Case Study: Superconductivity Critical Temperature Prediction

Megan Riley, Jaclyn Coate, Reagan Meagher

I. Introduction

Critical temperatures of metals can help scientists identify compounds that can be leveraged as superconductors. Superconductor metals allow for high flow of electrical current without overheating (maintaining a low temperature). The most well known usage of superconductive metal is for medical imaging machinery such as magnetic resonance imaging (MRI) systems. Preuss (2010) publishes with Berkeley Labs and discusses the potential of how superconductors could define our world moving forward with such things as: superefficient motors and generators; no transmission loss, long-distance, low-voltage electric grids; ultra-high-speed supercomputers; fast, magnetically levitated trains; inexhaustible fusion energy. Some of these solutions were already in the experimental and/or demonstration stages in 2010. Being able to identify those materials to be leveraged as superconductors by modeling and predicting the potential critical temperature, could change our future.

In the following case study, we examined the given dataset to predict the critical temperature of a compound given 81 variables and a secondary dataset containing the elemental components of the unknown compound. We leveraged linear regression to try and build a model that can best predict the critical temperature. We assessed the data and determined a transformed model and adding the highest correlated compounds would provide us with all features with high feature importance while also providing us with the lowest RMSE. We confirmed this by measuring the RMSE across potential models and their output outlined in our *Results* section.

Below we outline our method containing: exploratory data analysis, variable selection and validation tactics, models, feature importance, results, and discussion.

```r
library(tidyverse)
```

```
## -- Attaching packages --------------------------------- tidyverse 1.2.1 --

## v ggplot2 3.2.1     v purrr   0.3.4
## v tibble  2.1.3     v dplyr   1.0.3
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## Warning: package 'purrr' was built under R version 3.6.2

## Warning: package 'dplyr' was built under R version 3.6.2

## -- Conflicts ------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(ggplot2)
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```r
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
library(RColorBrewer)
library(ModelMetrics)
```

```
##
## Attaching package: 'ModelMetrics'

## The following object is masked from 'package:base':
##
##     kappa
```

```
library(mice)
```

```
## Warning: package 'mice' was built under R version 3.6.2

##
## Attaching package: 'mice'

## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

```
library(glmnet)
```

```
## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loaded glmnet 3.0-2
```

II. Data

In order to predict the critical temperature of our unknown compound we first need to review the datasets supplied. The referenced data was given to our team and we have stored and referenced it from our Github repository: 7333_Quantifying_The_World. The initial dataset will be termed our *'main'* dataset. It contains our previous critical temperatures and 80+ variables to help build our linear model for predicting the dependent variable: critical temperature. Additionally, we will use a secondary dataset which will be termed our *'chem'* dataset, it contains all the elements that make up our unknown compound and can be leveraged as additional variables for predictors.

With these two data sets we moved forward with an exploratory data analysis.

```
train_file =
  "https://raw.githubusercontent.com/JaclynCoate/7333_Quantifying_The_World/main/Unit4_CaseStudy2/data/t
unique_m_file =
  "https://raw.githubusercontent.com/JaclynCoate/7333_Quantifying_The_World/main/Unit4_CaseStudy2/data/u

train_main = read_csv(train_file)
```

```
## Parsed with column specification:
## cols(
##   .default = col_double()
## )

## See spec(...) for full column specifications.
```

```r
train_chem = read_csv(unique_m_file)
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   material = col_character()
## )
## See spec(...) for full column specifications.
```

III. Exploratory Data Analysis

The raw data files shared with our team were of two natures. The first is a full dataset with over 80 variables to choose from in order to model and predict critical temperature: *'main'*. The second; *'chem'*, is a full list of element compounds that are contained within the unknown compound we are currently tasked with predicting the critical temperature.

Initially we make sure we have a clean dataset, leveraging the mice package in R we are able to easily see we are working with a complete and clean dataset.

```r
print("Chem File NA Results: ")
```

```
## [1] "Chem File NA Results: "
```

```r
invisible(md.pattern(train_chem, plot = FALSE))
```

```
##  /\     /\
## {  `---'  }
## {  O   O  }
## ==>  V <==  No need for mice. This data set is completely observed.
##  \  \|/  /
##   `-----'
```

```r
print("Main File NA Result: ")
```

```
## [1] "Main File NA Result: "
```

```r
invisible(md.pattern(train_main, plot = FALSE))
```

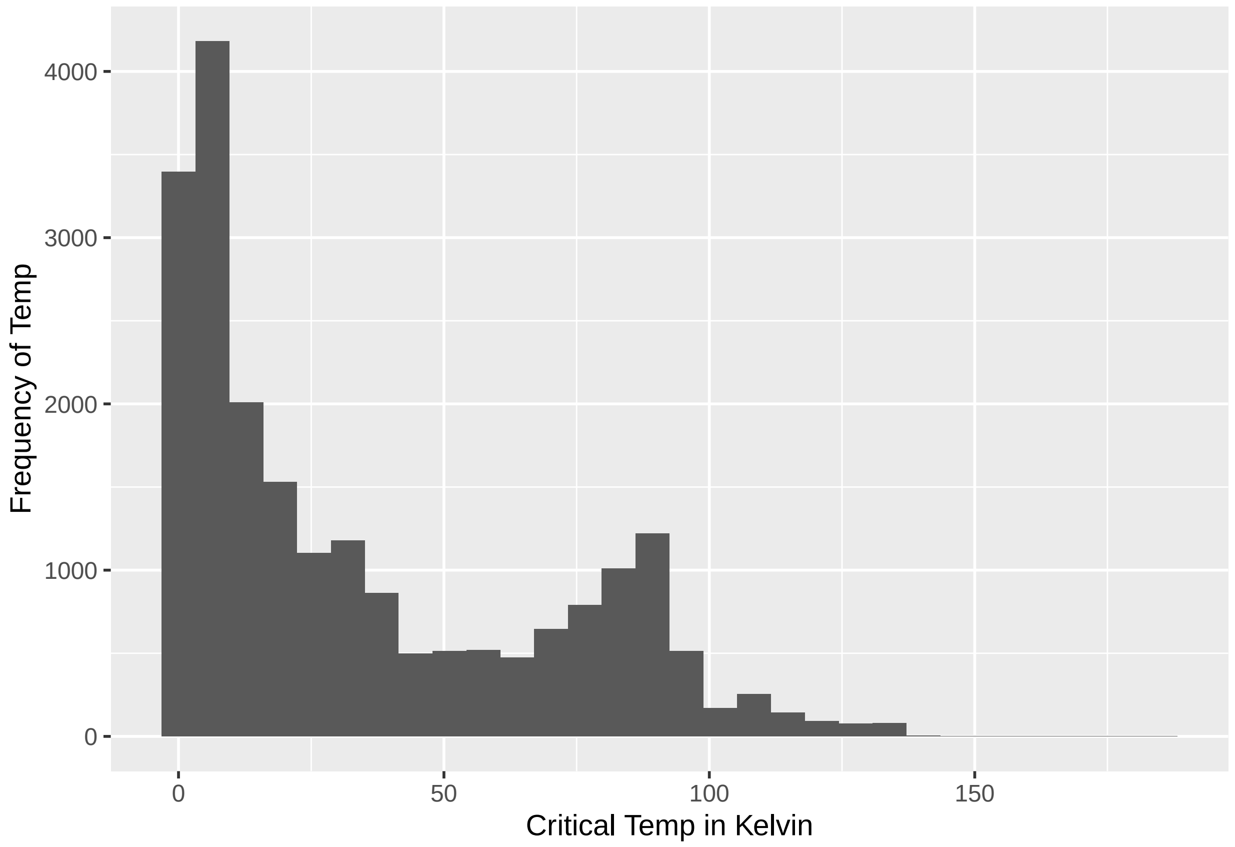
```
##  /\     /\
## {  `---'  }
## {  O   O  }
## ==>  V <==  No need for mice. This data set is completely observed.
##  \  \|/  /
##   `-----'
```

Our first evaluation of variables takes place on the dependent variable: *criticaltemperature*. When graphing the distribution of this variable we can see that see a very skewed set of data. In an attempt to correct the distribution for easier modeling. We move forward with a square rooted transformation and see we have a much more normal distribution of data, however, we do note a bi-modal distribution with what appears to be two small peaks.

We examined if there were any outliers that could easily be dropped to prevent skewing. While the graph overall is skewed, with a handful of larger temperatures, removing those outliers would influence the analysis. Many superconductive materials reach critical temperature near absolute zero, and while there are less materials that reach critical temperature at higher values, those are often the materials of interest as they make superconductive processes available with simpler methods. With that in mind we transformed the data without adjusting outliers.

```
ggplot(data = train_main, aes(x = critical_temp)) + geom_histogram() +
  labs(y = "Frequency of Temp", x = "Critical Temp in Kelvin")
```

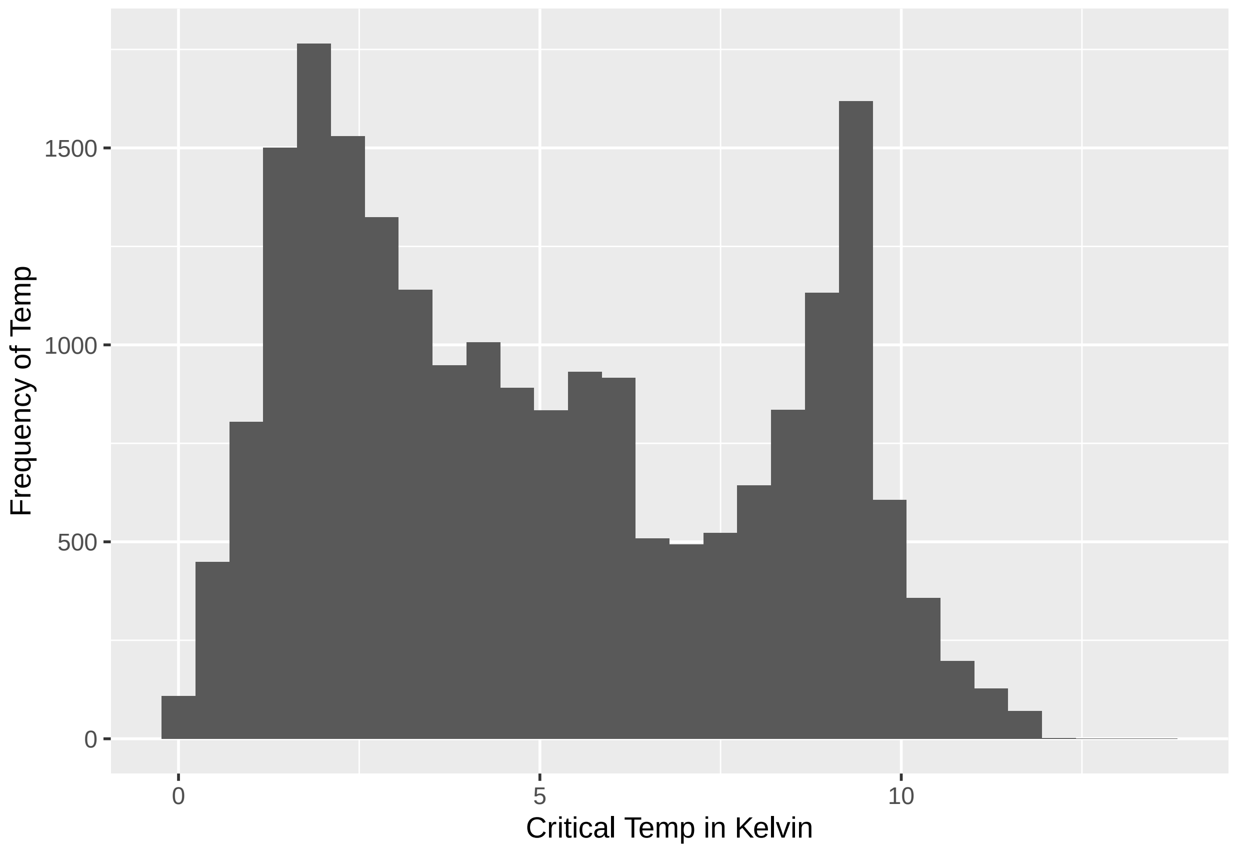
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



In addition to outliers, we examined the distribution of the variance and residuals. We do not see a perfect distribution, however, much better distribution than we saw before the transformation. Therefore we cautiously move forward with these assumptions met.

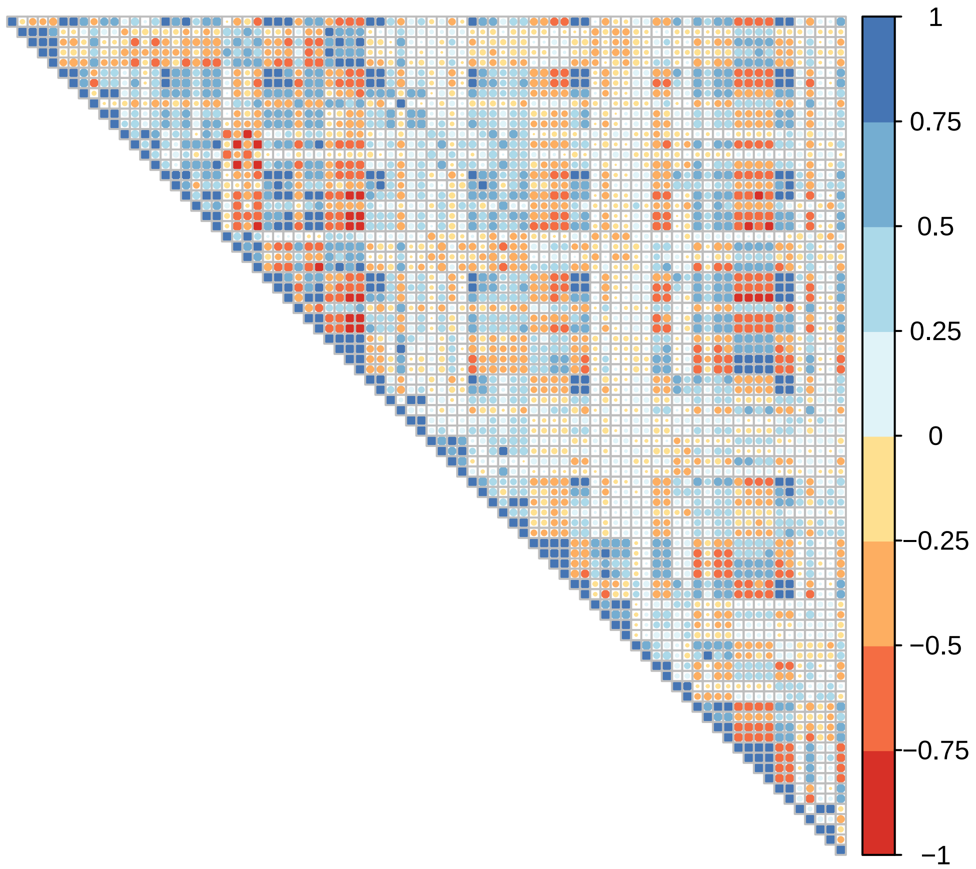
```
ggplot(data = train_main, aes(x = sqrt(critical_temp))) + geom_histogram() +
  labs(y = "Frequency of Temp", x = "Critical Temp in Kelvin")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



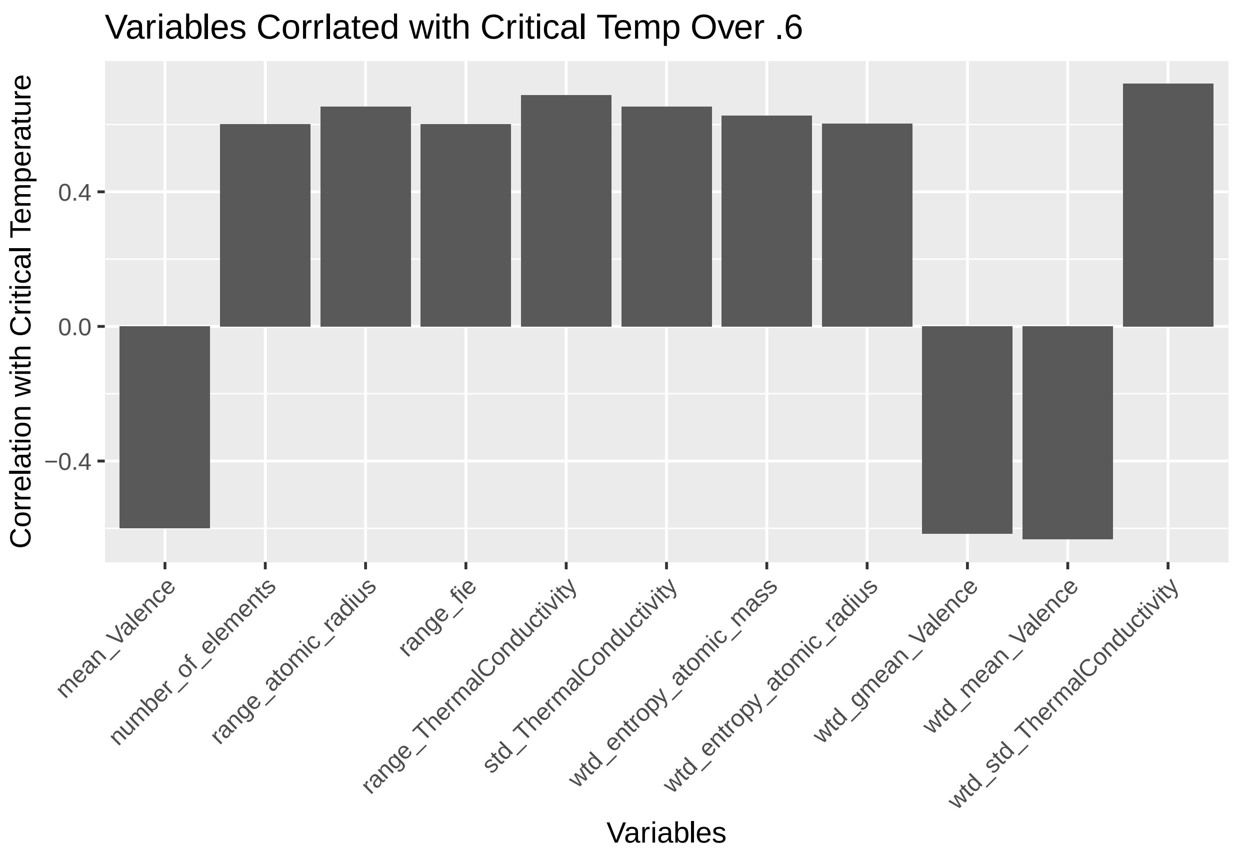
With an initial glance at the potential independent variables we can see there are core variables and then weighted versions of those variables. This instinctively tells us we are going to have an issue with multicollinearity and will need to assess the correlation between the variables. WIth our *main* dataset we do a large gradient grid to show us the correlation between those potential predictor variables and *criticaltemperature*

```r
M <-cor(train_main)

#Without labels, our last column is critical temperature.
corrplot(M, type="upper", order="original",
         col=brewer.pal(n=8, name="RdYlBu"), tl.pos = "n")
```

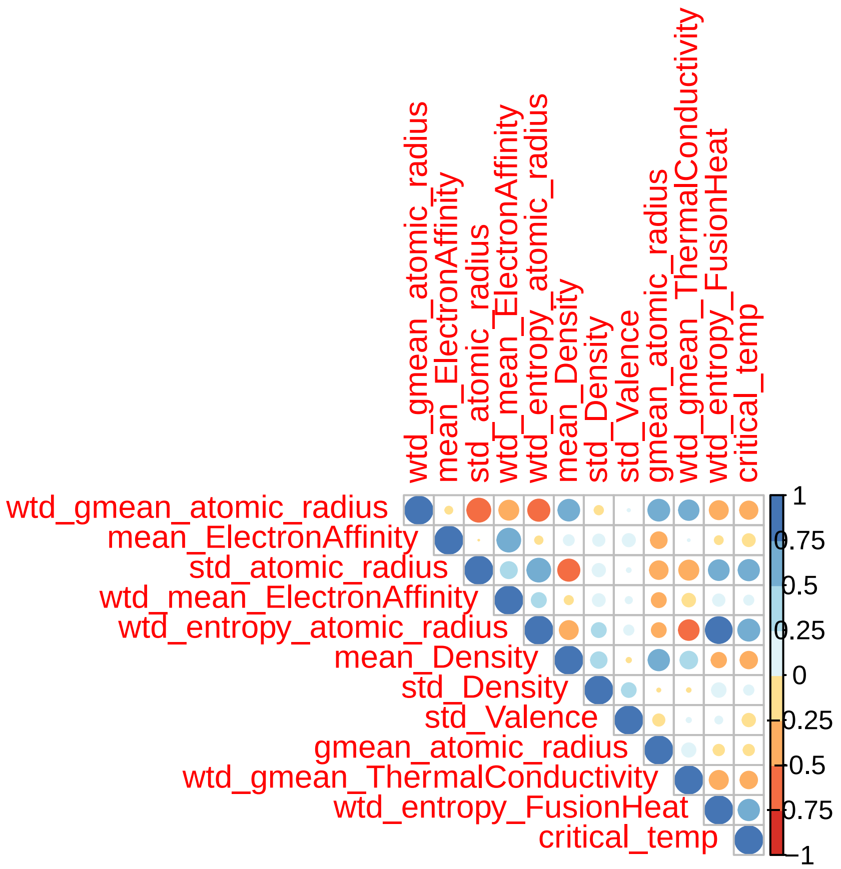


First we want to pull out those variables that are highly correlated with the independent variable. Once this is done we identify a top 11 variables that have a -.6 or .6 or higher correlation with *criticaltemperature*. Now we want to display that correlation between those independent variables to make sure they are actually independent as well as prevent any multicollinearity. In doing this we see that there is high correlation between many of the variables; however, primarily there is correlation between *std_entropy_atomic_radius* and almost every other predictor variable. We can either remove this single variable or keep it and remove all the additional variables. We choose to go with removing the additional variables in order to keep the model as interpretable as possible. Leaving us with the below list of predictor variables from our *main* dataset.

```
#Correlation with Critical Temperature
corr_temp = data.frame(vars = names(train_main[,-82]),
                       corr = cor(train_main[,-82],train_main$critical_temp))


corr_temp %>% filter(corr > .6 | corr < -.6) %>% ggplot(aes(x = vars, y = corr)) +
  geom_bar(stat = "identity") +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust=1)) +
  labs(y = "Correlation with Critical Temperature",x = "Variables") +
  ggtitle("Variables Corrlated with Critical Temp Over .6")
```



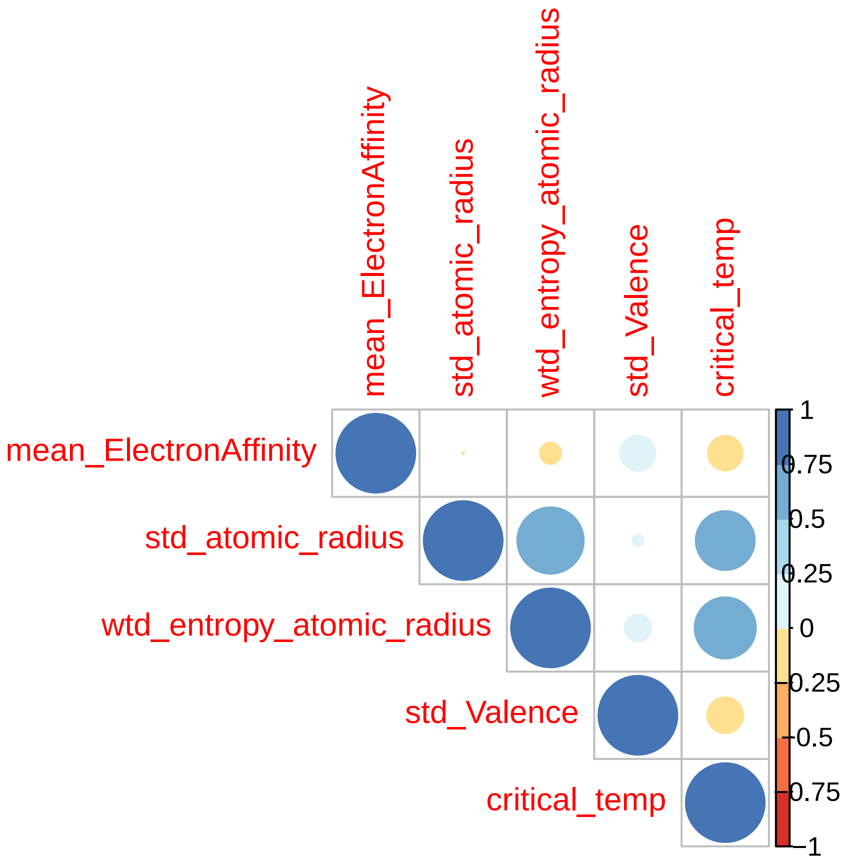
```
corr_var = corr_temp %>% filter(corr > .6 | corr < -.6)
corr_var_data = train_main[,c(corr_var$vars, 82)]
M_cor <-cor(corr_var_data)
corrplot(M_cor, type="upper", order="original",col=brewer.pal(n=8, name="RdYlBu"))
```



```r
drop_vars = c("wtd_gmean_atomic_radius","wtd_mean_ElectronAffinity",
              "wtd_entropy_FusionHeat", "wtd_gmean_ThermalConductivity",
              "gmean_atomic_radius","mean_Density", "std_Density")

corr_var_data_min = corr_var_data %>% select(-drop_vars)
```

```
## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(drop_vars)` instead of `drop_vars` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```

```r
M_cor <-cor(corr_var_data_min)
corrplot(M_cor, type="upper", order="original",col=brewer.pal(n=8, name="RdYlBu"))
```



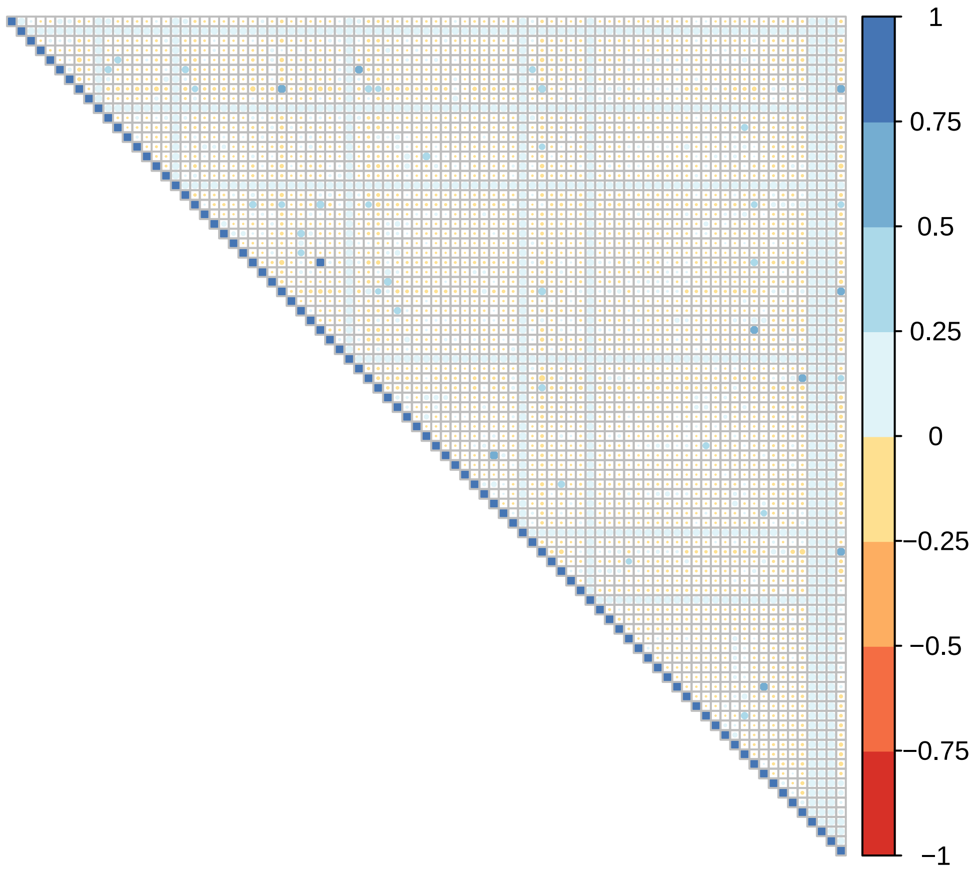
```
corr_var_data_min = corr_var_data_min %>% select(-wtd_entropy_atomic_radius)
```

We performed a similar exploratory analysis on the secondary dataset: *chem*. In our evaluation we again used a large gradient plot to review our correlation of elemental compounds against the *criticaltemperature* variable. We noticed four major elements that stand our and are highly correlated with the dependent variable: *O*, *Ca*, *Cu*, and *Ba* when we used a threshold of .5 to determine those variables the most important. Next we evaluated that multicollinearity between the element compounds. We did see some higher than desired correlation between the independent variables. However we hypothesize the correlation with critical temperature is too important with elements therefore we leave the four of interest. After model exploration we also found the multicolinearity did not adversely affect the effects of our features in the model.

```
M_1 <-cor(train_chem[,-88])
```

```
## Warning in cor(train_chem[, -88]): the standard deviation is zero
```

```
M_1[is.na(M_1)] <- 0
corrplot(M_1, type="upper", order="original",col=brewer.pal(n=8, name="RdYlBu"),
         tl.pos = "n")
```



```r
#Correlation with Critical Temperature
corr_chem_temp = data.frame(vars = names(train_chem[,-c(87,88)]) ,
                            corr = cor(train_chem[,-c(87,88)],train_main$critical_temp))
```

```
## Warning in cor(train_chem[, -c(87, 88)], train_main$critical_temp): the
## standard deviation is zero
```

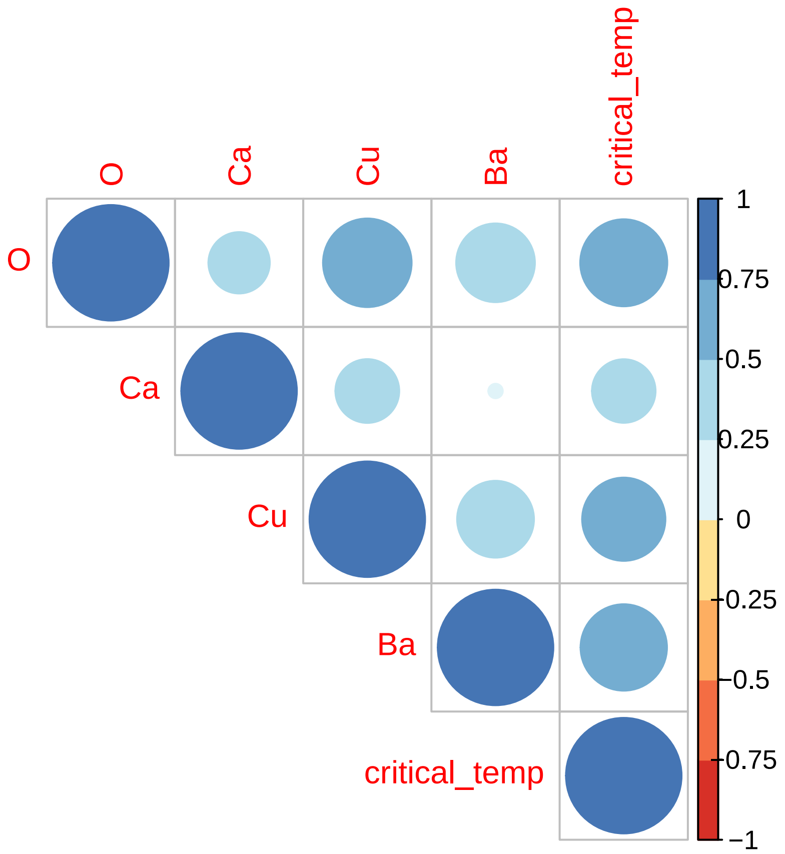
```r
tmp = corr_chem_temp %>% filter(corr > abs(.3))
#Top four chemicals correlated with critical temperature.O, Ca, Cu, Ba
print(tmp)
```

```
##    vars      corr
## O     O 0.5668517
## Ca   Ca 0.3022062
## Cu   Cu 0.5186202
## Ba   Ba 0.5587656
```

```r
chosen_var = c("O","Ca","Cu","Ba","critical_temp")
M_chem <-cor(train_chem %>% select(chosen_var))
```

```
## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(chosen_var)` instead of `chosen_var` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```

```r
corrplot(M_chem, type="upper", order="original",col=brewer.pal(n=8, name="RdYlBu"))
```



Our EDA has been able to help us with variable reduction and variable importance identification. Now we move forward with additional methods to confirm variable importance as well as review our validation methods for our multiple regression technique.

IV. Variable Selection and Validation

Based on our above exploratory data analysis we have been able to reduce our variable selection to the below lists making sure to account for multicollinearity.

List 4.1: Variable list from *main* dataset:

- mean_ElectronAffinity
- std_atomic_radius
- std_Density
- std_Valence

List 4.2: Variable list from *chem* dataset:

- O
- Ca
- Cu
- Ba

All models have to have a validation performance in order to determine if we have created a viable model to be used on additional unknown or unseen datasets. We completely held out ten percent of the data for our final model fit, prediction, and RMSE calculation for our results. Best practice is to never build a model based on all the data, due to data leakage and the potential of overfitting. We can be fairly confident that we did not overfit as our actual model was built on only the data on hand with our test split untouched until model validation and to compare RMSE. As we build our models, that train test split can be observed in each instance of model building at the final steps of *Section V. Model Type: Multiple Linear Regression*

V. Model Type: Multiple Linear Regression

There are many different ways to leverage data in order to build a model for data extrapolation. In this case study we leverage linear regression in order to try and determine the critical temperature of an unknown compound. Multiple linear regression is a powerful tool that uses a linear approach to modeling a relationship between a dependent and more than one independent variable. In moving forward with linear regression we have to be sure to meet certain assumptions that are listed below.

List 5.1: Variable list from *main* dataset:

- Normality of Residuals
  - When we review the residuals of no transformed data we see a severe skew of the data. When we perform a square root transformation we can elevate a much more evenly distributed residual output, but do note a bi-modal data distribution. We cautiously move forward with this assumption met.
- Constant Variance
  - When we review the variance distribution of no transformed data we see a severe skew of the data. When we perform a square root transformation we can elevate a much more evenly variance. We cautiously move forward with this assumption met.
- Independence
  - We assume independence of variables.
- Multicollinearity
  - With our large gradient graphs we can see that there are many variables that are correlated to both our dependent as well as each other. In *Section III: Exploratory Data Analysis* we have thoroughly assessed this issue and completed full variable selection that includes accounting for multicollinearity.
- Outliers
  - A quick examination of outliers shows no real data points that would be skewing the validated removal of any data points.

```r
set.seed(99)
#subseting main training file to just the least multi colinear var
train_subset = train_main %>% select(names(corr_var_data_min))

#total train set including chemicals
total_train = cbind(train_subset, "O" = train_chem$O, "Ca" = train_chem$Ca,
                    "Cu" = train_chem$Cu, "Ba"= train_chem$Ba)
#Building the sqrt version of temp
total_train$sqrt_temp = sqrt(total_train$critical_temp)

#Chem etc
total_train_transform = total_train %>% select(-critical_temp)



#No chem
total_train_nochem = train_subset
total_train_nochem$sqrt_temp = sqrt(total_train_nochem$critical_temp)
total_train_nochem = total_train_nochem %>% select(-critical_temp)
```

```r
#No transformation
total_train_notransform = total_train %>% select(-sqrt_temp)
```

List 5.2: MLR Models

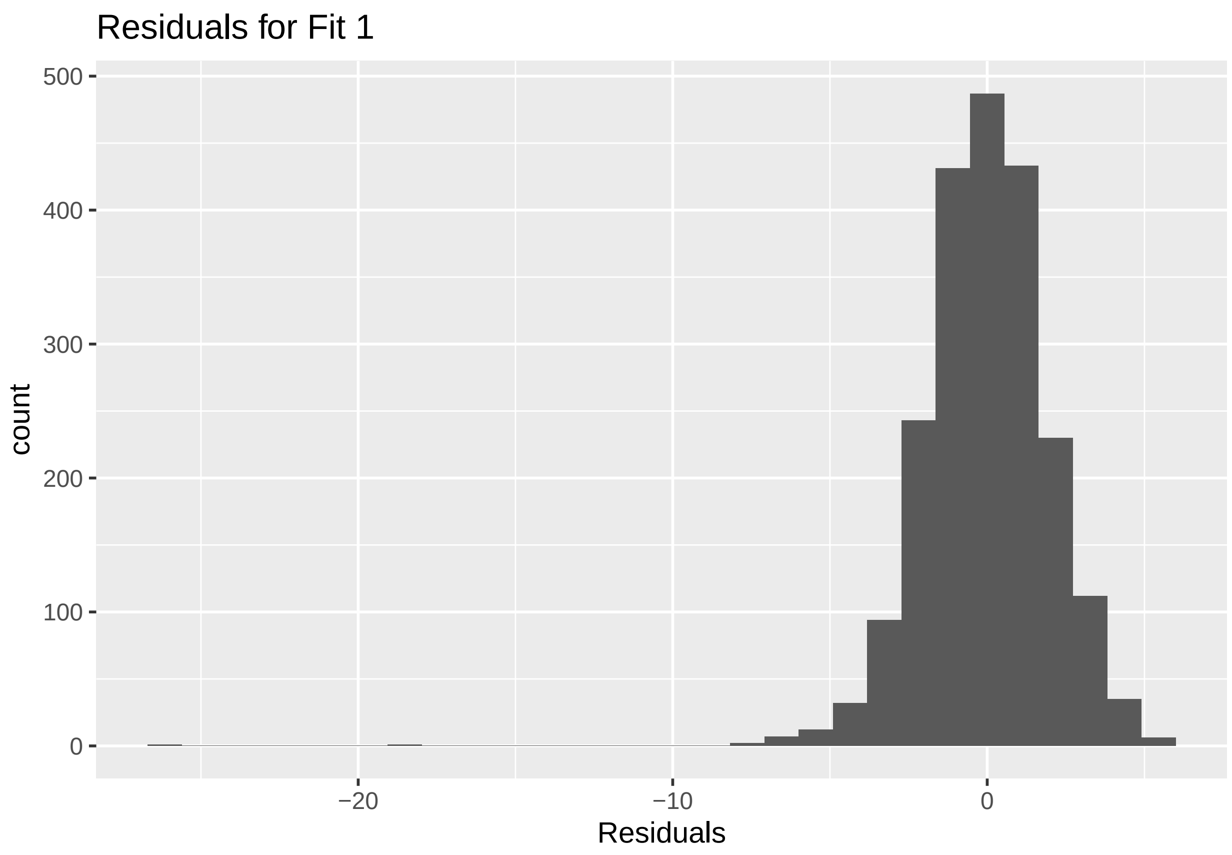
- Root squared transformed critical temperature variable and inclusion of highly correlated chemical compound from our *chem* dataset.
- Non transformed critical temperature variable and inclusion of highly correlated chemical compound from our *chem* dataset.
- Root squared transformed critical temperature variable and no inclusion of highly correlated chemical compounds from our *chem* dataset.

After our assumptions are met and we selected the variable inclusion based on those assumptions, we performed our cross validation methods as described in *Section IV: Variable Selection and Validation.*

Below we can see how our major models performs.

```r
#Model 1
df = total_train_transform
train_idx = sample(seq(1,dim(df)[1]),round(.9*dim(df)[1]))
temp_train = df[train_idx,]
temp_test = df[-train_idx,]
#Final Fit:
fit = lm(sqrt_temp ~ . ,data =temp_train)
predict = predict(fit, temp_test)

residuals = data.frame(resid = (temp_test$sqrt_temp - predict))
ggplot(residuals, aes(x = resid)) + geom_histogram()  + labs(x = "Residuals") +
  ggtitle("Residuals for Fit 1")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
rmse = rmse(temp_test$sqrt_temp, predict)
print("RMSE Fit 1 (squared): ")
```

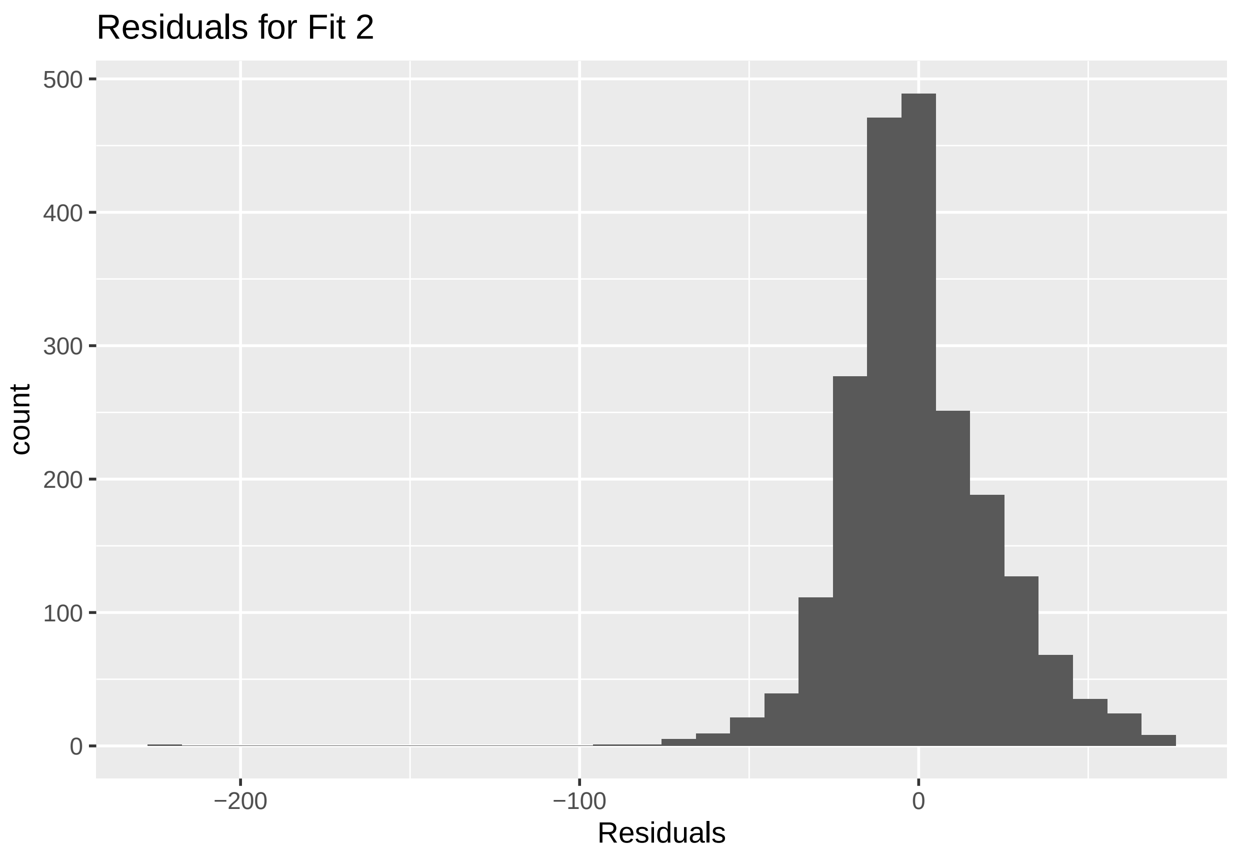
```
## [1] "RMSE Fit 1 (squared): "
```

```
print(rmse^2)
```

```
## [1] 4.017597
```

```
#Model 2
df = total_train_notransform
train_idx = sample(seq(1,dim(df)[1]),round(.9*dim(df)[1]))
temp_train = df[train_idx,]
temp_test = df[-train_idx,]

fit2 = lm(critical_temp ~ . ,data =temp_train)


predict = predict(fit2, temp_test)

residuals = data.frame(resid = (temp_test$critical_temp - predict))
ggplot(residuals, aes(x = resid)) + geom_histogram()  + labs(x = "Residuals") +
  ggtitle("Residuals for Fit 2")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
rmse2 = rmse(temp_test$critical_temp, predict)

print("RMSE Fit 2 : ")
```

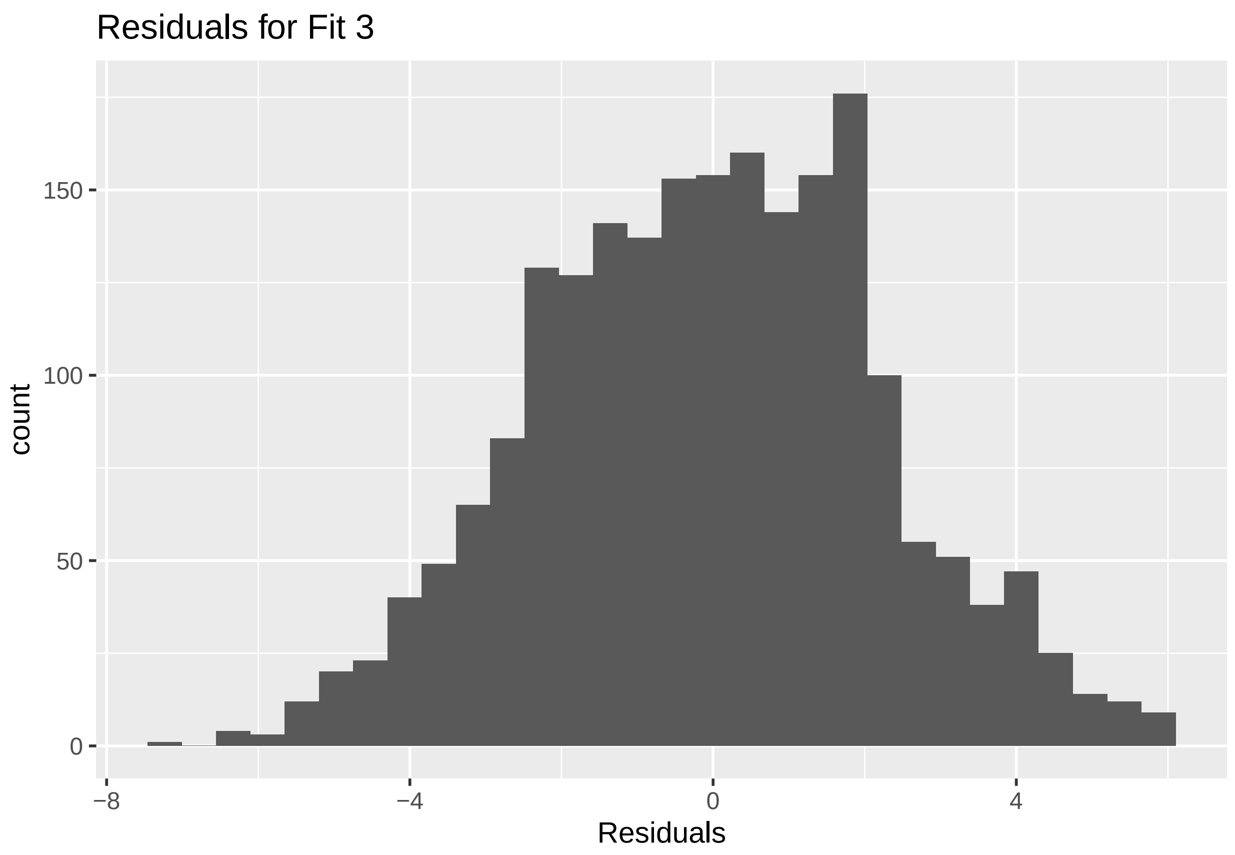
```
## [1] "RMSE Fit 2 : "
```

```
print(rmse2)
```

```
## [1] 21.99502
```

```
#Model 3
df = total_train_nochem
train_idx = sample(seq(1,dim(df)[1]),round(.9*dim(df)[1]))
temp_train = df[train_idx,]
temp_test = df[-train_idx,]

fit3 = lm(sqrt_temp ~ . ,data =temp_train)


predict = predict(fit3, temp_test)

residuals = data.frame(resid = (temp_test$sqrt_temp - predict))
ggplot(residuals, aes(x = resid)) + geom_histogram()  + labs(x = "Residuals") +
  ggtitle("Residuals for Fit 3")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```r
rmse3 = rmse(temp_test$sqrt_temp, predict)
print("RMSE Fit 3 (squared): ")
```

```
## [1] "RMSE Fit 3 (squared): "
```

```r
print(rmse3^2)
```

```
## [1] 5.118858
```

VI. Feature Importance

Below we have listed the summary of coefficients of our variables for our best model predicting the transformed square root of the critical temperature.

After our EDA we chose seven important variables. Each have importance and significance in the model at hand. Three of these variables were chosen from the main set of training variables after selecting for the variables most correlated with critical temperature and removal of variables with multi colinearity involved to reduce interpretability issues. With less multicollinearity issues in the chemical variables we focused on the top chemicals with correlation present to critical temperature.

In effect one of the strongest variables is the presence of Barium (Ba). With each additional unit of Barium, all other variables in consideration being equal, we expect the square root of the critical temperature to rise by .579 K. Calcium (Ca) is another important chemical in this regard, with a similar effect of .555 K rise in the square root temperature for each unit of Calcium. Oxygen(O) and Copper(Cu) are similarly positively effective in temperature with less extreme effects on the square root of critical temperature. Oxygen offers an increase of .126 K while Copper offers an increase of .156 K.

Overall the more diverse set of variables from the main training file offer less strong effects on predicting the square root of critical temperature. Standard valence has one of the strongest effects, where an increase of one element of standard valence on a superconductive material has a negative effect lowering the square root of critical temperature by .45 K, other variables held equal.

```
print("Coefficients of Best Model: ")
```

```
## [1] "Coefficients of Best Model: "
```

```
print(fit$coefficients)
```

```
##          (Intercept) mean_ElectronAffinity     std_atomic_radius
##           2.76571923           -0.01389776            0.05144572
##          std_Valence                     O                    Ca
##          -0.45176640            0.12650249            0.55541430
##                   Cu                    Ba
##           0.15631953            0.57926259
```

VII. Results

In *Table 7.1: Model Performance* we have output our RMSE by model for review. This allows us to see which linear regression model outputs the lowest loss, or in our case, lowest RMSE. The transformed critical temperature variable including chemical compound variable components performed the best, with the lowest RMSE of 3.57.

Table 7.1: Model Performance

Model	RMSE
Trans w/ Chem	4.02
Trans w/o Chem	21.99
NoTrans w/ Chem	5.12

VIII. Conclusion

We make a final conclusion on our analysis by comparing the RMSEs by model in *Table 7.1: Model Performance*. The data analysis indicates the best model, showing the lowest RMSE, is the model with a transformed outcome variable containing selected variables from both the *main* and *chem* datasets. Additionally, we review the summary output of our models. When reviewing the p-value we are able to determine if the relationship we have observed in the sample data provided would also exist in the larger population. As we review our summary output for our transformed model we observe that every chosen predictor variable has a p-value lower than a 0.05 significance level, indicating that those selected are all variables of statistical importance to our model.

Interpretation of coefficients allows us to state relevant changes of one statistically significant variable in comparison to our dependent variable *criticaltemperature*. For example, a change in Copper(Cu) of one part in the superconductive material, holding all other variables constant, can change the square root of the critical temperature by .156 K.

This being stated, we want to call attention to the fact that our model containing no dependent variable transformation produced an RMSE near to the best performing model, losing only one point in the RMSE. Some could argue that for the sake of interpretability (not needing to back transform to interpret coefficients), it could be worth losing 1 point of loss measurement by having a highly and easily interpretable model. A precise model predicting superconductor critical temperatures of newly created materials is an important tool for future chemical research. Therefore, we would recommend the lowest RMSE accounting for highest loss. Predicting superconductive materials that reach critical temperatures is a highly important and precise experimentation. An argument for the highest loss accounted for and moving forward with the most accurate predictions possible. Moving forward with these models can allow all researches to attempt to reach above absolute zero and open options for future scientific advantages.

IX. References

Preuss, Paul. SUPERCONDUCTORS FACE THE FUTURE. (2010, September 10). States News Service.

Nolan, D. and Lang, D.T. (2015). Data Science in R A Case Studies Approach to Computational Reasoning and Problem Solving. CRC Press.

Hamidieh, Kam, A data-driven statistical model for predicting the critical temperature of a superconductor, Computational Materials Science, Volume 154, November 2018, Pages 346-354, Web Link.