Case Study

Achyuth Sai Patha and Sam McNeill

2024-10-09

Introduction

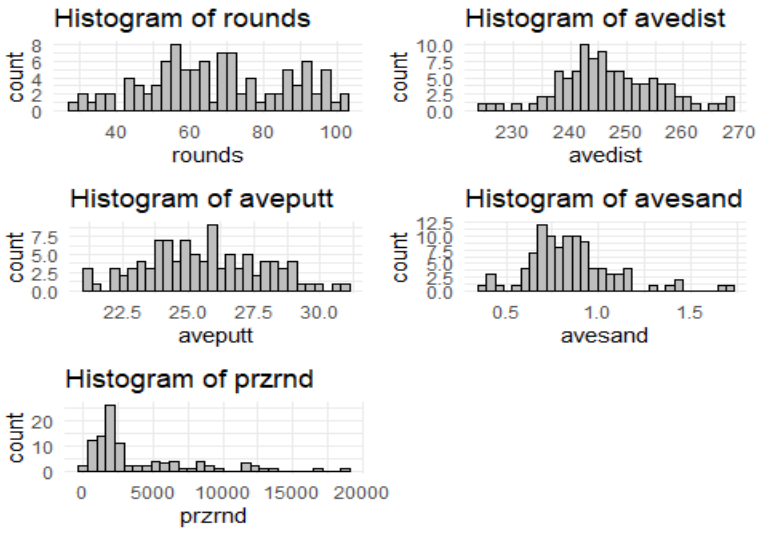
The aim of this study is to create a predictive model to determine the prize money a golfer will win per round ("przrnd") based on various performance variables from the Ladies Professional Golf Association.

- This model will be constructed using data from 100 golfers featured in the Ladies Professional Golf Association. There are nine variables included in the data set, with "przrnd" being the variable of interest.
- "przrnd": This variable represents the amount of prize money won per round of the golfer's career (in American dollars and cents)
- With "przrnd" as our response variable, that leaves 8 other possible predictor variables:
- "Golfer": This variable holds a string with the name of each of the 100 golfers. This can be thought of as a categorical variable with each name as a different level.
- "rounds": This variable holds integers for the total number of rounds each golfer has played in their career.
- "avedist": This variable represents the average distance each golfer hits the ball with a driver in yards (typically used at the start of a hole or on long holes)
- "pctfrwy": Defined as the percent of fairways hit in the golfer's career, this variable generally indicates accuracy and control from the tee.
- "pctgrn": This variable tracks the percentage a golfer's ball reaches the putting surface in par minus two strokes or better. This variable generally indicates overall play from tee to green.
- "aveputt": Average number of putts a golfer takes in an 18-hole round. Lower numbers often indicate that a golfer is more efficient at putting or chipping in the ball.
- "avesand": This variable counts the number of times, on average, a golfer hits from a sand pit. Lower numbers typically indicate the golfer is avoiding sand traps.
- "pctsandsv": This variable measures how often a golfer gets the ball into the hole in two shots or less after hitting from a sand pit. This is calculated by the following equation: (Number of successful sand saves) / (Number of sand bunker shots) x 100

- In order to balance over and under fitting, we must find the most impactful predictor variables to include in our model. This will be done by creating summary statistics and graphics, and model fitting.

Summary Statistics and Graphics

After thinking about the definition of each variable, none of them have practical negative values or zero points. The percentage variables will be left as is since they are already on an interpretable scale (0 - 100). Przrnd, rounds, avedist, aveputt, and avesand may need log transformations (if heavily skewed/have large outliers). Scaling predictors by removing the mean and dividing by the standard deviation (using "scale()") will be beneficial for interpretation and comparing effect sizes. Let's explore the non-percentage predictors.



- It appears that avesand and our przrnd are right skewed. Przrnd is heavily right skewed and should definitely be log transformed in the models as is it our response variable.

- There appears to be positive relationships between our response and predictors "rounds", "avedsit", and "pctgrn" (seen in the scatter plot matrix in the "R Code Appendix" section of this report). In golf terms, golfers that have played more rounds, hit the ball further with a driver, or have a higher percent green tend to earn more money.

- Also, there are negative relationships between the response and "aveputt" and "avesand" make sense in a golf perspective. Golfers that get the ball in the hole in fewer putts or hit less from in the sand tend to win more prize money.

- There also seems to be noticeable interactions between the predictors. These will be considered for future models.

Analysis

Model Fitting

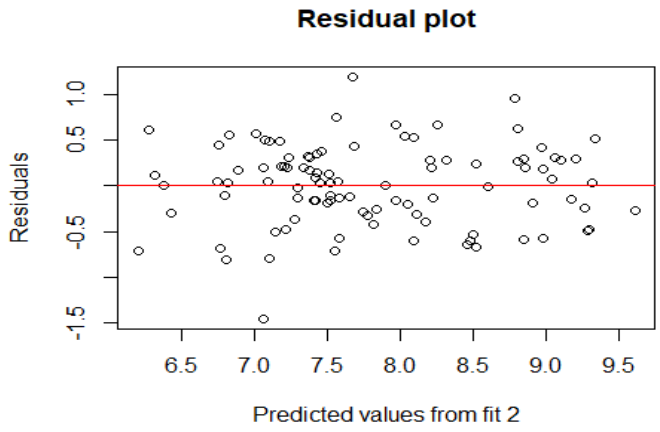
- The Golfer variable was removed from our models. First, it was causing divergent transitions in model fitting and errors in kfold comparisons. The decision to remove it from "all predictors" model makes sense since the "amount of prize money won per round" is likely not dependent on the name of the golfer. If this was the case, then the tournament is likely biased toward certain golfers and warrants further exploration. For the purposes of our analysis, we assume that the rounds of golf are fair and not biased toward any one golfer.
- The remaining predictor variables were scaled using the scale() function. Scaling allows us to compare effect sizes later.
- Models were fit using several different combinations of predictors and transformations. "Fit_all" modeled all the predictors except golfers. Fit 2 looked at log transforming the response and fit 3 looked at log transforming both the response and predictor "avesand".
- The scaled data presented problems when trying to cross validate fit 3 (which has a log transformed "avesand" predictor).
- After the scaling process, some of the values of "avesand" got scaled to be negative and the log of a negative number is undefined. So the variable had to get "unscaled", then transformed, and then rescaled.
- Fit 3 seemed to fit better, but only slightly (Log transforming avesand seemed to have a minimal effect on the fit on the model). Additionally, both fit 2 and fit 3 have similar out of sample and in sample variances (R^2).
- To aid in interpretation, fit 2 will move on and be used for comparing subsequent models.

```
fit_2 <- stan_glm(data = data_scaled, log(przrnd) ~ rounds + avedist + pctgrn + aveputt + avesand, refresh = 0)
```

- Next, prior distributions were tested for model fit. A horseshoe prior produced "divergent transitions", which makes the model less believable / trustworthy. This model was discarded.
- A normal prior was then tested and compared to fit 2 since normal priors tend to be more resistant to overfitting since it biases the coefficients toward 0. A model based on a prior would be relatively powerful at prediction since the model can be used to make direct probability statements about parameters. The normal prior assumes

that the coefficients of the predictors follow a normal distribution. Unfortunately, we do not have enough domain, or statistical, knowledge to confidently assume this.

- Since we can't confidently assume a normal prior in this case, fit 2 will move on again.
- Finally, a model using the interaction between rounds and avedist was fit. These two variables were chosen since they seemed to have the most probable relationship between themselves, and with the response.
- This model did not fit the data noticeably better than fit 2. Furthermore, interactions greatly decrease interpretation, so this model was discarded.
- After comparing different models, it appears that fit 2 results in the best fitting model while preserving interpretation.



- A residuals plot for this model shows no obvious patterns or relationships between the predicted values and the residuals. Also, there is random scattering around the center line with no obvious outliers, and most residuals are between -1 and 1.
- Based on residuals plot and model comparisons, this linear model appears to fit the data well and meet all assumptions.

Results and Conclusions

Final model and fitted equation

The final model is log(przrnd) = 7.835 + 0.512 (rounds) + 0.009 (avedist) + 0.321 (pctgrn) - 0.249 (aveputt) - 0.00119 (avesand) + 0.457 (epsilon)

Coefficients

- The coefficients represents the expected change in log of prize money won per round for a one standard deviation increase in each predictor while holding other

variables constant. Since the predictors were scaled before being fit, the coefficients are interpreted as standard deviations, and not their original units.

Intercept

Estimate: 7.837 (or exp(7.837) = $2532.60)

- This number reflects the log of prize money won when all the predictor variables are held at zero. While it isn't usually practical to interpret this in real-world terms, it does provides a foundational reference point for the model. Instead, we can exponentiate the log intercept to find the actual expected amount of prize money won when all predictor variables are held at zero.

Standard error: 0.0452

- The intercept varies by 0.0452 standard errors of log prize money per round. This is saying that that the true value of log prize money varies by 0.0452 standard errors.

95% CI: Between 7.747 and 7.926 (or $2314.62 and $2768.33)

- We are 95% confident that the true value of the intercept of log of prize money is between 7.747 and 7.926 (or $2314.62 and $2768.33)

Rounds

Estimate: 0.512 (or ~67% increase in response)

- For every one standard deviation increase in the amount of rounds played, there is a 0.512 increase in log prize money won per round when all other variables are held constant. This equates to a ~67% increase in prize money won per round of every standard deviation increase in amount of rounds played. The standard deviation of rounds played is 18.797. This means for every 18.797 rounds played, there is an expected increase of 67% in prize money won per round.

95% CI: Between 0.4127 and 0.6158

- We are 95% confident the true effect of a one standard deviation increase in rounds played lies between a 0.4127 and 0.6158 increase in log prize money won per round.

Average Distance (avedist)

Estimate: 0.0083(or ~0.87% increase in response)

- For each additional standard deviation increase in average distance, the log of prize money per round increases by an estimated 0.0083 units when all other variables are held constant. When translated to percent increase, this is a 0.87% increase in prize money won per round for every standard deviation increase in average distance. The standard deviation of average distance is 8.64348. This means for

every addition ~8.643 yards the golfer hits the ball with a driver, they are expected to gain 0.87% in prize money per round.

95% CI: Between -0.0785 and 0.0923

- We can not be 95% confident the true effect of a one standard deviation increase in average distance lies between -0.0785 and 0.0923. Since 0 is included in this credible interval, we can not be confident that there is a non-zero effect (of the standard deviation of average distance) on the log of prize money won per round.

Percentage of Greens in Regulation (pctgrn)

Estimate: 0.321

- Each standard deviation increase in percentage of greens in regulation correlates with an increase of about 0.321 in the log of prize money, when all other varibles are held constant. When translated to percent increase, this is a 37.84% increase in prize money won per round for every standard deviation increase in percentage of greens in regulation. The standard deviation of percentage of greens in regulation is 3.4819. This means for every additional ~3.5% that the golfer's ball reaches the putting surface in par minus two strokes or better, they are expected to gain 37.84% in prize money per round.

95% CI: Between 0.2012 and 0.4375

- We are 95% confident the true effect of a one standard deviation increase in percentage of greens in regulation lies between 0.2012 and 0.4375.

Average Putts (aveputt)

Estimate: -0.2493

- For every additional standard deviation of average number of putts per round, the log of prize money is expected to decrease by approximately 0.2493 units, when all other varibles are held constant. When translated to percent increase, this is a 22.069% decrease in prize money won per round for every standard deviation increase in average number of putts per round. The standard deviation of average number of putts per round is 2.2082. This means for every additional ~2.2 average putts per round, the golfer is expected to lose 22.069% in prize money per round.

95% CI: Between -0.3477 and -0.1513

- We are 95% confident the true effect of a one standard deviation increase of average number of putts per round lies between -0.3477 and -0.1513.

Average Sand Shots (avesand)

Estimate: -0.00016

- For every additional standard deviation of average number of hits from a sand pit, the log of prize money is expected to decrease by approximately 0.00016 units, when all other varibles are held constant. When translated to percent increase, this is a 0.0159% decrease in prize money won per round for every standard deviation increase in average number of hits from a sand pit. The standard deviation of average number of putts per round is 0.236. This means for every additional ~0.236 average hits from a sand pit, the golfer is expected to lose 0.0159% in prize money per round.

95% CI: Between -0.1105 and 0.1127

- We can not be 95% confident the true effect of a one standard deviation increase in average number of hits from a sand pit lies between -0.1105 and 0.1127. Since 0 is included in this credible interval, we can not be confident that there is a non-zero effect (of standard deviation of average number of hits from a sand pit) on the log of prize money won per round.

Sigma (epsilon or error)

Estimate: 0.457

- There is a residual error of 0.457, which states that the model predictions of log prize money per round differ from the actual values of the data by an expected value of 0.457. In the original scale, this suggests the model's predictions of prize money per round typically differ from the actual values by about 57.93% in either direction.

95% CI: Between 0.3991 and 0.5344

- We are 95% confident the true residual error of the model lies between 0.3991 and 0.5344.

— The goal of this study was to create a predictive model to determine the amount of prize money a golfer will win per round ("przrnd") based on various performance variables from the Ladies Professional Golf Association. A model was constructed based on initial observations of the data and response-predictor relationships. The data was transformed and models were fit in search of a balanced model that best described the data without losing interpretive power. The resulting model used 5 predictors and a log transformed response variable.

After analyzing the coefficients and credible intervals, we suggest finding a better fitting model to accurately predict the amount of prize money won per round. Confidence in this model is diminished after finding out that two of the predictor variables may not have an effect on the log response variable. Furthermore, the residual error of the model is inadequate when trying to accurately predict prize money per round. On average, predictions could be up to 57.93% higher or lower than the actual values, so we recommend using this model as a baseline when searching for a model with more predictive power.

R Code Appendix

```r
library(car)
```

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

```r
library(ggplot2)
library(gridExtra)
library(rstanarm)
```

```
## Loading required package: Rcpp

## This is rstanarm version 2.32.1

## - See https://mc-stan.org/rstanarm/articles/priors for changes to default
priors!

## - Default priors may change, so it's safest to specify priors, even if
equivalent to the defaults.

## - For execution on a local, multicore CPU with excess RAM we recommend
calling

##   options(mc.cores = parallel::detectCores())

##
## Attaching package: 'rstanarm'

## The following object is masked from 'package:car':
##
##     logit
```

```r
sum(is.na(data)) # No NAs
```

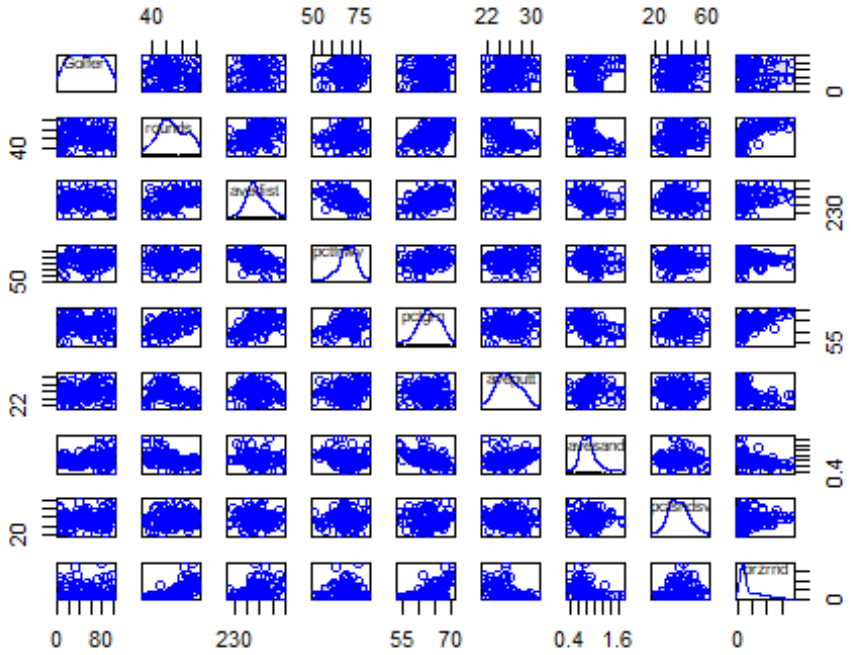
```
## [1] 0
```

```r
str(data) # One character variable (Golfer)
```

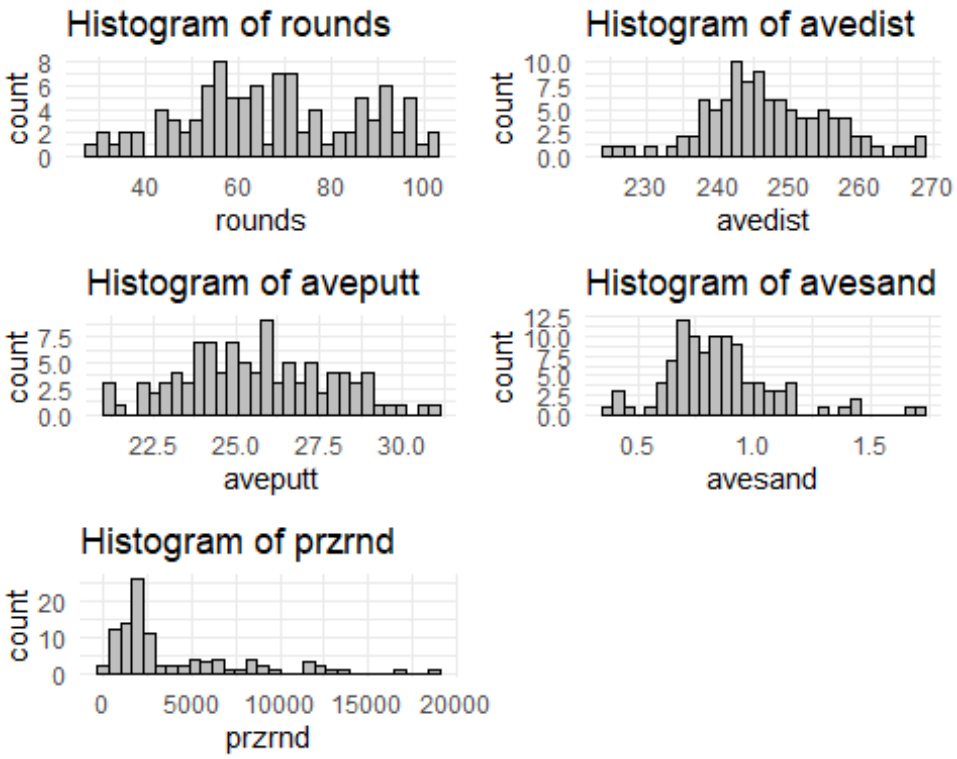
```
## 'data.frame':    100 obs. of  9 variables:
##  $ Golfer  : chr  "Audra Burks" "Nicole Perrot" "Nicole Castrale" "Lisa
Strom" ...
##  $ rounds  : int  34 31 87 38 93 31 79 77 93 96 ...
##  $ avedist : num  239 243 246 262 256 ...
##  $ pctfrwy : num  68.6 59.2 71.5 62.7 71.7 62.7 67.5 67 66.9 71.2 ...
##  $ pctgrn  : num  60.5 53.7 67 62.1 65.9 60.6 63 62.1 67.9 63.7 ...
##  $ aveputt : num  23.9 23.1 24.2 27.4 24.2 ...
##  $ avesand : num  0.824 1.419 0.425 0.921 0.796 ...
##  $ pctsndsv: num  39.3 38.6 32.4 40 36.5 35.8 34.5 52.3 32.1 46.4 ...
##  $ przrnd  : int  1160 979 6116 441 4490 272 2383 5335 11831 3889 ...
```

```r
# "Golfer" is a vector of characters, so we need to convert to factors to
create plots
data$Golfer <- as.factor(data$Golfer)
```

```
# Initial glance at relationships between variables
scatterplotMatrix(data)
```




```
# Check skewness (see if log transformations would be useful)
plots <- list()

variables <- c("rounds", "avedist", "aveputt", "avesand", "przrnd")

for (var in variables) {
  p <- ggplot(data, aes_string(x = var)) +
    geom_histogram(bins = 30, fill = "gray", color = "black") +
    ggtitle(paste("Histogram of", var)) +
    xlab(var) +
    theme_minimal()

  plots[[var]] <- p
}

## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## ℹ Please use tidy evaluation idioms with `aes()`.
## ℹ See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

grid.arrange(grobs = plots, ncol = 2)
```




```r
# Remove golfer from data set (explained in analysis)
data_no_golfer <- data
data_no_golfer$Golfer <- NULL

# Scale predictors so they are all on the same scale (so they can be compared
later)
predictors <- c("rounds", "avedist", "pctgrn", "aveputt", "avesand",
"pctsndsv")
data_scaled <- data_no_golfer
data_scaled[,predictors] <- scale(data_no_golfer[,predictors])
```

```r
# GLM with all predictors except Golfer name (explained below)
fit_all <- stan_glm(data = data_scaled, przrnd ~ ., refresh = 0)

# GLM with 5 predictors mentioned in summary stats with no interactions or
transformations
fit_1 <- stan_glm(data = data_scaled, przrnd ~ rounds + avedist + pctgrn +
aveputt + avesand, refresh = 0)

# GLM with 5 predictors and log transformations of response
fit_2 <- stan_glm(data = data_scaled, log(przrnd) ~ rounds + avedist + pctgrn
+ aveputt + avesand, refresh = 0)

# GLM with 5 predictors and log transformations of response and avesand
# (Need to log transform avesand first then scale)
```

```r
data_logavesand <- data_scaled

data_logavesand$avesand_unscaled <- data_scaled$avesand *
sd(data_no_golfer$avesand) + mean(data_no_golfer$avesand)

data_logavesand$log_avesand <- log(data_logavesand$avesand_unscaled)

data_logavesand$scaled_log_avesand <- scale(data_logavesand$log_avesand)

fit_3 <- stan_glm(data = data_logavesand, log(przrnd) ~ rounds + avedist +
pctgrn + aveputt + scaled_log_avesand, refresh = 0)


# Comparing first 4 models
kfold_all <- kfold(fit_all, k = 10)

## Fitting model 1 out of 10

## Fitting model 2 out of 10

## Fitting model 3 out of 10

## Fitting model 4 out of 10

## Fitting model 5 out of 10

## Fitting model 6 out of 10

## Fitting model 7 out of 10

## Fitting model 8 out of 10

## Fitting model 9 out of 10

## Fitting model 10 out of 10

kfold1 <- kfold(fit_1, k = 10)

## Fitting model 1 out of 10

## Fitting model 2 out of 10

## Fitting model 3 out of 10

## Fitting model 4 out of 10

## Fitting model 5 out of 10

## Fitting model 6 out of 10

## Fitting model 7 out of 10

## Fitting model 8 out of 10
```

```
## Fitting model 9 out of 10

## Fitting model 10 out of 10

kfold2 <- kfold(fit_2, k = 10)

## Fitting model 1 out of 10

## Fitting model 2 out of 10

## Fitting model 3 out of 10

## Fitting model 4 out of 10

## Fitting model 5 out of 10

## Fitting model 6 out of 10

## Fitting model 7 out of 10

## Fitting model 8 out of 10

## Fitting model 9 out of 10

## Fitting model 10 out of 10

kfold3 <- kfold(fit_3, k = 10)

## Fitting model 1 out of 10

## Fitting model 2 out of 10

## Fitting model 3 out of 10

## Fitting model 4 out of 10

## Fitting model 5 out of 10

## Fitting model 6 out of 10

## Fitting model 7 out of 10

## Fitting model 8 out of 10

## Fitting model 9 out of 10

## Fitting model 10 out of 10

# Subtract log(prszrnd) from fit 2 for comparison
kfold2_adj <- kfold2
kfold2_adj$pointwise[,1] <- kfold2$pointwise[,1] - log(data_scaled$przrnd)

kfold3_adj <- kfold3
kfold3_adj$pointwise[,1] <- kfold3$pointwise[,1] - log(data_scaled$przrnd)
```

```r
loo_compare(kfold_all,kfold1,kfold2_adj,kfold3_adj)
```

```
## Warning: Not all models have the same y variable. ('yhash' attributes do
not
## match)
```

```
##         elpd_diff se_diff
## fit_3     0.0       0.0
## fit_2    -1.2       1.5
## fit_1   -72.6      12.3
## fit_all -73.3      11.9
```

```r
# Testing R^2 for fit 2 and 3
round(median(bayes_R2(fit_2)), 2)
```

```
## [1] 0.77
```

```r
round(median(loo_R2(fit_2)), 2)
```

```
## Warning: Some Pareto k diagnostic values are too high. See help('pareto-k-
diagnostic') for details.
```

```
## [1] 0.75
```

```r
round(median(bayes_R2(fit_3)), 2)
```

```
## [1] 0.77
```

```r
round(median(loo_R2(fit_3)), 2)
```

```
## Warning: Some Pareto k diagnostic values are too high. See help('pareto-k-
diagnostic') for details.
```

```
## [1] 0.75
```

```r
# Trying a selection prior
p <- ncol(data_scaled) - 1
n <- nrow(data_scaled)
p0 <- 3
slab_scale <- sqrt(0.3/p0)*sd(data_scaled$przrnd)
global_scale <- (p0/(p - p0))/sqrt(n)
fit_hs <- stan_glm(data = data_scaled, log(przrnd) ~ ., refresh = 0,
  prior=hs(global_scale=global_scale,
  slab_scale=slab_scale))
```

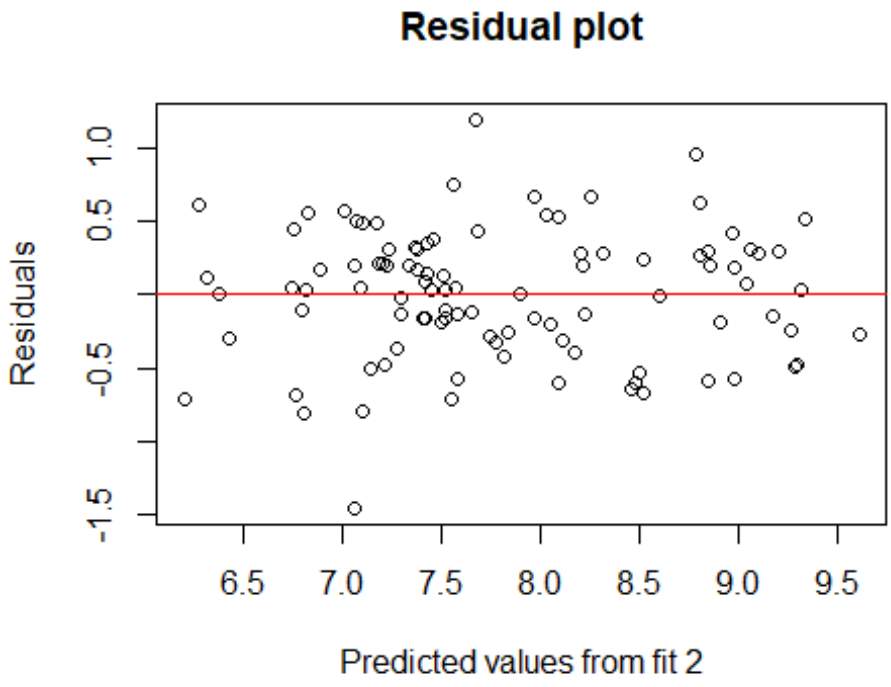
```
## Warning: There were 12 divergent transitions after warmup. See
## https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
## to find out why this is a problem and how to eliminate them.
```

```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

```r
# Horseshoe prior had "divergent transitions", so trying normal prior
fit_normal <- stan_glm(data = data_scaled, log(przrnd) ~ .,
                       prior = normal(location = 0, scale = 0.25),
                       refresh = 0)

kfold_normal <- kfold(fit_normal, k = 10)

## Fitting model 1 out of 10

## Fitting model 2 out of 10

## Fitting model 3 out of 10

## Fitting model 4 out of 10

## Fitting model 5 out of 10

## Fitting model 6 out of 10

## Fitting model 7 out of 10

## Fitting model 8 out of 10

## Fitting model 9 out of 10

## Fitting model 10 out of 10

loo_compare(kfold2,kfold_normal)

##            elpd_diff se_diff
## fit_normal  0.0       0.0
## fit_2      -1.0       1.3

# Try model with interaction between rounds and avgdist
fit_5 <- stan_glm(data = data_scaled, log(przrnd) ~ rounds + avedist + pctgrn
+ aveputt + avesand + rounds:avedist, refresh = 0)

kfold5 <- kfold(fit_5, k = 10)

## Fitting model 1 out of 10

## Fitting model 2 out of 10

## Fitting model 3 out of 10

## Fitting model 4 out of 10

## Fitting model 5 out of 10

## Fitting model 6 out of 10

## Fitting model 7 out of 10

## Fitting model 8 out of 10
```

```
## Fitting model 9 out of 10

## Fitting model 10 out of 10

loo_compare(kfold2,kfold5)

##       elpd_diff se_diff
## fit_2  0.0       0.0
## fit_5 -0.7       1.5

# Inspect and plot fit_2 residuals
fit_2

## stan_glm
##  family:       gaussian [identity]
##  formula:      log(przrnd) ~ rounds + avedist + pctgrn + aveputt + avesand
##  observations: 100
##  predictors:   6
## ------
##             Median MAD_SD
## (Intercept)  7.8    0.0
## rounds       0.5    0.1
## avedist      0.0    0.1
## pctgrn       0.3    0.1
## aveputt     -0.3    0.1
## avesand      0.0    0.1
##
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 0.5    0.0
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg

coef(fit_2)

##  (Intercept)       rounds      avedist       pctgrn      aveputt
avesand
##  7.835631570  0.513005426  0.008736354  0.319915012 -0.250628531 -
0.001551548

round(median(bayes_R2(fit_2)), 2)

## [1] 0.77

round(median(loo_R2(fit_2)), 2)

## Warning: Some Pareto k diagnostic values are too high. See help('pareto-k-
diagnostic') for details.

## [1] 0.75
```

```r
plot(fitted(fit_2), resid(fit_2),
 xlab = "Predicted values from fit 2",
 ylab = "Residuals",
 main = "Residual plot")
abline(h=0, col="red")
```




```r
# final model
model <- fit_2

#Extract coefficient
model_coef <- coef(fit_2)

# Final model equation
cat(paste("The final model is log(przrnd)
=",round(model_coef[1],3),"+",round(model_coef[2],3),"(rounds)
+",round(model_coef[3],3),"(avedist) + ",round(model_coef[4],3), "(pctgrn)
+",round(model_coef[5],3),"(aveputt) +",round(model_coef[6],5),"(avesand)
+",round(sigma(fit_2)[1],3),"(epsilon)"))

## The final model is log(przrnd) = 7.836 + 0.513 (rounds) + 0.009 (avedist)
+  0.32 (pctgrn) + -0.251 (aveputt) + -0.00155 (avesand) + 0.458 (epsilon)

# Credible intervals
CIs <- posterior_interval(fit_2, prob = 0.95)

# Standard errors of model
se <- se(fit_2)
```

```r
# Standard deviations of original data
sd <- apply(data_no_golfer, 2, sd)

# Transformed coeffs for interpretation to percent response
perc_coef <- (exp(model_coef)-1) * 100
```