3rd Qu.: 6416.333   3rd Qu.: 5794.965   3rd Qu.: 5766.015   3rd Qu.:1.324
Max.   :22590.000   Max.   :22590.000   Max.   :22590.000   Max.   :1.954
wtd_entropy_Density range_Density   wtd_range_Density  std_Density
Min.   :0.0000      Min.   :    0   Min.   :    0     Min.   :    0
1st Qu.:0.6887      1st Qu.: 6648   1st Qu.: 1657     1st Qu.: 2819
Median :0.8827      Median : 8959   Median : 2083     Median : 3302
Mean   :0.8560      Mean   : 8665   Mean   : 2903     Mean   : 3417
3rd Qu.:1.0809      3rd Qu.: 9779   3rd Qu.: 3409     3rd Qu.: 4004
Max.   :1.7034      Max.   :22589   Max.   :22434     Max.   :10724
wtd_std_Density mean_ElectronAffinity wtd_mean_ElectronAffinity
Min.   :    0   Min.   :  1.50        Min.   :  1.50
1st Qu.: 2564   1st Qu.: 62.09        1st Qu.: 73.35
Median : 3626   Median : 73.10        Median :102.86
Mean   : 3319   Mean   : 76.88        Mean   : 92.72
3rd Qu.: 3959   3rd Qu.: 85.50        3rd Qu.:110.74
Max.   :10411   Max.   :326.10        Max.   :326.10
gmean_ElectronAffinity wtd_gmean_ElectronAffinity entropy_ElectronAffinity
Min.   :  1.50         Min.   :  1.50             Min.   :0.0000
1st Qu.: 33.70         1st Qu.: 50.77             1st Qu.:0.8906
Median : 51.47         Median : 73.17             Median :1.1383
Mean   : 54.36         Mean   : 72.42             Mean   :1.0702
3rd Qu.: 67.51         3rd Qu.: 89.98             3rd Qu.:1.3459
Max.   :326.10         Max.   :326.10             Max.   :1.7677
wtd_entropy_ElectronAffinity range_ElectronAffinity wtd_range_ElectronAffinity
Min.   :0.0000               Min.   :  0.0          Min.   :  0.00
1st Qu.:0.6607               1st Qu.: 86.7          1st Qu.: 34.04
Median :0.7812               Median :127.0          Median : 71.16
Mean   :0.7708               Mean   :120.7          Mean   : 59.33
3rd Qu.:0.8775               3rd Qu.:138.6          3rd Qu.: 76.71
Max.   :1.6754               Max.   :349.0          Max.   :218.70
std_ElectronAffinity wtd_std_ElectronAffinity mean_FusionHeat
Min.   :  0.00       Min.   :  0.00           Min.   :  0.222
1st Qu.: 38.37       1st Qu.: 33.44           1st Qu.:  7.589
Median : 51.13       Median : 48.03           Median :  9.304
Mean   : 48.91       Mean   : 44.41           Mean   : 14.296
3rd Qu.: 56.22       3rd Qu.: 53.32           3rd Qu.: 17.114
Max.   :162.90       Max.   :169.08           Max.   :105.000
wtd_mean_FusionHeat gmean_FusionHeat  wtd_gmean_FusionHeat entropy_FusionHeat
Min.   :  0.222     Min.   :  0.222   Min.   :  0.222      Min.   :0.0000
1st Qu.:  5.033     1st Qu.:  4.110   1st Qu.:  1.322      1st Qu.:0.8333
```

```
 Median :  8.331     Median :  5.253   Median :  4.930      Median :1.1121
 Mean   : 13.848     Mean   : 10.137   Mean   : 10.141      Mean   :1.0933
 3rd Qu.: 18.514     3rd Qu.: 13.600   3rd Qu.: 16.429      3rd Qu.:1.3781
 Max.   :105.000     Max.   :105.000   Max.   :105.000      Max.   :2.0344
 wtd_entropy_FusionHeat range_FusionHeat wtd_range_FusionHeat std_FusionHeat
 Min.   :0.0000         Min.   :  0.00   Min.   :  0.000      Min.   : 0.000
 1st Qu.:0.6727         1st Qu.: 12.88   1st Qu.:  2.329      1st Qu.: 4.261
 Median :0.9950         Median : 12.88   Median :  3.436      Median : 4.948
 Mean   :0.9141         Mean   : 21.14   Mean   :  8.219      Mean   : 8.323
 3rd Qu.:1.1574         3rd Qu.: 23.20   3rd Qu.: 10.499      3rd Qu.: 9.041
 Max.   :1.7472         Max.   :104.78   Max.   :102.675      Max.   :51.635
 wtd_std_FusionHeat mean_ThermalConductivity wtd_mean_ThermalConductivity
 Min.   : 0.000     Min.   :  0.0266         Min.   :  0.0266
 1st Qu.: 4.603     1st Qu.: 61.0000         1st Qu.: 54.1810
 Median : 5.501     Median : 96.5044         Median : 73.3333
 Mean   : 7.718     Mean   : 89.7069         Mean   : 81.5491
 3rd Qu.: 8.018     3rd Qu.:111.0053         3rd Qu.: 99.0629
 Max.   :51.680     Max.   :332.5000         Max.   :406.9600
 gmean_ThermalConductivity wtd_gmean_ThermalConductivity
 Min.   :  0.0266          Min.   :  0.023
 1st Qu.:  8.3398          1st Qu.:  1.087
 Median : 14.2876          Median :  6.096
 Mean   : 29.8417          Mean   : 27.308
 3rd Qu.: 42.3713          3rd Qu.: 47.308
 Max.   :317.8836          Max.   :376.033
 entropy_ThermalConductivity wtd_entropy_ThermalConductivity
 Min.   :0.0000              Min.   :0.0000
 1st Qu.:0.4578              1st Qu.:0.2507
 Median :0.7387              Median :0.5458
 Mean   :0.7276              Mean   :0.5400
 3rd Qu.:0.9622              3rd Qu.:0.7774
 Max.   :1.6340              Max.   :1.6130
 range_ThermalConductivity wtd_range_ThermalConductivity
 Min.   :  0.00            Min.   :  0.00
 1st Qu.: 86.38            1st Qu.: 29.35
 Median :399.80            Median : 56.56
 Mean   :250.89            Mean   : 62.03
 3rd Qu.:399.97            3rd Qu.: 91.87
 Max.   :429.97            Max.   :401.44
 std_ThermalConductivity wtd_std_ThermalConductivity  mean_Valence
 Min.   :  0.00          Min.   :  0.00              Min.   :1.000
 1st Qu.: 37.93          1st Qu.: 31.99              1st Qu.:2.333
 Median :135.76          Median :113.56              Median :2.833
 Mean   : 98.94          Mean   : 96.23              Mean   :3.198
 3rd Qu.:153.81          3rd Qu.:162.71              3rd Qu.:4.000
 Max.   :214.99          Max.   :213.30              Max.   :7.000
 wtd_mean_Valence gmean_Valence   wtd_gmean_Valence entropy_Valence
 Min.   :1.000    Min.   :1.000   Min.   :1.000     Min.   :0.000
```

```
1st Qu.:2.117    1st Qu.:2.280   1st Qu.:2.091     1st Qu.:1.061
Median :2.618    Median :2.615   Median :2.434     Median :1.369
Mean   :3.153    Mean   :3.057   Mean   :3.056     Mean   :1.296
3rd Qu.:4.026    3rd Qu.:3.728   3rd Qu.:3.915     3rd Qu.:1.589
Max.   :7.000    Max.   :7.000   Max.   :7.000     Max.   :2.142
wtd_entropy_Valence range_Valence   wtd_range_Valence  std_Valence
Min.   :0.0000      Min.   :0.000   Min.   :0.0000    Min.   :0.0000
1st Qu.:0.7757      1st Qu.:1.000   1st Qu.:0.9215    1st Qu.:0.4518
Median :1.1665      Median :2.000   Median :1.0631    Median :0.8000
Mean   :1.0528      Mean   :2.041   Mean   :1.4830    Mean   :0.8393
3rd Qu.:1.3308      3rd Qu.:3.000   3rd Qu.:1.9184    3rd Qu.:1.2000
Max.   :1.9497      Max.   :6.000   Max.   :6.9922    Max.   :3.0000
wtd_std_Valence  critical_temp
Min.   :0.0000   Min.   :  0.00021
1st Qu.:0.3069   1st Qu.:  5.36500
Median :0.5000   Median : 20.00000
Mean   :0.6740   Mean   : 34.42122
3rd Qu.:1.0204   3rd Qu.: 63.00000
Max.   :3.0000   Max.   :185.00000
```

From the above summary statistics we can observe the following:

- The number of elements in a particular superconducting material can be between 1 to 9
- Many variables might contain outliers. This can be observed in variables such as mean_atomic_mass, wtd_mean_atomic_mass, gmean_atomic_mass, wtd_range_atomic_mass, wtd_range_atomic_radius where they have very high max value as compared to their median and 3rd quartile values. This behaviour can be observed more closely with the help of boxplots.
- For the target variable "critical_temp" lies between 0.00021-185K which indicates that the materials can attain superconductivity not only at very low temperatures but can occur at considerably high temperatures [1].

The given superconductor dataset has 10 features each of the following properties:

- Atomic Mass
- First Ionization Energy
- Atomic Radius
- Density
- Electron Affinity
- Fusion Heat
- Thermal Conductivity
- Valence

Atomic Mass, Atomic Radius, Density and Fusion Heat may have a higher contribution towards predicting the critical temperature [2]. Therefore, we first analyze features of these properties.

Each chemical property has 10 features each viz., mean, weighted mean, geometric mean, weighted geometric mean, entropy, weighted entropy, range, weighted range, standard deviation and weighted standard deviation. We will be analyzing each of these features and their effect on critical temperature

1.3.1 2.1 Analysis of Atomic Mass features

```
In [7]: # Extracting 10 features corresponding to atomic mass
        atomic_mass_df <- df_train[,1:11]
```

```
In [8]: attach(atomic_mass_df)
        head(atomic_mass_df)
```

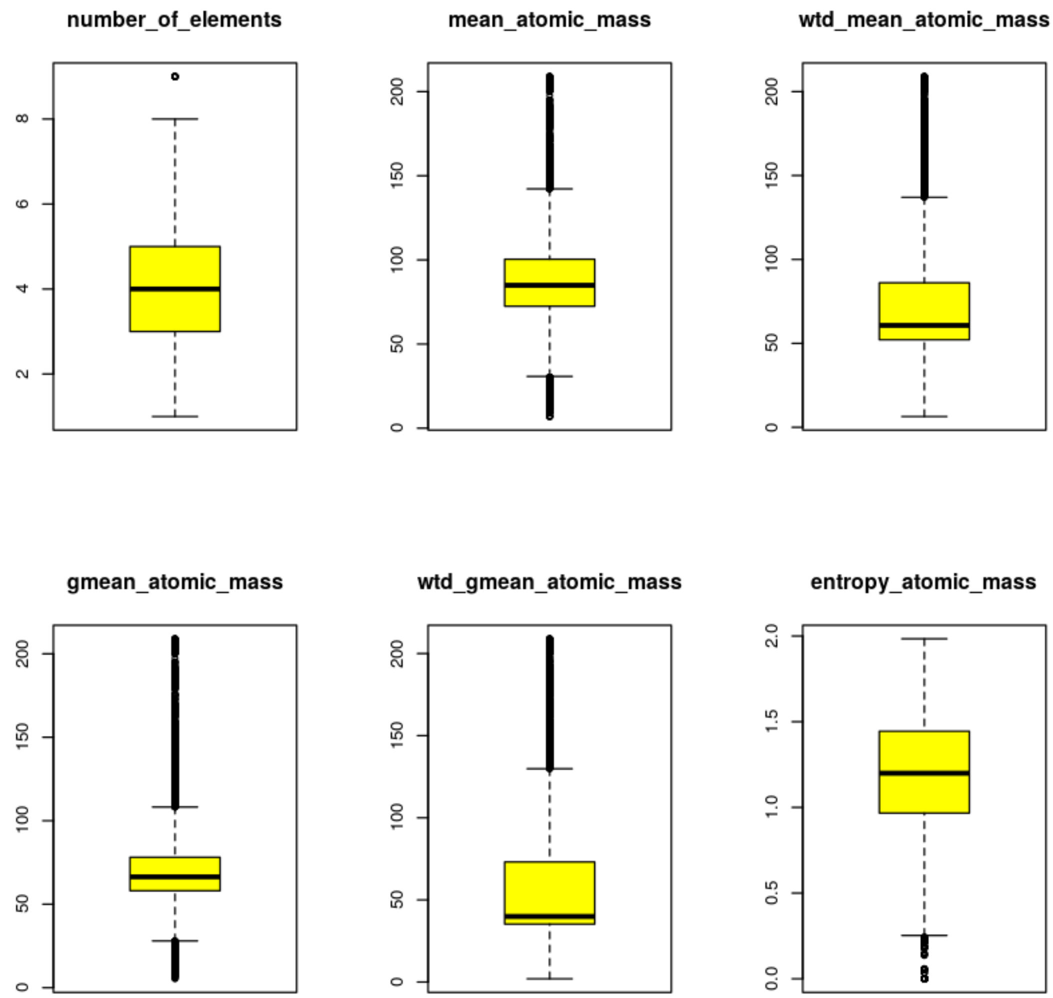
A data.frame: 6 Œ 11

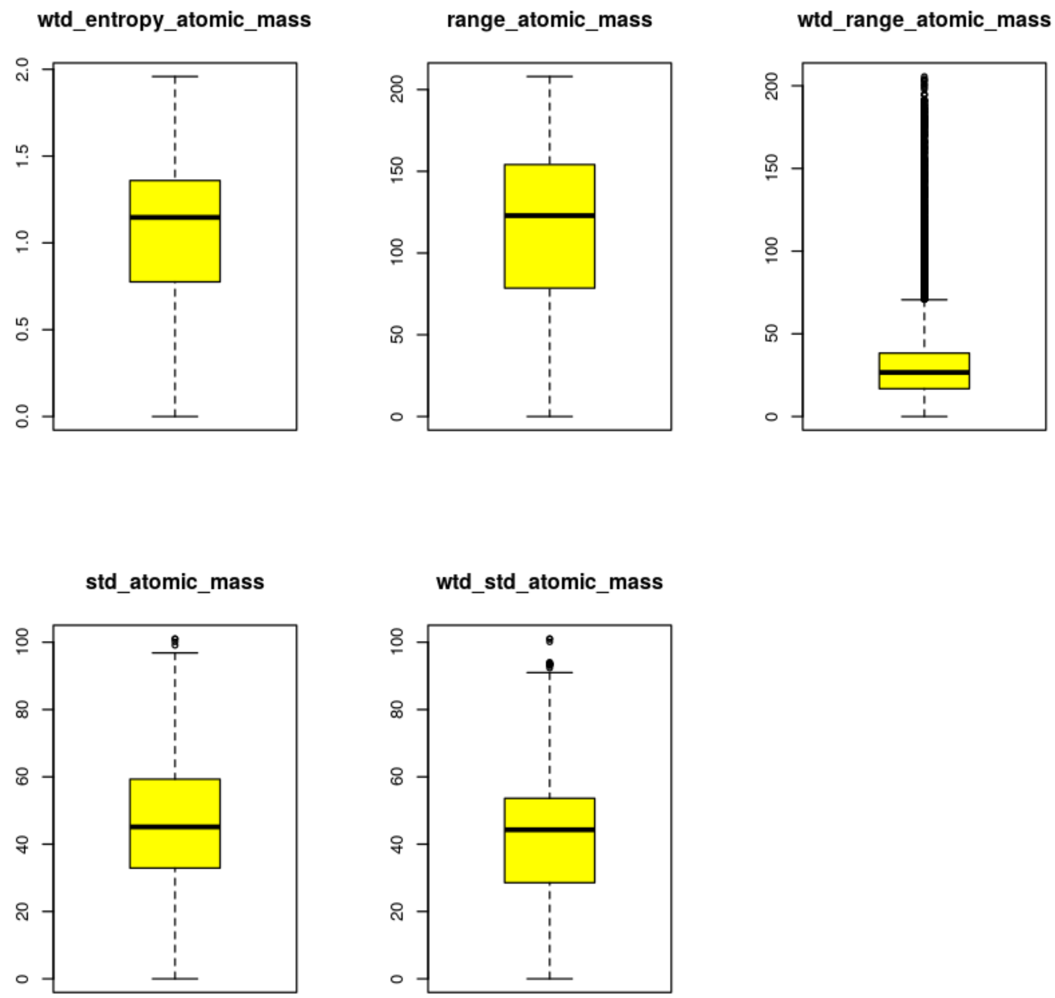
number_of_elements	mean_atomic_mass	wtd_mean_atomic_mass	gmean_atomi
<int>	<dbl>	<dbl>	<dbl>
4	88.94447	57.86269	66.36159
5	92.72921	58.51842	73.13279
4	88.94447	57.88524	66.36159
4	88.94447	57.87397	66.36159
4	88.94447	57.84014	66.36159
4	88.94447	57.79504	66.36159

The chemical property Atomic Mass has 10 features which are all numerical continuous variables. Here we will also be considering number of elements along with the atomic mass properties for analysis

Lets check the variable distributions with the help of box plots

```
In [9]: par(mfrow = c(2,3))
        for (i in 1:(length(atomic_mass_df))) {
        boxplot(atomic_mass_df[,i], main = names(atomic_mass_df[i]), type="l", col = 'yellow')
        }
```

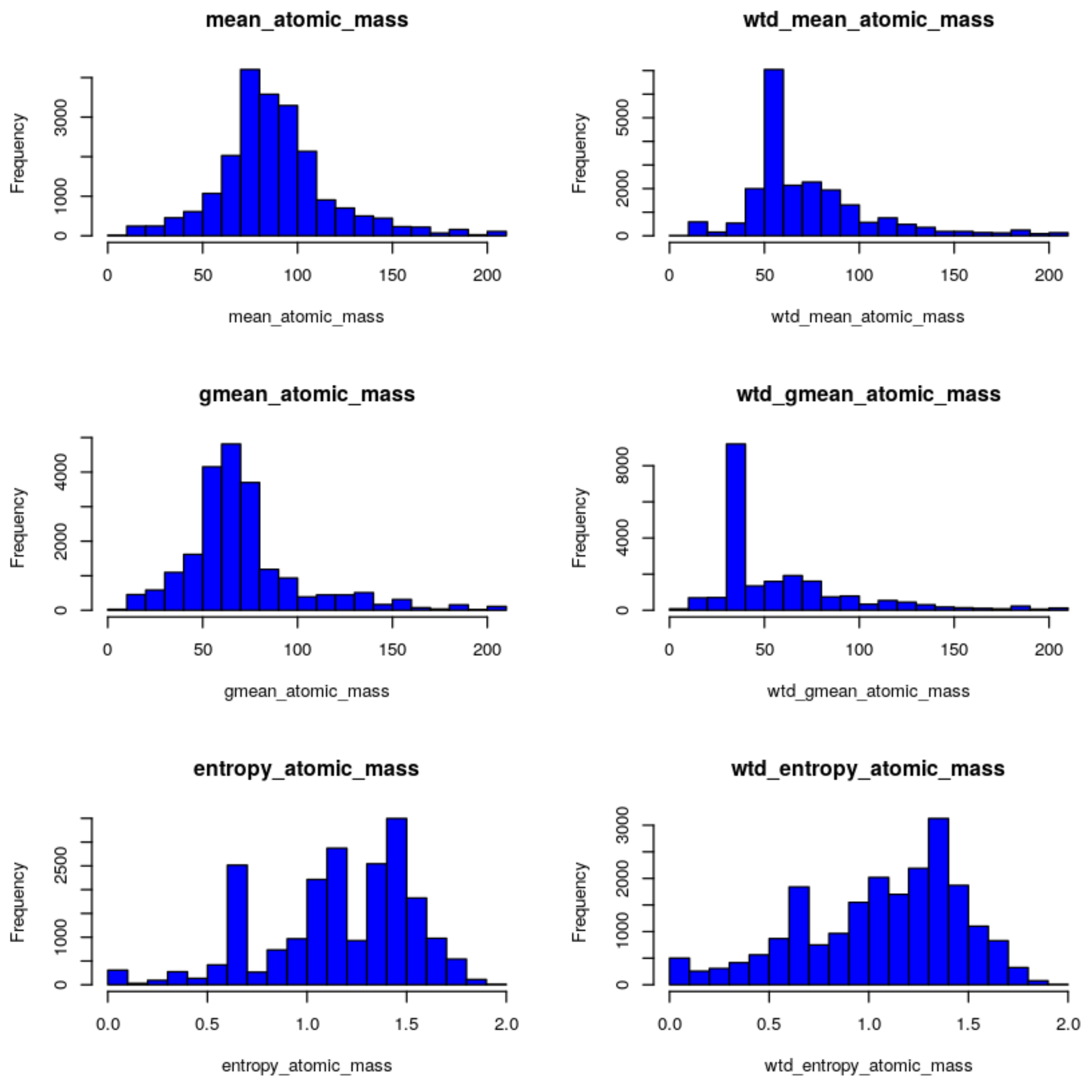

number_of_elements
2
4
6
8
mean_atomic_mass
0
50
100
150
200
wtd_mean_atomic_mass
0
50
100
150
200
gmean_atomic_mass
0
50
100
150
200
wtd_gmean_atomic_mass
0
50
100
150
200
entropy_atomic_mass
0.0
0.5
1.0
1.5
2.0

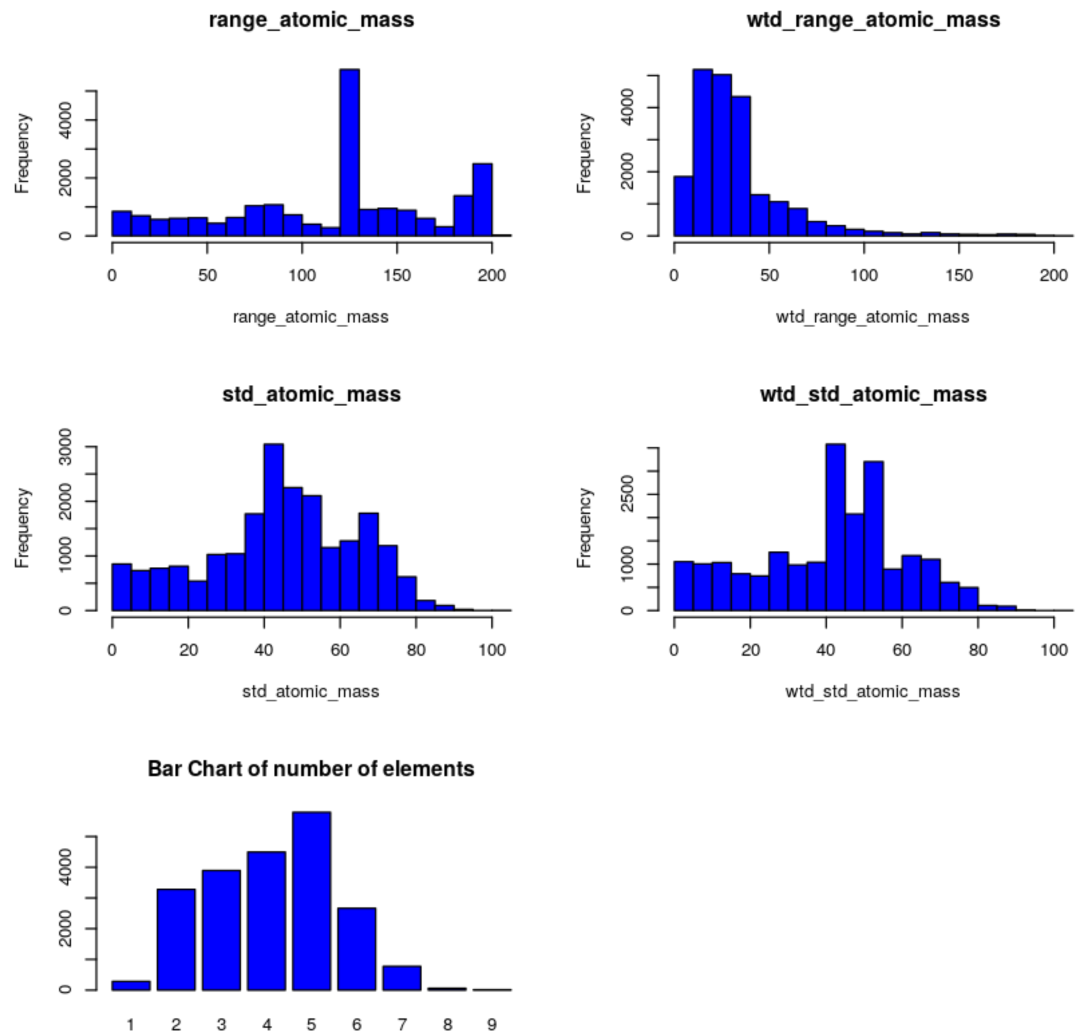


- Like we observed in the summary statistics and from the above box plots, it is evident that mean_atomic_mass, wtd_mean_atomic_mass, gmean_atomic_mass, wtd_gmean_atomic_mass and wtd_range_atomic_mass contain large amount of outliers.
- wtd_gmean_atomic_mass has less variation in the data

We can understand the variable distributions using histograms and bar charts

In [10]:
```
par(mfrow = c(3,2))
atm_col <- colnames(atomic_mass_df)
for (i in 2:(length(atm_col))) {
    hist(atomic_mass_df[,i], main=names(atomic_mass_df[i]), xlab = names(atomic_mass_c
}
plot(as.factor(number_of_elements),main="Bar Chart of number of elements", col="blue"
```


mean_atomic_mass
Frequency
0
1000
3000
0
50
100
150
200
mean_atomic_mass
wtd_mean_atomic_mass
Frequency
0
2000
5000
0
50
100
150
200
wtd_mean_atomic_mass
gmean_atomic_mass
Frequency
0
2000
4000
0
50
100
150
200
gmean_atomic_mass
wtd_gmean_atomic_mass
Frequency
0
4000
8000
0
50
100
150
200
wtd_gmean_atomic_mass
entropy_atomic_mass
Frequency
0
1000
2500
0.0
0.5
1.0
1.5
2.0
entropy_atomic_mass
wtd_entropy_atomic_mass
Frequency
0
1000
2000
3000
0.0
0.5
1.0
1.5
2.0
wtd_entropy_atomic_mass

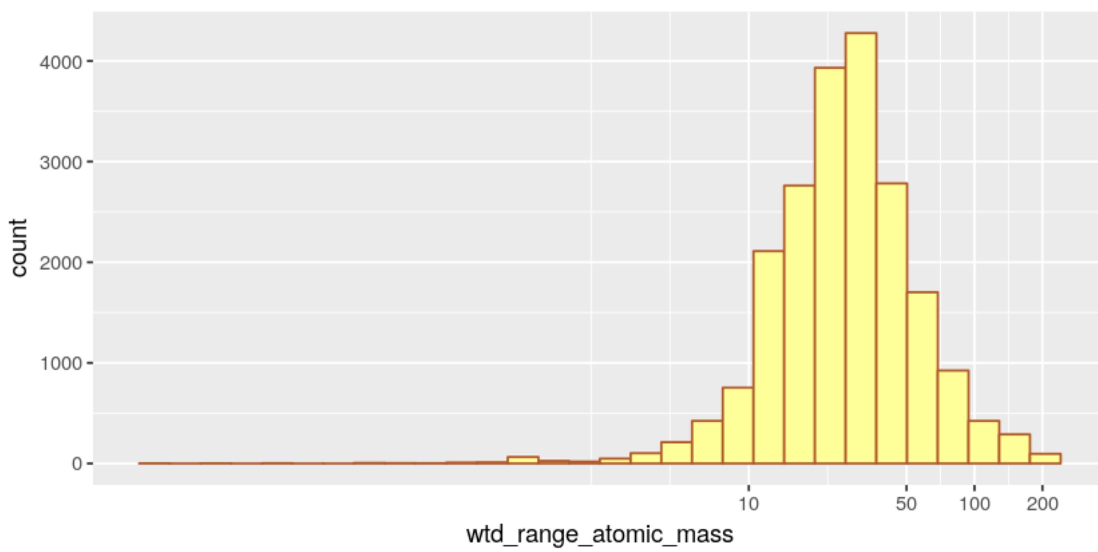


- From the above distributions, we can observe that mean_atomic_mass has a normal distribution
- Most of the materials have 5 number of elements involved in it.
- Most of the materials have gmean_atomic_mass between 50-80
- There is a spike observed in wtd_gmean_atomic_mass which indicates that more than 8000 of the superconducting materials have wtd_gmean_atomic_mass around 30
- wtd_range_atomic_mass distribution is right skewed with most of the materials having the range between 0-50.

Replotting wtd_range_atomic_mass to see if the log scale has normal distribution

In [11]:
```
# Set some colours using Colorbrewer
gg.colour <- brewer.pal(12,"Paired")[12]
gg.fill <- brewer.pal(12,"Paired")[11]

# Re-plot some of the charts using log scales to counteract the skew
p3 <- ggplot(aes(x=wtd_range_atomic_mass), data=atomic_mass_df) +
      geom_histogram(bins=30, colour=gg.colour, fill=gg.fill) +
      scale_x_log10(labels=comma,breaks=c(10,50,100,200,500,1000))
grid.arrange(p3, heights=unit(0.5, "npc"), nrow = 1)
```

```
Warning message:
Transformation introduced infinite values in continuous x-axisWarning message:
Removed 285 rows containing non-finite values (stat_bin).
```



The log of wtd_range_atomic_mass is not quite normal as it demonstartes a normal curve between 5-200 units but contains many outliers below 3 units.

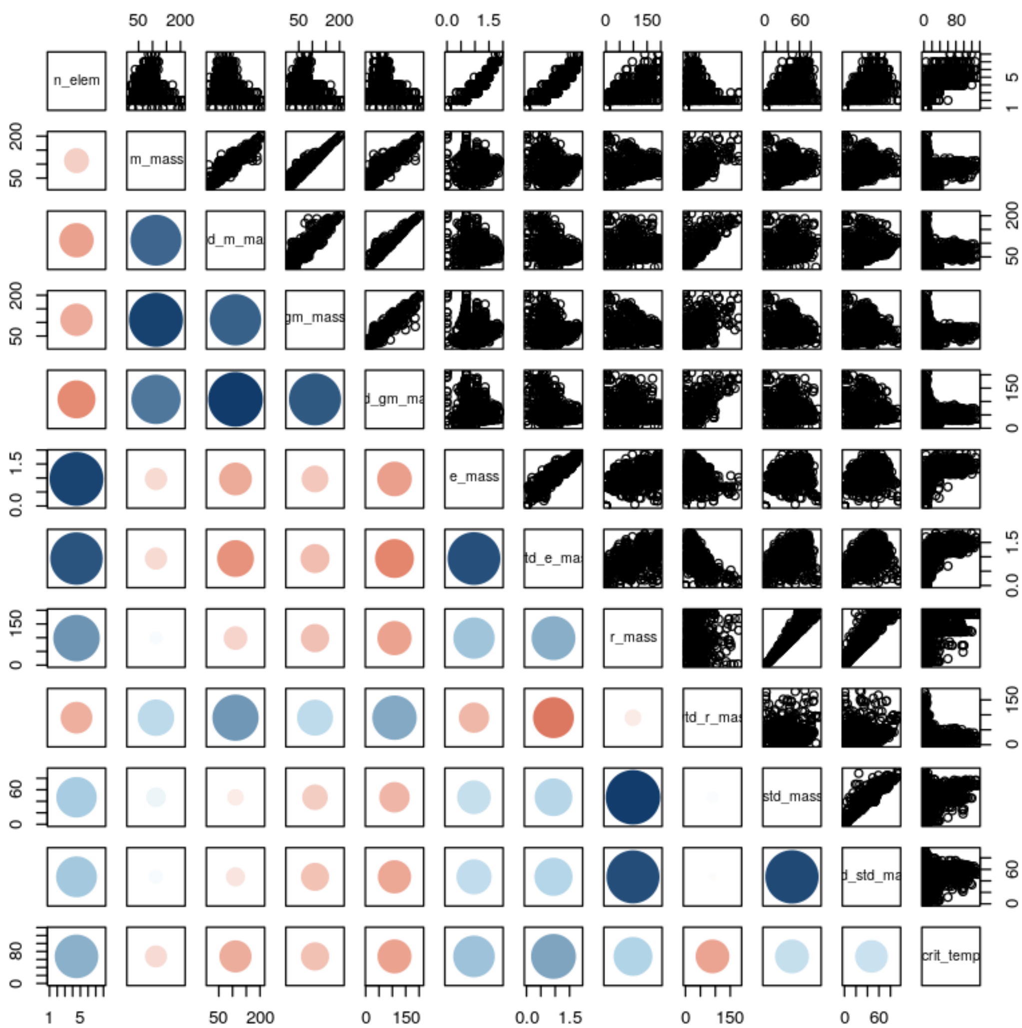
In [12]:
```
#function for correlation matrix
# DIY correlation plot
# http://stackoverflow.com/questions/31709982/how-to-plot-in-r-a-correlogram-on-top-o
# there's some truth to the quote that modern programming is often stitching together

colorRange <- c('#69091e', '#e37f65', 'white', '#aed2e6', '#042f60')
## colorRamp() returns a function which takes as an argument a number
## on [0,1] and returns a color in the gradient in colorRange
myColorRampFunc <- colorRamp(colorRange)

panel.cor <- function(w, z, ...) {
    correlation <- cor(w, z)

    ## because the func needs [0,1] and cor gives [-1,1], we need to shift and scale
    col <- rgb(myColorRampFunc((1 + correlation) / 2 ) / 255 )

    ## square it to avoid visual bias due to "area vs diameter"
    radius <- sqrt(abs(correlation))
    radians <- seq(0, 2*pi, len = 50) # 50 is arbitrary
    x <- radius * cos(radians)
    y <- radius * sin(radians)
    ## make them full loops
    x <- c(x, tail(x,n=1))
    y <- c(y, tail(y,n=1))

    ## trick: "don't create a new plot" thing by following the
    ## advice here: http://www.r-bloggers.com/multiple-y-axis-in-a-r-plot/
    ## This allows
    par(new=TRUE)
    plot(0, type='n', xlim=c(-1,1), ylim=c(-1,1), axes=FALSE, asp=1)
    polygon(x, y, border=col, col=col)
}

# usage e.g.:
# pairs(mtcars, upper.panel = panel.cor)
```

In [13]:
```
# Renaming the variables for correlation matrix
atomic_mass_df <- atomic_mass_df %>%
  rename(
    n_elem = number_of_elements,
    m_mass = mean_atomic_mass,
    wtd_m_mass = wtd_mean_atomic_mass,
    gm_mass = gmean_atomic_mass,
    wtd_gm_mass = wtd_gmean_atomic_mass,
    e_mass = entropy_atomic_mass,
```

```
                wtd_e_mass = wtd_entropy_atomic_mass,
                r_mass = range_atomic_mass,
                wtd_r_mass = wtd_range_atomic_mass,
                std_mass = std_atomic_mass,
                wtd_std_mass = wtd_std_atomic_mass
                )
```

Analyzing the relationship of the atomic mass features with the critical tempearture with the help of pairwise correlation plot

In [14]:
```
# pairwise correlation matrix with the target variable
atomic_mass_df <- cbind(atomic_mass_df, crit_temp=df_train[,82])
pairs(atomic_mass_df[sample.int(nrow(atomic_mass_df),1000),], lower.panel=panel.cor, 
```



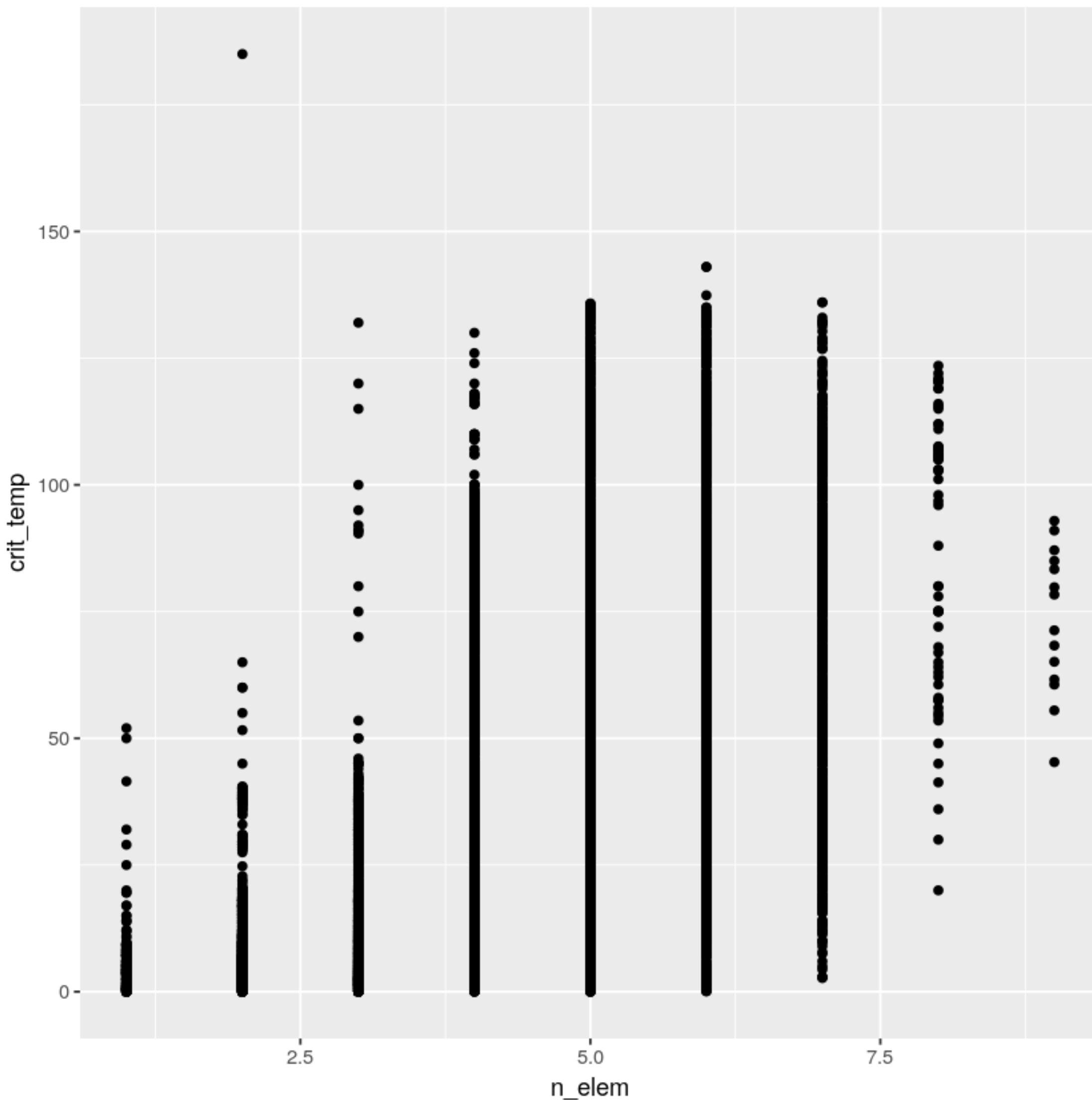
In the above pairwise correlation plot, the blue dots indicate positive correlation and red dots indicate negative correlation. The size of the dots determine the intensity of the correlation.

From the above pairwise correlation plot, we can observe the following:

- wtd_mean_atomic_mass is positively correlated to gmean_atomic_mass which means that with the increase in weighted mean, the gmean atomic mass will also increase.
- wtd_mean_atomic_mass is highly negatively correlated to critical temp as compared to gmean_atomic_mass. Therefore, on the basis of multicollinearity gmean_atomic_mass predictor can be eliminated.
- std_atomic_mass and wtd_std_atomic_mass are highly positively correlated but std_atomic_mass is also highly positively correlated to critical_temp. Therefore, wtd_std_atomic_mass can be eliminated.
- wtd_mean_atomic_mass is positively correlated to wtd_gmean_atomic_mass but wtd_mean_atomic_mass is highly negatively correlated to critical_temp. Therefore, wtd_gmean_atomic_mass can be eliminated.
- Similarly due to multicollinearity present between the predictors and target variable "critical_temp", predictors such as gmean_atomic_mass, wtd_std_atomic_mass, range_atomic_mass, wtd_gmean_atomic_mass, entropy_atomic_mass can be eliminated.
- Therefore, variables such as mean_atomic_mass, wtd_mean_atomic_mass, wtd_entropy_atomic_mass, wtd_range_atomic_mass, std_atomic_mass can be chosen as they seem to affect critical temperature of the material by some amount.

From the above plot it can also be observed that number_of_elements is positively correlated to the critical_temp. Let's have a closer view

```
In [15]: ggplot(data=atomic_mass_df, aes(x=n_elem, y=crit_temp))+ geom_point()
```



In the above graph, the X-axis represents the number of elements in the material and the Y-axis represents the critical temperature. It can be observed that the critical temperature of the materials increases with the increase in the number of elements. There is a large increase in the critical temperature for materials containing more than 3 elements.

1.3.2 2.2 Analysis of Atomic Radius features

```
In [16]: #Extracting 10 features corresponding to atomic radius
         atomic_radius_df <- df_train[,22:31]
```

```
In [17]: attach(atomic_radius_df)
         head(atomic_radius_df)
```

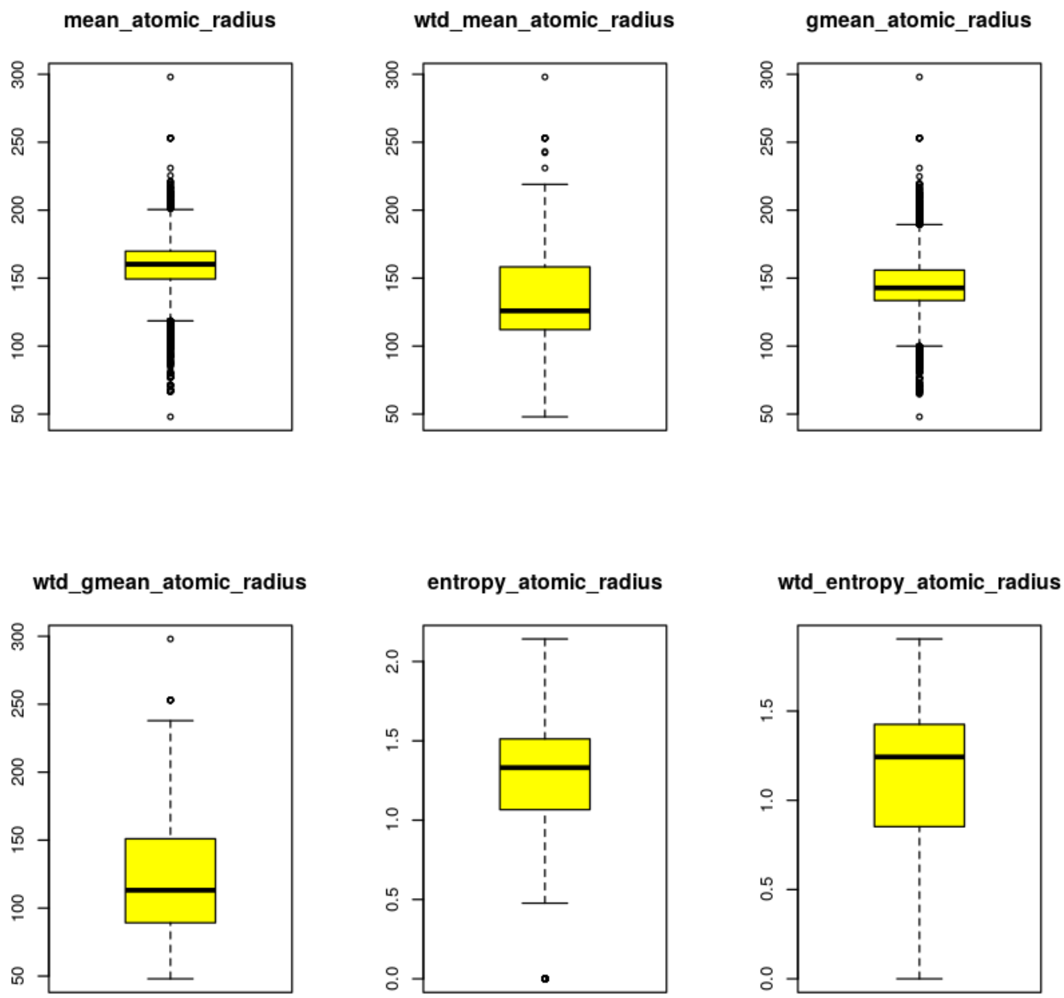
A data.frame: 6 Œ 10

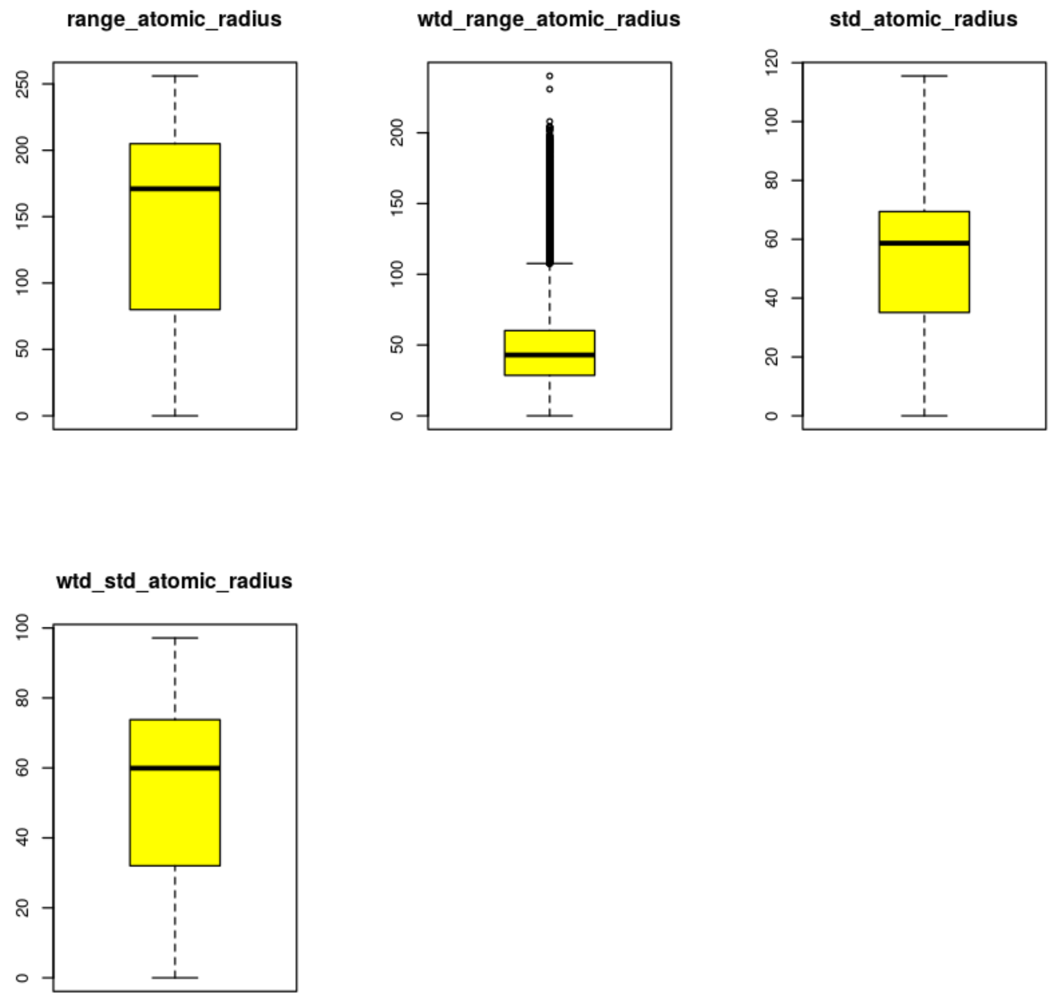
mean_atomic_radius <dbl>	wtd_mean_atomic_radius <dbl>	gmean_atomic_radius <dbl>	wtd_gme <dbl>
160.25	105.5143	136.1260	84.52842
161.20	104.9714	141.4652	84.37017
160.25	104.6857	136.1260	84.21457
160.25	105.1000	136.1260	84.37135
160.25	106.3429	136.1260	84.84344
160.25	108.0000	136.1260	85.47701

The chemical property Atomic Radius has 10 features which are all numerical continuous variables except range_atomic_radius.

Lets check the variable distributions with the help of box plots

```
In [18]: par(mfrow = c(2,3))
         for (i in 1:(length(atomic_radius_df))) {
         boxplot(atomic_radius_df[,i], main = names(atomic_radius_df[i]), type="l", col = 'yel
         }
```



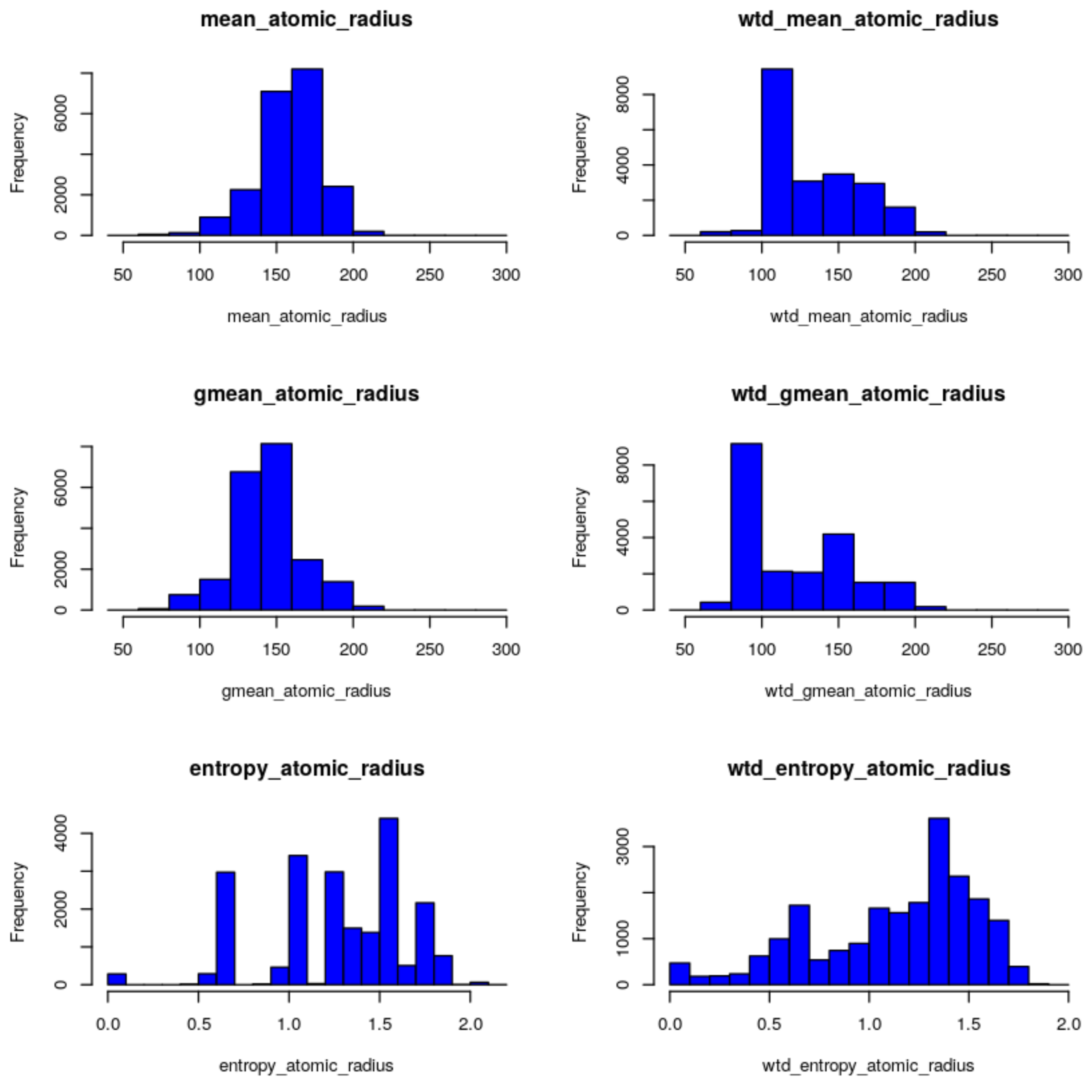


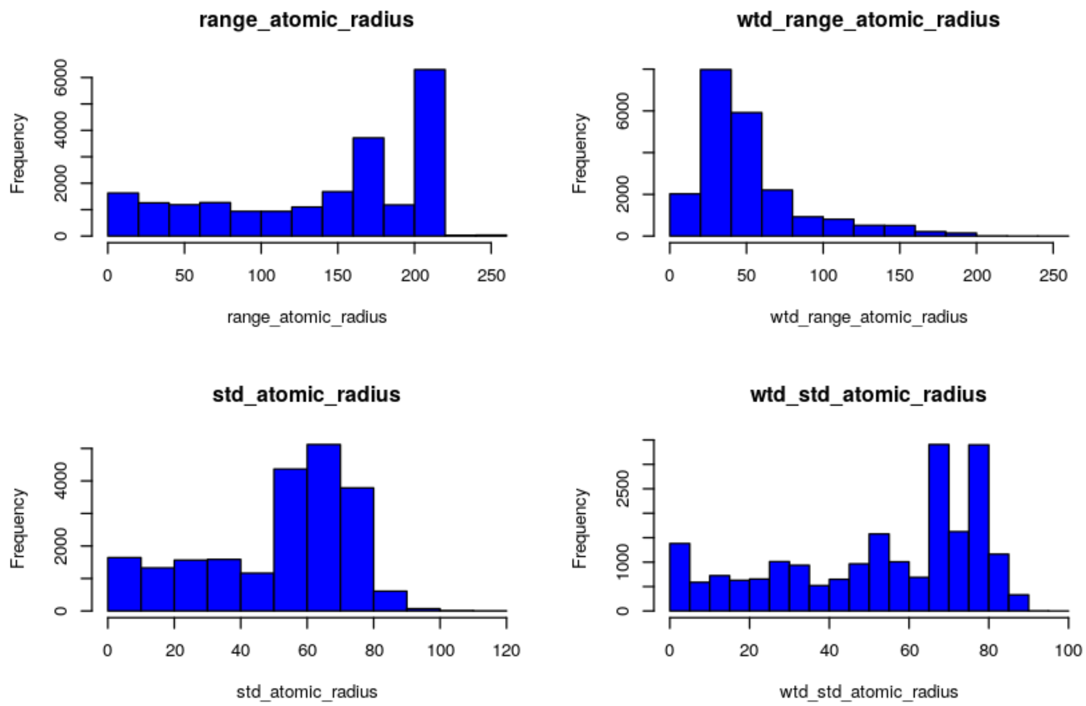
- From the above box plots, it is evident that mean_atomic_radius, gmean_atomic_radius, and wtd_range_atomic_radius contain large amount of outliers.
- range_atomic_radius in the superconducting materials lies between 0-250 units.
- All the variables show a considerate amount of variation in the data with proper IQR

We can understand the variable distributions of all the atomic radius features using histograms

In [19]:
```
par(mfrow = c(3,2))
atm_col <- colnames(atomic_radius_df)
```

```r
for (i in 1:(length(atm_col))) {
    hist(atomic_radius_df[,i], main=names(atomic_radius_df[i]), xlab = names(atomic_ra
}
```





From the above distributions, we can observe the following:

- Most of the materials have mean_atomic_radius between 140-180 units.
- More than 8000 of the materials have wtd_mean_atomic_radius around 100-120units
- The distribution of range_atomic_radius is observed to be uniform between 0-170 with unusual spikes at 180 and 200.
- wtd_range_atomic_radius distribution is right skewed with most of the materials having the range between 0-50.

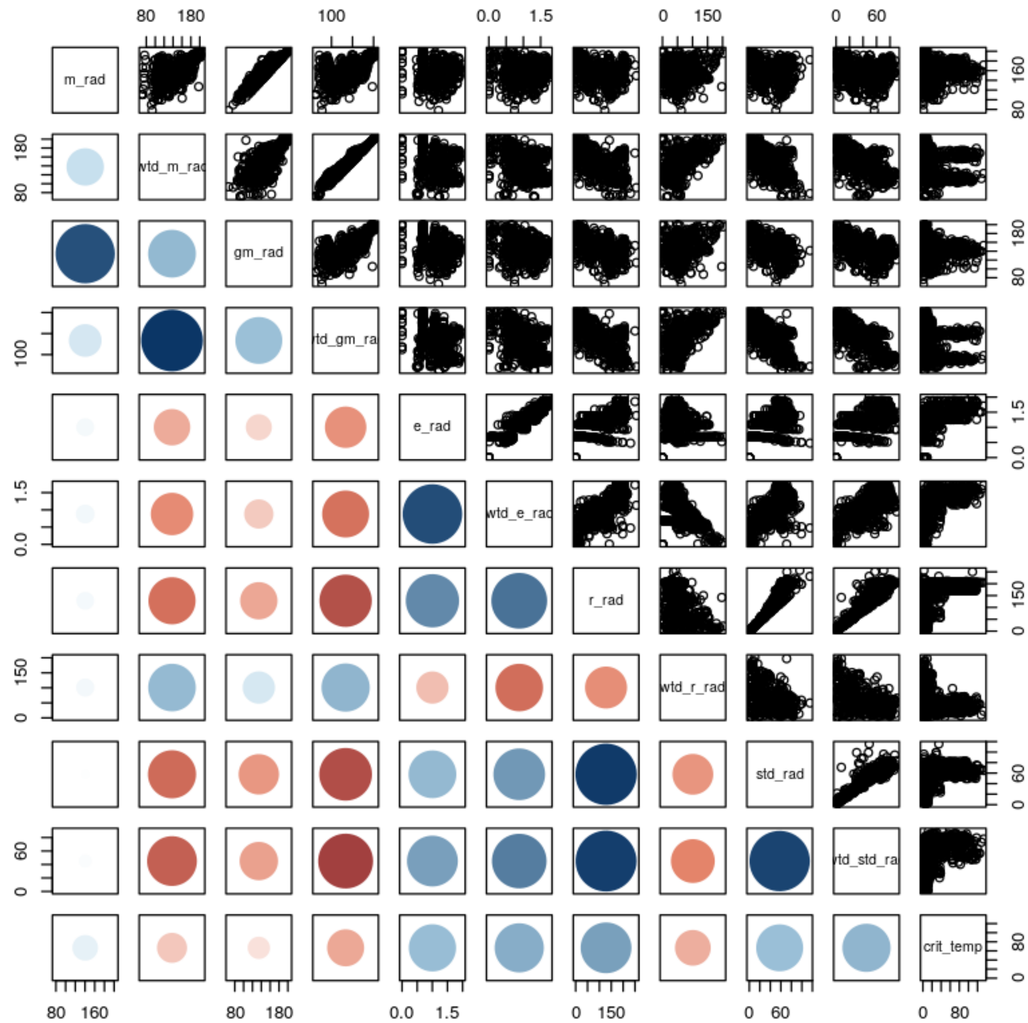
We will observe how these features of atomic radius affect crtical temperature of the material

```
In [20]: # Renaming the variables for correlation matrix
         atomic_radius_df <- atomic_radius_df %>%
```

```
        rename(
          m_rad = mean_atomic_radius,
          wtd_m_rad = wtd_mean_atomic_radius,
          gm_rad = gmean_atomic_radius,
          wtd_gm_rad = wtd_gmean_atomic_radius,
          e_rad = entropy_atomic_radius,
          wtd_e_rad = wtd_entropy_atomic_radius,
          r_rad = range_atomic_radius,
          wtd_r_rad = wtd_range_atomic_radius,
          std_rad = std_atomic_radius,
          wtd_std_rad = wtd_std_atomic_radius
          )
```

Analyzing the relationship of the atomic radius features with the critical tempearture with the help of pairwise correlation plot

```
In [21]: atomic_radius_df <- cbind(atomic_radius_df, crit_temp=df_train[,82])
         pairs(atomic_radius_df[sample.int(nrow(atomic_radius_df),1000),], lower.panel=panel.co
```



In the above pairwise correlation plot, the blue dots indicate positive correlation and red dots indicate negative correlation. The size of the dots determine the intensity of the correlation.

From the above pairwise correlation plot, we can observe the following:

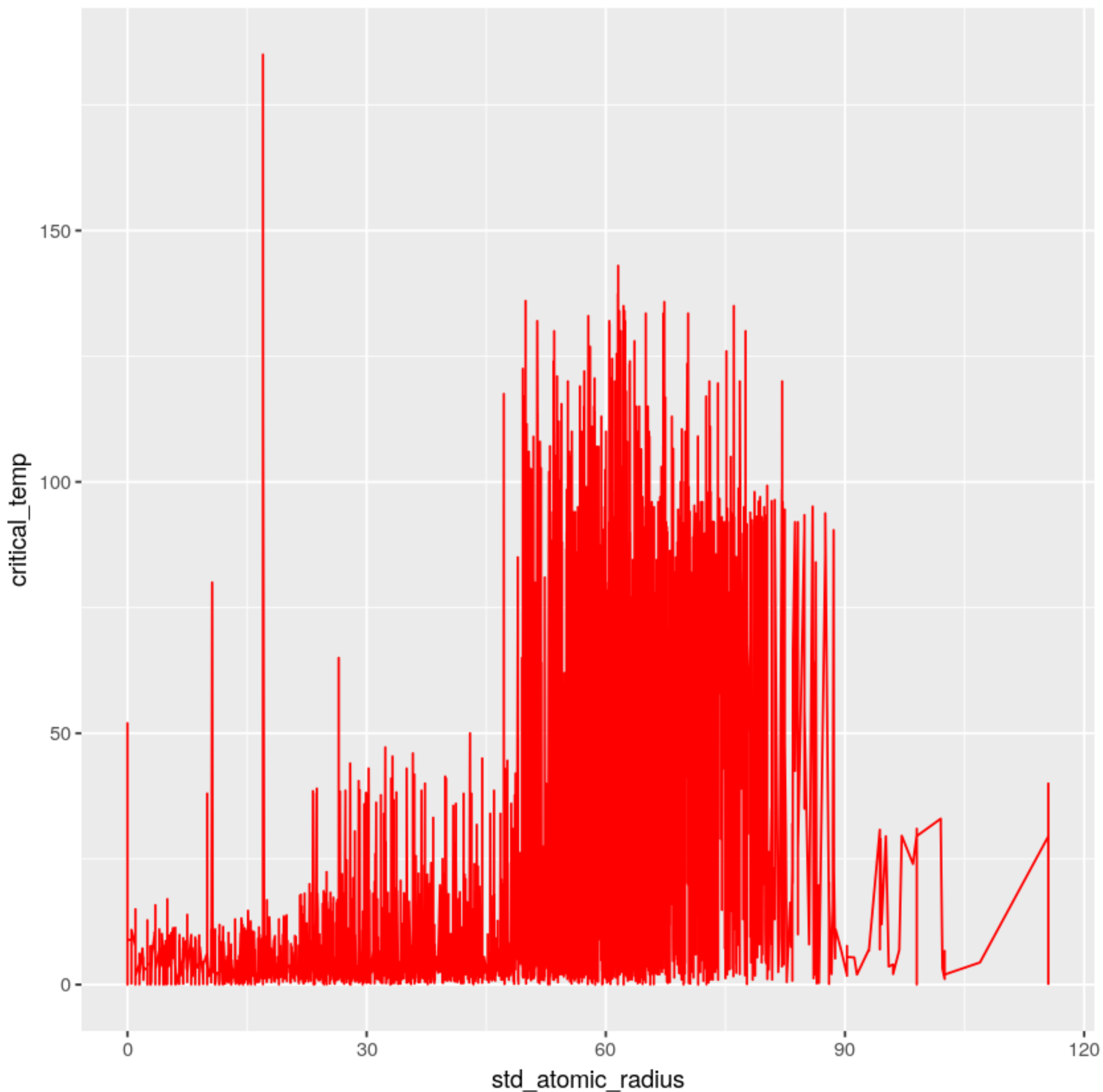
- mean_atomic_radius is positively correlated to gmean_atomic_radius which means that with the increase in the mean, the gmean atomic radius will also increase.
- mean_atomic_radius is positively correlated to critical temp as compared to gmean_atomic_radius. Therefore, on the basis of multicollinearity gmean_atomic_radius predictor can be eliminated.
- wtd_mean_atomic_radius and wtd_gmean_atomic_radius are highly positively correlated but wtd_gmean_atomic_radius is also highly negatively correlated to critical_temp. Therefore, wtd_mean_atomic_radius can be eliminated.

- entropy_atomic_radius is positively correlated to range_atomic_radius but range_atomic_radius is highly positively correlated to critical_temp. Therefore, entropy_atomic_radius can be eliminated.
- Similarly due to multicollinearity present between the predictors and target variable "critical_temp", predictors such as gmean_atomic_radius, wtd_mean_atomic_radius, entropy_atomic_radius, wtd_std_atomic_radius, range_atomic_radius, wtd_entropy_atomic_radius can be eliminated.
- Therefore, variables such as mean_atomic_radius, wtd_gmean_atomic_radius, std_atomic_radius, wtd_range_atomic_radius can be chosen as they seem to affect critical temperature of the material by some amount.

From the above plot it can also be observed that std_atomic_radius is positively correlated to the critical_temp and wtd_range_atomic_radius is highly negatively correlated to critical_temp.

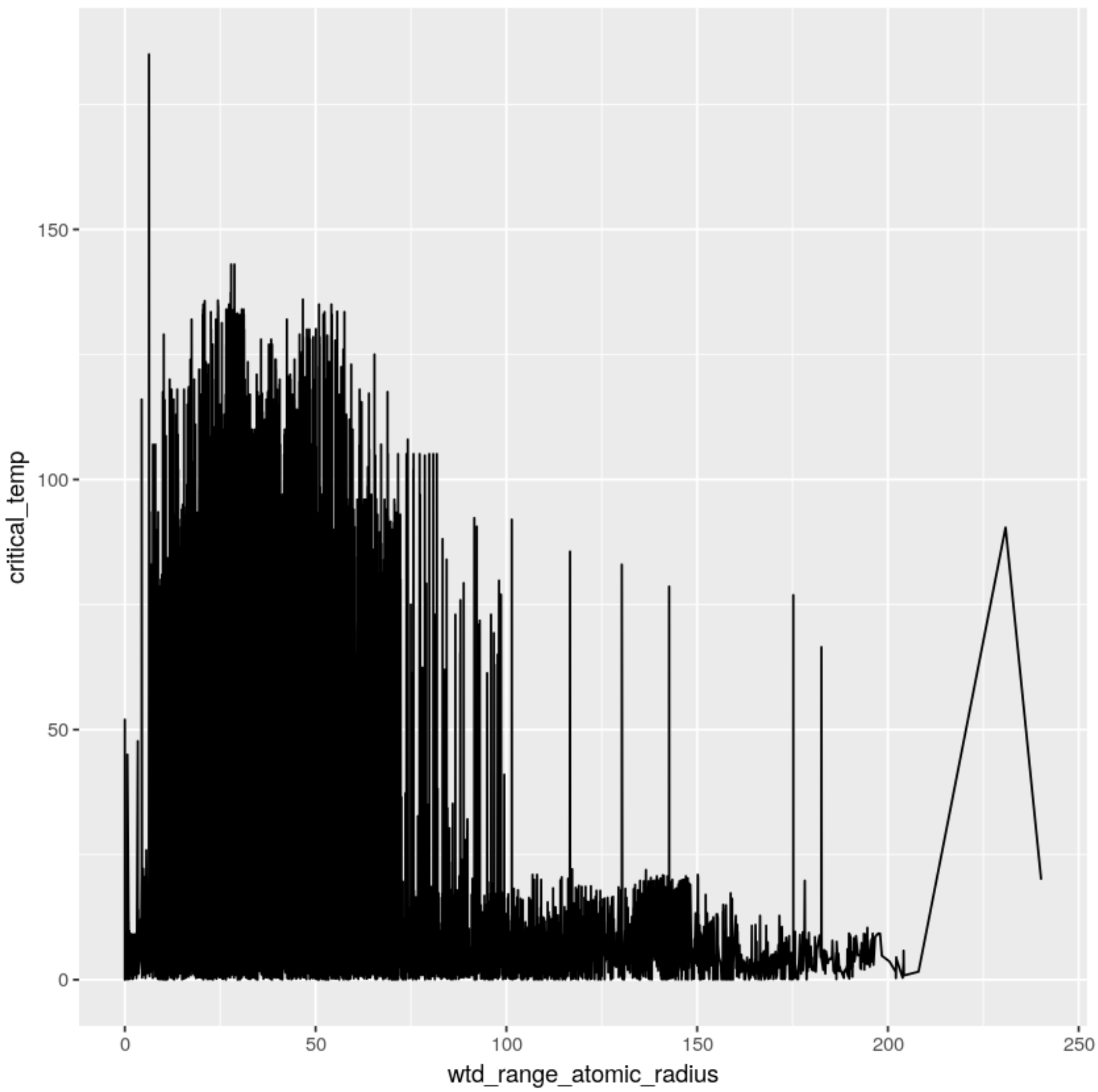
Let's have a closer view

```
In [22]: ggplot(data=atomic_radius_df, aes(x=std_rad, y=crit_temp))+ geom_line(col="red") + lal
```



In the above graph, the X-axis represents the standard atomic radius and the Y-axis represents the critical temperature. From the graph, it can be observed that as the standard atomic radius of the material increases the critical temperature also increases. The increase in the critical temperature is evident until standard atomic radius is 90units but for the materials above 90units there is sudden decrease in the critical tempearture.

```
In [23]: ggplot(data=atomic_radius_df, aes(x=wtd_r_rad, y=crit_temp))+ geom_line() + labs(x="w
```



In the above graph, the X-axis represents the weighted range atomic radius and the Y-axis represents the critical temperature. Weighted range atomic radius is calculated by taking a product of the proportion of the different elements used in the material and their respective atomic radius. From the above graph, it is observed that the critical temperature is very high for the materials with low weighted range atomic radius and it decreases with the increase in the weighted range atomic radius.

This behaviour may be observed due to the proportion of the elements used in the material i.e., as the proportion of the elements increases the overall weighted range atomic radius increases which may induce a decrease in the critical temperature of the material.

1.3.3 2.3 Analysis of Density features

In [24]:
```
#Extracting 10 features corresponding to density of the material
density_df <- df_train[,32:41]
```

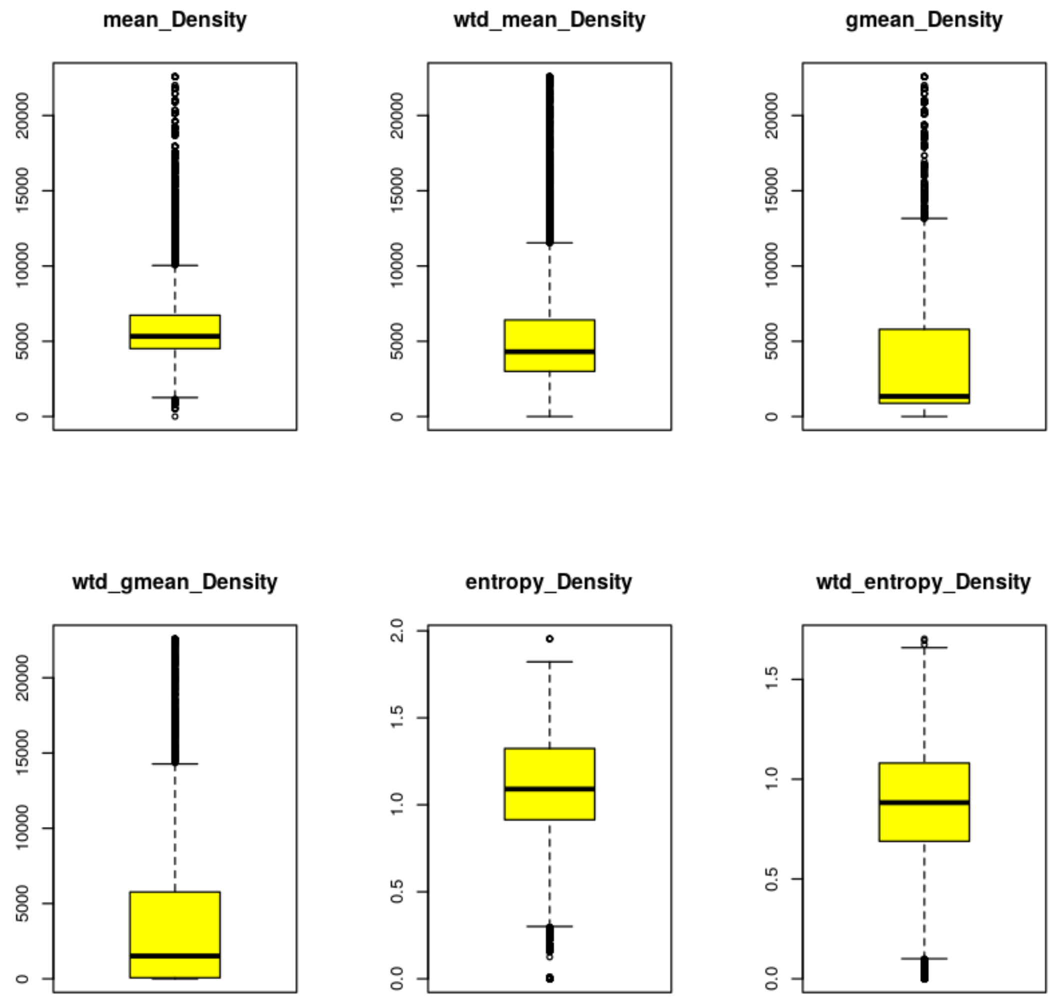
In [25]:
```
attach(density_df)
head(density_df)
```

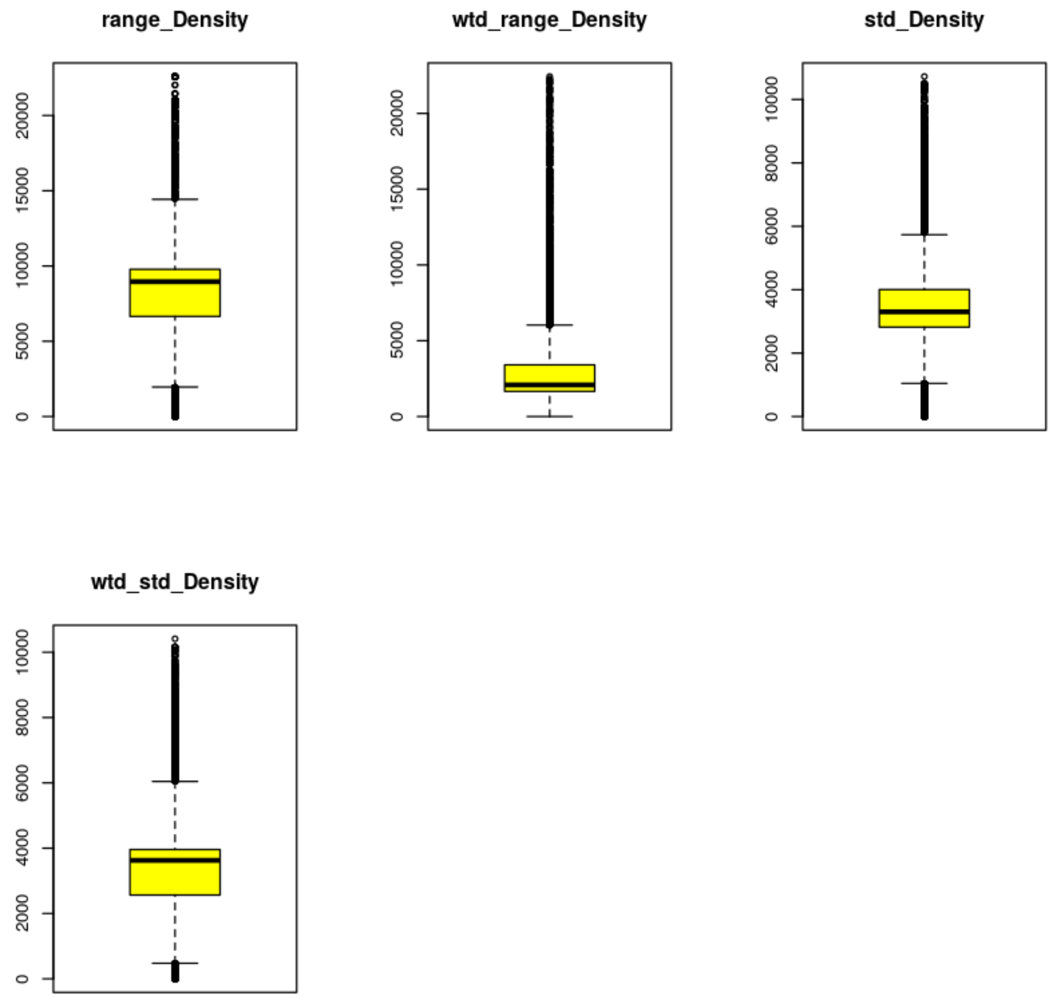
A data.frame: 6 Œ 10

mean_Density <dbl>	wtd_mean_Density <dbl>	gmean_Density <dbl>	wtd_gmean_Density <dbl>	entrop <dbl>
4654.357	2961.502	724.9532	53.54381	1.0331
5821.486	3021.017	1237.0951	54.09572	1.3144
4654.357	2999.159	724.9532	53.97402	1.0331
4654.357	2980.331	724.9532	53.75849	1.0331
4654.357	2923.845	724.9532	53.11703	1.0331
4654.357	2848.531	724.9532	52.27364	1.0331

The chemical property Density has 10 features which are all numerical continuous variables.
Lets check the variable distributions with the help of box plots

In [26]:
```
par(mfrow = c(2,3))
for (i in 1:(length(density_df))) {
boxplot(density_df[,i], main = names(density_df[i]), type="l", col = 'yellow')
}
```

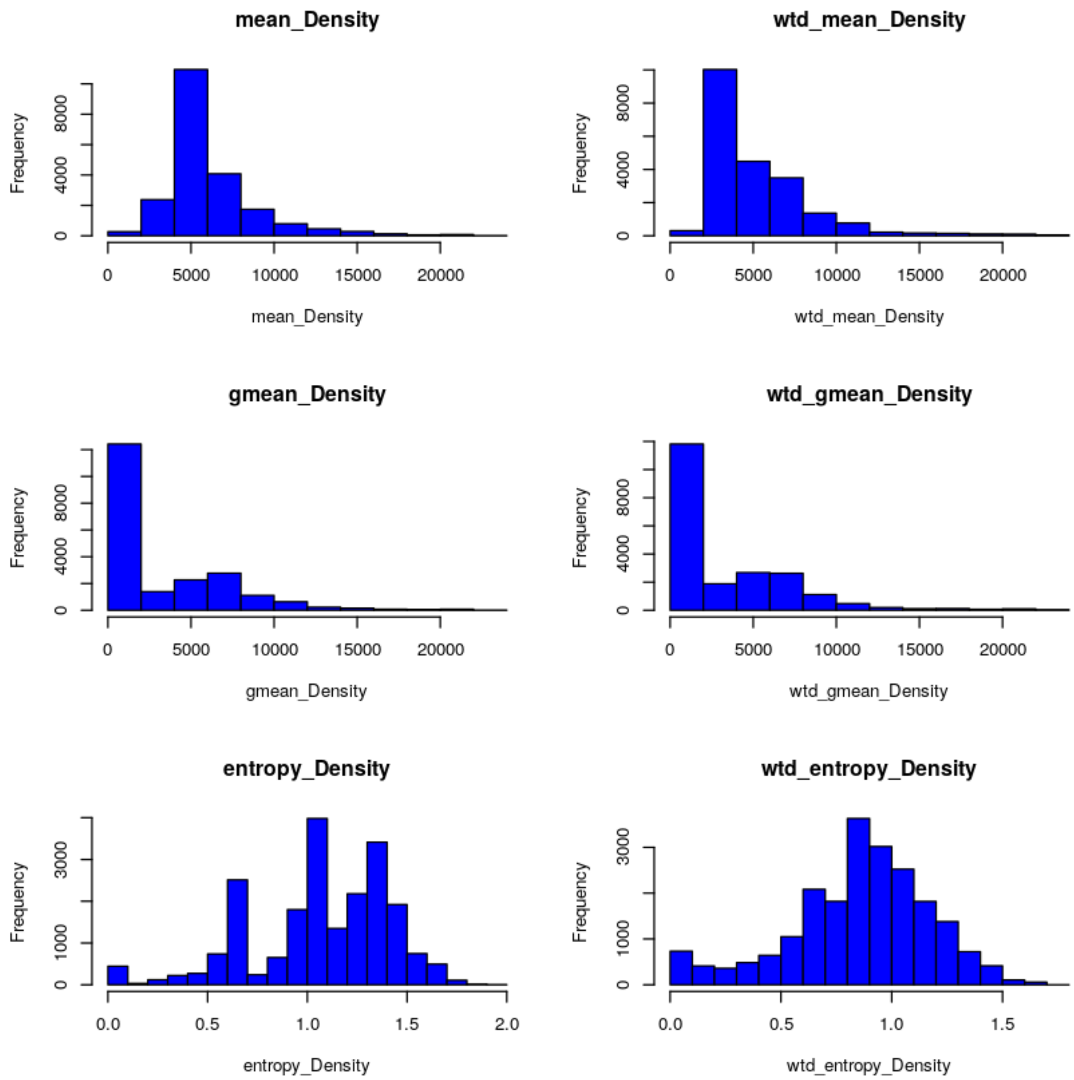

mean_Density
0
5000
10000
15000
20000
wtd_mean_Density
0
5000
10000
15000
20000
gmean_Density
0
5000
10000
15000
20000
wtd_gmean_Density
0
5000
10000
15000
20000
entropy_Density
0.0
0.5
1.0
1.5
2.0
wtd_entropy_Density
0.0
0.5
1.0
1.5

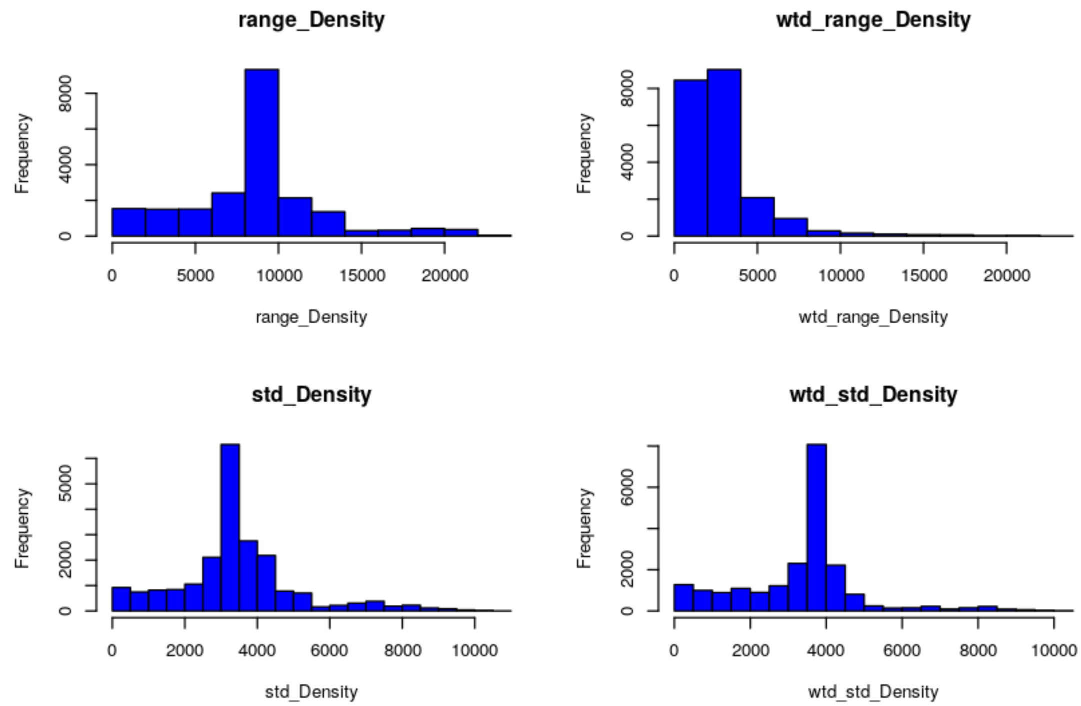


- From the above box plots, it is observed that all the variables contain outliers.
- gmean_Density, range_Density, wtd_range_Density and wtd_std_Density show less variation in the data as compared to all the other variables.
- range_Density can therefore be a candidate for variable elimination.

We can understand the variable distributions of all Density features using histograms

In [27]:
```
par(mfrow = c(3,2))
atm_col <- colnames(density_df)
for (i in 1:(length(atm_col))) {
    hist(density_df[,i], main=names(density_df[i]), xlab = names(density_df[i]), col=
}
```


mean_Density
Frequency
0
4000
8000
0
5000
10000
15000
20000
mean_Density
wtd_mean_Density
Frequency
0
4000
8000
0
5000
10000
15000
20000
wtd_mean_Density
gmean_Density
Frequency
0
4000
8000
0
5000
10000
15000
20000
gmean_Density
wtd_gmean_Density
Frequency
0
4000
8000
0
5000
10000
15000
20000
wtd_gmean_Density
entropy_Density
Frequency
0
1000
3000
0.0
0.5
1.0
1.5
2.0
entropy_Density
wtd_entropy_Density
Frequency
0
1000
3000
0.0
0.5
1.0
1.5
wtd_entropy_Density



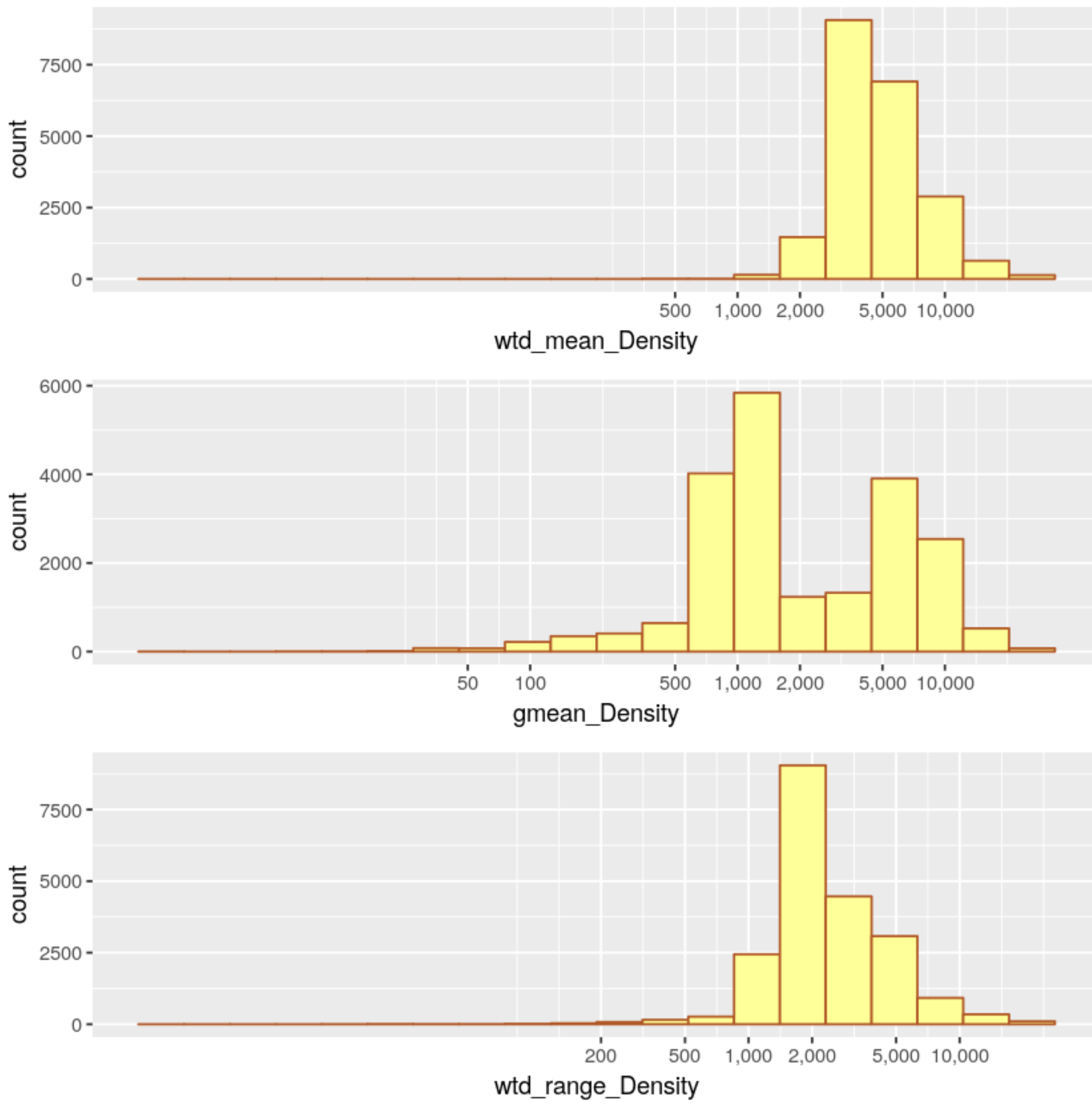
From the above distributions, we can observe the following:

- Most of the materials have mean_Density between 3000-6000 units.
- The distribution of wtd_mean_Density is right skewed with more than 8000 materials having wtd_mean_Density of 2000-4000 units.
- The ditributions of gmean_Density, wtd_range_Density and wtd_gmean_Density are also right skewed.
- The distribution of wtd_entropy_Density is normal which indicates that this feature may affect the critical temperature of the material.

Replotting wtd_mean_Density, gmean_Density, wtd_range_Density to see if the log scale has normal distribution

In [28]:
```
# Set some colours using Colorbrewer
gg.colour <- brewer.pal(12,"Paired")[12]
gg.fill <- brewer.pal(12,"Paired")[11]

# Re-plot some of the charts using log scales to counteract the skew
p1 <- ggplot(aes(x=wtd_mean_Density), data=density_df) +
      geom_histogram(bins=20, colour=gg.colour, fill=gg.fill) +
      scale_x_log10(labels=comma,breaks=c(500,1000,2000,5000,10000))
p3 <- ggplot(aes(x=gmean_Density), data=density_df) +
      geom_histogram(bins=20, colour=gg.colour, fill=gg.fill) +
      scale_x_log10(labels=comma,breaks=c(50,100,500,1000,2000,5000,10000))
p4 <- ggplot(aes(x=wtd_range_Density), data=density_df) +
      geom_histogram(bins=20, colour=gg.colour, fill=gg.fill) +
      scale_x_log10(labels=comma,breaks=c(200, 500,1000,2000,5000,10000))
grid.arrange(p1, p3, p4, ncol=1, nrow=3)
```

```
Warning message:
Transformation introduced infinite values in continuous x-axisWarning message:
Removed 287 rows containing non-finite values (stat_bin).
```



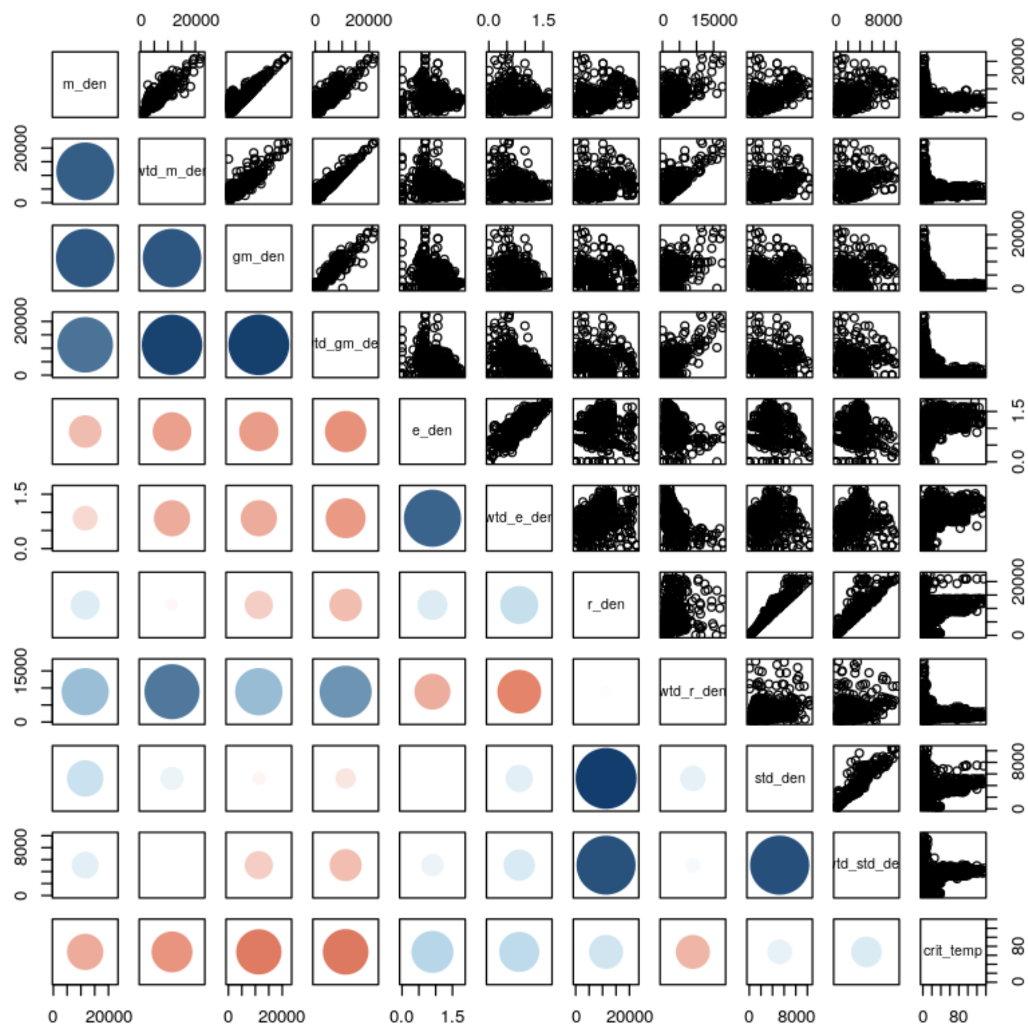
- The log of wtd_mean_Density is not quite normal as it demonstrates a normal curve between 2000-10000 units but contains many outliers below 1000 units.
- Similarly the log distributions of gmean_Density and wtd_range_Density are not normal as they contain many outliers

Analyzing the relationship of the Density features with the critical tempearture with the help of pairwise correlation plot

```
In [29]: # Renaming the variables for correlation matrix
         density_df <- density_df %>%
           rename(
             m_den = mean_Density,
             wtd_m_den = wtd_mean_Density,
```

```
            gm_den = gmean_Density,
            wtd_gm_den = wtd_gmean_Density,
            e_den = entropy_Density,
            wtd_e_den = wtd_entropy_Density,
            r_den = range_Density,
            wtd_r_den = wtd_range_Density,
            std_den = std_Density,
            wtd_std_den = wtd_std_Density
            )
```

```
In [30]: density_df <- cbind(density_df, crit_temp=df_train[,82])
         pairs(density_df[sample.int(nrow(density_df),1000),], lower.panel=panel.cor, col="bla
```



In the above pairwise correlation plot, the blue dots indicate positive correlation and red dots indicate negative correlation. The size of the dots determine the intensity of the correlation.

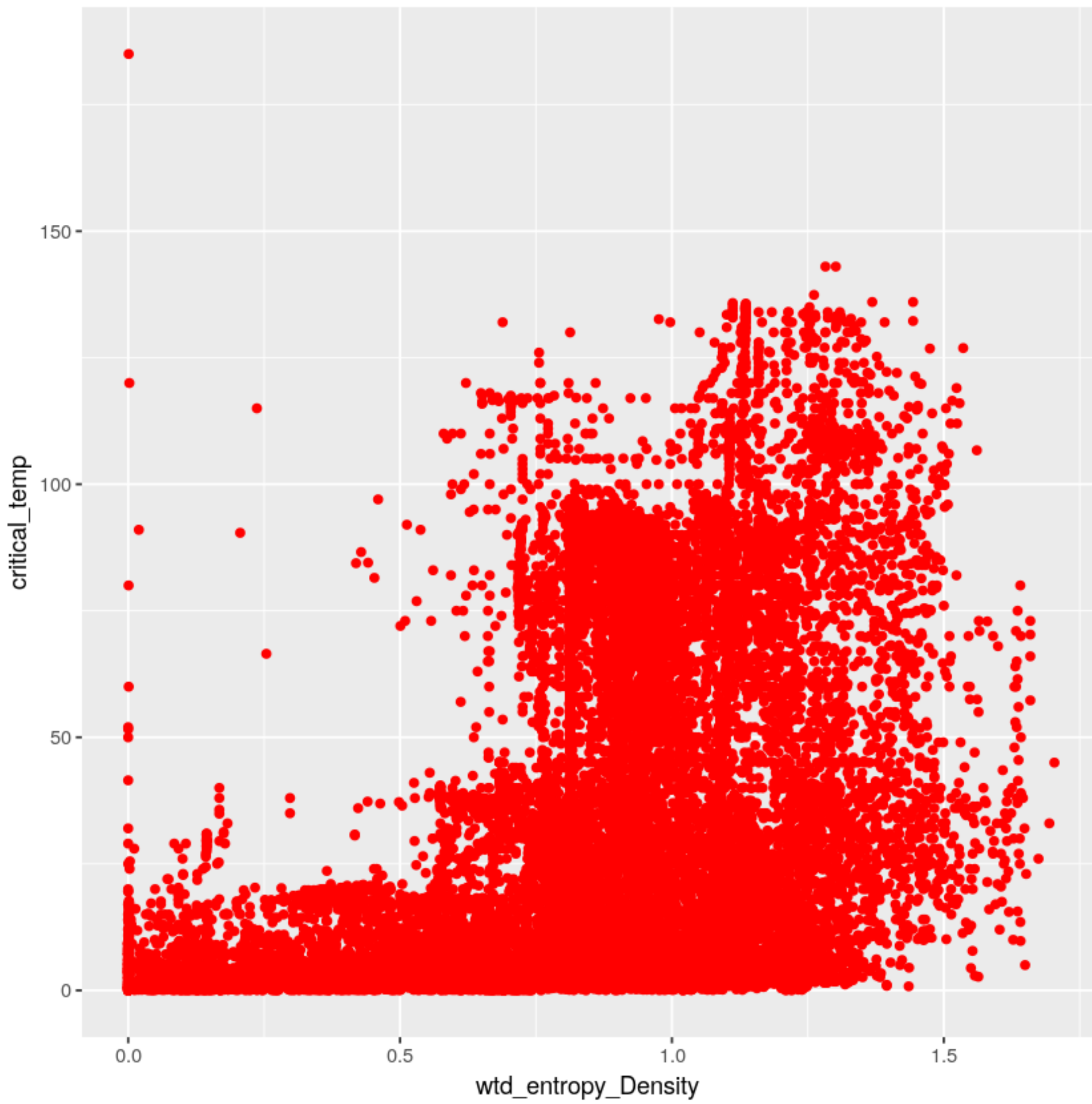
From the above pairwise correlation plot, we can observe the following:

- wtd_mean_Density is positively correlated to wtd_gmean_Density which means that with the increase in the mean Density, the weighted geometric mean Density will also increase.
- wtd_gmean_Density is highly negatively correlated to critical temp as compared to wtd_mean_Density. Therefore, on the basis of multicollinearity wtd_mean_Density predictor can be eliminated.
- gmean_Density and wtd_gmean_Density are highly positively correlated but wtd_gmean_Density is also highly negatively correlated to critical_temp. Therefore, gmean_Density can be eliminated.
- range_Density is positively correlated to std_Density but range_Density is highly positively correlated to critical_temp. Therefore, std_Density can be eliminated.
- Similarly due to multicollinearity present between the predictors and target variable "critical_temp", predictors such as wtd_mean_Density, gmean_Density, std_Density, range_Density, wtd_gmean_Density can be eliminated.
- Therefore, variables such as mean_Density, wtd_range_Density, entropy_Density, wtd_entropy_Density and wtd_std_Density can be chosen as they seem to affect critical temperature of the material by some amount.

From the above plot it can also be observed that entropy_Density is positively correlated to the critical_temp.

Let's have a closer view

```
In [31]: ggplot(data=density_df, aes(x=wtd_e_den, y=crit_temp))+ geom_point(col="red") + labs(
```



In the above graph, the X-axis represents the weighted entropy Density and the Y-axis represents the critical temperature. From the graph, it can be observed that as the weighted entropy Density of the material increases the critical temperature also increases. This indicates that wtd_entropy_Density may have some effect on the critical temperature of the superconductor.

1.3.4 2.4 Analysis of Fusion Heat features

```
In [32]: fusion_heat_df <- df_train[,52:61]
```

```
In [33]: attach(fusion_heat_df)
         head(fusion_heat_df)
```

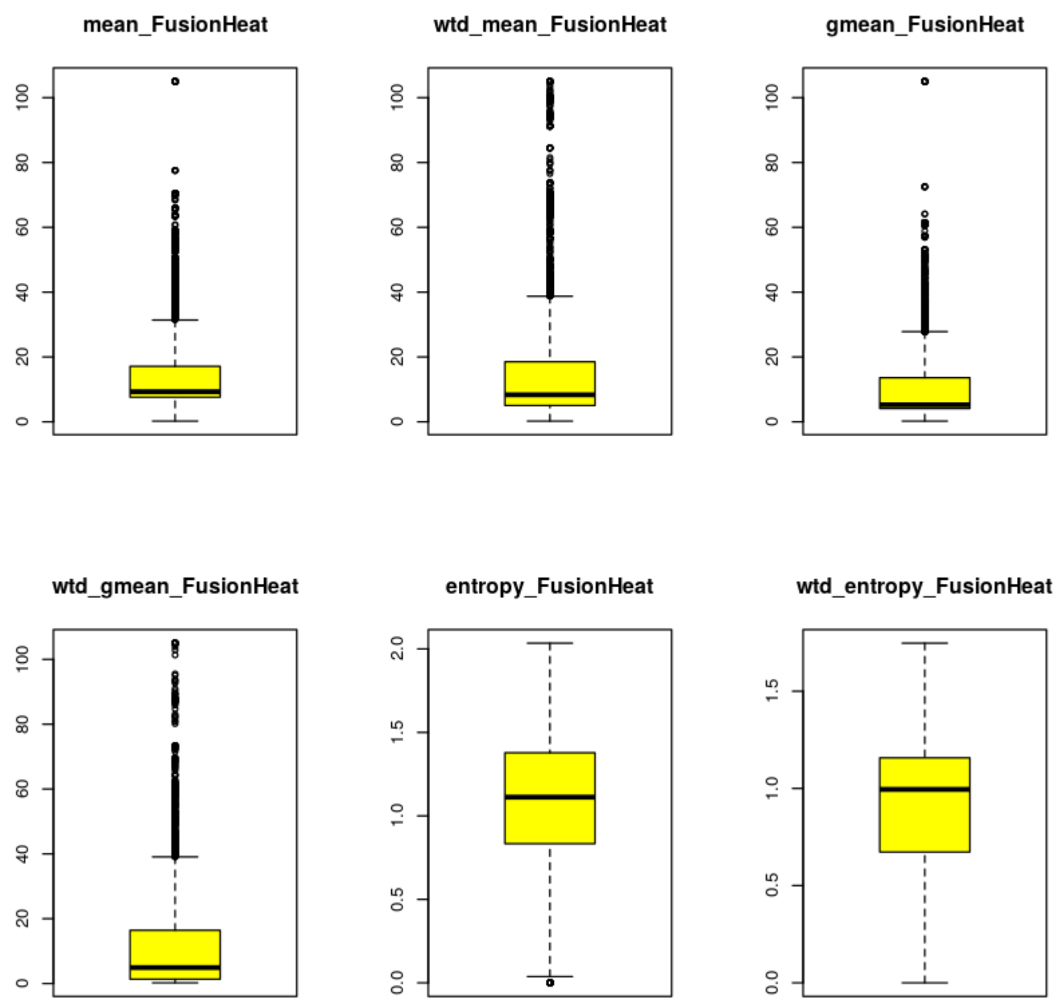
A data.frame: 6 Œ 10

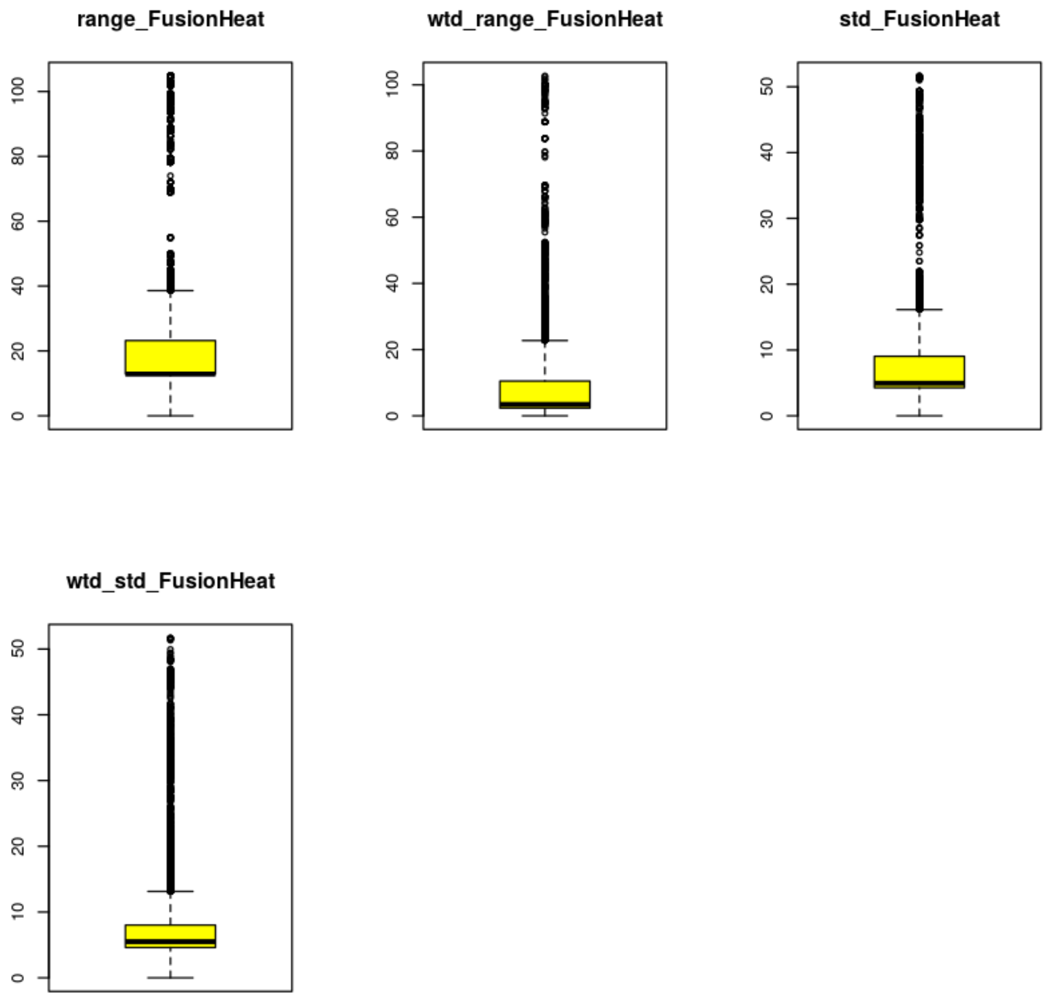
mean_FusionHeat <dbl>	wtd_mean_FusionHeat <dbl>	gmean_FusionHeat <dbl>	wtd_gmean_Fusi <dbl>
6.9055	3.846857	3.479475	1.040986
7.7844	3.796857	4.403790	1.035251
6.9055	3.822571	3.479475	1.037439
6.9055	3.834714	3.479475	1.039211
6.9055	3.871143	3.479475	1.044545
6.9055	3.919714	3.479475	1.051699

The chemical property Fusion Heat has 10 features which are all numerical continuous variables.

Lets check the variable distributions with the help of box plots

```
In [34]: par(mfrow = c(2,3))
         for (i in 1:(length(fusion_heat_df))) {
         boxplot(fusion_heat_df[,i], main = names(fusion_heat_df[i]), type="l", col = 'yellow')
         }
```

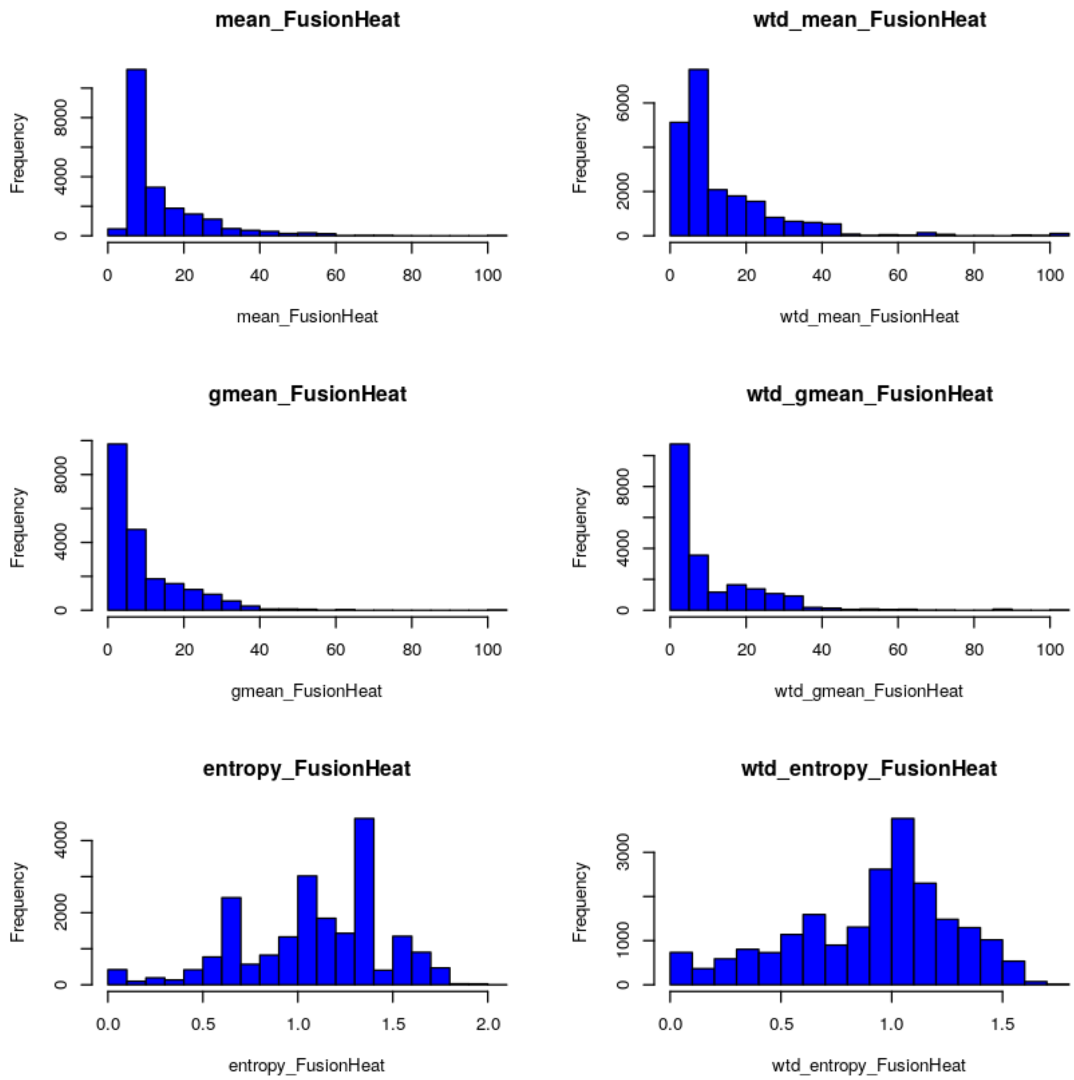


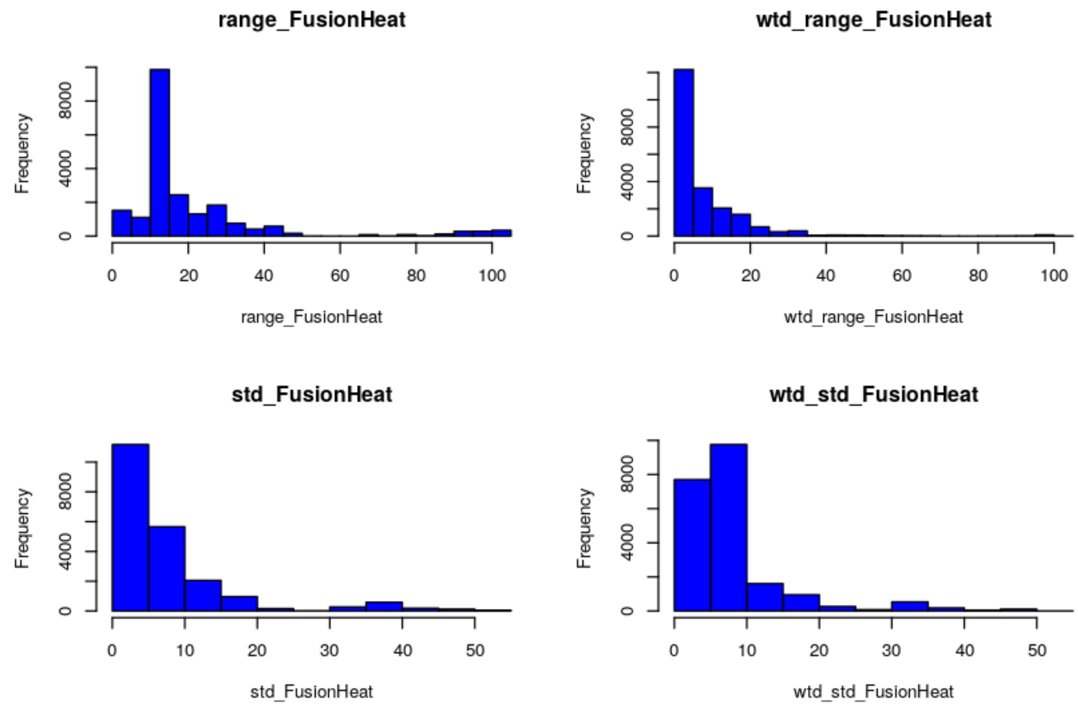


- From the above box plots, it is observed that all the variables contain outliers except entropy_FusionHeat and wtd_entropy_FusionHeat.
- mean_FusionHeat, gmean_FusionHeat, range_FusionHeat, wtd_range_FusionHeat and std_FusionHeat show less variation in the data as compared to all the other variables.
- The median of range_FusionHeat is also very low along with its variance and therefore, it an be a candidate for elimination.

We can understand the variable distributions of all FusionHeat features using histograms

```
In [35]: par(mfrow = c(3,2))
         atm_col <- colnames(fusion_heat_df)
         for (i in 1:(length(atm_col))) {
             hist(fusion_heat_df[,i], main=names(fusion_heat_df[i]), xlab = names(fusion_heat_
         }
```





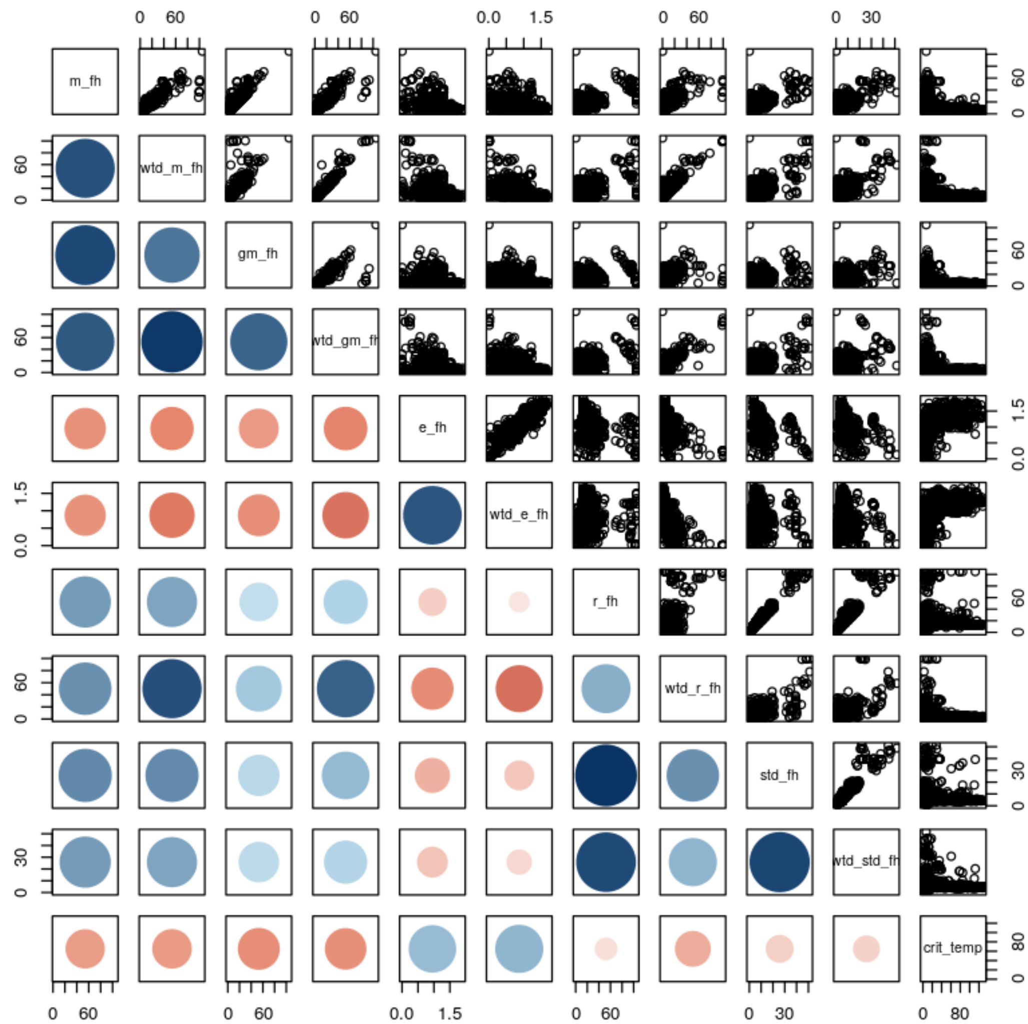
- From the above distributions, it can be observed that entropy_FusionHeat and wtd_entropy_FusionHeat are quite normally distributed.
- The distributions of all the other variables of FusionHeat are right skewed.

Analyzing the relationship of the FusionHeat features with the critical tempearture with the help of pairwise correlation plot

```
In [36]: # Renaming the variables for correlation matrix
         fusion_heat_df <- fusion_heat_df %>%
           rename(
             m_fh = mean_FusionHeat,
             wtd_m_fh = wtd_mean_FusionHeat,
             gm_fh = gmean_FusionHeat,
```

```
             wtd_gm_fh = wtd_gmean_FusionHeat,
             e_fh = entropy_FusionHeat,
             wtd_e_fh = wtd_entropy_FusionHeat,
             r_fh = range_FusionHeat,
             wtd_r_fh = wtd_range_FusionHeat,
             std_fh = std_FusionHeat,
             wtd_std_fh = wtd_std_FusionHeat
             )
```

```
In [37]: fusion_heat_df <- cbind(fusion_heat_df, crit_temp=df_train[,82])
         pairs(fusion_heat_df[sample.int(nrow(fusion_heat_df),1000),], lower.panel=panel.cor,
```



In the above pairwise correlation plot, the blue dots indicate positive correlation and red dots indicate negative correlation. The size of the dots determine the intensity of the correlation.

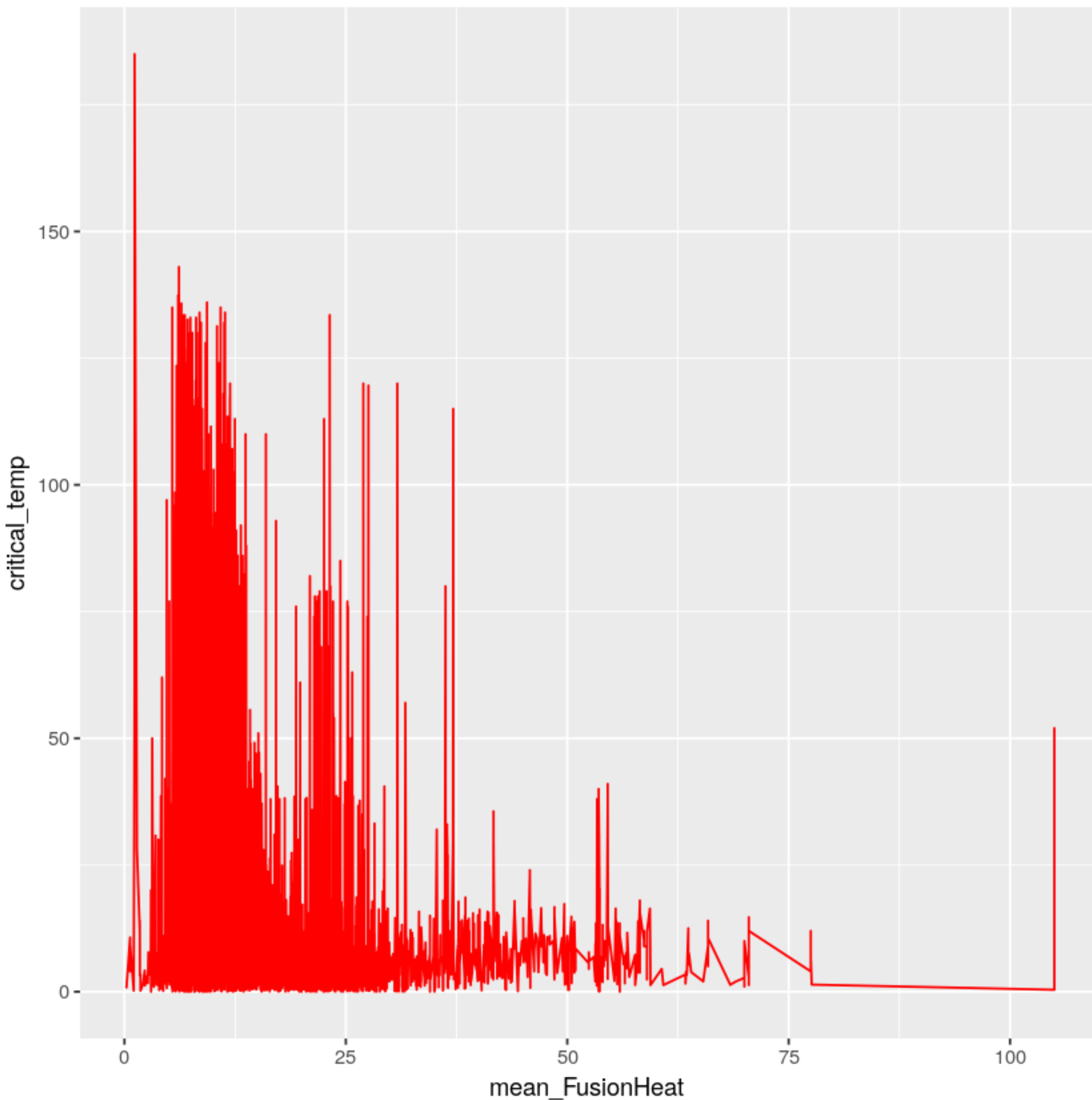
From the above pairwise correlation plot, we can observe the following:

- mean_FusionHeat is positively correlated to wtd_mean_FusionHeat which means that with the increase in the mean Density, the weighted mean FusionHeat will also increase.
- mean_FusionHeat is highly negatively correlated to critical temp as compared to wtd_mean_FusionHeat. Therefore, on the basis of multicollinearity wtd_mean_FusionHeat predictor can be eliminated.
- gmean_FusionHeat and wtd_gmean_FusionHeat are highly positively correlated but wtd_gmean_FusionHeat is also highly negatively correlated to critical_temp. Therefore, gmean_FusionHeat can be eliminated.
- entropy_FusionHeat is positively correlated to wtd_entropy_FusionHeat but wtd_entropy_FusionHeat is highly positively correlated to critical_temp. Therefore, entropy_FusionHeat can be eliminated.
- Similarly due to multicollinearity present between the predictors and target variable "critical_temp", predictors such as wtd_mean_FusionHeat, gmean_FusionHeat, entropy_FusionHeat, wtd_entropy_FusionHeat, wtd_std_FusionHeat can be eliminated.
- Therefore, variables such as mean_FusionHeat, wtd_range_FusionHeat, and std_FusionHeat can be chosen as they seem to affect critical temperature of the material by some amount.

From the above plot it can also be observed that mean_FusionHeat is negatively correlated to the critical_temp.

Let's have a closer view

```
In [38]: ggplot(data=fusion_heat_df, aes(x=m_fh, y=crit_temp))+ geom_line(col="red") + labs(x=
```



- In the above graph, the X-axis represents the mean_FusionHeat and Y-axis represents the critical_temp. Fusion heat of a superconducting material is the energy to change the state of the material from solid to liquid without changing the temperature. From the graph, it can be observed that as the mean_FusionHeat increases the critical temperature of the material decreases. As the energy to change the state increases which does not affect the temperature so this change of phase can take place at low temperatures and therefore, this trend might be observed in the superconducting material.

- From the above analysis of Atomic mass, Atomic Radius, Density and Fusion Heat properties of semiconductors the variables such as gmean_atomic_mass, wtd_std_atomic_mass, range_atomic_mass, wtd_gmean_atomic_mass, entropy_atomic_mass, gmean_atomic_radius, wtd_mean_atomic_radius, entropy_atomic_radius, wtd_std_atomic_radius, range_atomic_radius, wtd_entropy_atomic_radius, wtd_mean_Density, gmean_Density, std_Density, range_Density, wtd_gmean_Density, wtd_mean_FusionHeat, gmean_FusionHeat, entropy_FusionHeat, wtd_entropy_FusionHeat, wtd_std_FusionHeat, range_FusionHeat can be candidates for variable elimination due to multicollinearity and low variance.

1.4 3. Variable Identification and Explanation

From the above EDA it is evident that from 82 variables in the given dataset not all variables contribute towards predicting the critical temperature of the superconducting material.

We will be validating our intuition of non-informative predictors chosen for elimination through EDA with the help of R functions and discard them while building the model. This will allow us to build a model with greater accuracy and transparency.

We can achieve the identification of non-informative predictors using the following methods:

- Identifying variables with zero variance and eliminating them
- Identifying variables with high collinearity and eliminating them

1.4.1 3.1 Identifying variables with zero variance and eliminating them

This helps to identify those variables with less variation in the data and less unique values as compared to the sample size.

```
In [39]: # identifying variables with near zero variance
         near_zero_variance <- nearZeroVar(df_train, saveMetrics = TRUE, freqCut = 2,uniqueCut
```

In the near_zero_variance dataframe, the 4th column indicates whether the variable has near zero variance or not.

We extract all the variables with near zero variance

```
In [40]: nzv.true <- near_zero_variance[which(near_zero_variance$nzv == TRUE),]
         nzv.true
```

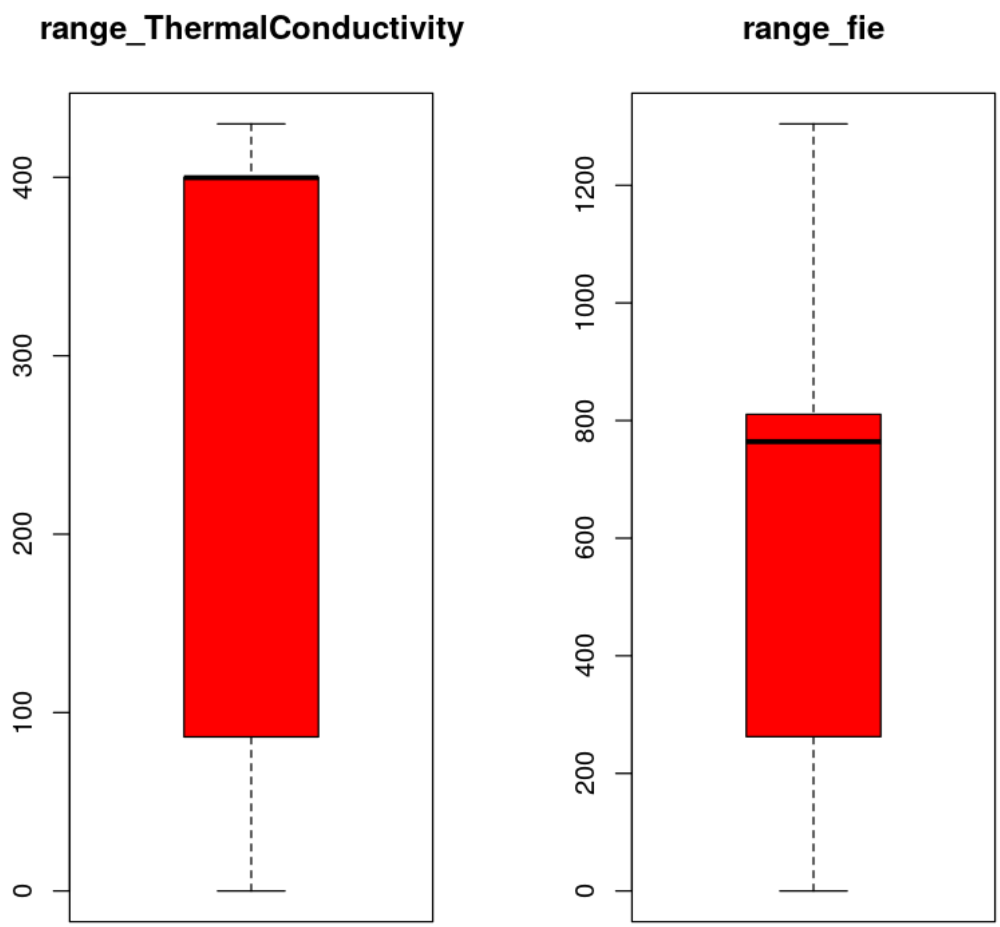
A data.frame: 4 Œ 4

	freqRatio <dbl>	percentUnique <dbl>	zeroVar <lgl>	nzv <lgl>
range_fie	3.009630	4.016366	FALSE	TRUE
range_Density	4.236315	4.185675	FALSE	TRUE
range_FusionHeat	13.076923	2.798288	FALSE	TRUE
range_ThermalConductivity	23.316854	2.125758	FALSE	TRUE

From the EDA for Atomic Radius and Fusion Heat we had chosen range_Density and range_FusionHeat as candidates for elimination which proves true from the above table.

Let's explore range_fie and range_ThermalConductivity through boxplots

```
In [41]: par(mfrow = c(1,2))
         boxplot(df_train[,'range_ThermalConductivity'], main = 'range_ThermalConductivity', ty
         boxplot(df_train[,'range_fie'], main = 'range_fie', type="l", col = 'red')
```



From the above boxplots, it is observed that the median which shows the variablity in the data is skewed. It also indicates that most of the data values in the sample data are identical and therefore, these variables have low variance.

Discarding all such near zero variance variables from the data for further analysis

```
In [42]: filtered_variables <- near_zero_variance[which(near_zero_variance$nzv == FALSE),]
         filtered_data <- df_train[rownames(filtered_variables)]
```

```
In [43]: dim(filtered_data)
```

1. 21263 2. 78

The dimension of the filtered data indicates that out of 82 variables we have eliminated 4 variables on the basis of low variance.

1.4.2 3.2 Identifying variables with high collinearity and eliminating them

Here we determine the pairwise correlation between the variables and eliminate those with the highest collinearity

In [44]:
```
# pairwise correlation for the filtered variables
correlation_mat <- cor(filtered_data[ , 1:78])
cor_ <- as.data.frame.table(correlation_mat)
colnames(cor_) <- c("a", "b", "cor")
cor_ <- cor_[cor_$a != cor_$b, ]
cor_ <- cor_[order(abs(cor_$cor), decreasing = TRUE), ]
cor_ <- cor_[seq(1, nrow(cor_), 2), ]
cor_$cor <- round(cor_$cor, 2)
rownames(cor_) <- 1:nrow(cor_)
print(cor_[1:10, ])
```

```
                         a                      b  cor
1     entropy_atomic_radius            entropy_fie 1.00
2         wtd_gmean_Valence        wtd_mean_Valence 0.99
3           entropy_Valence            entropy_fie 0.99
4             wtd_gmean_fie           wtd_mean_fie 0.99
5             gmean_Valence           mean_Valence 0.99
6           entropy_Valence  entropy_atomic_radius 0.99
7    wtd_gmean_atomic_radius wtd_mean_atomic_radius 0.98
8               std_Valence          range_Valence 0.97
9               entropy_fie     number_of_elements 0.97
10     std_ElectronAffinity range_ElectronAffinity 0.97
```

In [45]:
```
cor_matrix_critical_temp <- cor_[which(cor_$a == 'critical_temp'),]
cor_matrix_critical_temp[order(cor_matrix_critical_temp$cor, decreasing = TRUE), ]
```

A data.frame: 77 Œ 3

	a	b	cor
	<fct>	<fct>	<dbl>
233	critical_temp	wtd_std_ThermalConductivity	0.72
353	critical_temp	range_atomic_radius	0.65
355	critical_temp	std_ThermalConductivity	0.65
410	critical_temp	wtd_entropy_atomic_mass	0.63
471	critical_temp	wtd_entropy_atomic_radius	0.60
478	critical_temp	number_of_elements	0.60
483	critical_temp	wtd_std_atomic_radius	0.60
485	critical_temp	entropy_Valence	0.60
510	critical_temp	wtd_entropy_Valence	0.59
528	critical_temp	wtd_std_fie	0.58
557	critical_temp	entropy_fie	0.57
571	critical_temp	wtd_entropy_FusionHeat	0.56
578	critical_temp	std_atomic_radius	0.56
582	critical_temp	entropy_atomic_radius	0.56
602	critical_temp	entropy_FusionHeat	0.55
629	critical_temp	entropy_atomic_mass	0.54
637	critical_temp	std_fie	0.54
798	critical_temp	range_atomic_mass	0.49
881	critical_temp	wtd_range_ThermalConductivity	0.47
925	critical_temp	entropy_Density	0.46
1006	critical_temp	entropy_ElectronAffinity	0.44
1140	critical_temp	wtd_entropy_Density	0.40
1144	critical_temp	wtd_mean_fie	0.40
1201	critical_temp	wtd_entropy_fie	0.39
1233	critical_temp	wtd_mean_ThermalConductivity	0.38
1236	critical_temp	std_atomic_mass	0.38
1245	critical_temp	mean_ThermalConductivity	0.38
1312	critical_temp	wtd_std_atomic_mass	0.36
1379	critical_temp	wtd_gmean_fie	0.34
1493	critical_temp	wtd_std_ElectronAffinity	0.32
2034	critical_temp	mean_ElectronAffinity	-0.19
1998	critical_temp	std_FusionHeat	-0.20
2025	critical_temp	wtd_std_FusionHeat	-0.20
1963	critical_temp	std_Valence	-0.21
1872	critical_temp	gmean_atomic_mass	-0.23
1627	critical_temp	wtd_range_Density	-0.28
1572	critical_temp	wtd_std_Valence	-0.30
1579	critical_temp	wtd_mean_atomic_radius	-0.30
1500	critical_temp	wtd_range_FusionHeat	-0.31
1512	critical_temp	wtd_mean_atomic_mass	-0.31
1378	critical_temp	wtd_range_atomic_radius	-0.34
1404	critical_temp	wtd_range_atomic_mass	-0.34
1262	critical_temp	wtd_gmean_ThermalConductivity	-0.37
1268	critical_temp	wtd_gmean_atomic_mass	-0.37
1274	critical_temp	mean_Density	-0.37
1232	critical_temp	gmean_ElectronAffinity	-0.38
1170	critical_temp	wtd_mean_FusionHeat	-0.39
1204	critical_temp	gmean_ThermalConductivity	-0.39
1213	critical_temp	mean_FusionHeat	-0.39
1124	critical_temp	wtd_gmean_atomic_radius	-0.41

From the above pairwise correlation coefficient between the different predictors and target variable i.e, critical_temp, the correlation coefficients are all below 0.85

```
In [46]: summary(correlation_mat[upper.tri(correlation_mat)])
```

```
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.91426 -0.26366  0.04554  0.06660  0.38341  0.99774
```

From the above summary it is observed that more than 50% of the variables have the absolute correlation coefficient between 0-0.9. Thus, the cut-off used for eliminating highly correlated variables is 0.85

```
In [47]: # determining highly correlated variables
         highCor <- findCorrelation(correlation_mat, cutoff = 0.85, verbose = TRUE)
```

```
Compare row 20  and column  27 with corr  0.887
  Means:  0.513 vs 0.343 so flagging column 20
Compare row 26  and column  72 with corr  0.919
  Means:  0.509 vs 0.338 so flagging column 26
Compare row 27  and column  30 with corr  0.958
  Means:  0.499 vs 0.334 so flagging column 27
Compare row 30  and column  19 with corr  0.859
  Means:  0.485 vs 0.33 so flagging column 30
Compare row 72  and column  73 with corr  0.911
  Means:  0.477 vs 0.325 so flagging column 72
Compare row 73  and column  16 with corr  0.908
  Means:  0.469 vs 0.321 so flagging column 73
Compare row 16  and column  7 with corr  0.892
  Means:  0.462 vs 0.317 so flagging column 16
Compare row 7  and column  1 with corr  0.882
  Means:  0.455 vs 0.313 so flagging column 7
Compare row 34  and column  33 with corr  0.952
  Means:  0.463 vs 0.309 so flagging column 34
Compare row 19  and column  29 with corr  0.876
  Means:  0.452 vs 0.305 so flagging column 19
Compare row 1  and column  25 with corr  0.972
  Means:  0.439 vs 0.3 so flagging column 1
Compare row 25  and column  6 with corr  0.972
  Means:  0.425 vs 0.296 so flagging column 25
Compare row 33  and column  32 with corr  0.877
  Means:  0.431 vs 0.293 so flagging column 33
Compare row 71  and column  69 with corr  0.995
  Means:  0.416 vs 0.288 so flagging column 71
Compare row 69  and column  68 with corr  0.937
  Means:  0.406 vs 0.284 so flagging column 69
Compare row 6  and column  44 with corr  0.885
  Means:  0.386 vs 0.281 so flagging column 6
Compare row 67  and column  66 with corr  0.956
```

```
  Means:  0.396 vs 0.277 so flagging column 67
Compare row 55  and column  54 with corr  0.882
  Means:  0.372 vs 0.273 so flagging column 55
Compare row 24  and column  13 with corr  0.914
  Means:  0.397 vs 0.269 so flagging column 24
Compare row 68  and column  70 with corr  0.99
  Means:  0.368 vs 0.265 so flagging column 68
Compare row 54  and column  35 with corr  0.918
  Means:  0.334 vs 0.263 so flagging column 54
Compare row 13  and column  22 with corr  0.87
  Means:  0.364 vs 0.259 so flagging column 13
Compare row 32  and column  5 with corr  0.853
  Means:  0.344 vs 0.256 so flagging column 32
Compare row 22  and column  15 with corr  0.872
  Means:  0.331 vs 0.253 so flagging column 22
Compare row 8  and column  11 with corr  0.918
  Means:  0.316 vs 0.25 so flagging column 8
Compare row 5  and column  3 with corr  0.964
  Means:  0.308 vs 0.248 so flagging column 5
Compare row 17  and column  45 with corr  0.862
  Means:  0.317 vs 0.246 so flagging column 17
Compare row 53  and column  52 with corr  0.866
  Means:  0.305 vs 0.242 so flagging column 53
Compare row 52  and column  50 with corr  0.927
  Means:  0.289 vs 0.24 so flagging column 52
Compare row 61  and column  62 with corr  0.879
  Means:  0.285 vs 0.238 so flagging column 61
Compare row 11  and column  10 with corr  0.92
  Means:  0.274 vs 0.237 so flagging column 11
Compare row 46  and column  49 with corr  0.866
  Means:  0.305 vs 0.234 so flagging column 46
Compare row 49  and column  48 with corr  0.898
  Means:  0.283 vs 0.231 so flagging column 49
Compare row 51  and column  50 with corr  0.91
  Means:  0.246 vs 0.23 so flagging column 51
Compare row 4  and column  2 with corr  0.94
  Means:  0.265 vs 0.229 so flagging column 4
Compare row 23  and column  21 with corr  0.916
  Means:  0.26 vs 0.228 so flagging column 23
Compare row 39  and column  38 with corr  0.906
  Means:  0.21 vs 0.227 so flagging column 38
Compare row 41  and column  43 with corr  0.9
  Means:  0.233 vs 0.229 so flagging column 41
Compare row 12  and column  14 with corr  0.969
  Means:  0.214 vs 0.229 so flagging column 14
Compare row 42  and column  40 with corr  0.875
  Means:  0.206 vs 0.232 so flagging column 40
Compare row 65  and column  60 with corr  0.876
```

```
  Means:  0.212 vs 0.235 so flagging column 60
Compare row 74  and column  77 with corr  0.867
  Means:  0.187 vs 0.238 so flagging column 77
Compare row 74  and column  76 with corr  0.974
  Means:  0.168 vs 0.242 so flagging column 76
Compare row 57  and column  58 with corr  0.94
  Means:  0.177 vs 0.247 so flagging column 58
All correlations <= 0.85
```

```
In [48]: highCor.variables <- colnames(correlation_mat[,highCor])
         print(paste0("length of highly correlated variables:",length(highCor.variables)))
         highCor.variables
```

```
[1] "length of highly correlated variables:44"
```

1. 'wtd_std_fie' 2. 'wtd_entropy_atomic_radius' 3. 'range_atomic_radius' 4. 'wtd_std_atomic_radius' 5. 'entropy_Valence' 6. 'wtd_entropy_Valence' 7. 'entropy_fie' 8. 'wtd_entropy_atomic_mass' 9. 'wtd_gmean_Density' 10. 'std_fie' 11. 'number_of_elements' 12. 'entropy_atomic_radius' 13. 'gmean_Density' 14. 'wtd_gmean_Valence' 15. 'wtd_mean_Valence' 16. 'entropy_atomic_mass' 17. 'wtd_std_ThermalConductivity' 18. 'wtd_entropy_FusionHeat' 19. 'wtd_gmean_atomic_radius' 20. 'mean_Valence' 21. 'entropy_FusionHeat' 22. 'wtd_mean_fie' 23. 'wtd_mean_Density' 24. 'wtd_mean_atomic_radius' 25. 'range_atomic_mass' 26. 'wtd_gmean_atomic_mass' 27. 'wtd_entropy_fie' 28. 'wtd_gmean_FusionHeat' 29. 'gmean_FusionHeat' 30. 'gmean_ThermalConductivity' 31. 'wtd_std_atomic_mass' 32. 'range_ElectronAffinity' 33. 'wtd_std_ElectronAffinity' 34. 'wtd_mean_FusionHeat' 35. 'gmean_atomic_mass' 36. 'gmean_atomic_radius' 37. 'wtd_mean_ElectronAffinity' 38. 'std_Density' 39. 'gmean_fie' 40. 'mean_ElectronAffinity' 41. 'wtd_std_FusionHeat' 42. 'wtd_mean_ThermalConductivity' 43. 'wtd_std_Valence' 44. 'std_Valence'

findCorrelation function in R identifies the pairs of variables with pairwise correlation greater than 0.85

We can observe that the function has identified 44 variables for elimination. The variables chosen for elimination at the time of EDA also match the above set of variables.

We will eliminate these variables as retaining both the pairs will cancel out the effect on the target variable.

```
In [49]: filtered.var <- colnames(filtered_data)
         filtered.var <- filtered.var[!is.element(filtered.var, highCor.variables)]
         length(filtered.var)
```

34

After eliminating the variables on the basis of variance and high collinearity we are left with 34 variables that may have some effect on the target variable.

Dimension reduction can be further achieved using backward stepwise selection

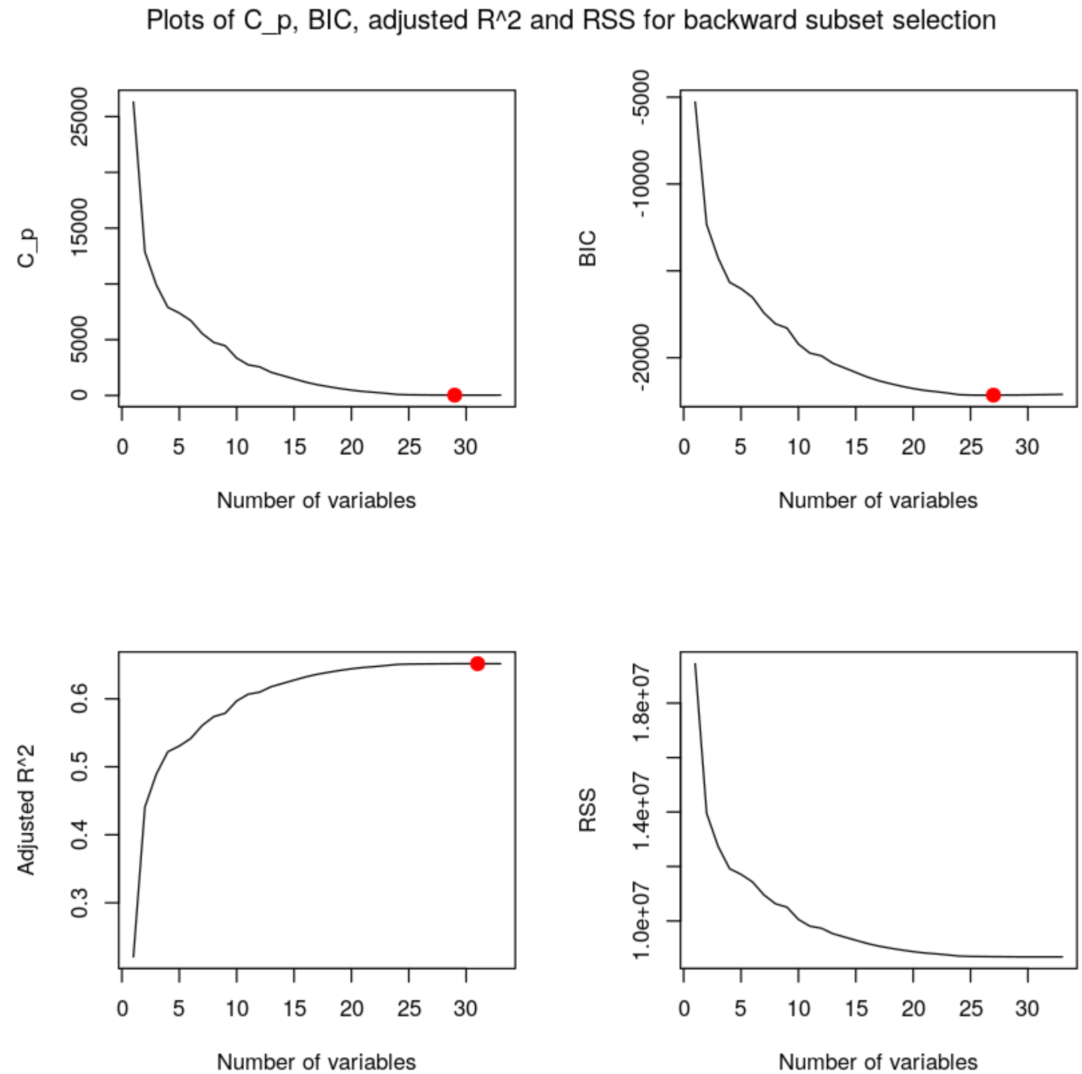
Applying backward selection on the filtered data

```
In [50]: filtered_data <- filtered_data[filtered.var]
         regfit.bwd <- regsubsets(filtered_data$critical_temp ~ ., data = filtered_data, nvmax
         reg.summary.bwd <- summary(regfit.bwd)
```

```
In [51]: cat("Cp - ",which.min(reg.summary.bwd$cp),"\n")
         cat("BIC - ",which.min(reg.summary.bwd$bic),"\n")
         cat("Adjusted R^2 - ",which.max(reg.summary.bwd$adjr2),"\n")
```

```
Cp -  29
BIC -  27
Adjusted R^2 -  31
```

```
In [52]: 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
         plot(reg.summary.bwd$bic, xlab = "Number of variables", ylab = "BIC", type = "l")
         points(which.min(reg.summary.bwd$bic), reg.summary.bwd$bic[which.min(reg.summary.bwd$b
         plot(reg.summary.bwd$adjr2, xlab = "Number of variables", ylab = "Adjusted R^2", type
         points(which.max(reg.summary.bwd$adjr2), reg.summary.bwd$adjr2[which.max(reg.summary.b
         plot(reg.summary.bwd$rss, xlab = "Number of variables", ylab = "RSS", type = "l")
         mtext("Plots of C_p, BIC, adjusted R^2 and RSS for backward subset selection", side =
```



The backward selection method provides 3 different models with different predictors based on different measuring parameters. To build an accurate model we will have to choose a model with low test error.

- We can observe that the Adjusted Rˆ2 gives the highest number of predictors. This may be observed as the Adjusted Rˆ2 increases with increase in the number of variables with low RSS. This model may fit the training data perfectly but will fail to fit the test data. Therefore, training error cannot be a good estimate for the test error. It is possible that instead of decreasing the test error may increase with the increase in the number of variables. Therefore, Adjusted Rˆ2 is not a good parameter for choosing the model.

- BIC tends to select the model with smallest value and low test error. It achieves this by applying heavy penalities on the models with large number of observations. BIC uses the log term which allows it to choose a better model than Cp [3].

- Therefore, we will be choosing a model on the basis of the BIC value which is the least i.e, 27 in our case.

Retrieving the predictors selected by the BIC model

```
In [53]: coef(regfit.bwd, which.min(reg.summary.bwd$bic))
```

(Intercept) -78.1385665094788 **mean_atomic_mass** 0.216724227748872 **wtd_range_atomic_mass** -0.23922865253437 **std_atomic_mass** 0.288643073513474 **mean_fie** 0.102126970928346 **wtd_gmean_fie** -0.0344425353221016 **mean_atomic_radius** 0.214109399719897 **wtd_range_atomic_radius** -0.111789240192576 **std_atomic_radius** 0.220906397428001 **mean_Density** -0.0034640029905625 **wtd_entropy_Density** 4.65687337931882 **wtd_range_Density** 0.00318626061304765 **wtd_std_Density** -0.00262536209049676 **wtd_gmean_ElectronAffinity** -0.0965586168147964 **entropy_ElectronAffinity** 16.0583415403056 **wtd_entropy_ElectronAffinity** -42.1556099400556 **wtd_range_ElectronAffinity** -0.209037848322798 **std_ElectronAffinity** 0.143718196533683 **mean_FusionHeat** 0.218528077081088 **std_FusionHeat** -0.550774747484175 **mean_ThermalConductivity** 0.212694136868428 **wtd_gmean_ThermalConductivity** -0.302795292985635 **entropy_ThermalConductivity** 12.9706541150817 **wtd_entropy_ThermalConductivity** 13.7847443794914 **wtd_range_ThermalConductivity** 0.245703506804995 **std_ThermalConductivity** 0.042785339199767 **gmean_Valence** 1.4780802012474 **range_Valence** -3.22623167644813

- The backward stepwise selection method has provided 27 most important features affecting the critical temperature of the superconducting material. The predictors are as follows:

"mean_atomic_mass","wtd_range_atomic_mass","std_atomic_mass","mean_fie","wtd_gmean_fie","mean_ato
"wtd_range_atomic_radius","std_atomic_radius","mean_Density","wtd_entropy_Density","wtd_range_Density",
"wtd_std_Density","wtd_gmean_ElectronAffinity","entropy_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","std_ThermalConductivity",
"gmean_Valence","range_Valence"

- We will be using these features for building statistical models and comparing the accuracy of the models.

1.5 4. Model Development

We will be building different statistical models on the superconductor data using the features selected by the backward stepwise selection method.

As the target variable in the superconductor data i.e., critical_temp is a numerical continuous variable, we will be analysing different regression models.

```
In [54]: # Features provided by backward stepwise selection
         selected_reduced_var <- c("mean_atomic_mass","wtd_range_atomic_mass","std_atomic_mass'
```

Filtering the given dataset based on the selected features

```
In [55]: filter_data <- df_train[selected_reduced_var]
```

Sampling the dataset into training and test data with 80% train data and 20% test data.

```
In [56]: sample_size <- floor(0.80 * nrow(filter_data))

         ## set the seed to make your partition reproducible
         set.seed(123)
         train_ind <- sample(seq_len(nrow(filter_data)), size = sample_size)

         train <- filter_data[train_ind, ]
         test <- filter_data[-train_ind, ]
```

```
In [59]: dim(train)
```

1. 17010 2. 28
We can observe that the training data has 17010 rows with 28 columns

```
In [60]: dim(test)
```

1. 4253 2. 28
We can observe that the test data has 4253 rows with 28 columns

1.5.1 4.1 Multiple Linear Regression

Fitting the linear regression model on the training data

```
In [57]: fit1 <- lm(critical_temp ~ ., data=train)
         summary(fit1)
```

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

Residuals:
     Min       1Q   Median       3Q      Max
-133.288  -12.609   -0.441   12.817  185.090

Coefficients:
                                  Estimate Std. Error t value Pr(>|t|)
(Intercept)                     -7.840e+01  5.034e+00 -15.575  < 2e-16 ***
mean_atomic_mass                 2.242e-01  1.441e-02  15.557  < 2e-16 ***
wtd_range_atomic_mass           -2.420e-01  1.452e-02 -16.664  < 2e-16 ***
std_atomic_mass                  2.945e-01  1.415e-02  20.810  < 2e-16 ***
mean_fie                         1.017e-01  4.781e-03  21.272  < 2e-16 ***
wtd_gmean_fie                   -3.493e-02  3.671e-03  -9.514  < 2e-16 ***
mean_atomic_radius               2.136e-01  1.736e-02  12.302  < 2e-16 ***
wtd_range_atomic_radius         -1.063e-01  8.964e-03 -11.854  < 2e-16 ***
std_atomic_radius                2.223e-01  1.876e-02  11.851  < 2e-16 ***
mean_Density                    -3.554e-03  1.913e-04 -18.581  < 2e-16 ***
wtd_entropy_Density              4.686e+00  1.311e+00   3.573 0.000354 ***
```

```
wtd_range_Density                3.168e-03  1.641e-04  19.308  < 2e-16 ***
wtd_std_Density                 -2.613e-03  1.788e-04 -14.613  < 2e-16 ***
wtd_gmean_ElectronAffinity      -9.111e-02  1.097e-02  -8.309  < 2e-16 ***
entropy_ElectronAffinity         1.534e+01  1.741e+00   8.810  < 2e-16 ***
wtd_entropy_ElectronAffinity    -4.252e+01  1.922e+00 -22.127  < 2e-16 ***
wtd_range_ElectronAffinity      -2.133e-01  1.746e-02 -12.219  < 2e-16 ***
std_ElectronAffinity             1.427e-01  1.437e-02   9.935  < 2e-16 ***
mean_FusionHeat                  2.291e-01  3.237e-02   7.078 1.52e-12 ***
std_FusionHeat                  -5.559e-01  3.477e-02 -15.991  < 2e-16 ***
mean_ThermalConductivity         2.159e-01  1.346e-02  16.041  < 2e-16 ***
wtd_gmean_ThermalConductivity   -3.121e-01  1.108e-02 -28.161  < 2e-16 ***
entropy_ThermalConductivity      1.331e+01  1.473e+00   9.036  < 2e-16 ***
wtd_entropy_ThermalConductivity  1.450e+01  1.382e+00  10.495  < 2e-16 ***
wtd_range_ThermalConductivity    2.547e-01  1.059e-02  24.047  < 2e-16 ***
std_ThermalConductivity          3.816e-02  1.460e-02   2.615 0.008943 **
gmean_Valence                    1.581e+00  3.817e-01   4.142 3.46e-05 ***
range_Valence                   -3.151e+00  1.653e-01 -19.067  < 2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 20.22 on 16982 degrees of freedom
Multiple R-squared:  0.6516,Adjusted R-squared:  0.651
F-statistic:  1176 on 27 and 16982 DF,  p-value: < 2.2e-16
```

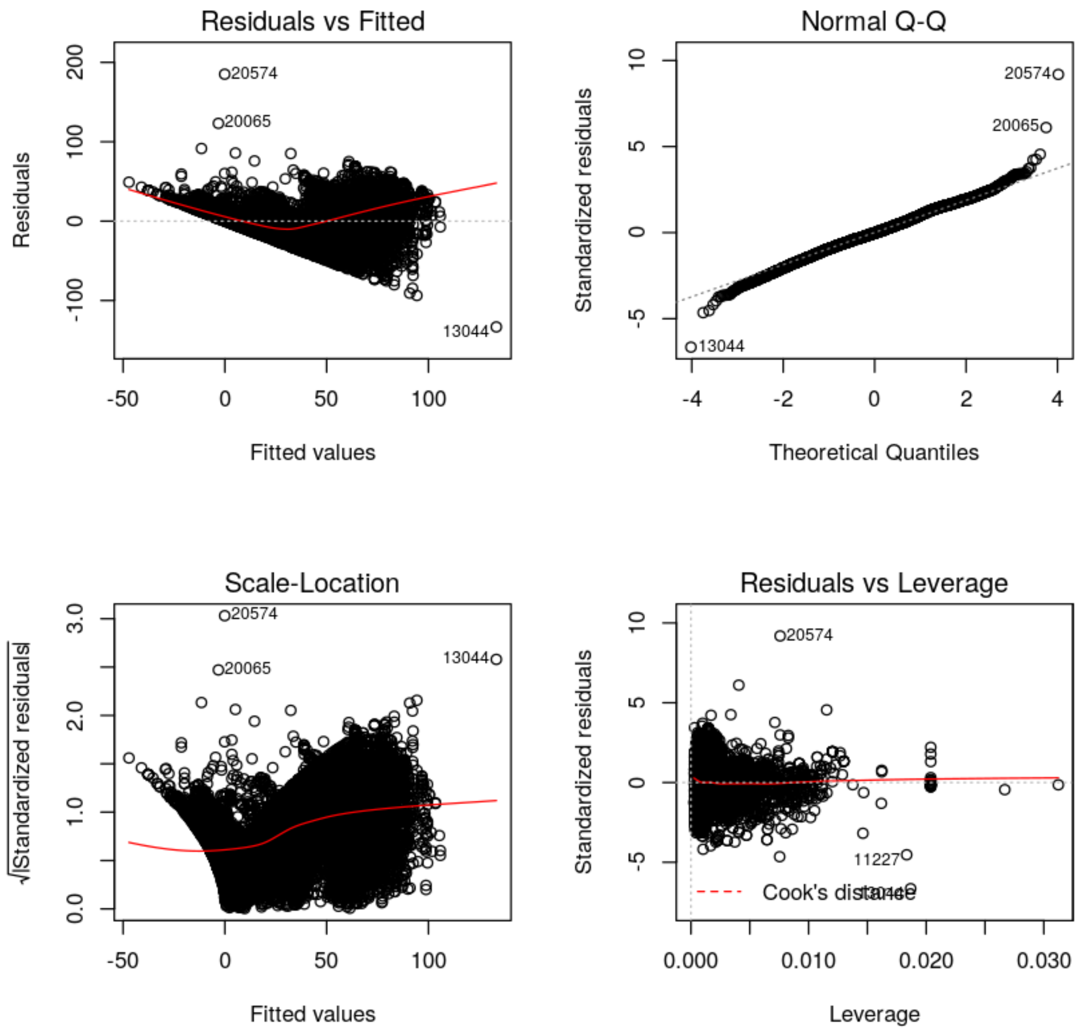
The adjusted R-squared (2) value indicates this model explains 65.1% of the variation in critical temperature.

The F-statistic 1176 has a p-value < 2.2e-16, so we can reject the null hypothesis (the model explains nothing) and accept the alternative which indicates that the model is useful.

The p-values for the coefficients show all the variables are significant at the 0.05 level.

Lets check the residuals using the plot function

```
In [58]: par(mfrow=c(2,2))
         plot(fit1)
```



From the above plots, we can observe the following:

- Residual vs Fitted - In this graph the residuals are not scattered evenly and there is a pattern observed which indicates that the relationship between the predictors and the response variable critical_temp is non-linear.
- Normal Q-Q - In the linear regression, we assume that the residuals are normally distributed with constant variance. This is validated using the Q-Q plot. The above Q-Q plot shows that most of the residuals follow the linear line but there are some outliers. This indicates that the residuals are normally distributed.
- Scale-Location - The graph shows the model violates the assumption of equal variance
- Residuals vs Leverage - The graph shows that all the data values are well inside the Cook's distance line which indicates that there are no influential outliers.

```
In [61]: glance(fit1) %>%
           dplyr::select(adj.r.squared, sigma, AIC, BIC, p.value)
```

A tibble: 1 Œ 5

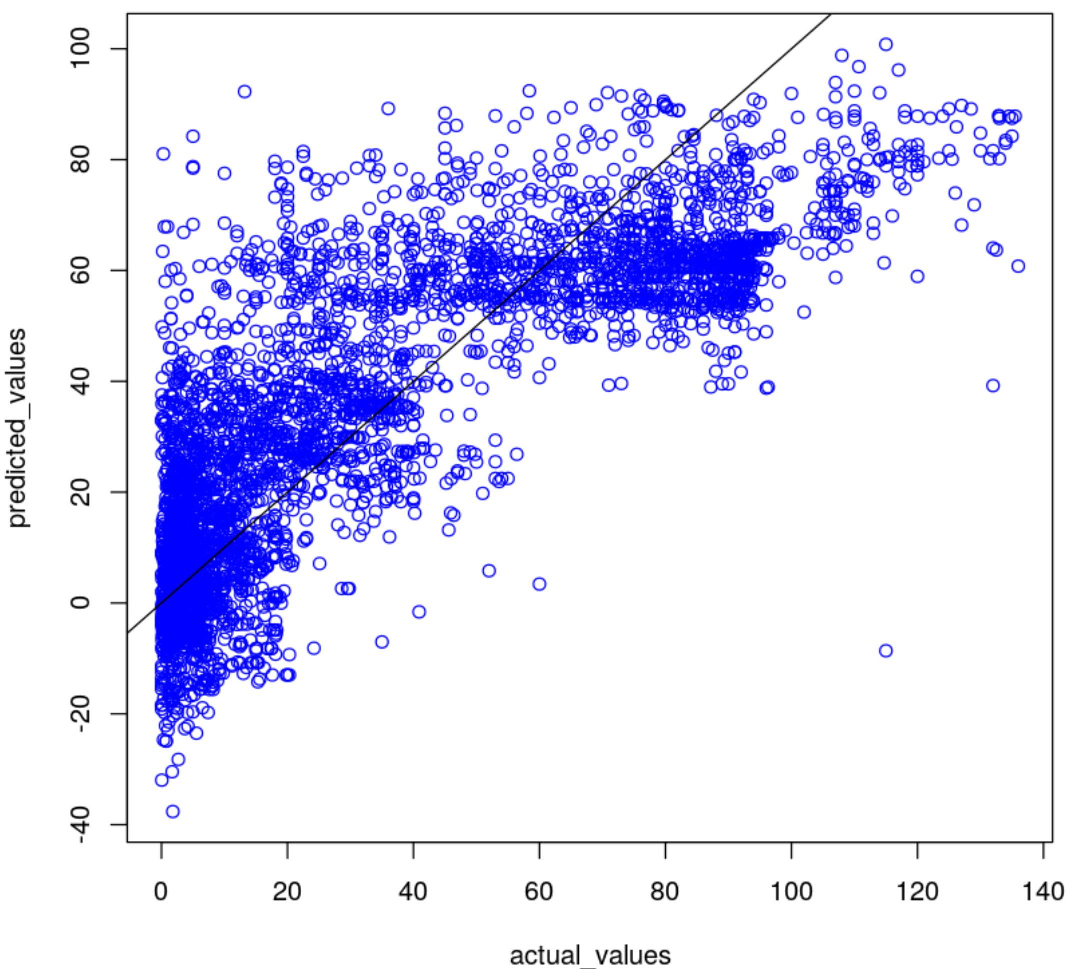
adj.r.squared <dbl>	sigma <dbl>	AIC <dbl>	BIC <dbl>	p.value <dbl>
0.6510073	20.22236	150593.2	150817.7	0

From the above table we can observe that for multiple linear regression model the **AIC** observed is **150593** and **BIC** observed is **150817**

Predicting critical temperatures of the test data

```
In [62]: pred_mod1 <- predict(fit1,test)
```

```
In [63]: plot(test$critical_temp,pred_mod1, col="blue", xlab="actual_values",ylab="predicted_va
         abline(a=0,b=1)
```



In the above graph, the X-axis represents actual values for critical temperature while Y-axis represents the predicted values of the critical temperature. It is observed that the predited values and observed values do not follow the linear regression line which indicates that linear model does not predict the test data accurately. It only predicts 65% of the variation in the data correctly.

```
In [64]: print(paste0("Linear regression model MSE:", mean((test$critical_temp-pred_mod1)^2)))
         print(paste0("Linear regression model RMSE:", RMSE(pred_mod1, test$critical_temp)))
```

```
[1] "Linear regression model MSE:410.123070773668"
[1] "Linear regression model RMSE:20.2514955194343"
```

The observed **MSE** for the linear model is **410.12** and the observed **RMSE** for the linear model is **20.25**

1.5.2 4.2 Lasso

Fitting the data to the Lasso model

In order to fit the lasso model to the data we convert the training and test data to corresponding matrices.

```
In [65]: train.mat <- model.matrix(critical_temp ~ ., data = train)[,-28]
         test.mat <- model.matrix(critical_temp ~ ., data = test)[,-28]
```

We generate a list of lambda values that will be used in cross-validation. With the generated list of possible lambda values, we fit the lasso model

```
In [66]: grid <- 10^seq(4, -2, length = 100)
```

The alpha is set to 1 for the lasso model and by default the cross validation uses 10 folds

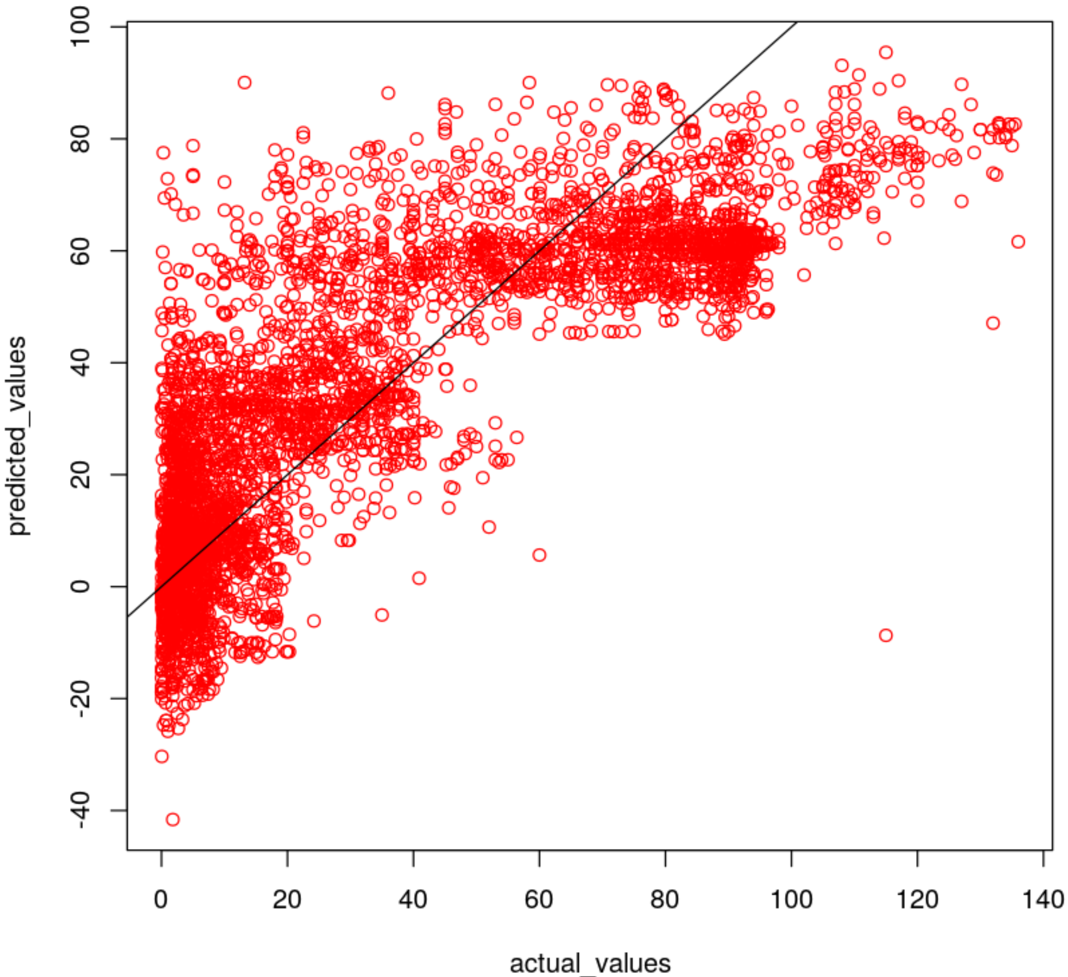
```
In [67]: set.seed(1)# the purpose of fixing the seed of the random number generator is to make
         fit.lasso <- glmnet(train.mat, train$critical_temp, alpha = 1, lambda = grid, thresh =
         cv.lasso <- cv.glmnet(train.mat, train$critical_temp, alpha = 1, lambda = grid, thresh
         bestlam.lasso <- cv.lasso$lambda.min
         bestlam.lasso
```

0.01

The best cross-validated lambda value is **0.01**. Using this value we will predict the test data over the lasso model

```
In [68]: pred.lasso <- predict(fit.lasso, s = bestlam.lasso, newx = test.mat)
```

```
In [69]: plot(test$critical_temp,pred.lasso, col="red", xlab="actual_values",ylab="predicted_va
         abline(a=0,b=1)
```



In the above graph, the X-axis represents actual values for critical temperature while Y-axis represents the predicted values of the critical temperature. It is observed that the predited values and observed values do not follow the linear regression line which indicates that Lasso model performs similar to the linear model and it covers only 64% of variations in the data.

```
In [70]: print(paste0("Lasso model MSE:",mean((test$critical_temp - pred.lasso)^2)))
         print(paste0("Lasso model RMSE:", RMSE(pred.lasso, test$critical_temp)))
         print(paste0("Lasso model R^2:", R2(pred.lasso, test$critical_temp, form = "traditiona
```

```
[1] "Lasso model MSE:421.305009140453"
[1] "Lasso model RMSE:20.5257158009277"
[1] "Lasso model R^2:0.642838762203165"
```

The observed **MSE** for the Lasso model is **421.30**

The observed **RMSE** for the Lasso model is **20.52**
The observed **R^2** for the Lasso model is **0.6428**

```
In [71]: predict(fit.lasso, s = bestlam.lasso, type = "coefficients")[1:28, ]
```

(Intercept) -86.091449028823 **(Intercept)** 0 **mean_atomic_mass** 0.251091098024042 **wtd_range_atomic_mass** -0.261214248761883 **std_atomic_mass** 0.286599719929642 **mean_fie** 0.10550139080775 **wtd_gmean_fie** -0.0367930540823694 **mean_atomic_radius** 0.22734084644434 **wtd_range_atomic_radius** -0.0946861826453636 **std_atomic_radius** 0.245793022460409 **mean_Density** -0.0037639340029349 **wtd_entropy_Density** 5.44645996718091 **wtd_range_Density** 0.00327847838037482 **wtd_std_Density** -0.00288283043214771 **wtd_gmean_ElectronAffinity** -0.0736995080719906 **entropy_ElectronAffinity** 7.03420052632276 **wtd_entropy_ElectronAffinity** -40.2393368080022 **wtd_range_ElectronAffinity** -0.222404174770319 **std_ElectronAffinity** 0.0877312791684606 **mean_FusionHeat** 0.294775389062425 **std_FusionHeat** -0.605835153926894 **mean_ThermalConductivity** 0.205507376715856 **wtd_gmean_ThermalConductivity** -0.306957040617012 **entropy_ThermalConductivity** 18.3938692161903 **wtd_entropy_ThermalConductivity** 8.81884051313616 **wtd_range_ThermalConductivity** 0.235197373667906 **std_ThermalConductivity** 0.089852054395355 **gmean_Valence** 1.77101665329915

Fitting the Lasso model did not reduce any features and showed that all the existing selected features are significant.

1.5.3 4.3 XGBoost - Extreme Gradient Boosting

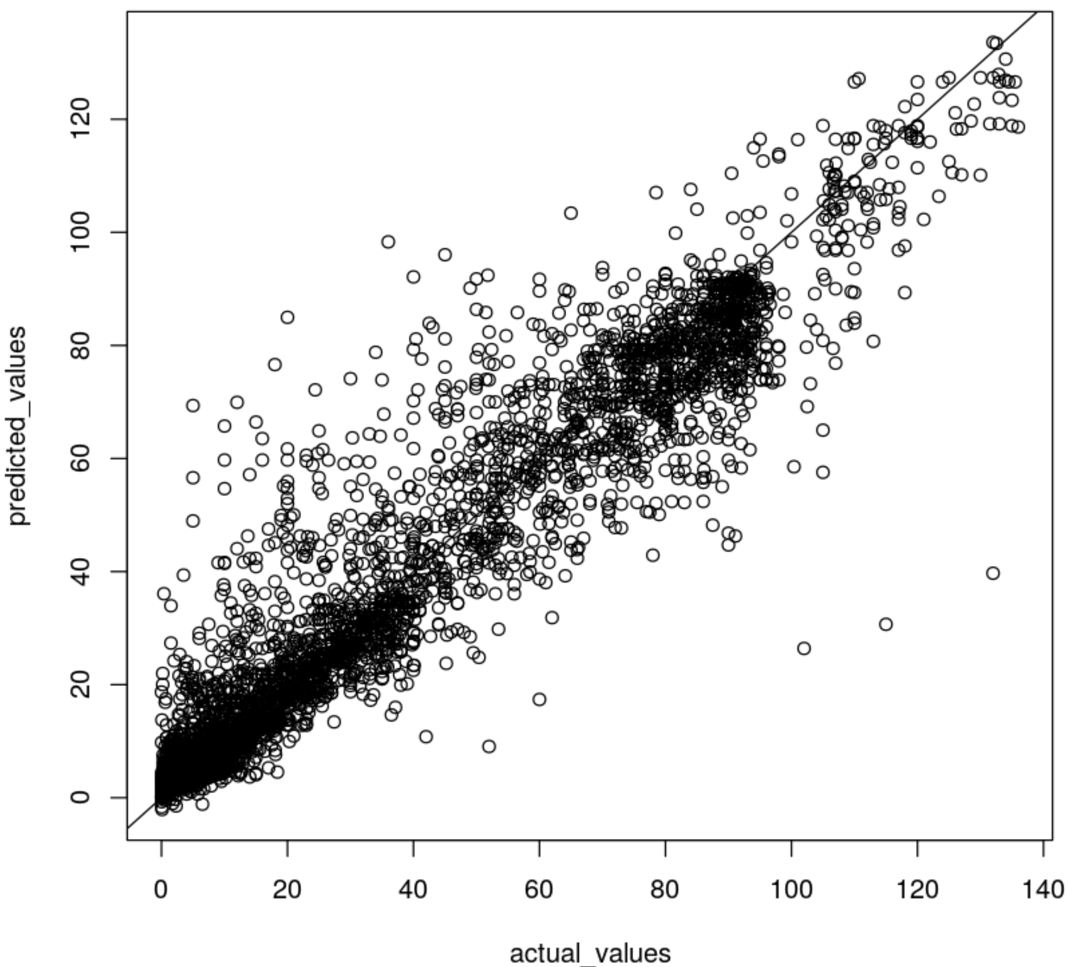
- XGBoost is a gradient boosting method that can be applied to linear as well as classification type of data.
- XGBoost uses the sequential decision trees to predict the values which makes it 10 times faster than all the other models.

We will be fitting the data to the xgboost model

```
In [72]: set.seed(123)
         ## Model parameters trained using xgb.cv function
         xgbFit <- xgboost(data = as.matrix(train[, -28]), nfold = 5, label = as.matrix(train$
             nrounds = 2200, verbose = FALSE, objective = "reg:linear", eval_metric = "rmse",
             nthread = 8, eta = 0.01, gamma = 0.0468, max_depth = 6, min_child_weight = 1.7817
             subsample = 0.5213, colsample_bytree = 0.4603)
```

```
In [73]: preds.xgb <- predict(xgbFit, newdata = as.matrix(test[, -28]))
```

```
In [74]: plot(test$critical_temp,preds.xgb, col="black", xlab="actual_values",ylab="predicted_
         abline(a=0,b=1)
```



In the above graph, the X-axis represents actual values for critical temperature while Y-axis represents the predicted values of the critical temperature. It is observed that the predited values and observed values do follow the linear regression line which indicates that XGBoost model fits the data extremely well by covering 91.4 % of the variations in the data.

```
In [75]: print(paste0("XGBoost model MSE:",mean((test$critical_temp - preds.xgb)^ 2)))
         print(paste0("XGBoost model RMSE:", RMSE(preds.xgb, test$critical_temp)))
         print(paste0("XGBoost model R^2:", R2(preds.xgb, test$critical_temp, form = "traditio
```

```
[1] "XGBoost model MSE:101.430409367693"
[1] "XGBoost model RMSE:10.071266522523"
[1] "XGBoost model R^2:0.914012390610035"
```

1.6 5. Model Comparsion

Comparsion between Linear model and Lasso Model

- By comparing the linear and lasso model, we can observe that there is not much difference observed in R^2 and RMSE value.
- The Linear model explains 65.16% of the data variation while the Lasso model explains 64.28%
- The RMSE of Linear model was observed to be 20.25 while the RMSE of the Lasso model was 20.52
- On comparsion it is evident that the Linear model performs much better than the Lasso model with less RMSE and increased R^2.

Comparsion between Linear, Lasso and XGBoost Model

- We compare all the three models based on R^2, MSE and RMSE values.
- XGBoost model outperforms with respect to all the models with R^2 value 0.91 which indicates that it covers 91% of the variations in the test data.
- The MSE and RMSE is observed to be decreased drastically to 101.43 and 10.07 respectively as compared to the other two models.
- Thus, we can choose XGBoost model to predict the critical temperature of the superconducting materials, given the chemial properties of the material.

1.7 6. Conclusion

- Here, we explored the superconductor data, determined which features contribute to predict the critical temperature and also built statistical models to predict the critical temperature of the superconducting material.
- We first explored the 4 main properties of superconductor such as Atomic Mass, Atomic Radius, Density and Fusion Heat which were chosen intuitively.
- With the help of box-plots we determined the predictors that can be candidates for elimination due to low variance.
- With the help of correlation plots we determined the predictors that can be candidates for elimination due to multicollinearity.
- We validated the chosen predictors for elimination during EDA with the help of R functions.
- After eliminating the predictors on the basis of low variance and high correlation we were left with 44 variables.
- We used backward stepwise selection method to further reduce the dimension of the data.
- This method gave us 27 variables which were highly contributing towards the prediction of the critical temperature.

- The major properties contributing towards predicting critical temperature of the superconducting material are: - Thermal Conductivity - Atomic radius - Electron affinity - Atomic Mass - Density - Valence
- We built 3 statistical models on these features and discovered that XGBoost model gives the highest accuracy with low root mean squared error as compared to others.

1.8 7. References

1. Matter, A., & Matter, A. (2019). Superconductivity and entropy. Retrieved 16 September 2019, from https://www.physicsforums.com/threads/superconductivity-and-entropy.700856/
2. (2019). Retrieved 16 September 2019, from https://arxiv.org/pdf/1803.10260.pdf
3. James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning (Vol. 112, p. 18). New York: springer.