Data Analysis Challenge

Domenic Galamini Jr.

18/09/2021

Question 1

"Who are the best (and worst) shooters? How confident are you in your answer(s)?"

To tackle this problem, shooters (and goalies) are modeled as random effects in multilevel/mixed models that aim to predict the probability of a goal at the point-of-release. From there, random effects (i.e., the parameters concerning the talent of shooters and goalies) are sampled from their posterior distributions to get a handle on uncertainty.

~~To supplement the aforementioned results, goal-probability-maximizing `locations_on_net` (e.g., high left, five-hole, low right etc.) are predicted for each shot and each shooter's ability (or inability) to place pucks in optimal locations is assessed. Again, the population of shooters is simply treated as a random effect.~~

Due to the uncertainty involved, the estimates put forth function best as directional indicators, best-suited to answering questions like "Is it likely that this shooter is above league-average?". Drawing strong conclusions or placing stock in player rankings is ill-advised.

1.2.1 Preparing the Data

First, let's attach our dependencies and set the working directory

```
library(data.table); library(lme4); library(mgcv); library(dplyr);
library(ggplot2); library(ggridges); library(kdensity); #library(hrbrthemes)

## Please change the following to your own working directory
setwd("C:/Users/Dom/Dropbox/My PC (DESKTOP-8IRMM3R)/Desktop/DAC")
```

Now, let's pull our data into R...

```
## Pull in the shots file and store it as a data frame
dat <- fread("shot-attempts.csv")

## Let's take a look...
head(dat)
```

```
##    season game_id shooter_id goalie_id shot_type shooter_hand goalie_glove
## 1:      1       1        383         2  Backhand            L            L
## 2:      1       1        246         1      Snap            R            L
## 3:      1       1        246         1  Backhand            R            L
## 4:      1       1         48         1     Wrist            L            L
```

```
## 5:      1       1        238         2     Wrist            L            L
## 6:      1       1        120         1      Slap            R            L
##    location_on_net rush_situation player_strength game_time x_coord y_coord
## 1:       Low Right           None             5v5     62.03   83.80    4.78
## 2:       High Left           Even             5v5     79.63   33.39   40.99
## 3:            None           None             5v5    101.27   28.86   25.40
## 4:       Five Hole           None             5v5    104.07   70.10   -4.27
## 5:            None           Even             5v5    135.83   61.17  -15.34
## 6:           Chest           Even             5v5    179.77   40.43   25.90
##    shot_result
## 1:        Save
## 2:        Save
## 3:       Block
## 4:        Save
## 5:       Block
## 6:        Save
```

Checking for NAs and nulls by column - looks like we have quite a few missing values for `shooter_hand` and goalie_glove

```
## NAs
sapply(dat, function(x) sum(is.na(x)))
```

```
##          season         game_id      shooter_id       goalie_id       shot_type
##               0               0               0               0               0
##    shooter_hand    goalie_glove location_on_net  rush_situation player_strength
##            4167            3653               0               0               0
##       game_time         x_coord         y_coord     shot_result
##               0               0               0               0
```

```
## Nulls
sapply(dat, function(x) sum(is.null(x)))
```

```
##          season         game_id      shooter_id       goalie_id       shot_type
##               0               0               0               0               0
##    shooter_hand    goalie_glove location_on_net  rush_situation player_strength
##               0               0               0               0               0
##       game_time         x_coord         y_coord     shot_result
##               0               0               0               0
```

Making sure it's sorted properly

```
## Sort by game_id and game_time
dat <- dat %>%
  arrange(game_id, game_time)
```

Remove shots below the goalline (plagued by recorder bias) and shots outside of the offensive zone (not relevant to shooter/goalie talent)

```
dat <- dat %>%
  filter(x_coord >= 25, x_coord <= 89)
```

Before jumping in, we need to create some new variables (e.g., distance, angle etc.) that might be useful later

```
dat <- dat %>%
  mutate(## Goal indicator
         isGoal = ifelse(shot_result == "Goal", 1, 0),

         ## SOG indicator
         isSOG = ifelse(shot_result %in% c("Goal", "Save"), 1, 0),

         ## Rebound proxy - A rebound will be defined to be a shot within 2
         # seconds of the shot prior. This may sound strict, but we are missing
         # the team name/id of the shooting team, so we need to rule out the
         # possibility of a transition play leading to an opposition shot
         isReb = ifelse(lag(shot_result) %in% c("Goal", "Block", "Miss") |
                        lag(game_time) < (game_time - 2) |
                        is.na(lag(game_time)), 0, 1),

         ## Angle change from previous shot for rebounds. Any angle change that
         # occurs below the goalline (or "behind-the-net") is truncated
         RebAngDelta = ifelse(isReb == 1 & (lag(x_coord) < 89 |
                                            x_coord < 89),
            abs(((atan2((pmin(x_coord, 89) - 89), y_coord) + pi) %% pi) -
                 (atan2((pmin(lag(x_coord), 89) - 89),
                        lag(y_coord)) + pi) %% pi), 0),

         ## Time since previous shot for rebounds
         tSincePrevShot = ifelse(!is.na(lag(game_time)),
                                 game_time - lag(game_time), 999),

         ## Approximate rebound angular velocity (radians / second)
         RebAngVel = RebAngDelta / tSincePrevShot,

         ## Shooter on his off wing
         off_wing = case_when(is.na(shooter_hand) ~ "Unkown",
                              (y_coord > 0 & shooter_hand == "L") |
                                (y_coord < 0 & shooter_hand == "R") ~ "Yes",
                              TRUE ~ "No"),

         ## Shot distance and angle
         dist = sqrt((x_coord - 89)^2 + (y_coord)^2),
         angle = atan2(y_coord, 89 - x_coord),

         ## Convert shooter_id and goalie_id into factors
         shooter_id = as.factor(shooter_id),
         goalie_id = as.factor(goalie_id))
```

In order to isolate shooting talent, it will be critical to first ensure that all other pre-release factors are controlled for...

1.2.2 Probability of a Goal for an Average Shooter/Goalie

In this section, each shot attempt is evaluated to ascertain the probability of scoring, ignoring shooter and goalie identity for the time being. The problem lends itself well to a Generalized Additive Model since the response is likely to be heavily driven by spatial data (i.e., x-y locations and their parametrizations). Let's begin with a baseline model to compare to...

```
## Beginning with a basic linear model as our baseline
linear_mod <- bam(isGoal ~ 1 + shot_type + off_wing + rush_situation +
                    player_strength + game_time + RebAngVel +
                    dist + I(abs(angle)),
                  data = dat, family = "binomial", method = "REML")
```

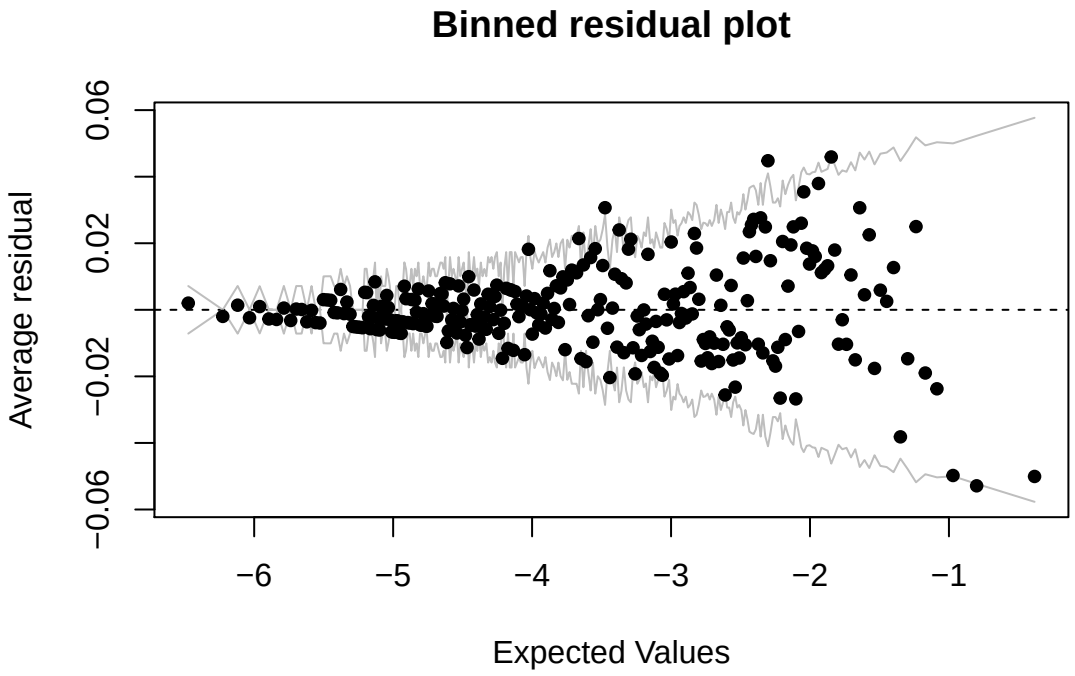
Using `mgcv::`, smooths and smooth surfaces can be built up from the weighted sums of basis functions - this will help us with spatial variables like distance and angle that can be non-linear in their relationships on the link scale and interact with other variables in the data set. The parameters of these smooth functions are automatically penalized via the restricted maximum likelihood (REML) method of estimation to avoid over-fitting the data. Below, a couple of candidate models are specified and their AICs are compared (Note: These models are quite conservative, primarily to avoid complete separation - the categorical variables could perhaps benefit from penalization or a Bayesian approach)

```
## Add in some smooth terms for distance and angle
shots_mod1 <- gam(isGoal ~ 1 + shot_type + off_wing + rush_situation +
                    player_strength + RebAngVel + s(dist) + s(angle),
                  data = dat, family = "binomial", method = "REML")

## Interact distance and angle (preserves their main smooth terms as well). Use
# a tensor product because they are on different scales
shots_mod2 <- gam(isGoal ~ 1 + shot_type + off_wing + rush_situation +
                    player_strength + RebAngVel + te(dist, angle),
                  data = dat, family = "binomial", method = "REML")

## Remove off_wing
shots_mod3 <- gam(isGoal ~ 1 + shot_type + rush_situation +
                    player_strength + RebAngVel + s(dist) + s(angle),
                  data = dat, family = "binomial", method = "REML")

## Compare AIC
AIC(linear_mod, shots_mod1, shots_mod2, shots_mod3)
```

```
##                   df      AIC
## linear_mod 18.00000 24876.41
## shots_mod1 27.94174 24823.80
## shots_mod2 26.77620 24842.34
## shots_mod3 25.97294 24820.21
```

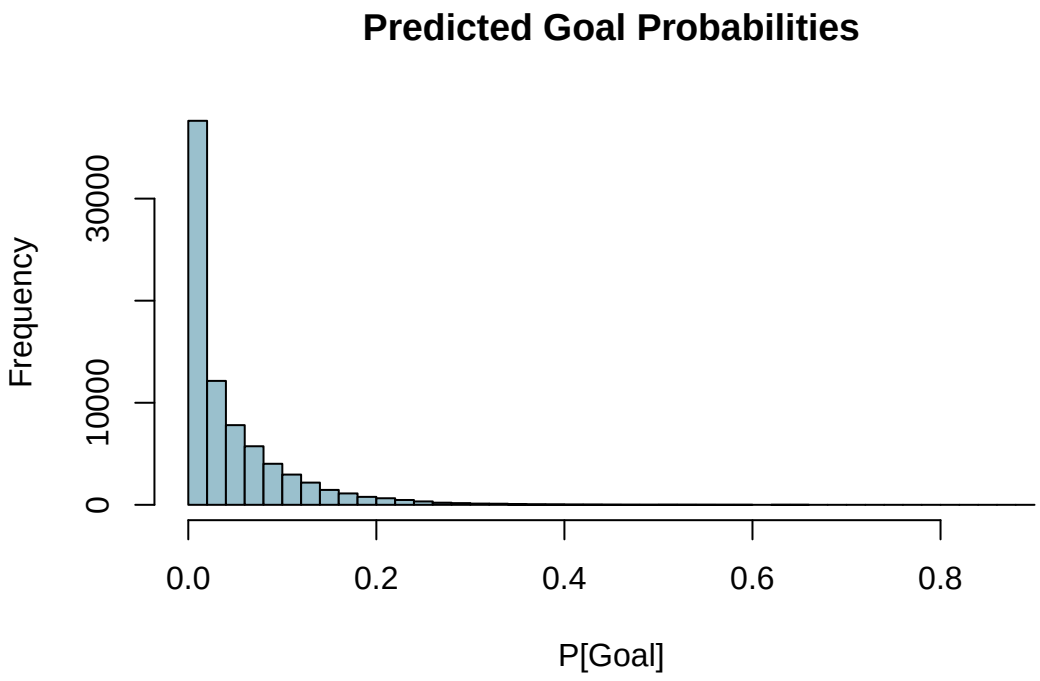
Taking a look at the binned residuals with the `arm::` package (~95% of points should fall within gray boundaries)

```
## Predictions on link (logit) scale - looks good!
arm::binnedplot(arm::logit(fitted(shots_mod3)),
                residuals(shots_mod3, type  = "response"))
```



Histogram of model predictions...

```
## Histogram of predictions
hist(fitted(shots_mod3), col = "lightblue3", breaks = 40,
     main = "Predicted Goal Probabilities", xlab = "P[Goal]")
```



1.2.3 Incorporating Shooter Effects

To account for shooter identity in our xG model, we could use the `gamm4::` package and simply add a random effect term for shooter identity to the model formula used in `shots_mod3`. This works, but it can take some time to run the model. For this analysis, I will use the logit transformation of our predicted probabilities from `shots_mod3` as an offset in a mixed model and then incorporate shooter identity as an intercept-varying random effect post-hoc, without the need to fit everything simultaneously. It's important to note that by doing this, we are assuming independence between the shooter effects and the other variables in the model. (A benefit of this shortcut, beyond tractability under a 24 hour deadline, is that it grants the option of constructing the underlying xG model with your algorithm of choice)

```
## Add predicted log-odds of scoring to the data
dat$shots_odds <- arm::logit(fitted(shots_mod3))

## Fit a GLMM with lme4
# I won't use their estimate here, but I'm also adding goaltenders as a control
shots_mer_mod <- glmer(data = dat, isGoal ~ 1 + offset(shots_odds) +
  (1|shooter_id) + (1|goalie_id), family = "binomial")
```

Extracting the shooter effects and plotting the top 5 and bottom 5

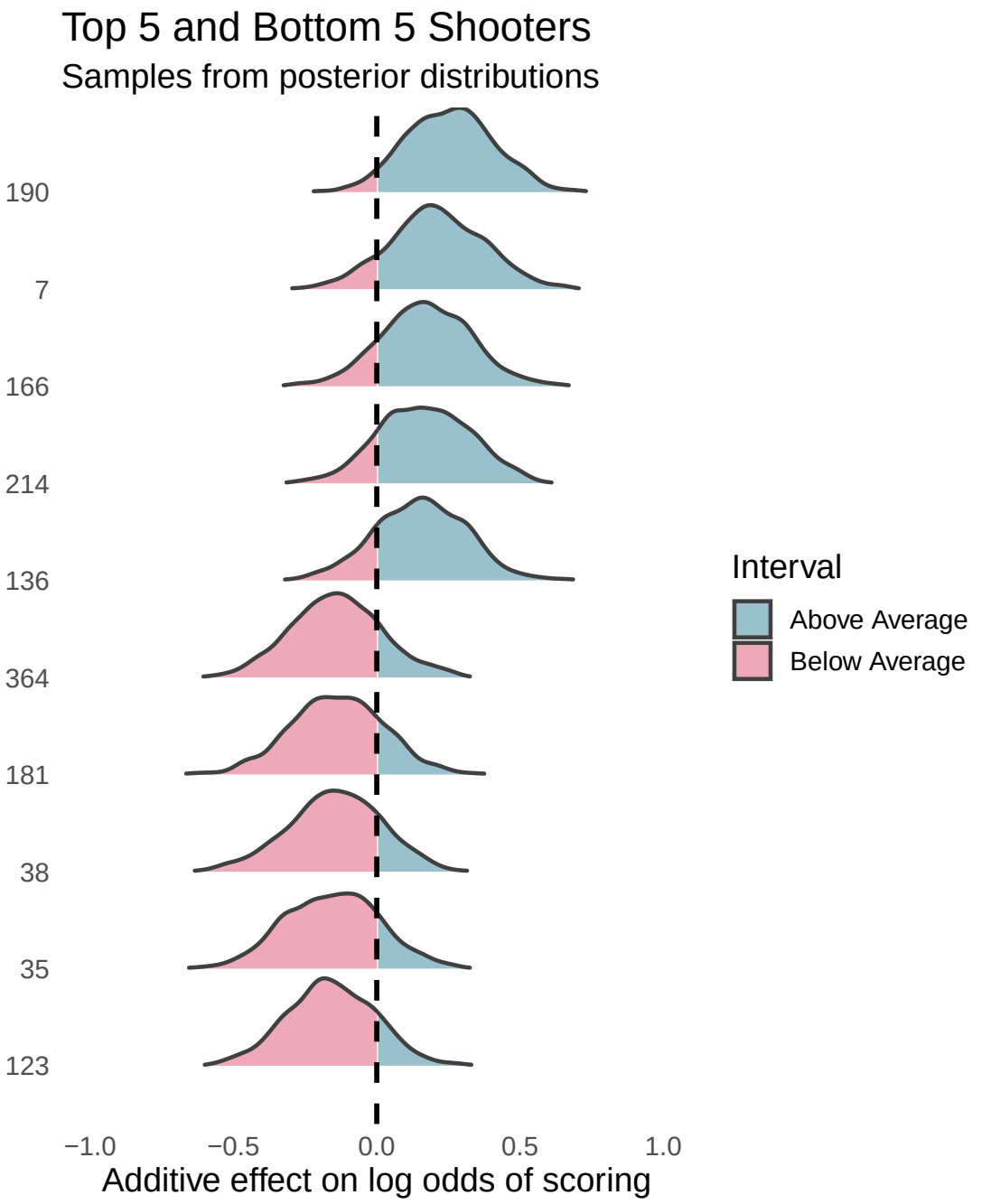
```
## Extract random effect estimates
shooters <- ranef(shots_mer_mod)$shooter_id %>%
  rename(eff = `(Intercept)`)

## Create a column to store the shooter id
shooters$shooter_id <- row.names(shooters)
```

```
## Get vector of shooter_ids of the top 5 and bottom 5
shooter_standouts <- c((shooters %>% arrange(desc(eff)))$shooter_id[1:5],
                       rev((shooters %>% arrange(eff))$shooter_id[1:5]))
```

Plotting the top shooters by their effects on the log-odds of scoring. The bulk of this messy code block is just retrieving shooter samples and smoothing the distribution over with a kernel density for plotting...

```
## Set minimal theme
theme_set(theme_minimal())

## Create a kernel density estimate for each set of posterior samples
dens1 <- kdensity((ranef(arm::sim(shots_mer_mod, n.sims = 1e3))$shooter_id)[,
  which(shooters$shooter_id == shooter_standouts[1]),1])
dens2 <- kdensity((ranef(arm::sim(shots_mer_mod, n.sims = 1e3))$shooter_id)[,
  which(shooters$shooter_id == shooter_standouts[2]),1])
dens3 <- kdensity((ranef(arm::sim(shots_mer_mod, n.sims = 1e3))$shooter_id)[,
  which(shooters$shooter_id == shooter_standouts[3]),1])
dens4 <- kdensity((ranef(arm::sim(shots_mer_mod, n.sims = 1e3))$shooter_id)[,
  which(shooters$shooter_id == shooter_standouts[4]),1])
dens5 <- kdensity((ranef(arm::sim(shots_mer_mod, n.sims = 1e3))$shooter_id)[,
  which(shooters$shooter_id == shooter_standouts[5]),1])
dens6 <- kdensity((ranef(arm::sim(shots_mer_mod, n.sims = 1e3))$shooter_id)[,
  which(shooters$shooter_id == shooter_standouts[6]),1])
dens7 <- kdensity((ranef(arm::sim(shots_mer_mod, n.sims = 1e3))$shooter_id)[,
  which(shooters$shooter_id == shooter_standouts[7]),1])
dens8 <- kdensity((ranef(arm::sim(shots_mer_mod, n.sims = 1e3))$shooter_id)[,
  which(shooters$shooter_id == shooter_standouts[8]),1])
dens9 <- kdensity((ranef(arm::sim(shots_mer_mod, n.sims = 1e3))$shooter_id)[,
  which(shooters$shooter_id == shooter_standouts[9]),1])
dens10 <- kdensity((ranef(arm::sim(shots_mer_mod, n.sims = 1e3))$shooter_id)[,
  which(shooters$shooter_id == shooter_standouts[10]),1])

## Specify the span to cover
span <- seq(from = -1, to = 1, by = 0.005)

## Make dataframe for plot
d <- data.frame(x = rep(span, 10),

                density = c(dens1(span), dens2(span), dens3(span),
                            dens4(span), dens5(span),
                            dens6(span), dens7(span), dens8(span),
                            dens9(span), dens10(span)),

                parameter = c(rep(shooter_standouts[1], length(span)),
                              rep(shooter_standouts[2], length(span)),
                              rep(shooter_standouts[3], length(span)),
                              rep(shooter_standouts[4], length(span)),
                              rep(shooter_standouts[5], length(span)),
                              rep(shooter_standouts[6], length(span)),
                              rep(shooter_standouts[7], length(span)),
                              rep(shooter_standouts[8], length(span)),
                              rep(shooter_standouts[9], length(span)),
                              rep(shooter_standouts[10], length(span))))
```

```r
d <- d %>% mutate(interval = ifelse(x <= 0, "Below Average", "Above Average"))
d$Interval <- as.factor(d$interval)
d$parameter <- factor(d$parameter, levels = rev(unique(d$parameter)))

## Plot with ggplot
ggplot(d) +
  aes(x = x, y = parameter, fill = Interval, height = density) +
  #theme_ipsum(base_size = 14) +
  ggridges::geom_density_ridges(stat = "identity", lwd = 0.8,
                                rel_min_height = 0.01, scale= 0.9,
                                col = "gray25") +
  geom_vline(xintercept = 0, col = "black", lty = 2, lwd = 1) +
  coord_cartesian(xlim = c(-1, 1)) +
  theme(text = element_text(size=14)) +
  scale_fill_manual(values = c("lightblue3", "pink2")) +
  labs(x = "Additive effect on log odds of scoring", y = "",
       title = "Top 5 and Bottom 5 Shooters",
       subtitle = "Samples from posterior distributions") +
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
          panel.background = element_blank())
```



Shooters 190, 7, 166, 214 and 136 were the model's highest rated shooters. Conversely, shooters 123, 35, 38, 181 and 364 were the worst. Amazingly, 17% of #5 ranked shooter 136's samples fell below average. Our lowest ranked shooter, 123, has a 15% probability of being better than league average. Perhaps most shockingly, there's a ~50% chance that at least one of our top 5-ranked shooters has below average shooting talent. These estimates do not inspire much confidence - even at the extremes.

Side note: In the `shots_mer_mod` fit, the league-wide distribution of shooting talent in the NHL was concurrently estimated. On the log-odds scale, NHL shooter effects are sampled from a Normal distribution with mean = 0 and std.dev = ~0.16. Using this fact, we can convert confidence intervals with log-odds units to percentiles (this is exactly what I did in my GM/coach friendly write-up, since I find it to be more intuitive and meaningful). See Appendix.

1.3 More Time?

A piece of information I haven't exploited yet is `location_on_net`. One idea is to incorporate this variable into the Expected Goals model (likely interacting it with shot location). Doing so would actually violate the standard definition of Expected Goals as we would be making predictions with post-release data. However, by utilizing such an approach, the marginal effect on goal probability associated with each shot placement category could be evaluated and for each individual shot, the predicted optimal shot target could be identified. Whether or not a shooter manages to hit this optimal target would act as our binary target variable. Our measure of shooting talent would no longer concern the scoring outcomes of shots, but instead a shooters success (or failure) in optimally placing shots on the net-face. The same work flow and methods used earlier in this question could be applied in the same way. Of course, one limitation is that this would only capture a part of the 'shot accuracy' component of finishing skill but it might still be an interesting supplemental piece of information.

Secondly, I would have used `gamm4::` instead of the post-hoc approach mentioned when implementing the shooter/goalie random effects.

1.4 More Data?

More observations, shot details and event data would allow for greater complexity when estimating goal probabilities. This would sharpen shooting talent estimates and better tease them apart from the influence of shot quality. With actual rebound data, would could also expand our definition of shooting talent to account for rebound creation.

Question 2

"Who are the best (and worst) goalies? How confident are you in your answer(s)?"

In question 1, I included a random effects term for goalies as well as shooters. In that case, they were more so a control variable given that the target involved missed & blocked shots - something they do not control. Nonetheless, it made sense to include them since they at the very least served as a proxy for identifying the opposing team, blended with a dash of goaltending talent.

To remedy this problem, we can just repeat the steps in question 1, but limit the data set to shots-on-goal.

2.2 A quick run through

Filtering the data down to shots-on-goal

```
## Removed blocked shots and missed shots
dat <- dat %>%
  filter(!(shot_result %in% c("Block", "Miss")))
```

Fitting a new xG model

```
## Keeping the same structure as in question 1
sog_mod <- gam(isGoal ~ 1 + shot_type + rush_situation +
                  player_strength + RebAngVel + s(dist) + s(angle),
                  data = dat, family = "binomial", method = "REML")
```

Adding random effects

```
## Add predicted SOG log-odds to the data
dat$sog_odds <- arm::logit(fitted(sog_mod))

## Fit a GLMM with lme4
sog_mer_mod <- glmer(data = dat, isGoal ~ 1 + offset(sog_odds) +
  (1|shooter_id) + (1|goalie_id), family = "binomial")

## Extract and simulate random effects
goalies <- merTools::REsim(sog_mer_mod, n.sims = 1e3) %>%
  ## Filter down to goalie estimates
  filter(groupFctr == "goalie_id") %>%
  ## Select ID, mean and sd
  select(groupID, mean, median, sd) %>%
  ## Compute 95% confidence intervals %>%
  mutate(lower_bound = mean - 1.96*sd,
         upper_bound = mean + 1.96*sd) %>%
  ## Sort by mean
  arrange(mean)
```

```
## Registered S3 method overwritten by 'broom.mixed':
##   method      from
##   tidy.gamlss broom
```

```
## View top 5 goalies
head(goalies, 5)
```

```
##   groupID        mean      median         sd lower_bound upper_bound
## 1       7 -0.17560779 -0.17586066 0.09536905  -0.3625311  0.01131554
## 2       6 -0.13004750 -0.13084245 0.10297613  -0.3318807  0.07178572
## 3      49 -0.09370196 -0.09498314 0.07851205  -0.2475856  0.06018167
## 4      40 -0.09188682 -0.09155611 0.08520046  -0.2588797  0.07510608
## 5      10 -0.08791980 -0.08848698 0.10755788  -0.2987332  0.12289366
```

Goalie IDs 7, 6, 49, 10 and 40 compose the 5 highest rated goalies. Keep in mind that a negative log-odds is good in this case. Not a single goalie has as estimate with a confidence interval that does not encapsulate the mean. In other words, league-average remains a feasible hypothesis for every single netminder in the data set.

For completeness, here are the bottom 5 goaltenders

```
## reverse order
goalies <- goalies %>%
  arrange(desc(mean))

head(goalies, 5)
```

```
##   groupID       mean    median         sd lower_bound upper_bound
## 1       4 0.16993598 0.1720805 0.09416633 -0.01463004   0.3545020
## 2      21 0.14588182 0.1467555 0.10257892 -0.05517286   0.3469365
## 3      15 0.10359009 0.1047815 0.08758788 -0.06808216   0.2752623
## 4      12 0.09879734 0.1003220 0.09497153 -0.08734685   0.2849415
## 5      11 0.07697575 0.0760621 0.11297028 -0.14444600   0.2983975
```

Goalies 4, 21, 15, 12 and 9 make up the list.

2.3 More Time?

With more time, I would have liked to investigate how the distribution of goalie effects differs based on shot placement. I would have also taken a crack at rebound control with my rebound proxy.

2.4 More Data?

A significant facet of goaltending talent is rebound control. With proper rebound data, we could evaluate individuals on their ability to prevent or create rebounds, convert that into goals and expand on the anaylsis done here.

Question 3

More Time?

I would have found the average number of shots faced by a goaltender in a season - call that number 'k'. I would have then sampled each goalie's random effect parameter estimate and simulated a season's worth of shot outcomes (goal vs. no-goal) across a random sample of k xG-values from the data set for each draw of their random effect parameter. I would repeat this process many times (200+) until I have a set of 'seasons' for each goalie. For a season i, I would return the top performing goalie (as measured by goals conceded vs. xgoals against) and find the individual who ranked 1st the most often across seasons $i = 1, 2, 3...200+$ and how often they managed to do so (it would likely be a very low %)

Appendix

Converting confidence intervals on the log-odds scale to percentiles

```
## Return median
goalies[["median_perc"]] <- pnorm(-goalies[["median"]], mean = 0,
                                  sd = 0.11) # 0.11 is population st.dev from
                                             # summary(sog_mer_mod)
## lower bound
goalies[["lb_perc"]] <- pnorm(-goalies[["upper_bound"]], mean = 0,
                                  sd = 0.11)

## upper bound
goalies[["ub_perc"]] <- pnorm(-goalies[["lower_bound"]], mean = 0,
                                  sd = 0.11)

head(goalies, 5)
```

```
##   groupID       mean    median         sd lower_bound upper_bound median_perc
## 1       4 0.16993598 0.1720805 0.09416633 -0.01463004   0.3545020  0.05886558
## 2      21 0.14588182 0.1467555 0.10257892 -0.05517286   0.3469365  0.09107881
## 3      15 0.10359009 0.1047815 0.08758788 -0.06808216   0.2752623  0.17040681
## 4      12 0.09879734 0.1003220 0.09497153 -0.08734685   0.2849415  0.18087947
## 5      11 0.07697575 0.0760621 0.11297028 -0.14444600   0.2983975  0.24463398
##        lb_perc   ub_perc
```

```
## 1 0.0006348418 0.5529034
## 2 0.0008053332 0.6920155
## 3 0.0061679864 0.7320184
## 4 0.0047935363 0.7864204
## 5 0.0033368290 0.9054330
```

Now for shooters

```
## Extract and simulate random effects
shooters <- merTools::REsim(shots_mer_mod, n.sims = 1e3) %>%
  ## Filter down to goalie estimates
  filter(groupFctr == "shooter_id") %>%
  ## Select ID, mean and sd
  select(groupID, mean, median, sd) %>%
  ## Compute 95% confidence intervals %>%
  mutate(lower_bound = mean - 1.96*sd,
         upper_bound = mean + 1.96*sd) %>%
  ## Sort by mean
  arrange(desc(mean))


## Return median
shooters[["median_perc"]] <- pnorm(shooters[["median"]], mean = 0,
                                   sd = 0.16196) # 0.11 is population st.dev from
                                            # summary(shots_mer_mod)
## lower bound
shooters[["lb_perc"]] <- pnorm(shooters[["lower_bound"]], mean = 0,
                                   sd = 0.16196)

## upper bound
shooters[["ub_perc"]] <- pnorm(shooters[["upper_bound"]], mean = 0,
                                   sd = 0.16196)

head(shooters, 5)
```

```
##   groupID      mean    median        sd lower_bound upper_bound median_perc
## 1     190 0.2521353 0.2496001 0.1513235 -0.04445887   0.5487294   0.9383564
## 2       7 0.2005520 0.2012425 0.1652174 -0.12327408   0.5243780   0.8929822
## 3     166 0.1798861 0.1797476 0.1585716 -0.13091429   0.4906864   0.8664632
## 4     214 0.1604427 0.1613569 0.1579168 -0.14907420   0.4699596   0.8404420
## 5     136 0.1597372 0.1660655 0.1541897 -0.14247465   0.4619490   0.8474008
##     lb_perc   ub_perc
## 1 0.3918482 0.9996480
## 2 0.2232870 0.9993975
## 3 0.2094554 0.9987759
## 4 0.1786719 0.9981443
## 5 0.1895135 0.9978294
```