REGRESSION ANALYSIS
GUIDELINES IN SAS AND R
A CASE STUDY

Sarah Milligan
SARAH MILLIGAN  Bridgewater State University
Guidelines for Regression Analysis in SAS and R: A Case Study

Sarah Milligan

Submitted in Partial Completion of the Requirements for Departmental Honors in Mathematics

Bridgewater State University

May 11, 2021

Dr. Uma Shama, Thesis Advisor

Dr. Wanchunzi Yu, Thesis Advisor

Dr. Mahmoud El-Hashash, Committee Member
## Contents

Abstract: ......................................................................................................................... 3

Introduction: ....................................................................................................................... 3

Background: ......................................................................................................................... 4

What is R? ............................................................................................................................ 5

Using R: ............................................................................................................................... 5

What is SAS? ......................................................................................................................... 6

Using SAS: ........................................................................................................................... 6

What is Regression Analysis? .............................................................................................. 8

Assumptions in Regression Analysis .................................................................................. 8

Normal Distribution (Normality) ......................................................................................... 8

Constant Variance ............................................................................................................. 9

NBA Salary Data Set: .......................................................................................................... 12

Steps for Regression Analysis in R and SAS: .................................................................... 13

Importing Dataset into R..................................................................................................... 13

NBA Salary Example in R: ............................................................................................... 14

Importing Dataset into SAS ............................................................................................... 16

NBA Salary SAS Example: ............................................................................................. 18

Preliminary Investigation .................................................................................................... 19

Finding the Original Model............................................................................................... 19

NBA Salary Example in R: ............................................................................................... 19

NBA Salary Example in SAS: .......................................................................................... 20

Checking Model Assumptions: ........................................................................................ 22

Normality Assumption: ..................................................................................................... 22

NBA Salary Example in R: ............................................................................................... 23

NBA Salary Example in SAS: .......................................................................................... 24

Constant Variance Assumption ....................................................................................... 27

NBA Salary Example in R: ............................................................................................... 27

NBA Salary Example in SAS: .......................................................................................... 28

Coefficient of Multiple Determination ($R^2$) .................................................................... 30

NBA Salary Example in R: ............................................................................................... 31

NBA Salary Example in SAS: .......................................................................................... 31

Adjusted Coefficient of Determination ($\text{Adj}R^2$).......................................................... 32
Abstract:
When a player is a free agent, an individual who is able to sign to any team, one wonders what their best option is. Will signing with Team A or Team B provide them with the largest salary? What factors will affect their salary the most? Does last year’s statistics have a strong impact on next year’s salary? These questions can be answered by performing a regression analysis on previous years data. The primary focus of this project is to determine the most important variables related to an NBA salary. Likewise, the statistical programs SAS and R will be compared in their efficiency along with the importance of using two programs. Additionally, a comprehensive guideline of completing a regression analysis will be developed. Specifically, the use of a Box-Cox Transformation will be considered along with the use of t-Tests, a Breusch-Pagan test (constant variance), and many more statistical tests and methods.

Acknowledgements:
I would like to express my sincere gratitude to the professors within the Mathematics Department at BSU. Specifically, to my two thesis advisors Dr. Uma Shama and Dr. Wanchunzi Yu. Every member of the BSU Mathematics department has helped me developed key skills for progressing towards my goal of becoming a Biostatistician.

I would also like to acknowledge the overwhelming support from my family members. To my parents, thank you for pushing me to do my best and teaching me success comes to those who strive for it. To my siblings, for inspiring me for the case study of my thesis work. To my Great Aunts, for supporting and encouraging me throughout my undergraduate career. To my Uncle Rusty, for believing in my abilities and inspiring my career path through his current work. To my Nana and Papa, for being continuously involved and interested in my education. Each one of you has positively contributed to my success, thank you all!

Introduction:
This research paper discusses multiple statistical strategies that can be implored for a Regression Analysis and will act as a guideline for any Regression Analysis applicable to any dataset. The case study that will be used as an example for each method will be based on NBA Statistics. The body and meaning of the data will show how skills can be used and interpreted on real life data. After discussing each step in Regression Analysis in a broad sense, an example in reference to the NBA Dataset will be shown.
Background:
The Basketball Association of America was formed in 1946, though after 3 years it merged with the National Basketball League which was renamed as the National Basketball Association (NBA). The League had its ups and downs due to financial loss, lack of interest, and competition from the National Baseball League. Though, overtime the popularity of the competition grew, particularly in the 1980’s. This is largely in part due to two of the most famously well-known basketball players of that era, Magic Johnson, and Larry Bird. These amazing players brought fun to the NBA. This pattern grew into the 1990’s with the age of Michael Jordan. “Jordan became the NBA and perhaps even bigger than the league” itself (NBA History). With the increasing interest in the NBA, the income of those who were a part of it grew rapidly. This spike in salaries made the NBA go from a sport to a job for many people. Thus, the interest of becoming a valuable, well-paid, and popular player increased. To make the best choice on what team to sign a contract with, each player has an agent. This agent helps the player discuss trade options, increase their publicity, and most importantly help them receive the highest salary possible. When a team is deciding which player to sign to their roster they consider many key attributes. These attributes include a players position, their average points per game, their age, total minutes played, total rebounds, total steals, total blocks, total fouls, total assists, and total turnovers. Each one of these aspects helps a player to be categorized. Positions are also a key attribute. There are five positions in basketball, the point guard (PG), shooting guard (SG), small forward (SF), power forward (PF), and the center (C) as seen in figure 1. Average points per game takes the number of total points scored out of $n$ games and takes there average. A player’s age has a weight on the possibility of retirement from the league. The amount of games played reflects highly on the importance of each individual statistic. As expected, the more games an individual plays the more reliable the information is. A turnover is when a player mistakenly allows the ball to be taken by the other team, this is a negative statistic.

These ten explanatory variables, along with current team conference and current team ranking, will be used to build a regression line to accurately predict a player’s salary for a particular team. A response variable is the outcome of the study. Explanatory variables are those that may influence the response variable. Here the response variable is the salary.

Some of these explanatory variables will be dropped from the model if they are not statistically significant to the model. To discover this, we will use the statistical software’s SAS (Statistical Analysis System) and R. These guidelines will include the coding, reasoning, and implications in both software’s. Doing this regression analysis on the NBA will create a proper representation on using the guidelines itself while maintaining an easily understandable project.
In general, a regression model is written as $y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \varepsilon_i$. Here $y_i$ represents the response in the $i$th trial. $\beta_0, \beta_1$ are the unknown parameters/regression coefficients, $x_i$ represents the value of the $i$th explanatory variable, $\varepsilon_i$ represents the random error.

What is R?
R is a Statistical Program that is exploratory and intuitive. Multiple statisticians create packages that are used by other individuals. There are numerous techniques in R that will seem “easy” such as multiplication, addition, subtraction, etc. This is due to the fact that R can be used as a calculator. Refer to figure 1.1 to see how R can be used as a calculator. R can also be used to compute complicated and intense statistical analysis. In this paper we will explore how to use R to perform Regression Analysis.

Using R:
To display a result, like in figure 1.1, we use the IDE RStudio. RStudio has 4 main components. Refer to Figure 1.2 for a visual.

1. Source (top left) → This is where code is inputted.
2. Console (bottom left) → This is where the results from the code are displayed. At times, putting the code in the console will work too. For example, Figure 1.1 is from the console.
3. Environment (top right) → This is where data sets are imported. This is also where any assigned variable will “live”. This meaning if we assigned a to be equivalent to 5 it would be displayed in the environment as a=5.
4. Viewer/Plots/Help (bottom right) → The last section performs many tasks. This is where any plots that are created through coding are displayed. Additionally, this is where packages are downloaded. Lastly, if there is ever a time that you need a code or error explained you can simply type it into the help bar. Likewise,
you can write a question mark followed by what you need help with. For example, ?mean

![Image]

**Figure 1.2**

**What is SAS?**

SAS is a Statistical Program that is far less intuitive than R. This program calls for extreme memorization of coding. There are many layers and steps that need to be done during each session for coding to be successful. SAS university edition is accessed through online source and starting the Oracle VM VirtualBox on your desktop. For the purposes of this analysis, we will explore the use of SAS for performing a Regression Analysis.

**Using SAS:**

SAS university edition has one main space where code, outputs, and data sets are seen. This is the area on the right in Figure 2.1. On the left is the “Server Files and Folders”. This is where one can access their saved workbooks and datasets. Each time a code in SAS is used the related data set must be reimported or rerun.
Once a workbook is selected, the right side expands. In Figure 2.2 below shows the new visual appearance of the right-hand side.
There are now 3 components of the right-hand side.

1. CODE: this is where one would input the code that should be run to complete analysis
   a. The running man, circled in green, is what will be used to run code in SAS. You will highlight the code you wish to run and then press this figure.
2. LOG: this is where any errors in your code can be seen and explained. If there is an issue in your code you will be brought to this page.
3. RESULTS: This is where all plots and analysis will be done. If you have a lot of code your RESULTS page will be long.

**What is Regression Analysis?**

Regression Analysis is a form of statistical analysis. In any statistical analysis you have a data set with variables. The goal is to see how these variables interact. Specifically, in a Regression Analysis, the variables will be selected and analyzed to fit a certain model that predicts a certain outcome. For example, in this paper the example analysis that will be examined is predicting NBA Salaries. By the end of the analysis there will be a sufficient model that adequately predicts these salaries.

**Assumptions in Regression Analysis**

There are 2 main assumptions for Regression Analysis. Both assumptions rely on the understanding of residuals. Residuals are estimates of *errors*. This meaning it measures how well the model fits each individual data point and how the errors within the model are related to one another.

**Normal Distribution (Normality)**

A model follows a normal distribution when your data follows a bell-shaped curve. The goal is to have the errors of the model generally be near the mean of the errors. If the errors more frequently follow the mean rather than straying away, then the model follows a normal distribution. This can be checked by looking at the Normal Q-Q plot. In Figure 3.1 there are two examples of Normal Q-Q Plots. The plot on the right displays a plot that follows a normal distribution. This can be seen by examining where the quantiles (black dots) are. Since the quantiles are “close” to the diagonal line it is said the data follows a normal distribution. On the other hand, the plot on the left represents a model that does not follow a normal distribution.
Normal Distribution can also be tested by using the Shapiro Wilk Normality Test. The Null Hypothesis ($H_0$) is that the model follows a normal distribution. This test will present a p-value when ran. In order for the model to be successful, this meaning we do not reject the Null Hypothesis, the p-value must be large. Otherwise, we do not accept the Null Hypothesis.

<table>
<thead>
<tr>
<th>Software</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>R - plot</td>
<td><code>Plot(model, which=1)</code></td>
</tr>
</tbody>
</table>
| R – test | Name: Shapiro Wilk Normality Test  
Null Hypothesis ($H_0$) = The model follows a normal distribution  
Code: `shaprio.test(model$residuals)`  
Determining the results:  
Large p-value – We do not reject $H_0$  
Small p-value – We reject $H_0$ |
| SAS- plot | `ods graphics on`  
`proc glm data = mydata PLOTS = all;`  
`model y = x1 x2 x3 … x_n;`  
`ods graphics off;` |
| SAS- test | `proc univariate data = outsalary normal;`  
`var resid ;`  
`run;` |

**Constant Variance**

Constant variance is satisfied when the individual error for each predicted value, against its fitted, is constant. In other words, the error of the model is constant for each individual point. The Residual v Fitted Plot compares the residuals against their fitted values. Visually, this is indicated by random behavior. In other words when the residuals are randomly and evenly dispersed it can be assumed that constant variance is satisfied.
Although, when the residuals are behaving systematically (following a specific curve), it can be assumed the model does not satisfy constant variance. Figure 3.2 represents one plot that displays constant variance (bottom) and another that does not display constant variance (top).

![Residuals vs Fitted](image1)

**Figure 3.2**

Constant Variance can also be tested with the Breusch-Pagan Test or Levenes Test. The Null Hypothesis ($H_0$) of the BP-Test and Levenes Test is that constant variance is satisfied. This test also presents a $p$-value when ran. If the $p$-value is large, then $H_0$ is not rejected. Otherwise, $H_0$ is not accepted.

<table>
<thead>
<tr>
<th>Software</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>R-plot</td>
<td><code>Plot(model, which=2)</code></td>
</tr>
</tbody>
</table>
| R – test | `library(lmtest)`
  Name: Breusch-Pagan Test
  Null Hypothesis ($H_0$) = Model satisfies constant variance.
  Package: lmtest
  Code: `library(lmtest)`
  `bptest(model)` |
<table>
<thead>
<tr>
<th>Determining the result:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Large p-value – We do not reject $H_0$</td>
</tr>
<tr>
<td>Small p-value- We reject $H_0$</td>
</tr>
<tr>
<td>SAS- plot</td>
</tr>
<tr>
<td>ods graphics on</td>
</tr>
<tr>
<td>proc glm data = mydata PLOTS = all;</td>
</tr>
<tr>
<td>model $y = x_1 \ x_2 \ x_3 \ldots \ x_n$;</td>
</tr>
<tr>
<td>ods graphics off</td>
</tr>
<tr>
<td>SAS- test</td>
</tr>
<tr>
<td>data outsalary2 ;</td>
</tr>
<tr>
<td>set outsalary ;</td>
</tr>
<tr>
<td>aresid = abs(resid);</td>
</tr>
<tr>
<td>run;</td>
</tr>
<tr>
<td>proc glm data = outsalary2;</td>
</tr>
<tr>
<td>model aresid = $y = x_1 \ x_2 \ x_3 \ldots \ x_n$;</td>
</tr>
<tr>
<td>run;</td>
</tr>
<tr>
<td>quit;</td>
</tr>
</tbody>
</table>
## NBA Salary Data Set:

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Variable Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>sal1920</td>
<td>This is the Y-Variable. This is the known salary for NBA players for the 2019-2020 season.</td>
</tr>
<tr>
<td>rank1920</td>
<td>This variable represents the teams rank during the 2019-2020 season.</td>
</tr>
<tr>
<td>conf</td>
<td>This variable represents the conference that each team is on. 1 denotes Eastern Conference and 0 denotes Western Conference</td>
</tr>
<tr>
<td>age1819</td>
<td>The age of each player during the 2018-2019 NBA season</td>
</tr>
<tr>
<td>pos</td>
<td>The position played by each player.</td>
</tr>
<tr>
<td>tmp1819</td>
<td>The total minutes played by each player during the 2018-2019 NBA season</td>
</tr>
<tr>
<td>reb1819</td>
<td>The total number of rebounds by each player during the 2018-2019 NBA season</td>
</tr>
<tr>
<td>as1819</td>
<td>The total number of assists by each player during the 2018-2019 NBA season</td>
</tr>
<tr>
<td>st1819</td>
<td>The total number of steals by each player during the 2018-2019 NBA season</td>
</tr>
<tr>
<td>blk1819</td>
<td>The total number of blocks by each player during the 2018-2019 NBA season</td>
</tr>
<tr>
<td>to1819</td>
<td>The total number of turnovers by each player during the 2018-2019 NBA season</td>
</tr>
<tr>
<td>pf1819</td>
<td>The total number of personal fouls by each player during the 2018-2019 NBA season</td>
</tr>
<tr>
<td>tp1819</td>
<td>The total number of points by each player during the 2018-2019 NBA season</td>
</tr>
</tbody>
</table>
Steps for Regression Analysis in R and SAS:
Importing Dataset into R

Importing a data set into R is straight forward. The task of making sure there are no errors or issues with a data set is a harder task. To import a data set into R you select “Import Data Set” in the environment. From here select the source your data set will be selected from. The options for sources can be seen in Figure 4.1 below.

Figure 4.1

The most popular choice to use is excel. Excel is an easy file for R to interact with. It will understand the motivation of the variables in the data frame. When a data frame is imported into R it will open a new window. In this window the dataset can be examined. R “defines” each variable as either Character, Numeric, or other types.

R defines a variable as character when it is reading it as a categorical variable. A categorical variable represents groups of data. For example, a variable that describes gender, states, schools, etc. would be categorical variables. On the other hand, when R defines a variable as numeric it is reading it as a quantitative variable. A quantitative variable represents numeric values. For example, a variable that describes number of objects, number of occurrences, weight of an object, etc. would be a quantitative variable.

When a data set is “printed” into the console, different terms are used. For numeric variables R uses the terms factor(fct) and double(dbl). Specifically, a double is a variable that was originally read as a numeric value by R. A factor is a variable that has character values but is being read as numeric code.

Another important aspect of a data set are the explanatory variables and response variables. A response variable (y-variable), often referred to as the dependent variables, is what your other variables (x-variables) are used to find. An explanatory variable (x-variable), also
commonly referred to as predictor or independent variables, determine the outcome of the x-variable.

Now that the classification of variables is noted. Let us discuss if R misinterprets the meaning of a variable. When this happens, code is used to correct the error. If one variable is a list of dates this will need to be clarified for R also. There are other shifts beyond the ones listed below. For help from R to categorize these variables use the help tool in the bottom right quadrant.

<table>
<thead>
<tr>
<th>Variable type</th>
<th>Code to Create this Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Factor</td>
<td>mydata$variable &lt;- as.factor(mydata$variable)</td>
</tr>
<tr>
<td>Date</td>
<td>mydata$variable &lt;- as.datemy(data$variable)</td>
</tr>
</tbody>
</table>

**NBA Salary Example in R:**

Figure 4.2 below shows R when the dataset “NBASalary” was imported from excel. R opens another tab in the source (top left) section. Here you can examine the skeleton behind your data set and be sure R interpreted your variables correctly. Additionally, the dataset becomes a part of the environment.

In the NBASalary data set, one of the variables is “pos” which represents the different positions an NBA player can have. This is a categorical variable. Due to this, R needs to be told to read it as a factor. Figures 4.3a-c shows the process of turning the variable “pos” into a factor.
Figure 4.3a shows how R is interpreting position as a character. This makes sense as position is a categorical variable. Although, in order to run the analysis, we need to turn position into a factor.

Figure 4.3b above shows the code that was ran to convert the categorical variable “pos” into a factor. This will allow R to incorporate “pos” into the analysis. Behind the scenes R is separating all the categories within “pos”.

For this project there were six positions, Point Guard, Shooting Guard, Small Forward, Power Forward, Forward and Center. For those who follow basketball, there are normally 5 positions. Within the data set a plethora of players played both Small Forward and Power Forward, hence the development of the position Forward.

> NBA$pos <- as.factor(NBA$pos)
> NBA
  sal1920 rank1920 rankyn conf age1819 pos tmp1819
   1 4.77e6 1 1 1 30 C 1606
   2 2.01e6 1 1 1 31 C 0
   3 2.96e6 1 1 1 22 PF 882
   4 2.58e7 1 1 1 24 PF 2358
   5 3.17e6 1 1 1 29 PF 402
   6 1.56e7 1 1 1 29 PG 2272
   7 3.06e7 1 1 1 27 SF 2393
   8 1.45e6 1 1 1 27 SF 0
   9 2.56e6 1 1 1 32 SF 2091
  10 7.00e6 1 1 1 32 SF 1231

Figure 4.3b

Figure 4.3c above shows the output in the console after running the code in the source. To display the revised dataset, R uses a Tibble 441 x 14.

Within the Tibble , R labels the type of variable under each respective one. Under “pos” we now see <fct>. This means that the code was successful and “pos” is now being read as a factor. Be careful, this does not mean the variable was changed in the dataset itself. It means within this specific R script R will read it as a factor.
This Tibble also tells us important information regarding our dataset. The dimensions 441 x 14 tells us the dataset NBASalary has 441 observations and 14 variables, including the response variable.

**Importing Dataset into SAS**

When inputting a data set into SAS you first want to ensure your data set is in your “myfolder” on the left-hand side of the screen. If it is not there, go into the folder on your computer and put a copy into that folder. It should then appear on your SAS Server.

For this example, Practice Upload 1.xlsx is brought into the SAS server. To open this file, it can either be dragged to the right area or double clicked. The Figure 4.4 below is what SAS will look like upon the initial examination and importing of this data set.

![Figure 4.4](image)

In order to use the dataset in analysis one must assign the data set a name. SAS will automatically call the dataset “IMPORT”, as seen in Figure 4.5 below. This is not useful and must be changed. To change this, scroll down in the top “file information” section and press change under “OUTPUT DATA”. Figure 4.5 below shows the new tab that is opened. Here where it says Data set: IMPORT change the name to whatever is logical for the given dataset. Press save and then run the dataset (this is done my pressing the running man figure in Figure 4.4 circled in green).
Now that the dataset is successfully imported, just as in R, you want to check to ensure SAS labeled your variables correctly. Whether they are categorical or quantitative. To check this, press the RESULTS section and scroll down to the “Alphabetic List of Variables and Attributes”. As in Figure 4.6 there is a column labeled “Type” here one sees “num” (numerical/quantitative) and “char” (character/categorical). Sometimes the type will be correct but when using a categorical variable, one will need to use the CLASS notation to signify it is categorical variable. This will be explained in the NBA example below.
NBA Salary SAS Example:

Figure 4.7 shows the SAS output once the NBASalary dataset is imported into the workbook. The first step in making this a working dataset is to change its name by scrolling down to DATA OUTPUT under File Information and selecting change. Once change is selected a new window will open and the “Data Set= IMPORT” should be changed to “Data Set = NBASalary”. Then press save.

Under “The CONTENTS Procedure” SAS lists the number of observations and variables. Figure 4.8 displays these results. SAS reads the same number of observations as R did. This is a good sign. Sometimes it is beneficial to do analysis in two program to ensure the correct results are seen. One may also check the “Alphabetic List of Variables and Attributes” to ensure SAS read the variables as the correct type (refer back to Figure 4.6). As seen in Figure 4.6, position is being read as a categorical variable. To specify this, a CLASS statement will be ran. This will be seen in the preliminary investigation.
Preliminary Investigation
Finding the Original Model

It is important to see where the model stands in terms of specific variables within the dataset. If initial analysis indicates a lack of significance variable there would be an issue regarding the significance of the dataset to the response variable. By looking at the original model one is able to see where the variables stand and check important aspects of the model. The original model contains all of the variables within the dataset. To find the original model tell your software to run analysis on all variables.

NBA Salary Example in R:

When printing a model into your console, it first needs a name. For the purposes of generality, mod1 will be used. To assign the model name a backwards arrow (<-) or an equal sign (=) can be used.

The next aspect of printing a model is to connect the explanatory and response variable. To do this, first write the type of model you wish to produce. For regression analysis the most commonly used model types are Linear Models and Generalized Linear Models. Refer to the code below. Note: R is case sensitive!

<table>
<thead>
<tr>
<th>Type of Model</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generalized Linear Model</td>
<td>mod1 &lt;- glm(Y~X1+X2+X3+...+Xn, data=mydata)</td>
</tr>
<tr>
<td>Linear Model</td>
<td>mod1 &lt;- lm(Y~X1+X2+X3+...+Xn, data=mydata)</td>
</tr>
</tbody>
</table>

Within the above code there are 3 key attributes, the model type, variable assignment, and data assignment. Model type was described above. Variable assignment is when R is told which variables are dependent and which are independent. The dependent y-variable is put on the left. Then the explanatory variables, x-variables, are connected with addition on the right. To create a connection between the y and x variables use a ~. Lastly, R needs to be told where the data came from. To do this we assign the data= with the dataset R should find the variables in. Figure 5.1 below displays the code to receive the full model and how R reads the code.
This code in the source includes the y variable: sal1920 and all 13 x variables. Notice that the dataset being used is specifically noted at the end of the code. This is a vital part of creating a model. If the code does not tell R where to find the variables it will result in an error.

When line 6-7 is ran R will not display the model. This is because by running the code R stores the model into the environment. In order to see the results of the code you must run `summary(model)`. The results of finding the summary of the model are shown in Figure 5.2 below.

In Figure 5.2 above, the stricts (pointed to by the red arrows) represent the variables that are statistically significant. Recall: A variable is statistically significant if its impact on the response variable is not by chance. For this example, it is great that 8 variables are shown to be statistically significant. In all regression analysis’s it is important to have significant variables!

**NBA Salary Example in SAS:**

Unlike R, SAS does not allow you to name models. A model is run based on an individual model statement. Although, as in R, one may use the generalized linear model or linear model for analysis. This will not impact your results at all. For this example, a generalized linear model will be used. The code in Figure 5.2 below follows the same idea as the code in R.

Although, an important aspect to notice is that the x variables are not attached with addition. This is normal to SAS. The variables are combined with spaces in between each one. Likewise, one may notice the “class pos (ref="C")”. This portion of the code is telling SAS that the variable “pos” in the model is categorical. Furthermore, it is telling SAS which category to use as the reference variable. This is needed in order for SAS to separate all of the levels of the
variable. Since two programs are being used the reference variables should be the same to ensure the results are mirrored. One can tell what the reference level is by checking which category is being excluded. NOTE: the CLASS statement is needed in order for SAS to recognize position as a categorical variable. This step mirrors how position had to be turned into a factor in R.

When the code in Figure 5.2 above is ran, SAS outputs the model summary. Figure 5.3 below is the outputted model summary. SAS, unlike R, does not provide asterisks next to the variables to signify significance. Thus, when checking variable significance, look at the p-value for each variable. The variables that have a p-value less than 0.05 are said to be significant. It is important to notice that the same variables that were significant in R, are significant in SAS.
Recall: Statistical Significance is when an explanatory variable has an impact on the response variable. This meaning, the influence caused by this variable was not by chance. In general, the x-variables that are significant are the variables that are included in the model, others are removed.

Checking Model Assumptions:

Normality Assumption:

As discussed in the Introduction section, normality refers to the model’s ability to follow a bell-shaped curve. With success, it can be assumed the model is normally distributed. Normal distribution refers to the sample mean of our data being normal.

Normality can be checked in two ways. One, by looking at the normal QQ plot. In a plot, normality is satisfied when the dots follow the diagonal line. Two, by using the Shapiro Wilk Normality Test.
NBA Salary Example in R:

To examine the normality assumption in R, first look at the normal QQ plot. Below in Figure 6.1 is the code and resulting plot.

Based on the QQ plot above, normality would not be satisfied. The residuals are veering off from the mean error, both at beginning and ending tails. This analysis and conclusion based on the QQ plot is confirmed by the very low p-value for the Shapiro Wilk Normality Test. Figure 6.2 Below
shows the code and result for this test. Due to the low p-value and QQ plot having veering tails it is concluded that Normality is not satisfied.

R code:

```r
> #Shapiro Wilk Normality Test for mod1
> shapiro.test(mod1$sresiduals)

Shapiro-Wilk normality test

data:  mod1$sresiduals
W = 0.96554, p-value = 1.155e-08
```

Figure 6.2

<table>
<thead>
<tr>
<th>R- plot</th>
<th>Plot(model, which=1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>R – test</td>
<td>Name: Shapiro Wilk Normality Test</td>
</tr>
<tr>
<td></td>
<td>Null Hypothesis (H₀) = The model follows a normal distribution</td>
</tr>
<tr>
<td></td>
<td>Code: shapiro.test(model$sresiduals)</td>
</tr>
<tr>
<td></td>
<td>Determining the results:</td>
</tr>
<tr>
<td></td>
<td>Large p-value – We do not reject H₀</td>
</tr>
<tr>
<td></td>
<td>Small p-value - We reject H₀</td>
</tr>
</tbody>
</table>

**NBA Salary Example in SAS:**

In order to check the normality on the original model, SAS must be told to compute the “ods graphics”. This tells SAS to output any diagnostic graphs. To do this SAS first must be told to turn the graphics on. The code `ods graphics on;` will do this. Figure 6.3 below shows the full coding. The two new additions are circled in red.
To find the output from this code one must scroll down. The output that is needed are the plots found under “Fit Diagnostics for sal1920” more specifically, for normality, the Residual v Quantile plot (1\textsuperscript{st} column second row) is needed. Figure 6.4 below shows this plot. This plot shows the tails of the residuals veering off. This indicates a lack of normality, just as in R.

![Residual vs Quantile Plot](image)

Figure 6.4

Although, this can also be checked through the Shapiro Wilk Normality Test. The code is not as intuitive as it was in R. The first step is to create an “output line” in the model code. Figure 6.5 below shows this output line (pointed to with a red arrow). What is chosen as the name for out, p, and r is optional. Out will signify the new data name, p with signify predicted values and r signifies residuals. The point of this line is to add the residual and predicted values to our data frame. This will allow diagnostics to be ran as both the Shapiro Wilk Normality Test and Levenes Test for Constant Variance depends on the residuals of the model.
Now that this line has been added the Shapiro Wilk Normality Test can be ran. The code can also be seen in Figure 6.4 above. Figure 6.6 below shows the output from the code, specifically the box labeled “Tests for Normality”. Note: there is a lot more analysis done from this code than is needed. For example, there are four normality tests that are ran. For the purposes of this paper, we will only focus on the Shapiro Wilk Results. As seen in Figure 6.6, the $p$-value is extremely small. Due to this the Null hypothesis is rejected and it is assumed that normality is not satisfied. This matches the assumption from the Normal QQ plot. Hence, just as from the output in R, we assume normality is not satisfied with the original model.

<table>
<thead>
<tr>
<th>Test</th>
<th>Statistic</th>
<th>$p$ Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shapiro-Wilk</td>
<td>W</td>
<td>0.965530</td>
</tr>
<tr>
<td>Kolmogorov-Smirnov</td>
<td>D</td>
<td>0.085466</td>
</tr>
<tr>
<td>Cramer-von Mises</td>
<td>W-Sq</td>
<td>0.86346</td>
</tr>
<tr>
<td>Anderson-Darling</td>
<td>A-Sq</td>
<td>4.559321</td>
</tr>
</tbody>
</table>

Figure 6.6

SAS- plot

```sas
ods graphics on;
proc glm data= mydata PLOTS = all;
model y = x1 x2 x3 ... xn; ods graphics off;
```

SAS- test

```sas
proc univariate data = outsalary normal;
var resid ;
run;
```
**Constant Variance Assumption**

As discussed in the introduction section, constant variance refers to the relationship between the residuals and the fitted values. In order for a model to have constant variance the variance between the residuals and fitted values should be constant. In a plot, this means the dots are evenly dispersed. If the dots are following a systematic pattern then the model does not have constant variance. Likewise, in the plot there will be a red line. If the line is horizontal at y=0 this is another indication of constant variance. On the other hand, if the red line is curved (indicating a functional relationship) then non-constant variance is assumed.

Constant variance can be checked in two ways. One, by looking at the Residuals v Fitted Values plot. Two, by using the Breusch-Pagan Test or Levenes Test for Constant Variance.

**NBA Salary Example in R:**

In order to check constant variance on the original model in R, both the Residual v Fitted Values Plot and BP Test will be analyzed.

![Residual v Fitted Value Plot for mod1](image)

*Figure 7.1*
Figure 7.1 shows the Residual vs Fitted Value Plot. From the systematic curve of the residuals, we assume Constant Variance is not satisfied. This systematic pattern is also seen by the red curve. Below in Figure 7.2, by using the BP Test, this assumption is supported. Due to the super small p-value from the BP Test, Constant Variance again is not satisfied.

```
R plot
```

```
R – test

library(CAR)
Name: Breusch-Pagan Test
Null Hypothesis (H_0) = Model satisfies constant variance.
Package: CAR
Code: library(CAR)
ncvTest(model)
Determining the result:
   Large p-value – We accept H_0
   Small p-value- We do not accept H_0
```

### NBA Salary Example in SAS:

In order to check constant variance on the original model, both the Residuals v Fitted Values plot and Levenes Test will be run. In Figure 7.3 below, the related Residual v Predicted Value plot is seen. This plot is found the in diagnostic plots from the SAS output. The “ods graphic on” code allows for this diagnostic to be visible.
The resulting plot shown in Figure 7.3 indicates non constant variance. This can be seen by the systematic trend that the residual errors are following. As a reminder, in order to accept constant variance, the plot needs to indicate random behavior.

![Figure 7.3](image)

In order to truly assume that this model does not satisfy constant variance the Levenes Test for constant variance will be run. This may be a different test than performed in R, but the results should be the same. For the purposes of this guideline, Levenes Test was a lot more applicable to coding in SAS at this level. Similar to running the Shapiro Wilk Normality Test, an output line in the model statement is needed, refer back to Figure 6.4 for a refresher on creating an output line. Below in Figure 7.4 the code for running the Levenes Test is seen. A new data frame called “outsalary” (this name is specific to this dataset and will vary as more of these tests are run) is created and added to the dataset. This new data frame will be the positive values of the residuals in the model. Then, a new model with the same variables as the original model, is run. This will create a new model that is attempting to match up with the residuals of the function to the variables. In other words, this model checks how the residuals and fitted values of the variables interact.

![Figure 7.4](image)
The resulting output from running this new model against the residuals of the original model is shown below in Figure 7.5. It is important to see that the dependent variable for this model is “aresid”. This indicates that this is the model that is being used to run the Levenes Test. The results of the Levenes Test are the p-value in the model row (circled in green). Since that p-value is extremely small the same conclusion is drawn, the original model does not satisfy constant variance.

![Figure 7.5](image)

### The GLM Procedure

**Dependent Variable: aresid**

<table>
<thead>
<tr>
<th>Source</th>
<th>DF</th>
<th>Sum of Squares</th>
<th>Mean Square</th>
<th>F Value</th>
<th>Pr &gt; F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>16</td>
<td>1.802442E15</td>
<td>1.126526E14</td>
<td>9.83</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Error</td>
<td>424</td>
<td>4.8575667E15</td>
<td>1.145652E13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Corrected Total</td>
<td>440</td>
<td>6.660095E15</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Coefficient of Multiple Determination (R²)**

The Coefficient of Determination is a tool to descriptively measure the association between the x and y variables. R² ranges between 0 and 1 and measures the amount of variance from the dependent variable that can be explained by the explanatory variables. Generally, the higher the R² the better. For example, if R²=0.87, then 87% of the response variable is explained by the x variables and only 13% by individual changes or other variables.

---

**SAS- plot**

```sas
ods graphics on;
proc glm data = mydata PLOTS = all;
model y = x1 x2 x3 ... xn / solution;
output out = outsalary i=aresid p = pred;
run;
quit;
```

**SAS - test**

```sas
Levenes Test
data outsalary2;
set outsalary;
aresid = abs(resid);
run;

proc glm data = outsalary2;
class pos (ref="C");
model aresid = rank1920 conf age1819 pos tmp1819 reb1819 as1819 st1819 blk1819 to1819 pf1819 tp1819;
run;
quit;
```
NBA Salary Example in R:

Figure 8.1 below is the model statement output. Refer to Figure 5.1 to refresh how to find the model summary in R. In the red circle is the Coefficient of Determination ($R^2$). For this model $R^2 = 0.6104$. This means that 61.04% of the model output results from the variables included and only 38.96% is from error/chance.

A higher $R^2$ would be preferred though 0.6104 is not terrible. However, improving this statistic would be beneficial for the model.

![RStudio Output](image)

Figure 8.1

NBA Salary Example in SAS:

Figure 8.2 below is the model statement output. Refer to Figure 5.2 to refresh how to find the model summary in SAS. In the red circle is the Coefficient of Determination ($R^2$). For this model $R^2 = 0.610366$. This is a good result as it matches the result in R. If they did not match that would indicate an issue. This means that 61.04% of the model output results from the variables included and only 38.96% is from error/chance.

Again, a higher $R^2$ would be preferred though 0.6104 is not terrible. However, improving this statistic would be beneficial for the model.
Adjusted Coefficient of Determination (AdjR²)

The Adjusted Coefficient of Multiple Determination has little effect on R² itself and follows the same interpretation. The main difference is the inclusion of the number of predictor variables and number of data points within the output. In other words, the statistic is adjusted for the number of variables being used in the model. Due to this, the adjusted coefficient of multiple determination is the more logical statistic to use for interpretation. Note: This is the preferred statistic. The regular coefficient of determination is still good, but the adjusted coefficient of determination is preferred. Figure 8.3 below shows a visual representation of the adjusted coefficient of determination. Note that the more associated the x and y are (a higher AdjR²) the closer the points are to the line.

Figure 8.2

<table>
<thead>
<tr>
<th>Source</th>
<th>DF</th>
<th>Sum of Squares</th>
<th>Mean Square</th>
<th>F Value</th>
<th>Pr &gt; F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>10</td>
<td>2.227424E10</td>
<td>1.392140E15</td>
<td>41.51</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Error</td>
<td>424</td>
<td>1.421903E16</td>
<td>3.353458E13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Corrected Total</td>
<td>440</td>
<td>3.649928E16</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

AdjR² = 1

AdjR² = 0.87

AdjR² = 0.43

AdjR² = 0

Figure 8.3
NBA Salary Example in R:

Figure 8.4 below is the model statement output. Refer to Figure 5.1 to refresh how to find the model summary in SAS. In the red circle is the adjusted coefficient of determination \( (adjR^2) \). For this model \( adjR^2 = 0.5957 \). This means that 59.57% of the model output results from the variables included and only 40.43% is from error/chance.

Again, a higher \( adjR^2 \) would be preferred though 0.5957 is not terrible. However, improving this statistic would be beneficial for the model.

NBA Salary Example in SAS:

Unfortunately, in SAS, by running a GLM, \( adjR^2 \) is not shown. Although, by knowing the formula and the regular coefficient of determination, the \( adjR^2 \) can be found. Figure 8.5a below shows the formula for \( adjR^2 \). Figure 8.5b shows how the \( adjR^2 \) is calculated from the original model. Refer back to the tibble in Figure 8.4 to see the total number of variable and datapoints (degrees of freedom). The numbers are circled in green.

\[
adjR^2 = 1 - \frac{(1-R^2)(N-1)}{N-p-1}
\]

\[
adjR^2 = 1 - \frac{(1-(0.610366)^2)(440)}{440-16-1} = 0.59565660
\]

Based on the calculations above the \( adjR^2 \) matches that shown in R. If they did not match that would indicate an issue. This means that 59.57% of the model output results from the variables included and only 40.436% is from error/chance.
Again, a higher adjR\(^2\) would be preferred though 0.5957 is not terrible. However, improving this statistic would be beneficial for the model.

**Correcting Model Assumptions**

**Correcting Normality**

**Box Cox Transformation**

In order to correct the normality assumption a Box Cox Transformation in run on the model. This procedure will tell us what transformation to compute on our response variable (y).

This process uses log likelihood to estimate the best lambda. Each lambda is associated with a given power transformation. Figure 9.1 below shows the most common Box Cox Transformations. Note: these are not all possible transformations, just the most common.

<table>
<thead>
<tr>
<th>(\lambda)</th>
<th>(Y')</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>(Y^2)</td>
</tr>
<tr>
<td>0.5</td>
<td>(\sqrt{Y})</td>
</tr>
<tr>
<td>0</td>
<td>(\log Y)</td>
</tr>
<tr>
<td>-0.5</td>
<td>(\frac{1}{\sqrt{Y}})</td>
</tr>
<tr>
<td>-1.0</td>
<td>(\frac{1}{Y})</td>
</tr>
</tbody>
</table>

Figure 9.1

**NBA Salary Example in R:**

The code to perform a Box-Cox Transformation in R is in the MASS package. In order to run this MASS must be imported. When ran, the code will output a log likelihood plot. This plot will signify which lambda to pick. To develop a better visual on the lambda the margins must be adjusted. When adjusting the margins, the code seq(a,b,c) is used. When deciding what a,b, and c to choose to think of the plot as a regular graph. The letters a and b create the domain of the plot, \([a,b]\). This meaning what x values will be included. The letter c denotes the increments the x-values should take from \([a,b]\). For example, if the code read seq(0,1,1/4) the x-values on the x-axis would be \(\{0,0.25,0.5,0.75,1\}\). Figure 9.2 below shows both the code and plot output for the Box Cox Transformation.
As seen in Figure 9.2 above, the margins are very wide. This makes estimating the lambda hard. By looking at the plot it is seen that the lambda is between 0 and 1. This indicates \( a = 0 \) and \( b = 1 \). Now an increment is chosen, this is the \( c \) value. A common value is \( 1/10 \). Figure 9.3 below shows the adjustment code for the plot, along with the resulting plot.

Now that there is an adjusted plot, one may estimate the lambda to be between 0.2 and 0.3, but closer to 0.2. This may be enough to move forward with the transformation but sometimes knowing the exact lambda is better. The exact lambda can be found by using log likelihood. Figure 9.4 below shows the code and resulting exact lambda. On the right-hand side is the exact lambda, which is directly related to the smallest log likelihood, is labeled \([1.0]\).
As seen in Figure 9.4 above, the exact lambda is 0.24242424, this is right in the middle of 0 and 0.5. Refer back to Figure 9.1 for the related lambdas. This indicates trying both a log and square root transformation. If the lambda was not in the middle of two values, one transformation may be chosen. Although, in an effort to not make the wrong assumption both transformations will be run.

To perform the transformation on y two new models will be created. One with a square root transformation on y and one with a log transformation. In R, sqrt(y) is used, and log(y) is used in the model statement to create this transformation. Figure 9.5 below shows the original, sqrt(y) and log(y) models that are now part of the analysis. On the right-hand side of Figure 9.5, one can see in the environment that these 2 new models were added.
Recall that the point of this transformation was to improve the normality of our model. Figure 9.6 below compares all three models and their normal QQ plots and Shapiro Wilk Normality Tests. One can see that the $\sqrt{y}$ transformation resulted in the best normality.
NBA Salary Example in SAS:

Unlike in R, SAS is much simpler when it comes to finding the exact lambda. Furthermore, SAS adjusts the margins of the related plot prior to outputting it. Figure 9.7 below shows the code and related plot outputted by SAS.

<table>
<thead>
<tr>
<th>Running Initial Box-Cox</th>
<th>Library(MASS)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Boxcox(model)</td>
</tr>
<tr>
<td>Running Box-Cox</td>
<td>boxcox(model, data=mydata, lambda = seq(0, 1, 1/10), plotit = TRUE)</td>
</tr>
<tr>
<td>with adjusted margins</td>
<td></td>
</tr>
<tr>
<td>for lambda</td>
<td></td>
</tr>
<tr>
<td>Finding Exact Lambda</td>
<td>b=boxCox(model, data=mydata, lambda = seq(0, 1, 1/10), plotit = TRUE)</td>
</tr>
<tr>
<td></td>
<td>lambda= b$x</td>
</tr>
<tr>
<td></td>
<td>loglik=b$y</td>
</tr>
<tr>
<td></td>
<td>bc = cbind(lambda, loglik)</td>
</tr>
<tr>
<td></td>
<td>bc = bc[order(-loglik), ]</td>
</tr>
<tr>
<td></td>
<td>bc</td>
</tr>
</tbody>
</table>

Figure 9.6

Figure 9.6
As seen in Figure 9.6 above, the lambda matches that found in R, hence by the same logic presented in the “NBA Salary Example in R” section, both the sqrt(y) and log(y) transformation will be performed. Although the process of the Box Cox Transformation is simpler in SAS, the process of creating the transformation on y is more strenuous. In SAS, it is necessary to add a new response variable with its given transformation into the dataset. In other words, two new variables sqrt(y) and log(y) must be added to the dataset.

Figure 9.7 below shows the code to add sqrt(y) and log(y) into the dataset. This portion is circled in green. Under each respective transformation is the related model. Notice how the y variable is now the new transformation. Another thing to notice is how the output statement (pointed to in red) have different names for the r and p. These names match that of the given y variable.

```
sqrt transformation on y;
   data NBASalary;
   set NBASalary;
   sqy = sqrt(sal1920);
   run;
ods graphics on;
proc glm data= NBASalary PLOTS=ALL ;
   class pos (ref="C");
   model sqy = rank1920 conf age1819 pos tmp1819 reb1819 as1819 st1819 b1k1819 to1819 pf1819 to1819 / solution;
   output out = outs salary r=resid root p = pred;
   run;
quit;
```

```
log transformation on y;
   data NBASalary;
   logy = log(sal1920);
   run;
ods graphics on;
proc glm data= NBASalary PLOTS=ALL ;
   class pos (ref="C");
   model logy = rank1920 conf age1819 pos tmp1819 reb1819 as1819 st1819 b1k1819 to1819 pf1819 to1819 / solution;
   output out = outs salary r=resid log p = pred;
   run;
quit;
```

Figure 9.7

Just as in R, it is important to see if the box cox transformation was successful. Figure 9.8 below compares all three models and their normal QQ plots and Shapiro Wilk Normality Tests. One can see that the sqrt(y) transformation resulted in the best normality.
Running Box-Cox and Finding Specific Lambda

```sas
proc transreg data = mydata ss2 detail;
model boxcox(y /lambda= 0 to 1 by 0.01) = identity(x1 + x2 + x3 + ...
... + x_n);
run;
```

NOTE: no categorical variables may be included in a box-cox transformation in SAS.
Correcting Non-Constant Variance:

Non-constant variance can occur for many reasons. In general, a model’s variance will be better with a simpler model. This meaning when one is deciding a model to pick, the simpler one is always better. To limit the number of variables within a data frame Principal Component Analysis can be executed. Additionally, a model’s variance can be negatively affected when one or more variables are highly correlated to one another. Checking each variable’s Variance Inflation Factor can help tell if any variables are too highly correlated. Lastly, a model’s variance may be impacted if the variables data points are systematically heightened within a certain part of the data frame. Centering the quantitative variables may help this.

Principal Component Analysis:

Principal Component Analysis (PCA) is a tool used to decrease the number of variables in a data set. The analysis will combine the variables that interact with the model in similar ways into one new variable. By doing so it creates a principal component. A principal component is a linear combination of multiple other variables within the dataset. This new variable preserves the important information from the variable while removing the negative impact on the variance.

An important aspect of PCA is that it can only be ran on quantitative variables. Thus, prior to performing this analysis a new data frame that only contains quantitative variables must be formed. Likewise, this dataset should not include the response variable.

The analysis will make p principal components, p being the number of variables in the dataset. The principal components that are chosen should have an eigenvalue above 1. Another way to decide which principal components to use is to choose the ones that, together, make up at least 70% of the variance. After the principal component(s) are chosen, a new model including only these components, will be created.

NBA Salary Example in R:

In R, in order to run a Principal Component Analysis, the factoextra package must be installed. Figure 10.1 shows the code to find the principal components of the quantitative variables.

To get the eigenvalues of the principal components line 97 must be run. Figure 10.2 is the resulting eigenvalue vectors.
In the above Figure 10.2 the eigenvalues are listed. As seen, the only eigenvalues that are above 1 are Dim 1 and Dim 2. This indicates that the first two eigenvectors should be chosen. When looking at “variance percent” the first two make up over 70%, this indicates that these two should be chosen. Another thing to notice is how there were 10 principal components made, this is due to there being 10 quantitative variables included in the PCA.

The next step is to find the related coefficients to each variable in the components. Line 91 in Figure 10.1 is the code that will output the coefficients for each variable. Figure 10.3 is the output from line 91 in Figure 10.1.

Based on the results from Figure 10.2 both PC1 and PC2 are chosen. The next step is to make the components and add them to the data set. Figure 10.3 displays the needed values in creating the principal components. Figure 10.4 below displays how PC1 and PC2 are added to the data set. Notice how the coefficients attached to each variable are the same from Figure 10.3. Also, the variables are attached by addition.
Once PC1 and PC2 are created a new model including both principal components is made. Figure 10.5 below shows the model being ran. Recall that the point of this analysis was to create constant variance. Figure 10.5 also displays the Residual v Fitted Values plot and the BP Test for constant variance.

Based on the results in Figure 10.5 Principal Component Analysis did not fix the issue of nonconstant variance and something else must be done to fix this.
### Running Principal Component Analysis
- this also presents the coefficients for each variable

Install factoextra
library(factoextra)
data.pca <- prcomp(mydata_nocategorical, scale = TRUE)
print(data.pca)

### Choosing a PC
get_eigenvalue(data.pca)

### Attaching PCA(s) to your dataset
attach(dataset)
dataset$PCA <- (C1*X1 + C2*X2 + ... + Cn*Xn)
attach(dataset)
PCA

Note: PCA represents the name of the variable that is developed.

### NBA Salary Example in SAS:
Proc Factor will be used in order to run Principal Component Analysis in SAS. The important part of this analysis in SAS is to ensure that it matches up with the analysis in R. Below in Figure 10.6 is the code for finding the principal components.

```sas
proc factor data = NBASalary eigenvalues
   mineigen = 1
   method = prin
   priors = one
   rotate = varimax
   round;
var conf age1819 tmp1819 reb1819
as1819 st1819 blk1819 to1819 pf1819 tp1819 ;
run;
```

Figure 10.6
In Figure 10.6, mineigen indicates the minimum eigenvalue. As described before, the only principal components that are included are those that have above an eigenvalue of 1. Method, priors, and rotate are all indicated to perform the PCA. Var tells SAS which variables to include in the analysis, like in R it is only the quantitative variables. Note that in SAS a new data set did not have to be made to run this analysis. Figure 10.7 below is the output from SAS when the code in Figure 10.6 is ran.

<table>
<thead>
<tr>
<th>Eigenvalue</th>
<th>Difference</th>
<th>Proportion</th>
<th>Cumulative</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6.1812701</td>
<td>0.6188</td>
<td>0.6188</td>
</tr>
<tr>
<td>2</td>
<td>1.09182164</td>
<td>0.1092</td>
<td>0.7280</td>
</tr>
<tr>
<td>3</td>
<td>0.95421763</td>
<td>0.092</td>
<td>0.8202</td>
</tr>
<tr>
<td>4</td>
<td>0.85333917</td>
<td>0.0853</td>
<td>0.9055</td>
</tr>
<tr>
<td>5</td>
<td>0.30174371</td>
<td>0.0302</td>
<td>0.9357</td>
</tr>
<tr>
<td>6</td>
<td>0.23948103</td>
<td>0.0239</td>
<td>0.9596</td>
</tr>
<tr>
<td>7</td>
<td>0.19701884</td>
<td>0.0197</td>
<td>0.9793</td>
</tr>
<tr>
<td>8</td>
<td>0.13192139</td>
<td>0.0132</td>
<td>0.9925</td>
</tr>
<tr>
<td>9</td>
<td>0.10337617</td>
<td>0.0103</td>
<td>0.9999</td>
</tr>
<tr>
<td>10</td>
<td>0.05094779</td>
<td>0.0051</td>
<td>1.0000</td>
</tr>
</tbody>
</table>

2 factors will be retained by the MINEIGEN criterion.

<table>
<thead>
<tr>
<th>Eigenvectors</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>age1819</td>
<td>-4</td>
<td>95</td>
</tr>
<tr>
<td>age1819</td>
<td>15</td>
<td>-22</td>
</tr>
<tr>
<td>tmp1819</td>
<td>37</td>
<td>-3</td>
</tr>
<tr>
<td>tmp1819</td>
<td>35</td>
<td>-12</td>
</tr>
<tr>
<td>ret1819</td>
<td>33</td>
<td>-5</td>
</tr>
<tr>
<td>ret1819</td>
<td>35</td>
<td>-3</td>
</tr>
<tr>
<td>as1819</td>
<td>27</td>
<td>18</td>
</tr>
<tr>
<td>st1819</td>
<td>37</td>
<td>1</td>
</tr>
<tr>
<td>st1819</td>
<td>37</td>
<td>5</td>
</tr>
<tr>
<td>blik1819</td>
<td>37</td>
<td>-1</td>
</tr>
<tr>
<td>to1819</td>
<td>37</td>
<td>18</td>
</tr>
<tr>
<td>to1819</td>
<td>37</td>
<td>5</td>
</tr>
<tr>
<td>pff1819</td>
<td>37</td>
<td>-1</td>
</tr>
<tr>
<td>pff1819</td>
<td>37</td>
<td>18</td>
</tr>
<tr>
<td>w119f</td>
<td>37</td>
<td>5</td>
</tr>
</tbody>
</table>

Printed values are multiplied by 100 and rounded to the nearest integer. Values greater than 0.316228 are flagged by an *.

Figure 10.7

Just as in R, both the first and second principal components have an eigenvalue above 1. Also, the proportion column indicates the same amount of variance seen in Figure 10.2. SAS also indicates that “2 factors will be retained by the MINEIGEN criterion”. The chart below the eigenvectors indicates the coefficients for each variable. At the bottom of the chart, it is indicated that the values were multiplied by 100 and rounded to the nearest integer. The numbers will need to be reverted back to their original form by dividing by 100. Then PC1 and PC2 can be developed and added into the dataset in SAS. Once the variable coefficients are converted they match those found in R; although, they are rounded to the nearest integer. Figure 10.8 shows the code for adding these two new variables to the data set.
After creating PC1 and PC2 and adding them into the dataset a new model with only PC1 and PC2 is created. Then the Residual vs Fitted Values plot and Levenes Test are run to investigate how this analysis effected the variance of the model. Figure 10.9 shows the results of this analysis along with the model statement. Note that the output statement is changed to fit the model type.

Based on the results in Figure 10.9 Principal Component Analysis did not fix the issue of nonconstant variance and something else must be done to fix this.
Variance Inflation Factor:

The Variance Inflation Factor refers to the collinearity between variables. What is collinearity? Variables are said to be collinear if one or more represent the same thing. In other words, variable $X_1$ and $X_2$ may be giving our data set the same information. This meaning they are correlated. This co/multicollinearity has a negative effect on the model's variance. Some problems include misleading and increasing coefficient estimates along with a skewed variance.

What is an example of 3 variables that are highly correlated? Think of a data set that involves human dieting. In this data set weight would be observed. Consider the following scenario. $X_1$ represents weight in lbs, $X_2$ represents weight in kilograms, and $X_3$ represents weight in ounces. These three variables are highly correlated since they provide the same information to the data set.

How do we identify multicollinearity? One way to detect multicollinearity is to find the Variance Inflation Factor (VIF) for each variable. This factor represents to what degree the variance of a coefficient is increased based on its collinearity. If the VIF is large, generally above 10, this indicates a high degree of multicollinearity and that given variable should be removed from the dataset.

What to do with variables that are highly correlated: When variables are found to have a high collinearity one of them should be removed.
NBA Salary Example in R:

To find Variance Inflation Factors in R the CAR package is needed. Then a simple code `vif(model)` can be ran. Figure 11.1 below shows the code for running this analysis on all three of the current models being analyzed.

![Figure 11.1](image)

Figure 11.1 shows the code for running this analysis on all three of the current models being analyzed. In all three outputs `to1819` is the only variable that has a VIF above 10. Due to this only `to1819` will be removed from all three models.

![Figure 11.2](image)

Recall that the VIF’s of each variable was checked in an attempt to fix constant variance. Three new models will be made, the only difference is the removal of `to1819`. Below in Figure 11.3 is models and their Residual v Fitted Value plot and BP Test are shown.
From the analysis shown in Figure 11.3, it is seen that removing to1819 did slightly improve the constant variance. Especially in the model with the square root transformation on y.

---

**Finding VIF**

<table>
<thead>
<tr>
<th>Library(CAR)</th>
<th>Vif(model)</th>
</tr>
</thead>
</table>

---

**Figure 11.3**
NBA Salary Example in SAS:
In SAS, Proc Reg will be ran to find the VIF’s of each variable. Figure 11.4 shows the model statements ran on each of the three current models.

```
proc reg data= NBASalary ;
  model sal1920 = rank1920 conf age1819 tmp1819 rebl1819
                   asl1819 st1819 blk1819 to1819 prf1819 tp1819 / vif;
  run ;
  quit;
```

Figure 11.4

Note how in Figure 11.4, “vif” was added at the end of each model statement and preceded by a forward slash (/). Figure 11.5 below shows the output for each model. In all three outputs to1819 is the only variable that has a VIF above 10. Due to this only to1819 will be removed from all three models.

Note how in Figure 11.4, “vif” was added at the end of each model statement and preceded by a forward slash (/). Figure 11.5 below shows the output for each model. In all three outputs to1819 is the only variable that has a VIF above 10. Due to this only to1819 will be removed from all three models.
Now three new models with the removal of to1819 should be made. Below in Figure 11.6 is models and their Residual v Fitted Value plot and Levenes Test are shown.

```r
ods graphics on;
proc glm data= NBASalary plots=all;
  class pos(ref="C");
  model sal9290 = rank1920 conf age1819 pos tmp1819 reb1819
  as1819 sti1819 blk1819 pf1819 tp1819 / solution;
  output out = outsalary r=resid p=pred;
run;
quit;
```

```
ods graphics on;
proc glm data= NBASalary plots=all;
  class pos(ref="C");
  model sqty = rank1920 conf age1819 pos tmp1819 reb1819
  as1819 sti1819 blk1819 pf1819 tp1819 / solution;
  output out = outsalary r=residqrt p=pred;
run;
quit;
```

---

**Figure 11.6**

The GLM Procedure

**Dependent Variable: resid**

<table>
<thead>
<tr>
<th>Source</th>
<th>DF</th>
<th>Sum of Squares</th>
<th>Mean Square</th>
<th>F Value</th>
<th>Pr &gt; F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>15</td>
<td>1.6724542E15</td>
<td>1.114972E14</td>
<td>9.46</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Error</td>
<td>425</td>
<td>5.01439E15</td>
<td>1.179561E13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Corrected Total</td>
<td>440</td>
<td>6.603601E15</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

The GLM Procedure

**Dependent Variable: resid**

<table>
<thead>
<tr>
<th>Source</th>
<th>DF</th>
<th>Sum of Squares</th>
<th>Mean Square</th>
<th>F Value</th>
<th>Pr &gt; F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>15</td>
<td>1.656390851</td>
<td>1100.26504</td>
<td>3.73</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Error</td>
<td>425</td>
<td>1260.98715</td>
<td>2955.100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Corrected Total</td>
<td>440</td>
<td>14261.08677</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

The GLM Procedure

**Dependent Variable: resid**

<table>
<thead>
<tr>
<th>Source</th>
<th>DF</th>
<th>Sum of Squares</th>
<th>Mean Square</th>
<th>F Value</th>
<th>Pr &gt; F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>15</td>
<td>18.2359438</td>
<td>1.2157296</td>
<td>3.08</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Error</td>
<td>425</td>
<td>167.7349778</td>
<td>0.3949703</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Corrected Total</td>
<td>440</td>
<td>185.9708216</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
From the analysis shown in Figure 11.6 it is seen that removing to1819 did slightly improve the constant variance. Especially in the model with the square root transformation on y.

| Finding VIF | proc reg data= mydata ;  
|            | model y = x1 x2 x3 ... xn / vif;  
|            | run ;  
|            | quit; |

**Centering Variables**

*What does it mean to center a variable?* Centering a variable can be done by subtracting the mean from the variable itself.

*What is the benefit of centering a variable?* One would choose to center a variable if the variable is highly skewed. For example, if a dataset tracked a person’s health over time physical abilities would be included. In general, an individual is more active and physically healthy between 18-45 years of age. Due to this, this variable is highly concentrated in the center. Therefore, in an attempt to correct nonconstant variance one may center this variable.

**NBA Salary Example in R:**

In this example centering the variable seems extremely logical. If you think of players within the NBA they all have a “prime”. This meaning they go through their career and peak around 22-30 years old then gradually decline. This peak also correlates to the other statistics in the NBA and hence directly correlated to the players overall salary. In Figure 12.1, developed from Tableau, this “peak” can visually be seen. This trend then prompts the models to have centered variables. By centering the variables, it will “level the playing field” for each player. In order to center the variables, the mean must be subtracted from each individual variable.

![Age Impact on Average Salary](image.png)

**Figure 12.1**

Centering variables in R is extremely easy. As noted earlier, R can be used as a calculator. Hence, subtracting the mean from the variable is as easy as coding: variable - mean(variable). Figure 12.3 below shows all variable being centered and then added to the dataset. These variables are denoted with a C after the variable name to keep track of the centered variables in the dataset. Note: only quantitative variables can be centered.
Now that all the variables are centered the three existing models are adjusted and recoded with the centered variables. The point of centering the variables is to improve constant variance. Figure 12.3 below shows the edited model statements along with the Residual v Fitted Value Plot and BP Test for Constant Variance.

```r
# Adjusted and recoded models

attach(NBASalary)
NBASalary$ageC <- (age1819-mean(age1819))
NBASalary$conf <- (conf-mean(conf))
NBASalary$tmpC <- (tmp1819-mean(tmp1819))
NBASalary$rebC <- (reb1819-mean(reb1819))
NBASalary$asC <- (as1819-mean(as1819))
NBASalary$stC <- (st1819-mean(st1819))
NBASalary$b1kC <- (b1k1819-mean(b1k1819))
NBASalary$toC <- (to1819-mean(to1819))
NBASalary$pC <- (p1819-mean(p1819))
NBASalary$tpC <- (tp1819-mean(tp1819))

attach(NBASalary)

mod1 <- lm(sal1920-rankyn-confC-ageC-pos+tmpC+rebC+ascC+stC+b1kC+pC+tpC, data=NBASalary)

modCroot <- lm(sqrt(sal1920)-rankyn-confC-ageC-pos+tmpC+rebC+ascC+stC+b1kC+pC+tpC, data=NBASalary)

bptest(mod1)
studentized Breusch-Pagan test
data: mod1
BP = 75.815, df = 15, p-value = 4.033e-10

bptest(modCroot)
studentized Breusch-Pagan test
data: modCroot
BP = 37.747, df = 15, p-value = 0.0009834

modClog <- lm(log(sal1920)-rankyn-confC-ageC-pos+tmpC+rebC+ascC+stC+b1kC+pC+tpC, data=NBASalary)

bptest(modClog)
studentized Breusch-Pagan test
data: modClog
BP = 46.106, df = 15, p-value = 5.114e-05
```
Based on the analysis in Figure 12.3, centering all of the variables did improve the constant variance, especially on modCroot. ModCroot now has the best constant variance of all the currently existing models.

<table>
<thead>
<tr>
<th>Centering and attaching variables to data set</th>
<th>attach(mydata)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mydata$X_{i} &lt;- (X_{i} - \text{mean}(X_{i}))</td>
<td></td>
</tr>
<tr>
<td>attach(mydata)</td>
<td></td>
</tr>
</tbody>
</table>

**NBA Salary Example in SAS:**

Centering variables in SAS is far less intuitive than in R. There is not a “mean” function in SAS. Hence first the means must be found for each variable. Then each centered variable will be attached the data set by subtracting the mean by hand, this meaning that each mean will be subtracted manually. Proc Means is used to find all of the means of each variable. Figure 12.4 below shows how to find the means of each variable along with the resting output.

![Figure 12.4](image)

After finding the mean for each variable, the new centered variables are created and attached to the dataset. Figure 12.5 below shows how each variable is individually added into the data set. Notice how each mean is written manually into the coding.

![Figure 12.5](image)
Now that all the variables are centered the three existing models are adjusted and recoded with the centered variables. The point of centering the variables is to improve constant variance. Figure 12.6 below shows the edited model statements along with the Residual v Fitted Value Plot and BP Test for Constant Variance.
Choosing a Model:
Before choosing a model one must have a plethora of options. Performing a Regression Analysis ensures a working and reliable model; although, it does not guarantee the best model. As shown in Figure 13.1 below, the number of possible regression models is $2^p$ where $p =$ the number of predictor variables. For example, if the dataset being used had 4 x-variables then there for $2^4=16$ possible models for this dataset.

*How to develop multiple models:* Models differ based on the variables in the model and the interactions between them. A key aspect of developing models is variable selection.

**Variable Selection:**
One method of variable selection is called the Stepwise Method. With this approach the software will add variables to the model and investigate its impact. From there the software will either include or exclude that variable. After working on one variable, it will consider the next one and so one. This process is repeated in the background until multiple working models are made.

Within the stepwise method there are three options. There is the forward, backward, and both way stepwise approaches. All three approaches work and not one is suggested over the other. However, since the goal is to develop as many models as possible one may consider using all three. Each method will produce one model. Note: It is possible for the same model to be made from different approaches.

Another method of variable selection is the Exhaustive Method. The Exhaustive Method is a series of code that prompts the software to create multiple subsets of different models based on the original. This meaning that there is a large number of models to choose from. This approach is only done in R.

*How do we decide which model to select?*
In order to select the best working model, one may think of important characteristics of a model. In the beginning of these guidelines the Adjusted Coefficient of Determination was discussed. Since this is a key characteristic one may extract the model with the highest Adjusted $R^2$ and use this as a possible model. Another way to extract a model is by using the Mallows $C_p$ Criterion. The $C_p$-Statistic measures the amount of bias a model has. For any given Regression Analysis, the amount of bias should be minimized. Hence, we will ask the software to extract the model with the lowest $C_p$-Statistic.

![Number of possible models: $2^p$ where $p =$ number of x-variables](Figure 13.1)
What is residual in a regression model?

Residual refers to the difference between the observed and fitted values. If there is no bias in the model then the values of the observed values (what we know to be true) and the predicted values (what the model says the be true) would be equal. The difference between the observed and fitted values is said to be the residual of the model. In any analysis the residual should be minimized.

NBA Salary Example in R:

Now that some relatively working models have been created, R can be used to create more. Another key aspect of Regression Analysis is finding which variables should be included in the model and which should not. This process can be done by including some, checking the result, including another, checking the results, etc. Although, this process can be long and tedious, especially with a large number of variables. Due to this, both the Stepwise and Exhaustive Methods will be used. The only model that satisfied both normality and constant variance was modCroot, this is the model with centered variables and a square root transformation on $y$. Due to this, the only model that will go through variable selection will be modCroot.

Figure 13.2 below displays the code in running the forward, backward and both ways stepwise method on modCroot. Lines 266-270 create the full model needed to run the analysis. This line only needs to be ran once. From here the 3 stepwise approaches can be made. The default step approach in R is the backwards approach, this is why running the backwards approach in line 271 is much shorter than running the forwards and bothways approaches in lines 276 and 281.

```r
# Backward variable selection
fullmodelroot <- lm(sqrt(salary) ~ rank + conf + age + pos + team + east + west + rank + conf + age + pos + team + east + west, data = NBAdata)
nothing <- lm(sqrt(salary) ~ 1, data = NBAdata)
backwardsroot <- step(fullmodelroot, direction = "backward")
formula(backwardsroot)
summary(backwardsroot)

# Forward variable selection
forwardmodel <- step(nothing, scope = list(lower = formula(nothing), upper = formula(fullmodelroot)), direction = "forward")
formula(forwardmodel)
summary(forwardmodel)

# Bothways variable selection
bothwaysmodel <- step(nothing, scope = list(lower = formula(nothing), upper = formula(fullmodelroot)), direction = "both", trace = 0)
formula(bothwaysmodel)
summary(bothwaysmodel)
```

Figure 13.2

After all three approaches are coded and ran, the formula string in lines 272, 277 and 282 tell R to show the chosen model and the corresponding variables. Figure 13.3 below shows the resulting formulas from the 3 stepwise approaches. One can see that the results from each approach resulted in the same model. This means that only one new model was formed from the stepwise approach for variable selection.
The new model along with its model summary is shown below in Figure 13.4. In the model summary it is seen that every variable is significant. This is important and should be expected from a model that resulted from variable selection.

```r
> formula(backwardsCroot)
sqrt(sal1920) ~ rankyn + ageC + tmpC + rebC + asC + blkC + pfc + tpc
> formula(forwardsCroot)
sqrt(sal1920) ~ tpc + ageC + pfc + tmpC + blkC + asC + rebC + rankyn
> formula(bothwaysCroot)
sqrt(sal1920) ~ tpc + ageC + pfc + tmpC + blkC + asC + rebC + rankyn
```

Figure 13.4

Since the Stepwise Approach for Variable Selection only presented us with a single new model, another approach should be used to broaden the number of optional models. The next method to use in the Exhaustive Approach. This method tells R to look at every single possible subset or model for a given set of x-variables. Again, only modCroot will be used in the exhaustive approach. The package needed to run this is the “leaps” package. Below in Figure 13.5 the code to run the exhaustive approach is shown. Notice how the cp criterion and adjr² are the two criterion used to select a model from all the subsets created. Another important aspect of this code is the nmax in line 295. This number tells R how many subsets to create, this number
should be greater than the number of variables in the model being used in the exhaustive approach.

```r
library(leaps)
subsetselectionmodCroot <- regsubsets(sqrt(salary~confirmed+age+pos+tmpC+
+rebc+asc+stc+b1c+
+pfc+tpeC,wt=NULL,nbest=1, nvmax=20, data=NBASalary)
ResultofSubsetSelectionmodCroot <- summary(subsetselectionmodCroot)
#check which subset has the maximum Adjusted R-Squared
BestSubSetmodCrootAdjr2 <- which.max(ResultofSubsetSelectionmodCroot$adjr2)
BestSubSetmodCrootAdjr2 #This says subset 11
BestSubSetmodCrootCp <- which.min(ResultofSubsetSelectionmodCroot$cp)
BestSubSetmodCrootCp #This says subset 8
#See plots of subset 8 to visualize the maximization and minimization of adjr2 and cp
par(mfrow=c(1,2))
plot(ResultofSubsetSelectionmodCroot$adjr2, xlab="variables selected", ylab="adjr2")
plot(ResultofSubsetSelectionmodCroot$cp, xlab="variables selected", ylab="cp")
#Now find the coefficients of subset 11 and 8
coef(subsetselectionmodCroot,11)
coef(subsetselectionmodCroot,8)
```

The output from lines 305-307 shows how the subsets are chosen based on minimizing or maximizing a criterion. Below in Figure 13.6 shows the resulting plots.

![Figure 13.5](image1)

On the left-hand-side plot, the numbers of variables added into the model are shown in relation to how it effects the adjr^2. It can be seen that subset 11 is the highest point, this matches to how subset 11 was selected by R. Likewise, subset 8 on the right-hand-side plot is the lowest point, hence it is the minimized cp criterion and was selected by R.
To find the variables included in this model one can run line 310 and 311 in Figure 13.5. The results from these lines are shown below in Figure 13.7. Also, in Figure 13.7, is the resulting models from the exhaustive approach. Note that one of the resulting models (modCroot3) is equivalent to modCroot1. Hence, no new model is made for this subset selection.

In summary, through the stepwise approach, one new model was given. From the exhaustive approach one new models were given. This meaning that there are now three total models that will undergo model comparison. Model comparison will allow R to choose the best model out of the current three. Figure 13.8 below shows the current three models.
### NBA Salary Example in SAS:

As in R, the goal is to find as many possible working models as possible. As described in figure 13.1, the number of possible is $2^p$, where $p$ is the number of x-variables. In this example the number of x-variables is 11, recall that to1819 was removed. This means that there are 2,048 possible models. With this in mind, it would be logical for SAS and R to present different results for variable selection.

Unlike R, SAS does not have an exhaustive approach, rather, in its stepwise approach it allows for multiple criterion for variable selection. The three criteria chosen for this data set was the cp criterion, adjr$^2$, and significance level. Significance level refers to the F-statistic for the model, which should be maximized. Figure 13.9 below shows the code for this analysis. Note that each approach: forward, backwards and bothways, has three corresponding criterions, as noted above. Note that when running the stepwise approach based on the significance level, the model may not be chosen by the significance level. Hence adjr$^2$ will be used for the “choose” portion of this method. This is seen in line 406.

<table>
<thead>
<tr>
<th>Method</th>
<th>Code</th>
</tr>
</thead>
</table>
| Stepwise Method (backwards) | \[
\text{fullmod} \leftarrow (Y \sim X_1 + X_2 + X_3 + \ldots + X_n, \text{data} = \text{mydata}) \\
\text{nothing} \leftarrow \text{lm}(Y \sim 1, \text{data} = \text{mydata}) \\
\text{backwardsmodel} = \text{step(}\text{fullmod}) \\
\text{formula(}\text{backwardsmodel}) \\
\text{summary(}\text{backwardsmodel})
\] |
| Stepwise Method (both ways) | \[
\text{fullmod} \leftarrow (Y \sim X_1 + X_2 + X_3 + \ldots + X_n, \text{data} = \text{mydata}) \\
\text{nothing} \leftarrow \text{lm}(Y \sim 1, \text{data} = \text{mydata}) \\
\text{bothways4a} = \text{step(}\text{nothing, list(lower=formula(}\text{nothing}), \text{upper=formula(}\text{fullmod}))}, \text{direction="both", trace=0}) \\
\text{formula(}\text{bothwaysmodel}) \\
\text{summary(}\text{bothwaysmodel})
\] |
| Exhaustive Method | \[
\text{library(leaps)} \\
\text{subsetselectionmodel} \leftarrow \text{regsubsets}(Y \sim X_1 + X_2 + X_3 + \ldots + X_n, \text{wt=NULL, nbest=1,nvmax=k+8, data=mydata}) \\
\text{ResultsofSubsetSelectionmodel} \leftarrow \text{summary(}\text{subsetselectionmodel}) \\
\text{ResultsofSubsetSelectionmodel}
\] |
| Extracting Model with Highest Adj $R^2$ | \[
\text{BestSubSetmodelAdjr}^2 \leftarrow \text{which.max(}\text{ResultsofSubsetSelectionmodel}\$\text{adjr}^2) \\
\text{BestSubSetmodelAdjr}^2
\] |
| Extracting Model with Lowest C$^p$-Statistics | \[
\text{BestSubSetmodelCp} \leftarrow \text{which.min(}\text{ResultsofSubsetSelectionmodel}\$\text{cp}) \\
\text{BestSubSetmodelCp}
\] |
proc glmselect data=NBASalary;
  class pos (ref="C");
  model srgty = rankyn confc ageC pos tmpC rebC
       asc stc blkc pfC tpC / selection = forward(select=adjrsq stop=adjrsq choose=adjrsq);
  run;
  quit;
proc glmselect data=NBASalary;
  class pos (ref="C");
  model srgty = rankyn confc ageC pos tmpC rebC
       asc stc blkc pfC tpC / selection = forward(select=cp stop=cp choose=cp);
  run;
  quit;
proc glmselect data=NBASalary;
  class pos (ref="C");
  model srgty = rankyn confc ageC pos tmpC rebC
       asc stc blkc pfC tpC / selection = backward(select=adjrsq stop=adjrsq choose=adjrsq);
  run;
  quit;
proc glmselect data=NBASalary;
  class pos (ref="C");
  model srgty = rankyn confc ageC pos tmpC rebC
       asc stc blkc pfC tpC / selection = backward(select=cp stop=cp choose=cp);
  run;
  quit;
proc glmselect data=NBASalary;
  class pos (ref="C");
  model srgty = rankyn confc ageC pos tmpC rebC
       asc stc blkc pfC tpC / selection = stepwise(select=adjrsq stop=adjrsq choose=adjrsq);
  run;
  quit;
proc glmselect data=NBASalary;
  class pos (ref="C");
  model srgty = rankyn confc ageC pos tmpC rebC
       asc stc blkc pfC tpC / selection = stepwise(select=cp stop=cp choose=cp);
  run;
  quit;
proc glmselect data=NBASalary;
  class pos (ref="C");
  model srgty = rankyn confc ageC pos tmpC rebC
       asc stc blkc pfC tpC / selection = stepwise(select=significancelevel stop=significancelevel choose=adjrsq);
  run;
  quit;

Figure 13.9
The resulting output from the forward stepwise selection is shown below in Figure 13.10. Looking at and interpreting the results for the backwards and bothways approaches is exactly the same.

![Figure 13.10](image)

Figure 13.10 above also shows the related variables in each selected model subset. One can be seen that two new models are created from this analysis. Figure 13.11 below shows the resulting two models. Once the backward and bothways stepwise approaches are analyzed, in the same way the forward approach was as seen in Figure 13.10, no new models were formed. This meaning the resulting models from the backward and bothways approach were the same as the resulting models in the forward approach.
In summary, the stepwise approach presented two new possible final models. Now model selection will reveal which model is the best model out of all models found through variable selection.

| Forward Variable Selection | proc glmselect data= mydata;  
model y = x1 x2 x3 ... xn  
/ selection = forward(select=[insert criterion]  
stop=[insert criertion] choose=[insert criterion]);  
run ;  
quit; |
|---------------------------|---------------------------------|
| Backward Variable Selection | proc glmselect data= mydata ;  
model y = x1 x2 x3 ... xn  
/ selection = backward(select=[insert criterion]  
stop=[insert criertion] choose=[insert criterion]);  
run ;  
quit; |
| Bothways Variable Selection | proc glmselect data= mydata ;  
model y = x1 x2 x3 ... xn  
/ selection = stepwise(select=[insert criterion]  
stop=[insert criertion] choose=[insert criterion]);  
run ;  
quit; |
Model Selection:
There are multiple criteria that can be used to select a model. Each criterion is derived by different aspects of the model. In general, doing more is better when deciding on a model.

**Bayesian Information Criterion (BIC):**
The Schwarz Bayesian Information Criterion is based on the log likelihood function. The BIC score represents how well the model will perform on a new dataset. A smaller BIC Score yields the better model. Below is the formula for finding the BIC Score.

\[
\text{BIC} = -2 \cdot \text{loglikelihood} + k \cdot \log(n)
\]

Since loglikelihood describes the amount of support the model gives the parameters in the model, a larger output is preferred. Hence the larger the loglikelihood the smaller the BIC. This meaning we aim for BIC to be minimized when choosing a model.

**Akaike information criterion (AIC):**
Like the Schwarz Bayesian Information Criterion, the Akaike information criterion also depends on the loglikelihood function. Additionally, like BIC, models with smaller AIC scores are preferred.

Look at the formula below on how the AIC score is found.

\[
\text{AIC} = -2 \cdot \text{loglikelihood} + 2n
\]

It is true that a simpler model is always preferred, but at what cost? BIC and AIC challenge our models to include more parameters to create a better fit. The challenge is to not over-fit the model. Overfitting a model would result in a loss of vital information. This meaning the output would be considering more variables that are not necessary to consider and thus shifting the models focus.

As seen in both formulas, the number of parameters is taken into consideration. In turn, it is almost as if the criterion is penalizing the model for having more parameters. However, this means in order to have a good score it holds each parameter to a higher standard. In other words, the model must work harder and prove that all parameters are needed to receive a good score.

**Corrected Akaike information criterion (AICc):**
The difference between AICc and AIC is very slim, yet powerful. AICc is preferred when dealing with a dataset with a small sample size. Below is the formula for AICc.

\[
\text{AICc} = \text{AIC} + \frac{2k(k+1)}{n-k-1}
\]
For datasets that have larger sample sizes the right-hand side of the equation will converge to the original AIC Score. Refer to Figure 14.1 to see how this is represented.

Note: treat k as a constant

Then ...

$$\lim_{n \to \infty} \left[ \text{AIC} + \frac{2k(k+1)}{n-k-1} \right]$$

$$= \left[ \text{AIC} + \frac{2k(k+1)}{\infty} \right]$$

$$= \text{AIC} + 0 = \text{AIC}$$

Figure 14.1

**NBA Salary Example in R:**

In R there is an intuitive coding that allows you to look at many aspects of a model to compare them. This is called the glance function. The glance function outputs 11 descriptive statistics of the model. Although, before diving into this it is important to be able to find the individual statistics AIC and BIC. The code, as seen in Figure 14.2 below, is extremely intuitive. Note that since our data set is large only finding the AIC is okay.

![R code output](image)

Figure 14.2

Based on the output and resulting BIC and AIC, the best model based on BIC is modCroot1 and the best model based on AIC is also modCroot1. In this case, based on this information modCroot1 would be chosen. Although, the glance function can provide more information.

Figure 14.3 below shows the glance code, which calls for the broom package, along with the resulting output.
The most important aspect of the glance function is the AIC and BIC output. As noted before the AIC and BIC indicate modCroot1 being the best model. This is reinforced by modCroot1 having the highest f-statistic, highest loglikelihood, and least degrees of freedom. Hence, modCroot1 is the best model and is chosen through model selection.

<table>
<thead>
<tr>
<th>Finding BIC</th>
<th>BIC(model)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finding AIC</td>
<td>AIC(model)</td>
</tr>
<tr>
<td>Running Glance Function</td>
<td>Library(broom) glance(model)</td>
</tr>
</tbody>
</table>

**NBA Salary Example in SAS:**

In SAS finding the BIC, AIC and AICC of a model involves a new method of “proc mixed”. Figure 14.4 shows the code to find these criteria.

```sas
proc mixed data = NBASalary method=ML;
    class pos (ref="C");
    model sqrtty = rankyn confC ageC pos tmpC rebC
                   asC stC blkC pFc tpC;
    run;
    quit;

proc mixed data = NBASalary method=ML;
    model sqrtty = tpC ageC pFc tmpC blkC asC rebC rankyn confC ;
    run;
    quit;

proc mixed data = NBASalary method=ML;
    model sqrtty = tpC ageC pFc tmpC blkC asC rebC rankyn ;
    run;
    quit;
```
Figure 14.5 below shows the output for each model. Based on these results the second model found in variable selection is the best model. Unlike R, SAS does not have a glance function. Although, the fit statistics does show the log likelihood, AIC, BIC and AICC. Likewise, other model properties can be found by running the model statement.

<table>
<thead>
<tr>
<th></th>
<th>Full centered model</th>
<th>Variable Selection 1</th>
<th>Variable Selection 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fit Statistics</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-2 Log Likelihood</td>
<td>7287.0</td>
<td>7273.1</td>
<td>7274.2</td>
</tr>
<tr>
<td>AIC (Smaller is Better)</td>
<td>7301.0</td>
<td>7295.1</td>
<td>7294.2</td>
</tr>
<tr>
<td>AICC (Smaller is Better)</td>
<td>7302.5</td>
<td>7295.8</td>
<td>7294.8</td>
</tr>
<tr>
<td>BIC (Smaller is Better)</td>
<td>7370.5</td>
<td>7340.1</td>
<td>7335.1</td>
</tr>
</tbody>
</table>

**Finding Fit Statistics**

```
proc mixed data = mydata method=ML;
  model y = x1 x2 x3 ... xn;
  asC stC blkC pfC tpC;
  run;
  quit;
```

**Adding Variable Interactions:**

Once a final model is found, it is important to check if any interactions are significant. An interaction is significant if its p value is below 0.05. An interaction is a combination of two or more x-variables by multiplication. For purposes of a model being applicable to a real life scenario, only two way interaction should be considered.

**NBA Salary Example in R:**

Figure 15.1 below shows how to check interactions in R along with the results. If an interaction is significant, R will place an asterisk next to it. Otherwise, look at the p-value. If the p-value is smaller than 0.05 then the interaction is significant.
From Figure 15.1 both ageC*tmpC and rebC*pfc were significant. Hence a new model with these two interactions should be made. Figure 15.2 below shows the new interaction model along with the model summary.

---

interactionmodel <- lm(sqrt(sall920)-ageC+tmpC+rebC+asc+bKc +pfc+pfc)^2, data=NBASalary)

Coefficients:

|                | Estimate | Std. Error | t value | Pr(>|t|) |
|----------------|----------|------------|---------|----------|
| (Intercept)    | 2.467e+03| 7.945e+01  | 31.058  | < 2e-16  |
| ageC           | 5.151e+01| 1.186e+01  | 4.385   | 3.87e-06 |
| tmpC           | 1.610e-01| 1.931e-01  | 0.833   | 0.405122 |
| rebC           | 2.037e-00| 9.379e-01  | 2.172   | 0.030398 |
| asc            | 1.705e-00| 1.091e+00  | 1.562   | 0.119064 |
| bKc            | 8.062e-01| 4.420e+00  | 0.182   | 0.853578 |
| pfc            | -6.367e+00| 1.833e+00 | -3.483   | 0.000548 |
| ageC:tmpC      | 9.433e-02| 3.274e-02  | 2.882   | 0.004163 |
| ageC:rebC      | 1.784e-01| 1.180e-01  | 1.512   | 0.131402 |
| ageC:asc       | 1.701e-01| 1.327e-01  | 1.283   | 0.200345 |
| ageC:bKc       | -3.712e-01| 5.923e-01 | -0.627   | 0.531165 |
| ageC:pfc       | -1.030e-01| 2.673e-01  | -0.385   | 0.700233 |
| ageC:tpC       | -1.077e-01| 5.943e-02  | -0.183   | 0.070623 |
| tmpC:rebC      | -1.715e-04| 1.558e-03  | -0.162   | 0.871312 |
| tmpC:asc       | 1.262e-03| 1.229e-03  | 0.105   | 0.290365 |
| tmpC:bKc       | 5.845e-03| 5.521e-03  | 0.109   | 0.290365 |
| tmpC:pfc       | -1.109e-03| 2.864e-03  | -0.387   | 0.098859 |
| tmpC:tpC       | -1.801e-04| 3.778e-04  | -0.477   | 0.038580 |
| rebC:asc       | -3.205e-03| 3.792e-03  | -0.845   | 0.398850 |
| rebC:bKc       | -1.236e-03| 9.101e-03  | -0.136   | 0.892044 |
| rebC:pfc       | -6.181e-02| 5.994e-03  | -2.700   | 0.007228 |
| rebC:tpC       | 1.602e-03| 1.100e-03  | 1.456   | 0.146248 |
| asc:bKc        | 2.532e-02| 2.199e-02  | 1.147   | 0.251898 |
| asc:pfc        | 9.183e-03| 1.232e-02  | -0.745   | 0.456729 |
| asc:tpC        | -6.573e-04| 3.121e-03  | -0.498   | 0.618941 |
| bKc:pfc        | 4.890e-02| 4.001e-02  | 1.222   | 0.222354 |
| bKc:tpC        | -6.356e-01| 7.963e-03  | -0.799   | 0.424503 |
| pfc:tpC        | 5.394e-04| 5.588e-03  | 0.150   | 0.880590 |

---

Residual standard error: 903 on 431 degrees of freedom
Multiple R-squared: 0.2611, Adjusted R-squared: 0.2084
F-statistic: 100.18 on 9 and 431 DF, p-value: < 2.2e-16

---

Figure 15.1

---

Figure 15.2
Based on the model summary of the interaction model tmpC is no longer significant. Although, since there is an interaction with tmpC involved that is significant within the model, tmpC cannot be removed. Therefore, the model seen in Figure 15.2 above is the final model.

### Checking Two Way Interactions

<table>
<thead>
<tr>
<th>Interaction</th>
<th>Coefficient 1</th>
<th>Coefficient 2</th>
<th>p-value 1</th>
<th>p-value 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>ageC*tmpC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ageC*pfC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tmpC*tmpC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ageC*tmpC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ageC*pfC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tmpC*pfC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ageC*tmpC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ageC*pfC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tmpC*pfC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ageC*tmpC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ageC*pfC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tmpC*pfC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ageC*tmpC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ageC*pfC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### NBA Salary Example in SAS:

Checking interactions in SAS is quite time consuming. Figure 15.3 below shows the code for finding interactions in SAS along with a portion of the output. As seen in Figure 15.3, SAS presents more than just the two way interactions and does not signify which interactions are significant. Due to this, the interactions and p-values must be looked at individually.

As described before, only the two way interactions will be considered. It can be seen that both ageC*tmpC and tpC*blkC are significant (pointed to with red arrow). After scrolling through the additional output not shown above the significant interactions in this model include: tpC*ageC, tpC*pfC, tmpC*ageC, tpC*blkC, blkC*rankyn.
A new model is made to include all of the above interactions. Figure 15.4 below shows the code for this interaction model along with the model summary.

```plaintext
ods graphics on;
proc glm data = NBASalary;
   model sqrtY = tpC ageC pfC tmpC blkC asC rebC rankyn tpC*ageC
      tpC*pfC tmpC*ageC tpC*blkC blkC*rankyn ;
run;
quit;
```

| Parameter | Estimate  | Standard Error | t Value | Pr > |t| |
|-----------|-----------|----------------|---------|------|---|
| Intercept | 2418.843845 | 85.54665429 | 28.28 | <0.0001 | |
| tpC       | 1.546376   | 0.24437926 | 6.33  | <0.0001 | |
| ageC      | 58.364247  | 11.56924788 | 5.05  | <0.0001 | |
| pfC       | -7.128882  | 1.10857477 | -6.55 | <0.0001 | |
| tmpC      | 0.218008   | 0.15439863 | 1.41  | 0.1587  | |
| blkC      | 8.003049   | 3.24615159 | 2.47  | 0.0141  | |
| asC       | 1.799652   | 0.51655996 | 3.47  | 0.0000  | |
| rebC      | 0.983568   | 0.40471679 | 2.43  | 0.0155  | |
| rankyn    | -191.777961| 88.33164570 | -2.17 | 0.0305 | |
| tpC*ageC  | -0.059449  | 0.04417762 | -1.35 | 0.1791 | |
| tpC*pfC   | -0.004039  | 0.00141026 | -2.86 | 0.0044 | |
| ageC*tmpC | 0.101310   | 0.02738298 | 3.70  | 0.0002 | |
| tpC*blkC  | 0.004081   | 0.00375268 | 1.08  | 0.2825 | |
| blkC*rankyn | -3.633465  | 2.80085428 | -1.29 | 0.1965 | |

Figure 15.4

Figure 15.4 shows that the interactions tpC*blkC tpC*ageC and blkC*rankyn are not significant and therefore should be removed from the model. Figure 15.5 below shows the final model in SAS after removing these three interactions.

```plaintext
ods graphics on;
proc glm data = NBASalary PLOTS=all;
   model sqrtY = tpC ageC pfC tmpC blkC asC rebC rankyn
      tpC*pfC tmpC*ageC ;
output out = outsalary r=residfinal p=predfinal;
run;
quit;
```

Figure 15.5

**Conclusions:**

**Case Study Results:**

Since analysis was done in both R and SAS, there are currently two “best” models. When comparing the AIC and BIC results from both models (refer to Figure 14.2 and 14.5), the model
in R is the true “best” model. Hence the final model for this analysis is represented in Figure 15 below.

\[
\hat{y} = (58.93x_3 + 0.2189x_5 + 2.179x_6 + 1.928x_7 + 6.158x_9 + -7.783x_{11} + 1.145x_{12} + 0.07137x_3x_5 + -0.008581x_6x_{11} + 246)^2
\]

Notice how the variables and intercept is squared, this is due to how the y variable had a square root transformation. Based on the final model the most important variables to predicting a players NBA Salary are, ages, total minutes played, rebounds, assists, blocks, personal fouls, total points, the interaction of age and total minutes played and the interaction between rebounds and personal fouls. These variables and interactions are the factors that effect a players salary the most, whether it be in a positive or negative way.

Appendix:

References/Works Cited:


“2019-20 NBA Team Ratings.” Basketball.

“2020-21 NBA Player Contracts.” Basketball.


“2019-20 NBA Team Ratings.” Basketball.


jcf2d, Written by. “University of Virginia Library Research Data Services Sciences.” Research Data Services Sciences.


“NBA Stats & League Leaders.” *NBA Stats & League Leaders* | FOX Sports.


**R Code:**

```r
#Changing "pos" to a factor
NBASalary$pos <- as.factor(NBASalary$pos)

NBASalary

#Finding Full Model
mod1 <- lm(sal1920~rank1920+conf+age1819+pos+tmp1819+reb1819+as1819+st1819+
blk1819+to1819+pf1819+tp1819, data= NBASalary)

summary(mod1)

#Normal Q-Q Plot for mod1
plot(mod1, which=2)

#Shapiro Wilk Normality Test for mod1
shapiro.test(mod1$residuals)

#Residual v Fitted Value Plot for mod1
plot(mod1,which=1)

#BP test for Constant Variance mod(1)
library(lmtest)
bptest(mod1)

#Check model summary for statistically significant
#variables, Coefficient of Determination and Adjusted
#Coefficient of Determination
summary(mod1)

##########################################
```


###Perform a Box-Cox Transformation in an attempt to fix normality assumption in mod(1)###

library(MASS)

boxcox(mod1)

#Adjusting margins in the plot to have a better visual estimate of lambda
boxcox(mod1, data=NBASalary, lambda = seq(0, 1, 1/10), plotit = TRUE)

#Finding the exact lambda of mod(1) to be more precise

#with the drawn conclusions

b=boxcox(mod1, data=NBASalary, lambda = seq(0, 1, 1/10), plotit = TRUE)

lambda= b$x

loglik=b$y

bc = cbind(lambda, loglik)

bc = bc[order(-loglik), ]

bc

#Creating two new models from Box Cox Transformation

mod2root <- lm(sqrt(sal1920)~rank1920+conf+age1819+pos+tmp1819+reb1819+as1819+st1819+blk1819+to1819+pf1819+tp1819, data=NBASalary)

summary(mod2root)

mod3log <- lm(log(sal1920)~rank1920+conf+age1819+pos+tmp1819+reb1819+as1819+st1819+blk1819+to1819+pf1819+tp1819, data=NBASalary)

summary(mod3log)

###Check model assumptions for mod2root and mod3log###

#Normal Q-Q Plot for mod2root

plot(mod2root, which=2)

#Shapiro Wilk Normality Test for mod2root

shapiro.test(mod2root$residuals)

#Residual v Fitted Value Plot for mod2root

plot(mod2root, which=1)
#BP test for Constant Variance mod2root
library(car)
ncvTest(mod2root)

#Normal Q-Q Plot for mod3log
plot(mod3log, which=2)

#Shapiro Wilk Normality Test for mod3log
shapiro.test(mod3log$residuals)

#Residual v Fitted Value Plot for mod3log
plot(mod3log,which=1)

#BP test for Constant Variance mod3log
library(car)
ncvTest(mod3log)

######################################################
########Performing Principal Component Analysis########
######################################################

##install fatoextra##
##only with numeric values!!
library(factoextra)
nba.pca <- prcomp(NBASalary_Numeric_norank, scale = TRUE)
print(nba.pca)
get_eigenvalue(nba.pca)

#Creating PCA1 and PCA 2 variables
attach(NBASalary)
NBASalary$PCA1 <- ((-0.03623085)*conf+(0.14967190)*age1819+
(0.36856571)*tmp1819+(0.34957210)*reb1819+
(0.33026540)*as1819+(0.35147127)*st1819+(0.27454553)*blk1819+(0.36922331)*to1819+(0.36511855)*pf1819+
(0.37499915)*tp1819)
PCA1

attach(NBASalary)
NBASalary$PCA2 <- ((0.950287863)*conf+(-0.223078630)*age1819+ (-0.026008047)*tmp1819+(0.119911327)*reb1819+ (-0.054476138)*as1819+(-...
0.030073797)*st1819+(0.160227188)*blk1819+(0.014087289)*to1819+(0.048324252)*pf1819+(-0.007432231)*tp1819)

PCA2
#Checking diagnostics on new modPCA model
modPCA <- lm(sal1920~PCA1+PCA2, data= NBASalary)
summary(modPCA2)

#Normal Q-Q Plot for modPCA
plot(modPCA, which=2)
#Shapiro Wilk Normality Test for modPCA
shapiro.test(modPCA$residuals)
#Residual v Fitted Value Plot for modPCA
plot(modPCA,which=1)
#BP test for Constant Variance modPCA
library(lmtest)
bptest(modPCA)

#The goal was to fix constant variance, this did not work so stop here with this model

#############################################################
###########Variance Inflation Factor####################
#############################################################
library(car)
vif(mod1)
vif(mod2root)
vif(mod3log)

#Revise mod1, mod2root and mod3log by removing to1819
#Check model assumptions of mod1, mod2root and mod3log
mod1<- lm(sal1920~rank1920+conf+age1819+pos+tmp1819+reb1819+as1819+st1819+blk1819+pf1819+tp1819, data= NBASalary)
summary(mod1)

#Normal Q-Q Plot for mod1
plot(mod1, which=2)
# Shapiro Wilk Normality Test for mod1
shapiro.test(mod1$residuals)

# Residual v Fitted Value Plot for mod1
plot(mod1, which=1)

# BP test for Constant Variance mod1
library(lmtest)
bptest(mod1)

mod2root <- lm(sqrt(sal1920)~rank1920+conf+age1819+pos+
    tmp1819+reb1819+as1819+st1819+blk1819
    +pf1819+tp1819, data=NBASalary)

summary(mod2root)

# Normal Q-Q Plot for mod2root
plot(mod2root, which=2)

# Shapiro Wilk Normality Test for mod2root
shapiro.test(mod2root$residuals)

# Residual v Fitted Value Plot for mod2root
plot(mod2root, which=1)

# BP test for Constant Variance mod2root
library(lmtest)
bptest(mod2root)

mod3log <- lm(log(sal1920)~rank1920+conf+age1819+pos+
    tmp1819+reb1819+as1819+st1819+blk1819
    +pf1819+tp1819, data=NBASalary)

summary(mod3log)

# Normal Q-Q Plot for mod3log
plot(mod3log, which=2)

# Shapiro Wilk Normality Test for mod3log
shapiro.test(mod3log$residuals)

# Residual v Fitted Value Plot for mod3log
plot(mod3log, which=1)

# BP test for Constant Variance mod3log
library(lmtest)

bptest(mod3log)

#################################################################
# Centering Variables#############################################
#################################################################
attach(NBASalary)

NBASalary$ageC <- (age1819 - mean(age1819))
NBASalary$confC <- (conf - mean(conf))
NBASalary$tmpC <- (tmp1819 - mean(tmp1819))
NBASalary$rebC <- (reb1819 - mean(reb1819))
NBASalary$asC <- (as1819 - mean(as1819))
NBASalary$stC <- (st1819 - mean(st1819))
NBASalary$blkC <- (blk1819 - mean(blk1819))
NBASalary$toC <- (to1819 - mean(to1819))
NBASalary$pfC <- (pf1819 - mean(pf1819))
NBASalary$tpC <- (tp1819 - mean(tp1819))

attach(NBASalary)

# New models with centered variables
mod1 <- lm(sal1920 ~ rankyn + confC + ageC + pos + tmpC
          + rebC + asC + stC + blkC
          + pfC + tpC, data = NBASalary)

summary(mod1)
# Normal Q-Q Plot
plot(mod1, which=2)
# Shapiro Wilk Normality Test
shapiro.test(mod1$residuals)
# Residual v Fitted Value Plot
plot(mod1, which=1)
# BP test for Constant Variance
library(lmtest)
bptest(mod1)
modCroot <- lm(sqrt(sal1920)~rankyn+confC+ageC+pos+tmpC +rebC+asC+stC+blkC +pfC+tpC, data= NBASalary)
summary(modCroot)

#Normal Q-Q Plot
plot(modCroot, which=2)

#Shapiro Wilk Normality Test
shapiro.test(modCroot$residuals)

#Residual v Fitted Value Plot
plot(modCroot,which=1)

#BP test for Constant Variance
library(lmtest)
bptest(modCroot)

modClog <- lm(log(sal1920)~rankyn+confC+ageC+pos+tmpC +rebC+asC+stC+blkC +pfC+tpC, data= NBASalary)
summary(modClog)

#Normal Q-Q Plot
plot(modClog, which=2)

#Shapiro Wilk Normality Test
shapiro.test(modClog$residuals)

#Residual v Fitted Value Plot
plot(modClog,which=1)

#BP test for Constant Variance
library(lmtest)
bptest(modClog)

######################################################
################Variable Selection#######################
######################################################
##Only model moving forward##
modCroot <- lm(sqrt(sal1920)~rankyn+confC+ageC+pos+tmpC +rebC+asC+stC+blkC+pfC+tpC, data= NBASalary)

#Step-Wise Approach with all three directions#
# Backward variable selection
fullmodCroot <- lm(sqrt(sal1920) ~ rankyn + confC + ageC + pos + tmpC + rebC + asC + stC + blkC + pfC + tpC, data = NBASalary)
nothing <- lm(sqrt(sal1920) ~ 1, data = NBASalary)
backwardsCroot = step(fullmodCroot)
formula(backwardsCroot)
summary(backwardsCroot)

# forward variable selection
forwardsCroot = step(nothing, scope=list(lower=formula(nothing),upper=formula(fullmodCroot)), direction="forward")
formula(forwardsCroot)
summary(forwardsCroot)

# bothways variable selection
bothwaysCroot = step(nothing, list(lower=formula(nothing),upper=formula(fullmodCroot)), direction="both", trace=0)
formula(bothwaysCroot)
summary(bothwaysCroot)

## new model from step wise with mod2root##
modCroot1 <- lm(sqrt(sal1920) ~ rankyn + ageC + tmpC + rebC + asC + blkC + pfC + tpC, data = NBASalary)
summary(modCroot1)

# Exhaustive approach
library(leaps)
subsetselectionmodCroot <- regsubsets(sqrt(sal1920) ~ rankyn + confC + ageC + pos + tmpC + rebC + asC + stC + blkC + pfC + tpC, wt=NULL, nbest=1, nvmax=20, data = NBASalary)
ResultsofSubsetSelectionmodCroot <- summary(subsetselectionmodCroot)
ResultsofSubsetSelectionmodCroot

BestSubSetmodCrootAdjr2 <- which.max(ResultsofSubsetSelectionmodCroot$adjr2)
BestSubSetmodCrootAdjr2 # This says subset 11

BestSubSetmodCrootCp <- which.min(ResultsofSubsetSelectionmodCroot$cp)
BestSubSetmodCrootCp # This says subset 8

par(mfrow=c(1,2))
plot(ResultofSubsetSelectionmodCroot$adjr2, xlab="variables selected", ylab="adjr2")
plot(ResultofSubsetSelectionmodCroot$cp, xlab = "Variables selected", ylab = "cp")

#Now find the coefficients of subset 11 and 8
coef(subsetselectionmodCroot,11)
coef(subsetselectionmodCroot,8)

#Two new models from exhaustive approach
modCroot2 <- lm(sqrt(sal1920)~rankyn+confC+ageC+pos+tmpC+rebC+asC+blkC+pfC+tpC, data=NBASalary)

#Currently Existing Models
modCroot <- lm(sqrt(sal1920)~rankyn+confC+ageC+pos+tmpC
+rebC+asC+stC+blkC
+pfC+tpC, data=NBASalary)
summary(modCroot)
modCroot1 <- lm(sqrt(sal1920)~rankyn+ageC+tmpC+rebC+asC+blkC+pfC+tpC, data=NBASalary)
summary(modCroot1)
modCroot2 <- lm(sqrt(sal1920)~rankyn+confC+ageC+pos+tmpC+rebC+asC+blkC+pfC+tpC, data=NBASalary)
summary(modCroot2)

#model comparison
#Choosing a model
#Re-check BIC, AIC : we want to minimize
AIC(modCroot)
AIC(modCroot1)
AIC(modCroot2)
BIC(modCroot)
BIC(modCroot1)
BIC(modCroot2)

#compare best models normality
par(mfrow=c(2,2))
plot(modCroot, which=2, xlab="modCroot")
plot(modCroot1, which=2, xlab="modCroot1")
plot(modCroot2, which=2, xlab="modCroot2")

# Normality Test
shapiro.test(modCroot$residuals)
shapiro.test(modCroot1$residuals)
shapiro.test(modCroot2$residuals)

# compare best models constant variance
par(mfrow=c(2,2))
plot(modCroot, which=1)
plot(modCroot1, which=1)
plot(modCroot2, which=1)

#BP test for Constant Variance
library(lmtest)
bptest(modCroot)
bptest(modCroot1)
bptest(modCroot2)
bptest(modCroot3)

#glance involves all of the above
library(broom)

# glanced involves all of the above

# from this we choose modCroot2 as our best model
summary(modCroot2)

# interactions in R

checkinteractions<- lm(sqrt(sal1920)~(ageC +tmpC+rebC+asC+blkC  +pfC+tpC)^2, data=NBASalary)
summary(checkinteractions)

interactionmodel <- lm(sqrt(sal1920)~ageC+tmpC+rebC+asC+blkC+pfC+tpC+ageC:tmpC+rebC:pfC,data=NBASalary)
summary(interactionmodel)

library(lmtest)
bptest(interactionmodel)

finalmodel <- lm(sqrt(sal1920)~ageC+tmpC+rebC+asC+blkC+pfC+tpC+ageC:tmpC+rebC:pfC,data=NBASalary)
summary(finalmodel)
# BP test for Constant Variance

library(lmtest)

bptest(finalmodel)

# Normality Test

shapiro.test(finalmodel$residuals)

# Normal QQ plot and residual vs fitted plot

par(mfrow=c(1,2))

plot(finalmodel, which=1)

plot(finalmodel, which=2)

AIC(finalmodel)

BIC(finalmodel)

**SAS Code:**

*Finding Original Model Parameters and Plots for Constant Variance and Normality;

ods graphics on;

proc glm data= NBASalary PLOTS=ALL ;
   class pos (ref="C");
   model sal1920 = rank1920 conf age1819 pos tmp1819 reb1819
         as1819 st1819 blk1819 to1819 pf1819 tp1819 / solution;
   output out = outsalary r=resid p = pred;
   run ;
   quit;

*This runs the Shapiro Wilk Normality Test;

proc univariate data = outsalary normal;
   var resid ;
   run;

*Runing Leveans Test for Constant Variance ;

data outsalary2 ;
set outsalary ;

aresid = abs(resid);
PROC GLM DATA = OUTSALARY2;
  CLASS POS (REF="C");
    MODEL ARESID = RANK1920 CONF AGE1819 POS TMP1819 REB1819
               AS1819 ST1819 BLK1819 TO1819 PF1819 TP1819;
  RUN;
  QUIT;
*boxcox transformation;
PROC TRANSREG DATA = NBASALARY SS2 DETAIL;
  MODEL BOXCOX(SAL1920 /LAMBDA= 0 TO 1 BY 0.01) = IDENTITY(RANK1920 CONF AGE1819 TMP1819 REB1819
               AS1819 ST1819 BLK1819 TO1819 PF1819 TP1819);
  RUN;
*sqrt transformation on y ;
DATA NBASALARY;
  SET NBASALARY;
  SQRTY = SQRT(SAL1920);
  RUN;
ODS GRAPHICS ON;
PROC GLM DATA= NBASALARY PLOTS=ALL ;
  CLASS POS (REF="C");
  MODEL SQRTY = RANK1920 CONF AGE1819 POS TMP1819 REB1819
               AS1819 ST1819 BLK1819 TO1819 PF1819 TP1819 / SOLUTION;
  OUTPUT OUT = OUTSALARY R=RESIDROOT P = PRED;
  RUN;
  QUIT;
*normality test on sqrt transformation ;
PROC UNIVARIATE DATA = OUTSALARY NORMAL;
  VAR RESIDROOT ;
  RUN;
*log transformation on y;
data NBASalary;
   set NBASalary;
   logy = log(sal1920);
run;
ods graphics on;
proc glm data= NBASalary PLOTS=ALL ;
   class pos (ref="C");
   model logy = rank1920 conf age1819 pos tmp1819 reb1819 as1819 st1819 blk1819 to1819 pf1819 tp1819 / solution;
   output out = outsalary r=residlog p = pred;
   run ;
quit;

*normality test on log transformation ;
proc univariate data = outsalary normal;
   var residlog ;
   run;

*Principal Component Analysis ;
proc factor data = NBASalary eigenvectors
   mineigen = 1
   method = prin
   priors = one
   rotate = varimax
   round;
   var conf age1819 tmp1819 reb1819 as1819 st1819 blk1819 to1819 pf1819 tp1819  ;
   run;

*Adding PCA1 and PCA2 as new variables to the data frame;
data NBASalary;
set NBASalary;

PCA1 = (-4/100)*conf+(15/100)*age1819+(37/100)*tmp1819+(35/100)*reb1819+
     (33/100)*as1819+(35/100)*st1819+(27/100)*blk1819+(37/100)*to1819+(37/100)*pf1819+(37/100)*tp1819;
Run;

data NBASalary;

set NBASalary;

PCA2 = (95/100)*conf+(-22/100)*age1819+(-3/100)*tmp1819+(12/100)*reb1819+
     (-5/100)*as1819+(-3/100)*st1819+(16/100)*blk1819+(1/100)*to1819+(5/100)*pf1819+(-1/100)*tp1819;
run;

*Create a new model containing the two PCA's;
ods graphics on;
proc glm data= NBASalary plots=ALL ;
   model sal1920 = PCA1 PCA2 / solution;
   output out = outsalary r=residPCA p = predPCA;
   run ;
   quit;

*Runing Leveans Test for Constant Variance;
proc glm data= outsalary2 ;
data outsalary2 ;
set outsalary ;
aresid = abs(residPCA);
run;
proc glm data = outsalary2;
   model aresid =PCA1 PCA2;
   run;
   quit;

*Finding Variance Inflation Factors
position was taken out inorder to run the vif on the model using
proc reg, this will alter the results so it will not be exact to R although, it will still be representative of the results;

proc reg data= NBASalary;
    model sal1920 = rank1920 conf age1819 tmp1819 reb1819 as1819 st1819 blk1819 to1819 pf1819 tp1819/ vif;
    run;
    quit;

proc reg data= NBASalary;
    model sqrty = rank1920 conf age1819 tmp1819 reb1819 as1819 st1819 blk1819 to1819 pf1819 tp1819/ vif;
    run;
    quit;

proc reg data= NBASalary;
    model logy = rank1920 conf age1819 tmp1819 reb1819 as1819 st1819 blk1819 to1819 pf1819 tp1819/ vif;
    run;
    quit;

*investigate the 3 models above after removing to1819 since it had a high VIF;

ods graphics on;
proc glm data= NBASalary PLOTS=ALL;
    class pos (ref="C");
    model sal1920 = rank1920 conf age1819 pos tmp1819 reb1819 as1819 st1819 blk1819 pf1819 tp1819 / solution;
    output out = outsalary r=resid1 p = pred;
    run;
    quit;

*This runs the Shapiro Wilk Normality Test;
proc univariate data = outsalary normal;
    var resid1;
run;

*Runing Leveans Test for Constant Variance  ;
data  outsalary2 ;
set  outsalary ;
aresid = abs(resid1);
run;

proc glm data = outsalary2;
  class pos (ref="C");
  model aresid =rank1920 conf age1819 pos tmp1819 reb1819
              as1819 st1819 blk1819 pf1819 tp1819 ;
  run;
quit;

ods graphics on;
proc glm data= NBASalary PLOTS=ALL ;
  class pos (ref="C");
  model sqrty = rank1920 conf age1819 pos tmp1819 reb1819
              as1819 st1819 blk1819 pf1819 tp1819 / solution;
  output out = outsalary r=residsqrt p = pred;
  run ;
quit;

*This runs the Shapiro Wilk Normality Test;
proc univariate data = outsalary normal;
  var residsqrt ;
  run;

*Runing Leveans Test for Constant Variance  ;
data  outsalary2 ;
set  outsalary ;
aresid = abs(residsqrt);
run;
proc glm data = outsalary2;
    class pos (ref="C");
    model aresid =rank1920 conf age1819 pos tmp1819 reb1819
            as1819 st1819 blk1819 pf1819 tp1819;
    run;
    quit;
ods graphics on;
proc glm data=NBASalary PLOTS=ALL;
    class pos (ref="C");
    model logy = rank1920 conf age1819 pos tmp1819 reb1819
            as1819 st1819 blk1819 pf1819 tp1819 / solution;
    output out = outsalary r=residlog p = pred;
    run;
    quit;
*This runs the Shapiro Wilk Normality Test;
proc univariate data = outsalary normal;
    var residlog;
    run;
*Runing Leveans Test for Constant Variance ;
data outsalary2;
set outsalary;
aresid = abs(residlog);
run;
proc glm data = outsalary2;
    class pos (ref="C");
    model aresid =rank1920 conf age1819 pos tmp1819 reb1819
            as1819 st1819 blk1819 pf1819 tp1819;
    run;
    quit;
*Centering all Variables ONLY QUANTATIVE VARIABLES;
*First find the mean of each variable;
proc means data= NBAsalary;
  VAR rank1920 conf age1819 tmp1819 reb1819
       as1819 st1819 blk1819 to1819 pf1819 tp1819;
  run;
data NBASalary;
  set NBASalary;
  confC = conf-0.5238095;
run;
data NBASalary;
  set NBASalary;
  ageC = age1819-24.9750567;
run;
data NBASalary;
  set NBASalary;
  tmpC = tmp1819-1160.52;
run;
data NBASalary;
  set NBASalary;
  rebC = reb1819-251.2426304;
run;
data NBASalary;
  set NBASalary;
  rebC = reb1819-131.9047619;
run;
data NBASalary;
  set NBASalary;
  asC = as1819-131.9047619;
run;
data NBASalary;
  set NBASalary;
stC = st1819-42.3582766;
run;
data NBASalary;
  set NBASalary;
blkC = blk1819-27.5782313;
run;
data NBASalary;
  set NBASalary;
toC = to1819-73.0385488;
run;
data NBASalary;
  set NBASalary;
pfC = pf1819-112.9342404;
run;
data NBASalary;
  set NBASalary;
  tpC = tp1819-612.0204082;
run;
*make 3 new models with centered variable and rankyn (yes no variable);
  ods graphics on;
proc glm data= NBASalary PLOTS=ALL ;
  class pos (ref="C");
  model sal1920 = rankyn confC ageC pos tmpC rebC
            asC stC blkC pfC tpC / solution;
  output out = outsalary r=residc p = pred;
  run ;
quit;

*This runs the Shapiro Wilk Normality Test;
proc univariate data = outsalary normal;
        var residc ;
        run;
*Runing Leveans Test for Constant Variance ;
data  outsalary2 ;
set  outsalary ;
aresid = abs(residc);
run;
proc glm data = outsalary2;
        model aresid = rankyn confC ageC pos tmpC rebC
                asC stC blkC pfC tpC ;
        run;
        quit;
ods graphics on;
proc glm data= NBASalary PLOTS=ALL ;
        class pos (ref="C");
        model sqrty = rankyn confC ageC pos tmpC rebC
                asC stC blkC pfC tpC / solution;
        output out = outsalary r=residsqrtec p = predsqrt;
        run ;
        quit;
*This runs the Shapiro Wilk Normality Test;
proc univariate data = outsalary normal;
        var residsqrtec ;
        run;
*Runing Leveans Test for Constant Variance ;
data  outsalary2 ;
set outsalary;
aresid = abs(residsqrtc);
run;
proc glm data = outsalary2;
    model aresid = rankyn confC ageC pos tmpC rebC
        asC stC blkC pfC tpC;
    run;
    quit;
ods graphics on;
proc glm data= NBASalary PLOTS=ALL;
    class pos (ref="C");
    model logy = rankyn confC ageC pos tmpC rebC
            asC stC blkC pfC tpC / solution;
    output out = outsalary r=residlogc p = predlog;
    run;
    quit;
*This runs the Shapiro Wilk Normality Test;
proc univariate data = outsalary normal;
    var residlogc;
    run;
*Runing Leveans Test for Constant Variance
data outsalary2;
set outsalary;
aresid = abs(residlogc);
run;
proc glm data = outsalary2;
    model aresid = rankyn confC ageC pos tmpC rebC
        asC stC blkC pfC tpC;
*stepwisemethod for model selection forward

out of the three existing models above the only ones with constant variance and normality
is sqrty, so only this model will undergo variable selection;

proc glmselect data= NBASalary ;
 class pos (ref="C");
 model sqrty = rankyn confC ageC pos tmpC rebC asC stC blkC pfC tpC / selection = forward(select=adjrsq stop=adjrsq choose=adjrsq);
 run ;
 quit;

proc glmselect data= NBASalary ;
 class pos (ref="C");
 model sqrty = rankyn confC ageC pos tmpC rebC asC stC blkC pfC tpC / selection = forward(select=cp stop=cp choose=cp);
 run ;
 quit;

proc glmselect data= NBASalary ;
 class pos (ref="C");
 model sqrty = rankyn confC ageC pos tmpC rebC asC stC blkC pfC tpC/ selection = forward(select=significancelevel stop=significancelevel choose=adjrsq);
 run ;

quit;
*two new models from stepwise variable selection;
ods graphics on;
proc glm data=NBASalary PLOTS=ALL;
   model sqrty = tpC ageC pfC tmpC blkC asC rebC rankyn confC / solution;
   output out = outsalary r=residsqrt1 p = predsqrt1;
   run;
   quit;
ods graphics on;
proc glm data=NBASalary PLOTS=ALL;
   model sqrty = tpC ageC pfC tmpC blkC asC rebC rankyn / solution;
   output out = outsalary r=residsqrt2 p = predsqrt2;
   run;
   quit;
*stepwisemethod for model selection backwards;
proc glmselect data=NBASalary;
   class pos (ref="C");
   model sqrty = rankyn confC ageC pos tmpC rebC asC stC blkC pfC tpC / selection = backward(select=adjrsq stop=adjrsq choose=adjrsq);
   run;
   quit;
proc glmselect data=NBASalary;
   class pos (ref="C");
   model sqrty = rankyn confC ageC pos tmpC rebC asC stC blkC pfC tpC / selection = backward(select=cp stop=cp choose=cp);
   run;
   quit;
proc glmselect data=NBASalary;
   class pos (ref="C");
model sqrty = rankyn confC ageC pos tmpC rebC asC stC blkC pfC tpC / selection = backward(select=significancelevel stop=significancelevel choose=adjrsq);
run ;
quit;

*no new models from backward selection;

*stepwise method for model selection both ways;
proc glmselect data= NBASalary ;
class pos (ref="C");
model sqrty = rankyn confC ageC pos tmpC rebC asC stC blkC pfC tpC / selection = stepwise(select=adjrsq stop=adjrsq choose=adjrsq);
run ;
quit;

proc glmselect data= NBASalary ;
class pos (ref="C");
model sqrty = rankyn confC ageC pos tmpC rebC asC stC blkC pfC tpC / selection = stepwise(select=cp stop=cp choose=cp);
run ;
quit;

proc glmselect data= NBASalary ;
class pos (ref="C");
model sqrty = rankyn confC ageC pos tmpC rebC asC stC blkC pfC tpC / selection = stepwise(select=significancelevel stop=significancelevel choose=adjrsq);
run ;
quit;

*no new models from bothways selection;

*Existing models;

*modelsqrt;
ods graphics on;
proc glm data=NBASalary PLOTS=ALL;
   class pos (ref="C");
   model sqrty = rankyn confC ageC pos tmpC rebC asC stC blkC pfC tpC / solution;
   output out = outsalary r=residsqrt p = predsqrt;
run;
quit;

*This runs the Shapiro Wilk Normality Test;
proc univariate data = outsalary normal;
   var residsqrt ;
   run;
*running Levene's Test for Constant Variance;
data  outsalary2 ;
set  outsalary ;
  aresid = abs(residsqrt);
run;
proc glm data = outsalary2;
   class pos (ref="C");
      model aresid = rank1920 conf age1819 pos tmp1819 reb1819 as1819 st1819 blk1819 pf1819 tp1819;
run;
quit;
*modelsqrt1;
ods graphics on;
proc glm data=NBASalary PLOTS=ALL;
   model sqrty = tpC ageC pfC tmpC blkC asC rebC rankyn confC / solution;
   output out = outsalary r=residsqrt1 p = predsqrt1;
run;
quit;

*This runs the Shapiro Wilk Normality Test;
proc univariate data = outsalary normal;
   var residsqrt1;
run;

*Runing Leveans Test for Constant Variance;
data outsalary2;
set outsalary;
aresid = abs(residsqrt1);
run;
proc glm data = outsalary2;
   model aresid =tpC ageC pfC tmpC blkC asC rebC rankyn confC;
run;
quit;

*model sqrt2;
ods graphics on;
proc glm data= NBASalary PLOTS=ALL;
   model sqrty = tpC ageC pfC tmpC blkC asC rebC rankyn / solution;
   output out = outsalary r=residsqrt2 p = predsqrt2;
run;
quit;

*This runs the Shapiro Wilk Normality Test;
proc univariate data = outsalary normal;
   var residsqrt2;
run;

*Runing Leveans Test for Constant Variance;
data outsalary2;
set outsalary;
aresid = abs(residsqrt2);
run;
proc glm data = outsalary2;
  model aresid = tpC ageC pfC tmpC blkC asC rebC rankyn;
  run;
  quit;
*Finding AIC, BIC and AICC for model comparison ;
proc mixed data = NBASalary method=ML;
  class pos (ref="C");
  model sqrty = rankyn confC ageC pos tmpC rebC asC stC blkC pfC tpC;
  run;
  quit;
proc mixed data = NBASalary method=ML;
  model sqrty = tpC ageC pfC tmpC blkC asC rebC rankyn confC ;
  run;
  quit;
proc mixed data = NBASalary method=ML;
  model sqrty = tpC ageC pfC tmpC blkC asC rebC rankyn ;
  run;
  quit;
*according to the criterions for model comparison sqrt2 should be chosen
*check interactions;
proc glm data = NBASalary;
  model sqrty = tpC|ageC|pfC|tmpC|blkC|asC|rebC|rankyn ;
  run;
  quit;
*new model with possible interactions;
ods graphics on;
proc glm data = NBASalary;
   model sqrty = tpC ageC pfC tmpC blkC asC rebC rankyn tpC*ageC
               tpC*pfC tmpC*ageC tpC*blkC blkC*rankyn ;
run;
quit;
*remove tpC*blkC tpC*ageC and blkC*rankyn because they were insignificant in the model;
ods graphics on;
proc glm data = NBASalary PLOTS=all;
   model sqrty = tpC ageC pfC tmpC blkC asC rebC rankyn
               tpC*pfC tmpC*ageC ;
   output out = outsalary r=residfinal p = predfinal;
run;
quit;
*This runs the Shapiro Wilk Normality Test;
proc univariate data = outsalary normal;
   var residfinal ;
run;
*Runing Leveans Test for Constant Variance ;
data  outsalary2 ;
set  outsalary ;
aresid = abs(residfinal);
run;
proc glm data = outsalary2;
   class pos (ref="C");
   model aresid = rank1920 conf age1819 pos tmp1819 reb1819
               as1819 st1819 blk1819 to1819 pf1819 tp1819;
run;
quit;