Linear Models Project: Modelling Hockey Win Percentage

Gina O'Riordan - East Group

May 4, 2018

Introduction

This project aims to predict the win percentage of a hockey team based on a set of predictors. The hockey data being used has 1519 observations in 27 variables but only 13 variables will be included in the model. The response variable win percentage ('winpert') will be created in the pre-work step. The variables are below, listed first by abbreviation and then by the full variable name after the dash:

- lgID - League ID $
- tmID - Team ID $
- franchID - Franchise ID $
- confID - Conference ID $
- divID - Division ID $
- rank - Final standing
- playoff - Playoff result $
- G - Games
- W - Wins
- L - Losses
- T - Ties
- OTL - Overtime losses
- Pts - Points
- SoW - Shootout wins
- SoL - Shootout losses
- GF - Goals for
- GA - Goals against
- name - Full team name $
- PIM - Penalty minutes
- BenchMinor - Bench minors (minutes)
- PPG - Power play goals
- PPC - Power play chances
- SHA - Shorthanded goals against
- PKG - Power play goals against
- PKC - Penalty kill chances
- SHF - Shorthanded goals for

The $ at the end of the variable full name (lgID, tmID, franchID, confID, divID, playoff, & name) signifies that since these data variables are not utilized in the model, the various abbreviations used in the data will not be described.

Pre-Work: Load the Data & Create Win Percentage Variable

Before starting data analysis, we will initialize the environment in R Studio, load the initial tidyverse library needed for the exploratory data analysis graphs, and set the working directory for the data files.

```
#initialize the environment
rm(list=ls())
```

```r
#load libraries
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.4.3
```

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

```
## v ggplot2 2.2.1     v purrr   0.2.4
## v tibble  1.4.2     v dplyr   0.7.4
## v tidyr   0.8.0     v stringr 1.2.0
## v readr   1.1.1     v forcats 0.2.0
```

```
## Warning: package 'ggplot2' was built under R version 3.4.3
```

```
## Warning: package 'tibble' was built under R version 3.4.3
```

```
## Warning: package 'tidyr' was built under R version 3.4.3
```

```
## Warning: package 'readr' was built under R version 3.4.3
```

```
## Warning: package 'purrr' was built under R version 3.4.4
```

```
## Warning: package 'dplyr' was built under R version 3.4.3
```

```
## Warning: package 'stringr' was built under R version 3.4.3
```

```
## Warning: package 'forcats' was built under R version 3.4.3
```

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

```r
#working directory set
setwd("~/DataScience/Linear Models/Project")
```

Then the data is imported from the Hockey.csv file (downloaded from Nexus) and viewed to ensure there were no import errors.

```r
#Download the data
hockey <- read.csv("Hockey.csv", header = T)
glimpse(hockey)
```

```
## Observations: 1,519
## Variables: 27
## $ year       <int> 1909, 1909, 1909, 1909, 1909, 1909, 1909, 1910, 191...
## $ lgID       <fct> NHA, NHA, NHA, NHA, NHA, NHA, NHA, NHA, NHA, NHA, N...
## $ tmID       <fct> COB, HAI, LES, MOS, MOW, OT1, REN, MOC, MOW, OT1, Q...
## $ franchID   <fct> BKN, MTL, TBS, MOS, MTW, STE, REN, MTL, MTW, STE, B...
## $ confID     <fct> , , , , , , , , , , , , , , , , , , , , , , , , , ,
## $ divID      <fct> , , , , , , , , , , , , , , , , , , , , , , , , , ,
## $ rank       <int> 4, 5, 7, 6, 1, 2, 3, 2, 4, 1, 5, 3, 4, 3, 2, 1, 1, ...
## $ playoff    <fct> , , , , , , , , , , , , , , , , , , , , , , , LCS, ,
## $ G          <int> 12, 12, 12, 12, 12, 12, 12, 16, 16, 16, 16, 16, 18,...
## $ W          <int> 4, 4, 2, 3, 11, 9, 8, 8, 7, 13, 4, 8, 8, 9, 9, 10, ...
## $ L          <int> 8, 8, 10, 8, 1, 3, 3, 8, 9, 3, 12, 8, 10, 9, 9, 8, ...
## $ T          <int> 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ OTL        <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
## $ Pts        <int> 8, 8, 4, 7, 22, 18, 17, 16, 14, 26, 8, 16, 16, 18, ...
## $ SoW        <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
## $ SoL        <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
## $ GF         <int> 79, 77, 59, 52, 91, 89, 96, 66, 73, 122, 65, 91, 59...
```

```
## $ GA         <int> 104, 83, 100, 95, 41, 66, 54, 62, 88, 69, 97, 101, ...
## $ name       <fct> Cobalt Silver Kings, Haileybury Hockey Club, Les Ca...
## $ PIM        <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
## $ BenchMinor <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
## $ PPG        <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
## $ PPC        <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
## $ SHA        <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
## $ PKG        <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
## $ PKC        <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
## $ SHF        <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
```

Note that the ties variable 'T' is confusing for R as the same capitalized letter 'T' is used to signify True in function calculations. Therefore, we rename the variable to be 'Ties' to avoid confusion moving forward.

```
#rename the ties (T) variable to 'Ties'
names(hockey)[12] <- "Ties"
```

The final step before data analysis is to create the Win Percentage or 'winpert' variable by dividing the number of wins (w) by the number of games played (G). This will be the response variable in the models created in subsequent steps; we inspect it quickly before moving to step 1. We find that the values lie between 0 and 1, as expected for a percentage calculation, and there are no missing/NA values.

```
#establish win percentage variable
hockey$winpert <- hockey$W/hockey$G
#review new variable
summary(hockey$winpert)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.3654  0.4512  0.4452  0.5250  0.9167
```

```
sum(is.na(hockey$year))
```

```
## [1] 0
```

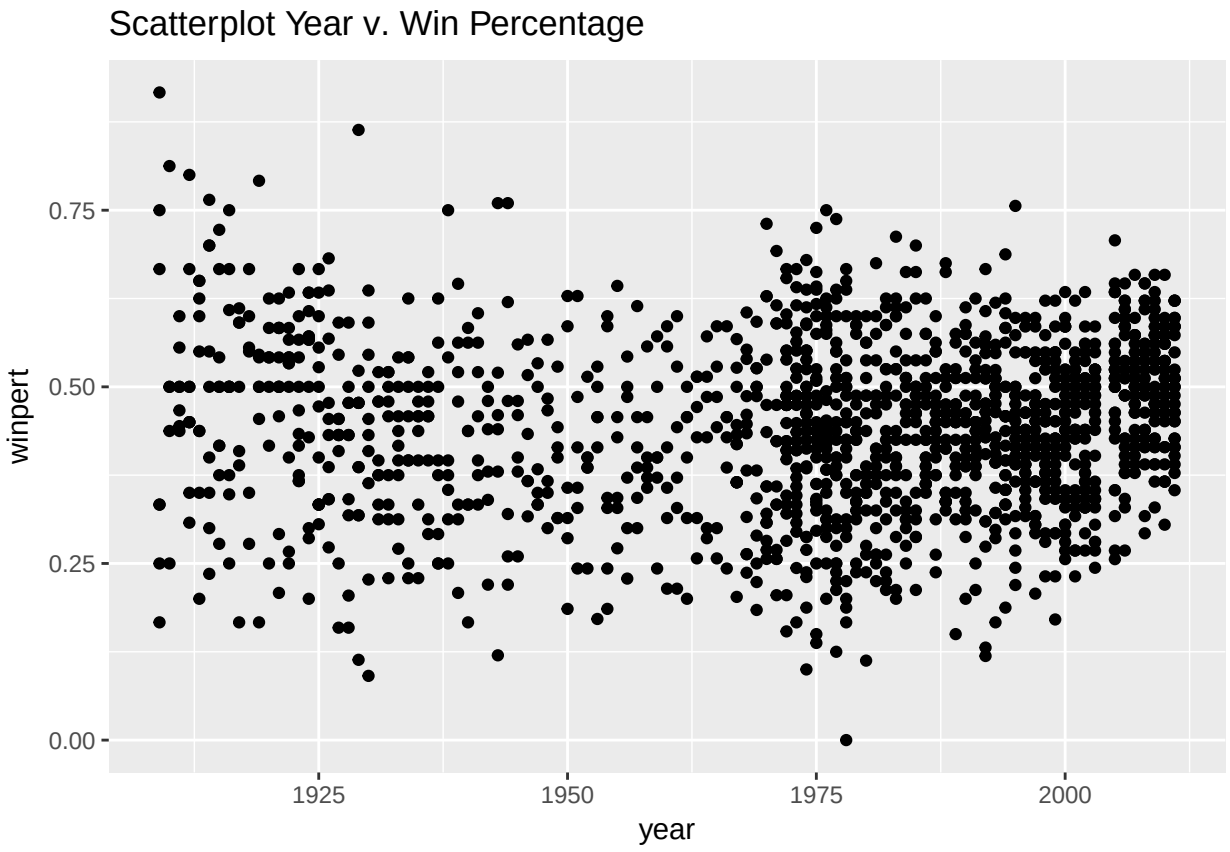
Step 1: Exploratory Data Analysis (EDA)

Before creating linear models, each variable must be inspected for outliers, missing data, and any unusual patterns to be inspected. I reviewed each variable's summary, number of missing values, and unique values (or levels for factors). Then I created a histogram for each and scatter plot with the 'win percentage' variable for the y-axis. Since there were a large number of predictors in this data set, only four variables are highlighted in this EDA. The 'year,' 'rank,' 'L' (losses), 'Pts' variables have been plotted in scatter plots, with the 'winpert' (win percentage) response variable on the y-axis.

Year

The year variable ranges from 1909-2011 with a majority of data points after 1970; the win percentage range seems to shrink as the years progress towards 2011 as more teams joined the league overtime.

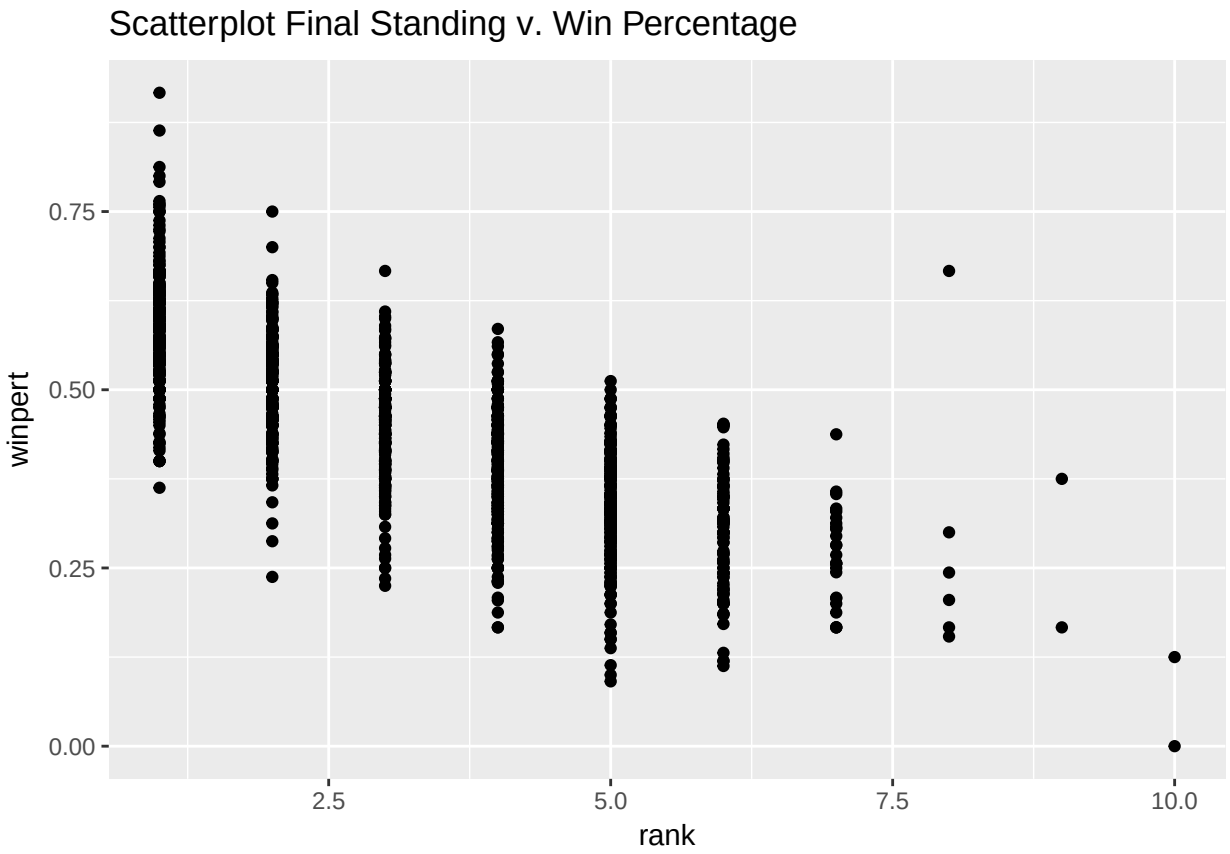
```
scplot_year <- ggplot(data = hockey, aes(x = year, y = winpert)) +
  geom_point() + labs(title = "Scatterplot Year v. Win Percentage")
scplot_year
```



Final Standing (Rank)

The rank variable is an integer variable with a range 1-10, with vertical lines at each integer since a rank is a round number; lower ranks mean a better team (i.e. number 1 rank is the best) and thus higher win percentage.

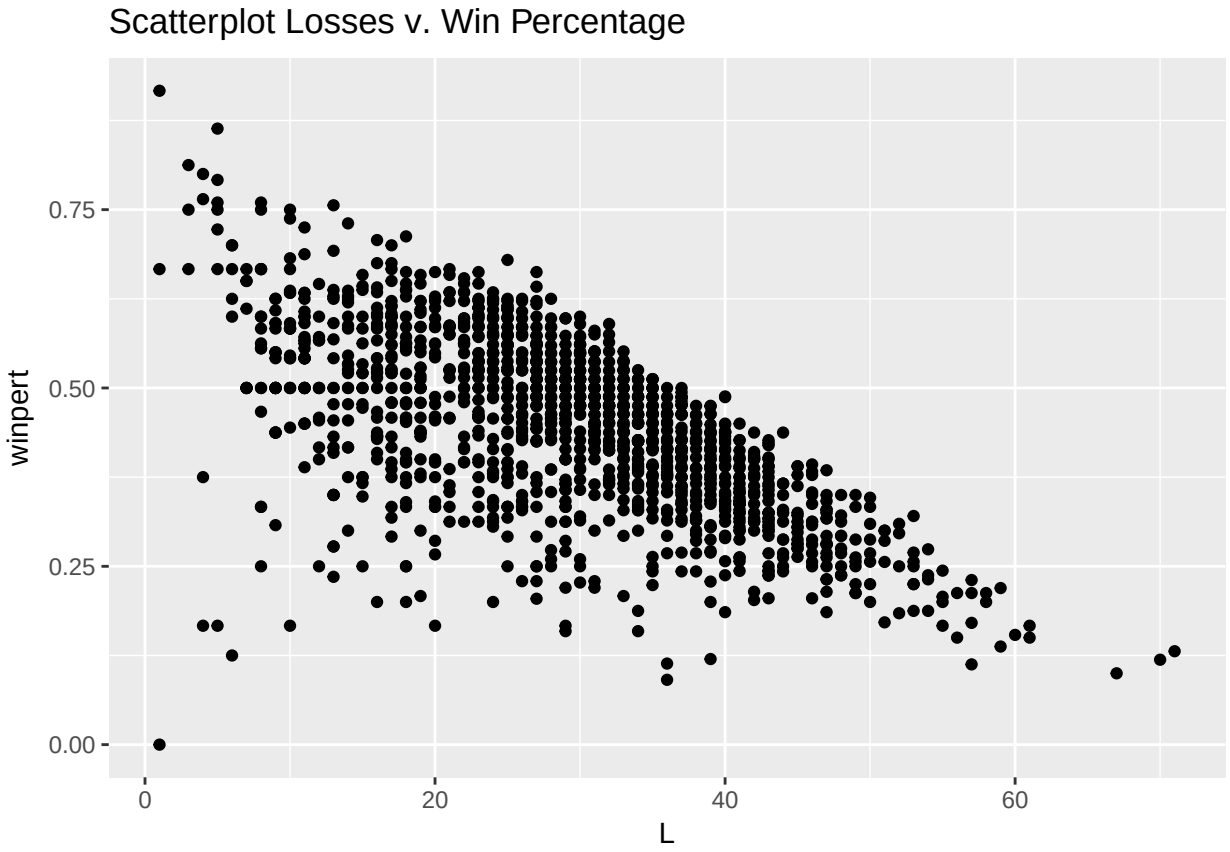
```
scplot_rank <- ggplot(data = hockey, aes(x = rank, y = winpert)) +
  geom_point() + labs(title = "Scatterplot Final Standing v. Win Percentage")
scplot_rank
```



Losses

The losses variable is an integer variable ranging from 1 to 71; as expected, the scatter plot shows a negative linear relationship to the amount of games lost and the win percentage.

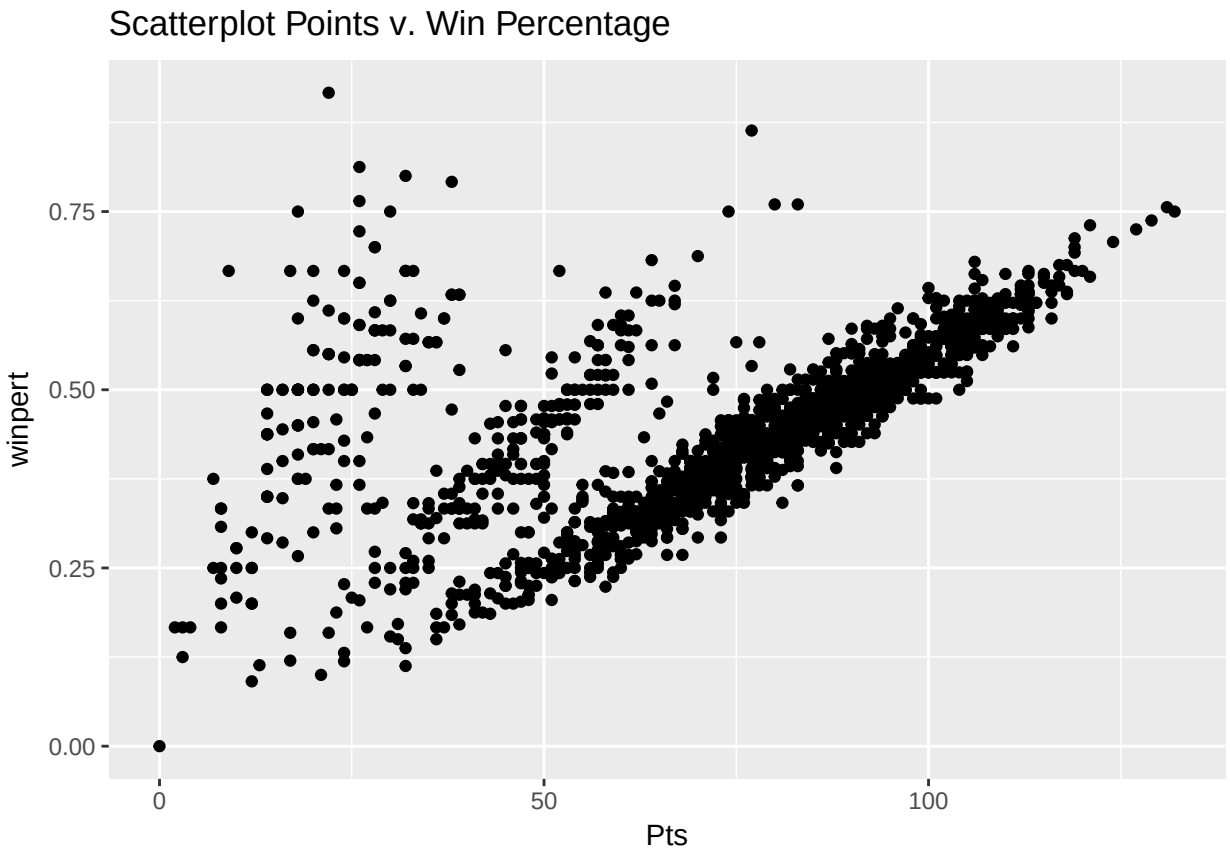
```
scplot_losses <- ggplot(data = hockey, aes(x = L, y = winpert)) +
  geom_point() + labs(title = "Scatterplot Losses v. Win Percentage")
scplot_losses
```



Points

The points variable is an integer variable ranging from 0 to 132; as expected, the scatter plot shows a positive linear relationship to the amount of points and the win percentage as the team with the most points wins the game.

```
scplot_points <- ggplot(data = hockey, aes(x = Pts, y = winpert)) +
  geom_point() + labs(title = "Scatterplot Points v. Win Percentage")
scplot_points
```



Step 2: Perform Model Selction Via a Testing-Based Approach

Now that the data has been analyzed, an initial linear model needs to be formed. The following variables were not included because they were a name/identifier variable for the hockey teams: lgID, tmID, franchID, confID, divID, playoff, & name. Then certain wins/losses variables caused errors in the model and were also removed: G, W, L, Ties, OTL, SoW, SoL. Overall, the linear model's response was winpert and the predictors were the remaining variables in the hockey data set. The model is named mod_full to differentiate it from models created in later steps.

```
mod_full <- lm(winpert~year+rank+Pts+GF+GA+PIM+BenchMinor+PPG+PPC+SHA+PKG+PKC+SHF, data=hockey)
summary(mod_full)
```

```
##
## Call:
## lm(formula = winpert ~ year + rank + Pts + GF + GA + PIM + BenchMinor +
##     PPG + PPC + SHA + PKG + PKC + SHF, data = hockey)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.104422 -0.022182 -0.001564  0.019737  0.181762
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.681e+00  2.674e-01  -6.287 4.80e-10 ***
## year         9.888e-04  1.358e-04   7.284 6.54e-13 ***
## rank        -4.375e-03  1.059e-03  -4.130 3.92e-05 ***
```

```
## Pts          3.134e-03  1.750e-04  17.910  < 2e-16 ***
## GF           6.715e-04  7.218e-05   9.304  < 2e-16 ***
## GA          -8.624e-04  5.453e-05 -15.815  < 2e-16 ***
## PIM         -1.770e-06  4.667e-06  -0.379 0.704661
## BenchMinor  -3.735e-04  1.513e-04  -2.468 0.013740 *
## PPG          7.179e-04  1.399e-04   5.133 3.42e-07 ***
## PPC         -2.182e-04  3.638e-05  -5.998 2.78e-09 ***
## SHA          1.256e-03  3.421e-04   3.673 0.000253 ***
## PKG          5.214e-04  1.492e-04   3.494 0.000497 ***
## PKC         -1.584e-04  4.055e-05  -3.906 0.000100 ***
## SHF          9.371e-04  3.039e-04   3.084 0.002098 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03402 on 1010 degrees of freedom
##   (495 observations deleted due to missingness)
## Multiple R-squared:  0.9095, Adjusted R-squared:  0.9083
## F-statistic: 780.4 on 13 and 1010 DF,  p-value: < 2.2e-16
```

The model shows that all but one of the predictors (PIM) are significant in the model, and the R-squared value is 0.9095.

For step 2, we are asked to apply the fast backward variable selection (fastbw()) function to the data. The rms library is required and loaded first. Then the full model needs to be created using the ols function instead of the lm for the fastbw() function to work properly. The final step is to run fastbw() with arguments of 'p' for p-values and sls, or significance for staying in the model value, of alpha = 0.05.

```
#required library needed first
require(rms)
```

```
## Loading required package: rms

## Warning: package 'rms' was built under R version 3.4.3

## Loading required package: Hmisc

## Warning: package 'Hmisc' was built under R version 3.4.3

## Loading required package: lattice

## Loading required package: survival

## Loading required package: Formula

##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:dplyr':
##
##     src, summarize

## The following objects are masked from 'package:base':
##
##     format.pval, units

## Loading required package: SparseM

##
## Attaching package: 'SparseM'

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

```
##     backsolve
```

```
#create full model using ols
ols_mod <- ols(winpert~year+rank+Pts+GF+GA+PIM+BenchMinor+PPG+PPC+
                SHA+PKG+PKC+SHF, data=hockey)
#run fastb2() with alpha sls = 0.05
fastbw(ols_mod, rule="p", sls=0.05)
```

```
##
##  Deleted Chi-Sq d.f. P      Residual d.f. P      AIC   R2
##  PIM     0.14   1    0.7046 0.14     1    0.7046 -1.86 0.909
##
## Approximate Estimates after Deleting Factors
##
##                  Coef      S.E.  Wald Z         P
## Intercept  -1.6866867 2.671e-01  -6.315 2.700e-10
## year        0.0009911 1.356e-04   7.307 2.724e-13
## rank       -0.0043440 1.056e-03  -4.113 3.900e-05
## Pts         0.0031555 1.656e-04  19.056 0.000e+00
## GF          0.0006638 6.927e-05   9.583 0.000e+00
## GA         -0.0008583 5.349e-05 -16.048 0.000e+00
## BenchMinor -0.0003796 1.505e-04  -2.523 1.165e-02
## PPG         0.0007176 1.399e-04   5.131 2.881e-07
## PPC        -0.0002182 3.637e-05  -5.998 2.001e-09
## SHA         0.0012395 3.392e-04   3.655 2.577e-04
## PKG         0.0005182 1.490e-04   3.478 5.055e-04
## PKC        -0.0001646 3.717e-05  -4.427 9.554e-06
## SHF         0.0009282 3.030e-04   3.064 2.186e-03
##
## Factors in Final Model
##
##  [1] year       rank       Pts        GF         GA         BenchMinor
##  [7] PPG        PPC        SHA        PKG        PKC        SHF
```

The output of fastbw() suggests removing the PIM variable from the model. To obtain the adjusted R-squared for this modified model, we create the lm of the function without PIM as a predictor. This model is called mod_1.

```
mod_1 <- lm(winpert~year+rank+Pts+GF+GA+BenchMinor+PPG+PPC+
              SHA+PKG+PKC+SHF, data=hockey)
summary(mod_1)
```

```
##
## Call:
## lm(formula = winpert ~ year + rank + Pts + GF + GA + BenchMinor +
##     PPG + PPC + SHA + PKG + PKC + SHF, data = hockey)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.103852 -0.022080 -0.001546  0.019737  0.181778
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.687e+00  2.670e-01  -6.318 3.97e-10 ***
## year         9.911e-04  1.356e-04   7.310 5.40e-13 ***
## rank        -4.344e-03  1.056e-03  -4.115 4.19e-05 ***
```

```
## Pts          3.156e-03  1.655e-04  19.064  < 2e-16 ***
## GF           6.638e-04  6.925e-05   9.587  < 2e-16 ***
## GA          -8.583e-04  5.346e-05 -16.055  < 2e-16 ***
## BenchMinor  -3.796e-04  1.504e-04  -2.524 0.011764 *
## PPG          7.176e-04  1.398e-04   5.133 3.41e-07 ***
## PPC         -2.182e-04  3.636e-05  -6.000 2.74e-09 ***
## SHA          1.240e-03  3.390e-04   3.656 0.000269 ***
## PKG          5.182e-04  1.489e-04   3.479 0.000524 ***
## PKC         -1.645e-04  3.715e-05  -4.429 1.05e-05 ***
## SHF          9.282e-04  3.028e-04   3.065 0.002234 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03401 on 1011 degrees of freedom
##   (495 observations deleted due to missingness)
## Multiple R-squared:  0.9094, Adjusted R-squared:  0.9084
## F-statistic: 846.2 on 12 and 1011 DF,  p-value: < 2.2e-16
```

The R-squared value for mod_1 is 0.9094. Since this value is only 0.0001 lower than the original model, this proves that removing the PIM variable does not change the prediction power of the model. Thus the mod_1 is a slightly simpler model with less predictors and still a strong model for predicting win percentage. This step matches my intuition that Penalty Minutes (PIM) would not be a strong predictor of win percentage. Teams that have more penalty minutes have fewer players on the ice during a game; this could cause them to lose the game due to power play goals scored by their opponents, but not necessarily. It is a weak correlation to win percentage at best. Also based on the exploratory data analysis scatter plot, there was seemingly no correlation between win percentage and penalty minutes.

Step 3: Perform Model Selection Via a Criterion-Based Approach

The Akaike Information Criterion (AIC) balances model fit or predictive power with model simplicity. To attain the AIC, we use the extractAIC() function on the original or full model.

```
extractAIC(mod_full)
```

```
## [1]    14.00 -6910.02
```

Per the output of the function, we see there are 13 predictors plus 1 intercept in the full model (totaling 14) and the AIC is -6910.02. However, this value is only relevant when comparing it to another model's AIC and is impossible to evaluate otherwise. The stepAIC() function gives us that required reference point; it uses an iterative process to calculate AIC's for various models, starting with the full model. Each subsequent model has removed the predictor that reduces the AIC the most, a quality found in the 'AIC' column in the output. Note that to run the stepAIC() function, we must first require the MASS library.

```
require(MASS)
```

```
## Loading required package: MASS
```

```
## Warning: package 'MASS' was built under R version 3.4.4
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
mod_step <- stepAIC(mod_full)
```

```
## Start:  AIC=-6910.02
## winpert ~ year + rank + Pts + GF + GA + PIM + BenchMinor + PPG +
##     PPC + SHA + PKG + PKC + SHF
##
##              Df Sum of Sq    RSS     AIC
## - PIM         1   0.00017 1.1691 -6911.9
## <none>                    1.1689 -6910.0
## - BenchMinor  1   0.00705 1.1760 -6905.9
## - SHF         1   0.01101 1.1799 -6902.4
## - PKG         1   0.01413 1.1830 -6899.7
## - SHA         1   0.01561 1.1845 -6898.4
## - PKC         1   0.01766 1.1866 -6896.7
## - rank        1   0.01974 1.1886 -6894.9
## - PPG         1   0.03049 1.1994 -6885.6
## - PPC         1   0.04164 1.2105 -6876.2
## - year        1   0.06140 1.2303 -6859.6
## - GF          1   0.10018 1.2691 -6827.8
## - GA          1   0.28947 1.4584 -6685.4
## - Pts         1   0.37125 1.5401 -6629.6
##
## Step:  AIC=-6911.87
## winpert ~ year + rank + Pts + GF + GA + BenchMinor + PPG + PPC +
##     SHA + PKG + PKC + SHF
##
##              Df Sum of Sq    RSS     AIC
## <none>                    1.1691 -6911.9
## - BenchMinor  1   0.00736 1.1764 -6907.4
## - SHF         1   0.01086 1.1799 -6904.4
## - PKG         1   0.01400 1.1831 -6901.7
## - SHA         1   0.01546 1.1845 -6900.4
## - rank        1   0.01958 1.1886 -6896.9
## - PKC         1   0.02268 1.1918 -6894.2
## - PPG         1   0.03047 1.1995 -6887.5
## - PPC         1   0.04163 1.2107 -6878.0
## - year        1   0.06180 1.2309 -6861.1
## - GF          1   0.10627 1.2753 -6824.8
## - GA          1   0.29805 1.4671 -6681.3
## - Pts         1   0.42027 1.5893 -6599.4
```

```
mod_step$anova
```

```
## Stepwise Model Path
## Analysis of Deviance Table
##
## Initial Model:
## winpert ~ year + rank + Pts + GF + GA + PIM + BenchMinor + PPG +
##     PPC + SHA + PKG + PKC + SHF
##
## Final Model:
## winpert ~ year + rank + Pts + GF + GA + BenchMinor + PPG + PPC +
##     SHA + PKG + PKC + SHF
##
##
##    Step Df     Deviance Resid. Df Resid. Dev       AIC
## 1                           1010   1.168899 -6910.020
```

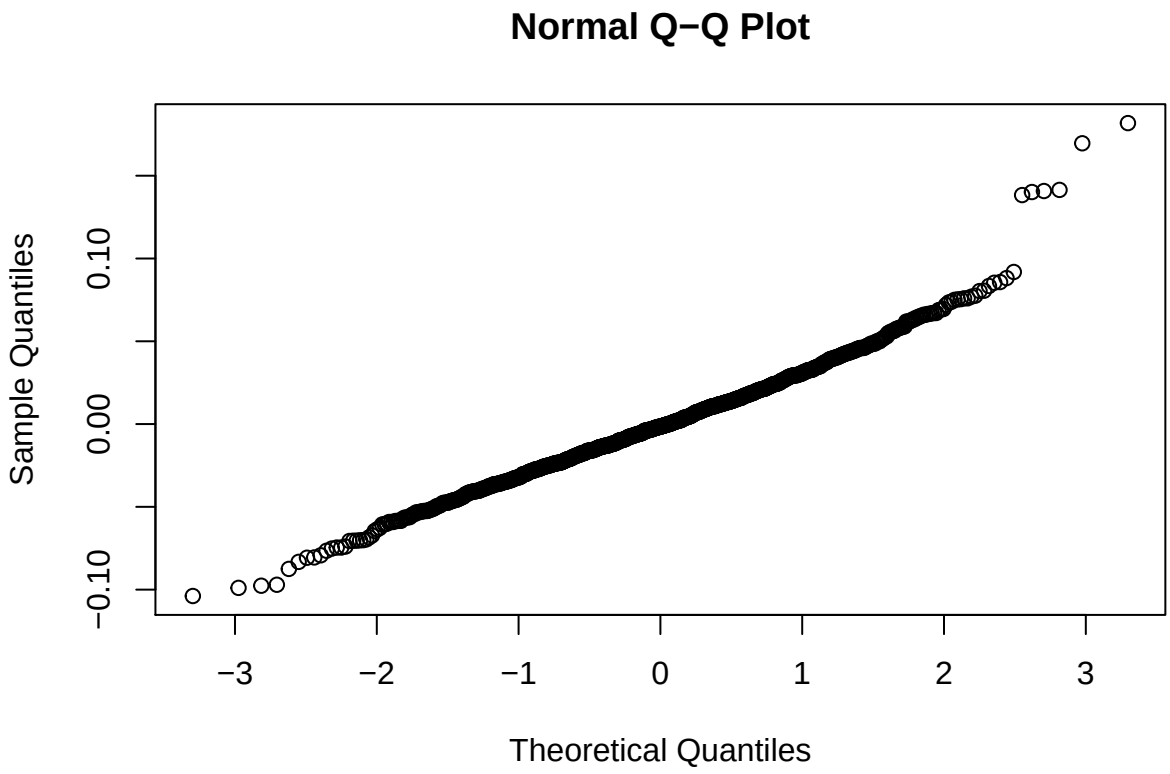
```
## 2 - PIM  1 0.0001663647      1011   1.169065 -6911.874
```

The output of the final line of code notes that the AIC is only significantly improved by removing the PIM variable from the model. This result matches the testing-based procedure from Step 2, and thus we will proceed to the next step with the mod_1 (equal to the 'final model' seen in the code output above).

Step 4: Apply Diagnostics to Model

Now that the model has been created, we will perform diagnostics on it to check it's adherence to the mathematical assumptions: normally distributed, constant variance (homoscedasticity), mean of zero, independent, and a linear association between x and y. The first tests will check that the model is normally distributed by viewing a Q-Q plot of the residuals.

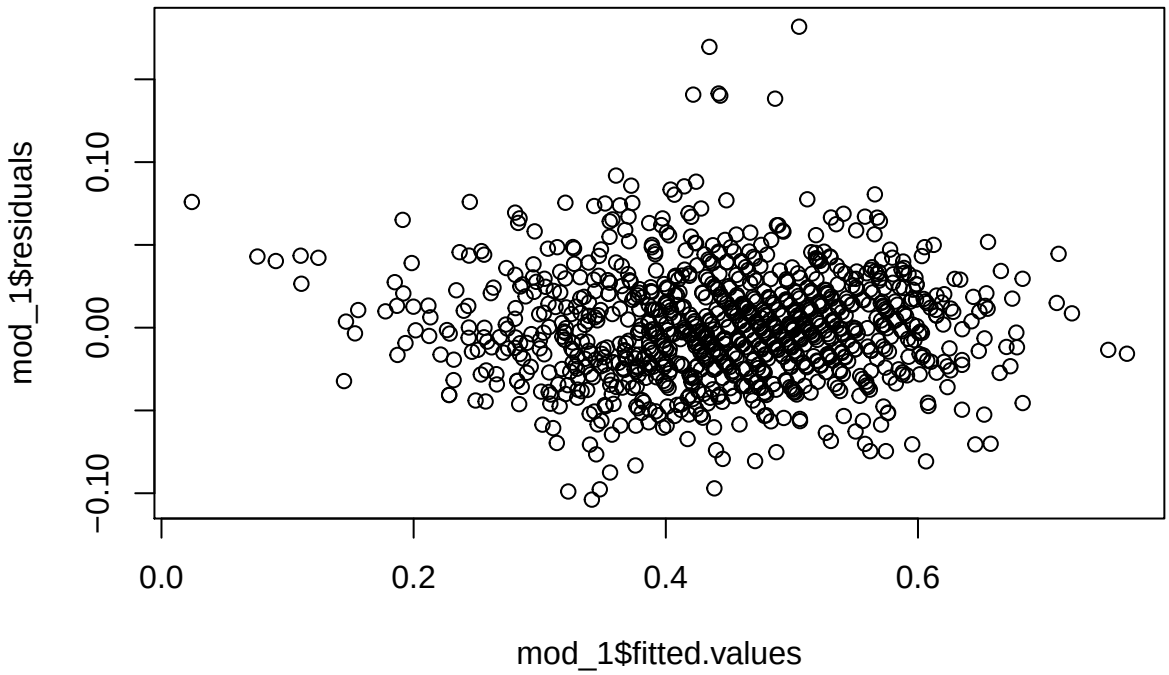
```
qqnorm(mod_1$residuals)
```



The straight line, except for a few points at each end of the data causing a slight curve, demonstrates that the model assumption of normal distribution is upheld.

Constant variance, or homoscedasticity, can be determined by plotting fitted values vs residuals in the model.

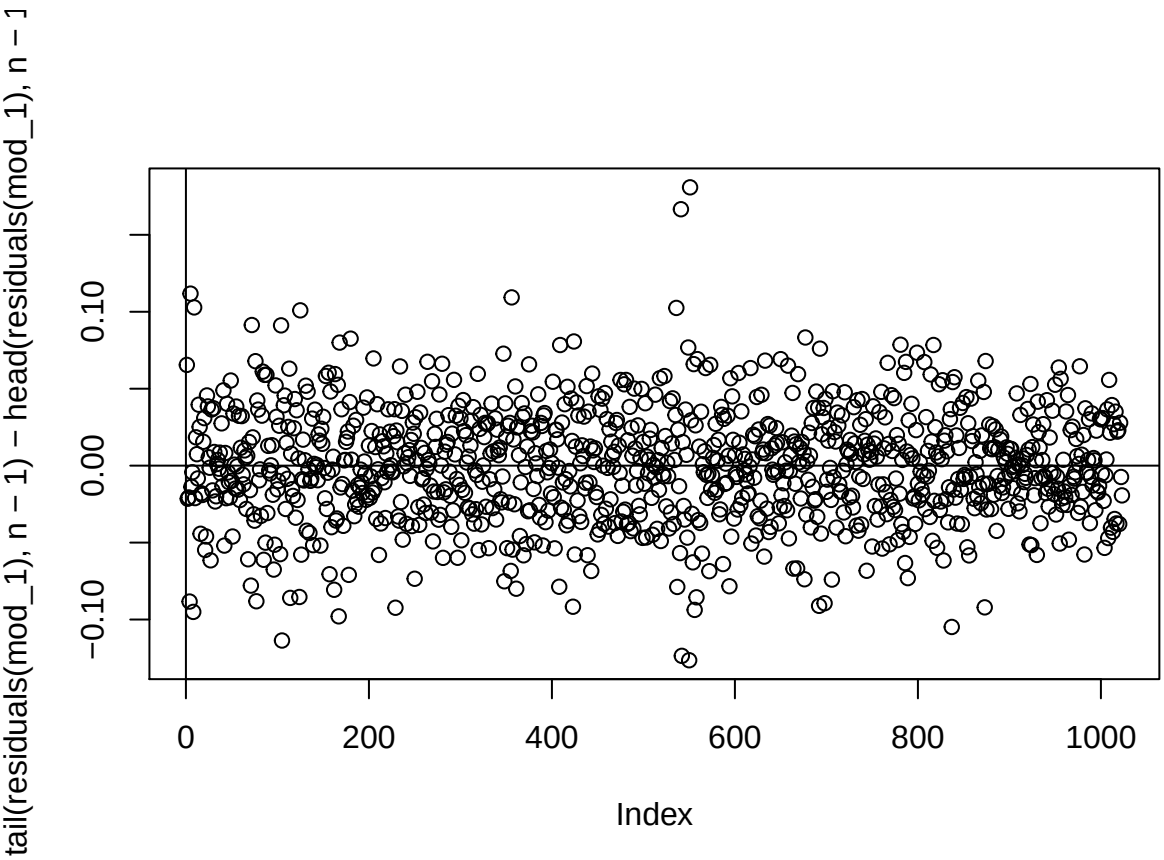
```
plot(mod_1$fitted.values, mod_1$residuals)
```



There is a fairly random pattern, and thus the model assumption of constant error variance is upheld.

To determine if the model errors are correlated or not, we make a lagged residual plot. To do this, once the plot is made, we draw a vertical line at 0 and look at the data points around that line.

```
n <- length(residuals(mod_1))
plot(tail(residuals(mod_1), n-1) ~ head(residuals(mod_1), n-1))
abline(h=0, v=0)
```



Because there is a random scatter of points above and blow the line, the model errors are uncorrelated and the assumption of independence is upheld.

Step 5: Investigate Fit for Individual Observations

The model has been simplified, has strong predictive power, and upholds the mathematical assumptions. The next step is to evaluate individual observations to determine if there are outliers or influential observations among them that might disturb the model.

Outliers can be determined by calculating the standard residuals of the model and determining which of those, as absolute values, are greater than 3. This is the rule of thumb for determining outliers in a model.

```
rstd <-rstandard(mod_1)
length(which(abs(rstd)>3))
```

```
## [1] 7
```

There are 7 outlier values in the data. Next we need to determine if any values are influential, or skewing the model's predictive power. This can be done by determine the Cook's distances for the model then comparing that to a threshold value at the 50th percentile of the F distribution with (p+1) and n-(p+1) as the numerator and denominator, respectively.

```
#Cook's distance
cook <- cooks.distance(mod_1)

#Threshold for Cook's distance, 50th percentile of F distribution with (p+1) and n-(p+1) numerator and
n <- dim(model.matrix(mod_1))[1]
```

```
p <- dim(model.matrix(mod_1))[2]
num.df <- p
den.df <- n-p
F.thresh <- qf(0.5,num.df,den.df)
F.thresh
```

```
## [1] 0.9498412
```

```
#compare Cook's distance to the F threshold to determine influential observations
which(cook>F.thresh)
```

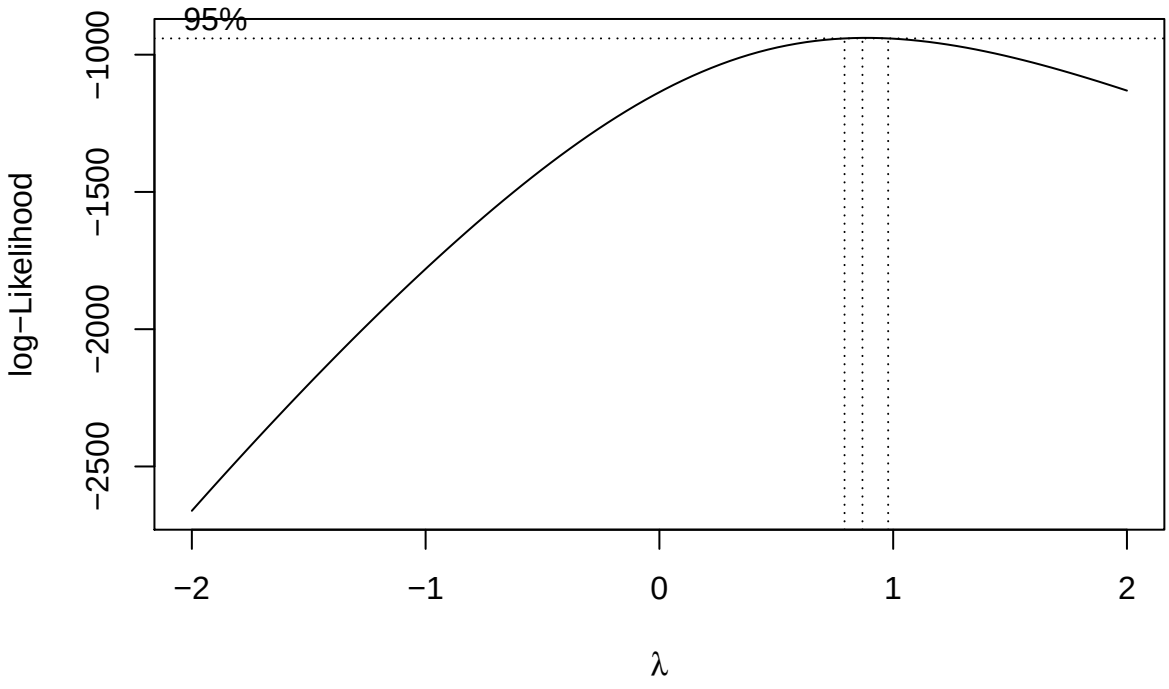
```
## named integer(0)
```

The code output states that there are no influential observations. Therefore, no model transformation is needed.

Step 6: Apply Transformations to Model As Needed

Since a transformation is not needed, we run the Box-Cox method anyway to see if the result agrees with the assessment from step 4 above. The MASS library is required to then run the boxcox() function on mod_1.

```
library(MASS)
bc <- boxcox(mod_1)
```



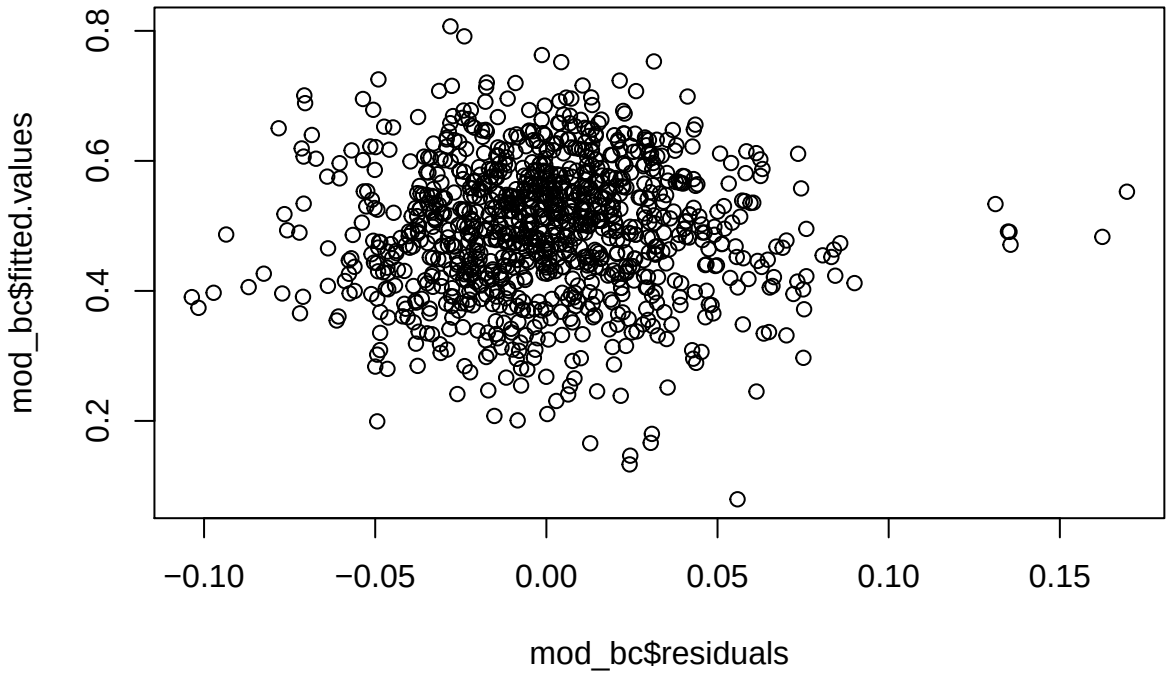
The graph that demonstrates the optimal value of lambda for a transformation is just under 1, about 0.9. Note that since zero is not included in the 95% confidence lines around lambda, log(y) is not a plausible transformation of y. Instead, we will test the transformation model made by raising the response variable to the power of lambda.

```
lambda <- bc$x[which.max(bc$y)]
lambda
```

```
## [1] 0.8686869
```

```
mod_bc <- lm(winpert^lambda~year+rank+Pts+GF+GA+BenchMinor+PPG+PPC+
               SHA+PKG+PKC+SHF, data=hockey)

plot(mod_bc$residuals, mod_bc$fitted.values)
```



The residual vs. fitted values plot looks the same as the non-transformed model thus the transformation is not needed; the results are in agreement with step 4.

Step 7: Report Inferences and Make Predictions Using the Final Model

The final parameter estimates and p-values for the final model are as follows:

Predictor	Parameter Estimate	P-Value
(Intercept)	-1.687e+00	3.97e-10
Year	9.911e-04	5.40e-13
Rank	-4.344e-03	4.19e-05
Pts	3.156e-03	< 2e-16
GF	6.638e-04	< 2e-16
GA	-8.583e-04	< 2e-16
BenchMinor	-3.796e-04	0.011764
PPG	7.176e-04	3.41e-07

Predictor	Parameter Estimate	P-Value
PPC	-2.182e-04	2.74e-09
SHA	1.240e-03	0.000269
PKG	5.182e-04	0.000524
PKC	-1.645e-04	1.05e-05
SHF	9.282e-04	0.002234

The R-squared for the model is 0.9094.

I feel the most important predictor is the 'Pts' variable; below is a computation of a 95% confidence interval for the slope of all variables, demonstrating the 'Pts' interval is 0.0028307321 and 3.480338e-03.

```
confint(mod_1)
```

```
##                     2.5 %        97.5 %
## (Intercept) -2.2105765223 -1.162797e+00
## year         0.0007250644  1.257138e-03
## rank        -0.0064154952 -2.272504e-03
## Pts          0.0028307321  3.480338e-03
## GF           0.0005279494  7.997113e-04
## GA          -0.0009632448 -7.534222e-04
## BenchMinor  -0.0006747378 -8.444100e-05
## PPG          0.0004433057  9.919801e-04
## PPC         -0.0002895133 -1.468175e-04
## SHA          0.0005742468  1.904845e-03
## PKG          0.0002259230  8.104208e-04
## PKC         -0.0002374582 -9.164353e-05
## SHF          0.0003339264  1.522423e-03
```

The 95% confidence interval for a prediction using the median of the predictors is found below. This is done by computing a 95% confidence value for the predicted value of win percentage at the median values. First a model matrix of the model must be made, then medians taken into account before the confidence interval can be calculated.

```
X <- model.matrix(mod_1)
(x0 <- apply(X,2,median))
```

```
## (Intercept)        year        rank         Pts          GF          GA
##         1.0      1993.0         3.0        84.0       251.0       251.0
##  BenchMinor         PPG         PPC         SHA         PKG         PKC
##        16.0        63.0       337.5         8.0        63.0       336.0
##         SHF
##         8.0
```

```
predict(mod_1, new=data.frame(t(x0)), interval='confidence')
```

```
##         fit       lwr       upr
## 1 0.4519957 0.4497676 0.4542238
```

The confidence interval is 0.4497676 and 0.4542238. To create the 95% prediction interval for the predicted value of win percentage at the median values, we use the pre-work done in the code above then change the predict() function's interval argument to 'prediction.'

```
predict(mod_1, new=data.frame(t(x0)), interval='prediction')
```

```
##         fit       lwr       upr
## 1 0.4519957 0.3852299 0.5187615
```

This interval is 0.3852299 and 0.5187615.

Conclusion

Through the steps above, I feel confident that the win percentage variable is satisfactorily described by the mod_1 model. One way the model could have been improved was by adding data regarding goalie and other position statistics (defensive positions v. offensive, etc.). This would give a very interesting view of how different player roles play a part in win percentage. Also I would be curious if the win percentage would be effected by the percentage of the team's players who came from college hockey teams versus those who don't. Traditionally, players recruited at younger ages have stronger talent and skill, but this may not affect the team's win percentage as hockey is a team sport, not an individual one.