Project 4

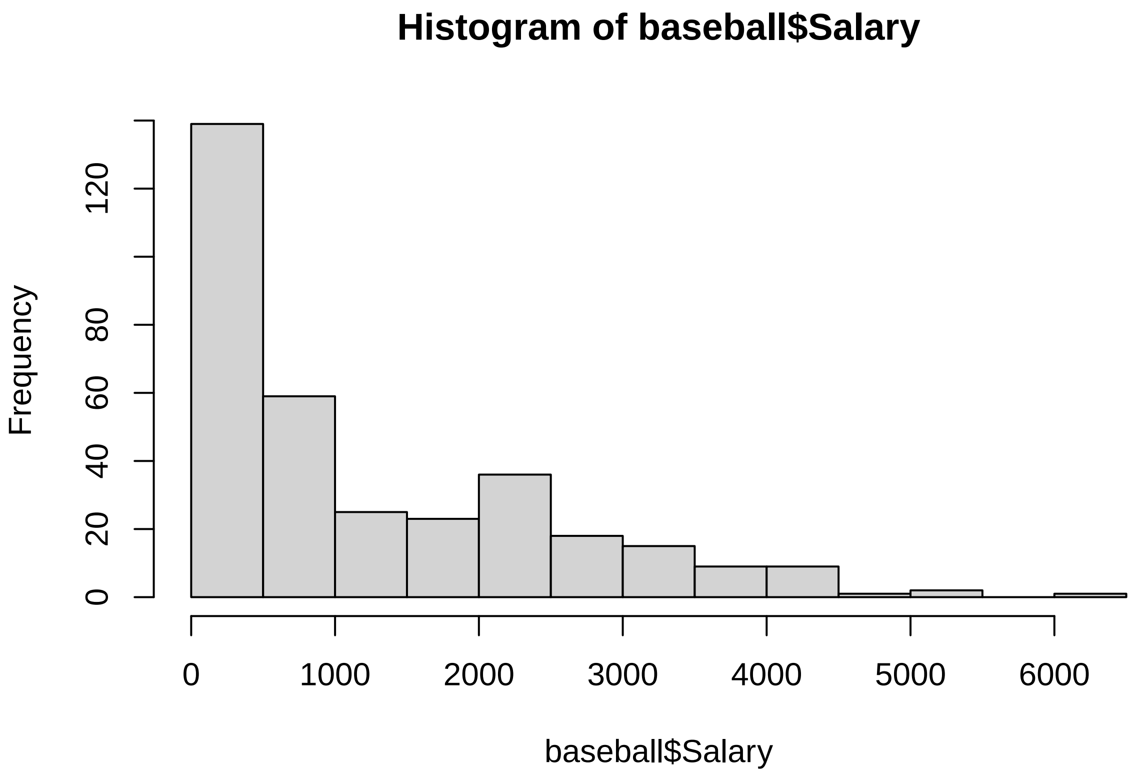
2022-11-02

1. Exploratory Data Analysis

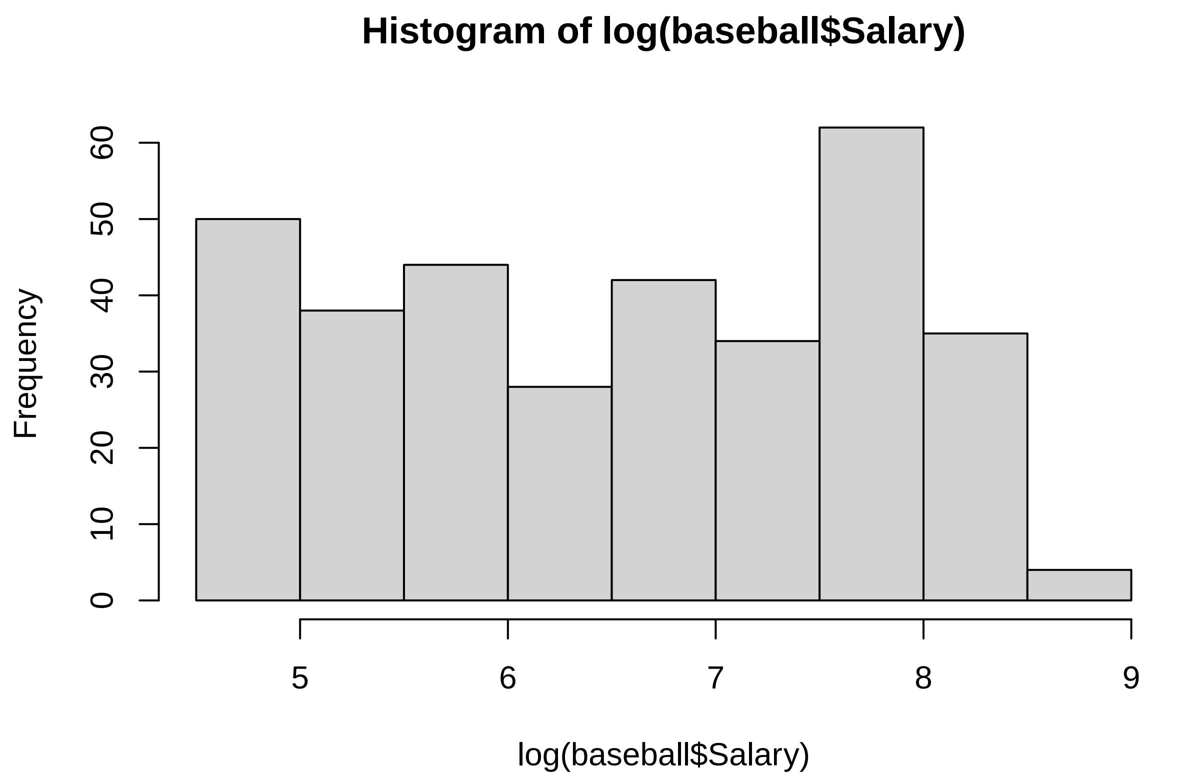
```r
baseball = read.csv("baseball.dat.txt")
```

(a)

```r
hist(baseball$Salary)
```



```r
hist(log(baseball$Salary))
```



```
baseball$log.Salary=log(baseball$Salary)
```

The first plot shows the untransformed data are heavily skewed to the left. Applying the logarithm transform (the result is shown in the second plot) helps alleviate some of this skew and makes the data more amenable to model development and analysis.

(b)

```
anyNA(baseball)
```

```
## [1] FALSE
```

From inspection and the above function, we observe no missing data. The types of the variables are as follows:

1. Salary: continuous (in the technical sense) but represented as an integer for simplicity
2. log.Salary: continuous
3. AVG: continuous
4. OBP: continuous
5. Runs: count
6. Hits: count
7. Doubles: count
8. Triples: count
9. HR: count
10. RBI: count
11. Walks: count

12. SO: count
13. SB: count
14. Errs: count
15. free.agency.eligibility: categorical
16. free.agent.in.1991.2: categorical
17. arbitration.eligibility: categorical
18. arbitration.in.1991.2: categorical
19. Name: categorical

Based on this analysis, there are four continuous variables (including Salary and log.Salary), ten count variables, and five categorical variables.

2. Linear Regression with Variable Selection

(a)

```
sample = sample(c(TRUE, FALSE), nrow(baseball), replace=TRUE, prob=c(2/3,1/3))
train = subset(baseball, select = -c(Salary, Name))[sample, ]
test = subset(baseball, select = -c(Salary, Name))[!sample, ]
```

(b)

The three variable selection methods I will employ are:

1. Best subset selection using the leaps algorithm and Bayesian Information Criterion (BIC)
2. Best subset selection using the genetic algorithm and Akaike Information Criterion (AIC)
3. Least Absolute Shrinkage and Selection Operator (LASSO) regression

The implementations and associated best models are shown below in part (d).

(c)

The key quantities involved in variable selection are all variables except “Name” and “Salary.” I will fit all models so that they attempt to predict “log.Salary” from all remaining variables. For variable selection method 1 I use all default parameters. For variable selection method 2, the important non-default parameters I chose are fitfunc, and method. These represent the fact that I fit the data using a linear model and I used the genetic algorithm to explore the candidate set. For variable selection method 3, the important non-default parameters I chose are nfolds and lambda.min. These represent the number of folds used during k-fold cross validation and the minimum value lambda can take.

(d)

```
library(bestglm)
```

```
## Loading required package: leaps
```

```
formula = log.Salary ~ AVG +  OBP + Runs + Hits +
    Doubles + Triples + HR + RBI + Walks + SO + SB + Errs + free.agency.eligibility + free.agent.in.199

y = train[, all.vars(formula)[1]]
X = as.data.frame(model.matrix(as.formula(formula),train))
```

```r
#
# dat.tmp = as.data.frame(cbind(X, y))

result.leaps_BIC = bestglm(Xy=train, IC="BIC", intercept=TRUE, TopModels=1)$BestModel

result.leaps_BIC
```

```
##
## Call:
## lm(formula = y ~ ., data = data.frame(Xy[, c(bestset[-1], FALSE),
##     drop = FALSE], y = y))
##
## Coefficients:
##             (Intercept)                     Hits                      RBI
##                4.849606                 0.004736                 0.009257
## free.agency.eligibility  arbitration.eligibility
##                1.569874                 1.238033
```

```r
library(glmulti)
```

```
## Loading required package: rJava
```

```r
fit = glmulti(log.Salary ~ . , data = train, fitfunc = lm,
 intercept = TRUE, level = 1, method="g",
 confsetsize=1, plotty = FALSE)
```

```
## Initialization...
## TASK: Genetic algorithm in the candidate set.
## Initialization...
## Algorithm started...
##
## After 10 generations:
## Best model: log.Salary~1+AVG+Hits+Doubles+HR+RBI+Walks+SO+Errs+free.agency.eligibility+free.agent.in
## Crit= 380.986198098245
## Mean crit= 380.986198098245
## Change in best IC: -9619.01380190176 / Change in mean IC: -9619.01380190176
##
## After 20 generations:
## Best model: log.Salary~1+OBP+Runs+Hits+Doubles+HR+Walks+SO+SB+free.agency.eligibility+arbitration.eli
## Crit= 378.095110719389
## Mean crit= 378.095110719389
## Change in best IC: -2.89108737885562 / Change in mean IC: -2.89108737885562
##
## After 30 generations:
## Best model: log.Salary~1+AVG+Runs+Hits+HR+Walks+SO+SB+free.agency.eligibility+free.agent.in.1991.2+a
## Crit= 376.102535546764
## Mean crit= 376.102535546764
## Change in best IC: -1.99257517262538 / Change in mean IC: -1.99257517262538
##
## After 40 generations:
## Best model: log.Salary~1+AVG+Runs+Hits+HR+Walks+SO+SB+free.agency.eligibility+free.agent.in.1991.2+a
## Crit= 376.102535546764
## Mean crit= 376.102535546764
## Change in best IC: 0 / Change in mean IC: 0
##
```

```
## After 50 generations:
## Best model: log.Salary~1+AVG+Runs+Hits+HR+Walks+SO+SB+free.agency.eligibility+free.agent.in.1991.2+a
## Crit= 376.102535546764
## Mean crit= 376.102535546764
## Change in best IC: 0 / Change in mean IC: 0
##
## After 60 generations:
## Best model: log.Salary~1+AVG+Runs+Hits+HR+Walks+SO+SB+free.agency.eligibility+free.agent.in.1991.2+a
## Crit= 376.102535546764
## Mean crit= 376.102535546764
## Change in best IC: 0 / Change in mean IC: 0
##
## After 70 generations:
## Best model: log.Salary~1+AVG+Runs+Hits+HR+Walks+SO+SB+free.agency.eligibility+arbitration.eligibilit
## Crit= 374.748995323039
## Mean crit= 374.748995323039
## Change in best IC: -1.35354022372439 / Change in mean IC: -1.35354022372439
##
## After 80 generations:
## Best model: log.Salary~1+AVG+Runs+Hits+HR+Walks+SO+SB+free.agency.eligibility+arbitration.eligibilit
## Crit= 374.748995323039
## Mean crit= 374.748995323039
## Change in best IC: 0 / Change in mean IC: 0
##
## After 90 generations:
## Best model: log.Salary~1+AVG+Runs+Hits+HR+Walks+SO+SB+free.agency.eligibility+arbitration.eligibilit
## Crit= 374.748995323039
## Mean crit= 374.748995323039
## Change in best IC: 0 / Change in mean IC: 0
##
## After 100 generations:
## Best model: log.Salary~1+AVG+Runs+Hits+HR+Walks+SO+SB+free.agency.eligibility+arbitration.eligibilit
## Crit= 374.748995323039
## Mean crit= 374.748995323039
## Change in best IC: 0 / Change in mean IC: 0
##
## After 110 generations:
## Best model: log.Salary~1+AVG+Runs+Hits+HR+Walks+SO+SB+free.agency.eligibility+arbitration.eligibilit
## Crit= 374.748995323039
## Mean crit= 374.748995323039
## Change in best IC: 0 / Change in mean IC: 0
##
## After 120 generations:
## Best model: log.Salary~1+AVG+Runs+Hits+HR+Walks+SO+SB+free.agency.eligibility+arbitration.eligibilit
## Crit= 374.748995323039
## Mean crit= 374.748995323039
## Improvements in best and average IC have bebingo en below the specified goals.
## Algorithm is declared to have converged.
## Completed.
```

```
result.genetic_BIC = attributes(fit)$objects[[1]]

result.genetic_BIC$coef
```

```
##             (Intercept)                         AVG                      Runs
##             5.494058436            -2.196766127            -0.008585304
##                    Hits                      HR                   Walks
##             0.010566237             0.035264696             0.007433685
##                      SO                      SB free.agency.eligibility
##            -0.007558254             0.009424969             1.496185849
## arbitration.eligibility
##             1.182133605
```

```
library(ncvreg);
#Cross Validation
cvfit.L1 = cv.ncvreg(X=X,y=y, nfolds=10,
    penalty="lasso", lambda.min=.005)

beta.hat = coef(cvfit.L1)  # THE LASSO COEFFICIENTS WITH MINIMUM CV ERRO

cutoff = 0.0001
terms = names(beta.hat)[abs(beta.hat) > cutoff]
formula.LASSO = as.formula(paste(c("log.Salary ~ ", terms[-1]), collapse=" + "))
result.L1 = lm(formula.LASSO, data = train)
summary(result.L1)
```

```
##
## Call:
## lm(formula = formula.LASSO, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.42628 -0.29644 -0.00255  0.32087  1.24938
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)              5.469954   0.291639  18.756  < 2e-16 ***
## AVG                     -2.122701   1.187170  -1.788  0.07517 .
## Hits                     0.006679   0.002464   2.711  0.00724 **
## Doubles                  0.001042   0.007647   0.136  0.89177
## Triples                 -0.010616   0.019221  -0.552  0.58131
## HR                       0.017731   0.010372   1.710  0.08879 .
## RBI                      0.005745   0.004559   1.260  0.20891
## Walks                    0.004185   0.002526   1.657  0.09903 .
## SO                      -0.006732   0.002027  -3.321  0.00105 **
## SB                       0.005763   0.004220   1.366  0.17343
## Errs                    -0.004874   0.007007  -0.696  0.48746
## free.agency.eligibility  1.533602   0.103927  14.757  < 2e-16 ***
## free.agent.in.1991.2    -0.095053   0.126579  -0.751  0.45350
## arbitration.eligibility  1.232951   0.108257  11.389  < 2e-16 ***
## arbitration.in.1991.2   -0.212786   0.219471  -0.970  0.33336
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5342 on 217 degrees of freedom
## Multiple R-squared:  0.8062, Adjusted R-squared:  0.7936
## F-statistic: 64.46 on 14 and 217 DF,  p-value: < 2.2e-16
```

(e)

```
genetic_BIC_yhat = predict(result.genetic_BIC, test)
L1_yhat = predict(result.L1, test)
leaps_BIC_yhat = predict(result.leaps_BIC, test)
```

```
sum((test$log.Salary - genetic_BIC_yhat)^2)
```

```
## [1] 37.05357
```

```
sum((test$log.Salary - leaps_BIC_yhat)^2)
```

```
## [1] 33.97954
```

```
sum((test$log.Salary - L1_yhat)^2)
```

```
## [1] 33.20897
```

3. Final Fit

```
all_data=rbind(train, test)
fit.final = lm(formula.LASSO, data = all_data )
summary(fit.final)
```

```
##
## Call:
## lm(formula = formula.LASSO, data = all_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.42954 -0.28965 -0.04063  0.32862  1.45979
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)                5.273340   0.235432  22.399  < 2e-16 ***
## AVG                       -1.337387   0.950275  -1.407 0.160283
## Hits                       0.006243   0.002056   3.036 0.002592 **
## Doubles                   -0.002391   0.006564  -0.364 0.715861
## Triples                   -0.016831   0.015859  -1.061 0.289351
## HR                         0.004824   0.008736   0.552 0.581199
## RBI                        0.010058   0.003837   2.621 0.009176 **
## Walks                      0.002664   0.001960   1.359 0.175023
## SO                        -0.005475   0.001629  -3.360 0.000872 ***
## SB                         0.005033   0.003251   1.548 0.122577
## Errs                      -0.008098   0.005789  -1.399 0.162823
## free.agency.eligibility    1.580951   0.083511  18.931  < 2e-16 ***
## free.agent.in.1991.2      -0.254985   0.106229  -2.400 0.016948 *
## arbitration.eligibility    1.315555   0.091270  14.414  < 2e-16 ***
## arbitration.in.1991.2     -0.081353   0.185783  -0.438 0.661758
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5363 on 322 degrees of freedom
## Multiple R-squared:  0.8009, Adjusted R-squared:  0.7922
## F-statistic: 92.52 on 14 and 322 DF,  p-value: < 2.2e-16
```

Variable selection with LASSO results in a model using 14 out of 16 non-intercept variables for the training data frame. The excluded variables are OBP and Runs.

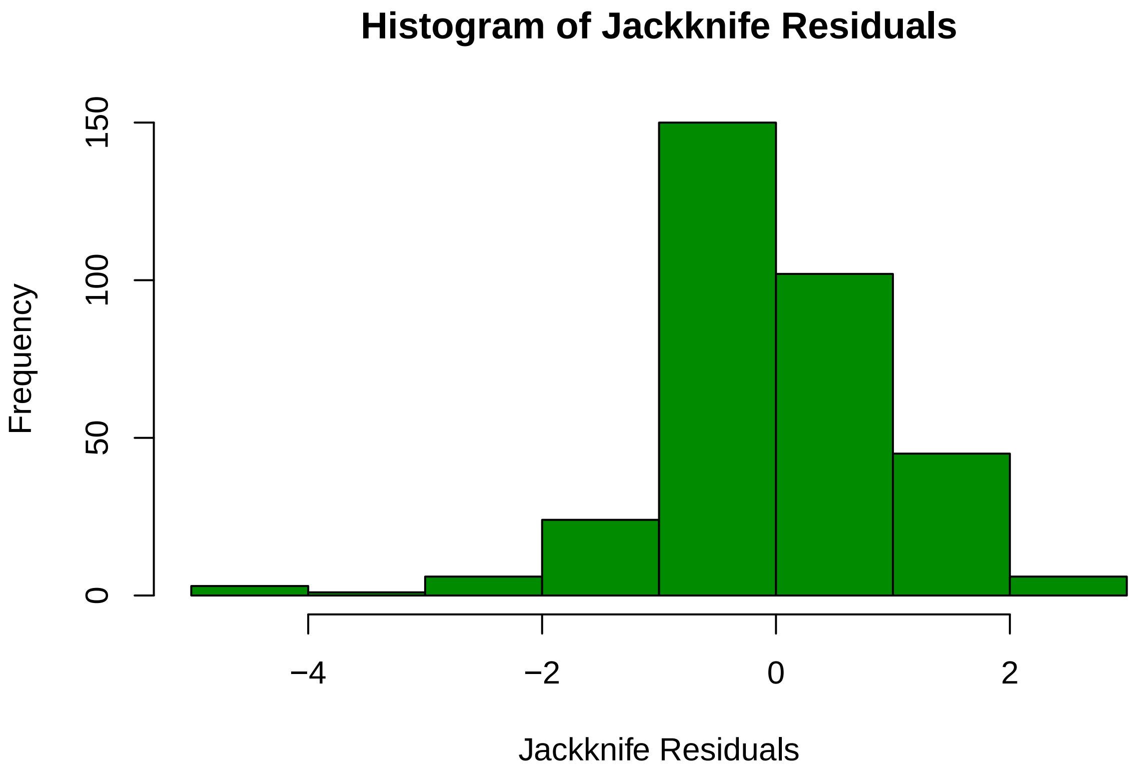
4. Model Diagnostics

(a) Normality

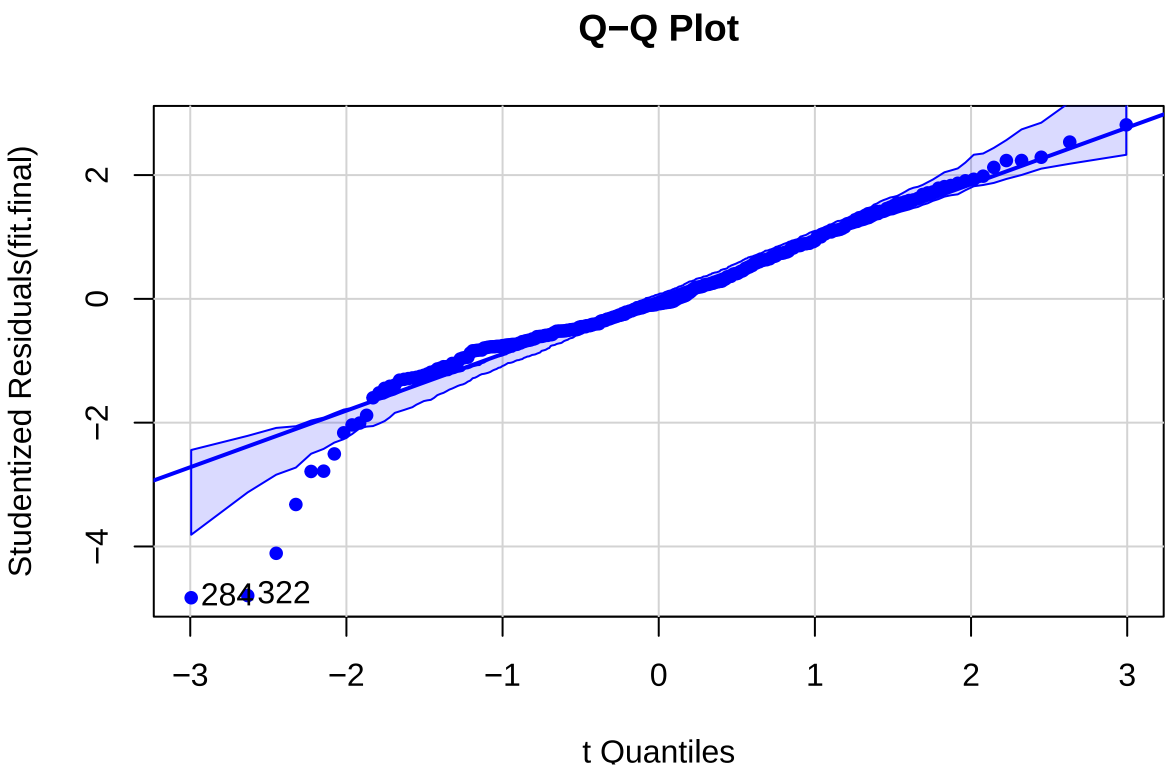
```
library(car)
```

```
## Loading required package: carData
```

```
r.jack = rstudent(fit.final)
hist(r.jack, xlab="Jackknife Residuals", col="green4", main = "Histogram of Jackknife Residuals")
```



```
qqPlot(fit.final, pch=19, cex=.8, col="blue", main="Q-Q Plot")
```



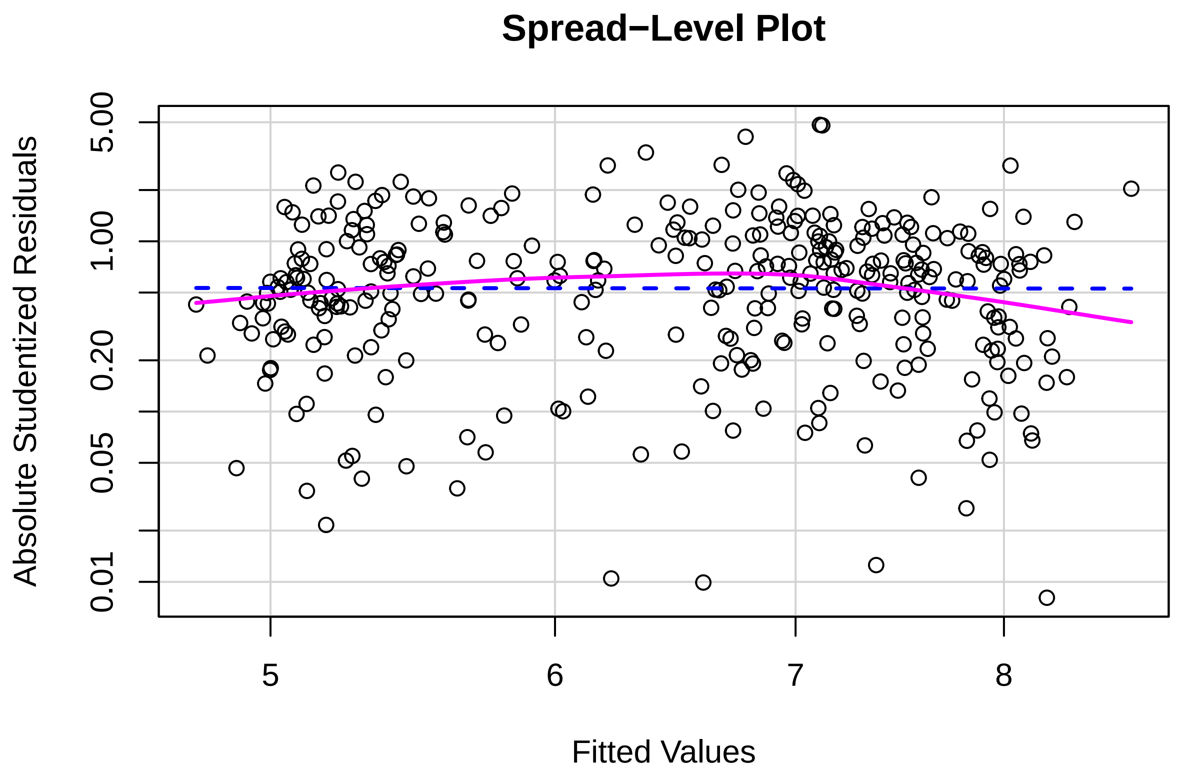
```
## 322 284
## 225 320
```

```
shapiro.test(r.jack)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  r.jack
## W = 0.94772, p-value = 1.485e-09
```

(b) Homoscedasticity

```
spreadLevelPlot(fit.final, main = "Spread-Level Plot")
```



```
##
## Suggested power transformation:  1.014462
```

```
ncvTest(fit.final)
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 0.2720911, Df = 1, p = 0.60193
```

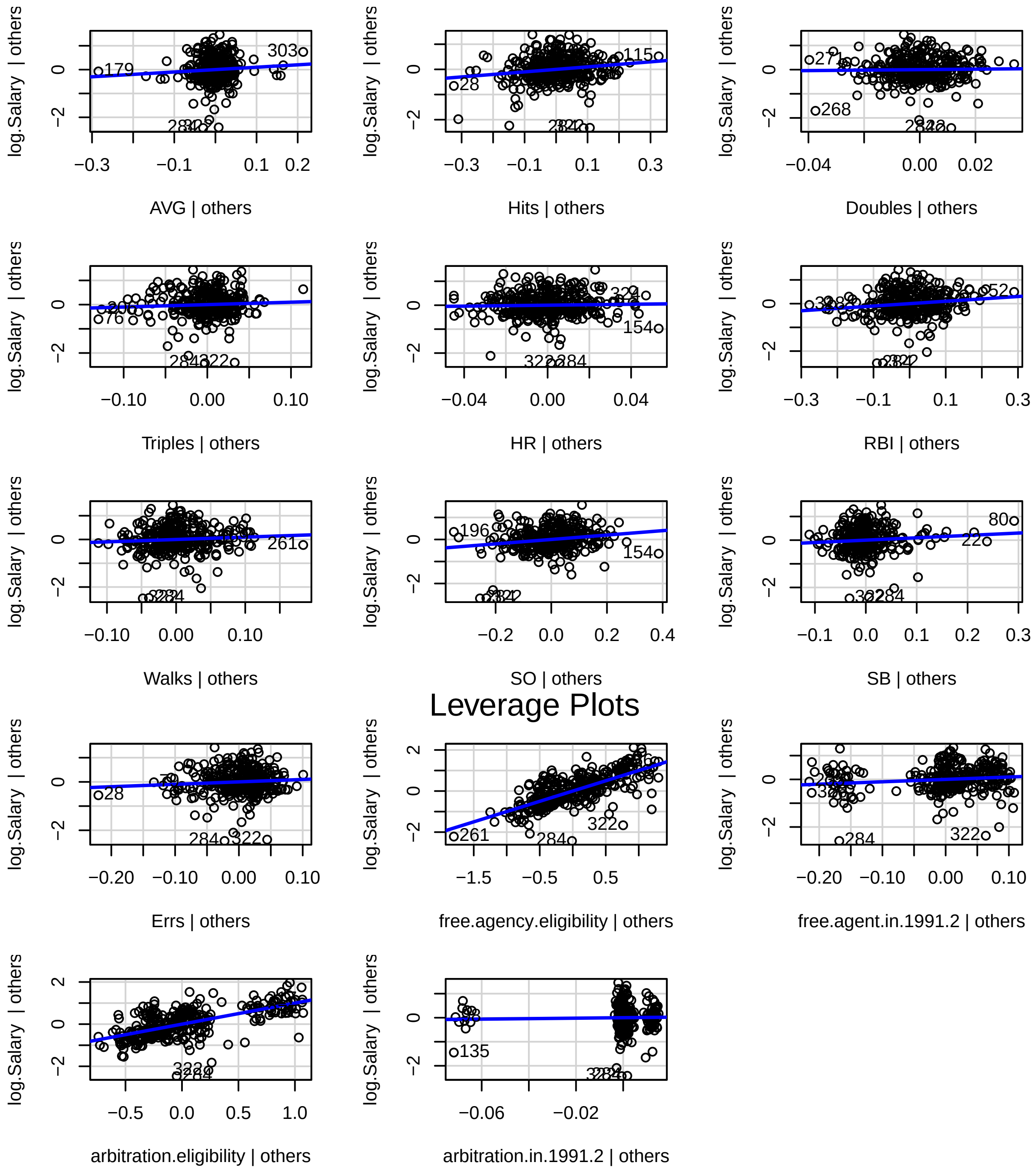
(c) Independence

```
durbinWatsonTest(fit.final)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1      0.05758477      1.880886    0.28
##  Alternative hypothesis: rho != 0
```

(d) Linearity

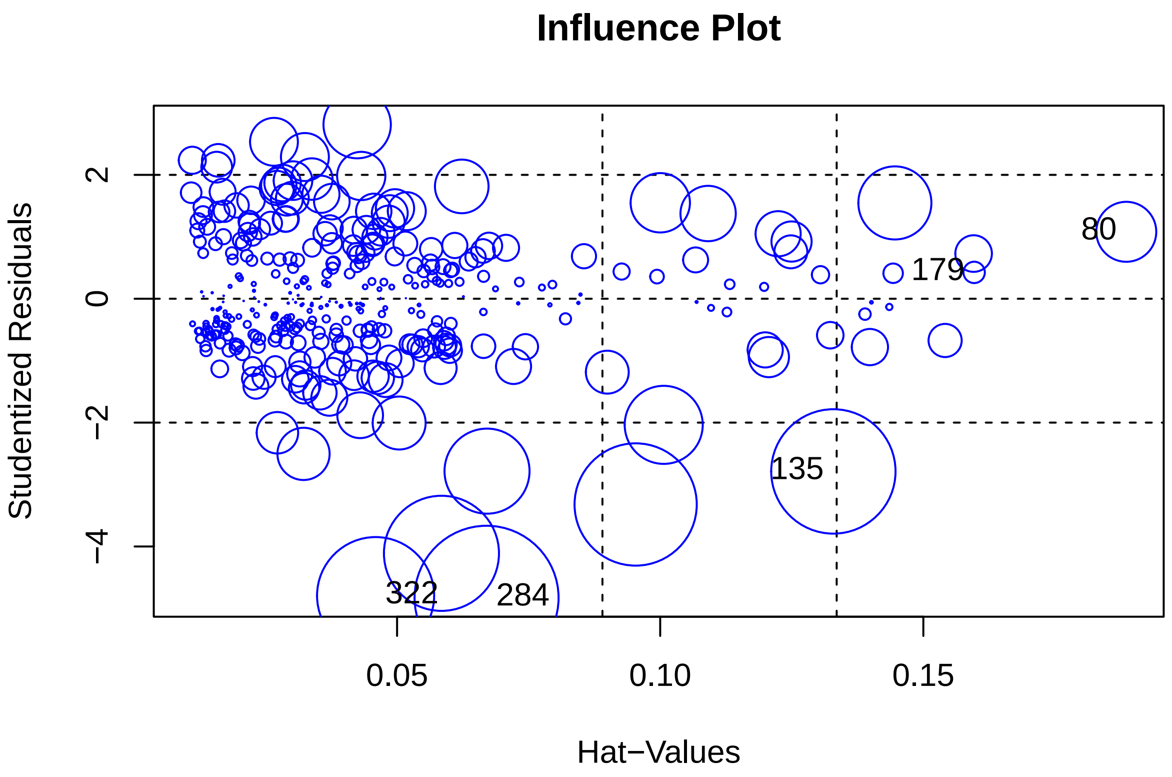
```
leveragePlots(fit.final)
```



(e) Outlier Detection

```r
influencePlot(fit.final, id.method="identify", col="blue",
main="Influence Plot")
```

```
## Warning in plot.window(...): "id.method" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "id.method" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "id.method" is not
## a graphical parameter
## Warning in axis(side = side, at = at, labels = labels, ...): "id.method" is not
## a graphical parameter

## Warning in box(...): "id.method" is not a graphical parameter

## Warning in title(...): "id.method" is not a graphical parameter

## Warning in plot.xy(xy.coords(x, y), type = type, ...): "id.method" is not a
## graphical parameter
```



```
##         StudRes        Hat       CookD
## 135 -2.7882378 0.13290142 0.077801425
## 322 -4.7938514 0.04592104 0.069028217
## 80   1.0807476 0.18855934 0.018085146
## 179  0.4252747 0.15965559 0.002296577
## 284 -4.8284019 0.06700912 0.104393625
```

The negative outliers in y tend to be more influential than the positive outliers. For example, the bubble plot identifies 322, 284, and 135 as influential outliers which have negative studentized residuals. There are three influential outliers in the input space identified by the bubble plot. These are 135, 179, and 80.

(f) Multicollinearity

```
no_intercept = lm(log.Salary ~ +AVG + Hits + Doubles + Triples + HR + RBI + Walks +
    SO + SB + Errs + free.agency.eligibility + free.agent.in.1991.2 +
    arbitration.eligibility + arbitration.in.1991.2 -1, data=all_data, x=TRUE)
kappa(no_intercept$x)
```

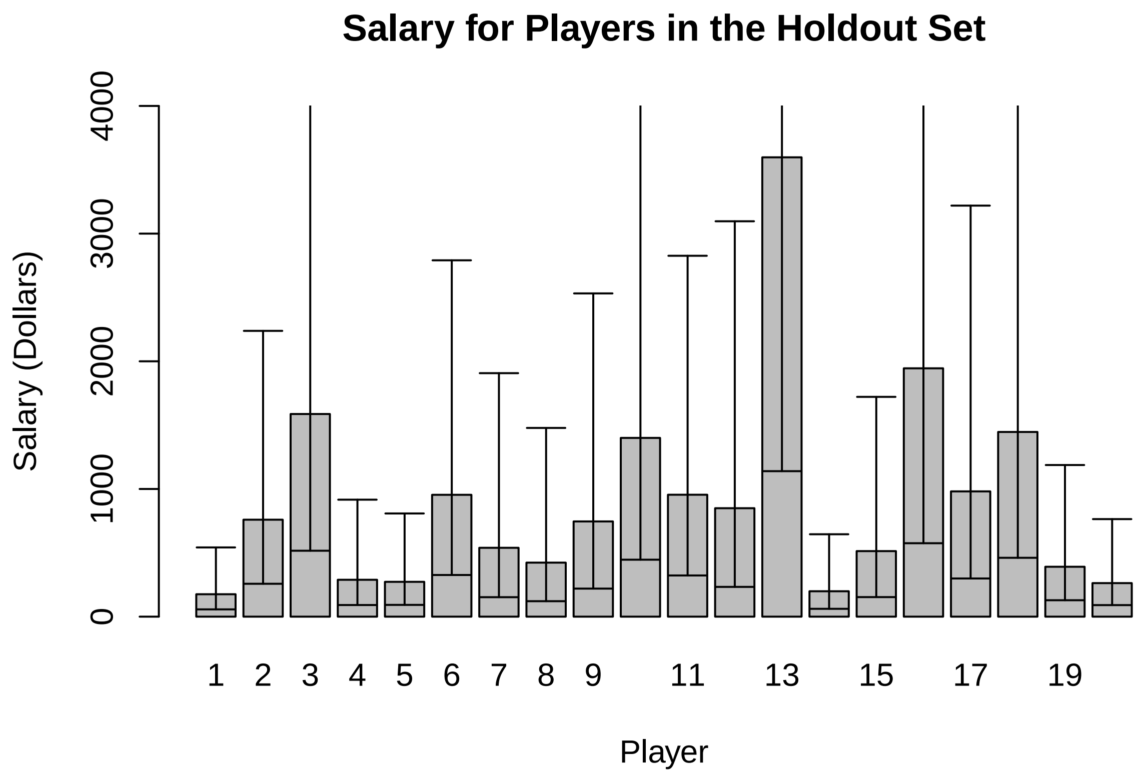
```
## [1] 741.5499
```

```
vif(no_intercept)
```

```
## Warning in vif.default(no_intercept): No intercept: vifs may not be sensible.

##                     AVG                    Hits                 Doubles
##                6.322979               51.631732               19.529349
##                 Triples                      HR                     RBI
##                3.511430               14.690695               48.327823
##                   Walks                      SO                      SB
##                8.291034               10.796023                2.522311
##                    Errs free.agency.eligibility    free.agent.in.1991.2
##                3.154481                3.131448                1.529926
## arbitration.eligibility   arbitration.in.1991.2
##                1.840444                1.199596
```

5. Model Development

```
holdout_data = read.csv("bb92-test.csv")
colnames(holdout_data) = colnames(baseball)[c(-1,-18, -19)]
holdout_pred = predict(fit.final, holdout_data, interval = "prediction")
my_barplot = barplot(exp(holdout_pred[,1]) , xlab = "Player",
ylab = "Salary (Dollars)", main = "Salary for Players in the Holdout Set",
ylim = c(0,4000))

arrows(x0 = my_barplot,
       y0 = exp(holdout_pred[,3]),
       y1 = exp(holdout_pred[,2]),
       angle = 90,
       code = 3,
       length = 0.1)
```


Salary for Players in the Holdout Set
Salary (Dollars)
0
1000
2000
3000
4000
1
2
3
4
5
6
7
8
9
11
13
15
17
19
Player