# 551 Case Study Group5 update 3

Ethan Straub, Leila Naderi, Benjamin Frazier

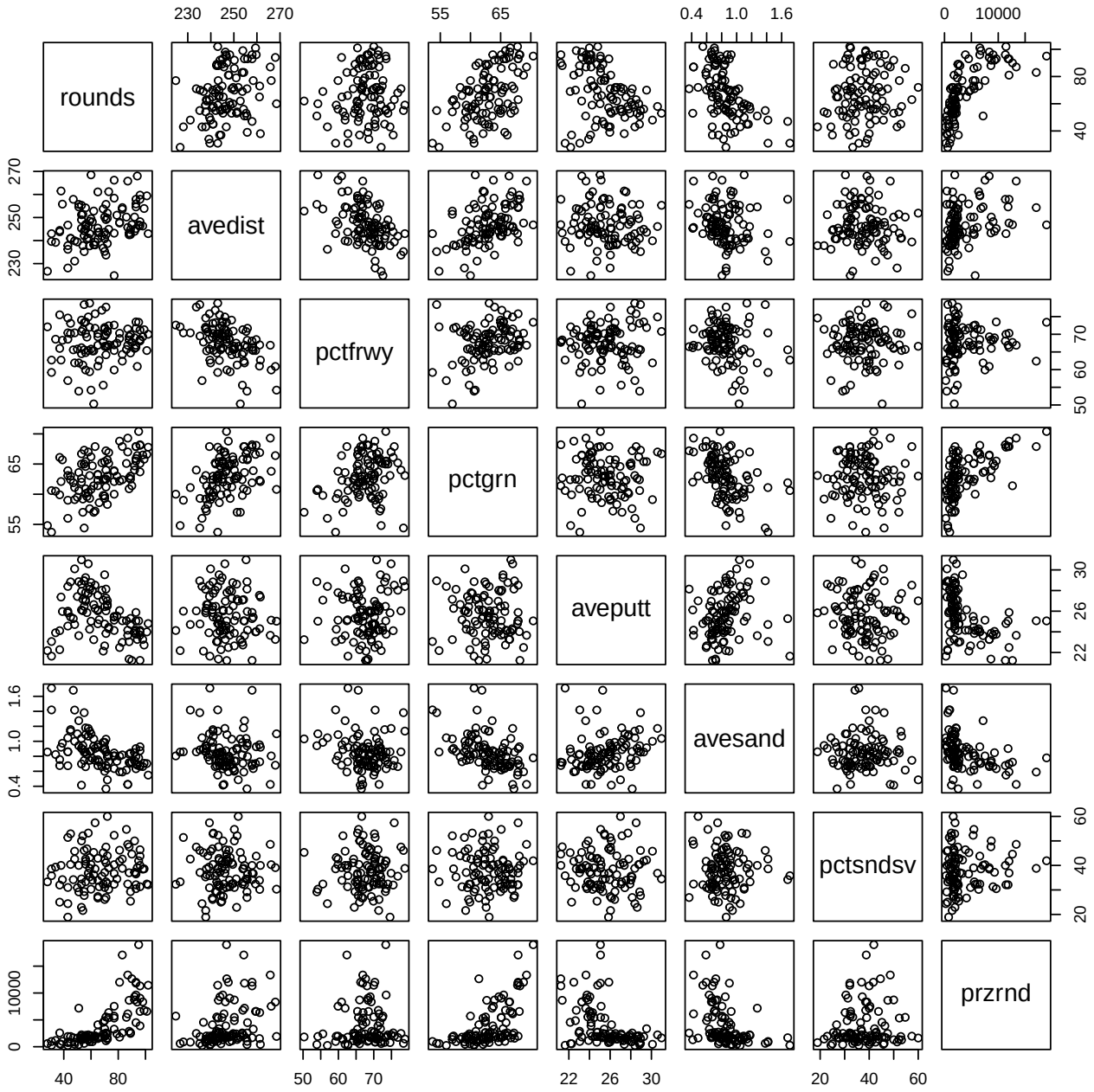
2024-09-28

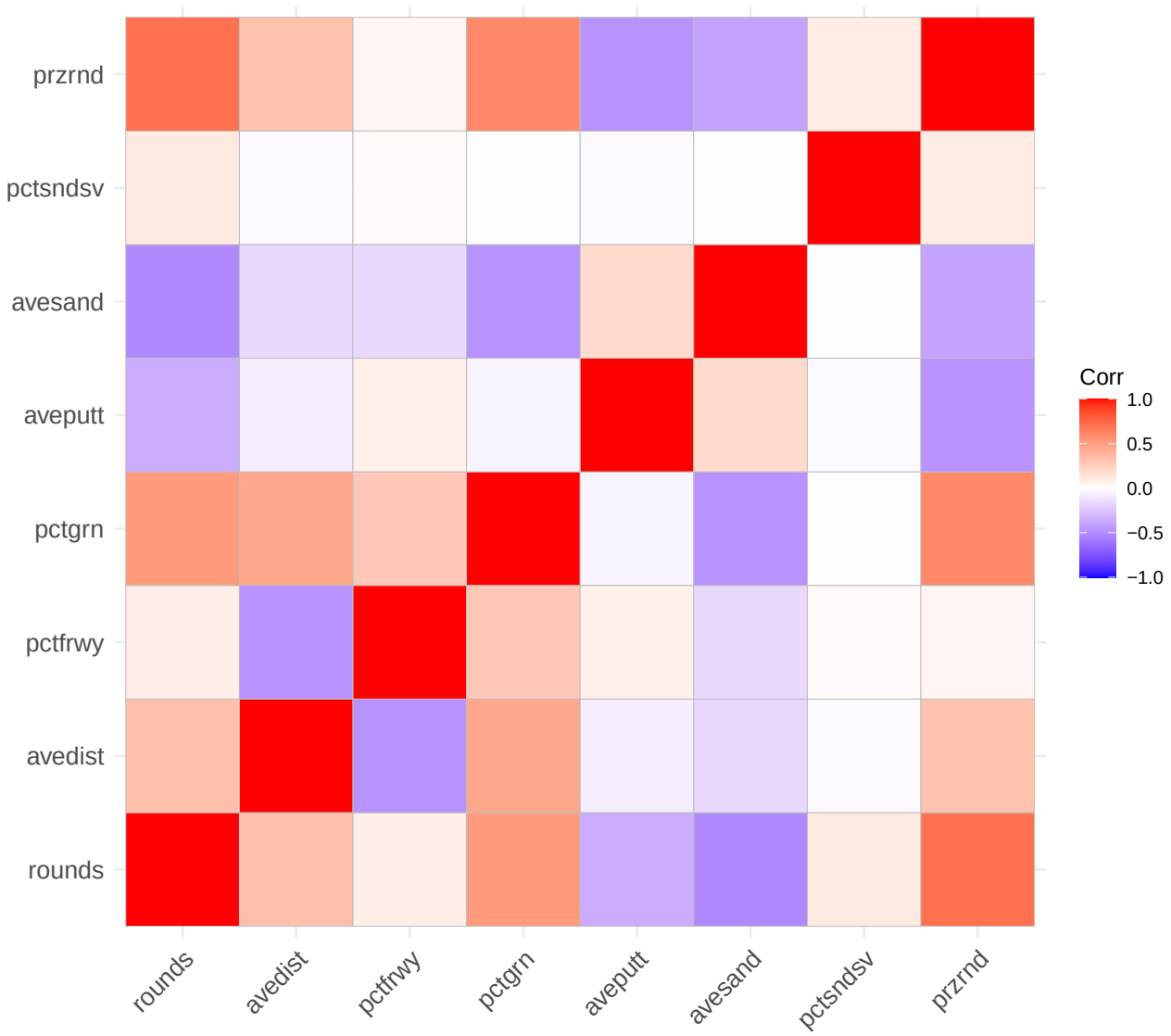
## Introduction

In professional sports, understanding the factors that contribute to success is essential for athletes, coaches, and analysts. In the context of professional golf, performance metrics such as driving distance, putting efficiency, and accuracy in reaching greens are critical in determining a player's earnings. This study focuses on golfers in the Ladies Professional Golf Association (LPGA), with the goal of constructing a model to predict the prize money earned per round (przrnd). By analyzing several key performance metrics, this study aims to identify the predictors that have the greatest influence on a golfer's earnings.

The response variable in this analysis is the prize money per round (przrnd), and several performance and experience-related predictors are considered. In this case study, all predictor variables are quantitative. The variables include the number of rounds played (rounds), average drive distance (avedist), percentage of fairways hit (pctfrwy), percentage of greens in regulation (pctgrn), average number of putts per round (aveputt), average sand shots per round (avesand), and sand save percentage (pctsndsv). The golfer's name (Golfer) is a categorical variable, but it is not used in the analysis as a predictor since it doesn't affect the outcome.

The primary objective of this study is to develop a statistical model that explains the relationship between these performance variables and prize money earned. By identifying the most significant predictors, we can provide insights into which aspects of a golfer's game contribute most to their financial success. This information can be invaluable for golfers and their coaches in making data-driven decisions about training and strategy, allowing them to focus on areas that yield the greatest return on investment. Furthermore, this model may help golf analysts and enthusiasts better understand the factors that distinguish top performers in the LPGA.


rounds
avedist
pctfrwy
pctgrn
aveputt
avesand
pctsndsv
przrnd




From the pairs plot, we can see that the distribution of the response variable (Prize money per round) is highly skewed right. It may be a good idea to log transform this variable in a model. Log transforming will also make the pr

There are no exceptionally strong relationships between variables, but there are a few noticeable trends.

Golfers who hit longer drives on average tend to hit the fairway less. This reflects the trade off between power and accuracy. It is easy to hit a ball far, and it is easy to hit a ball short with a good line. But it is difficult to hit it far and with the right line. We can create a variable that reflects a golfer's long game by multiplying these variables together after standardizing (to make the contribution from each variable the same).

Average sand shots and percent green have a decently large negative correlation. Most sand traps in golf are close to the green. So if a golfer misses a lot of greens, it is likely that they hit a sand trap instead.

Here are some other relationships in the data that make intuitive sense. A better golfer who makes more money per round will generally have longer drives, more accurate fairway and green shots, less putts, hit less sand traps, and hit good shots from the sand.

The highest correlation between two variables in the dataset is between number of rounds played and prize money per round. Golfers with better scores often survive any tournament cuts. This survivorship bias explains why they make more money per round on average combined with most tournaments awarding most

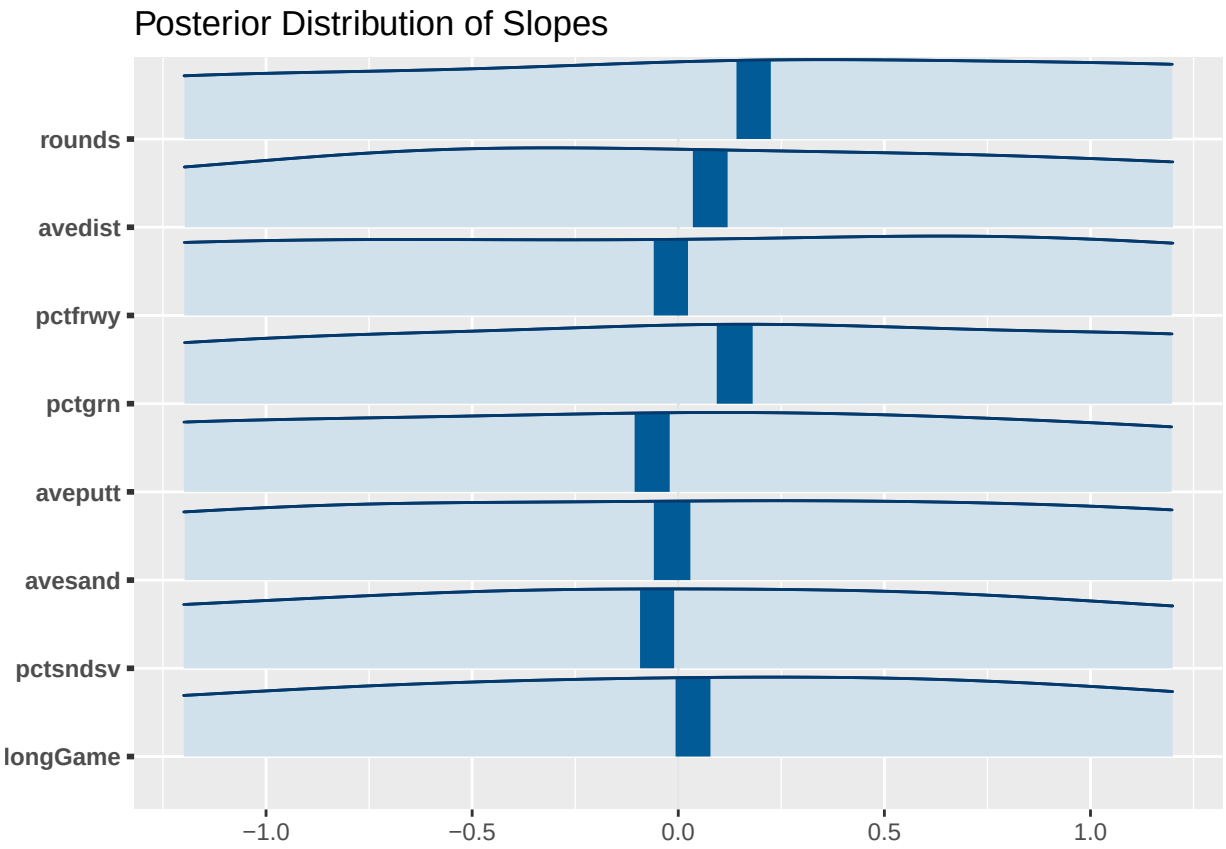
prize money to the top players (Golf channel staff, 8/2024)[1].

Variables avesand and avgputt are both negatively correlated with both rounds and przrnd, these make sense as both the lpga and pga leagues often have "cuts" in tournaments so the players with higher scores get eliminated and do less rounds thus earning less money per round. If average hits per round was included, it seems fair to say it would be highly correlated with the "winners". Both these variables are likely highly correlated with total hits per round thus are once removed proxies for total scores.

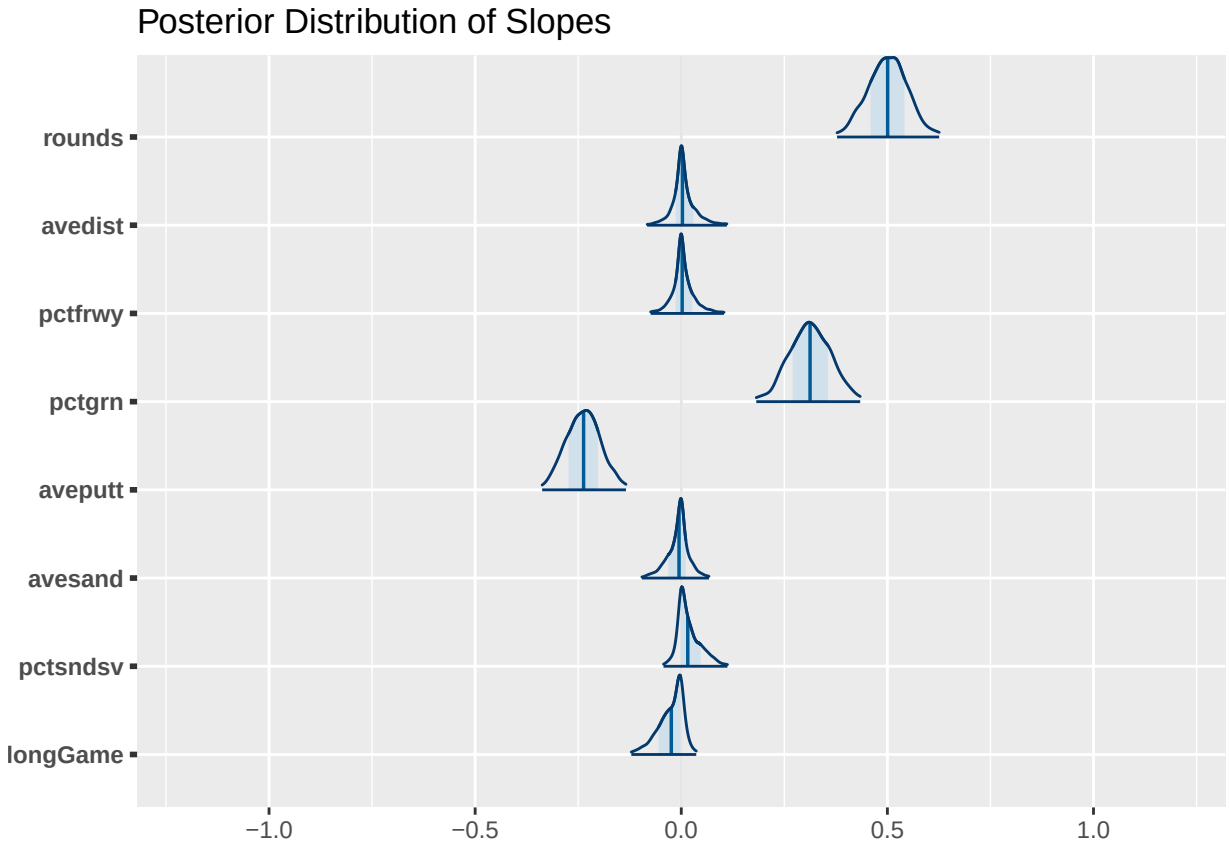
We also created a variable called roundsSquared which is rounds$^2$. Since rounds has semi strong relatiopships with many other variables, maybe including this variable in a model can increase prediction accuracy.

# Analysis

```
## Warning: Removed 8 rows containing missing values or values outside the scale range
## (`geom_segment()`).
```




When using Y on the original scale, there is clear non-linearity of residuals in the residuals vs fitted plot, and the variance of every slope estimate is too large we also investigated at posterior prediction of the non-log transformed values which show very off predictions. Log transforming Y is definitely a better option.




With a horseshoe prior using the rule of thumb parameters, the model keeps pctgrn, aveputt, and rounds and sets every other coefficient to be close to 0.

```
##         elpd_diff se_diff
## model_3  0.0       0.0
## model_1 -0.7       1.7
## model_2 -2.1       2.3
```

The above table compares estimated log scores of different models. Model 3 includes the predictors rounds, pctgrn, aveputt, and rounds$^2$ with the default normal prior. Models 1 and 2 include every predictor. Model 1 has a horseshoe prior with p0 = 4. Model 2 has a normal prior with scale = 0.25. Model 3 performed the best. The next table will examine models like model 3.

```
##         elpd_diff se_diff
## model_2  0.0       0.0
## model_1 -0.1       0.4
## model_3 -0.3       1.2
```

In this table, model 3 is the same as the previous table. Model 2 is the same as model 3 with the exception of removing the rounds$^2$ variable. These models seem to perform similarly, but the models without the rounds$^2$ variable are more interpretable.

```
## stan_glm
##  family:       gaussian [identity]
##  formula:      log(przrnd) ~ rounds + aveputt + pctgrn
##  observations: 100
##  predictors:   4
## ------
##             Median MAD_SD
## (Intercept)  7.84   0.04
```

```
## rounds       0.50   0.06
## aveputt     -0.24   0.05
## pctgrn       0.32   0.05
##
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 0.45   0.03
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

**Final Model: Log(Prize Money per Round) = 7.84 + 0.5(Standardized Number of Rounds) - 0.25(Standardized Number of Putts per Round) + 0.32(Standardized Percentage of Greens in Regulation)**

# Results and Conclusion

The Bayesian linear regression model was used to predict the log of prize money per round (`przrnd`) based on three key variables: the number of rounds played (`rounds`), the percentage of greens hit in regulation (`pctgrn`), and the average number of putts per round (`aveputt`).

$\beta_0 = 7.84$, which is the expected value of log(przrnd) where all rounds, aveputt and pctgrn are all zero.

$\beta_1 = 0.50$, where a one standard deviation increase in *rounds* is associated with approximately at 50% increase in *przrnd*.

$\beta_2$ = -0.25, where a one standard deviation increase in *aveputt* is associated with approximately at 50% decrease in *przrnd*.

$\beta_3 = 0.32$, where a one standard deviation increase in *pctgrn* is associated with approximately at 32% increase in *przrnd*.

The results indicate that both `rounds` and `pctgrn` positively influence prize money. This highlights the importance of accuracy in reaching the green within the expected number of strokes, which is a key indicator of performance.

On the other hand, `aveputt` has a negative impact on prize money This suggests that poor putting performance (requiring more putts) negatively affects a golfer's earnings, reinforcing the importance of putting efficiency in determining success. Overall, the model provides insights into how these performance metrics contribute to a golfer's financial success, with putting and greens-in-regulation being particularly influential factors. The standard deviation of the residuals (sigma = 0.5) indicates that, while the model explains a significant portion of the variance, there is still some unexplained variability in prize money, likely due to other factors not included in the model.

# References:


1 https://www.nbcsports.com/golf/news/2024-aig-womens-open-prize-money-full-purse-payout-from-st-andrews

2 Andrew Gelman, Jennifer Hill. Regression and Other Stories - Examples, avehtari.github.io/ROS-Examples/examples.html#Introduction. Accessed 11 Oct. 2024.

3 Gelman, Andrew, et al. Regression and Other Stories. Cambridge University Press, 2021.

## Appendix: All code for this report

```
knitr::opts_chunk$set(echo = FALSE)
suppressMessages(library(ggcorrplot))
suppressMessages(library(ggplot2))
suppressMessages(library(scales))
suppressMessages(library(ggrepel))
suppressMessages(library(patchwork))
suppressMessages(library(tidyverse))


df = read.csv("LPGA.csv")
# Taking names out of the design matrix
data = df[,-1]
pairs(data)

corMat = cor(data)
ggcorrplot(corMat)
przrnd = data$przrnd
dta_temp = as.data.frame(scale(data[,c(1:7)]))
dta_temp$longGame = dta_temp$avedist * dta_temp$pctfrwy
dta_temp$roundsSquared = dta_temp$rounds^2
std_designMat = as.data.frame(scale(dta_temp))

#pairs(std_designMat)

# corMat2 = cor(std_designMat)
# ggcorrplot(corMat2)

# glimpse(std_designMat)
# # can delete
#
# # Convert Data Frame to Long Format
# df_long <- data %>%
#   select_if(is.numeric) %>%  # Select only numeric columns
#   pivot_longer(cols = everything(), names_to = "Variable", values_to = "Value")
#
# # Create Histograms with Facets
# ggplot(df_long, aes(x = Value)) +
#   geom_histogram(fill = "skyblue", color = "black", bins = 30) +
#   facet_wrap(~ Variable, scales = "free") +
#   theme_minimal() +
#   labs(title = "Histograms of All Numeric Variables in mtcars",
#        x = "Value",
#        y = "Frequency")
#



# Functions

suppressMessages(library(rstanarm))
suppressMessages(library(bayesplot))
```

```r
suppressMessages(library(loo))



fit = function(Y = przrnd, logY = TRUE, X = std_designMat, prior = "normal", scale = 2.5, p0 = 4){
  fullMat = cbind(X, Y)
  if(prior == "normal"){
    if(logY == TRUE){
    fit = stan_glm(log(Y) ~ . , data = fullMat,
                 prior=normal(scale=scale), refresh = 0)
    }
    else{
    fit = stan_glm(Y ~ . , data = fullMat,
                prior=normal(scale=scale), refresh = 0)
    }
  }
  else if(prior == "horseshoe"){
    p = ncol(X)
    n = nrow(X)
    global_scale <- (p0/(p - p0))/sqrt(n)
    if(logY == TRUE){
    slab_scale <- sqrt(0.3/p0)*sd(log(fullMat$Y))
    fit <- stan_glm(log(Y) ~ ., data=fullMat, refresh=0,
    prior=hs(global_scale=global_scale,
    slab_scale=slab_scale))
    }
    else{
    slab_scale <- sqrt(0.3/p0)*sd(fullMat$Y)
    fit <- stan_glm(Y ~ ., data=fullMat, refresh=0,
    prior=hs(global_scale=global_scale,
    slab_scale=slab_scale))
    }
  }
  else{
    print("spelling error on the prior")
  }
  return(fit)
}

plots = function(logY = TRUE, X = std_designMat, coefDist = TRUE, residFit = TRUE, Y = przrnd, prior =
  fit = fit(Y, logY, X, prior, scale, p0)
  if(residFit == TRUE){
    plot(fit$fitted.values, fit$residuals, xlab = "Fitted Values",
         ylab = "Residuals", abline(h=c(-sigma(fit), 0, sigma(fit)), lwd=3,
                                    lty=c(2, 1, 2), col = "gray"), pch = 16)
  }
  if(coefDist == TRUE){
    pt <- mcmc_areas(as.matrix(fit), pars=vars(-'(Intercept)',-sigma),
                 prob_outer=0.95, area_method = "scaled height") +
    xlim(c(-1.2,1.2))
    pt <- pt + scale_y_discrete(limits = rev(levels(pt$data$parameter))) +        ggtitle("Posterior Di
    pt
```

```
  }



}

mypost_predic = function(logY = TRUE, X = std_designMat, coefDist = TRUE, residFit = TRUE, Y = przrnd, p

  fit = fit(Y, logY, X, prior, scale, p0)

    if(residFit == TRUE){
    plot(fit$fitted.values, fit$residuals, xlab = "Fitted Values",
         ylab = "Residuals", abline(h=c(-sigma(fit), 0, sigma(fit)), lwd=3,
                                    lty=c(2, 1, 2), col = "gray"), pch = 16)
  }


    if(mypost_pred == T ){

    y_rep <- posterior_predict(fit)

    post_pred_density <- ppc_dens_overlay(przrnd , y_rep[1:100, ]) + scale_y_continuous(breaks=NULL)

    plot(post_pred_density)
  }

}


compare = function(Y = przrnd, logY = c(TRUE, TRUE), X = list(std_designMat, std_designMat[,c(2:8)]), p
  numModels = length(prior)
  fit_list = vector("list", length = numModels)
  for(i in 1:numModels){
    fit = fit(Y, logY[i], X[[i]], prior[i], scale[i], p0[i])
    fit_list[[i]] = fit
  }
  objList = vector("list", length = numModels)
  if(kf == TRUE){
    for(i in 1:numModels){
      if(i == 1){
        model_1 = fit_list[[i]]
        objList[[i]] = rstanarm::kfold(model_1, K = k, scale = NULL)
      }
      if(i == 2){
        model_2 = fit_list[[i]]
        objList[[i]] = rstanarm::kfold(model_2, K = k, scale = NULL)
      }
      if(i == 3){
        model_3 = fit_list[[i]]
        objList[[i]] = rstanarm::kfold(model_3, K = k, scale = NULL)
      }
    }
```

```
  }
  else{
    for(i in 1:numModels){
      if(i == 1){
        model_1 = fit_list[[i]]
        objList[[i]] = loo(model_1)
      }
      if(i == 2){
        model_2 = fit_list[[i]]
        objList[[i]] = loo(model_2)
      }
      if(i == 3){
        model_3 = fit_list[[i]]
        objList[[i]] = loo(model_3)
      }
      if(i == 4){
        model_4 = fit_list[[i]]
        objList[[i]] = rstanarm::kfold(model_4, K = k, scale = NULL)
      }
    }
  }
  do.call(rstanarm::loo_compare, objList)
}
suppressMessages(plots(X = std_designMat[,c(1:8)], logY = FALSE, residFit = FALSE))



#suppressMessages(mypost_predic(X = std_designMat[,c(1:8)], logY = FALSE))
# suppressMessages(plots(X = std_designMat[,c(2:9)], residFit = FALSE))
suppressMessages(plots(X = std_designMat[,c(1:8)], prior = "horseshoe", residFit = FALSE))
#suppressMessages(plots(X = std_designMat[,c(1:8)], prior = "horseshoe", residFit = FALSE))
#suppressMessages(compare(X = list(std_designMat, std_designMat[,c(1:8)], std_designMat[,c(1,4,5,9)], s
suppressMessages(compare(X = list(std_designMat, std_designMat, std_designMat[,c(1,4,5,8)]), prior = c(
suppressMessages(compare(X = list(std_designMat[,c(1,4,5)], std_designMat[,c(1,4,5)], std_designMat[,c(
fullMat = cbind(przrnd, std_designMat)
model = stan_glm(log(przrnd) ~ rounds + aveputt + pctgrn, data = fullMat, refresh = 0, prior = normal(s
print(model, digits = 2)
```