Advanced topics

Multinomial logistic regression and multilevel models

July 20th, 2021

Example: NFL Expected Points

What does football **play-by-play** data look like? Each row is a play with contextual information:

- **Possession team:** team with the ball, on offense (opposing team is on defense)
- **Down:** 4 downs to advance the ball 10 (or more) yards
  - New set of downs, else turnover to defense
- **Yards to go:** distance in yards to advance
- **Yard line:** distance in yards away from opponent's endzone (100 to 0) - the field position
- **Time remaining:** seconds remaining in game, each game is 3600 seconds long
  - 4 quarters, halftime in between, followed by a potential overtime (900 seconds)

Example: NFL Expected Points

Drive: a series of plays, changes with possession and the types of scoring events:

- **No Score:** 0 points - turnover the ball or half/game ends
- **Field Goal:** 3 points - kick through opponent's goal post
- **Touchdown:** 7 points - enter opponent's end zone
- **Safety:** 2 points for opponent - tackled in own endzone

Next Score: type of next score (current drive or future drives) with respect to possession team

- For: Touchdown (7), Field Goal (3), Safety (2)
- Against: -Touchdown (-7), -Field Goal (-3), -Safety (-2)
- No Score

Note: treating point-after-touchdown attempts (PATs) separately

Example: NFL Expected Points

Expected Points: Measure the value of play in terms of $\mathbb{E}[\text{points of next scoring play}]$

- i.e., historically, how many points have teams scored when in similar situations?

Explanatory variables: $\mathbf{X} =$ {down, yards to go, yard line, ...}

Want to **estimate the probabilities** of each scoring event to compute expected points:

- Outcome probabilities: $P(Y = y|\mathbf{X})$
- Expected Points $= E(Y|\mathbf{X}) = \sum_{y \in Y} y \cdot P(Y = y|\mathbf{X})$

How do we model more than two categories???

Review: logistic regression

Response variable Y has two possible values: 1 or 0, we estimate the probability

$$p(x) = P(Y = 1|X = x)$$

Assuming that we are dealing with two classes, the possible observed values for Y are 0 and 1,

$$Y|x \sim \text{Binomial}(n = 1, p = \mathbb{E}[Y|x]) = \text{Bernoulli}(p = \mathbb{E}[Y|x])$$

To limit the regression betweewn $[0, 1]$: use the **logit** function, aka the **log-odds ratio**

$$\text{logit}(p(x)) = \log\left[\frac{p(x)}{1 - p(x)}\right] = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p$$

meaning

$$p(x) = \frac{e^{\beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p}}{1 + e^{\beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p}}$$

Multinomial logistic regression

We can extend this to K classes (via the softmax function):

$$P(Y = k^* \mid X = x) = \frac{e^{\beta_{0k^*} + \beta_{1k^*} x_1 + \cdots + \beta_{pk^*} x_p}}{\sum_{k=1}^{K} e^{\beta_{0k} + \beta_{1k} x_1 + \cdots + \beta_{pk} x_p}}$$

We only estimate coefficients for $K - 1$ classes **relative to reference class**

For example, let K be the reference then we use $K - 1$ logit transformations

- Use $\boldsymbol{\beta}$ for vector of coefficients and $\mathbf{X}$ for matrix of predictors

$$\log\left(\frac{P(Y=1|\mathbf{X})}{P(Y=K|\mathbf{X})}\right) = \boldsymbol{\beta}_1 \cdot \mathbf{X}$$
$$\log\left(\frac{P(Y=2|\mathbf{X})}{P(Y=K|\mathbf{X})}\right) = \boldsymbol{\beta}_2 \cdot \mathbf{X}$$
$$\log\left(\frac{P(Y=K-1|\mathbf{X})}{P(Y=K|\mathbf{X})}\right) = \boldsymbol{\beta}_{K-1} \cdot \mathbf{X}$$

Multinomial logistic regression for next score

$Y \in$ {Touchdown (7), Field Goal (3), Safety (2), No Score (0), -Safety (-2), -Field Goal (-3), -Touchdown (-7)}

$\mathbf{X} =$ {down, yards to go, yard line, ...}

Model is specified with **six logit transformations** relative to **No Score**:

$$\log\left(\frac{P(Y=\text{Touchdown}\,|\mathbf{X})}{P(Y=\text{No Score}\,|\mathbf{X})}\right) = \mathbf{X}\cdot\boldsymbol{\beta}_{\text{Touchdown}}$$

$$\log\left(\frac{P(Y=\text{Field Goal}\,|\mathbf{X})}{P(Y=\text{No Score}\,|\mathbf{X})}\right) = \mathbf{X}\cdot\boldsymbol{\beta}_{\text{Field Goal}}\,,$$

$$\vdots$$

$$\log\left(\frac{P(Y=-\text{Touchdown}\,|\mathbf{X})}{P(Y=\text{No Score}\,|\mathbf{X})}\right) = \mathbf{X}\cdot\boldsymbol{\beta}_{-\text{Touchdown}}\,,$$

- Model is generating probabilities, agnostic of value associated with each next score type
- Fit multinomial logistic regression model in `R` with `nnet` package

NFL play-by-play data (2010 to 2020)

Initialized NFL play-by-play dataset with next score in half for each play

- Followed steps in script by Ben Baldwin (which copies my steps here)

```
library(tidyverse)
nfl_ep_model_data <- read_rds(file = "http://www.stat.cmu.edu/cmsac/sure/2021/materials/data/mode

nfl_ep_model_data <- nfl_ep_model_data %>%
  mutate(Next_Score_Half = fct_relevel(Next_Score_Half, "No_Score"),
         # log transform of yards to go and indicator for two minute warning:
         log_ydstogo = log(ydstogo),
         # Changing down into a factor variable:
         down = factor(down))
```

How to fit the model?

```
init_ep_model <- multinom(Next_Score_Half ~ half_seconds_remaining + yardline_100 + down + log_yd
                          data = nfl_ep_model_data, maxit = 300)
```

What does the summary() function return?

Leave-one-season-out cross-validation

```r
library(nnet)
init_loso_cv_preds <-
  map_dfr(unique(nfl_ep_model_data$season),
          function(x) {
            # Separate test and training data:
            test_data <- nfl_ep_model_data %>% filter(season == x)
            train_data <- nfl_ep_model_data %>% filter(season != x)

            # Fit multinomial logistic regression model:
            ep_model <-
              multinom(Next_Score_Half ~ half_seconds_remaining + yardline_100 + down + log_ydstc

            # Return dataset of class probabilities:
            predict(ep_model, newdata = test_data, type = "probs") %>%
              as_tibble() %>%
              mutate(Next_Score_Half = test_data$Next_Score_Half,
                     season = x)
              })
```

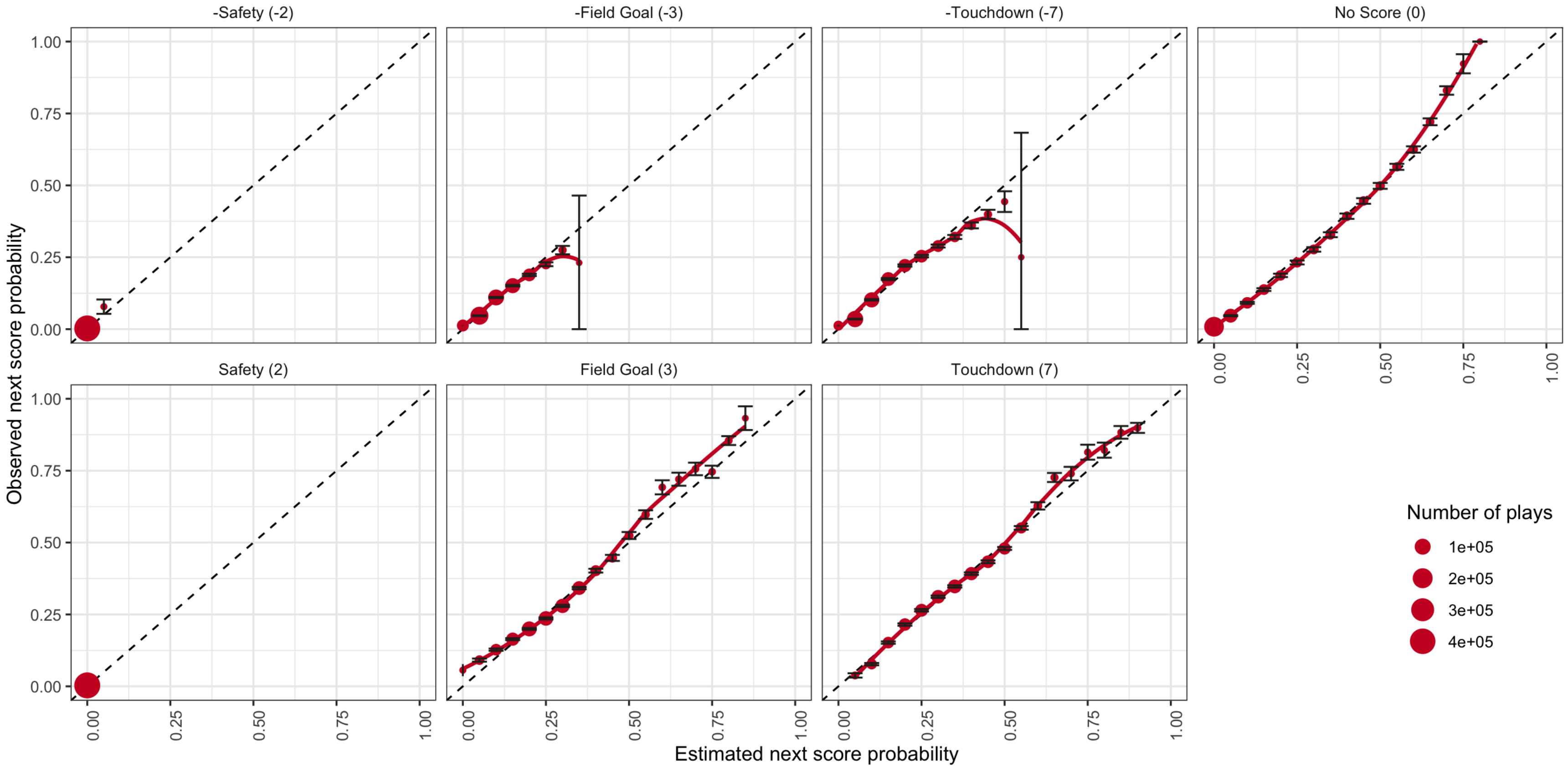
Calibration results for each scoring event

```
ep_cv_loso_calibration_results <- init_loso_cv_preds %>%
  pivot_longer(No_Score:Touchdown,
               names_to = "next_score_type",
               values_to = "pred_prob") %>%
  mutate(bin_pred_prob = round(pred_prob / 0.05) * .05) %>%
  group_by(next_score_type, bin_pred_prob) %>%
  summarize(n_plays = n(),
            n_scoring_event = length(which(Next_Score_Half == next_score_type)),
            bin_actual_prob = n_scoring_event / n_plays,
            bin_se = sqrt((bin_actual_prob * (1 - bin_actual_prob)) / n_plays)) %>%
  ungroup() %>%
  mutate(bin_upper = pmin(bin_actual_prob + 2 * bin_se, 1),
         bin_lower = pmax(bin_actual_prob - 2 * bin_se, 0))
```

Calibration results for each scoring event

```
ep_cv_loso_calibration_results %>%
  mutate(next_score_type = fct_relevel(next_score_type, "Opp_Safety", "Opp_Field_Goal",
                                       "Opp_Touchdown", "No_Score", "Safety", "Field_Goal", "Touc
  next_score_type = fct_recode(next_score_type, "-Field Goal (-3)" = "Opp_Field_Goal", "-Safety (
                               "Field Goal (3)" = "Field_Goal", "No Score (0)" = "No_Score",
                               "Touchdown (7)" = "Touchdown", "Safety (2)" = "Safety")) %>%
  ggplot(aes(x = bin_pred_prob, y = bin_actual_prob)) +
  geom_abline(slope = 1, intercept = 0, color = "black", linetype = "dashed") +
  geom_smooth(se = FALSE) +
  geom_point(aes(size = n_plays)) +
  geom_errorbar(aes(ymin = bin_lower, ymax = bin_upper)) + #coord_equal() +
  scale_x_continuous(limits = c(0,1)) +
  scale_y_continuous(limits = c(0,1)) +
  labs(size = "Number of plays", x = "Estimated next score probability",
       y = "Observed next score probability") +
  theme_bw() +
  theme(strip.background = element_blank(),
        axis.text.x = element_text(angle = 90),
        legend.position = c(1, .05), legend.justification = c(1, 0)) +
  facet_wrap(~ next_score_type, ncol = 4)
```

Calibration results for each scoring event



How do we evaluate players?

Expected points added (EPA): change in expected points between plays

Goal: divide credit between players involved in a play, i.e. who deserves what portion of EPA?

Load dataset of 2021 passing plays:

```
nfl_passing_plays <-
  read_csv("http://www.stat.cmu.edu/cmsac/sure/2021/materials/data/eda_projects/nfl_passing_plays
  # Only keep rows with passer and receiver information known:
  filter(!is.na(passer_player_id), !is.na(receiver_player_id), !is.na(epa)) %>%
  # Combine passer and receiver unique IDs:
  mutate(passer_name_id = paste0(passer_player_name, ":", passer_player_id),
         receiver_name_id = paste0(receiver_player_name, ":", receiver_player_id))
```

Data displays **group structure** and **different levels of variation within groups**

- e.g., quarterbacks have more passing attempts than receivers have targets

Every play is a **repeated measure of performance**

- i.e., the plays (observations) are NOT independent

Mixed-effects / random-effects / multilevel / hierarchical models

Example of a **varying-intercept** model:

$$EPA_i \sim Normal(Q_{q[i]} + C_{c[i]} + X_i \cdot \beta,\ \sigma^2_{EPA}),\ \text{for } i\ =\ 1, \ldots, n \text{ plays}$$

Groups are given a model - treating the levels of groups as similar to one another with **partial pooling**

$$Q_q \sim Normal(\mu_Q,\ \sigma^2_Q),\ \text{for } q = 1, \ldots,\ \text{number of QBs},$$
$$C_c \sim Normal(\mu_C,\ \sigma^2_C),\ \text{for } c = 1, \ldots,\ \text{number of receivers}$$

Each individual estimate (e.g., Q_q) is pulled toward it's group mean (e.g., μ_Q)

- i.e., QBs and receivers involved in fewer plays will be pulled closer to their overall group averages as compared to those involved in more plays
- serves as a form of **regularization** of coefficient estimates
- Q_q and C_c are **random effects**, while β are **fixed effects**
  - but these are confusing terms that no one agrees on

Fitting multilevel models with `lme4`

Include variables as usual but now introduce new term for varying intercepts: `(1|GROUP)`

```
library(lme4)
passing_lmer <- lmer(epa ~ shotgun + air_yards + (1|passer_name_id) + (1|receiver_name_id),
                     data = nfl_passing_plays)
summary(passing_lmer)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula:
## epa ~ shotgun + air_yards + (1 | passer_name_id) + (1 | receiver_name_id)
##    Data: nfl_passing_plays
##
## REML criterion at convergence: 63854.4
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -7.6812 -0.5542 -0.0227  0.5774  5.3139
##
## Random effects:
##  Groups           Name        Variance Std.Dev.
##  receiver_name_id (Intercept) 0.00316  0.05621
##  passer_name_id   (Intercept) 0.01050  0.10247
##  Residual                     2.33186  1.52704
```

Variance partition coefficients and intraclass correlations

We partition the variance in the response between the groupings in the data

Want to know the proportion of variance attributable to **variation within groups** compared to **between groups**

Can compute the variance partition coefficient (VPC) or intraclass correlation (ICC):

$$\hat{\rho}_Q = \frac{\text{Between QB variability}}{\text{Total variability}} = \frac{\hat{\sigma}^2_Q}{\hat{\sigma}^2_Q + \hat{\sigma}^2_C + \hat{\sigma}^2_{EPA}}$$

- Closer to 0: responses are more independent, the multilevel model structure is not as relevant
- Closer to 1: repeated observations provide no new information, multilevel group structure is important

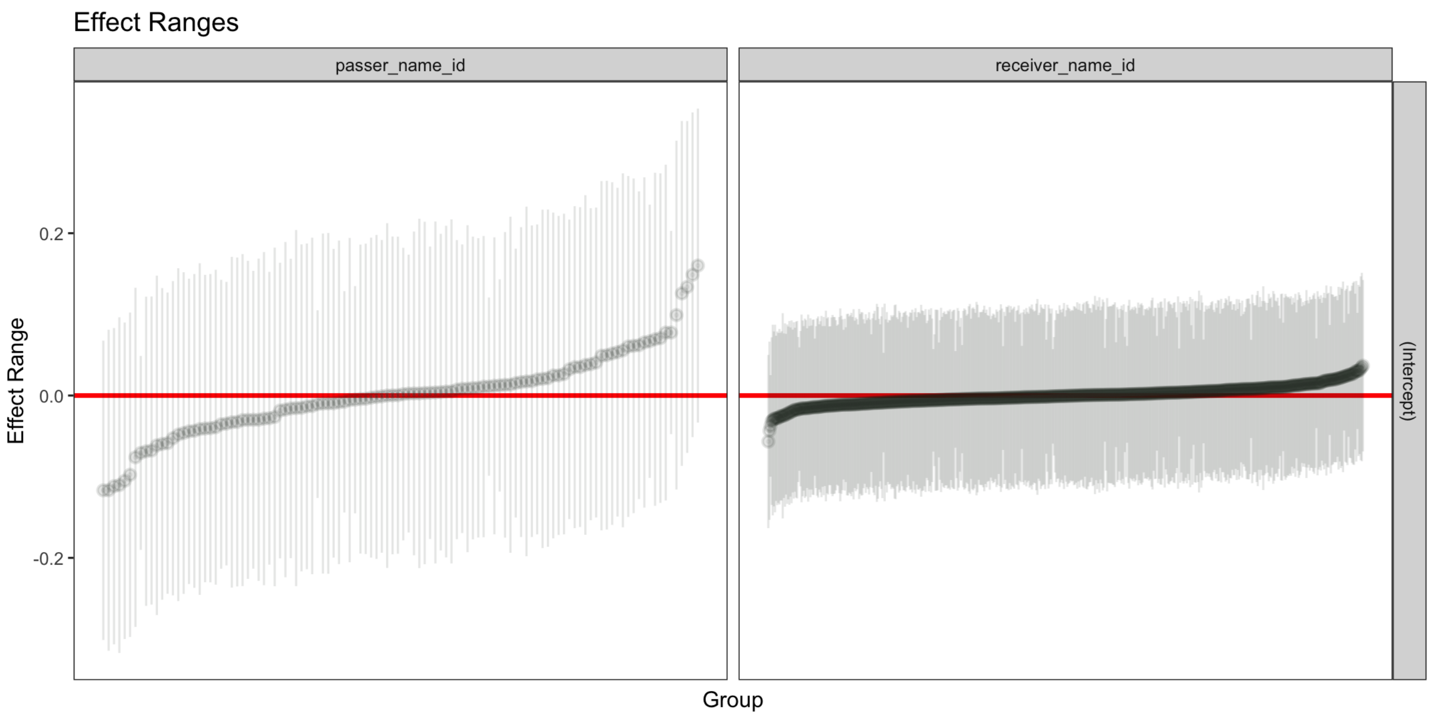
```
VarCorr(passing_lmer) %>% as_tibble() %>% mutate(icc = vcov / sum(vcov)) %>% dplyr::select(grp, 
```

```
## # A tibble: 3 × 2
##   grp                  icc
##   <chr>              <dbl>
## 1 receiver_name_id 0.00135
## 2 passer_name_id   0.00448
## 3 Residual         0.994
```

Exploring the player-level effects using `merTools`

Compare random effects with uncertainty via parametric bootstrapping

```
library(merTools)
player_effects <- REsim(passing_lmer)
plotREsim(player_effects)
```



Best and worst players? (by effects)

```
player_effects %>%
  as_tibble() %>%
  group_by(groupFctr) %>%
  arrange(desc(mean)) %>%
  slice(1:5, (n() - 4):n()) %>%
  ggplot(aes(x = reorder(groupID, mean))) +
  geom_point(aes(y = mean)) +
  geom_errorbar(aes(ymin = mean - 2 * sd,
                    ymax = mean + 2 * sd)) +
  facet_wrap(~groupFctr, ncol = 1, scales = "
  geom_vline(xintercept = 0, linetype = "dash
             color = "red") +
  coord_flip() +
  theme_bw()
```