Three Approaches to Predicting Major League Baseball Player Salary

Andrew Bates

Executive Summary

In this paper we compare the ability of three types of models to predict Major League Baseball player salary given performance metrics from the 1986 season and over a players career. We build and evaluate one model that falls under Leo Breiman's *data modeling* culture (linear regression), a model from his *algorithmic modeling* culture (random forest), and a model that straddles the two groups (lasso). Predictive power is compared via error on cross-validation, error from predicting on a test set, and the increase in prediction error when going from cross-validation to test set. Random forest is the best predictor with a mean cross-validation error of 0.43 and a test set error of 0.46. Linear regression and lasso are comparable to each other with cross-validation errors of 0.53 and 0.54, respectively, and test set errors of 0.69 and 0.71. Both linear models had an approximately 30% increase in error going from cross-validation to test set predictions while random forest saw an increase of only 7%. In addition to predictive performance we compare the interpretability of each model. For linear regression this takes the form of the usual inferences on the regression coefficients. For lasso and random forest interpretability is viewed in the context of variable importance measures. The linear regression model included only one career level variable while the random forest deemed the career level covariates most important. The lasso was in between as it did not include all career variables but of those included, half were deemed most important.

1 Introduction

In 2001 Leo Breiman described two approaches to analyzing data with statistical models (Breiman 2001). In *data modeling* we specify a stochastic model to describe the data, one that has known theoretical properties, and the main purpose is usually to make inferences on the population of interest. In the other approach, what Breiman calls *algorithmic modeling*, the focus is less on the the structure of the data itself and more about the output of the model. We are not concerned with finding a model that satisfies the theoretical assumptions needed to make inferences but are interested in whether the model can make accurate predictions on newly collected data. In this paper we compare Breiman's two modeling paradigms by analyzing Major League Baseball data with the goal of developing a model to predict a players salary. We examine one data model (linear regression), one algorithmic model (random forest), and one model at the intersection of the two approaches (LASSO). For each model, we discuss some advantages and disadvantages in terms of both predictive capability and interpretability.

2 Methods

In this analysis we use the `Hitters` data from the R package `ISLR` (James et al. 2017), a companion package to *An Introduction to Statistical Learning with Applications in R* (James et al. 2013) containing the data sets used in the book. The Hitters data contains information on 322 players from the 1986 and 1987 Major League Baseball (MLB) seasons. There are 19 covariates included in this data set that can mostly be broken down into two categories: performance metrics for the 1986 season (number of home runs, etc.), and performance metrics based on a given players career (career runs, etc.) as of 1986. There are 16 continuous variables and three categorical variables, two of which indicate a players league (American or National) and division (East or West). For the 1987 season we have the players salary on opening day along with their league at the beginning of the season.

Salary has 59 missing observations, 18% of the data[1]. This was too many observations to ignore so we imputed the values using k-nearest neighbors before proceeding with the analysis. After examining histograms of the continuous variables, it was evident that transformations were in order. Salary, along with several covariates, were heavily right-skewed. We chose log transformations because it is a familiar technique for handling skewness and allows us to readily interpret linear regression coefficients. In all, 11 of the 20 variables were log transformed. All numeric variables were then subsequently centered about the mean and scaled by the standard deviation prior to model fitting.

Before fitting models the data was split into a training and testing set with 20% reserved for testing. All three models were trained using 5-fold cross-validation with the final model chosen to be the one with the lowest root mean squared error (RMSE). In each cross-validation run for linear regression a stepwise procedure was used with Akaike Information Criteria (AIC) as the model selection criterion. Diagnostics were run on the model chosen via cross-validation and some covariates were subsequently omitted based on variance inflation factors and correlations between the covariates. A grid search was used for hyperparameter tuning of the lasso and random forest with 10 values considered for each. For lasso the hyperparameter is the penalty on the L_1 norm of the regression coefficients and for random forest the hyperparameter is the number of randomly selected covariates considered at each split. A comparison of predictive ability for the three models was made through cross-validation RMSE and error on the testing set as well as the increase in prediction error going from cross-validation to test set. We investigate interpretability via coefficient interpretation for linear regression and variable importance for lasso and random forest.

The analysis was conducted using the statistical software R (R Core Team 2018). This document was written using the R packages `R Markdown` (Allaire et al. 2018), `knitr` (Xie 2018b), and `bookdown` (Xie 2018a). All materials used to conduct the analysis and compose the report can be found at https://github.com/asbates/stat696/tree/master/reports/baseball.

3 Analysis

3.1 Exploratory Analysis

There are a few areas of concern with the data used in this analysis. The first being a number of missing values for player salary. Almost 20% of the data has missing salary values. Most of these are likely due to retirement as 57 players retired in 1986[2]. Regardless of the reason, with so many missing values we decided to impute them using k-nearest neighbors. The data set is not particularly large at 322 observations so omitting missing values would be leaving out a large portion of the data.

[1] None of the covariates had massing values.

[2] http://www.baseball-almanac.com/yearly/final.php?l=NL&y=1986

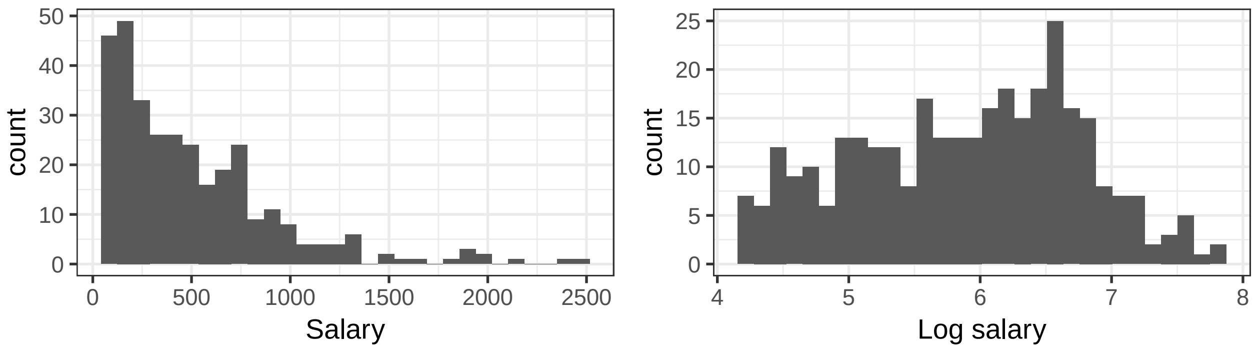

Figure 1: Histogram of player salary. The left plot is prior to transformation and the right plot is after log transforming.

Another potential issue with this data set is the correlations between the covariates. A plot of the correlation matrix for the continuous variables is given in the appendix (Figure 3). Several covariates have extremely high correlations, up to 98%. We also see groupings of covariates that have large correlations with each other. There are two groups of six variables each. One group contains career level variables and the other contains season level covariates. For example, the number of hits and at bats in 1986 have a correlation of 97%. This should not be surprising as players can only get a hit if they are at bat. But the more troubling issue is that both have high correlations with number of runs (92% for hits and 91% for at bats) and the number of runs batted in (81% for hits and 82% for at bats). Additionally, they both have moderate correlations with number of home runs and number of walks. These are likely to present problems, especially in the linear regression model. However, the initial model selection procedure for linear regression is based on prediction error and fits subsets of the covariates so it may not include an entire group of correlated variables. For now we refrain from removing any variables and reassess the issue after model selection.

Over half of the continuous variables exhibit skewness to some degree. This is of highest concern for the linear regression model but it may also affect the fit for lasso and random forest if most of the values are clumped together. Log transformations were used to account for skewness, favored for the interpretability of coefficients in the linear regression model. An example of this skewness is seen in Figure 1 where we have a histogram of salary along with a histogram of log salary. The log transformation clearly helps with skewness but also note that after transformation salary is approximately normally distributed. Histograms of the remaining variables are provided in the appendix for reference (Figure 5) as well as histograms of the log transformed variables before and after transformation (Figures 6 and 7). Also included are scatter plots of each covariate versus log salary to assess the plausibility of the linearity assumption for the linear regression model (Figures 8 and 9). The plots show that each predictor variable has an approximately linear relationship with log salary.

3.2 Modeling Fitting

After splitting the data into a testing and training set, each type of model was fit to the training data set with 5-fold cross-validation. The same cross-validation training and testing sets were used for each type of model. In each case, the final model was the one that resulted in the lowest root mean squared error (RMSE), averaged over the cross-validation runs. Prior to performing cross validation each continuous predictor was first centered by the mean and scaled by the standard deviation.

For the linear regression model a backwards stepwise procedure was used on each cross validation run. The resulting fit was the one with the lowest Akaike Information Criteria and this was then used to predict on the cross-validation hold out set. The model with the lowest mean cross-validation error was then evaluated. The coefficient for log put outs had a reasonably large p-value at 0.13 but the p-values for the remaining coefficients were satisfactory. The real concern with this model is the variance inflation factors (VIF). Number of at bats had a VIF of 19.2 and number of hits had a VIF of 16.9. These are quite large and indicate collinearity issues. This should not be surprising as we noted earlier that these variables are highly correlated (97%) and they both have moderate to high correlations with other variables. Number of at bats generally has higher correlations with other covariates than number of hits so it was decided to remove at bats and keep number of hits.

After removing at bats from the linear regression model the collinearity issue was mostly abated. The largest VIF for this model was moderate at 5.9 (for number of runs batted in). Further diagnostics were done through plotting residuals versus fitted values and a QQ plot, both of which can be found in Figure 11 in the appendix. We see one concerning point with a rather large residual. The residual plot also shows a few points with variance a bit larger than most of the others. For the quantile-quantile plot, we see a point of concern that distances itself from the remaining points. However, recall that the primary purpose of this analysis is to compare predictive performance across each class of model and not necessarily to construct a linear regression model that perfectly satisfies its assumptions. Overall, both of the diagnostic plots in Figure 11 are reasonable approximations and are satisfactory for our purposes. Proceeding with this model, cross-validation was then performed using the same fit on each run in order to get an estimate of out-of-sample RMSE and allow for comparison with the other models.

Lasso and random forest were fit with ten values of their respective hyperparameters on each cross-validation iteration. The parameter for the lasso is the penalty parameter on the L_1 norm of the coefficients. Ten equally spaced values were used ranging from 0.01 to 1. The optimal parameter was 0.01. For the random forest the hyperparameter is the number of randomly selected predictor variables considered at each split. Values ranged from 2 to 19 with the optimal parameter found to be 3.

3.3 Prediction Results

Table 1 displays the prediction error for each model across the cross-validation runs and on the testing set. As one would expect, the random forest model outperforms linear regression and lasso on both cross-validation error and test set error. The random forest is approximately 50% better than the alternatives in terms of relative error. The errors for linear regression and lasso are approximately the same which is a bit remarkable given that lasso often has improved predictive performance compared to linear regression (James et al. 2013, 203). Also of note, and perhaps most important, is the percent increase in error on the testing set compared to cross-validation error. Both linear regression and lasso saw an increase of about 30% while the random forest increase was only 7%.

Part of the performance gap between the linear methods and random forest is likely related to the groups of predictors with high correlations. Recall that there are two groups of six variables where each variable in a group has moderate to large correlation with the others. The optimal number of

Table 1: Prediction results for linear regression, lasso, and random forest. Included are the mean cross-validation RMSE and test set RMSE. Also included is the increase in error from cross validation to testing and relative test set error.

Model	**CV error**	**Test error**	**% Increase in error**	**Test relative error**
Random Forest	0.43	0.46	7.3	1.00
Linear Regression	0.53	0.69	30.0	1.49
LASSO	0.54	0.71	32.0	1.53

variables considered at each split for the random forest was three so it may be that only one or two predictors from each group is selected at a time. This would result in a better fit for each tree in the forest and hence a better fit when their predictions are averaged.

3.4 Model Interpretation

Interpretation is another aspect for which the three models considered here differ. Even if the main objective for building a model is prediction accuracy one usually wants to have an understanding of how the model is working. However, the type of model dictates what kinds of interpretations we can make. For linear regression, inferences most often take the form of interpreting the regression coefficients. The lasso is based on a standard linear regression model but the nature of the fitting process makes linear-regression-like inferences difficult. Random forests are similarly difficult to interpret as we do in linear regression. Although there has been some recent development on inference for both lasso (Lee et al. 2016) and random forest (Mentch and Hooker 2016), they are beyond the scope of this analysis. Instead we take the traditional approach of model interpretation through feature importance measures. Although we can not make the same type of statements about the lasso and random forest models that we can in linear regression, variable importance measures still give us insight into how the models are behaving.

A summary of the linear regression fit is given in Table 2. Included are coefficient estimates of the model along with their associated standard errors, p-values, and 95% confidence intervals. Because log transformations and centering and scaling of the covariates was performed, the interpretations of the coefficients are not as straightforward as when no transformations are performed. There are three types of predictors in the model and each warrants a slightly different interpretation. The simplest case is the covariates that were not log transformed, such as number of hits. Since the predictors were centered and scaled prior to fitting, we can interpret number of hits as follows. Hits has a standard deviation of 46 and its coefficient estimate is 0.18. So every 46 hits a player gets per season is associated with a 100 * 0.18 or 18% increase in salary[3]. Now let's consider a covariate that was log transformed, log home runs. Log home runs has a standard deviation of 1 and a coefficient of 0.11. Suppose log home runs for a given player increases by one standard deviation. If we take 1 + 1 and raise it to the power of 0.11 the result is 1.079. Subtract one from this and multiply by 100 and we get 7.9% which is the associated increase in salary we expect. The remaining type of predictor is the categorical variable for division. We leave this to the reader to interpret.

We should note that these inferences do not take into account the model selection procedure and as such should be handled with caution. Tibshirani et al. (2016) discuss how to account for this

[3]Of course, interpretations for linear regression must be made with the caveat that only the variable under question changes and the others are held fixed.

Table 2: Summary for linear regression model of log salary against hits, log home runs, runs batted in, walks, log put outs, log assists, log career runs batted in, and division. Coefficient estimates are provided along with their standard errors, p-values, and 95% confidence intervals.

Term	**Estimate**	**SE**	**p-value**	**95% CI**
Intercept	5.88	0.03	0.000	(5.82 , 5.94)
Hits	0.18	0.06	0.005	(0.06 , 0.31)
Log home runs	0.11	0.05	0.033	(0.01 , 0.21)
RBIs	-0.18	0.08	0.023	(-0.33 , -0.02)
Walks	0.08	0.04	0.052	(-0.00 , 0.17)
Log put outs	0.04	0.04	0.208	(-0.02 , 0.11)
Log assists	0.04	0.03	0.205	(-0.02 , 0.11)
Log career RBIs	0.59	0.04	0.000	(0.51 , 0.67)
Division: West	-0.09	0.03	0.005	(-0.16 , -0.03)

with forward stepwise regression but we use backwards selection here. In a higher stakes setting one should consider the implications of the model selection mechanism but for reasons of simplicity we assume our inferences are reasonable approximations.

For lasso and random forest we interpret the models via variable importance measures. For variable importance of the lasso we use the absolute value of the regression coefficients. Feature importance for random forest is based on the difference in out-of-bag predictions before and after permuting a variable. Importance measures for both models are then scaled so that the most influential feature has an importance of 100. Plots of variable importance measures for lasso and random forest are given in Figure 2. Only variables with a positive importance are plotted. For the lasso the most important feature is log career walks. This may be because if a player can get on base, regardless of how they do so, they at least have a chance to score. The top 6 features for random forest are career level variables which seems reasonable because they take into account a players performance over their career instead of just a single season. Sometimes in baseball a players performance in a given season does not reflect their performance over their career; they may play better or worse this season compared to the past five. The random forest seems to be taking this into account. In contrast, the linear regression model only included one career level variable, log career runs batted in. The liner regression model may be suffering in terms of predictive error because it did not include many career level variables. The lasso falls somewhere in between. Four career level covariates are included in the lasso model with two of them having the highest importance.

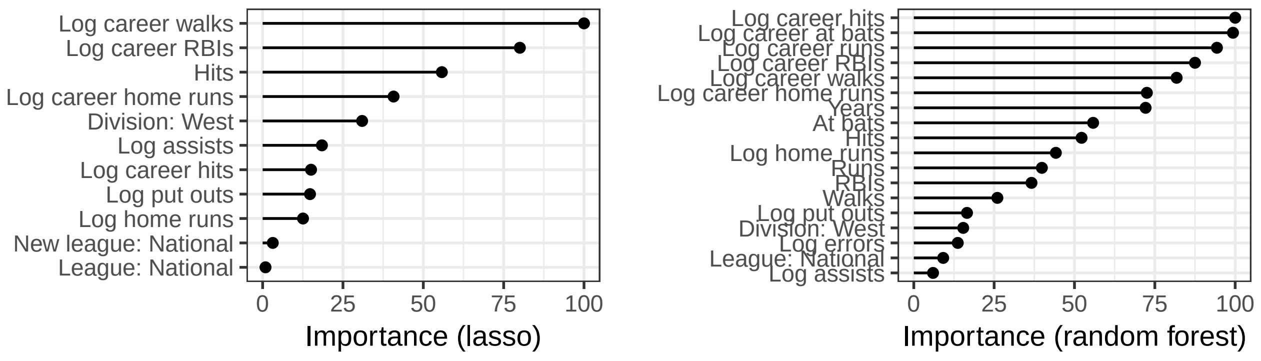

Figure 2: Variable importance plots for lasso (left) and random forest (right). Importance measures are scaled so that the most influential feature has an importance value of 100. Only variables with importance larger than 0 are included.

4 Conclusion

In this analysis we compared the ability of three classes of models to predict a baseball players salary given previous season information and overall career performance. The linear regression and lasso models achieved a comparable test set RMSE of 0.69 and 0.71, respectively. The random forest significantly outperformed the other models with a test set RMSE of 0.46. Random forest also had less of a gap between its cross-validation and testing error compared to the other models. Linear regression and lasso had an approximately 30% increase in error while random forest had an increase of only 7%.

In terms of interpretability, the linear regression model has the advantage in that we can understand how a particular covariate relates to the response rather than just which covariates are important. That being said, we did not account for the linear regression model selection procedure so the inferences from the model are made with possibly strong assumptions. Although we did not make the same types of inferences for lasso and random forest, we nevertheless can obtain a high level understanding of the models through variable importance measures. There has been some recent results that allow for inferences in these models. They were beyond the scope of this analysis but perhaps a future study could investigate these methods. Also note that the transformations used in this analysis were fairly simple and the number of variables was not very large. Had we considered more complex transformations or had say, 30 predictors instead of 19, the interpretation of the linear regression model would be more difficult.

It should be noted that this analysis has limitations. One issue is that the data is over 30 years old. Variables that have an impact on salary today are likely different than three decades ago. There have also been a substantial amount of new metrics introduced since 1986 that may have more of an impact on salary than the simple ones included in this data[4]. It would be interesting to have a similar analysis performed on a contemporary data set to see if that is the case. The results might also have been different if a few more variables were included that would have been available in 1986. For example, player position could have allowed the models to treat pitchers different than outfielders. Alternative performance metrics may have also mitigated the issues with correlations in this data set.

[4]See for example http://m.mlb.com/glossary/advanced-stats for a list of modern metrics.

A Supplementary Figures

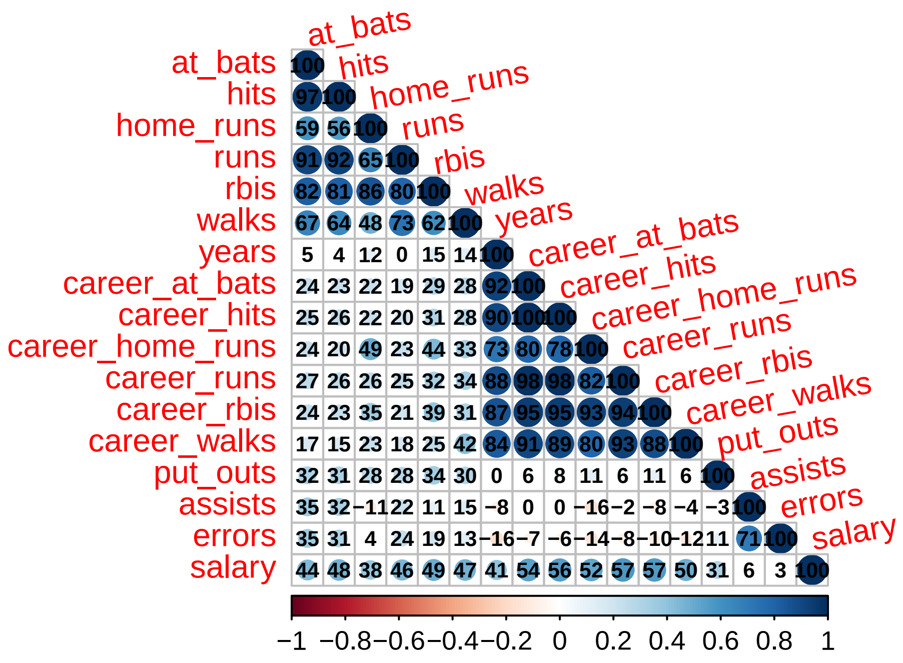

Figure 3: Correlation matrix plot of continuous variables prior to transformation. Correlations are displayed as a percent.

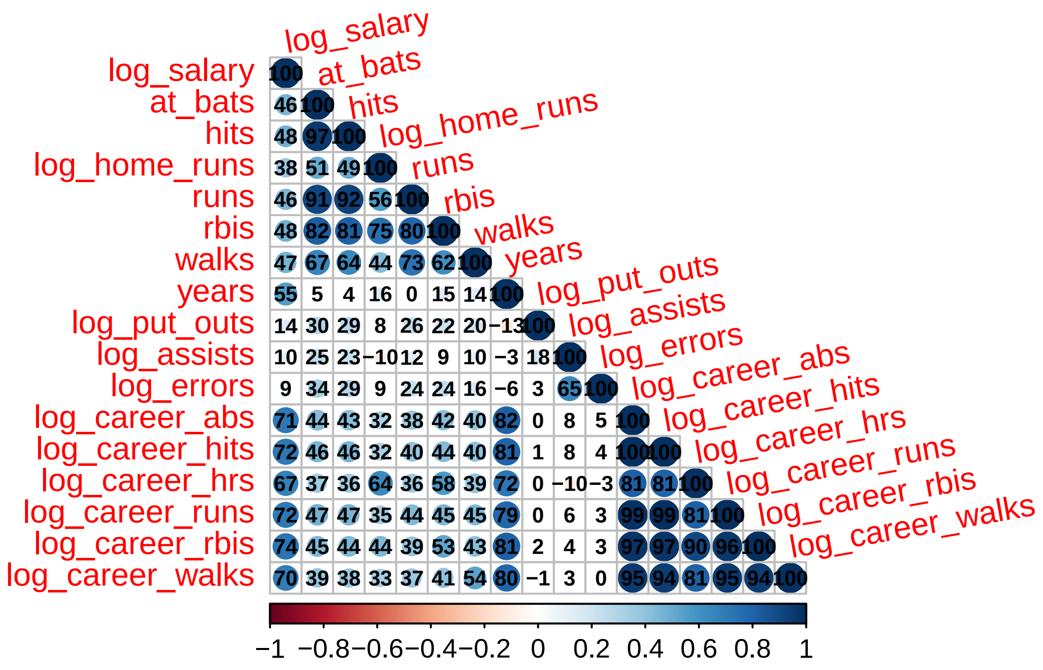

Figure 4: Correlation matrix plot of continuous variables after log transformation. Correlations are displayed as a percent.

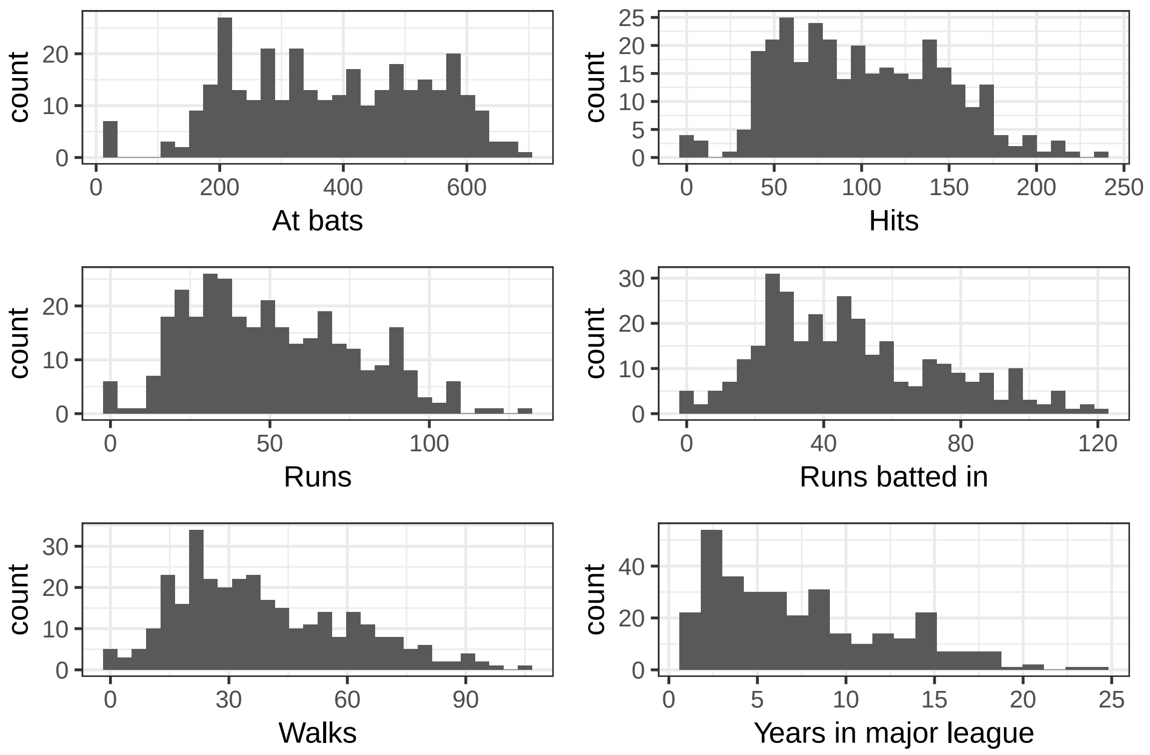

Figure 5: Histograms of covariates for which log transformations were not performed

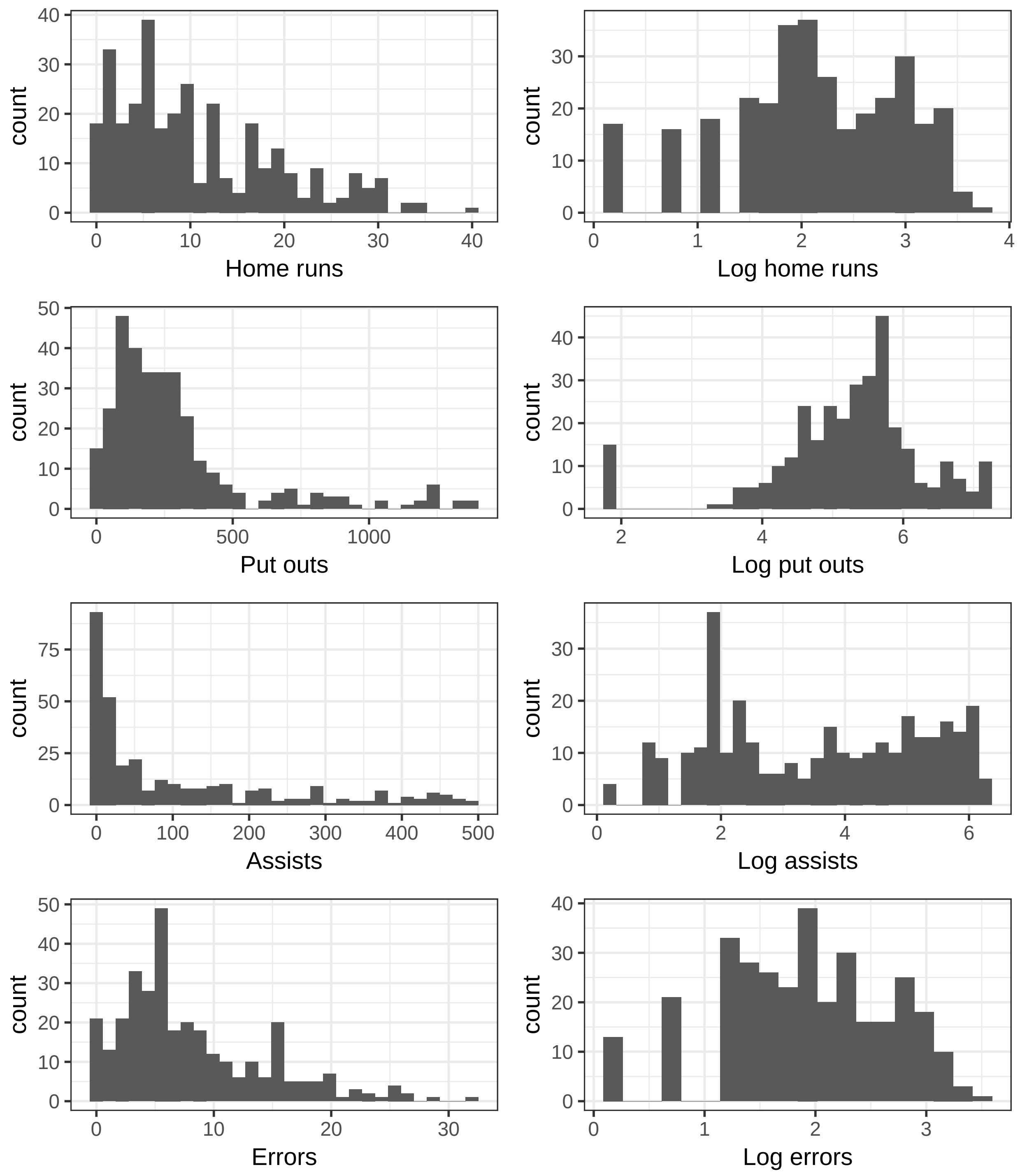

Figure 6: Histograms of season level covariates that were log transformed. The left side is the variable on the original scale and the right side is the variable on the log scale.

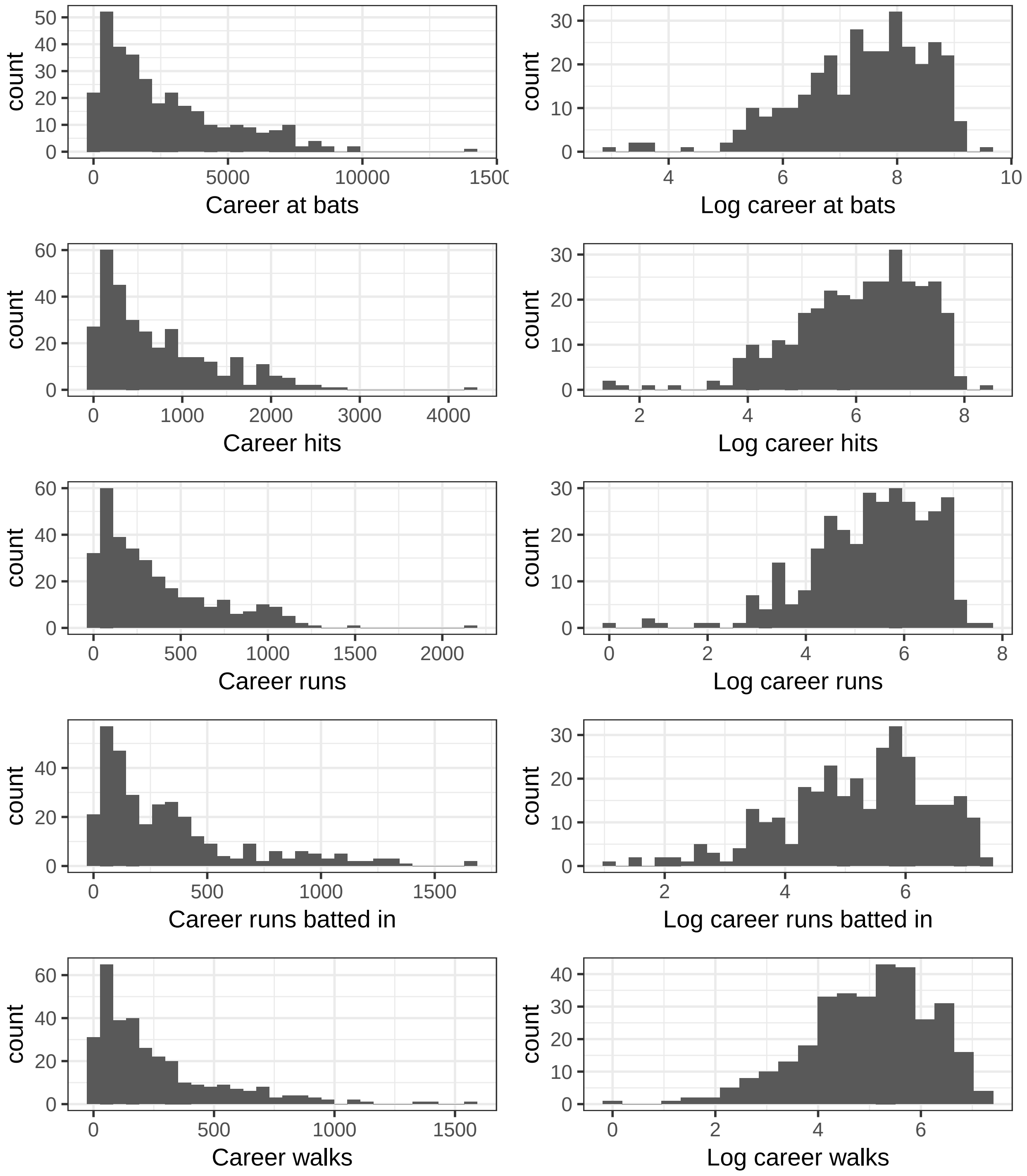

Figure 7: Histograms of career level covariates that were log transformed. The left side is the variable on the original scale and the right side is the variable on the log scale.

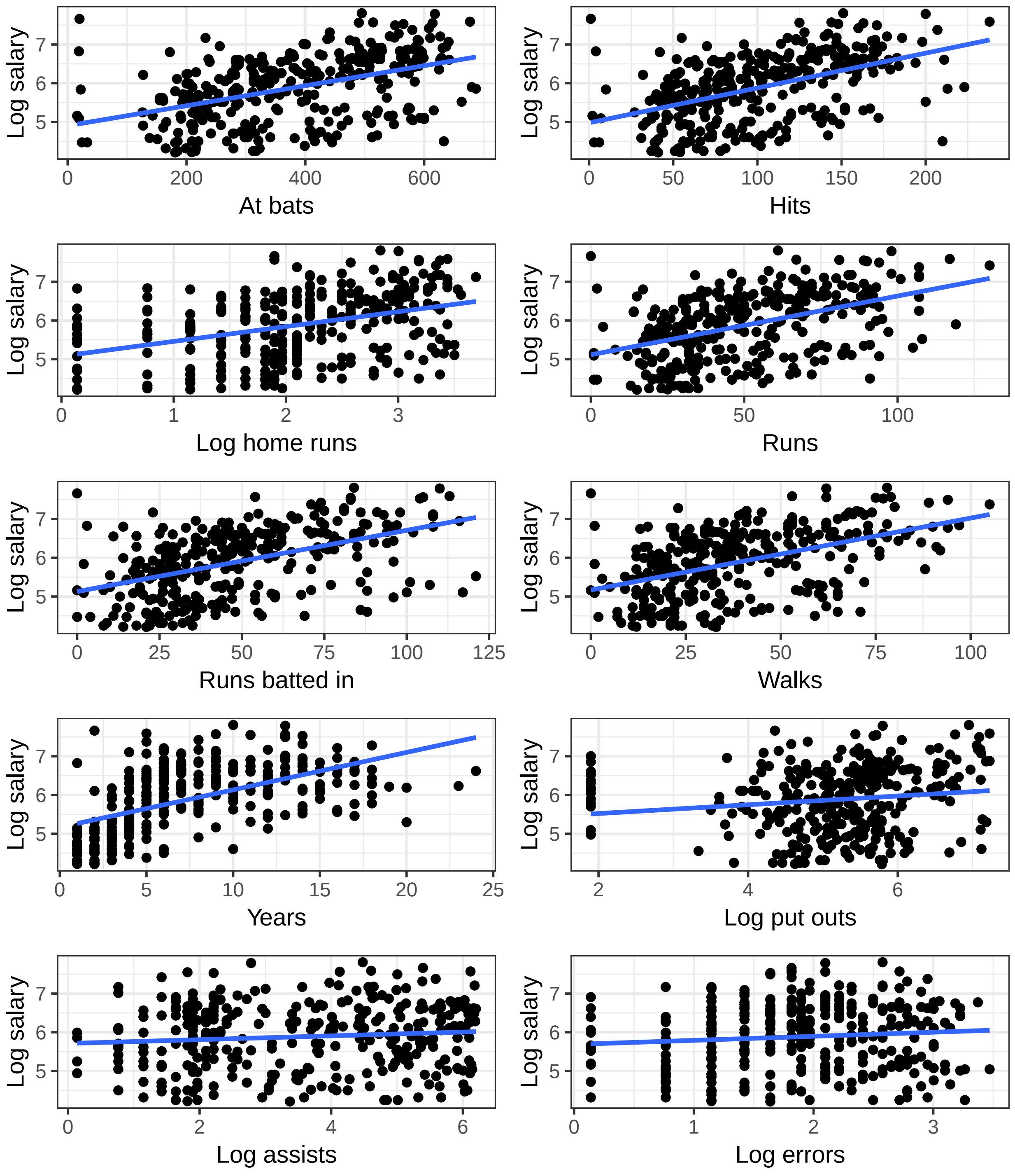

Figure 8: Scatter plots of log salary vs. season level covariates. The blue line is a smoothing line.

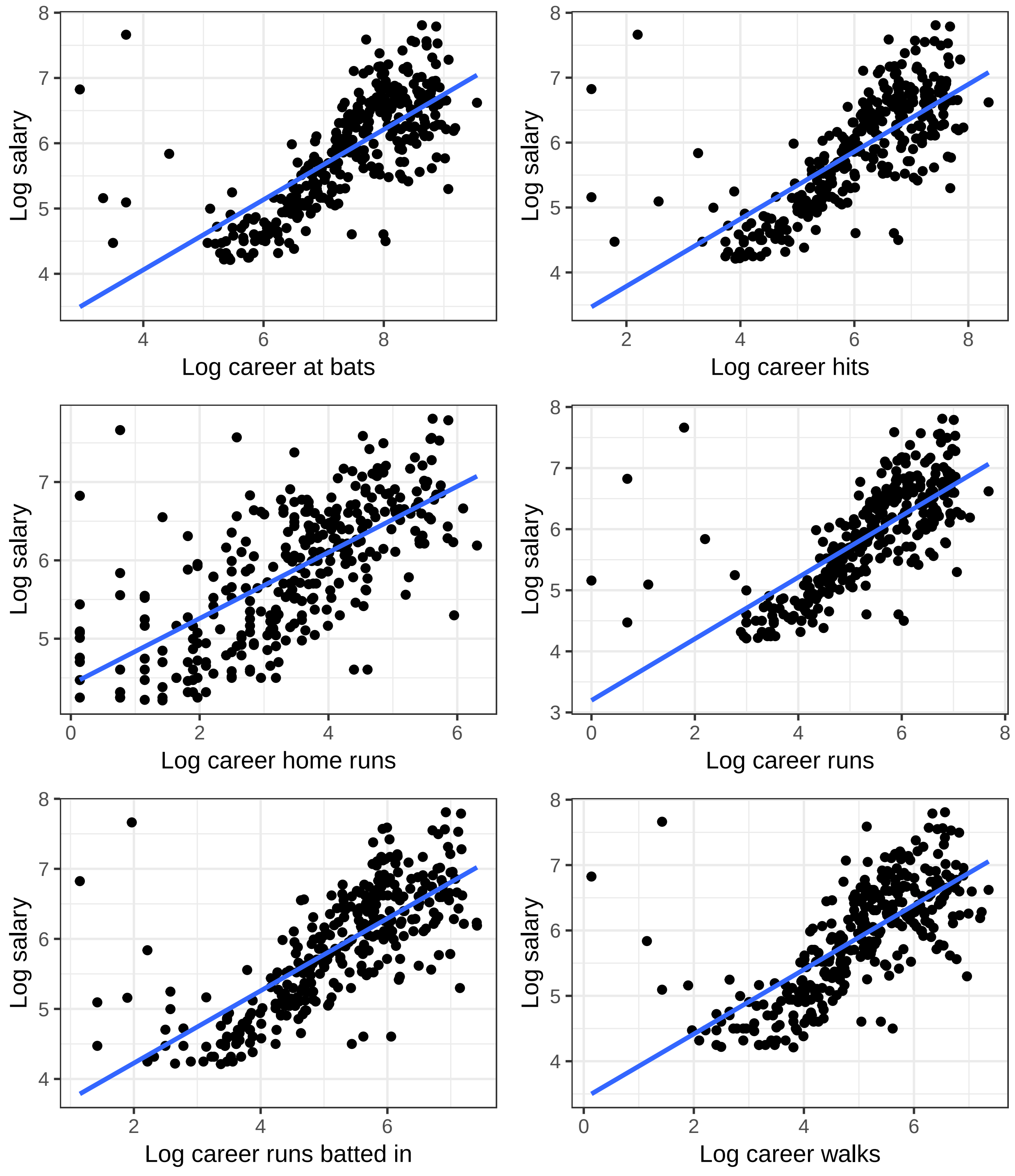

Figure 9: Scatter plots of log salary vs. career level covariates. The blue line is a smoothing line.

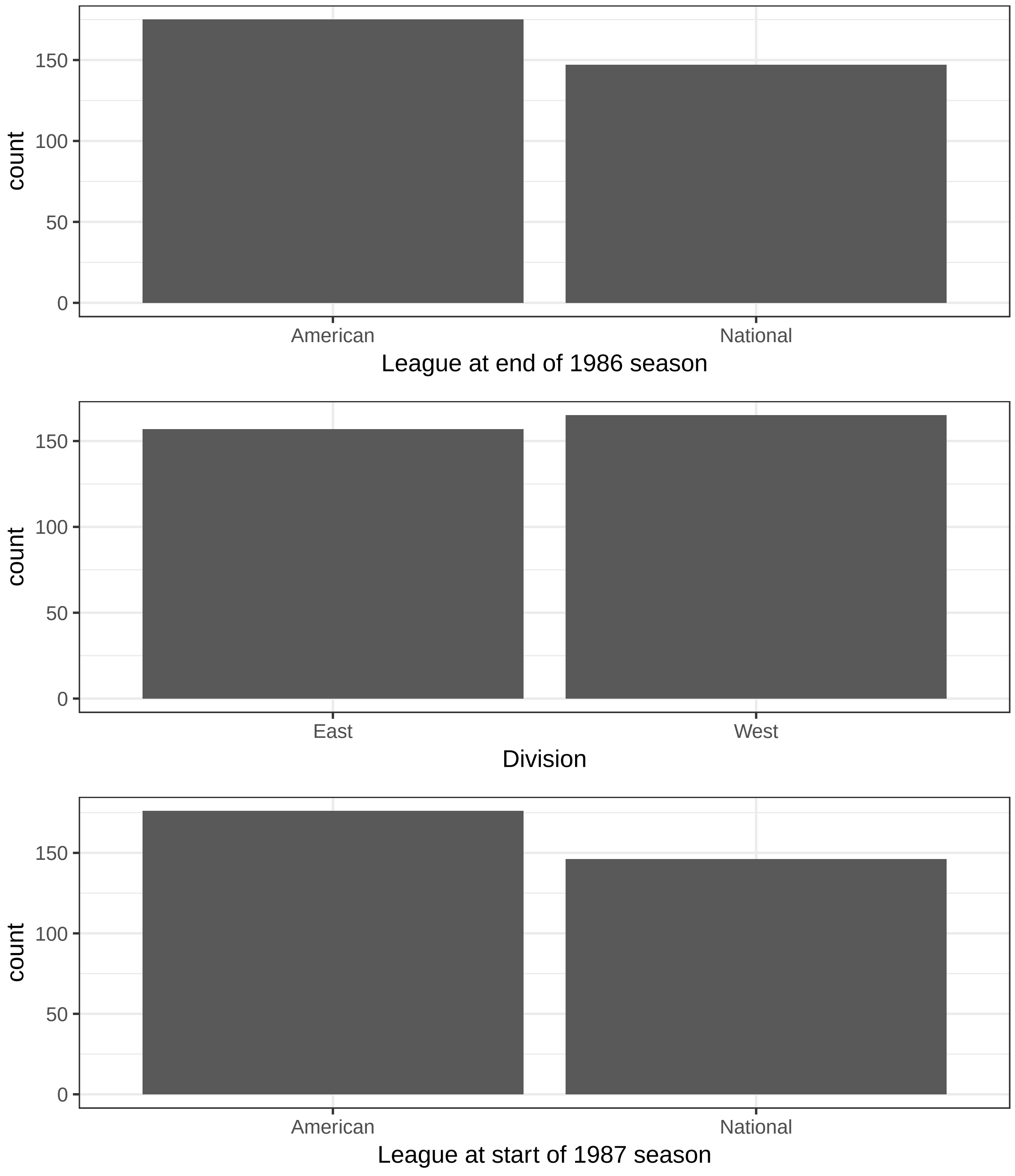

Figure 10: Bar plots of categorical variables.

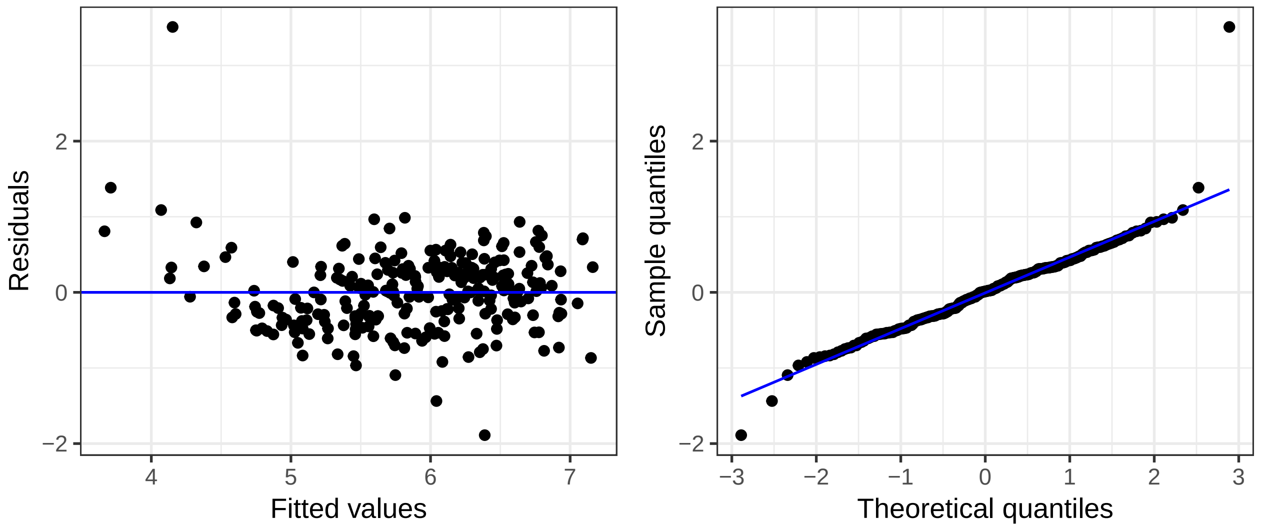

Figure 11: Diagnostic plots for the linear regression model. The left hand plot is residuals vs. fitted values. The right hand plot is a QQ plot of the residuals vs. normal quantiles. The blue lines are reference lines indicating zero (left) and the y=x line (right).

B R Code

```r
library(here)
library(ISLR)
library(MASS) # load MASS first to prevent from masking dplyr::select
library(dplyr)
library(purrr)
library(recipes)
library(corrplot)
library(ggplot2)
library(caret)
library(broom)
library(car)
library(glmnet)
library(randomForest)





# ===============================
# ========= setup ===============
# ===============================

data("Hitters")

bball_raw <- Hitters %>%
  rename(
    at_bats = AtBat,
    hits = Hits,
    home_runs = HmRun,
    runs = Runs,
    rbis = RBI,
    walks = Walks,
    years = Years,
    career_at_bats = CAtBat,
    career_hits = CHits,
    career_home_runs = CHmRun,
    career_runs = CRuns,
    career_rbis = CRBI,
    career_walks = CWalks,
    league = League,
    division = Division,
    put_outs = PutOuts,
    assists = Assists,
    errors = Errors,
    salary = Salary,
    new_league = NewLeague
  ) %>%
  as_tibble()

str(bball_raw)

# any NAs?
map_int(bball_raw, ~sum(is.na(.)))
# covariates don't have any missing values but response (salary) has 59


# ------ impute missing values of salary -------
bball <- recipe(salary ~., data = bball_raw) %>%
  step_knnimpute(salary) %>%
  prep() %>%
  bake(new_data = bball_raw)

# just a check to make sure there are no NAs
map_int(bball, ~sum(is.na(.)))


# ============================
# =========== EDA ============
# ============================

# correlation plot
bball %>%
  select_if(is.numeric) %>%
  cor() %>%
  corrplot(type = "lower",
           addCoef.col = "black",
           addCoefasPercent = TRUE,
           number.cex = 0.7,
           tl.srt = 10)


# function to produce histograms
gghist <- function(data, var, bins = 30){
  var <- enquo(var)
  ggplot(data, aes(!!var)) +
    geom_histogram(bins = bins)
}

# histograms of continuous variables
# those with a * indicate a transformation is in order
gghist(bball, salary) # *
gghist(bball, at_bats)
gghist(bball, hits)
gghist(bball, home_runs)  # *
gghist(bball, runs)
gghist(bball, rbis)
gghist(bball, walks)
gghist(bball, years, bins = 20)
gghist(bball, career_at_bats) # *
gghist(bball, career_hits) # *
gghist(bball, career_home_runs) # *
gghist(bball, career_runs) # *
gghist(bball, career_rbis)  # *
gghist(bball, career_walks, bins = 30) # *
gghist(bball, put_outs) # *
gghist(bball, assists) # *
gghist(bball, errors) # *


# categorical variables
ggplot(bball, aes(x = league)) +
  geom_bar()

ggplot(bball, aes(x = division)) +
  geom_bar()

ggplot(bball, aes(x = new_league)) +
  geom_bar()



# ==============================
# ====== TRANSFORMATIONS =======
# ==============================

# variables needing transformation:
# salary
# home_runs
# career_at_bats
# career_hits
# career_home_runs
# career_runs
# career_rbis
# career_walks
# put_outs
# assists
# errors

# we will use log transformations. but...
# to do log transformations, we need to find which variables contain zeros
# for those that do, we use a modified log transformation

map_int(bball, ~any(. == 0))

# function to perform modified log transformation
# note that this will produce warnings about NaNs being produced
#  but these can be ignored b/c the final transformation is correct
mod_log <- function(x){
  ifelse(x > 0, log(x + 0.15), -log(-x + 0.15))
}

bball_logged <- bball %>%
  mutate(
    log_salary = log(salary),
    log_career_abs = log(career_at_bats),
    log_career_hits = log(career_hits),
    log_career_runs = log(career_runs)
  ) %>%
  mutate(
    log_home_runs = mod_log(home_runs),
    log_career_hrs = mod_log(career_home_runs),
    log_career_rbis = mod_log(career_rbis),
    log_career_walks = mod_log(career_walks),
    log_put_outs = mod_log(put_outs),
    log_assists = mod_log(assists),
    log_errors = mod_log(errors)
  )

# just a check to make sure no -Inf produced b/c of zeros
map_int(bball_logged, ~any(. == -Inf))


# =======================================
# ====== POST TRANSFORMATION EDA ========
# =======================================

# --- histograms of transformed variables
# while not all of them look great, they are better than before
gghist(bball_logged, log_salary)
gghist(bball_logged, log_home_runs, bins = 20)
gghist(bball_logged, log_career_abs)
gghist(bball_logged, log_career_hits)
gghist(bball_logged, log_career_hrs)
gghist(bball_logged, log_career_runs)
gghist(bball_logged, log_career_rbis)
gghist(bball_logged, log_career_walks, bins = 20)
gghist(bball_logged, log_put_outs)
gghist(bball_logged, log_assists)
gghist(bball_logged, log_errors, bins = 20)


# select appropriate variables for modeling
bball_log_only <- bball_logged %>%
  select(log_salary,
         at_bats,
         hits,
         log_home_runs,
         runs,
         rbis,
         walks,
         years,
         league,
         division,
         new_league,
         log_put_outs,
         log_assists,
         log_errors,
         log_career_abs,
         log_career_hits,
         log_career_hrs,
         log_career_runs,
         log_career_rbis,
         log_career_walks
         )

# function to produce scatter plots
ggscatter <- function(data, var){
  var <- enquo(var)
  ggplot(data, aes(x = !!var, y = log_salary)) +
    geom_point() +
    geom_smooth(method = "lm", formula = y ~ x, se = FALSE)
}

# scatter plots of salary vs. covariate with smoothing line
# they are all approximately linear, no drastic deviations
ggscatter(bball_log_only, at_bats)
ggscatter(bball_log_only, hits)
ggscatter(bball_log_only, log_home_runs)
ggscatter(bball_log_only, runs)
ggscatter(bball_log_only, rbis)
ggscatter(bball_log_only, walks)
ggscatter(bball_log_only, years)
ggscatter(bball_log_only, log_put_outs)
ggscatter(bball_log_only, log_assists)
ggscatter(bball_log_only, log_errors)
ggscatter(bball_log_only, log_career_abs)
ggscatter(bball_log_only, log_career_hits)
ggscatter(bball_log_only, log_career_hrs)
ggscatter(bball_log_only, log_career_runs)
ggscatter(bball_log_only, log_career_rbis)
ggscatter(bball_log_only, log_career_walks)


# correlation matrix
bball_log_only %>%
  select_if(is.numeric) %>%
  cor() %>%
  corrplot(type = "lower",
           addCoef.col = "black",
           addCoefasPercent = TRUE,
           number.cex = 0.7,
           tl.srt = 10)

# =============================
# ====== MODELING SETUP =======
# =============================

# create train/test data
set.seed(42)
train_index <- createDataPartition(bball_log_only$log_salary,
                                   p = 0.8,
                                   list = FALSE)
training <- bball_log_only[train_index, ]
testing <- bball_log_only[-train_index, ]

dim(training) # 259 observations for training
dim(testing) # 63 observations for testing


model_recipe <- recipe(log_salary ~., data = training) %>%
  step_dummy(league, division, new_league) %>%
  step_center(all_predictors()) %>%
  step_scale(all_predictors())


# specify training scheme
train_control <- trainControl(method = "cv", number = 5)

# specify training parameters to try
lasso_grid <- expand.grid(alpha = 1,
                          lambda = seq(0.01, 1, length.out = 10))



# ==============================
# ===== LINEAR REGRESSION ======
# ==============================


set.seed(30)
lr_trained <- train(model_recipe,
                 data = training,
                 method = "lmStepAIC",
                 trControl = train_control,
                 trace = 0) # keep MASS::stepAIC from printing every output

lr_trained # RMSE = 0.552
lr_step <- lr_trained$finalModel # this gives the model
summary(lr_step)
# high p-values: log_put_outs (0.13)

vif(lr_step)  # at_bats = 19, hits = 17


# remove at_bats and keep hits
# b/c of the remaining variables in the model,
#   at_bats has higer correlations than hits

# new model specification with smaller # of variables
lr_small_recipe <- recipe(log_salary ~ hits +
                            log_home_runs +
                            rbis +
                            walks +
                            division +
                            log_put_outs +
                            log_assists +
                            log_career_rbis,
                          data = training) %>%
  step_dummy(division) %>%
  step_center(all_predictors()) %>%
  step_scale(all_predictors())

set.seed(30)
lr_step_small_trained <- train(lr_small_recipe,
                      data = training,
                      method = "lm",
                      trControl = train_control)
```

```r
lr_step_small_trained # RMSE = 0.530
lr_step_small <- lr_step_small_trained$finalModel

# --- diagnostics -----
summary(lr_step_small) # high p-values: log_put_outs (0.2)
#                                       & log_career_rbis (0.2)
vif(lr_step_small) # these are much better. largest is 5.9

lr_aug <- augment(lr_step_small)

# residuals vs. fitted values
ggplot(lr_aug, aes(x = .fitted, y = .resid)) +
  geom_point() +
  geom_hline(yintercept = 0, color = "blue")

# qq plot
ggplot(lr_aug, aes(sample = .resid)) +
  stat_qq(distribution = qnorm) +
  stat_qq_line(distribution = qnorm, color = "blue")

# residual plot has one troublesome point
# the qq plot looks good
# since the primary goal of this analysis is predictions,
#  we will not worry about the lone point in the residual plot


varImp(lr_step_small) # absolute value of t-statistic for each parameter

plot(varImp(lr_step_small))


# save models to save time compiling report
# these are saved locally
saveRDS(lr_trained, here("reports", "baseball", "results", "lr_trained.rds"))
saveRDS(lr_step, here("reports", "baseball", "results", "lr_step.rds"))
saveRDS(lr_step_small_trained,
        here("reports", "baseball", "results", "lr_step_small_trained.rds"))
saveRDS(lr_step_small,
        here("reports", "baseball", "results", "lr_step_small.rds"))

# ======================
# ======= LASSO =========
# ======================

set.seed(30)
# note: this gives a warning about missing values in performance metrics
# this can be ignored because we are considering RMSE
```

```r
lasso_trained <- train(model_recipe,
                      data = training,
                      method = "glmnet",
                      trControl = train_control,
                      tuneGrid = lasso_grid)

lasso_trained # RMSE = 0.536

lambda <- lasso_trained$bestTune$lambda
coef(lasso_trained$finalModel, s = lambda) # model coefficients

plot(lasso_trained)

# variable importance measures and plot
lasso_imp <- varImp(lasso_trained)$importance %>%
  mutate(
    Term = c("At bats", "Hits", "Log home runs", "Runs",
             "RBIs", "Walks", "Years", "Log put outs",
             "Log assists", "Log errors", "Log career at bats",
             "Log career hits", "Log career home runs", "Log career runs",
             "Log career RBIs", "Log career walks", "League: National",
             "Division: West", "New league: National")
  ) %>%
  filter(Overall > 0)

ggplot(lasso_imp, aes(x = reorder(Term, Overall), y = Overall)) +
  geom_point() +
  geom_segment(aes(x = Term, xend = Term, y = 0, yend = Overall)) +
  labs(x = "",
       y = "Lasso variable importance (%)") +
  coord_flip()

# save model
saveRDS(lasso_trained,
        here("reports", "baseball", "results", "lasso_trained.rds"))



# ============================
# ===== RANDOM FOREST ========
# ============================

set.seed(30)
rf_trained <- train(model_recipe,
                    data = training,
                    method = "rf",
                    trControl = train_control,
                     tuneLength = 10,
                     importance = TRUE)

rf_trained # RMSE = 0.429

plot(rf_trained)

# variable importance measures and plot
rf_imp <- varImp(rf_trained)$importance %>%
  mutate(
    Term = c("At bats", "Hits", "Log home runs", "Runs",
             "RBIs", "Walks", "Years", "Log put outs",
             "Log assists", "Log errors", "Log career at bats",
             "Log career hits", "Log career home runs", "Log career runs",
             "Log career RBIs", "Log career walks", "League: National",
             "Division: West", "New league: National")
  ) %>%
  filter(Overall > 0)

ggplot(rf_imp, aes(x = reorder(Term, Overall), y = Overall)) +
  geom_point() +
  geom_segment(aes(x = Term, xend = Term, y = 0, yend = Overall)) +
  labs(x = "",
       y = "Random forest variable importance (%)") +
  coord_flip()


# save model
saveRDS(rf_trained,
        here("reports", "baseball", "results", "rf_trained.rds"))




# ============================
# ====== TEST SET ERROR ======
# ============================


# --- linear regression ----
lr_test_pred <- predict(lr_step_small_trained, newdata = testing)

# testing RMSE = 0.687
postResample(pred = lr_test_pred, obs = testing$log_salary)


# ---- lasso ------
lasso_test_pred <- predict(lasso_trained, newdata = testing)
```

```r
# testing RMSE = 0.706
postResample(pred = lasso_test_pred, obs = testing$log_salary)


# ----- random forest -----
rf_test_pred <- predict(rf_trained, newdata = testing)

# testing RMSE = 0.460
postResample(pred = rf_test_pred, obs = testing$log_salary)
```

References

Allaire, JJ, Yihui Xie, Jonathan McPherson, Javier Luraschi, Kevin Ushey, Aron Atkins, Hadley Wickham, Joe Cheng, and Winston Chang. 2018. *Rmarkdown: Dynamic Documents for R.* https://CRAN.R-project.org/package=rmarkdown.

Breiman, Leo. 2001. "Statistical Modeling: The Two Cultures (with Comments and a Rejoinder by the Author)." *Statist. Sci.* 16 (3). The Institute of Mathematical Statistics: 199–231. doi:10.1214/ss/1009213726.

Breiman, Leo, Adele Cutler, Andy Liaw, and Matthew Wiener. 2018. *RandomForest: Breiman and Cutler's Random Forests for Classification and Regression.* https://CRAN.R-project.org/package=randomForest.

Fox, John, Sanford Weisberg, and Brad Price. 2018. *Car: Companion to Applied Regression.* https://CRAN.R-project.org/package=car.

Henry, Lionel, and Hadley Wickham. 2018. *Purrr: Functional Programming Tools.* https://CRAN.R-project.org/package=purrr.

James, Gareth, Daniela Witten, Trevor Hastie, and Rob Tibshirani. 2013. *An Introduction to Statistical Learning Wiht Applications in R.* Springer.

———. 2017. *ISLR: Data for an Introduction to Statistical Learning with Applications in R.* https://CRAN.R-project.org/package=ISLR.

Jed Wing, Max Kuhn. Contributions from, Steve Weston, Andre Williams, Chris Keefer, Allan Engelhardt, Tony Cooper, Zachary Mayer, et al. 2018. *Caret: Classification and Regression Training.* https://CRAN.R-project.org/package=caret.

Kuhn, Max, and Hadley Wickham. 2018. *Recipes: Preprocessing Tools to Create Design Matrices.* https://CRAN.R-project.org/package=recipes.

Lee, Jason D., Dennis L. Sun, Yuekai Sun, and Jonathan E. Taylor. 2016. "Exact Post-Selection Inference, with Application to the Lasso." *Ann. Statist.* 44 (3). The Institute of Mathematical Statistics: 907–27. doi:10.1214/15-AOS1371.

Mentch, Lucas, and Giles Hooker. 2016. "Quantifying Uncertainty in Random Forests via Confidence Intervals and Hypothesis Tests." *Journal of Machine Learning Research* 17 (26): 1–41. http://jmlr.org/papers/v17/14-168.html.

Müller, Kirill. 2017. *Here: A Simpler Way to Find Your Files.* https://CRAN.R-project.org/package=here.

R Core Team. 2018. *R: A Language and Environment for Statistical Computing.* Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Ripley, Brian. 2018. *MASS: Support Functions and Datasets for Venables and Ripley's Mass.* https://CRAN.R-project.org/package=MASS.

Tibshirani, Ryan J., Jonathan Taylor, Richard Lockhart, and Robert Tibshirani. 2016. "Exact Post-Selection Inference for Sequential Regression Procedures." *Journal of the American Statistical Association* 111 (514). Taylor & Francis: 600–620. doi:10.1080/01621459.2015.1108848.

Wei, Taiyun, and Viliam Simko. 2017. *Corrplot: Visualization of a Correlation Matrix.* https:

//CRAN.R-project.org/package=corrplot.

Wickham, Hadley, Winston Chang, Lionel Henry, Thomas Lin Pedersen, Kohske Takahashi, Claus Wilke, and Kara Woo. 2018. *Ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics*. https://CRAN.R-project.org/package=ggplot2.

Wickham, Hadley, Romain François, Lionel Henry, and Kirill Müller. 2018. *Dplyr: A Grammar of Data Manipulation*. https://CRAN.R-project.org/package=dplyr.

Xie, Yihui. 2018a. *Bookdown: Authoring Books and Technical Documents with R Markdown*. https://CRAN.R-project.org/package=bookdown.

———. 2018b. *Knitr: A General-Purpose Package for Dynamic Report Generation in R*. https://CRAN.R-project.org/package=knitr.

Zhu, Hao. 2018. *KableExtra: Construct Complex Table with 'Kable' and Pipe Syntax*. https://CRAN.R-project.org/package=kableExtra.