FIFA PLAYER DATA TO PREDICT POINTS IN LA LIGA

ANTONIA GEORGE
MIGUEL CONNER

Project for Statistical Methods and Inference
BARCELONA SCHOOL OF ECONOMICS

Barcelona School of Economics

End of Semester Project

20/12/2022

Supervisors:

Geert Mesters
David Rossell

Table of Contents

Chapter 1

Introduction

1.1 Introduction

The aim of this study was to predict the number of points a team would score at the end of the season of the Spanish Primera Division. To achieve this goal, we used data from the video game FIFA. The game contains player attributes from 2015 to 2022. In part one, a LASSO regression model and a Bayesian regression model were used to obtain predictions for the 2022 season outcome. Bayesian model selection was used to assess which features were most important to the overall outcome. In part two, a hierarchical model was used to take into account that our data included players within teams. A hierarchical model was also used to predict player wages.

1.2 Related Work

We examined several papers which used a variety of methodologies to approach problems that were similar to ours. There is plenty of interest in predicting sports outcomes, especially as online sports betting becomes more popular and accessible across the world. Many applications using sports data are aimed at predicting future outcomes such as points, scores, match outcomes, or league rankings. Our research revealed that Bayesian models are becoming more and more popular for these sports outcome studies on the outcomes of sporting events. In particular, a few of the studies we examined used Bayesian regression models and Monte Carlo Markov Chain approaches to predict future outcomes[3][2]. Additionally, Bayesian hierarchical models are also commonly used, as we saw in a paper that was aimed at predicting football match outcomes[1]. Another paper we looked at used ANOVA, k-means clustering, multidimensional scaling, and decision tree analysis to analyze win/draw/loss outcomes for football matches[5]. We also examined a literature review[6] that analyzed different methods used for sports data analysis. It outlined that the top models used for sports analysis in the last several years are Bayesian hierarchical models, Bayesian regression, and empirical Bayesian approaches. Models accounting for time, such as time series, were also among the more commonly used approaches. In our opinion, the study by Parim et al. was the most interesting and advanced study that we observed. The study took in more complex and situational variables such as aerials won, whether a team scored

first, and home-field advantage. This data allowed for a more comprehensive look at team performance on a game-to-game basis, and allowed greater efficacy of predictions on the individual game level.

1.3 Dataset

Our dataset included 52 features, which include different measures used to score players in FIFA from 2015-2021. The independent variables used are:

```
 [1] "overall"                   "potential"                 "value_eur"                 "wage_eur"
 [5] "age"                       "height_cm"                 "weight_kg"                 "weak_foot"
 [9] "skill_moves"               "international_reputation"  "release_clause_eur"        "pace"
[13] "shooting"                  "passing"                   "dribbling"                 "defending"
[17] "physic"                    "attacking_crossing"        "attacking_finishing"       "attacking_heading_accuracy"
[21] "attacking_short_passing"   "attacking_volleys"         "skill_dribbling"           "skill_curve"
[25] "skill_fk_accuracy"         "skill_long_passing"        "skill_ball_control"        "movement_acceleration"
[29] "movement_sprint_speed"     "movement_agility"          "movement_reactions"        "movement_balance"
[33] "power_shot_power"          "power_jumping"             "power_stamina"             "power_strength"
[37] "power_long_shots"          "mentality_aggression"      "mentality_interceptions"   "mentality_positioning"
[41] "mentality_vision"          "mentality_penalties"       "mentality_composure"       "defending_marking_awareness"
[45] "defending_standing_tackle" "defending_sliding_tackle"  "goalkeeping_diving"        "goalkeeping_handling"
[49] "goalkeeping_kicking"       "goalkeeping_positioning"   "goalkeeping_reflexes"      "goalkeeping_speed"
```

Our pre-processing included removing some features that are not relevant and engineering one variable (player position). Afterwards, we re-scaled the numerical variables. In our regression problems, we aggregated individual player data to the team level by computing the mean of the attribute among all the players of a team for a given year. Our training sample size is n = 140, which includes a row for each team's aggregated attributes for each year from 2015-2021 (7 years). Our reasoning for using this data is that each year, a new FIFA game is released and the metrics for each player attribute are determined based on actual performance of that player in real life from the previous season. These statistical attributes are re-assessed every year and are updated according to each player's performance and ability. We are using these player attributes as a proxy to assess a team's ability in real life by aggregating the player attributes to the team level.

In addition to the FIFA video game data, we separately scraped the end of season points from 2015-2022 for each team in our data set. In addition to our training data set, we included a testing data set that had the same features from the training set, but for the 2022 season. Our sample size for the testing data set is n = 20, where each row was a different team. Our dependent variable for both the training and the testing data set was the number of points each team had at the end of each season. We compared the testing set predictions with the actual points the team achieved in the 2022 season.

The FIFA dataset was from Kaggle[4] and the La Liga data end-of-season data (our observed values) was taken from Wikipedia.

Chapter 2

Regression

In this chapter we will look at different kinds of regression in our prediction problem. Our goal is to predict how many points a team will have at the end of the season. We will start by averaging the stats of each player on a team, and do a regression at the team level. In the next chapter, we will look at another way of making a prediction at the player level.

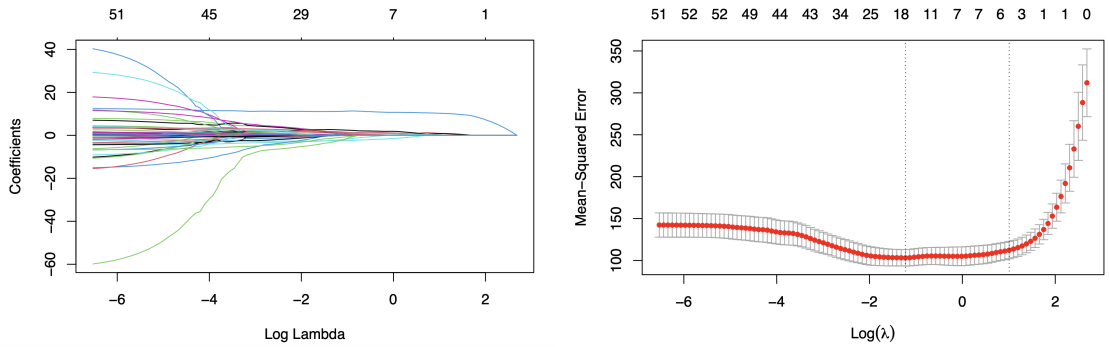
2.1 Lasso Model

We will begin our analysis with a LASSO regression, motivated by the fact that we have over 50 variables for each team and we want to know which are most important. Using a LASSO regression allows us to penalize including additional variables into the model. This type of method is used to encourage simpler models. Of course, L_1 will not give us our most accurate prediction, but it will give us a sense of what's going on under the hood so that we can make predictions with other methods. The results of our LASSO regression are shown in Fig. (2.1), and gave us a RMSE of 6.27.

In the LASSO regression, we found one very important variable: international reputation. This is measured on a scale of 1 to 5, so it is a variable that summarizes many of the other variables, so it is no surprise that it came out on top. We also saw that dribbling, age, and release clause amount were all smaller, but negatively correlated. These are sensible: you want a team that doesn't have dribblers who tend to lose the ball, and younger players usually have the advantage of being faster. Finally, we see the model also selects a few skill and physical attributes. In the end, this model was shockingly close given how simple it was.

2.2 Bayesian Regression Model

In this section we use a Bayesian Regression model approach to incorporate prior information about our parameters and we use this information to derive posterior distributions. We chose to use a Bayesian approach because this allows us to make more robust inferences. We will compare the predictions obtained by the Bayesian Regression model with the predictions of our lasso model. We used the rstanarm package to fit a Bayesian Regression model using the stanglm function. The response variable is gaussian, so we specified the family



	word	beta_hat
1	intercept	52.17143
2	international_reputation	11.12791
3	dribbling	-2.70233
4	movement_reactions	2.58930
5	age	-2.48630
6	release_clause_eur	-2.22120
7	attacking_heading_accuracy	1.68062
8	goalkeeping_positioning	1.37290
9	value_eur	1.20551
10	attacking_finishing	1.15722
11	goalkeeping_reflexes	-1.12758
12	power_stamina	0.97659
13	defending_marking_awareness	0.91311
14	movement_agility	0.76300
15	power_jumping	0.40762
16	height_cm	0.19937
17	defending_sliding_tackle	-0.15589
18	mentality_composure	-0.10865
19	weight_kg	-0.01496
20	overall	0.00000

Figure 2.1: *Plots showing the values of the regression coefficients as lambda changes (left), and the change in MSE as lambda increases (right). Below: coefficients from our Lasso regression.*

parameter to be gaussian. The model was fit via (Hamiltonian) MCMC, the default for the algorithm parameter. We obtained point estimates, medians of each variable estimate computed from simulation, and uncertainty estimates for our model. The uncertainty estimates are the standard deviations reported, labeled MADSD, computed from the same set of simulation draws. We can see from the full results of the analysis that some of the most important features according to the Bayesian Regression model were international reputation, pace, defending, skill dribbling, and movement-agility. It is difficult, though, to make explicit conclusions about feature importance because our data may have multicollinearity, which could be weakening the effect of our point intervals. We cannot say for certain which variables are important if there is too high of a correlation between the variables.

Using our testing set, we used our Bayesian Regression model to predict the outcomes of the 2022 season and compared the results with the actual results. The results are shown in the table below, comparing the actual points scored in 2022 to our Bayesian model's prediction of how many points that team will score. We can see that our model did a pretty

good job at predicting the outcomes of the 2022 season. The correlation between the actual points scored and the predicted points was around 0.82. We obtained a root mean squared error (RMSE) of 8.835 for the testing set predictions. Comparing this to the measure obtained from the lasso model, which was an RMSE of 6.27, we see that the lasso model had a lower root mean squared error for the predictions for the testing set.

	club_name	Pts.	predictions_bayes
1	Athletic Club de Bilbao	55	56.66665
2	Atlético de Madrid	71	73.87399
3	CA Osasuna	47	46.14914
4	Cádiz CF	39	42.45438
5	Deportivo Alavés	31	46.35545
6	Elche CF	42	49.97121
7	FC Barcelona	73	75.04346
8	Getafe CF	39	45.18942
9	Granada CF	38	44.71812
10	Levante Unión Deportiva	35	46.97023
11	Rayo Vallecano	42	51.01301
12	RC Celta de Vigo	46	43.19493
13	RCD Espanyol de Barcelona	42	38.59277
14	RCD Mallorca	39	43.86118
15	Real Betis Balompié	65	50.99236
16	Real Madrid CF	86	65.09661
17	Real Sociedad	62	57.34609
18	Sevilla FC	70	61.47239
19	Valencia CF	48	55.44545
20	Villarreal CF	59	48.99848

Figure 2.2: *Bayesian Regression model point estimates and standard deviations.*

2.3 Bayesian Model Selection

We use Bayesian model selection to choose the best statistical model from a set of candidate models using Bayesian probability theory. By comparing the relative plausibility of different models given our dataset, we can gain a better understanding of which model is most plausible. We performed a Bayesian Model Selection method on our data using the mombf package. We used Zellner's prior and a Beta-Binomial(1,1) prior as we have a high dimensional dataset. By comparing the relative performance of different models, we can get a better idea of which models are more or less likely, given our data. We can see that the model with the highest posterior probability is the one with only column 11 included. This is the most likely model according to the Bayesian Model Selection process, and the posterior probability is 0.611. The second most likely model is the one with columns 11 and 35, and the third most likely is the model with columns 11 and 21. We can see that the Bayesian Model Selection method is putting a high likelihood on column 11 being in the model. In fact, column 11 is included in all of the top 10 models with highest posterior probabilities. We use this analysis to gain a deeper understanding of which variables are most important to the model. We note here that there are limitations to this type of analysis. In particular, we expect many of our variables to be highly correlated. When we obtain different statistics or measures for the same player, we expect that better players will have better statistics across the board. Additionally, some variables such as wage, release clause amount, value, and international reputation would likely be quite correlated. High correlations between multiple features in our models leads us to be wary of making concrete

conclusions on which features are actually in the model.

	modelid	family	pp
1	11	normal	0.611394250
2	11,35	normal	0.054748127
3	11,21	normal	0.038237312
4	6,11	normal	0.023953269
5	11,36	normal	0.016248735
6	6,11,21	normal	0.012387920
7	3,11	normal	0.011155557
8	11,32	normal	0.010848025
9	11,12	normal	0.008034734
10	11,40	normal	0.007273579

Figure 2.3: *Bayesian model selection posterior probabilities.*

Chapter 3

Hierarchical Modeling

3.1 Going to the Player Level

In football we are dealing with a team of 11 players, as well as a whole set of substitutes. If our goal is to predict the results of an entire team using player data, the simplest method would be to take the mean of the teams stats and compare the teams by means of player data. This is what we did in the first part of the analysis, and indeed it provided some reasonable results. However, we are losing some very valuable information when we take the mean of stats for each team. For example, if you have a Leo Messi on your team, he might have a bigger impact than his fractional contribution to player mean. So the questions is: how can we estimate a teams points based on individual players, while at the same time acknowledging that many of the players play together on the same team? Let's try a **Hierarchical Model**.

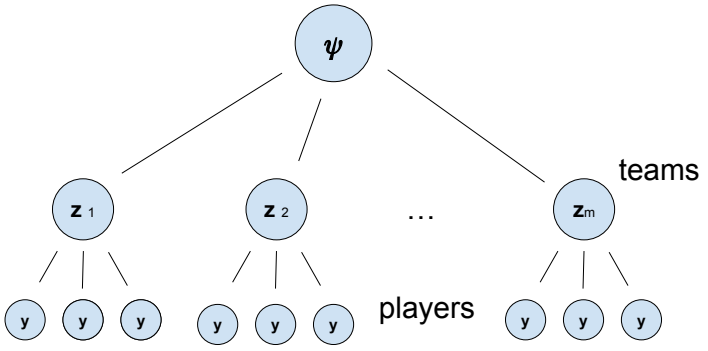


Figure 3.1: *A diagram of our hierarchical model.*

3.1.1 Our Points Model

A hierarchical model allows us to include different "levels" in our analysis. Observations in our dataset are not independent of each other because players are a part of the overall team. We use a hierarchical model to account for the fact that different teams have different

Table 3.1: *Different hierarchies used in our model with the associated RMSE of the predictions with the actual team points for the 2022 season.*

RMSE	hierarchical model layout
12.17	pos
7.52	club_name
7.46	club_name:pos

Table 3.2: *Table showing the different hierarchies used in our model with the associated RMSE of the predictions with the actual wages for the 2022 season.*

RMSE	hierarchical model layout
24591.14	club_name
21600.1	club_name:pos
21468.42	club_name + club_name:pos

characteristics that could affect the performance of a player on that team. Fig.(3.1) shows the traditional layout of a hierarchical model, where we have our observations y_{ij} (one for each player), grouped by z_i categories (one for each team).

We will again use the 2022 predictions as a measure of error. We see our results in Table (3.1), showing a clear preference for using a hierarchical model with team, and a stronger preference for nesting that with the player's position.

However, in solving this problem, we have come upon a realization: perhaps a hierarchical model is not the best answer to this problem. We need a single value for an entire team of players, and this model is most suited for different values for each player. This will be further discussed in the next chapter.

3.2 Predicting Player Wages

While our application of the hierarchical model to player data was quite ... novel, let us now shift to a more appropriate application of such a model: how much we should pay a player.

Player wage is a much better fit than team points because unlike team points, a wage is inherently different for every player. What team the player plays for, and what position the player plays should also make a big difference in their perceived value in the market. So we will try to make a model that has a team hierarchy and then within each team has a player position hierarchy (defense, midfield, striker, goalkeeper, and substitute).

3.2.1 Our Wage Model

We will focus on adjusting the team, and the position as the hierarchical variables, and compare how the different models perform on our 2022 season test set. In addition, we will be using the `age`, `international reputation`, and `overall` as the other predictor variables, as they have been selected as relevant variables for this variable. Our results are presented in Table (3.2). We achieved an RMSE of 21468.42 in our best model.

Chapter 4

Discussion

4.0.1 Regression

To summarize, we performed and analyzed several different models in order to examine how player data from FIFA video game measures could be used to predict actual league outcomes. Due to the large number of features in our dataset, we used a LASSO and Bayesian Regression model to obtain coefficient estimates and point intervals. We assessed these models' predictive accuracy by obtaining predictions on the testing data set and we found that the LASSO model performed better than the Bayesian regression model to obtain future predictions.

We also performed a Bayesian model selection to get a sense of which features seemed most important and found that the release clause variables was an important variable in all ten of the models with highest posterior probability. We also discussed the limitations to making conclusions about feature importance using regression because our data has multi-collinearity.

4.0.2 Linear Hierarchical Models

In implementing the hierarchical model we tried two different problems, that used similar hierarchical structures. In both, we found that hierarchical models reduced the error than when we made the same predictions without those hierarchies. This fits with our intuition: We might expect a player that plays for F.C. Barcelona will probably be different from a player on a smaller market team. We would also expect forwards and other players that score more goals also steal the spotlight, so positions are also relevant. So our model results fit with our intuition.

Now, let us discuss the applicability of hierarchical models to the problems discussed. In the first problem, we are computing a number of points for a player, that corresponds to the end of the teams points at the end of the season. If we average all the results and group by team, we get a RMSE of 7.50, which is not as good as our LASSO model, for example. But what actually is this metric?

Let's take a look at the difference between the actual values and the predicted result of this mysterious model. A large negative value seems to imply that we expected much better for that player. We then can interpret players who have a large negative difference as

Table 4.1: *Table showing biggest differences in predicted points for a player versus actual points for the team from our hierarchical model.*

Team	Act. Pts.	Pred. Pts.	Player	Difference
Levante UD	35	67.29	S. Mustafi	-32.29
Atlético Madrid	71	100.34	J. Oblak	-29.33
...	...	...	...	...
Real Madrid	86	48.77	Luis López	37.22
Real Madrid	86	48.70	Salazar	37.30

under performing. Given their stats, we expect them to do well, but the are brought down by the quality of their team. Many of these are "big" players on smaller teams. Or maybe they had an injury, and their team suffered because they weren't there, as was the case for Shkodran Mustafi, the player who had the largest positive difference.

We can also consider that maybe players are overvalued in the video game, after all, humans have to assign these ratings. In the case of J. Oblak, he is predicted as having 100 points! He played well in the 2020-2021 season and his team won the league that season, which surely influenced his high rating. But this season he and his team were unable to live up to that level, and so we see such a high prediction that is much closer to last seasons (86) than this seasons.

On the other side, we see players who have much lower scores as being carried by their team. For example, Real Madrid has the 14 players with the biggest difference between predicted points and actual points. In looking at the data, most of these players are substitutes, and very young. They did not participate much in the matches, and as a result their actual stats have very little value in predicting the total points.

This brings to light a major flaw in the analysis: we are looking at player data of players who aren't playing many of the games. It definitely makes sense to include the statistics of substitutes in the team (since a team's "depth" is what a major component in overcoming injuries and grueling schedules), but maybe there is a better way to weight their value.

However, with our dataset, we run into a limitation of this model. In part one, we were trying to predict total points the team would score at the end of the season. So, if we are trying to predict team points, we will set our prediction as total points. This is a little unusual for a hierarchical model in that all members of a particular team will have the same number of points! Our data does not include the points a player scored individually, so we cannot use this hierarchical model to obtain the predictions of a particular player's points scored. We can run the hierarchical model, but because the players on the same team all have the same number of points (the points the team overall scored), this defeats the purpose of running a hierarchical model to obtain the effect of player's ability, given the team they are on. For this reason, we decided to use the hierarchical model to study the predicted wage.

For future work, we would like to implement a hierarchical model that takes into account player ability and the effect of the team to predict the goals that player would score. We would obtain historical data for the individual players' goals each year to be able to study this effect better. Another application to our analysis could be to obtain data on individual matches and attempt to model match outcomes based on player within team data.

Bibliography

[1] Gianluca Baio and Marta Blangiardo. Bayesian hierarchical model for the prediction of football results. *Journal of Applied Statistics*, 37(2):253–264, Jan 2010.

[2] Jay Boice. Fivethirtyeight- how our club soccer predictions work, Jul 2022.

[3] Dimitris Karlis and Ioannis Ntzoufras. Bayesian modelling of football outcomes: using the Skellam's distribution for the goal difference. *IMA Journal of Management Mathematics*, 20(2):133–145, 09 2008.

[4] Stefano Leone. Fifa 22 complete player dataset, 2021.

[5] Coşkun Parim, Mehmet Şamil Güneş, Ali Hakan Büyüklü, and Doğan Yıldız. Prediction of match outcomes with multivariate statistical methods for the group stage in the UEFA champions league. *J. Hum. Kinet.*, 79(1):197–209, July 2021.

[6] Edgar Santos-Fernandez, Paul Wu, and Kerrie L. Mengersen. Bayesian statistics meets sports: a comprehensive review. *Journal of Quantitative Analysis in Sports*, 15(4):289–312, 2019.

Appendix A

R Code

Stats Project

Miguel/Antonia

2022-12-19

Load packages and set working directory

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-4
```

```
library(gridExtra)
#source(file.path("./SIM_project/helper_functions/routines_seminar1.R"))
library("readxl")
library(lme4)
library(arm)
```

```
## Loading required package: MASS
```

```
##
## arm (Version 1.13-1, built: 2022-8-25)
```

```
## Working directory is /Users/Miguel/Desktop/SIM_project
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
```

```
## v ggplot2 3.4.0      v purrr   0.3.4
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.2      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::combine() masks gridExtra::combine()
## x tidyr::expand()  masks Matrix::expand()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
## x tidyr::pack()    masks Matrix::pack()
## x dplyr::select()  masks MASS::select()
## x tidyr::unpack()  masks Matrix::unpack()
```

Set seed

```
set.seed(1234)
```

Load data

Remove some unnecessary columns

```
f22 <- f22 %>% dplyr::select(-c("player_face_url", "club_logo_url", "club_flag_url",
                          "nation_logo_url", "nation_flag_url", "player_url",
                          "real_face"))
```

```
f21 <- f21 %>% dplyr::select(-c("player_face_url", "club_logo_url", "club_flag_url",
                           "nation_logo_url", "nation_flag_url", "player_url",
                           "real_face"))

f20 <- f20 %>% dplyr::select(-c("player_face_url", "club_logo_url", "club_flag_url",
                           "nation_logo_url", "nation_flag_url", "player_url",
                           "real_face"))

f19 <- f19 %>% dplyr::select(-c("player_face_url", "club_logo_url", "club_flag_url",
                           "nation_logo_url", "nation_flag_url", "player_url",
                           "real_face"))

f18 <- f18 %>% dplyr::select(-c("player_face_url", "club_logo_url", "club_flag_url",
                           "nation_logo_url", "nation_flag_url", "player_url",
                           "real_face"))

f17 <- f17 %>% dplyr::select(-c("player_face_url", "club_logo_url", "club_flag_url",
                           "nation_logo_url", "nation_flag_url", "player_url",
                           "real_face"))

f16 <- f16 %>% dplyr::select(-c("player_face_url", "club_logo_url", "club_flag_url",
                           "nation_logo_url", "nation_flag_url", "player_url",
                           "real_face"))

f15 <- f15 %>% dplyr::select(-c("player_face_url", "club_logo_url", "club_flag_url",
                           "nation_logo_url", "nation_flag_url", "player_url",
                           "real_face"))
```

TIMELINE

Season Starts (Fall 2020) Season Ends (Spring 2021) Game (FIFA 2022) is made and players are rated (Summer 2021) New Teams (Summer 2021) (These come from the 2022 edition of the game because they are from the 2021-2022 season.) Predictions are Made for next season (Summer 2021) New Season (2021 -> 2022)

So, we are going to use the FIFA 2022 ratings to predict the 2021-2022 season with the FIFA 2022 rosters. The 2021-2022 points data goes with FIFA 2022, since that is what we are trying to predict.

Import REAL end of season info (total number of points, wins, losses, etc. for each team)

```
table22 <- read_excel("./data/LaLiga_results.xlsx", sheet = "2021-2022")
table21 <- read_excel("./data/LaLiga_results.xlsx", sheet = "2020-2021")
table20 <- read_excel("./data/LaLiga_results.xlsx", sheet = "2019-2020")
table19 <- read_excel("./data/LaLiga_results.xlsx", sheet = "2018-2019")
table18 <- read_excel("./data/LaLiga_results.xlsx", sheet = "2017-2018")
table17 <- read_excel("./data/LaLiga_results.xlsx", sheet = "2016-2017")
table16 <- read_excel("./data/LaLiga_results.xlsx", sheet = "2015-2016")
table15 <- read_excel("./data/LaLiga_results.xlsx", sheet = "2014-2015")
```

View data

```
table
```

```
## function (..., exclude = if (useNA == "no") c(NA, NaN), useNA = c("no",
##     "ifany", "always"), dnn = list.names(...), deparse.level = 1)
## {
```

```
##     list.names <- function(...) {
##         l <- as.list(substitute(list(...)))[-1L]
##         if (length(l) == 1L && is.list(..1) && !is.null(nm <- names(..1)))
##             return(nm)
##         nm <- names(l)
##         fixup <- if (is.null(nm))
##             seq_along(l)
##         else nm == ""
##         dep <- vapply(l[fixup], function(x) switch(deparse.level +
##             1, "", if (is.symbol(x)) as.character(x) else "",
##             deparse(x, nlines = 1)[1L]), "")
##         if (is.null(nm))
##             dep
##         else {
##             nm[fixup] <- dep
##             nm
##         }
##     }
##     miss.use <- missing(useNA)
##     miss.exc <- missing(exclude)
##     useNA <- if (miss.use && !miss.exc && !match(NA, exclude,
##         nomatch = 0L))
##         "ifany"
##     else match.arg(useNA)
##     doNA <- useNA != "no"
##     if (!miss.use && !miss.exc && doNA && match(NA, exclude,
##         nomatch = 0L))
##         warning("'exclude' containing NA and 'useNA' != \"no\"' are a bit contradicting")
##     args <- list(...)
##     if (length(args) == 1L && is.list(args[[1L]])) {
##         args <- args[[1L]]
##         if (length(dnn) != length(args))
##             dnn <- paste(dnn[1L], seq_along(args), sep = ".")
##     }
##     if (!length(args))
##         stop("nothing to tabulate")
##     bin <- 0L
##     lens <- NULL
##     dims <- integer()
##     pd <- 1L
##     dn <- NULL
##     for (a in args) {
##         if (is.null(lens))
##             lens <- length(a)
##         else if (length(a) != lens)
##             stop("all arguments must have the same length")
##         fact.a <- is.factor(a)
##         if (doNA)
##             aNA <- anyNA(a)
##         if (!fact.a) {
##             a0 <- a
##             op <- options(warn = 2)
##             a <- factor(a, exclude = exclude)
##             options(op)
```

```
##         }
##         add.na <- doNA
##         if (add.na) {
##             ifany <- (useNA == "ifany")
##             anNAc <- anyNA(a)
##             add.na <- if (!ifany || anNAc) {
##                 ll <- levels(a)
##                 if (add.ll <- !anyNA(ll)) {
##                   ll <- c(ll, NA)
##                   TRUE
##                 }
##                 else if (!ifany && !anNAc)
##                   FALSE
##                 else TRUE
##             }
##             else FALSE
##         }
##         if (add.na)
##             a <- factor(a, levels = ll, exclude = NULL)
##         else ll <- levels(a)
##         a <- as.integer(a)
##         if (fact.a && !miss.exc) {
##             ll <- ll[keep <- which(match(ll, exclude, nomatch = 0L) ==
##                 0L)]
##             a <- match(a, keep)
##         }
##         else if (!fact.a && add.na) {
##             if (ifany && !aNA && add.ll) {
##                 ll <- ll[!is.na(ll)]
##                 is.na(a) <- match(a0, c(exclude, NA), nomatch = 0L) >
##                   0L
##             }
##             else {
##                 is.na(a) <- match(a0, exclude, nomatch = 0L) >
##                   0L
##             }
##         }
##         nl <- length(ll)
##         dims <- c(dims, nl)
##         if (prod(dims) > .Machine$integer.max)
##             stop("attempt to make a table with >= 2^31 elements")
##         dn <- c(dn, list(ll))
##         bin <- bin + pd * (a - 1L)
##         pd <- pd * nl
##     }
##     names(dn) <- dnn
##     bin <- bin[!is.na(bin)]
##     if (length(bin))
##         bin <- bin + 1L
##     y <- array(tabulate(bin, pd), dims, dimnames = dn)
##     class(y) <- "table"
##     y
## }
## <bytecode: 0x7f79ab8d74f8>
```

```
## <environment: namespace:base>
```

```
f22
```

```
## # A tibble: 19,239 x 104
##     ...1 sofifa_id short~1 long_~2 playe~3 overall poten~4 value~5 wage_~6   age
##    <dbl>     <dbl> <chr>   <chr>   <chr>     <dbl>   <dbl>   <dbl>   <dbl> <dbl>
##  1     1    158023 L. Mes~ Lionel~ RW, ST~      93      93  7.8 e7  320000    34
##  2     2    188545 R. Lew~ Robert~ ST           92      92  1.20e8  270000    32
##  3     3     20801 Cristi~ Cristi~ ST, LW       91      91  4.5 e7  270000    36
##  4     4    190871 Neymar~ Neymar~ LW, CAM      91      91  1.29e8  270000    29
##  5     5    192985 K. De ~ Kevin ~ CM, CAM      91      91  1.25e8  350000    30
##  6     6    200389 J. Obl~ Jan Ob~ GK           91      93  1.12e8  130000    28
##  7     7    231747 K. Mba~ Kylian~ ST, LW       91      95  1.94e8  230000    22
##  8     8    167495 M. Neu~ Manuel~ GK           90      90  1.35e7   86000    35
##  9     9    192448 M. ter~ Marc-A~ GK           90      92  9.9 e7  250000    29
## 10    10    202126 H. Kane Harry ~ ST           90      90  1.30e8  240000    27
## # ... with 19,229 more rows, 94 more variables: dob <date>, height_cm <dbl>,
## #   weight_kg <dbl>, club_team_id <dbl>, club_name <chr>, league_name <chr>,
## #   league_level <dbl>, club_position <chr>, club_jersey_number <dbl>,
## #   club_loaned_from <chr>, club_joined <date>,
## #   club_contract_valid_until <dbl>, nationality_id <dbl>,
## #   nationality_name <chr>, nation_team_id <dbl>, nation_position <chr>,
## #   nation_jersey_number <dbl>, preferred_foot <chr>, weak_foot <dbl>, ...
```

```
colnames(f22)
```

```
##   [1] "...1"                        "sofifa_id"
##   [3] "short_name"                  "long_name"
##   [5] "player_positions"            "overall"
##   [7] "potential"                   "value_eur"
##   [9] "wage_eur"                    "age"
##  [11] "dob"                         "height_cm"
##  [13] "weight_kg"                   "club_team_id"
##  [15] "club_name"                   "league_name"
##  [17] "league_level"                "club_position"
##  [19] "club_jersey_number"          "club_loaned_from"
##  [21] "club_joined"                 "club_contract_valid_until"
##  [23] "nationality_id"              "nationality_name"
##  [25] "nation_team_id"              "nation_position"
##  [27] "nation_jersey_number"        "preferred_foot"
##  [29] "weak_foot"                   "skill_moves"
##  [31] "international_reputation"    "work_rate"
##  [33] "body_type"                   "release_clause_eur"
##  [35] "player_tags"                 "player_traits"
##  [37] "pace"                        "shooting"
##  [39] "passing"                     "dribbling"
##  [41] "defending"                   "physic"
##  [43] "attacking_crossing"          "attacking_finishing"
##  [45] "attacking_heading_accuracy"  "attacking_short_passing"
##  [47] "attacking_volleys"           "skill_dribbling"
##  [49] "skill_curve"                 "skill_fk_accuracy"
##  [51] "skill_long_passing"          "skill_ball_control"
##  [53] "movement_acceleration"       "movement_sprint_speed"
##  [55] "movement_agility"            "movement_reactions"
##  [57] "movement_balance"            "power_shot_power"
```

```
##  [59] "power_jumping"               "power_stamina"
##  [61] "power_strength"              "power_long_shots"
##  [63] "mentality_aggression"        "mentality_interceptions"
##  [65] "mentality_positioning"       "mentality_vision"
##  [67] "mentality_penalties"         "mentality_composure"
##  [69] "defending_marking_awareness" "defending_standing_tackle"
##  [71] "defending_sliding_tackle"    "goalkeeping_diving"
##  [73] "goalkeeping_handling"        "goalkeeping_kicking"
##  [75] "goalkeeping_positioning"     "goalkeeping_reflexes"
##  [77] "goalkeeping_speed"           "ls"
##  [79] "st"                          "rs"
##  [81] "lw"                          "lf"
##  [83] "cf"                          "rf"
##  [85] "rw"                          "lam"
##  [87] "cam"                         "ram"
##  [89] "lm"                          "lcm"
##  [91] "cm"                          "rcm"
##  [93] "rm"                          "lwb"
##  [95] "ldm"                         "cdm"
##  [97] "rdm"                         "rwb"
##  [99] "lb"                          "lcb"
## [101] "cb"                          "rcb"
## [103] "rb"                          "gk"
```

Filter for only Spain Primera Division

```
f22_sp <- f22 %>% filter(league_name=='Spain Primera Division')
f21_sp <- f21 %>% filter(league_name=='Spain Primera Division')
f20_sp <- f20 %>% filter(league_name=='Spain Primera Division')
f19_sp <- f19 %>% filter(league_name=='Spain Primera Division')
f18_sp <- f18 %>% filter(league_name=='Spain Primera Division')
f17_sp <- f17 %>% filter(league_name=='Spain Primera Division')
f16_sp <- f16 %>% filter(league_name=='Spain Primera Division')
f15_sp <- f15 %>% filter(league_name=='Spain Primera Division')
```

Pick only the columns that we want.

```
cols_to_choose <- c("overall", "potential", "value_eur", "wage_eur", "age", #"dob",
                    "height_cm", "weight_kg",  "club_name", #"club_team_id",
                    #"league_level",
                    "preferred_foot", "weak_foot", "skill_moves", "international_reputation", #"work_ra
                    "release_clause_eur", #"player_tags", #"player_traits",
                    "pace",
                    "club_position",
                    "shooting", "passing", "dribbling", "defending",
                    "physic", "attacking_crossing", "attacking_finishing", "attacking_heading_accuracy"
                    "attacking_short_passing", "attacking_volleys", "skill_dribbling", "skill_curve",
                    "skill_fk_accuracy", "skill_long_passing", "skill_ball_control", "movement_accelera
                    "movement_sprint_speed", "movement_agility", "movement_reactions", "movement_balanc
                    "power_shot_power", "power_jumping", "power_stamina", "power_strength",
                    "power_long_shots", "mentality_aggression", "mentality_interceptions", "mentality_p
                    "mentality_vision", "mentality_penalties", "mentality_composure", "defending_markin
                    "defending_standing_tackle", "defending_sliding_tackle", "goalkeeping_diving", "goal
                    "goalkeeping_kicking", "goalkeeping_positioning", "goalkeeping_reflexes", "goalkeep
                    #"ls", "st", "rs","lw",
                    #"lf", "cf", "rf", "rw",
```

```
                    #"lam", "cam", "ram", "lm",
                    #"lcm", "cm", "rcm", "rm",
                    #"lwb", "ldm", "cdm", "rdm",
                    #"rwb", "lb", "lcb", "cb",
                    #"rcb", "rb", "gk"
                    )
```

Make the column selection.

```
f22_sp <- f22_sp[cols_to_choose]
f21_sp <- f21_sp[cols_to_choose]
f20_sp <- f20_sp[cols_to_choose]
f19_sp <- f19_sp[cols_to_choose]
f18_sp <- f18_sp[cols_to_choose]
f17_sp <- f17_sp[cols_to_choose]
f16_sp <- f16_sp[cols_to_choose]
f15_sp <- f15_sp[cols_to_choose]
```

Make simplified position: GK/Defense/Midfield/Forward/Reserve

```
get_pos <- function(setx) {
  setx$pos <- c(rep("POS", nrow(setx)))
  defc <- setx$club_position %in% c("LCB", "LB", "RB", "RCB", "CB", "RWB", "LWB")
  midc <- setx$club_position %in% c("CDM","LCM","RCM","RM","CAM","LW","LM","RW","RDM","LDM","CM")
  forc <- setx$club_position %in% c("RS", "ST", "LS", "CF", "LF", "RF")
  subc <- setx$club_position %in% c("SUB", "RES")
  gkc <- setx$club_position %in% c("GK")
  setx$pos[defc] <- "defender"
  setx$pos[midc] <- "midfielder"
  setx$pos[forc] <- "forward"
  setx$pos[subc] <- "sub/res"
  setx$pos[gkc] <- "goalkeeper"
  return(setx)
}

f22_sp <- get_pos(f22_sp)
f21_sp <- get_pos(f21_sp)
f20_sp <- get_pos(f20_sp)
f19_sp <- get_pos(f19_sp)
f18_sp <- get_pos(f18_sp)
f17_sp <- get_pos(f17_sp)
f16_sp <- get_pos(f16_sp)
f15_sp <- get_pos(f15_sp)
```

Preprocessing of data

```
# Replace NAs by 0
f22_sp[is.na(f22_sp)] <- 0
f21_sp[is.na(f21_sp)] <- 0
f20_sp[is.na(f20_sp)] <- 0
f19_sp[is.na(f19_sp)] <- 0
f18_sp[is.na(f18_sp)] <- 0
f17_sp[is.na(f17_sp)] <- 0
f16_sp[is.na(f16_sp)] <- 0
f15_sp[is.na(f15_sp)] <- 0
# Group dataframes by team.
```

```r
f22_gp <- f22_sp %>% group_by(club_name) %>% summarise(across(where(is.numeric), mean))
f21_gp <- f21_sp %>% group_by(club_name) %>% summarise(across(where(is.numeric), mean))
f20_gp <- f20_sp %>% group_by(club_name) %>% summarise(across(where(is.numeric), mean))
f19_gp <- f19_sp %>% group_by(club_name) %>% summarise(across(where(is.numeric), mean))
f18_gp <- f18_sp %>% group_by(club_name) %>% summarise(across(where(is.numeric), mean))
f17_gp <- f17_sp %>% group_by(club_name) %>% summarise(across(where(is.numeric), mean))
f16_gp <- f16_sp %>% group_by(club_name) %>% summarise(across(where(is.numeric), mean))
f15_gp <- f15_sp %>% group_by(club_name) %>% summarise(across(where(is.numeric), mean))
# Join dataframes
f22_gp <- left_join(f22_gp, table22, by = c("club_name"="Equipo"))
f21_gp <- left_join(f21_gp, table21, by = c("club_name"="Equipo"))
f20_gp <- left_join(f20_gp, table20, by = c("club_name"="Equipo"))
f19_gp <- left_join(f19_gp, table19, by = c("club_name"="Equipo"))
f18_gp <- left_join(f18_gp, table18, by = c("club_name"="Equipo"))
f17_gp <- left_join(f17_gp, table17, by = c("club_name"="Equipo"))
f16_gp <- left_join(f16_gp, table16, by = c("club_name"="Equipo"))
f15_gp <- left_join(f15_gp, table15, by = c("club_name"="Equipo"))
# # Check for NAs in dataframe.
# for(i in 1:length(cols_to_choose)){
#   print(sum(is.na(f22_gp[i])))
# }
```

Add a multiplier on points for second division teams going to first division.

```r
# second_div_mult <- 0.5
```

Create new dataframe

```r
new_df <- bind_rows(f21_gp, f20_gp, f19_gp, f18_gp, f17_gp, f16_gp, f15_gp)
```

Remove some columns

```r
colnames(new_df)
```

```
##  [1] "club_name"                   "overall"
##  [3] "potential"                   "value_eur"
##  [5] "wage_eur"                    "age"
##  [7] "height_cm"                   "weight_kg"
##  [9] "weak_foot"                   "skill_moves"
## [11] "international_reputation"    "release_clause_eur"
## [13] "pace"                        "shooting"
## [15] "passing"                     "dribbling"
## [17] "defending"                   "physic"
## [19] "attacking_crossing"          "attacking_finishing"
## [21] "attacking_heading_accuracy"  "attacking_short_passing"
## [23] "attacking_volleys"           "skill_dribbling"
## [25] "skill_curve"                 "skill_fk_accuracy"
## [27] "skill_long_passing"          "skill_ball_control"
## [29] "movement_acceleration"       "movement_sprint_speed"
## [31] "movement_agility"            "movement_reactions"
## [33] "movement_balance"            "power_shot_power"
## [35] "power_jumping"               "power_stamina"
## [37] "power_strength"              "power_long_shots"
## [39] "mentality_aggression"        "mentality_interceptions"
## [41] "mentality_positioning"       "mentality_vision"
## [43] "mentality_penalties"         "mentality_composure"
```

```
## [45] "defending_marking_awareness" "defending_standing_tackle"
## [47] "defending_sliding_tackle"    "goalkeeping_diving"
## [49] "goalkeeping_handling"        "goalkeeping_kicking"
## [51] "goalkeeping_positioning"     "goalkeeping_reflexes"
## [53] "goalkeeping_speed"           "Pos."
## [55] "Pts."                        "PJ"
## [57] "G"                           "E"
## [59] "P"                           "GF"
## [61] "GC"                          "Dif."
## [63] "Des"                         "Asc"
```

```
cols_to_drop <- c( "Pos.", "PJ", "G", "E", "P", "GF", "GC", "Dif.", "Des", "Asc")
new_df <- new_df[, !colnames(new_df) %in% cols_to_drop]
f22_gp <- f22_gp[, !colnames(f22_gp) %in% cols_to_drop]
```

TRAIN

```
y <- new_df$Pts.
x <- dplyr::select(new_df, -c("Pts.", "club_name") ) %>% as.matrix() %>% scale()
```

TEST

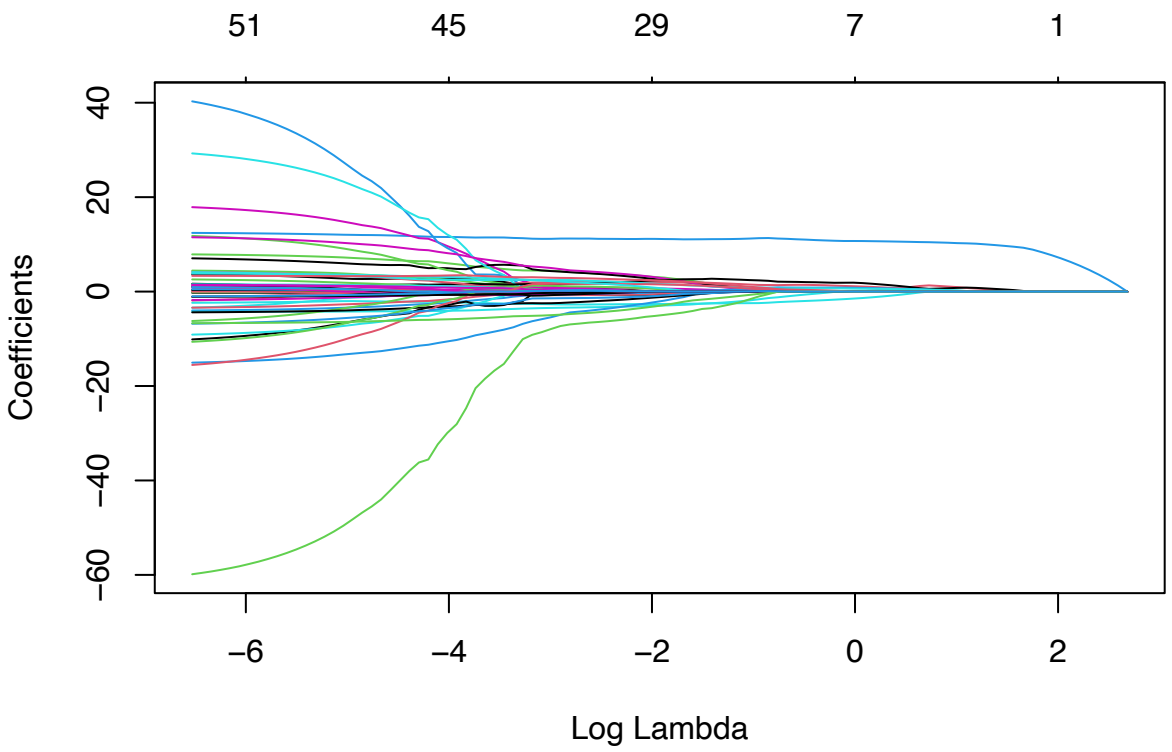
```
y_te <- f22_gp[c("club_name","Pts.")]
x_te <- dplyr::select(f22_gp, -c("Pts.", "club_name") ) %>% as.matrix() %>% scale()
```

LASSO MODEL

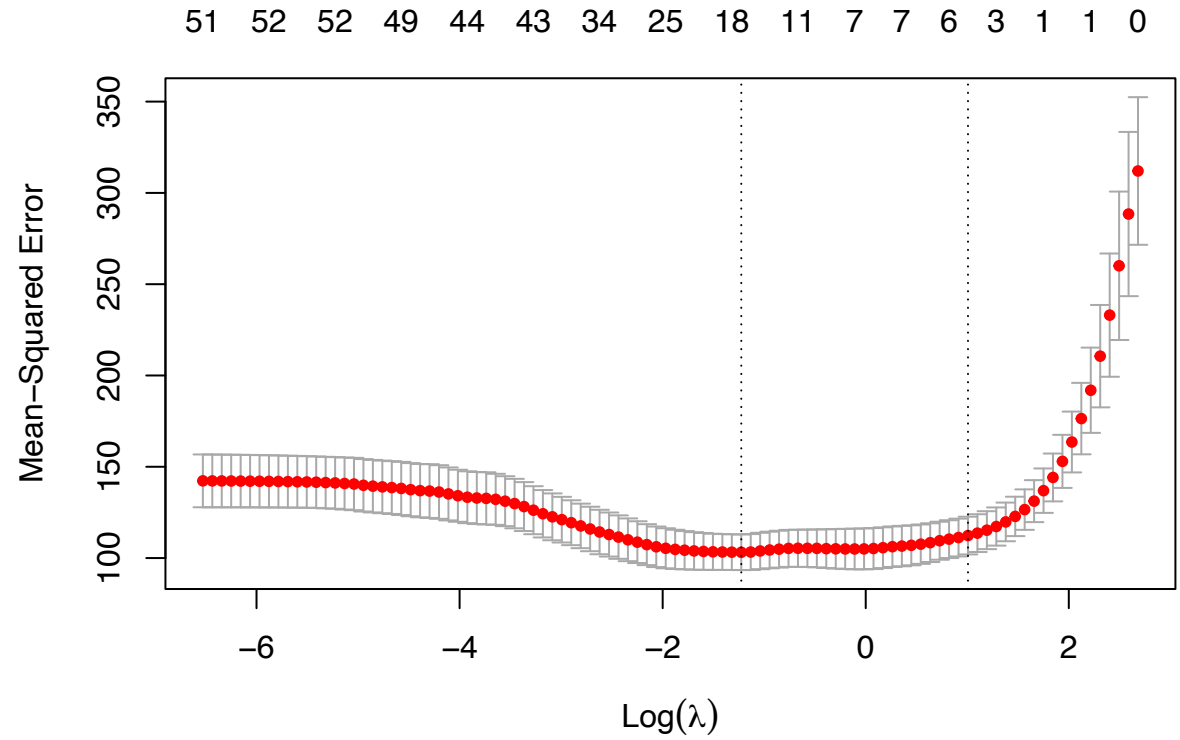
```
fit.lasso= cv.glmnet(x=x, y=y, family = "gaussian", nfolds = 5, alpha=1, type.measure = "mse")

# model
lasso.err <- assess.glmnet(fit.lasso, newx=x, newy=y, family='gaussian')
lasso.err
```

```
## $mse
## lambda.1se
##   101.0203
## attr(,"measure")
## [1] "Mean-Squared Error"
##
## $mae
## lambda.1se
##   8.004357
## attr(,"measure")
## [1] "Mean Absolute Error"
```

```
plot(fit.lasso$glmnet.fit, xvar='lambda')
```




```
plot(fit.lasso)
```




```
# try model w min value of lambda
b.lasso= as.vector(coef(fit.lasso, s='lambda.min'))
betas = data.frame("word" = c("intercept", colnames(x)), "beta_hat"= round(b.lasso, 5))
betas %>% arrange(desc(abs(beta_hat)))
```

```
##                          word beta_hat
## 1                   intercept 52.17143
## 2     international_reputation 11.12791
## 3                    dribbling -2.70233
## 4           movement_reactions  2.58930
## 5                          age -2.48630
## 6           release_clause_eur -2.22120
## 7   attacking_heading_accuracy  1.68062
## 8      goalkeeping_positioning  1.37290
## 9                    value_eur  1.20551
## 10         attacking_finishing  1.15722
## 11        goalkeeping_reflexes -1.12758
## 12              power_stamina  0.97659
## 13 defending_marking_awareness  0.91311
## 14            movement_agility  0.76300
## 15               power_jumping  0.40762
## 16                   height_cm  0.19937
## 17    defending_sliding_tackle -0.15589
## 18         mentality_composure -0.10865
## 19                   weight_kg -0.01496
## 20                     overall  0.00000
## 21                   potential  0.00000
## 22                    wage_eur  0.00000
## 23                   weak_foot  0.00000
## 24                 skill_moves  0.00000
## 25                       pace  0.00000
## 26                    shooting  0.00000
## 27                     passing  0.00000
## 28                   defending  0.00000
## 29                      physic  0.00000
## 30          attacking_crossing  0.00000
## 31     attacking_short_passing  0.00000
## 32           attacking_volleys  0.00000
## 33             skill_dribbling  0.00000
## 34                 skill_curve  0.00000
## 35           skill_fk_accuracy  0.00000
## 36          skill_long_passing  0.00000
## 37          skill_ball_control  0.00000
## 38       movement_acceleration  0.00000
## 39       movement_sprint_speed  0.00000
## 40            movement_balance  0.00000
## 41            power_shot_power  0.00000
## 42              power_strength  0.00000
## 43            power_long_shots  0.00000
## 44        mentality_aggression  0.00000
## 45     mentality_interceptions  0.00000
## 46       mentality_positioning  0.00000
## 47            mentality_vision  0.00000
## 48         mentality_penalties  0.00000
## 49   defending_standing_tackle  0.00000
## 50          goalkeeping_diving  0.00000
## 51        goalkeeping_handling  0.00000
## 52         goalkeeping_kicking  0.00000
## 53           goalkeeping_speed  0.00000
```

```r
# Test base model on 22 dataset
predictions <- predict(fit.lasso, newx = x_te)
y_te_sorted <- y_te
y_te_sorted$pred <-predictions
y_te_sorted <- y_te_sorted %>% arrange(desc(Pts.))
mean((y_te_sorted$pred-y_te_sorted$Pts.)^2)^0.5
```

```
## [1] 6.336078
```

HIERARCHICAL MODEL

```r
fp22_hm <- left_join(f22_sp, table22, by = c("club_name"="Equipo"))
fp21_hm <- left_join(f21_sp, table21, by = c("club_name"="Equipo"))
fp20_hm <- left_join(f20_sp, table20, by = c("club_name"="Equipo"))
fp19_hm <- left_join(f19_sp, table19, by = c("club_name"="Equipo"))
fp18_hm <- left_join(f18_sp, table18, by = c("club_name"="Equipo"))
fp17_hm <- left_join(f17_sp, table17, by = c("club_name"="Equipo"))
fp16_hm <- left_join(f16_sp, table16, by = c("club_name"="Equipo"))
fp15_hm <- left_join(f15_sp, table15, by = c("club_name"="Equipo"))


cols_to_drop <- c( "Pos.", "PJ", "G", "E", "P", "GF", "GC", "Dif.", "Des", "Asc", "league_level")

y_te2 <- fp22_hm[c("Pts.","club_name")]
hm_data <-bind_rows(fp21_hm, fp20_hm, fp19_hm, fp18_hm, fp17_hm, fp16_hm, fp15_hm)
hm_data<- hm_data[, !colnames(hm_data) %in% cols_to_drop]
fp22_hm<- fp22_hm[, !colnames(fp22_hm) %in% cols_to_drop]

# fp22_hm[] <- lapply(fp22_hm, function(x) if(is.numeric(x)){
#   scale(x, center=TRUE, scale=TRUE)
#   } else x)

# Check for NAs in dataframe.
for(i in 1:length(colnames(fp22_hm))){
  print(sum(is.na(fp22_hm[i])))
}
```

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

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

```r
# fp22_hm$ran <- sample(-10:10, 633, rep = TRUE)
# fp22_hm$ranpts <- fp22_hm$Pts.+ fp22_hm$ran

# later add right/left foot as binary variable
```

```r
hm_datasc <- hm_data %>%
  dplyr::select(-c(preferred_foot, club_name, Pts., club_position, pos)) %>%
  scale() %>%
  data.frame () %>%
  mutate_if(is.character, as.numeric)
hm_datasc <- cbind(hm_datasc, 'club_name'= hm_data$club_name,
                   'Pts.'=hm_data$Pts.,
                   'club_position'=hm_data$club_position,
                   'pos'=hm_data$pos)

fp22_hm_sc <- fp22_hm %>%
  dplyr::select(-c(preferred_foot, club_name, Pts., club_position, pos)) %>%
  scale() %>%
  data.frame () %>%
  mutate_if(is.character, as.numeric)
fp22_hm_sc <- cbind(fp22_hm_sc,
                    'club_name'= fp22_hm$club_name,
                    'Pts.'=fp22_hm$Pts.,
                    'club_position'=fp22_hm$club_position,
                    'pos'=fp22_hm$pos)


hm_datasc <- data.frame(hm_datasc)
fp22_hm_sc <- data.frame(fp22_hm_sc)


str(hm_datasc)
```

```
## 'data.frame':    4267 obs. of  56 variables:
##  $ overall                   : num  2.79 2.51 2.37 2.23 2.23 ...
##  $ potential                 : num  2.71 2.71 2.71 2 2 ...
##  $ value_eur                 : num  7.61 8.93 8.13 2.02 6.01 ...
##  $ wage_eur                  : num  9.25 1.57 3.95 4.66 5.54 ...
##  $ age                       : num  1.848 0.457 0.689 2.08 1.616 ...
##  $ height_cm                 : num  -1.744 1.193 1.03 0.541 0.704 ...
##  $ weight_kg                 : num  -0.394 2.017 1.696 1.213 1.053 ...
##  $ weak_foot                 : num  1.535 -0.107 1.535 -0.107 1.535 ...
##  $ skill_moves               : num  1.656 -1.795 -1.795 0.506 1.656 ...
##  $ international_reputation  : num  4.94 2.22 2.22 3.58 3.58 ...
##  $ release_clause_eur        : num  5.88 6.84 6.3 1.81 4.51 ...
##  $ pace                      : num  0.997 -2.536 -2.536 0.415 0.54 ...
##  $ shooting                  : num  1.729 -2.284 -2.284 0.769 1.424 ...
##  $ passing                   : num  1.488 -2.554 -2.554 0.822 1.044 ...
##  $ dribbling                 : num  1.445 -2.585 -2.585 0.512 1.063 ...
##  $ defending                 : num  -0.482 -2.004 -2.004 1.522 -0.402 ...
##  $ physic                    : num  0.25 -2.562 -2.562 1.115 0.725 ...
##  $ attacking_crossing        : num  1.442 -2.207 -1.953 0.479 0.935 ...
##  $ attacking_finishing       : num  2.054 -1.935 -1.792 0.629 1.722 ...
##  $ attacking_heading_accuracy: num  0.732 -2.202 -2.416 1.905 1.852 ...
##  $ attacking_short_passing   : num  1.555 -1.435 -0.314 0.994 1.243 ...
##  $ attacking_volleys         : num  2.11 -1.83 -1.78 1.11 2.01 ...
##  $ skill_dribbling           : num  1.727 -2.423 -1.979 0.196 1.283 ...
##  $ skill_curve               : num  1.863 -2.119 -1.87 0.917 1.266 ...
##  $ skill_fk_accuracy         : num  2.34 -1.87 -1.98 1.39 1.23 ...
```

```
##  $ skill_long_passing         : num  1.915 -1.185 0.213 1.429 0.821 ...
##  $ skill_ball_control         : num  1.615 -1.923 -1.923 0.918 1.294 ...
##  $ movement_acceleration      : num  1.786 -1.611 -1.964 0.442 0.795 ...
##  $ movement_sprint_speed      : num  1.005 -0.435 -1.155 0.285 0.429 ...
##  $ movement_agility           : num  1.8428 0.0908 -2.0993 0.8938 0.9668 ...
##  $ movement_reactions         : num  2.61 1.97 1.76 2.39 2.29 ...
##  $ movement_balance           : num  2.2005 -1.2084 -1.653 0.0515 0.2738 ...
##  $ power_shot_power           : num  1.407 -0.206 0.212 0.989 1.287 ...
##  $ power_jumping              : num  0.0799 0.9246 1.009 2.1916 1.009 ...
##  $ power_stamina              : num  0.476 -1.585 -1.984 1.075 0.875 ...
##  $ power_strength             : num  0.223 0.941 0.941 1.499 1.1 ...
##  $ power_long_shots           : num  1.956 -2.06 -2.158 0.487 1.221 ...
##  $ mentality_aggression       : num  -0.854 -1.388 -0.907 1.602 0.16 ...
##  $ mentality_interceptions    : num  -0.543 -1.453 -1.323 1.538 -0.586 ...
##  $ mentality_positioning      : num  1.753 -2.163 -2.163 0.798 1.61 ...
##  $ mentality_vision           : num  2.258 0.363 0.679 0.742 1.753 ...
##  $ mentality_penalties        : num  1.34 -2.49 -1.65 2.36 1.88 ...
##  $ mentality_composure        : num  1.633 0.721 0.786 1.372 1.437 ...
##  $ defending_marking_awareness: num  -0.776 -0.997 -1.085 1.569 -0.112 ...
##  $ defending_standing_tackle  : num  -0.731 -1.686 -1.645 1.469 -1.188 ...
##  $ defending_sliding_tackle   : num  -1.09 -1.34 -1.67 1.66 -1.34 ...
##  $ goalkeeping_diving         : num  -0.184 5.97 6.041 -0.184 -0.184 ...
##  $ goalkeeping_handling       : num  -0.184 6.507 5.998 -0.184 -0.184 ...
##  $ goalkeeping_kicking        : num  -0.183 5.738 6.497 -0.183 -0.183 ...
##  $ goalkeeping_positioning    : num  -0.184 6.31 6.166 -0.184 -0.184 ...
##  $ goalkeeping_reflexes       : num  -0.184 6.094 6.094 -0.184 -0.184 ...
##  $ goalkeeping_speed          : num  -0.212 5.155 4.432 -0.212 -0.212 ...
##  $ club_name                  : chr  "FC Barcelona" "Atlético de Madrid" "FC Barcelona" "Real Madrid
##  $ Pts.                       : num  79 86 79 84 84 84 84 84 84 79 ...
##  $ club_position              : chr  "CAM" "GK" "GK" "LCB" ...
##  $ pos                        : chr  "midfielder" "goalkeeper" "goalkeeper" "defender" ...
```

```
# Important Variables from LASSO
# international_reputation  10.71554
# movement_reactions    1.87503
# age   -1.60636
# attacking_heading_accuracy    1.19219
# power_jumping 0.82994
# power_stamina 0.65873
# goalkeeping_positioning

ML_ex <- lmer(Pts. ~international_reputation + movement_reactions +
                age + attacking_heading_accuracy +
                power_jumping + power_stamina + goalkeeping_positioning + (1|club_name:pos), data=hm_da
pts_pred<-predict(ML_ex, newdata=fp22_hm_sc)
ML_ex
```

```
## Linear mixed model fit by maximum likelihood  ['lmerMod']
## Formula: Pts. ~ international_reputation + movement_reactions + age +
##     attacking_heading_accuracy + power_jumping + power_stamina +
##     goalkeeping_positioning + (1 | club_name:pos)
##    Data: hm_datasc
##       AIC       BIC    logLik  deviance  df.resid
##  30630.51  30694.10 -15305.26  30610.51      4257
## Random effects:
```

```
##  Groups        Name        Std.Dev.
##  club_name:pos (Intercept) 14.571
##  Residual                   8.164
## Number of obs: 4267, groups:  club_name:pos, 155
## Fixed Effects:
##                (Intercept)    international_reputation
##                  46.85585                     0.41940
##         movement_reactions                         age
##                   -0.01333                    -0.11843
## attacking_heading_accuracy               power_jumping
##                   -0.24771                     0.24605
##              power_stamina     goalkeeping_positioning
##                    0.25257                     0.04086
```

```
d <- f22 %>% filter(league_name=='Spain Primera Division')
compare <- cbind(y_te2, pts_pred, d$short_name)
compare <- compare %>% mutate("diff" = Pts.-pts_pred)
sd_players = mean((compare$diff)^2)^0.5
sd_players
```

```
## [1] 7.464471
```

```
compare_mean <- compare %>% group_by(club_name) %>% summarise(across(where(is.numeric), mean)) %>% arra
#[1] 12.16879 age + overall + international_reputation  + (1|pos)
#[1] 7.522987  age + overall + international_reputation  + (1|club_name)
#[1] 7.4702 age + overall + international_reputation  + (1|club_name:pos)

# 12.42    (1|pos)
# 7.512    (1|club_name)
# 7.464    (1|club_name:pos)



sd_teams = (mean(compare_mean$diff^2))^0.5
sd_teams
```

```
## [1] 7.500568
```

```
compare_mean
```

```
## # A tibble: 20 x 4
##    club_name                Pts. pts_pred     diff
##    <chr>                   <dbl>    <dbl>    <dbl>
##  1 Real Madrid CF             86     84.2   1.81
##  2 FC Barcelona               73     87.0 -14.0
##  3 Atlético de Madrid         71     78.9  -7.93
##  4 Sevilla FC                 70     66.3   3.68
##  5 Real Betis Balompié        65     49.1  15.9
##  6 Real Sociedad              62     53.8   8.20
##  7 Villarreal CF              59     58.9   0.0815
##  8 Athletic Club de Bilbao    55     53.2   1.83
##  9 Valencia CF                48     56.1  -8.08
## 10 CA Osasuna                 47     39.1   7.95
## 11 RC Celta de Vigo           46     47.7  -1.74
## 12 Elche CF                   42     38.8   3.24
## 13 Rayo Vallecano             42     43.2  -1.16
```

```
## 14 RCD Espanyol de Barcelona    42     45.9  -3.86
## 15 Cádiz CF                     39     44.1  -5.08
## 16 Getafe CF                    39     46.8  -7.77
## 17 RCD Mallorca                 39     33.7   5.35
## 18 Granada CF                   38     38.6  -0.611
## 19 Levante Unión Deportiva      35     41.9  -6.88
## 20 Deportivo Alavés             31     45.4 -14.4
```

Try to predict player value from other data.

```
hm_datasc["value_eur"] <- hm_data$value_eur
fp22_hm_sc["value_eur"] <- fp22_hm$value_eur

ML_ex_o <- lmer(value_eur ~  age + overall + international_reputation + wage_eur + (1|club_name:pos), da
v_pred <-predict(ML_ex_o, newdata=fp22_hm_sc)
ML_ex_o
```

```
## Linear mixed model fit by maximum likelihood  ['lmerMod']
## Formula: value_eur ~ age + overall + international_reputation + wage_eur +
##     (1 | club_name:pos)
##    Data: hm_datasc
##      AIC      BIC   logLik deviance df.resid
## 144809.6 144854.1 -72397.8 144795.6     4260
## Random effects:
##  Groups        Name        Std.Dev.
##  club_name:pos (Intercept) 3194343
##  Residual                  5472760
## Number of obs: 4267, groups:  club_name:pos, 155
## Fixed Effects:
##              (Intercept)                       age                   overall
##                  8912361                 -2933624                   5090214
## international_reputation                 wage_eur
##                  1702288                  6058432
```

```
compare_o <- cbind(y_te2, v_pred, d[c("short_name", "overall", "value_eur")])
compare_o <- compare_o %>% mutate("diff" = round(value_eur-v_pred))
mean((compare_o$diff)^2)^0.5
```

```
## [1] 8739835
```

```
# 9026972

mean(compare_o$value_eur)
```

```
## [1] 11504265
```

Try to predict player wage from data

```
hm_datasc["wage_eur"] <- hm_data$wage_eur
fp22_hm_sc["wage_eur"] <- fp22_hm$wage_eur
```

```
ML_ex_o <- lmer(wage_eur ~  age + overall + international_reputation + (1|club_name) + (1|pos), data=hm
w_pred <-predict(ML_ex_o, newdata=fp22_hm_sc)
ML_ex_o
```

```
## Linear mixed model fit by maximum likelihood  ['lmerMod']
## Formula:
## wage_eur ~ age + overall + international_reputation + (1 | club_name) +
##     (1 | pos)
##    Data: hm_datasc
##       AIC       BIC    logLik  deviance  df.resid
##  99916.50  99961.01 -49951.25  99902.50      4260
## Random effects:
##  Groups    Name        Std.Dev.
##  club_name (Intercept) 17492
##  pos       (Intercept)  4514
##  Residual               28930
## Number of obs: 4267, groups:  club_name, 31; pos, 5
## Fixed Effects:
##              (Intercept)                       age                   overall
##                    34199                     -4464                     17014
## international_reputation
##                    25166
```

```
compare_w <- cbind(y_te2, v_pred, d[c("short_name", "overall", "wage_eur")])
compare_w <- compare_w %>% mutate("diff" = round(wage_eur-w_pred))
mean((compare_w$diff)^2)^0.5
```

```
## [1] 24548.54
```

```
# 24591.14 (1|club_name)
# 24548.54 (1|pos) + (1|club_name)
# 21600.1  (1|club_name:pos)
# 21468.42 (1|club_name) + (1|club_name:pos)


mean(hm_data$wage_eur)
```

```
## [1] 35958.75
```

—- BAYESIAN REGRESSION MODEL —-

```
# Using scaled_new_df dataframe for Bayesian regression model
scaled_new_df <- bind_cols(x, y)
```

```
## New names:
## * `` -> `...53`
```

```
names(scaled_new_df)[names(scaled_new_df) == '...53'] <- 'Pts.'
```

Fitting the model

```
library(rstanarm)
```

```
## Loading required package: Rcpp
## This is rstanarm version 2.21.3
## - 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 objects are masked from 'package:arm':
##
##     invlogit, logit
```

```
# Fit the model:
bayes_reg_model <- rstanarm::stan_glm(Pts. ~ ., family = gaussian(), data=scaled_new_df)
```

```
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.000112 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 1.12 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 3.24338 seconds (Warm-up)
## Chain 1:                2.8784 seconds (Sampling)
## Chain 1:                6.12178 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1.5e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.15 seconds.
## Chain 2: Adjust your expectations accordingly!
```

```
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 3.00327 seconds (Warm-up)
## Chain 2:                3.03436 seconds (Sampling)
## Chain 2:                6.03762 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1.4e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.14 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 3.1903 seconds (Warm-up)
## Chain 3:                4.1147 seconds (Sampling)
## Chain 3:                7.30501 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 2.1e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.21 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
```

```
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 3.25883 seconds (Warm-up)
## Chain 4:                3.91911 seconds (Sampling)
## Chain 4:                7.17794 seconds (Total)
## Chain 4:
```

View a summary of the model:

```
summary(bayes_reg_model)
```

```
##
## Model Info:
##  function:     stan_glm
##  family:       gaussian [identity]
##  formula:      Pts. ~ .
##  algorithm:    sampling
##  sample:       4000 (posterior sample size)
##  priors:       see help('prior_summary')
##  observations: 140
##  predictors:   53
##
## Estimates:
##                              mean   sd    10%   50%   90%
## (Intercept)                 52.2    0.8  51.2  52.2  53.2
## overall                      3.6    7.4  -5.8   3.5  13.3
## potential                   -1.6    7.3 -11.1  -1.7   7.8
## value_eur                    7.1    4.4   1.6   7.1  12.8
## wage_eur                    -3.8    2.7  -7.2  -3.7  -0.3
## age                         -2.3    2.4  -5.4  -2.3   0.7
## height_cm                    1.2    2.0  -1.2   1.2   3.7
## weight_kg                   -0.3    1.6  -2.4  -0.3   1.7
## weak_foot                    0.0    1.5  -2.0   0.0   1.9
## skill_moves                 -0.3    2.4  -3.4  -0.3   2.9
## international_reputation    12.5    2.9   8.9  12.5  16.2
## release_clause_eur          -4.2    2.0  -6.7  -4.1  -1.7
## pace                        16.2    7.8   6.4  16.2  26.1
## shooting                   -11.5   13.9 -29.1 -11.2   6.4
## passing                     -2.6   13.1 -19.2  -2.8  14.1
## dribbling                  -48.1   18.6 -71.9 -48.1 -24.2
## defending                   32.9   17.0  11.2  32.7  54.9
## physic                      -2.5    7.2 -11.5  -2.5   6.8
## attacking_crossing           0.9    3.3  -3.4   1.0   5.2
## attacking_finishing          8.2    7.6  -1.5   8.2  18.0
## attacking_heading_accuracy  -0.7    2.2  -3.5  -0.7   2.1
## attacking_short_passing      4.6    5.3  -2.2   4.7  11.1
```

```
## attacking_volleys             0.7    2.7  -2.7   0.7   3.9
## skill_dribbling              22.8   11.0   8.6  22.9  36.8
## skill_curve                  -1.4    3.2  -5.4  -1.4   2.8
## skill_fk_accuracy             0.3    2.2  -2.5   0.3   3.1
## skill_long_passing           -2.7    4.0  -7.9  -2.7   2.3
## skill_ball_control            9.2    5.6   2.2   9.2  16.1
## movement_acceleration       -13.8    5.1 -20.5 -13.8  -7.3
## movement_sprint_speed        -8.3    5.9 -15.6  -8.4  -0.5
## movement_agility             10.2    3.5   5.7  10.2  14.6
## movement_reactions            1.2    3.8  -3.6   1.1   6.0
## movement_balance              3.6    2.7   0.1   3.6   7.0
## power_shot_power              2.7    2.9  -0.9   2.7   6.5
## power_jumping                 0.3    2.0  -2.2   0.4   2.8
## power_stamina                 3.8    3.2  -0.2   3.9   7.8
## power_strength                0.6    4.0  -4.6   0.6   5.8
## power_long_shots              1.7    4.2  -3.8   1.7   7.1
## mentality_aggression          0.7    2.2  -2.1   0.6   3.5
## mentality_interceptions      -4.7    4.7 -11.0  -4.6   1.3
## mentality_positioning         0.8    2.6  -2.5   0.7   4.1
## mentality_vision              2.3    4.5  -3.4   2.2   8.0
## mentality_penalties           1.5    2.1  -1.1   1.5   4.1
## mentality_composure          -4.2    2.3  -7.1  -4.2  -1.3
## defending_marking_awareness -12.5    7.0 -21.6 -12.4  -3.7
## defending_standing_tackle    -8.6    6.1 -16.3  -8.6  -0.8
## defending_sliding_tackle     -5.8    3.9 -10.7  -5.9  -0.9
## goalkeeping_diving            3.9    2.6   0.5   3.9   7.1
## goalkeeping_handling          1.0    2.7  -2.4   1.0   4.5
## goalkeeping_kicking          -1.1    2.1  -3.8  -1.1   1.5
## goalkeeping_positioning       3.5    2.6   0.2   3.5   6.8
## goalkeeping_reflexes         -6.6    3.1 -10.6  -6.6  -2.7
## goalkeeping_speed            -0.8    1.3  -2.5  -0.8   0.8
## sigma                         9.1    0.7   8.3   9.1  10.1
##
## Fit Diagnostics:
##            mean   sd   10%   50%   90%
## mean_PPD 52.2    1.1 50.8  52.2  53.6
##
## The mean_ppd is the sample average posterior predictive distribution of the outcome variable (for de
##
## MCMC diagnostics
##                             mcse Rhat n_eff
## (Intercept)                 0.0  1.0  6208
## overall                     0.1  1.0  2475
## potential                   0.2  1.0  2270
## value_eur                   0.1  1.0  3299
## wage_eur                    0.0  1.0  3421
## age                         0.0  1.0  2301
## height_cm                   0.0  1.0  3305
## weight_kg                   0.0  1.0  3745
## weak_foot                   0.0  1.0  3335
## skill_moves                 0.0  1.0  4189
## international_reputation    0.0  1.0  4726
## release_clause_eur          0.0  1.0  3086
## pace                        0.2  1.0  2651
```

```
## shooting                    0.4  1.0  1321
## passing                     0.3  1.0  1669
## dribbling                   0.5  1.0  1487
## defending                   0.4  1.0  1449
## physic                      0.2  1.0  1470
## attacking_crossing          0.1  1.0  2258
## attacking_finishing         0.2  1.0  1312
## attacking_heading_accuracy  0.0  1.0  2131
## attacking_short_passing     0.1  1.0  2470
## attacking_volleys           0.0  1.0  3949
## skill_dribbling             0.3  1.0  1582
## skill_curve                 0.1  1.0  3433
## skill_fk_accuracy           0.0  1.0  3679
## skill_long_passing          0.1  1.0  3043
## skill_ball_control          0.1  1.0  2037
## movement_acceleration       0.1  1.0  3147
## movement_sprint_speed       0.1  1.0  2936
## movement_agility            0.1  1.0  2612
## movement_reactions          0.1  1.0  3671
## movement_balance            0.1  1.0  2580
## power_shot_power            0.1  1.0  1946
## power_jumping               0.0  1.0  2828
## power_stamina               0.1  1.0  1955
## power_strength              0.1  1.0  1789
## power_long_shots            0.1  1.0  1900
## mentality_aggression        0.0  1.0  2450
## mentality_interceptions     0.1  1.0  2000
## mentality_positioning       0.0  1.0  4235
## mentality_vision            0.1  1.0  1850
## mentality_penalties         0.0  1.0  3850
## mentality_composure         0.0  1.0  3035
## defending_marking_awareness 0.2  1.0  1466
## defending_standing_tackle   0.1  1.0  1827
## defending_sliding_tackle    0.1  1.0  2945
## goalkeeping_diving          0.0  1.0  3185
## goalkeeping_handling        0.0  1.0  3428
## goalkeeping_kicking         0.0  1.0  3365
## goalkeeping_positioning     0.0  1.0  3876
## goalkeeping_reflexes        0.1  1.0  3118
## goalkeeping_speed           0.0  1.0  2948
## sigma                       0.0  1.0  2438
## mean_PPD                    0.0  1.0  4496
## log-posterior               0.2  1.0  1000
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample
```

To obtain summary of medians

```
bayes_reg_model
```

```
## stan_glm
##  family:       gaussian [identity]
##  formula:      Pts. ~ .
##  observations: 140
##  predictors:   53
```

```
## ------
##                               Median MAD_SD
## (Intercept)                    52.2    0.8
## overall                         3.5    7.3
## potential                      -1.7    7.3
## value_eur                       7.1    4.3
## wage_eur                       -3.7    2.6
## age                            -2.3    2.4
## height_cm                       1.2    2.0
## weight_kg                      -0.3    1.6
## weak_foot                       0.0    1.5
## skill_moves                    -0.3    2.4
## international_reputation       12.5    2.9
## release_clause_eur             -4.1    2.0
## pace                           16.2    7.8
## shooting                      -11.2   14.0
## passing                        -2.8   13.3
## dribbling                     -48.1   18.2
## defending                      32.7   16.5
## physic                         -2.5    7.3
## attacking_crossing              1.0    3.4
## attacking_finishing             8.2    7.5
## attacking_heading_accuracy     -0.7    2.2
## attacking_short_passing         4.7    5.3
## attacking_volleys               0.7    2.6
## skill_dribbling                22.9   10.8
## skill_curve                    -1.4    3.1
## skill_fk_accuracy               0.3    2.2
## skill_long_passing             -2.7    4.0
## skill_ball_control              9.2    5.6
## movement_acceleration         -13.8    5.3
## movement_sprint_speed          -8.4    5.8
## movement_agility               10.2    3.4
## movement_reactions              1.1    3.7
## movement_balance                3.6    2.7
## power_shot_power                2.7    2.9
## power_jumping                   0.4    2.0
## power_stamina                   3.9    3.2
## power_strength                  0.6    4.0
## power_long_shots                1.7    4.1
## mentality_aggression            0.6    2.1
## mentality_interceptions        -4.6    4.6
## mentality_positioning           0.7    2.6
## mentality_vision                2.2    4.5
## mentality_penalties             1.5    2.1
## mentality_composure            -4.2    2.3
## defending_marking_awareness   -12.4    7.0
## defending_standing_tackle      -8.6    6.0
## defending_sliding_tackle       -5.9    3.8
## goalkeeping_diving              3.9    2.5
## goalkeeping_handling            1.0    2.8
## goalkeeping_kicking            -1.1    2.0
## goalkeeping_positioning         3.5    2.6
## goalkeeping_reflexes           -6.6    3.0
```

```
## goalkeeping_speed            -0.8    1.3
##
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 9.1    0.7
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

To obtain prediction intervals:

```
m= coef(bayes_reg_model)
q= posterior_interval(bayes_reg_model)[1:length(m),]
b.intervals= round(cbind(m, q), 3)
b.intervals
```

```
##                                 m      5%     95%
## (Intercept)                52.174  50.908  53.417
## overall                     3.452  -8.775  16.036
## potential                  -1.709 -13.334  10.214
## value_eur                   7.065  -0.062  14.419
## wage_eur                   -3.740  -8.346   0.889
## age                        -2.326  -6.237   1.527
## height_cm                   1.213  -1.980   4.415
## weight_kg                  -0.343  -3.034   2.299
## weak_foot                   0.001  -2.455   2.447
## skill_moves                -0.280  -4.234   3.701
## international_reputation   12.517   7.818  17.264
## release_clause_eur         -4.149  -7.471  -0.936
## pace                       16.169   3.142  28.728
## shooting                  -11.239 -34.408  10.822
## passing                    -2.840 -24.176  18.547
## dribbling                 -48.090 -78.605 -16.940
## defending                  32.724   5.309  61.649
## physic                     -2.471 -14.124   9.365
## attacking_crossing          0.957  -4.470   6.372
## attacking_finishing         8.245  -4.153  20.802
## attacking_heading_accuracy -0.709  -4.410   2.915
## attacking_short_passing     4.679  -4.071  13.002
## attacking_volleys           0.719  -3.765   4.923
## skill_dribbling            22.937   4.219  40.679
## skill_curve                -1.356  -6.515   3.865
## skill_fk_accuracy           0.336  -3.208   3.947
## skill_long_passing         -2.719  -9.401   3.691
## skill_ball_control          9.237   0.031  18.279
## movement_acceleration     -13.796 -22.208  -5.371
## movement_sprint_speed      -8.439 -17.724   1.571
## movement_agility           10.207   4.354  16.017
## movement_reactions          1.150  -5.012   7.437
## movement_balance            3.623  -0.780   7.933
## power_shot_power            2.682  -1.929   7.605
## power_jumping               0.383  -2.852   3.558
## power_stamina               3.883  -1.371   8.874
## power_strength              0.589  -6.037   7.144
```

```
## power_long_shots              1.697  -5.243   8.744
## mentality_aggression          0.609  -2.892   4.225
## mentality_interceptions      -4.647 -12.502   2.967
## mentality_positioning         0.702  -3.416   5.005
## mentality_vision              2.183  -4.949   9.652
## mentality_penalties           1.492  -1.939   4.819
## mentality_composure          -4.178  -7.970  -0.442
## defending_marking_awareness -12.423 -24.204  -1.033
## defending_standing_tackle    -8.646 -18.365   1.642
## defending_sliding_tackle     -5.923 -11.990   0.721
## goalkeeping_diving            3.853  -0.390   8.080
## goalkeeping_handling          1.015  -3.356   5.388
## goalkeeping_kicking          -1.118  -4.493   2.322
## goalkeeping_positioning       3.505  -0.767   7.809
## goalkeeping_reflexes         -6.628 -11.723  -1.563
## goalkeeping_speed            -0.811  -2.965   1.306
```

Making predictions:

```
# Making predictions:
x_te_df <- as.data.frame(x_te)
predictions_bayes <- predict(bayes_reg_model, newdata = x_te_df)

pred_bayes_compare <- cbind(y_te, predictions_bayes)
pred_bayes_compare
```

```
##                    club_name Pts. predictions_bayes
## 1    Athletic Club de Bilbao   55          56.66665
## 2         Atlético de Madrid   71          73.87399
## 3                 CA Osasuna   47          46.14914
## 4                   Cádiz CF   39          42.45438
## 5           Deportivo Alavés   31          46.35545
## 6                   Elche CF   42          49.97121
## 7               FC Barcelona   73          75.04346
## 8                  Getafe CF   39          45.18942
## 9                 Granada CF   38          44.71812
## 10   Levante Unión Deportiva   35          46.97023
## 11            Rayo Vallecano   42          51.01301
## 12          RC Celta de Vigo   46          43.19493
## 13 RCD Espanyol de Barcelona   42          38.59277
## 14              RCD Mallorca   39          43.86118
## 15       Real Betis Balompié   65          50.99236
## 16            Real Madrid CF   86          65.09661
## 17             Real Sociedad   62          57.34609
## 18                Sevilla FC   70          61.47239
## 19              Valencia CF   48          55.44545
## 20             Villarreal CF   59          48.99848
```

Mean squared error to assess accuracy of Bayesian Regression

```
mse_bayes = mean((pred_bayes_compare$Pts. - pred_bayes_compare$predictions_bayes)^2)
mse_bayes
```

```
## [1] 78.06409
```

Root mean squared error to assess accuracy of Bayesian Regression

```
root_mse_bayes = sqrt(mean((pred_bayes_compare$Pts. - pred_bayes_compare$predictions_bayes)^2))
root_mse_bayes
```

```
## [1] 8.835389
```

Correlation between predicted and actual values of testing set

```
cor(pred_bayes_compare$Pts., pred_bayes_compare$predictions_bayes)
```

```
## [1] 0.8200798
```

—- BAYESIAN MODEL SELECTION ——

Fitting the model

```
library(mombf)
```

```
## Loading required package: mvtnorm
## Loading required package: ncvreg
## Loading required package: mgcv
## Loading required package: nlme
##
## Attaching package: 'nlme'
## The following object is masked from 'package:dplyr':
##
##     collapse
## The following object is masked from 'package:lme4':
##
##     lmList
## This is mgcv 1.8-40. For overview type 'help("mgcv-package")'.
```

```
# Fit Bayesian model selection model
fit.bayesmodelselection <- modelSelection(Pts. ~ ., data=scaled_new_df, priorCoef=zellnerprior(taustd=1)
```

```
## Greedy searching posterior mode... Done.
## Running Gibbs sampler.......... Done.
```

Viewing summary of bayesian model selection

```
head(postProb(fit.bayesmodelselection),10)
```

```
##    modelid family          pp
## 1       11 normal 0.611394250
## 2    11,35 normal 0.054748127
## 3    11,21 normal 0.038237312
## 4     6,11 normal 0.023953269
## 5    11,36 normal 0.016248735
```

```
## 6  6,11,21 normal 0.012387920
## 7     3,11 normal 0.011155557
## 8    11,32 normal 0.010848025
## 9    11,12 normal 0.008034734
## 10   11,40 normal 0.007273579
```

——— VISUALIZATIONS ———-

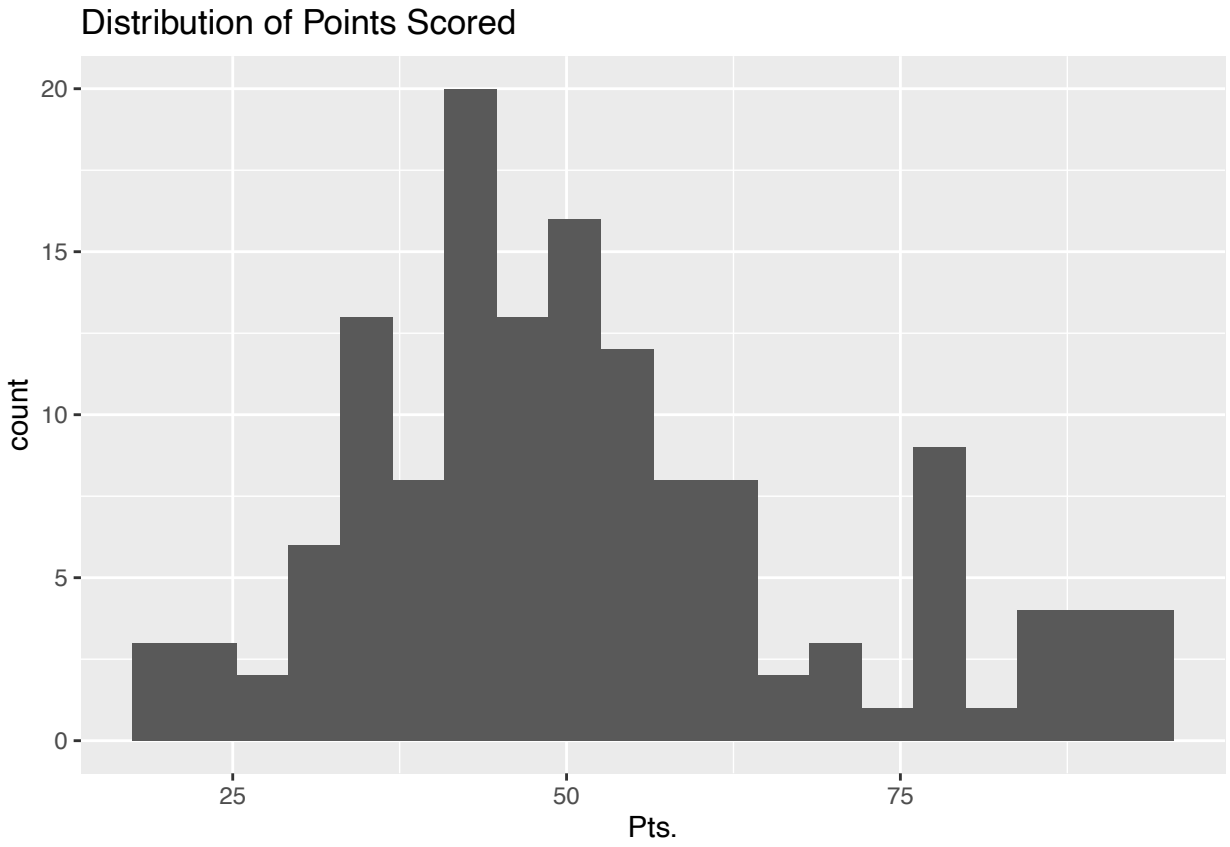
```
colnames(scaled_new_df)
```

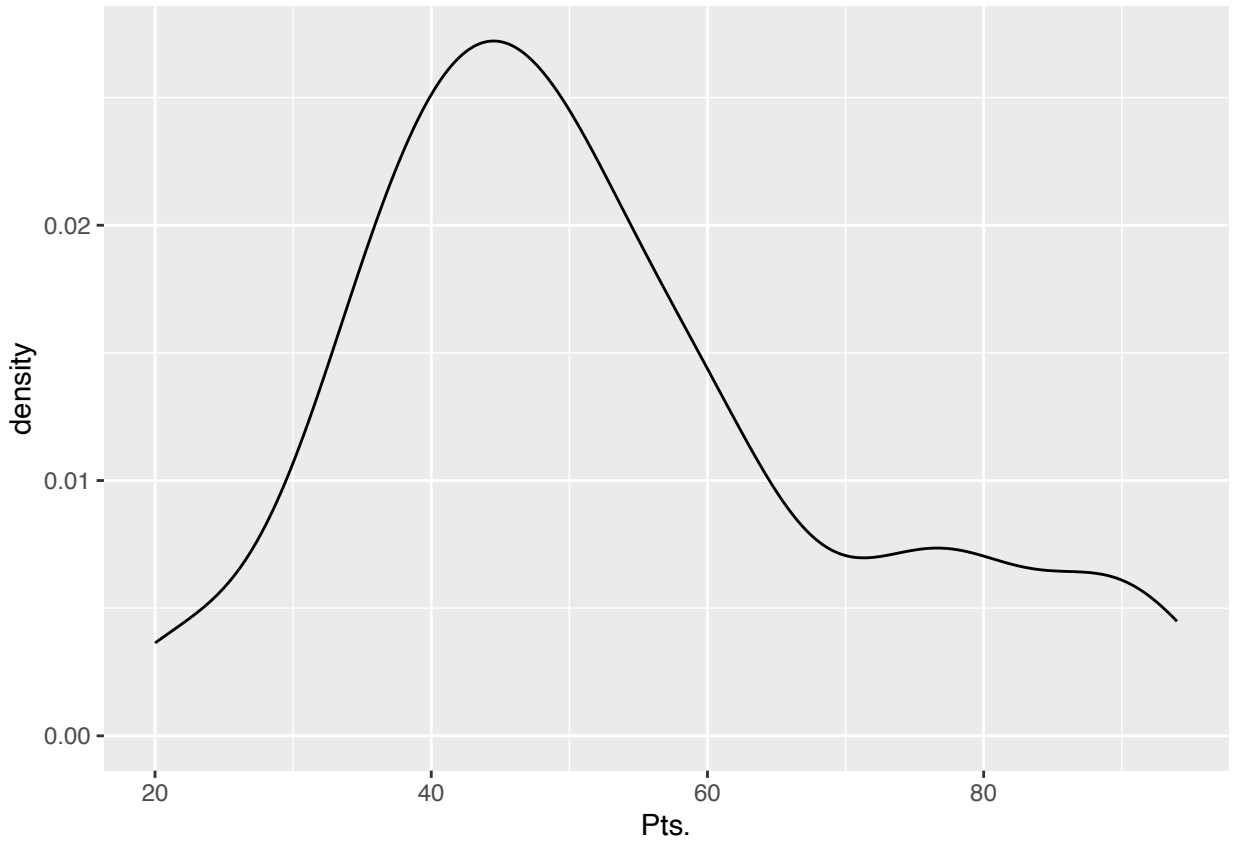
```
##  [1] "overall"                    "potential"
##  [3] "value_eur"                  "wage_eur"
##  [5] "age"                        "height_cm"
##  [7] "weight_kg"                  "weak_foot"
##  [9] "skill_moves"                "international_reputation"
## [11] "release_clause_eur"         "pace"
## [13] "shooting"                   "passing"
## [15] "dribbling"                  "defending"
## [17] "physic"                     "attacking_crossing"
## [19] "attacking_finishing"        "attacking_heading_accuracy"
## [21] "attacking_short_passing"    "attacking_volleys"
## [23] "skill_dribbling"            "skill_curve"
## [25] "skill_fk_accuracy"          "skill_long_passing"
## [27] "skill_ball_control"         "movement_acceleration"
## [29] "movement_sprint_speed"      "movement_agility"
## [31] "movement_reactions"         "movement_balance"
## [33] "power_shot_power"           "power_jumping"
## [35] "power_stamina"              "power_strength"
## [37] "power_long_shots"           "mentality_aggression"
## [39] "mentality_interceptions"    "mentality_positioning"
## [41] "mentality_vision"           "mentality_penalties"
## [43] "mentality_composure"        "defending_marking_awareness"
## [45] "defending_standing_tackle"  "defending_sliding_tackle"
## [47] "goalkeeping_diving"         "goalkeeping_handling"
## [49] "goalkeeping_kicking"        "goalkeeping_positioning"
## [51] "goalkeeping_reflexes"       "goalkeeping_speed"
## [53] "Pts."
```

```
scaled_new_df[1, ]
```

```
## # A tibble: 1 x 53
##   overall potential value_eur wage_eur     age height_cm weigh~1 weak_~2 skill~3
##     <dbl>     <dbl>     <dbl>    <dbl>   <dbl>     <dbl>   <dbl>   <dbl>   <dbl>
## 1   0.282     0.277     0.465   -0.445 -0.0530      1.39    2.24   0.463    2.03
## # ... with 44 more variables: international_reputation <dbl>,
## #   release_clause_eur <dbl>, pace <dbl>, shooting <dbl>, passing <dbl>,
## #   dribbling <dbl>, defending <dbl>, physic <dbl>, attacking_crossing <dbl>,
```

```
## #   attacking_finishing <dbl>, attacking_heading_accuracy <dbl>,
## #   attacking_short_passing <dbl>, attacking_volleys <dbl>,
## #   skill_dribbling <dbl>, skill_curve <dbl>, skill_fk_accuracy <dbl>,
## #   skill_long_passing <dbl>, skill_ball_control <dbl>, ...
```




density
0.02
0.01
0.00
20
40
60
80
Pts.