Exploring and Analyzing Hockey Scoring Sequences

DAR Assignment 2 (Fall 2023)

Your Name Here

2023-09-18

Assignment 2 Notebook Overview

This notebook is broken into two main parts:

```
* Part 1 is a guide to pulling the Hockey Analysis repo from github
* Part 2 is an introduction to the RPI Hockey motion capture data
```

This R Notebook and its related R scripts provide a very basic introduction to the RPI Hockey Motion Capture dataset. This data will be used by the **Hockey Analytics** group during DAR F23.

The RPI github repository for all the code required for this notebook may be found at:

- https://github.rpi.edu/DataINCITE/Hockey_Fall_2023

TBD: The `Hockey_Fall_2023` github also contains topical notebooks which provide tutorials regarding a number of different research-worthy starting points. Feel free to examine these notebooks.

DAR ASSIGNMENT 2: CLONING A NOTEBOOK AND UPDATING THE REPOSITORY

In this assignment we're asking you to...

- clone the `Hockey_Fall_2023` github repository...
- create a personal branch using git, and...
- make additions to the repository by creating a new, customized notebook.

The instructions which follow explain how to accomplish this.

Cloning an RPI github repository

The recommended procedure for cloning and using this repository is as follows:

- Access the RPI network via VPN
  - See https://itssc.rpi.edu/hc/en-us/articles/360008783172-VPN-Connection-and-Installation for information
- Access RStudio Server on the IDEA Cluster at http://lp01.idea.rpi.edu/rstudio-ose/
  - You must be on the RPI VPN!!
- Access the Linux shell on the IDEA Cluster by clicking the **Terminal** tab of RStudio Server (lower left panel).
  - You now see the Linux shell on the IDEA Cluster
  - `cd` (change directory) to enter your home directory using: `cd ~`
  - Type `pwd` to confirm
  - NOTE: Advanced users may use `ssh` to directly access the Linux shell from a macOS or Linux command line

- Type `git clone https://github.rpi.edu/DataINCITE/Hockey_Fall_2023` from within your `home` directory
  - This will create a new directory `Hockey_Fall_2023`
- In the Linux shell, `cd` to `Hockey_Fall_2023/StudentNotebooks/Assignment01`
  - Type `ls -al` to list the current contents
  - Don't be surprised if you see many files!
- In the Linux shell, type `git checkout -b dar-yourrcs` where `yourrcs` is your RCS id
  - For example, if your RCS is `erickj4`, your new branch should be `dar-erickj4`
  - It is *critical* that you include your RCS id in your branch id
- Now in the RStudio Server UI, navigate to the `Hockey_Fall_2023/StudentNotebooks/Assignment01` directory via the **Files** panel (lower right panel)
  - Under the **More** menu, set this to be your R working directory
  - Setting the correct working directory is essential for interactive R use!

REQUIRED FOR ASSIGMENT 2

1. In RStudio, make a **copy** of `darf23-assignment2-template.Rmd` file using a *new, original, descriptive* filename that **includes your RCS ID!**
   - Open `darf23-assignment2-template.Rmd`
   - **Save As...** using a new filename that includes your RCS ID
   - Example filename for user `erickj4`: `erickj4-assignment1-f23.Rmd`
   - POINTS OFF IF:
     - You don't create a new filename!
     - You don't include your RCS ID!
     - You include `template` in your new filename!
2. Edit your new notebook using RStudio and save
   - Change the `title:` and `subtitle:` headers (at the top of the file)
   - Change the `author:`
   - Don't bother changing the `date:`; it should update automagically...
   - **Save** your changes
3. Use the RStudio `Knit` command to create an HTML file; repeat as necessary
   - Use the down arrow next to the word `Knit` and select **Knit to HTML**
   - You may also knit to PDF...
4. In the Linux terminal, use `git add` to add each new file you want to add to the repository
   - Type: `git add yourfilename.Rmd`
   - Type: `git add yourfilename.html` (created when you knitted)
   - Add your PDF if you also created one...
5. When you're ready, in Linux commit your changes:
   - Type: `git commit -m "some comment"` where "some comment" is a useful comment describing your changes
   - This commits your changes to your local repo, and sets the stage for your next operation.
6. Finally, push your commits to the RPI github repo
   - Type: `git push origin dar-yourrcs` (where `dar-yourrcs` is the branch you've been working in)
   - Your changes are now safely on the RPI github.
7. **REQUIRED:** On the RPI github, submit a pull request.
   - In a web browser, navigate to https://github.rpi.edu/DataINCITE/Hockey_Fall_2023
   - In the branch selector drop-down (by default says **master**), select your branch
   - **Submit a pull request for your branch**
   - One of the DAR instructors will merge your branch, and your new files will be added to the master branch of the repo.

Please also see these handy github "cheatsheets":

- https://education.github.com/git-cheat-sheet-education.pdf

Setup: Define functions

The major functions of this notebook are contained in the file `AnalysisCodeFunc.R`. This code chunk *sources* (imports) a helper script that defines various functions used through this notebook for data processing and analysis.

Setting program parameters

This section sets the dimensions of the data structures used in this notebook, based on the captures video.

```
# Size of rink image and of all plots
xsize <- 2000
ysize <- 850

# FPS of the video
fps <- 29.97

# Coordinates to the goal pipes
pipes_x <- 1890
lpipe_y <- 395
rpipe_y <- 455
```

Data preparation

Based on our settings, this section reads in the captured image data.

The code is highly dependent upon the following directory structure:

- The file path determined by the filepath variable contains folders named with the game number, followed by 'p', followed by the period number
- Each "period" folder contains a folder named `Sequences`.
- Each `ThatSequencesfolder contains sequence folders, sequence_folders`, that contain all the relevant sequence data.

```
# Set filepaths and read the rink

# This file path should contain the hockey rink images and all the sequences
filepath <- '../FinalGoalShots/'

# See above for explanation of file path syntax
games <- c(24, 27, 33, 34)
# Only take the first and third periods. These are when the opposing team shoots on our goal. Our shots
periods <- map(games, ~ str_c(., 'p', c(1, 3))) %>% unlist

# Get the 'Sequences' folder for every period
period_folders <- map(periods, ~ {
  str_c(filepath, ., '/Sequences')
})

# Get every folder inside each 'Sequences' folder
sequence_folders <- period_folders %>%
  map(~ str_c(., '/', list.files(.))) %>%
  unlist

# Read the rink images and format them to a raster used for graphing
```

```r
rink_raster <- makeRaster(filepath, 'Rink_Template.jpeg')
half_rink_raster <- makeRaster(filepath, 'Half_Rink_Template.jpeg')

# As every folder is run through the `combinePasses` function, the info.csv file in each sequence folde
info <- matrix(0, nrow = 0, ncol = 4) %>%
  data.frame %>%
  set_names(c('possessionFrame', 'shotFrame', 'outcome', 'rightHanded'))

# Read in all the sequences
# NOTE: This step takes a long time (minutes)
sequences = sequence_folders %>% map(combinePasses)
```

```
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
```

```
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
```

```
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
```

```
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
```

```
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
```

```
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
```

```
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
```

```
## New names:
## New names:
## * `` -> `...1`
```

```
# Change outcomes to more verbose names
info$outcome %<>% fct_recode(Goal = 'G', Save = 'GB', 'Defender Block' = 'DB', Miss = 'M')
```

Shot statistics retrieval

TODO: Explain what's happending here

```
# Get stats for the shot in every sequence
shots_stats.df <- seq_along(sequences) %>%
  map_dfr(goalShotStats) %>%
  # Some models can't use logical data
  mutate_if(is.logical, as.numeric)
```

Prediction modelling

TODO: This is beyond Assignment 2

```
# Split data into training and validation sets
outcomes.goal <- (info$outcome == 'Goal') %>% as.numeric %>% as.factor

set.seed(300)

train_ratio <- 1 / 2

t_ind <- sample(1:nrow(shots_stats.df), nrow(shots_stats.df) * train_ratio)
v_ind <- (1:nrow(shots_stats.df))[-t_ind]

t_shots.df <- shots_stats.df[t_ind,] %>%
  # Equivalent to rownames(t_shots.df) = NULL
  'rownames<-'(NULL)

v_shots.df <- shots_stats.df[v_ind,] %>%
  'rownames<-'(NULL)

t_goals <- outcomes.goal[t_ind]
v_goals <- outcomes.goal[v_ind]
```

Comparing classification and feature selection algorithms

TODO: This is beyond Assignment 2

```
# Evaluate the combinations of different classification and feature selection algorithms. Choosing your
set.seed(300)

# Set alpha levels for lasso and correlation analysis
lasso_thresh <- 0.1
cor_thresh <- 0.4

# All features
all_ft <- 1:ncol(t_shots.df)
```

```
# Lasso features
las_initial <- glmnet(t_shots.df,t_goals,family="binomial",alpha=1,nlambda=500,standardize=FALSE)
```

```
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```
glmfit <- glmnet(t_shots.df,t_goals,family="binomial",alpha=1,nlambda=500,standardize=FALSE)
```

```
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```
las_coefs <- coef(las_initial, s = lasso_thresh) %>% (\(x) x[2:nrow(x)])
las_ft <-  las_coefs %>% as.logical

# Correlation features
cor_coefs <- t_shots.df %>% map_dbl(~ cor.test(., t_goals %>% as.numeric)$p.value)
cor_ft <- cor_coefs < cor_thresh

# Boruta features along with one that includes undecided
bor_ft <- Boruta(t_goals ~ ., t_shots.df)$finalDecision == 'Confirmed'
bor_tent_ft <- Boruta(t_goals ~ ., t_shots.df)$finalDecision != 'Rejected'

model_funcs <- list(randomForest, lda, glmnet)
model_names <- list('RF', 'LDA', 'Logistic')

feat_selections <- list(las_ft, cor_ft, bor_ft, bor_tent_ft, all_ft)
feat_names <- list('lasso', 'correlation', 'boruta', 'boruta tentative', 'all')
```

The following tables show the balanced accuracy and PR AUC values for every combination of feature selection and model used.

- Models used were Random Forest, LDA, and Logistic Regression.
- Feature selection methods were Lasso, Pearson correlation coeficcient, Boruta, Boruta including inconclusive features, and all features.
- The model chosen as the best is *logistic regression* with features chosen by Lasso. Even though it doesn't have the highest balanced accuracy, that is partially due to the fact that the same 0.5 cutoff is used for every model. Its PR AUC score reveals it suggests the most accurate probabilities.

```
set.seed(300)
# Create tables that have every combination of model and feature selection and give their balanced accu
balanced_accs <- outer(model_funcs, feat_selections, Vectorize(\(x, y) makeModel(x, y)$acc)) %>%
  set_rownames(model_names) %>%
  set_colnames(feat_names)
```

```
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```
praucs <- outer(model_funcs, feat_selections, Vectorize(\(x, y) makeModel(x, y)$prauc)) %>%
  set_rownames(model_names) %>%
  set_colnames(feat_names)
```

```
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```
kable(balanced_accs, caption = 'Balanced Accuracy')
```

Table 1: Balanced Accuracy

	lasso	correlation	boruta	boruta tentative	all
RF	0.5000000	0.5000000	NA	0.5620567	0.5000000
LDA	0.6560284	0.5620567	NA	0.4787234	0.5620567
Logistic	0.7393617	0.7801418	NA	0.7287234	0.7393617

```
kable(praucs, caption = 'PR AUC')
```

Table 2: PR AUC

	lasso	correlation	boruta	boruta tentative	all
RF	0.463826672522325	0.460336912542795	NULL	0.310210991965024	0.440722703222703
LDA	0.408155173286752	0.317322261072261	NULL	0.314625509294627	0.258345274007039
Logistic	0.51952861952862	0.370875509294627	NULL	0.306584609030261	0.51952861952862

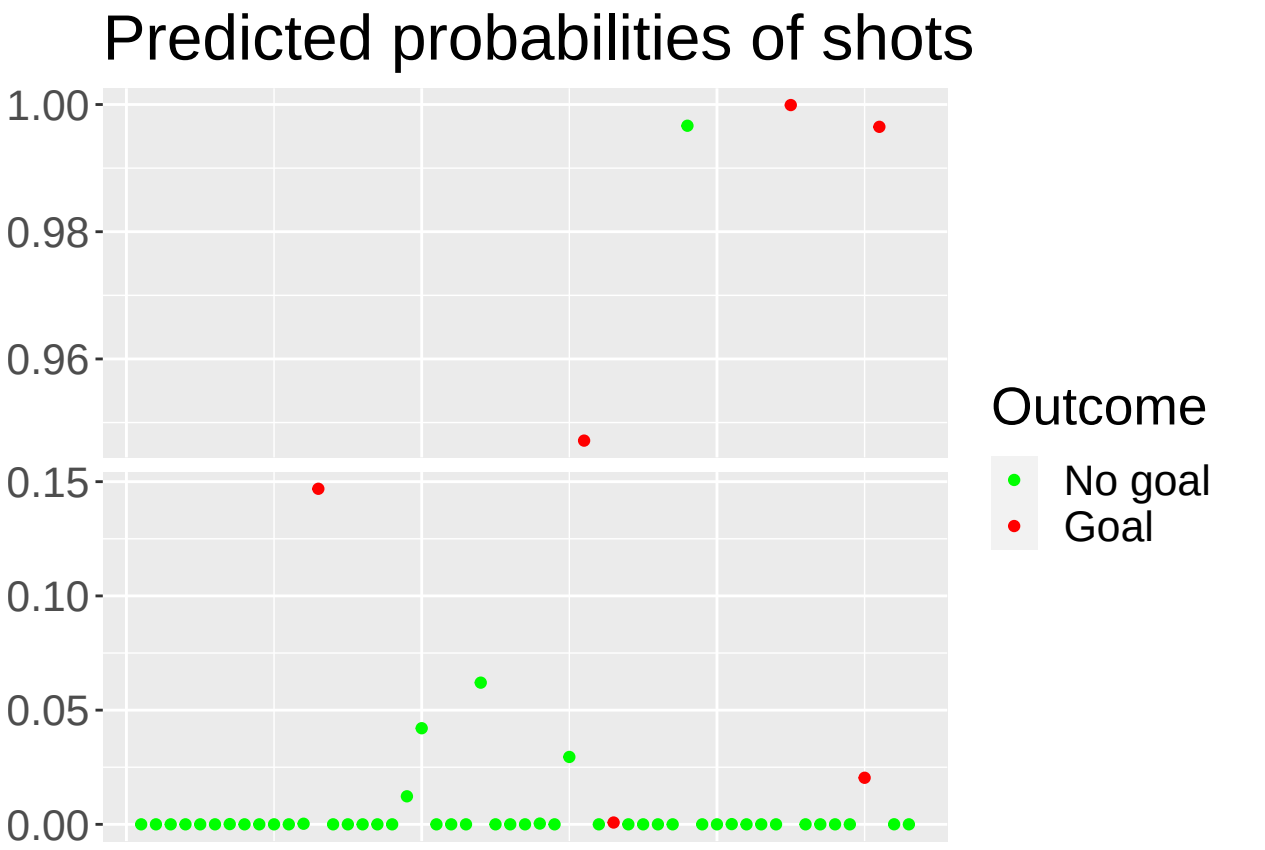
```
kable(colnames(shots_stats.df)[las_ft] %>% data.frame %>% set_names('Features selected'))
```

Features selected
puckDist
puckAngle
puckSpeed
goalieDist
goalieAngle
posTime

```
best_model <- makeModel(glmnet, las_ft)
```

```
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```
pred_scores <- predictedProbs(best_model$model, v_shots.df[best_model$features])
true_goal <- v_goals %>% fct_recode('No goal' = '0', 'Goal' = '1')

model_performance <- cbind.data.frame(pred_scores, true_goal) %>%
  set_names(c('pred', 'act')) %>%
  mutate(ind = row_number()) %>%
  mutate(bin = pred < 0.5)

# Create plot
ggplot(model_performance) +
  geom_point(aes(x = ind, y = pred, color = act)) +
  scale_color_discrete(type = c('green', 'red')) +
  labs(title = 'Predicted probabilities of shots', color = 'Outcome', x = NULL, y = NULL) +
  theme(axis.ticks.x = element_blank(),
        axis.text.x = element_blank(),
        strip.background = element_blank(),
        strip.text.y = element_blank(),
        text = element_text(size = 20))+
  facet_grid(bin ~ ., scale='free_y')
```




```
ggsave('pred.png')
```

```
## Saving 6.5 x 4.5 in image
```

Visualizing the logistic regression model

KernalSHAP is used to visualize the logistic regression model used. The first chart is a bar plot that shows the relative importance of each feature. The second is a "beeswarm" plot that shows the effect of each occurrence of each variable on the prediction result.

```
shap_feat <- best_model$features
shap_model <- makeModel(best_model$func, features = shap_feat)$model
```

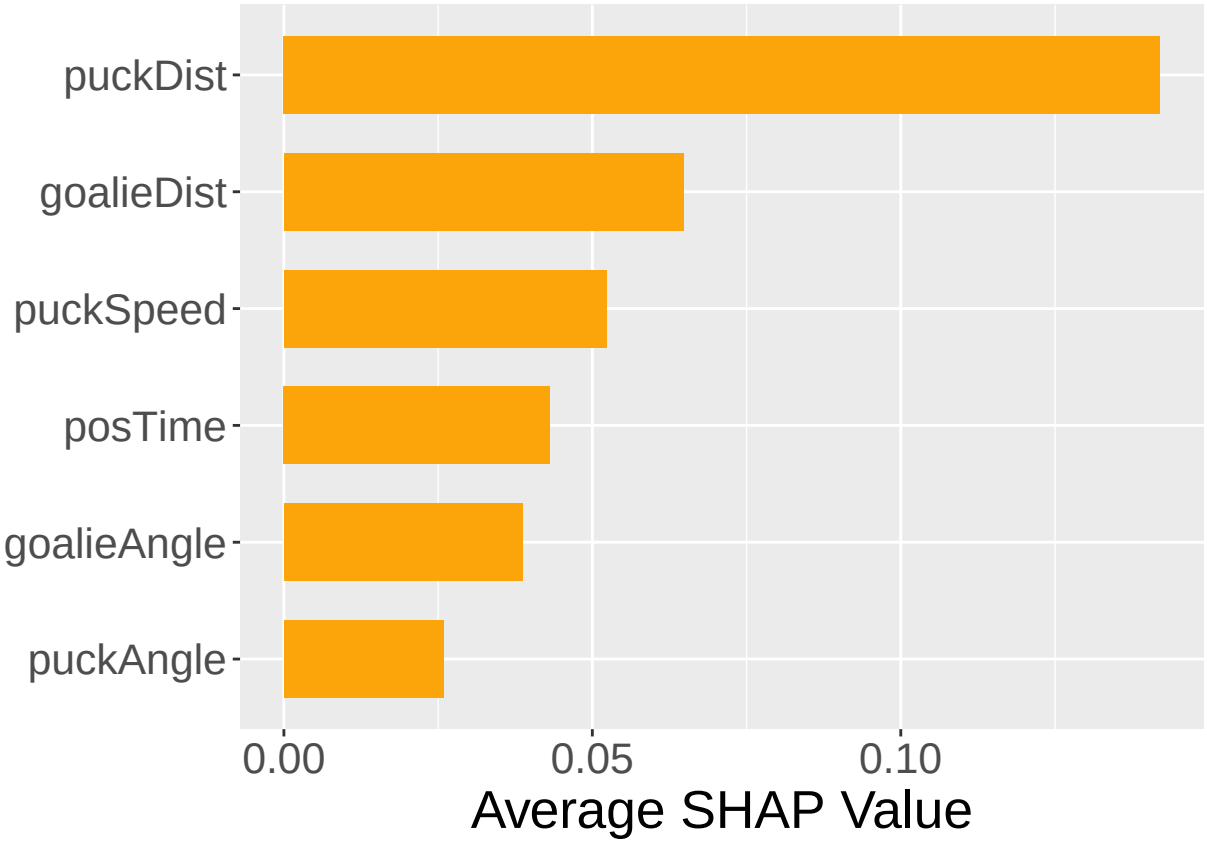
```
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```
kernel_shap <- kernelshap(shap_model, v_shots.df, v_shots.df, pred_fun = function(obj, X) predictedProbs
```

```
## Kernel SHAP values by the hybrid strategy of degree 2
```

```
##   |                                                                      |
```

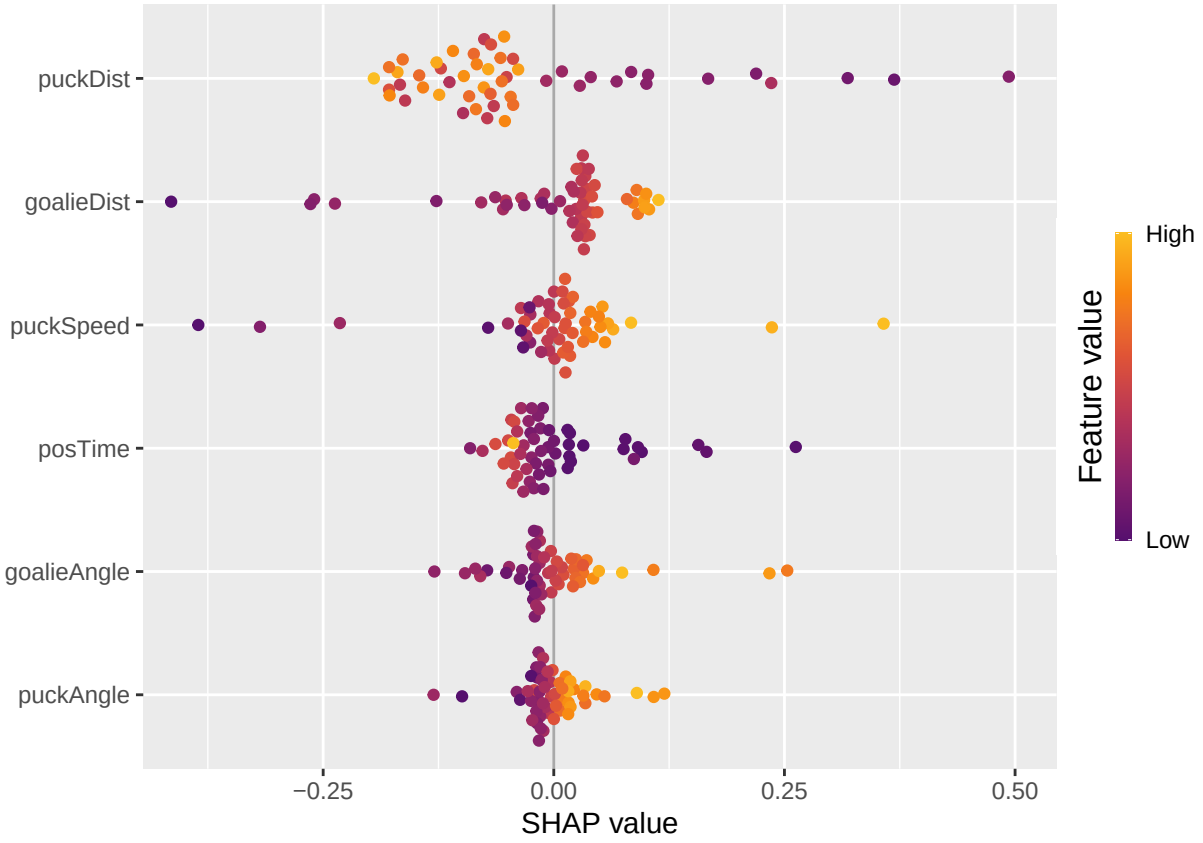
```
shap_viz <- shapviz(kernel_shap)

sv_importance(shap_viz, kind = 'bar', max_display = sum(shap_feat)) +
  theme(text = element_text(size = 20)) +
  labs(x = 'Average SHAP Value')
```




```
ggsave('bars.png')
```

```
## Saving 6.5 x 4.5 in image
```

```
sv_importance(shap_viz, kind = 'beeswarm', max_display = sum(shap_feat)) +
  # Take out major outliers without causing error
  scale_x_continuous(limits = c(-0.4, 0.5), oob = oob_keep)
```



This heatmap shows the predicted likelihood of scoring a goal given a player's location on the rink. In addition, all shot locations and outcomes are plotted. The model used is a Logistic Regression that has been fed only puck distance and angle

```
set.seed(300)
# Create a model that only uses puck distance and angle
heatmap_model <- makeModel(best_model$func, c(1, 2))$model
```

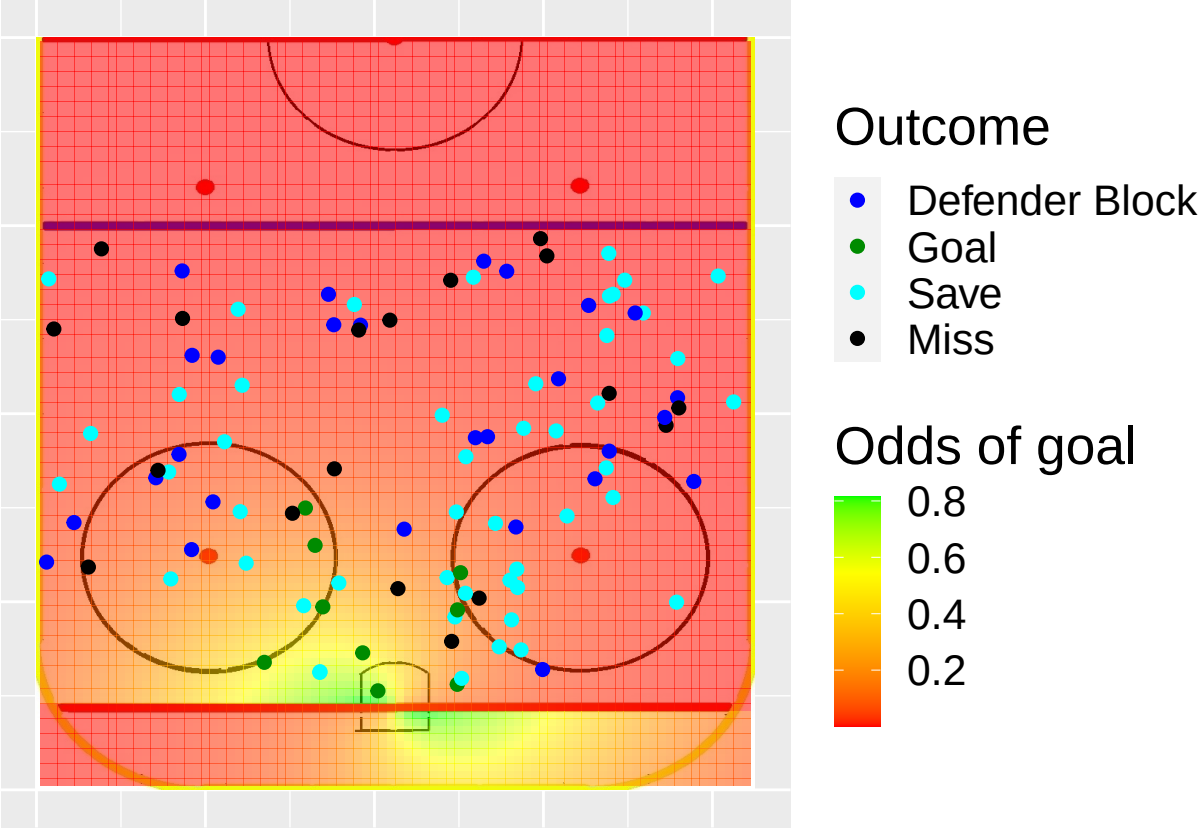
```
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```
# Calculate the odds of goal for x and y coords across the rink
coord_scores <- outer(xsize - (1:(xsize/20) * 10), 1:(ysize/10) * 10, Vectorize(\(x, y) standardized_sh

# Change from wide to long format
coord_scores.df <- coord_scores %>%
  melt %>%
  set_names(c('x', 'y', 'value'))

# Graph the heatmap
halfRinkGraph(coord_scores.df) +
  # Fill background with odds for that shot
  geom_tile(aes(x = y * 10, y = xsize - x * 10, fill = value, alpha = 0.5)) +
  # Dot for each player colored by outcome
  geom_point(data = shots_stats.df, aes(color = info$outcome, x = shotStatX(shots_stats.df), y = shotSta
  scale_fill_gradientn('Odds of goal', colors = c('red', 'orange', 'yellow', 'green')) +
  scale_color_discrete('Outcome', type = c('blue', 'green4', 'cyan', 'black')) +
```

```
  # Remove X and Y labels and alpha legend
  labs(y = NULL, x = NULL) +
  guides(alpha = 'none') +
  theme(text = element_text(size = 20))
```

```
## Warning: Removed 184 rows containing missing values (`geom_tile()`).
```

```
## Warning: Removed 3 rows containing missing values (`geom_point()`).
```




```
ggsave('heatmap.png')
```

```
## Saving 6.5 x 4.5 in image
```

```
## Warning: Removed 184 rows containing missing values (`geom_tile()`).
## Removed 3 rows containing missing values (`geom_point()`).
```

This table shows the values of all variables given outcome

```
# Using levels only returns entries with that outcome, while using unlevels returns entries without tha
outcomeMeans <- function(levels = c(), unlevels = unique(info$outcome)){
  return(shots_stats.df[info$outcome %in% levels | !info$outcome %in% unlevels,] %>% colMeans)
}

shot_type_stats.m <- tibble('Goals' = outcomeMeans('Goal'),
                            'Nongoals' = outcomeMeans(unlevels = 'Goal'),
                            'Shots on goal' = outcomeMeans(c('Goal', 'Save')),
                            'Goalie block' = outcomeMeans('Save'),
                            'Misses' = outcomeMeans('Miss'),
                            'Defender block' = outcomeMeans('Defender Block')) %>%
  t %>%
```

```
  'colnames<-'(colnames(shots_stats.df))

splitKable(shot_type_stats.m, 2, sigfigs = 3)
```

```
##
##
## |               | puckDist| puckAngle| puckSpeed| shooterSpeed| goalieDist| goalieAngle|
## |:--------------|--------:|---------:|---------:|------------:|----------:|-----------:|
## |Goals          |  153.297|   101.085|    42.286|       14.984|     53.767|      87.722|
## |Nongoals       |  427.267|    84.206|    41.540|       15.938|     70.524|      76.351|
## |Shots on goal  |  361.554|    83.572|    42.567|       16.888|     65.147|      73.515|
## |Goalie block   |  399.040|    80.419|    42.617|       17.231|     67.196|      70.958|
## |Misses         |  463.746|    88.769|    42.029|       14.323|     81.184|      82.812|
## |Defender block |  453.868|    88.008|    39.201|       14.679|     69.187|      81.793|
##
##
## |               | posTime| sameDefenders| oppDefenders| goalieScreened| rightHanded|
## |:--------------|-------:|-------------:|------------:|--------------:|-----------:|
## |Goals          |   5.333|         0.000|        1.444|          0.333|       0.333|
## |Nongoals       |  31.365|         0.573|        2.115|          0.719|       0.479|
## |Shots on goal  |  27.271|         0.339|        1.763|          0.576|       0.407|
## |Goalie block   |  31.220|         0.400|        1.820|          0.620|       0.420|
## |Misses         |  23.895|         0.526|        2.158|          0.789|       0.474|
## |Defender block |  36.889|         0.926|        2.630|          0.852|       0.593|
```

```
hand_stats <- shots_stats.df %>%
  group_by(rightHanded) %>%
  summarise(across(everything(), mean))

splitKable(hand_stats, 2)
```

```
##
##
## | rightHanded| puckDist| puckAngle| puckSpeed| shooterSpeed| goalieDist|
## |-----------:|--------:|---------:|---------:|------------:|----------:|
## |           0| 404.6215|  80.18428|   41.8373|     14.71233|   66.18993|
## |           1| 402.8260|  91.90298|   41.3375|     17.16263|   72.40029|
##
##
## | goalieAngle|  posTime| sameDefenders| oppDefenders| goalieScreened|
## |-----------:|--------:|-------------:|------------:|--------------:|
## |    63.70455| 29.03571|     0.5535714|     1.982143|      0.6071429|
## |    92.89345| 29.24490|     0.4897959|     2.142857|      0.7755102|
```

This is the result of a k-means clustering on the data. The points were clustered almost perfectly based on location

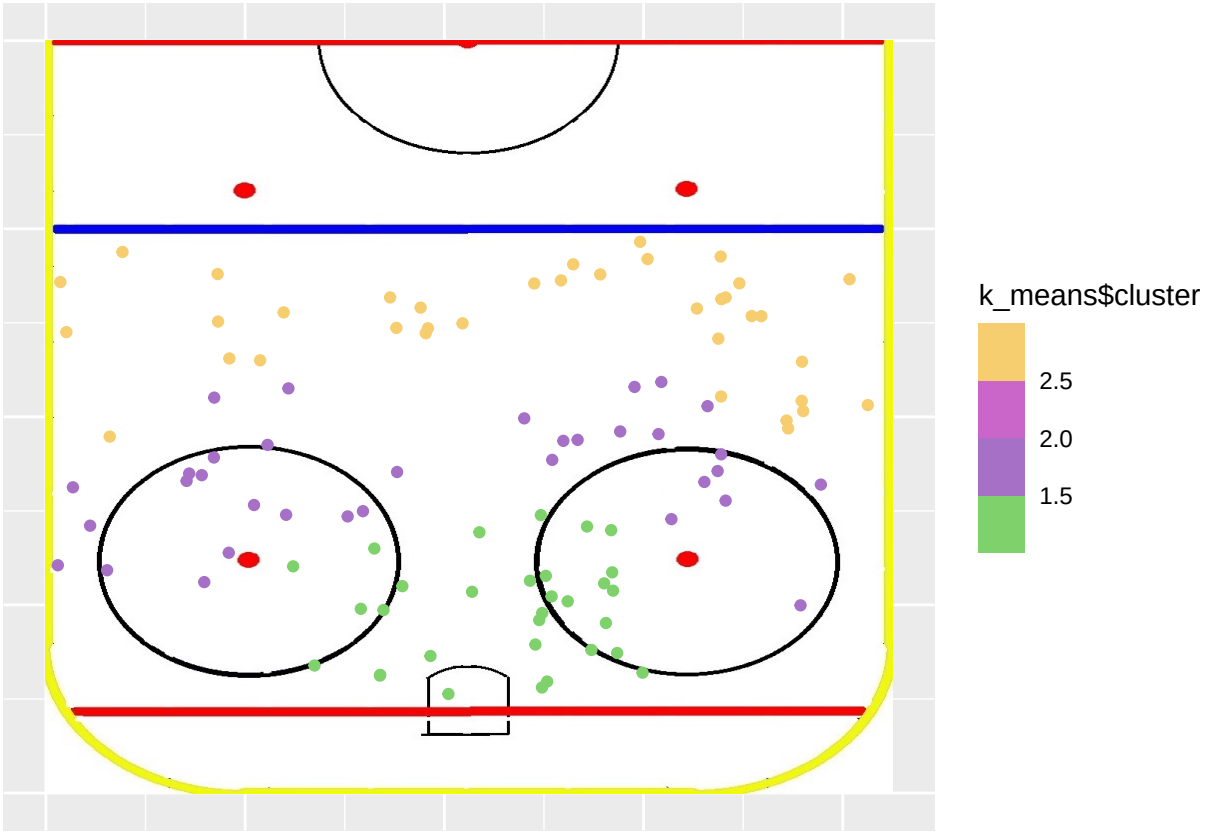
```
set.seed(100)
n_clust <- 3

k_means <- kmeans(shots_stats.df, n_clust)

# Add number of points in each cluster to data
cluster_data = cbind(k_means$size, k_means$centers)
colnames(cluster_data)[1] = 'Number of players'
```

```
splitKable(cluster_data, 2, sigfigs = 3)
```

```
##
##
## | Number of players| puckDist| puckAngle| puckSpeed| shooterSpeed| goalieDist|
## |-----------------:|--------:|---------:|---------:|------------:|----------:|
## |                31|  171.789|    78.853|    39.540|       17.254|     49.096|
## |                34|  396.650|    96.943|    45.507|       18.618|     69.640|
## |                40|  589.643|    81.326|    39.886|       12.425|     84.112|
##
##
## | goalieAngle| posTime| sameDefenders| oppDefenders| goalieScreened| rightHanded|
## |-----------:|-------:|-------------:|------------:|--------------:|-----------:|
## |      66.711|  21.871|         0.129|        1.355|          0.355|       0.419|
## |      91.767|  35.853|         0.441|        2.118|          0.735|       0.529|
## |      73.278|  29.050|         0.900|        2.550|          0.900|       0.450|
```

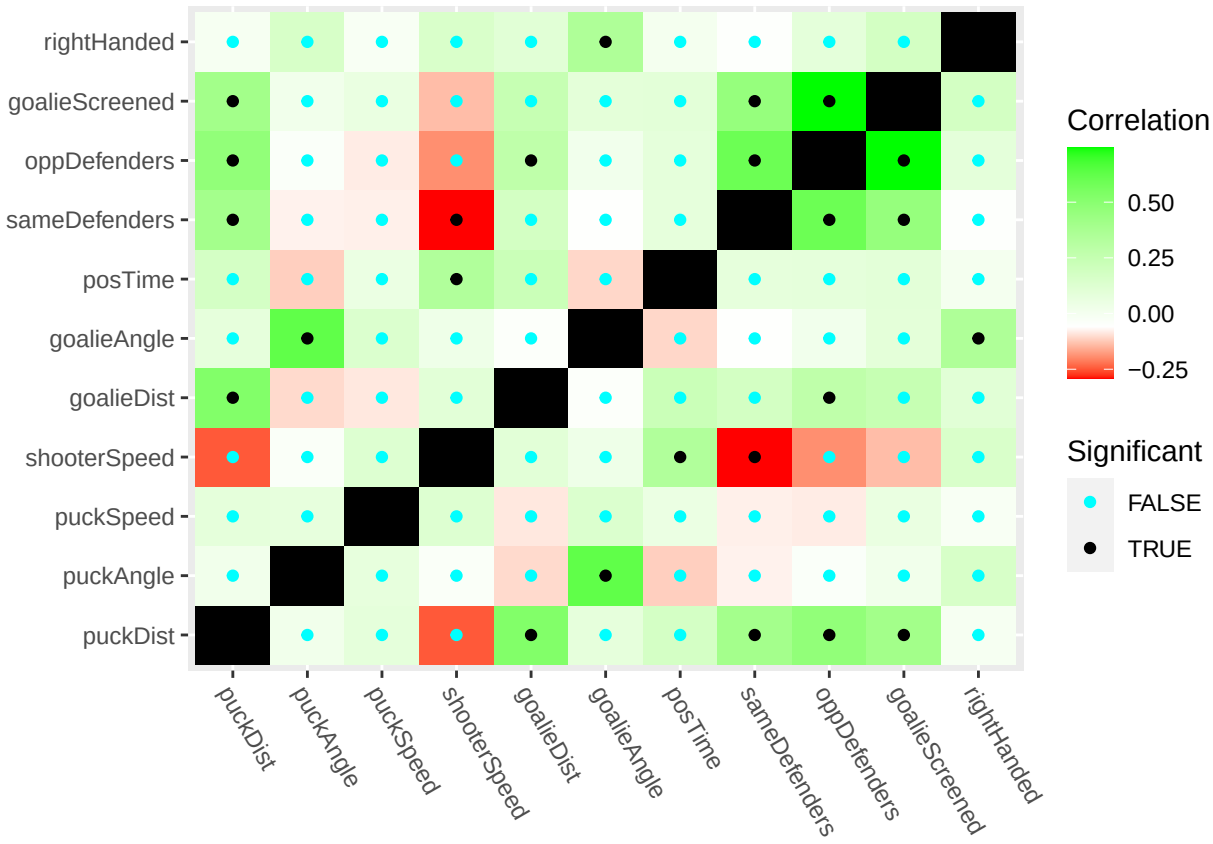
```
# Graph the rink with players colored by cluster
halfRinkGraph(shots_stats.df) +
  geom_point(aes(color = k_means$cluster, x = shotStatX(shots_stats.df), y = shotStatY(shots_stats.df)))
  scale_color_stepsn(colors = c('green', 'purple', 'yellow'))
```

```
## Warning: Removed 3 rows containing missing values (`geom_point()`).
```



This chart shows the correlations of all variables. Correlations whose p-values are under 0.005 are marked as being significant.

```
corrMatrix(shots_stats.df, thresh = 0.005)
```



Notable correlations:

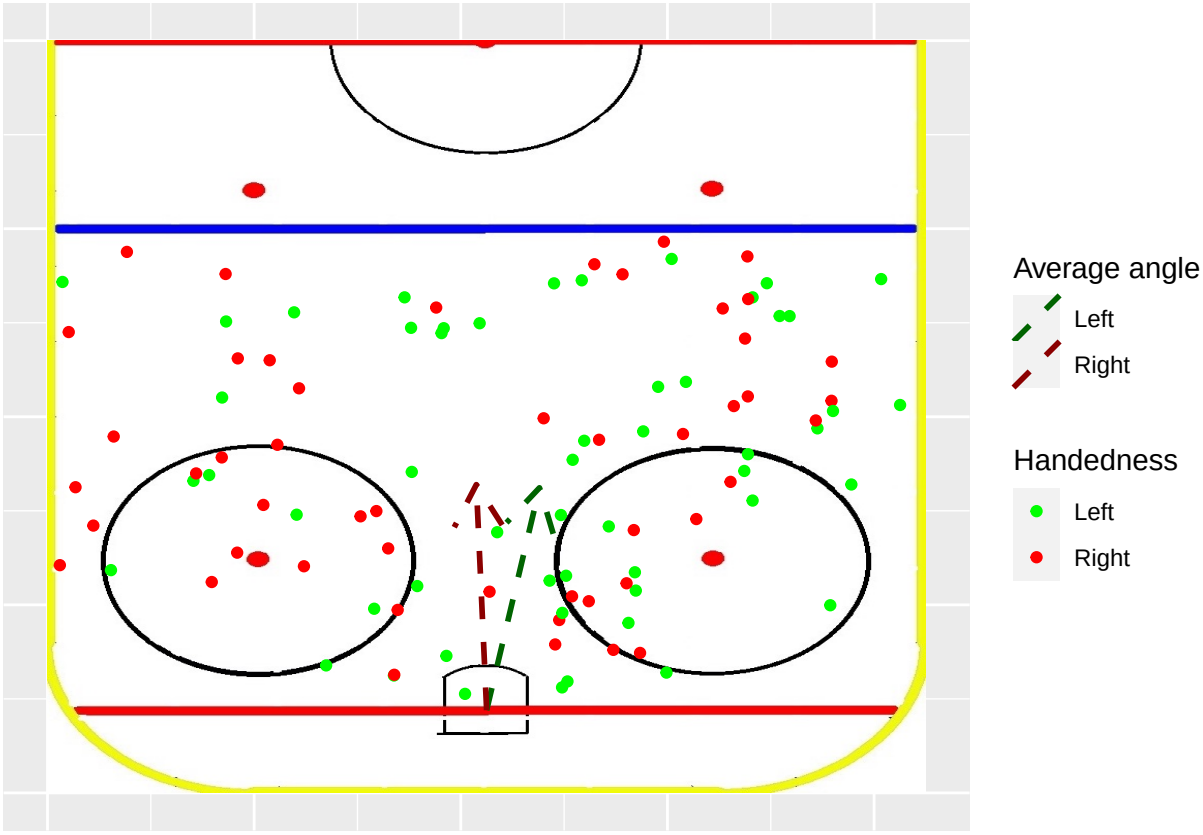
- Being right-handed is highly correlated with the angle the goalie is at. The goalie angle itself is highly correlated with the puck angle, which makes sense if the goalie tracks the puck and moves to intercept it
- The goalie is more likely to be screened the further away the puck is. The number of defenders on both teams is also positively correlated with puck distance
- The goalie being far away from the net (relatively) is positively correlated with the puck being far away as well. Goalie distance is also positively correlated with posession time.
- Shooter speed is negatively correlated with the number of defenders and positively with posession time

This first figure below is a graph of the rink with the handedness of players labeled and arrows representing the average angle of the puck for each handedness. Left-handed players average shooting from their left of the goal, but right-handed players average shooting from the center. Overall, most shots come from the sides much more often than from the middle

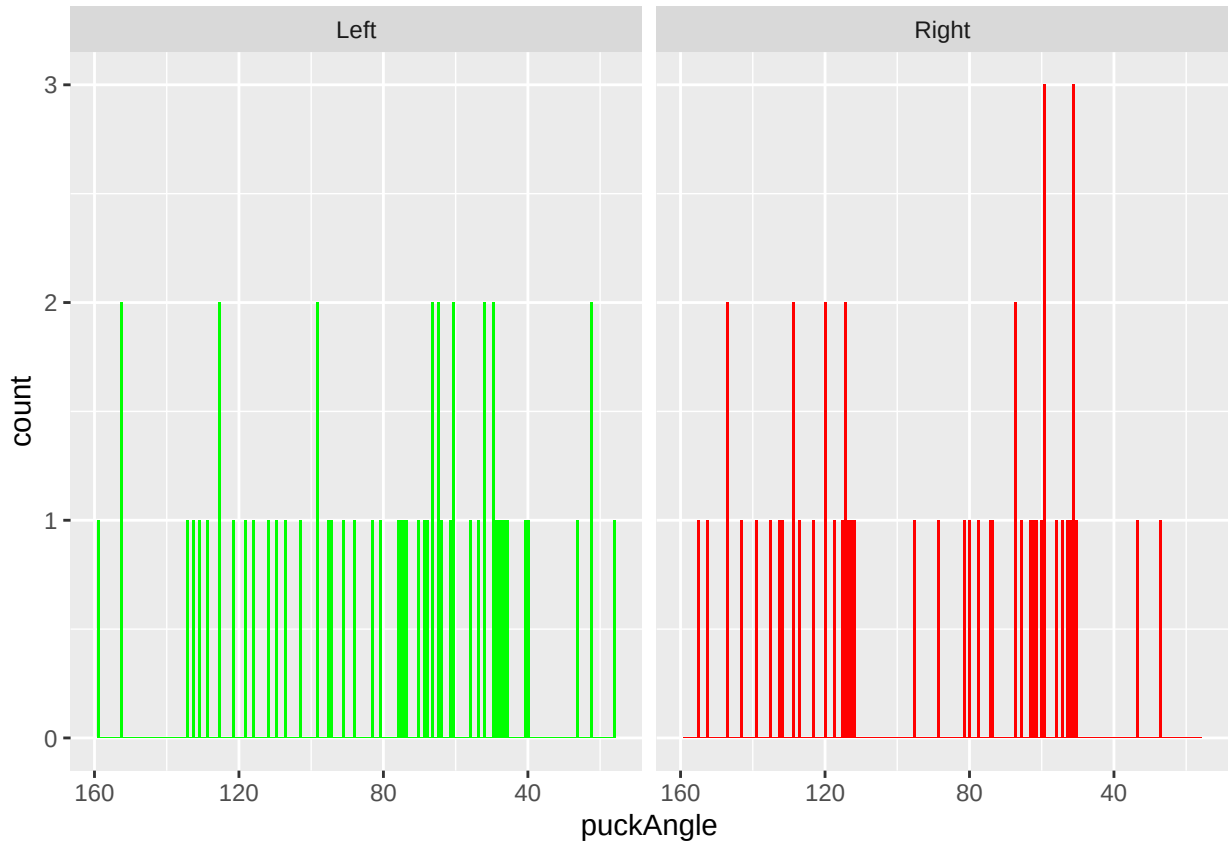
```
angle_means <- shots_stats.df %>%
  group_by(rightHanded) %>%
  summarise(mean(puckAngle)) %>%
  set_names(c('rightHanded', 'meanAngle')) %>%
  # Data for graphing
  mutate(xstart = ysize / 2) %>%
  mutate(ystart = pipes_x) %>%
  mutate(radius = 300)

halfRinkGraph(shots_stats.df) +
  # Graph players colored by handedness
  geom_point(aes(color = as.logical(rightHanded), x = shotStatX(shots_stats.df), y = shotStatY(shots_sta
  scale_color_discrete('Handedness', type = c('green1', 'red'), labels = c('Left', 'Right')) +
```

```
  new_scale_color() +
  # Arrow pointing to average direction
  geom_spoke(data = angle_means, aes(x = xstart, y = ystart, angle = torad(meanAngle), radius = radius,
  scale_color_discrete('Average angle', type = c('darkgreen', 'darkred'), labels = c('Left', 'Right')) +
  labs(x = NULL, y = NULL)
```

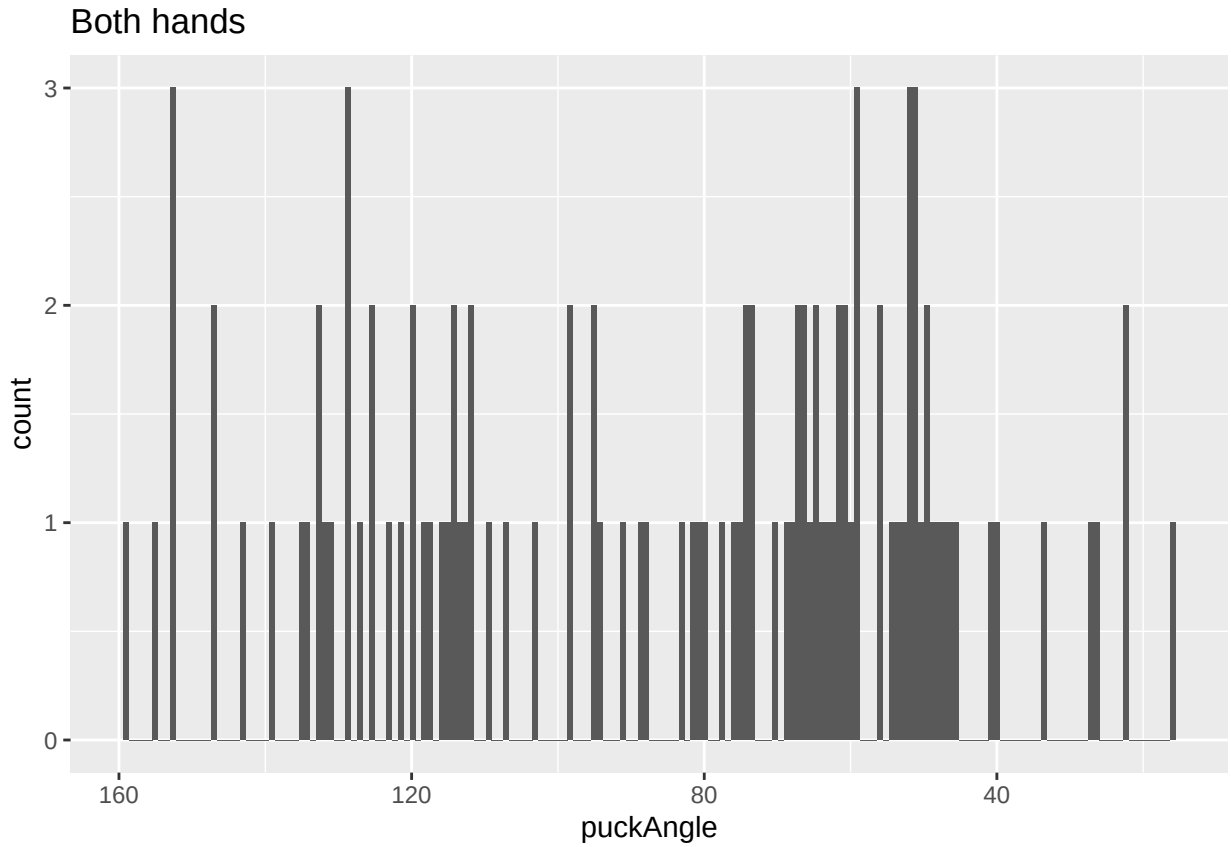
```
## Warning: Removed 3 rows containing missing values (`geom_point()`).
```




```
# Plot histogram distributions of left and right handed player shot angles
ggplot(shots_stats.df %>%
         mutate(rightHanded = factor(rightHanded, levels = c(0, 1), labels = c('Left', 'Right')))) +
  geom_histogram(aes(x = puckAngle, fill = rightHanded), bins = 180) +
  # Reverse the scale so angles on histogram correspond with the rink image
  scale_x_reverse() +
  scale_fill_discrete(type = c('green', 'red')) +
  facet_wrap(~ rightHanded) +
  guides('fill' = 'none')
```




```
# Plot total distribution of shot angles
ggplot(shots_stats.df) +
  geom_histogram(aes(x = puckAngle), bins = 180) +
  scale_x_reverse() +
  labs(title = 'Both hands')
```

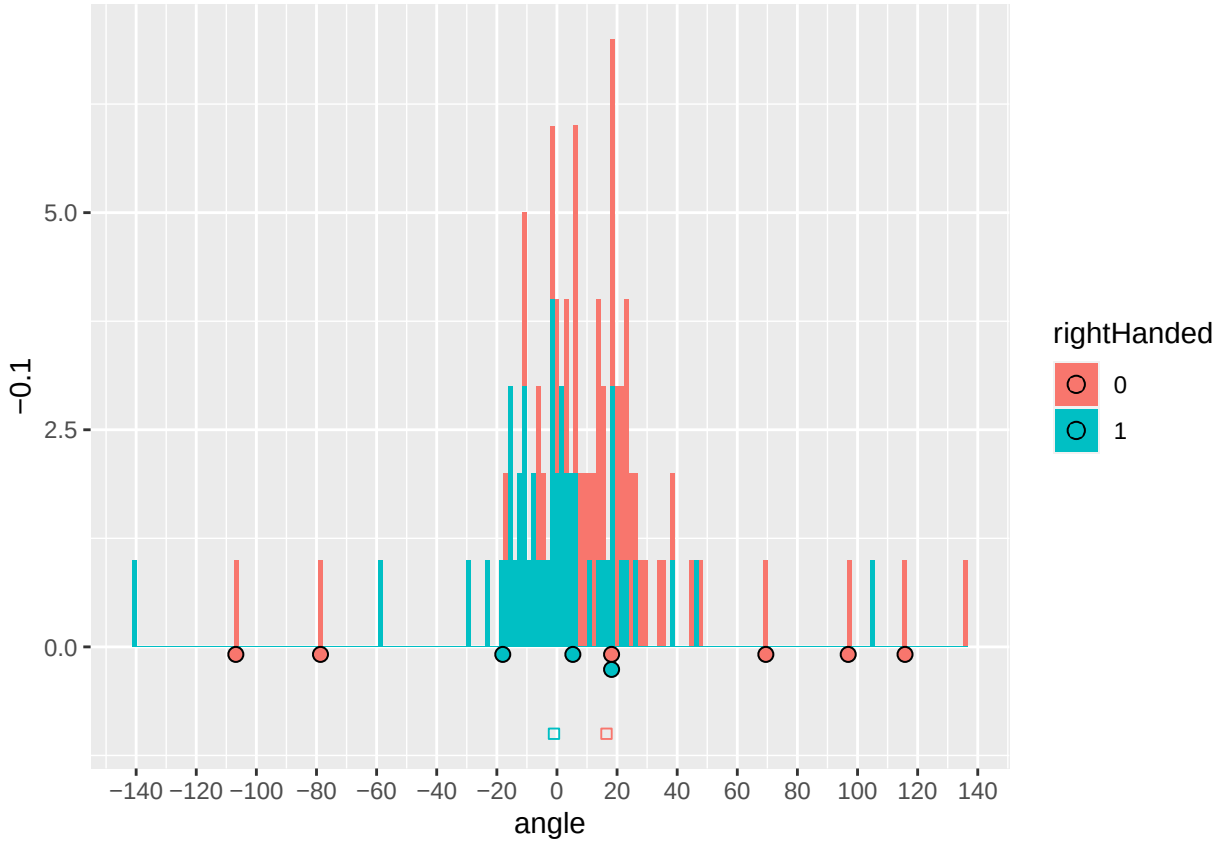


These histograms show the distribution of the differences between the goalie's and puck's angle from the goal. In most observations, the goalie was in a 30 degree radius of the puck. Red dots mark angle differences corresponding to goals. The outliers disproportinately represent goals, even when only shots on goal are considered, meaning that a high angle difference is more likely to yield a goal. The scatterplot shows the correlation between the absolute values of the differences in angle and the possession time of the puck, which is a slight downward trend indicating that the longer the player held the puck, the more the goalie was able to get into position.

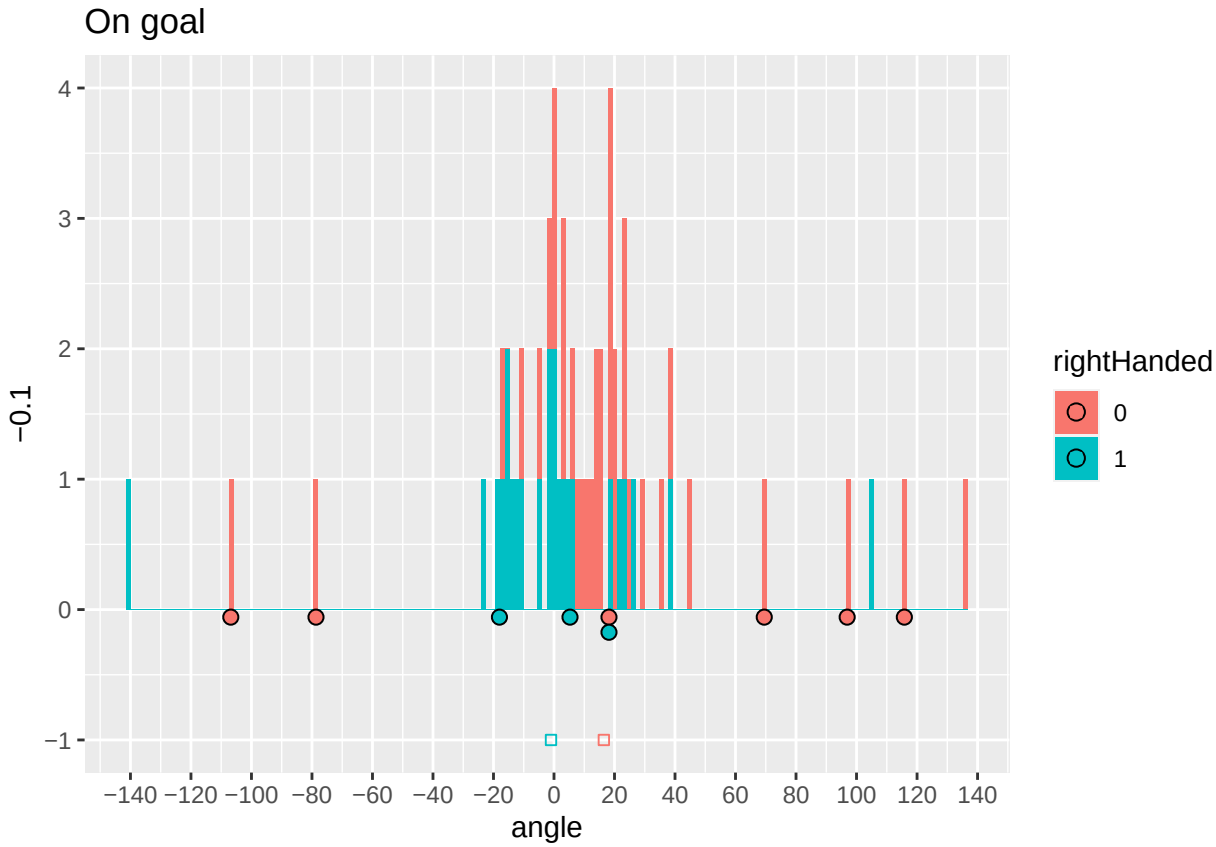
Additionally, for shots from left-handed players, the goalie was almost 20 degrees off on average (11 if we discount outliers by only taking angles -40 <= x <= 40). This effect is not present for right handed player's shots

```
# Create the dataframe with angle difference, posession time, and outcome
angle_diff <- (shots_stats.df['puckAngle'] - shots_stats.df$goalieAngle) %>%
  set_names('angle') %>%
  mutate(posTime = shots_stats.df$posTime) %>%
  mutate(goal = factorBool(outcomes.goal))%>%
  mutate(rightHanded = as.factor(shots_stats.df$rightHanded))

hand_averages <- angle_diff %>%
  group_by(rightHanded) %>%
  summarise(across(everything(), mean))

# When adding a list of ggplot objects, it adds each one in turn
histogram_base <- list(
  geom_histogram(aes(x = angle, fill = rightHanded), bins = 180),
  scale_x_continuous(breaks = breaks_extended(n = 20)),
  geom_dotplot(data = angle_diff[outcomes.goal %>% factorBool, ], aes(x = angle, y = -0.1, fill = rightH
```

```
  geom_point(data = hand_averages, aes(x = angle, y = -1, color = rightHanded), shape = 0)
)

# Plot the histogram of all angle differences
ggplot(angle_diff) +
  histogram_base
```




```
# Plot the histogram of
ggplot(angle_diff[info$outcome %in% c('Goal', 'Save'), ]) +
  histogram_base +
  labs(title = 'On goal')
```


On goal
4
3
2
1
0
−1
−0.1
−140 −120 −100 −80 −60 −40 −20 0 20 40 60 80 100 120 140
angle
rightHanded
0
1