Regression to the Mean: A Simulation of Football Players

Business Reseach Methods class notes

Miguel Salema

May 20, 2024

Simulate the Data

We start by loading the `tidyverse` package, which provides tools for data manipulation and visualization. Next, we set a random seed (8152) to ensure that our simulation results are reproducible. We then simulate the '*skill*' of 1,000 soccer players using a log-normal distribution and create a dataset (`df`) that records each player's performance across 10 matches, resulting in 10,000 observations. For each match, we add a random error to the player's skill to calculate the number of goals scored, rounding to the nearest whole number. To ensure that the number of goals cannot be negative, we replace any negative values with zero.

```
##### Load the packages ------------------------------------------------####

library(tidyverse)
library(kableExtra)
library(scales)

##### Simulation ------------------------------------------------------####

# NOTE: Run it all at once for the seed to apply to every function
set.seed(8152)

# Randomly assign a skill to 1000 players using a log-normal distribution
x <- rlnorm(1000, sd = 0.5)
# hist(x, breaks = 100)

# Create a tibble with a player ID, the match, skill and an error
df <- tibble(player =  rep(1:1000, 10),
```

```
              skill = rep(x, 10),
              match = rep(1:10, each = length(x)),
              error = rnorm(1:10000, sd = 2))

# create the outcome variable
df <- df %>%
  mutate(goals = round(skill + error))

# right censor (no negative goals)
df <- df %>%
  mutate(goals = ifelse(goals < 0, 0, goals))
```

This simulation setup allows us to analyze regression to the mean in soccer performance by considering both inherent player abilities and the variability in match outcomes. For player i in game number t, we are modelling the number of goals as:

$$goals_{it} = skill_i + luck_{it}, \tag{1}$$

where $skill_i$ depends only of the player and $luck_{it}$ is the amount of luck a player has in a given game. $luck_{it}$ has an expected value of zero, because it represents the residual error we introduced.[1] Consequently, we can express the expected number of goals of player i in game t as:

$$\mathbb{E}[goals_{it}] = \mathbb{E}[skill_i] + \mathbb{E}[luck_{it}] = \mathbb{E}[skill_i]. \tag{2}$$

Therefore, to find out how many goals a player is expected to score, we must discover his or her skill.

Explore the Data

We can select data for a specific player, in this case, player number 548, from our simulated dataset. We can see that we have panel data, since this player appears 10 times (the number of games). Furthermore, notice how skill is fixed: it does not change from game to game.

```
# View(df)

# chose one player to follow
df_p_548 <- df %>%
  filter(player == 548) %>%
  mutate(across(names(df), round, 2))
```

[1] However, because we applied right-censoring to ensure that goal counts cannot be negative, the average error in our sample is not exactly zero, unlike its true value.

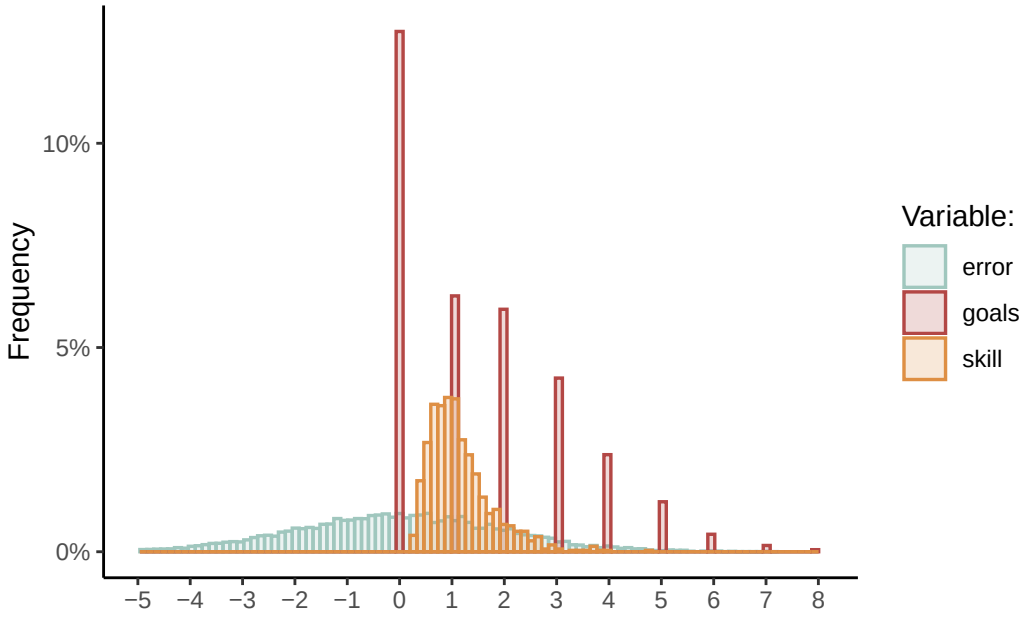
```r
# Print a small table:
df_p_548 %>%
  kbl(booktabs = T,
      linesep = "") %>%
  kable_styling(latex_options = c("hold_position"),
                position = "center") %>%
  column_spec(5, color = "white",
              background = spec_color(df_p_548$goals,
                                      palette = c("#b34745", "#de8f44", "#A0C7BE")))
```

player	skill	match	error	goals
548	0.78	1	-3.47	0
548	0.78	2	-0.43	0
548	0.78	3	2.38	3
548	0.78	4	0.45	1
548	0.78	5	3.15	4
548	0.78	6	-4.61	0
548	0.78	7	-0.61	0
548	0.78	8	0.91	2
548	0.78	9	-1.51	0
548	0.78	10	2.57	3

Next, we use `ggplot` to do a histogram of the distributions for the variables skill, error and goals. notice how only the error has negative values and a bigger variance than skill. Moreover, the variable goals is clearly discrete.

```r
df %>%
  pivot_longer(c(skill, goals, error),
               names_to = "type",
               values_to = "value") %>%
  ggplot(aes(x = value,
             fill = type,
             color = type)) +
  # Histograms with fill colors
  geom_histogram(aes(y = after_stat(count)/sum(after_stat(count))),
                 bins = 100,
                 alpha = 0.2,
                 position = "identity",
                 linewidth = 0.6) +
  scale_y_continuous(labels = scales::percent) +
  scale_fill_manual(values = c("#A0C7BE", "#b34745", "#de8f44")) +
```

```
  scale_color_manual(values = c("#A0C7BE", "#b34745", "#de8f44")) +
  scale_x_continuous(limits = c(-5, 8.1),
                     breaks = seq(-5, 8, by = 1)) +
  labs(x = "",
       y = "Frequency",
       color = "Variable:",
       fill = "Variable:") +
  theme_classic()
```



The Best Players of Match 1

We filter the dataset to include only the first match and select the top seven goal scorers using `slice_max`. Next, we extract the player IDs of these top performers and store them in the `top_m1` vector. This enables us to track and analyze the subsequent performances of these specific players to observe regression to the mean.

```
# go to the first period, see the top 7
df_top_1st_period <- df %>%
  filter(match == 1) %>%
  slice_max(goals, n = 7)
```

```r
# keep the IDs of the players stored in a vector
top_m1 <- df_top_1st_period$player
```

Compare the best players of match 1 to their performace in match 2

To test whether top-performing players in the first match perform worse in the second match, we filter the dataset to include only the players in the `top_m1` vector and their performances in matches 1 and 2. We then reshape the data from a long to a wide format using `pivot_wider`, creating separate columns for the error and goals in each match.

```r
df_top_1st2nd <- df %>%
  filter(player %in% top_m1 & match %in% c(1, 2)) %>%
  pivot_wider(names_from = match,
              values_from = c(error, goals),
              names_sep = "_t")


# Print a small table:
df_top_1st2nd %>%
  mutate(across(names(df_top_1st2nd), round, 2)) %>%
  kbl(booktabs = T,
      linesep = "") %>%
  kable_styling(latex_options = c("hold_position"),
                position = "center") %>%
  column_spec(5, color = "white",
              background = "#A0C7BE") %>%
  column_spec(6, color = "white",
              background = spec_color(df_top_1st2nd$goals_t2,
                                      palette = c("#b34745", "#de8f44", "#A0C7BE")))
```

player	skill	error_t1	error_t2	goals_t1	goals_t2
168	2.61	4.35	3.73	7	6
301	2.45	6.03	1.77	8	4
593	1.86	6.56	-1.90	8	0
628	0.88	6.15	-2.61	7	0
813	2.71	4.27	-0.46	7	2
923	2.08	5.32	-1.85	7	0
991	0.27	6.97	1.01	7	1

You are watching the second match of player 593 on TV and the commentator says the following:

> "I already expected 593 to do a poor match. It's only normal. After an excelent performace, an athlete will do worst because they get nervous."

Given the previous table, do you confirm this theory?

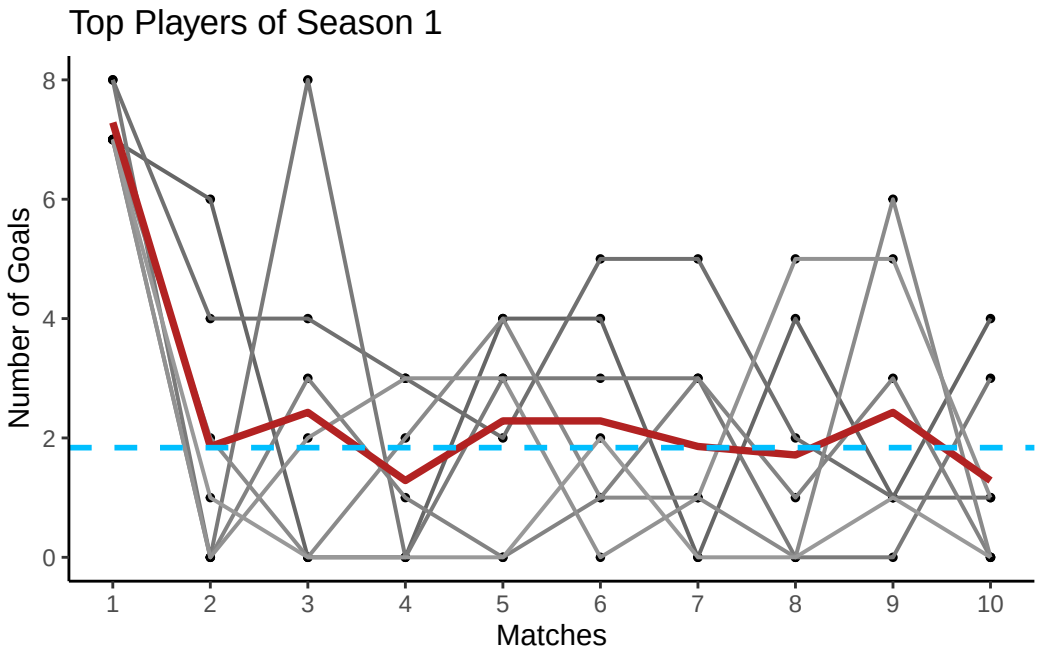
Follow the players' average over time

Finally, we create a plot to visualize the goal-scoring performance of the top seven players from the first match across all ten matches. First, we calculate the average number of goals scored per match by these players and store it in `df_av_goals`. Then, using `ggplot2`, we plot each player's goals over the matches in grey, the average goals per match in red and add a horizontal dashed line representing the overall average skill level of the top players.

```
# average goals per season
df_av_goals <- df %>%
  filter(player %in% top_m1) %>%
  group_by(match) %>%
  summarise(goals = mean(goals)) %>%
  ungroup()

# we  can do a plot using two dataframes!
df %>%
  filter(player %in% top_m1) %>%
  ggplot(aes(x = match,
             y = goals)) +
  geom_point(size = 1) +
  geom_line(aes(color = as.factor(player)),
            linewidth = 0.7) +
  scale_color_grey(start = 0.4, end = .6) +
  scale_x_continuous(breaks = 1:10) +
  geom_line(color = "firebrick",
            data = df_av_goals,
            linewidth = 1.3) +
  # they stability at the qualkity mean
  geom_hline(yintercept = mean(df_top_1st_period$skill),
             color = "deepskyblue",
             linetype = "dashed",
             linewidth = 1) +
```

```
labs(x = "Matches",
     y = "Number of Goals",
     title = "Top Players of Season 1") +
theme_classic() +
theme(legend.position = "none")
```



This visualization demonstrates that, after the initial match, the average number of goals scored by these players tends to converge towards their average skill level, illustrating regression to the mean. In the first match, the selected top performers exhibit higher goal counts due to both their inherent skill and the higher variability (error) associated with extreme performances. However, in subsequent matches, the influence of random fluctuations diminishes, and performances stabilize around their true skill levels. This pattern confirms the concept that exceptionally good performances are often followed by more typical performances, not necessarily because the athletes become nervous, but because their extraordinary initial results were partly due to favorable randomness.