TP Final Estimacion Bayesiana

Code

Matias Moran y Matias Gangui

En este trabajo final vamos a utilizar tecnicas de estimacion bayesiana vistas en la materia para poder obtener y analizar informacion sobre como se comportan los precios de mercado de los jugadores de futbol a nivel internacional.

El dataset que usamos consiste del precio historico de mercado de **8572** jugadores de futbol que alguna vez jugaron algun partido en las competiciones mas importantes del mundo (UCL, Serie A, La Liga, Premier y Bundesliga).

El dataset esta basado en una recopilacion (https://www.kaggle.com/datasets/davidcariboo/player-scores) de datos de la pagina **transfermarkt** la cual contiene informacion historica de ligas, equipos y jugadores de futbol de todo el mundo.

Setup

Importamos las librerias

Hide

```
# Cargamos los paquetes
library(bayesrules)
library(tidyverse)
library(rstanarm)
library(bayesplot)
library(tidybayes)
library(broom.mixed)
library(RColorBrewer)
```

Cargamos el dataset y hacemos feature engineering y reescalamos

Hide

```
# Cargamos el dataset
football = read.csv("players_age_valuation.csv", header = TRUE)

football$player_id <- as.character(football$player_id)

football <- football %>%
  mutate(age_2 = age^2)
football <- football %>%
  mutate(log_market_value_in_eur = log(market_value_in_eur))


head(football)
```

player_id <chr>	name <chr>	a.. <int>	market_value_in_eur <int>	ag… <dbl>	log_market_value_in_eur <dbl>
1 6893	Gabriel Tamas	20	900000	400	13.71015
2 10	Miroslav Klose	26	7000000	676	15.76142
3 26	Roman Weidenfeller	24	1500000	576	14.22098
4 65	Dimitar Berbatov	23	8000000	529	15.89495
5 77	Lúcio	26	13000000	676	16.38046
6 80	Tom Starke	23	400000	529	12.89922

6 rows

Data Exploration

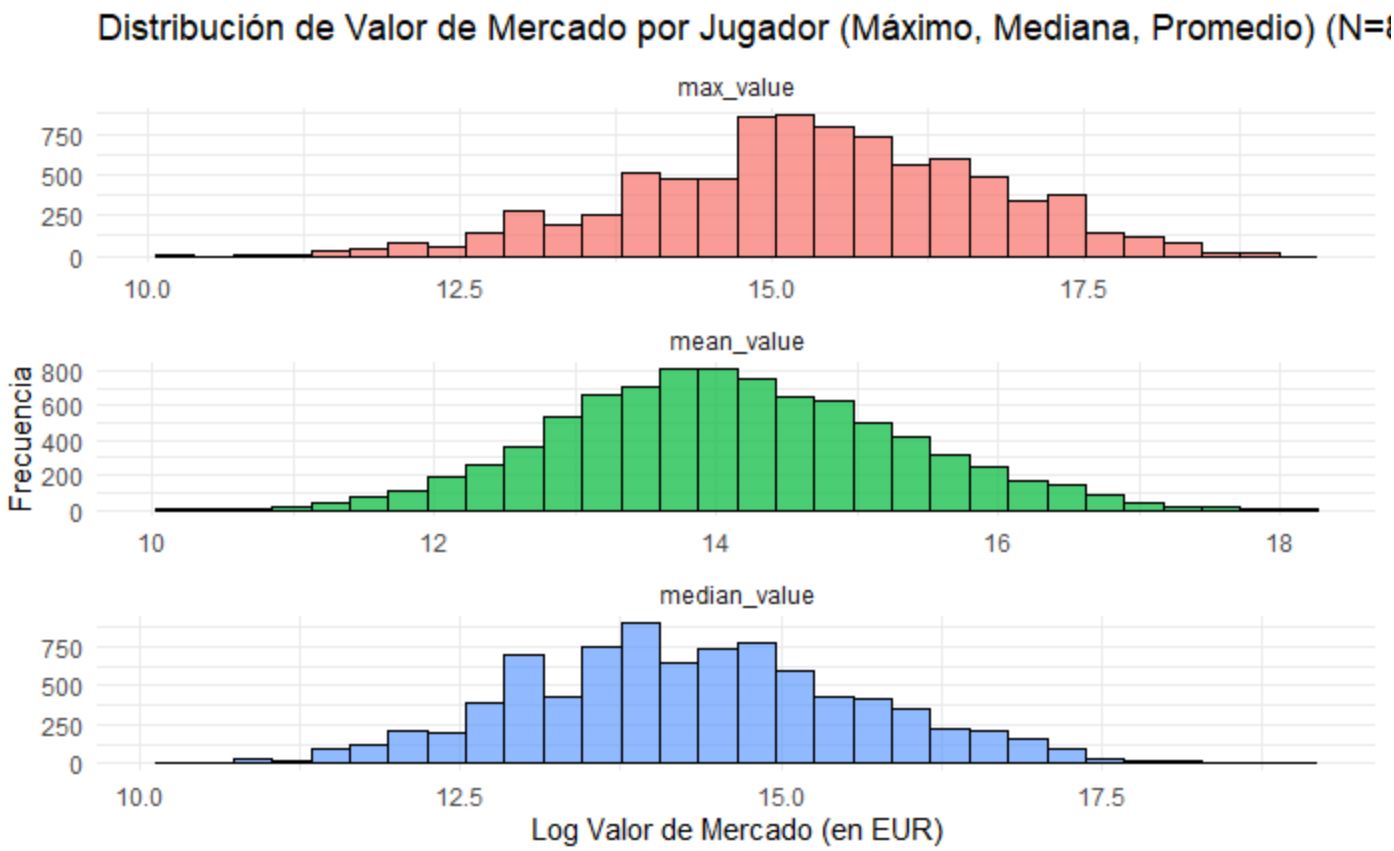
Primero que nada vamos a ver la data de los jugadores para poder entender mejor el problema y poder armar el modelo

Distribucion de market value

Con el dataset completo de los 8572 jugadores para cada jugador vamos a tomar el maximo, promedio y mediana de todos sus log market values historicos en su carrera y vamos a hacer un histograma para ver como se distribuyen los precios de los mismos.

Hide

```
players_stats <- football %>%
  group_by(player_id) %>%
  summarize(
    max_value = max(log_market_value_in_eur, na.rm = TRUE),
    median_value = median(log_market_value_in_eur, na.rm = TRUE),
    mean_value = mean(log_market_value_in_eur, na.rm = TRUE)
  ) %>%
  ungroup() %>%
  pivot_longer(cols = c(max_value, median_value, mean_value),
               names_to = "measure",
               values_to = "log_market_value")

ggplot(data = players_stats, aes(x = log_market_value, fill = measure)) +
  geom_histogram(color = "black", alpha = 0.7, position = "dodge") +
  facet_wrap(~ measure, scales = "free", ncol = 1) +
  ggtitle("Distribución de Valor de Mercado por Jugador (Máximo, Mediana, Promedio) (N=8572)") +
  labs(x = "Log Valor de Mercado (en EUR)", y = "Frecuencia") +
  theme_minimal() +
  theme(legend.position = "none")
```



Vemos que la forma del histograma de los precios (maximos, promedio y mediana) de cada jugador tiene una forma normal, Esto nos dice que la mayoria de los jugadores tienen precios parecidos y hay algunos pocos que tienen precios muy altos y otros muy bajos.

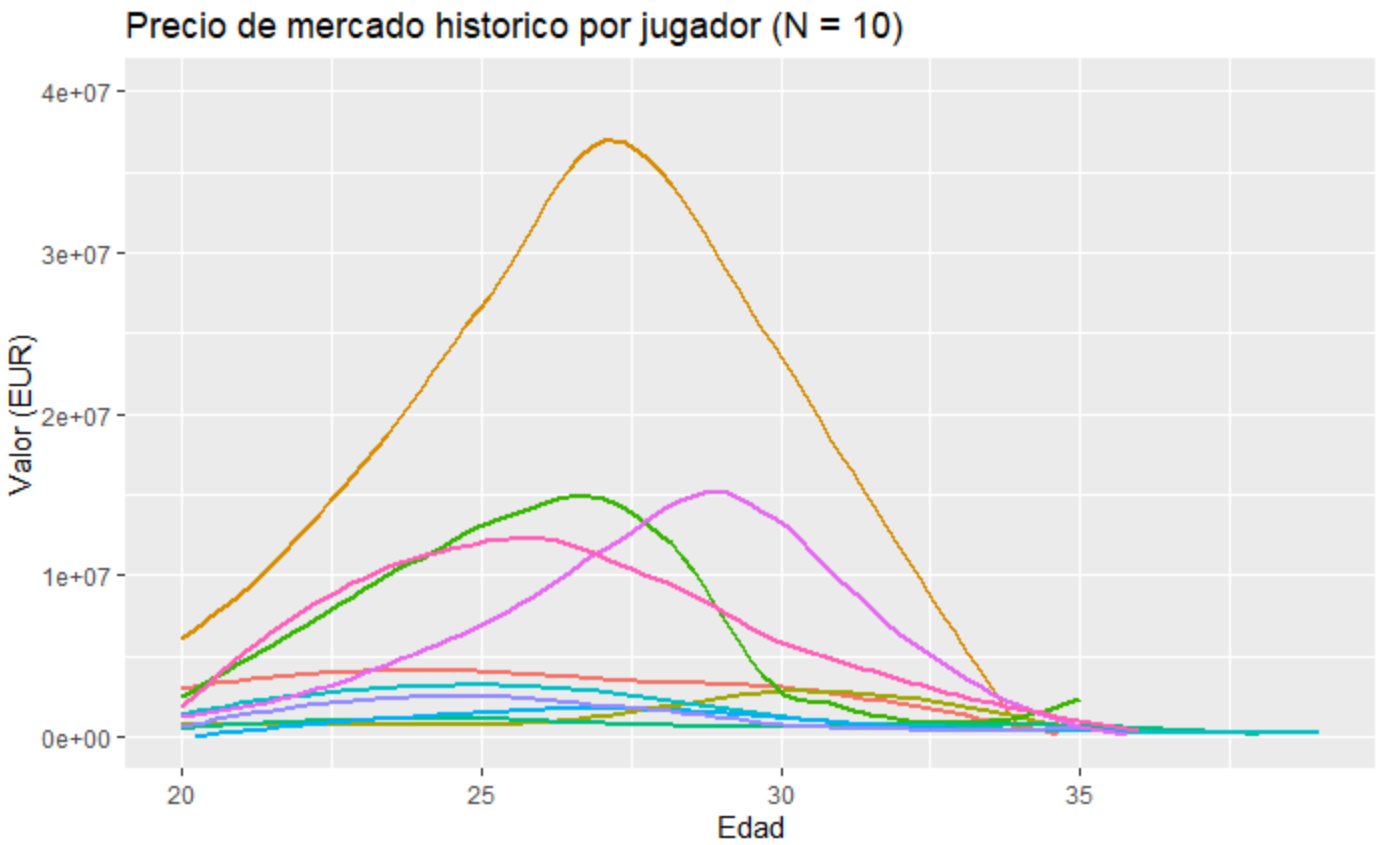
Todo esto nos hace pensar que es razonable asignarle a cada jugador un intercept normalmente distribuido a la hora de modelar.

Luego, vamos a ver como se comporta el precio de mercado de cada jugador en el tiempo que dura su carrera, vamos a agarrar a 10 jugadores y a ver su historial de precios.

Hide

```
set.seed(314159)

players_with_history = football %>%
  filter(age %in% c(20, 35)) %>%
  group_by(player_id) %>%
  filter(any(age == 20) & any(age == 35)) %>%
  pull(player_id) %>%
  unique()

# El dataset cracks va a contener 54 jugadores de futbol de los 8572 originales, es el que vamos
a usar para entrenar al modelo
cracks = football[football$player_id %in% head(players_with_history, 50),]
player_ids = c(28003, 8198, 68290, 342229)
cracks = rbind(cracks, football[football$player_id %in% player_ids,])

# Tomamos 10 jugadores del dataset y vemos sus historiales de precio
cracks_10 = football[football$player_id %in% head(players_with_history, 10),]

ggplot(data = cracks_10, aes(age, market_value_in_eur, color = name)) +
  ggtitle("Precio de mercado historico por jugador (N = 10)") +
  geom_smooth(method = "loess", se = FALSE) +
  theme(legend.position = "none") +
  labs(x="Edad", y="Valor (EUR)") +
  scale_y_continuous(labels = scales::label_dollar()) +
  lims(y = c(0,40000000))
```

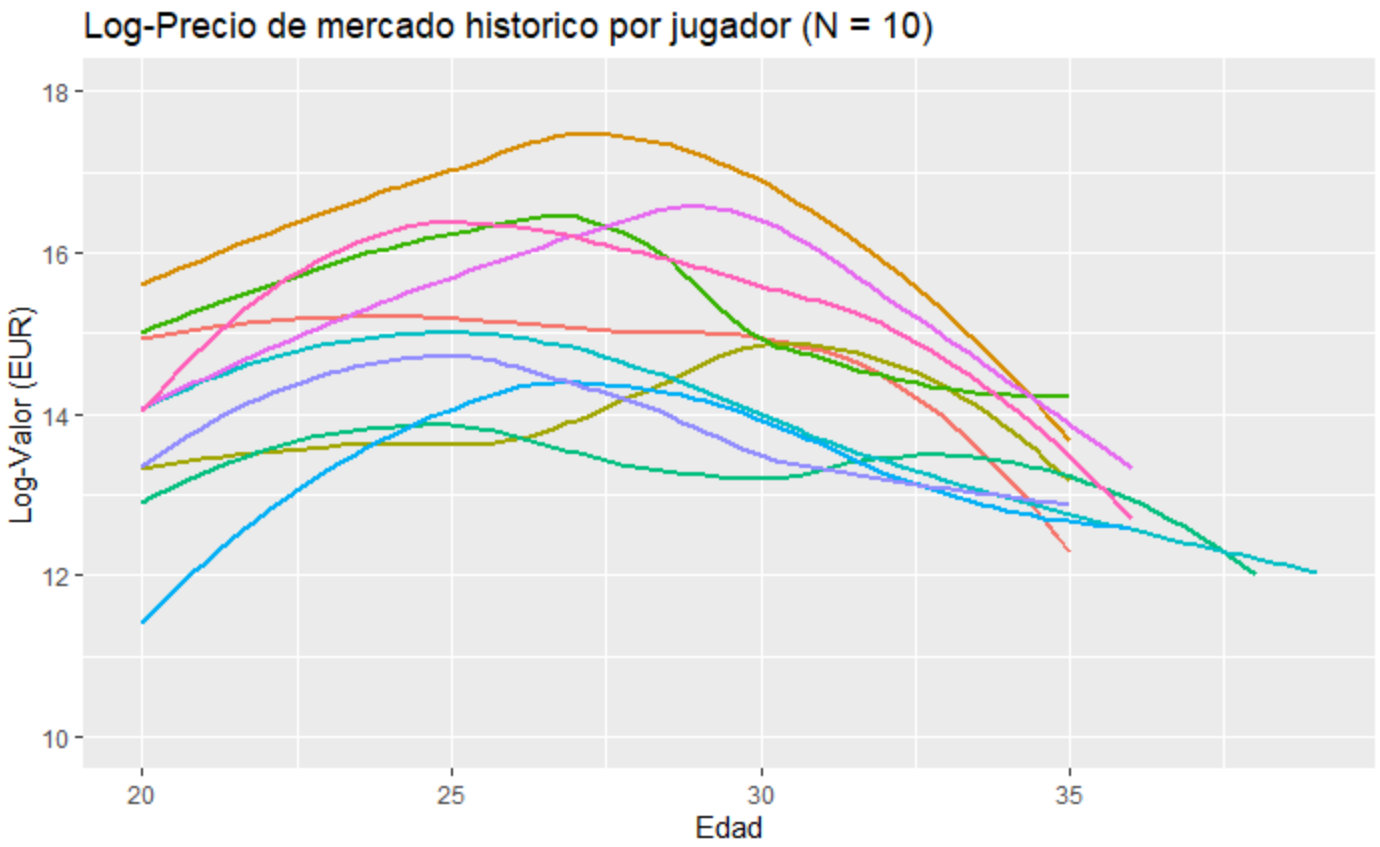
```
Scale for y is already present.
Adding another scale for y, which will replace the existing scale.
```



Hide

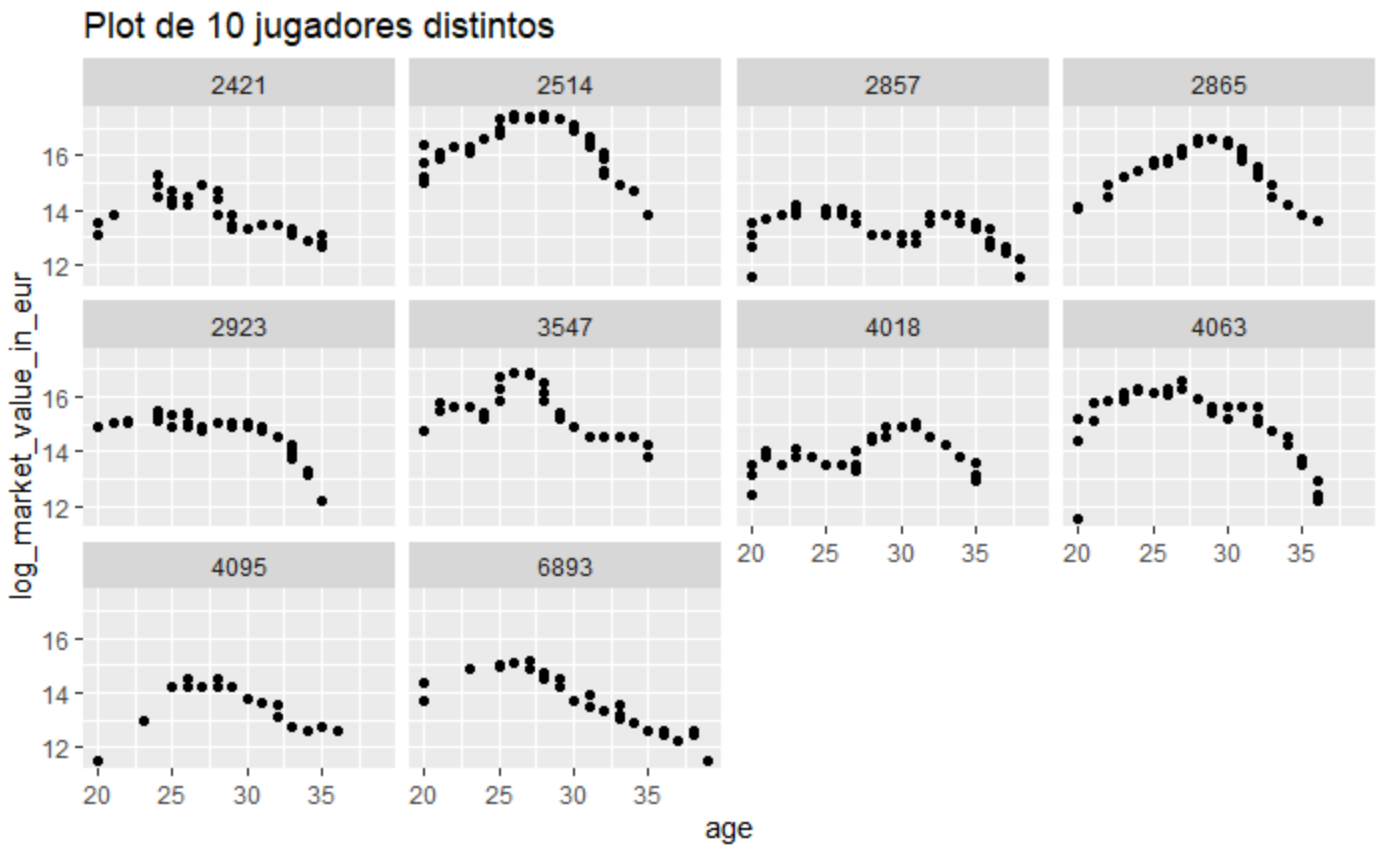
```
ggplot(data = cracks_10, aes(age, log_market_value_in_eur, color = name)) +
  ggtitle("Log-Precio de mercado historico por jugador (N = 10)") +
  geom_smooth(method = "loess", se = FALSE) +
  theme(legend.position = "none") +
  labs(x="Edad", y="Log-Valor (EUR)") +
  scale_y_continuous(labels = scales::label_dollar()) +
  lims(y = c(10,18))
```

```
Scale for y is already present.
Adding another scale for y, which will replace the existing scale.
```



Hide

```
ggplot(cracks_10, aes(x = age, y = log_market_value_in_eur)) +
  geom_point() +
  facet_wrap(~ player_id) +
  ggtitle("Plot de 10 jugadores distintos")
```



Vemos como la tendencia general del historial de precios de un jugador tiene una forma generalmente unimodal en la cual un jugador alcanza su maximo precio de mercado y luego a medida que envejeze su precio va decayendo.

Esto nos hace pensar que una funcion cuadratica podria aproximar bien la curva de precios de un jugador.

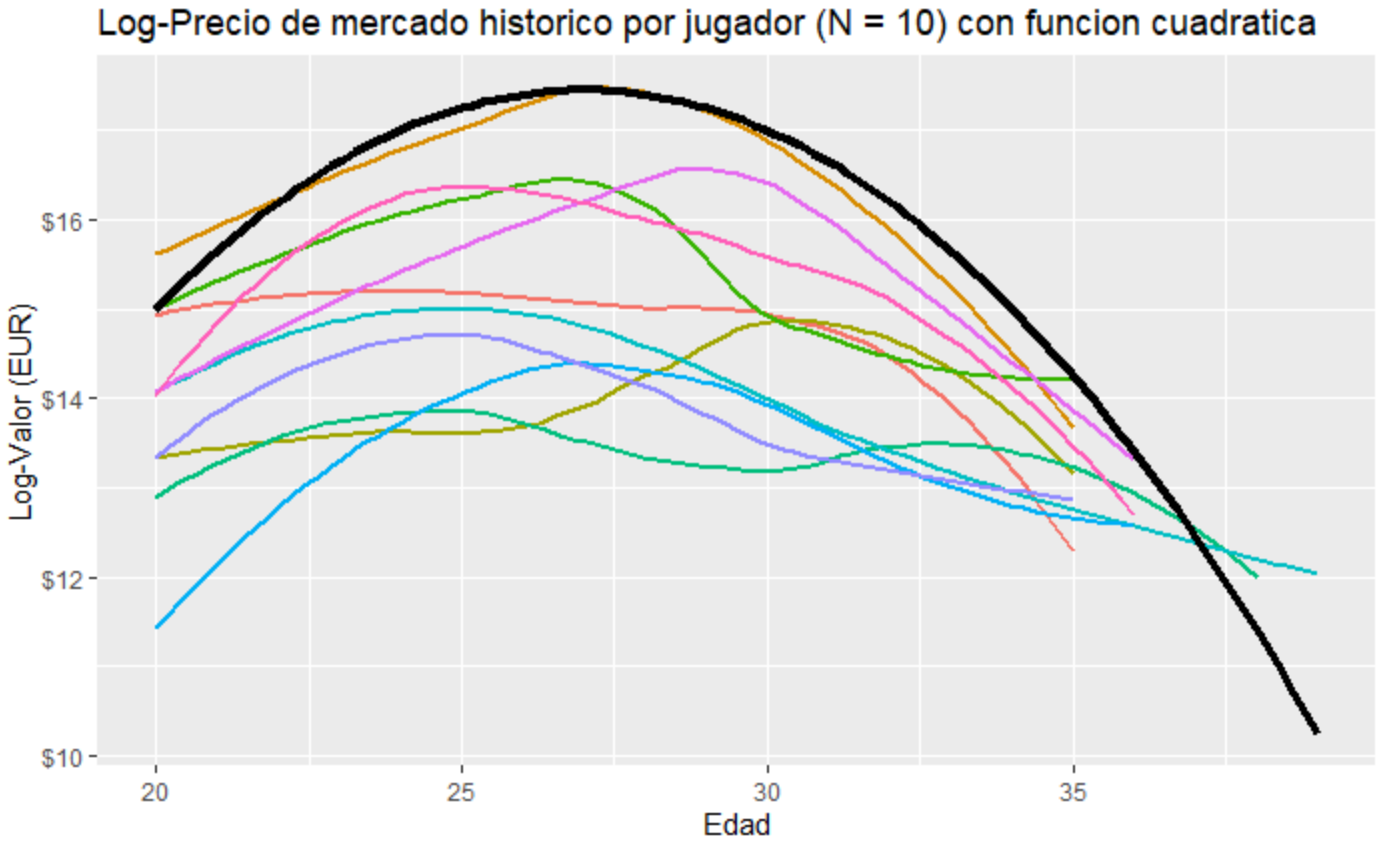
Modelado

Dada la forma de los datos vamos a predecir el logaritmo del precio de mercado de un jugador en base a su edad, vamos a hacer un modelo jerarquico a nivel de jugador y vamos a tratar de modelar una relacion cuadratica y vamos a asignar a cada jugador su propio intercept y haremos que el intercept de cada jugador este normalmente distribuido como vimos en el grafico anterior.

Este modelo tiene la limitacion de que va a hacer que todos los jugadores tengan la misma tendencia a lo largo de su carrera y va a capturar la relacion entre edad y precio para el jugador tipico.

Hide

```
set.seed(314159)

quadratic_fun <- function(x, a, b, c) {
  return(a*x^2 + b*x + c)
}

ggplot(data = cracks_10, aes(age, log_market_value_in_eur, color = name)) +
  ggtitle("Log-Precio de mercado historico por jugador (N = 10) con funcion cuadratica") +
  geom_smooth(method = "loess", se = FALSE) +
  geom_line(data = data.frame(age = seq(min(cracks_10$age), max(cracks_10$age), length.out = 10
0)),
            aes(x = age, y = quadratic_fun(age, -0.05, 2.7, -19)), color = "black", size = 1.5)
+
  theme(legend.position = "none") +
  labs(x="Edad", y="Log-Valor (EUR)") +
  scale_y_continuous(labels = scales::label_dollar())
```



La funcion cuadratica que vemos que puede llegar a aproximar bien a los datos es $y = -19 + 2.7x - 0.05x^2$ esta funcion sale de crear una que este centrada aproximadamente en los picos de precio y tenga una tendencia similar. Vamos a usar los coeficientes de esta funcion para setear los priors de nuestro modelo

Definicion del modelo:

$$\text{data:} \quad Y_{ij}|\beta_{0j}, \beta_1, \beta_2, \sigma_y \sim \mathcal{N}(\mu_{ij}, \sigma_y^2) \quad \text{with} \quad \mu_{ij} = \beta_{0j} + \beta_1 \cdot X_{ij} + \beta_2 \cdot X_{ij}^2$$

$$\text{priors:} \quad \beta_{0j}|\beta_0, \sigma_0 \sim \mathcal{N}(\beta_0, \sigma_0^2)$$

$$\beta_{0c} \sim \mathcal{N}(-19, 20^2)$$

$$\sigma_0 \sim Exp(1)$$

$$\beta_1 \sim \mathcal{N}(2.7, 5^2)$$

$$\beta_2 \sim \mathcal{N}(-0.05, 0.5^2)$$

$$\sigma_y \sim Exp(1)$$

Donde: Y_{ij} es el precio del jugador j a su edad i

X_{ij} es una edad del jugador j

β_{0j} es el intercept de la regresion para el jugador j

β_1 el coeficiente global lineal (el cambio tipico lineal entre edad y precio)

β_2 el coeficiente global cuadratico (el cambio tipico cuadratico entre edad y precio)

σ_0 variabilidad entre los diferentes jugadores

σ_y variabilidad entre valores de un jugador y su modelo de regresion, nos dice la fuerza de relacion entre edad y precio

Hide

```
set.seed(314159)

model_1 <- stan_glmer(
  log_market_value_in_eur ~ age + age_2 + (1 | player_id),
  data = cracks,
  family = gaussian,
  prior_intercept = normal(-19, 20, autoscale = TRUE), #prior para los diferentes intercepts
  prior = normal(location = c(2.7, -0.05), scale = c(5, 0.5)), #prior para los coeficientes
  prior_aux = exponential(1, autoscale = TRUE), #prior para sigma (ruido estadistico)
  prior_covariance = decov(reg = 1, conc = 1, shape = 1, scale = 1), # sigma_mu equivalente a ex
p(1)
  chains = 4,
  iter = 10000,
  seed = 314159
)
```

```
SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.000129 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 1.29 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:    1 / 10000 [  0%]  (Warmup)
Chain 1: Iteration: 1000 / 10000 [ 10%]  (Warmup)
Chain 1: Iteration: 2000 / 10000 [ 20%]  (Warmup)
Chain 1: Iteration: 3000 / 10000 [ 30%]  (Warmup)
Chain 1: Iteration: 4000 / 10000 [ 40%]  (Warmup)
Chain 1: Iteration: 5000 / 10000 [ 50%]  (Warmup)
Chain 1: Iteration: 5001 / 10000 [ 50%]  (Sampling)
Chain 1: Iteration: 6000 / 10000 [ 60%]  (Sampling)
Chain 1: Iteration: 7000 / 10000 [ 70%]  (Sampling)
Chain 1: Iteration: 8000 / 10000 [ 80%]  (Sampling)
Chain 1: Iteration: 9000 / 10000 [ 90%]  (Sampling)
Chain 1: Iteration: 10000 / 10000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 61.377 seconds (Warm-up)
Chain 1:                27.206 seconds (Sampling)
Chain 1:                88.583 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 9.4e-05 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.94 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:    1 / 10000 [  0%]  (Warmup)
Chain 2: Iteration: 1000 / 10000 [ 10%]  (Warmup)
Chain 2: Iteration: 2000 / 10000 [ 20%]  (Warmup)
Chain 2: Iteration: 3000 / 10000 [ 30%]  (Warmup)
Chain 2: Iteration: 4000 / 10000 [ 40%]  (Warmup)
Chain 2: Iteration: 5000 / 10000 [ 50%]  (Warmup)
Chain 2: Iteration: 5001 / 10000 [ 50%]  (Sampling)
Chain 2: Iteration: 6000 / 10000 [ 60%]  (Sampling)
Chain 2: Iteration: 7000 / 10000 [ 70%]  (Sampling)
Chain 2: Iteration: 8000 / 10000 [ 80%]  (Sampling)
Chain 2: Iteration: 9000 / 10000 [ 90%]  (Sampling)
Chain 2: Iteration: 10000 / 10000 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 71.438 seconds (Warm-up)
Chain 2:                27.97 seconds (Sampling)
Chain 2:                99.408 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'continuous' NOW (CHAIN 3).
```

```
Chain 3:
Chain 3: Gradient evaluation took 9.2e-05 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.92 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:    1 / 10000 [  0%]  (Warmup)
Chain 3: Iteration: 1000 / 10000 [ 10%]  (Warmup)
Chain 3: Iteration: 2000 / 10000 [ 20%]  (Warmup)
Chain 3: Iteration: 3000 / 10000 [ 30%]  (Warmup)
Chain 3: Iteration: 4000 / 10000 [ 40%]  (Warmup)
Chain 3: Iteration: 5000 / 10000 [ 50%]  (Warmup)
Chain 3: Iteration: 5001 / 10000 [ 50%]  (Sampling)
Chain 3: Iteration: 6000 / 10000 [ 60%]  (Sampling)
Chain 3: Iteration: 7000 / 10000 [ 70%]  (Sampling)
Chain 3: Iteration: 8000 / 10000 [ 80%]  (Sampling)
Chain 3: Iteration: 9000 / 10000 [ 90%]  (Sampling)
Chain 3: Iteration: 10000 / 10000 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 62.251 seconds (Warm-up)
Chain 3:                29.093 seconds (Sampling)
Chain 3:                91.344 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 0.000118 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 1.18 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:    1 / 10000 [  0%]  (Warmup)
Chain 4: Iteration: 1000 / 10000 [ 10%]  (Warmup)
Chain 4: Iteration: 2000 / 10000 [ 20%]  (Warmup)
Chain 4: Iteration: 3000 / 10000 [ 30%]  (Warmup)
Chain 4: Iteration: 4000 / 10000 [ 40%]  (Warmup)
Chain 4: Iteration: 5000 / 10000 [ 50%]  (Warmup)
Chain 4: Iteration: 5001 / 10000 [ 50%]  (Sampling)
Chain 4: Iteration: 6000 / 10000 [ 60%]  (Sampling)
Chain 4: Iteration: 7000 / 10000 [ 70%]  (Sampling)
Chain 4: Iteration: 8000 / 10000 [ 80%]  (Sampling)
Chain 4: Iteration: 9000 / 10000 [ 90%]  (Sampling)
Chain 4: Iteration: 10000 / 10000 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 62.248 seconds (Warm-up)
Chain 4:                27.879 seconds (Sampling)
Chain 4:                90.127 seconds (Total)
Chain 4:
```

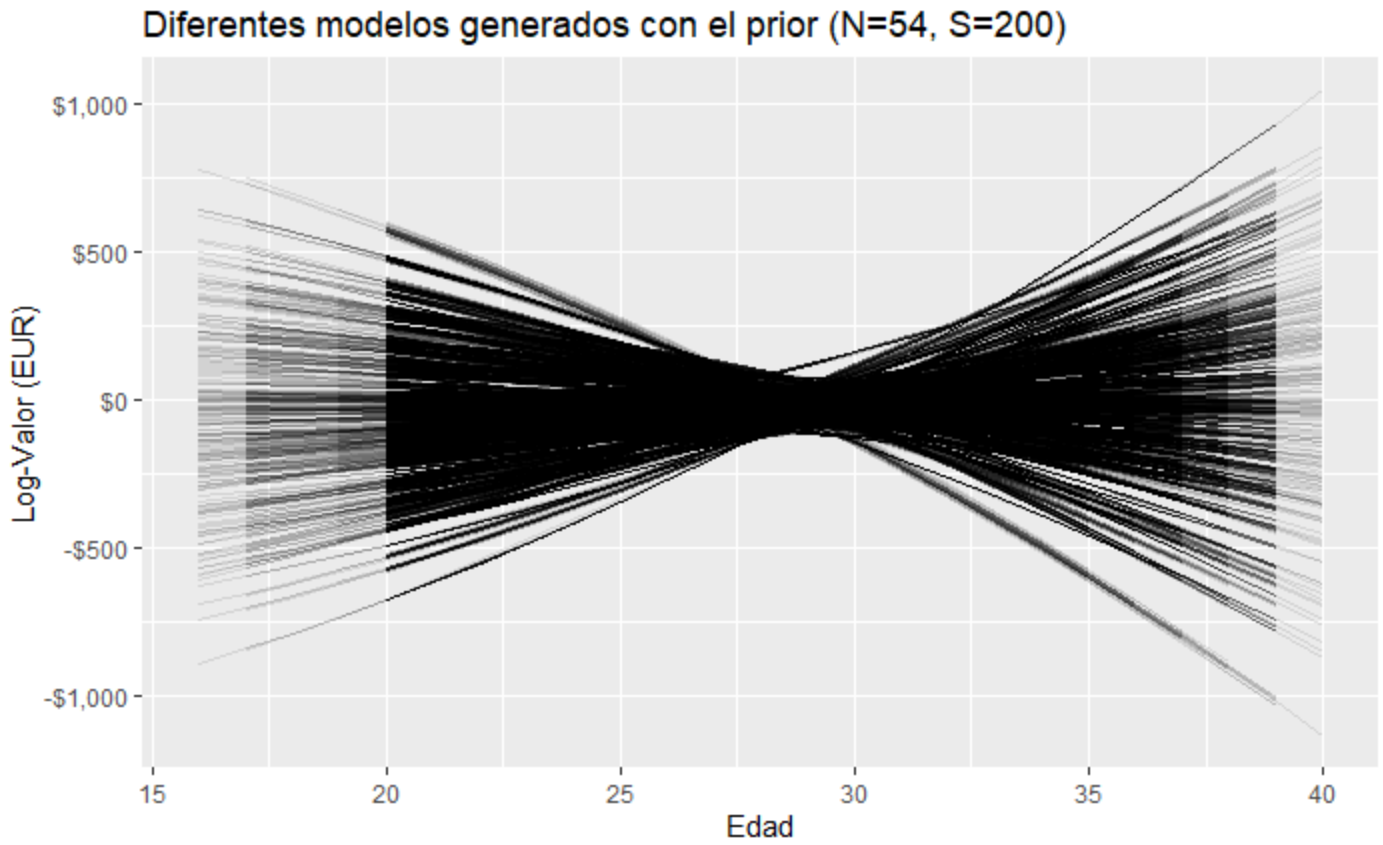
Generamos posibles regresiones usando samples del prior para ver si el modelo es coherente

Hide

```
model_1_prior <- update(model_1, prior_PD = TRUE)
```

Hide

```
cracks %>%
  add_linpred_draws(model_1_prior, ndraws = 200) %>%
  ggplot(aes(x = age, y = log_market_value_in_eur)) +
  geom_line(aes(y = .linpred, group = paste(player_id, .draw)), alpha = 0.1) +
  ggtitle("Diferentes modelos generados con el prior (N=54, S=200)") +
  labs(x="Edad", y="Log-Valor (EUR)") +
  scale_y_continuous(labels = scales::label_dollar())
```



Ahora vamos a ver los resultados de nuestro modelo

Hide

```
prior_summary(model_1)
```

```
Priors for model 'model_1'
------
Intercept (after predictors centered)
  Specified prior:
    ~ normal(location = -19, scale = 20)
  Adjusted prior:
    ~ normal(location = -19, scale = 36)

Coefficients
 ~ normal(location = [ 2.70,-0.05], scale = [5.0,0.5])

Auxiliary (sigma)
  Specified prior:
    ~ exponential(rate = 1)
  Adjusted prior:
    ~ exponential(rate = 0.55)

Covariance
 ~ decov(reg. = 1, conc. = 1, shape = 1, scale = 1)
------
See help('prior_summary.stanreg') for more details
```

Hide

```
tidy_summary_1 <- tidy(model_1, effects = "fixed",
                       conf.int = TRUE, conf.level = 0.95)
tidy_summary_1
```

term <chr>	estimate <dbl>	std.error <dbl>	conf.low <dbl>	conf.high <dbl>
(Intercept)	-6.5533460	0.5832947216	-7.71378343	-5.40198932
age	1.6336574	0.0393001097	1.55515544	1.71189497
age_2	-0.0302452	0.0006840102	-0.03161065	-0.02887785

3 rows

Vemos que:

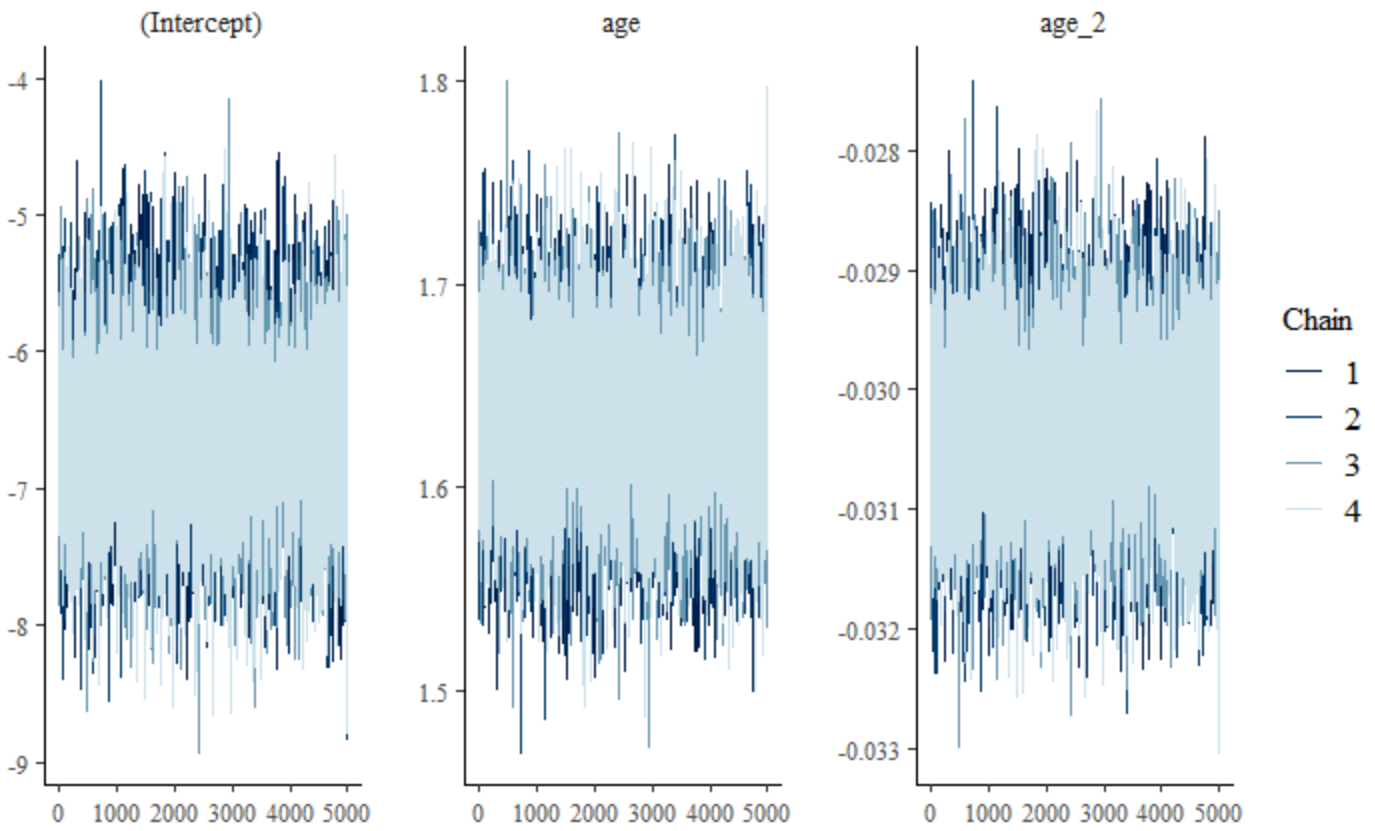
el intervalo de credibilidad del 95% para β_2 esta entre -0.031 y -0.028, como el intervalo completo es negativo, nos da evidencia significativa de que la tendencia del precio de un jugador tipico aumenta hasta llegar sus valores mas altos y luego empieza a decaer.

Diagnostico del modelo

Chequeamos que las cadenas se vean bien y hacemos varios sanity checks vemos las trazas de las cadenas, un plot de densidad y su historial de autocorrelacion para ver que esten bien.

Hide

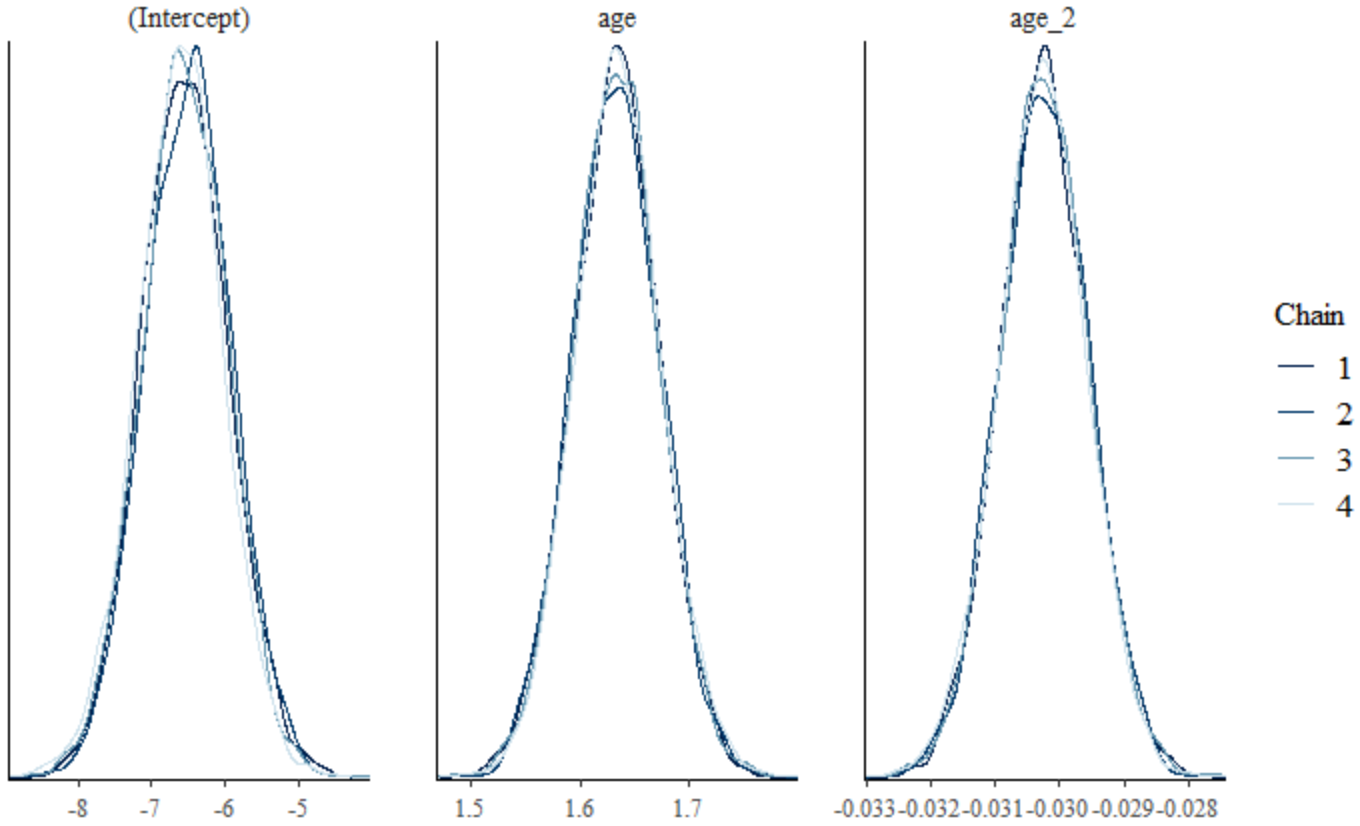
```
mcmc_trace(model_1, pars = c("(Intercept)","age","age_2"))
```



Vemos que las trazas de las cadenas estan bien pudieron converger a la posterior y no presentan muchos rechazos ni autocorrelacion, (solo mostramos las cadenas de los parametros globales por temas de visualizacion, los parametros de los intercepts presentan los mismos resultados)

Hide

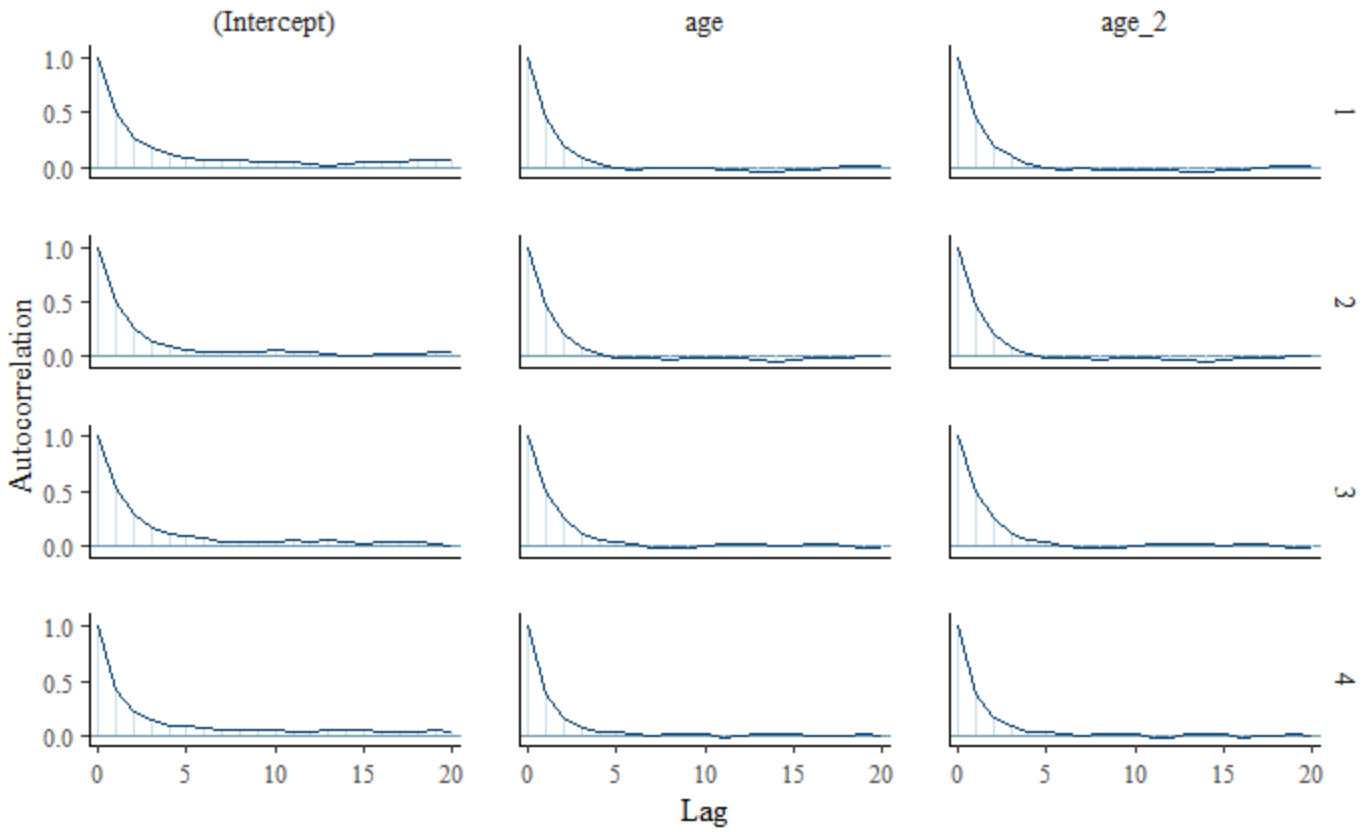
```
mcmc_dens_overlay(model_1, pars = c("(Intercept)","age","age_2"))
```



Vemos que la densidad de las cadenas es parecida (solo mostramos las cadenas de los parametros globales por temas de visualizacion, los parametros de los intercepts presentan los mismos resultados)

Hide

```
mcmc_acf(model_1, pars = c("(Intercept)","age","age_2"))
```



Vemos que la autocorrelacion de las cadenas baja a medida que van convergiendo a la posterior (solo mostramos las cadenas de los parametros globales por temas de visualizacion, los parametros de los intercepts presentan los mismos resultados)

Hide

```
rhat(model_1)
```

(Intercept)	age
1.0033054	1.0006558
age_2	b[(Intercept) player_id:10003]
1.0006348	1.0138140
b[(Intercept) player_id:10255]	b[(Intercept) player_id:10329]
1.0152135	1.0129196
b[(Intercept) player_id:12124]	b[(Intercept) player_id:12704]
1.0157299	1.0153341
b[(Intercept) player_id:13217]	b[(Intercept) player_id:13443]
1.0163534	1.0128723
b[(Intercept) player_id:15074]	b[(Intercept) player_id:15511]
1.0151950	1.0158779
b[(Intercept) player_id:15650]	b[(Intercept) player_id:15680]
1.0151857	1.0146427
b[(Intercept) player_id:16120]	b[(Intercept) player_id:16136]
1.0164939	1.0164846
b[(Intercept) player_id:16498]	b[(Intercept) player_id:16816]
1.0143546	1.0150186
b[(Intercept) player_id:16873]	b[(Intercept) player_id:16902]
1.0169682	1.0159105
b[(Intercept) player_id:17127]	b[(Intercept) player_id:17396]
1.0170214	1.0159572
b[(Intercept) player_id:18829]	b[(Intercept) player_id:18871]
1.0180425	1.0134172
b[(Intercept) player_id:19041]	b[(Intercept) player_id:19447]
1.0171944	1.0165535
b[(Intercept) player_id:2421]	b[(Intercept) player_id:2514]
1.0156323	1.0166386
b[(Intercept) player_id:28003]	b[(Intercept) player_id:2857]
1.0175251	1.0166881
b[(Intercept) player_id:2865]	b[(Intercept) player_id:2923]
1.0157434	1.0153821
b[(Intercept) player_id:342229]	b[(Intercept) player_id:3547]
1.0154163	1.0150411
b[(Intercept) player_id:4018]	b[(Intercept) player_id:4063]
1.0148035	1.0163766
b[(Intercept) player_id:4095]	b[(Intercept) player_id:4276]
1.0124200	1.0163007
b[(Intercept) player_id:4360]	b[(Intercept) player_id:4698]
1.0169129	1.0141842
b[(Intercept) player_id:5404]	b[(Intercept) player_id:6033]
1.0162015	1.0160749
b[(Intercept) player_id:6160]	b[(Intercept) player_id:6287]
1.0153789	1.0148410
b[(Intercept) player_id:68290]	b[(Intercept) player_id:6838]
1.0170896	1.0157453
b[(Intercept) player_id:6893]	b[(Intercept) player_id:7600]
1.0160104	1.0167957
b[(Intercept) player_id:7767]	b[(Intercept) player_id:7914]
1.0146846	1.0166622
b[(Intercept) player_id:8184]	b[(Intercept) player_id:8198]
1.0166517	1.0175196

```
           b[(Intercept) player_id:8552]           b[(Intercept) player_id:8730]
                                1.0166465                               1.0142281
           b[(Intercept) player_id:8883]           b[(Intercept) player_id:9824]
                                1.0161858                               1.0153639
           b[(Intercept) player_id:9988]                               sigma
                                1.0170311                               0.9999184
Sigma[player_id:(Intercept),(Intercept)]
                                1.0040554
```

Vemos que los coeficientes de R-hat son cercanos a uno para todos los parametros, indicando que la variabilidad entre las cadenas y la variabilidad de cada una son bastante similares.

Hide

```
neff_ratio(model_1)
```

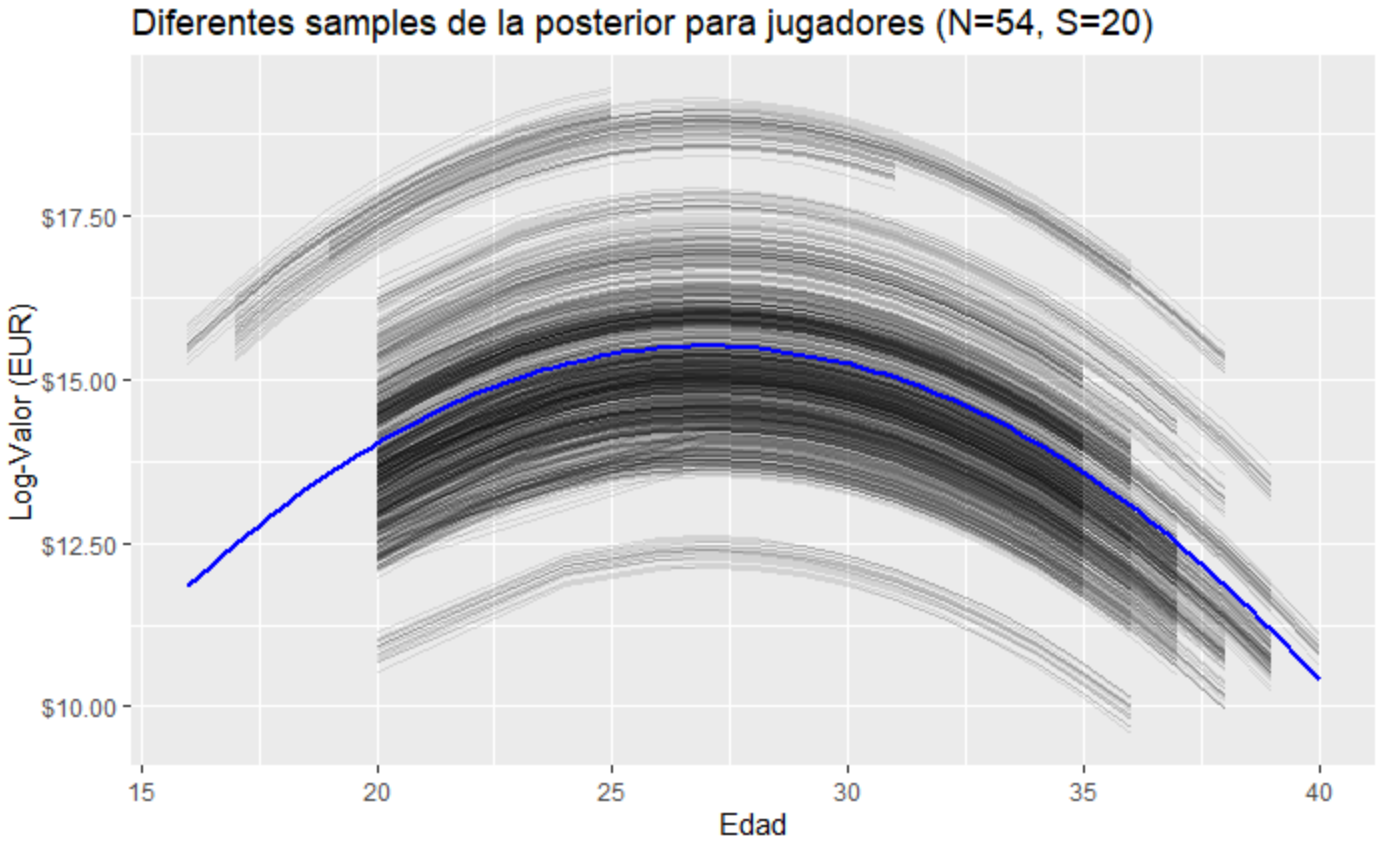
(Intercept)	age
0.16880	0.38250
age_2	b[(Intercept) player_id:10003]
0.38195	0.04030
b[(Intercept) player_id:10255]	b[(Intercept) player_id:10329]
0.03620	0.04435
b[(Intercept) player_id:12124]	b[(Intercept) player_id:12704]
0.03425	0.03825
b[(Intercept) player_id:13217]	b[(Intercept) player_id:13443]
0.03740	0.04150
b[(Intercept) player_id:15074]	b[(Intercept) player_id:15511]
0.03575	0.03620
b[(Intercept) player_id:15650]	b[(Intercept) player_id:15680]
0.03840	0.03790
b[(Intercept) player_id:16120]	b[(Intercept) player_id:16136]
0.03555	0.03465
b[(Intercept) player_id:16498]	b[(Intercept) player_id:16816]
0.03785	0.03890
b[(Intercept) player_id:16873]	b[(Intercept) player_id:16902]
0.03500	0.03470
b[(Intercept) player_id:17127]	b[(Intercept) player_id:17396]
0.03295	0.03450
b[(Intercept) player_id:18829]	b[(Intercept) player_id:18871]
0.03240	0.04150
b[(Intercept) player_id:19041]	b[(Intercept) player_id:19447]
0.03225	0.03495
b[(Intercept) player_id:2421]	b[(Intercept) player_id:2514]
0.03520	0.03330
b[(Intercept) player_id:28003]	b[(Intercept) player_id:2857]
0.03205	0.03540
b[(Intercept) player_id:2865]	b[(Intercept) player_id:2923]
0.03750	0.03535
b[(Intercept) player_id:342229]	b[(Intercept) player_id:3547]
0.03755	0.03700
b[(Intercept) player_id:4018]	b[(Intercept) player_id:4063]
0.03845	0.03540
b[(Intercept) player_id:4095]	b[(Intercept) player_id:4276]
0.04480	0.03645
b[(Intercept) player_id:4360]	b[(Intercept) player_id:4698]
0.03300	0.03885
b[(Intercept) player_id:5404]	b[(Intercept) player_id:6033]
0.03390	0.03585
b[(Intercept) player_id:6160]	b[(Intercept) player_id:6287]
0.03710	0.03845
b[(Intercept) player_id:68290]	b[(Intercept) player_id:6838]
0.03355	0.03675
b[(Intercept) player_id:6893]	b[(Intercept) player_id:7600]
0.03695	0.03170
b[(Intercept) player_id:7767]	b[(Intercept) player_id:7914]
0.03940	0.03380
b[(Intercept) player_id:8184]	b[(Intercept) player_id:8198]
0.03345	0.03410

```
           b[(Intercept) player_id:8552]           b[(Intercept) player_id:8730]
                                 0.03385                                 0.04045
           b[(Intercept) player_id:8883]           b[(Intercept) player_id:9824]
                                 0.03730                                 0.03690
           b[(Intercept) player_id:9988]                                   sigma
                                 0.03440                                 0.58310
Sigma[player_id:(Intercept),(Intercept)]
                                 0.05500
```

Vemos que el effective samples para los parametros globales estan entre el 16 y 58 porciento, lo cual creemos razonable pero para los intercepts esto baja a 3%, esto nos llamo la atencion, pero el resto de metricas nos indican una buena salud de las cadenas

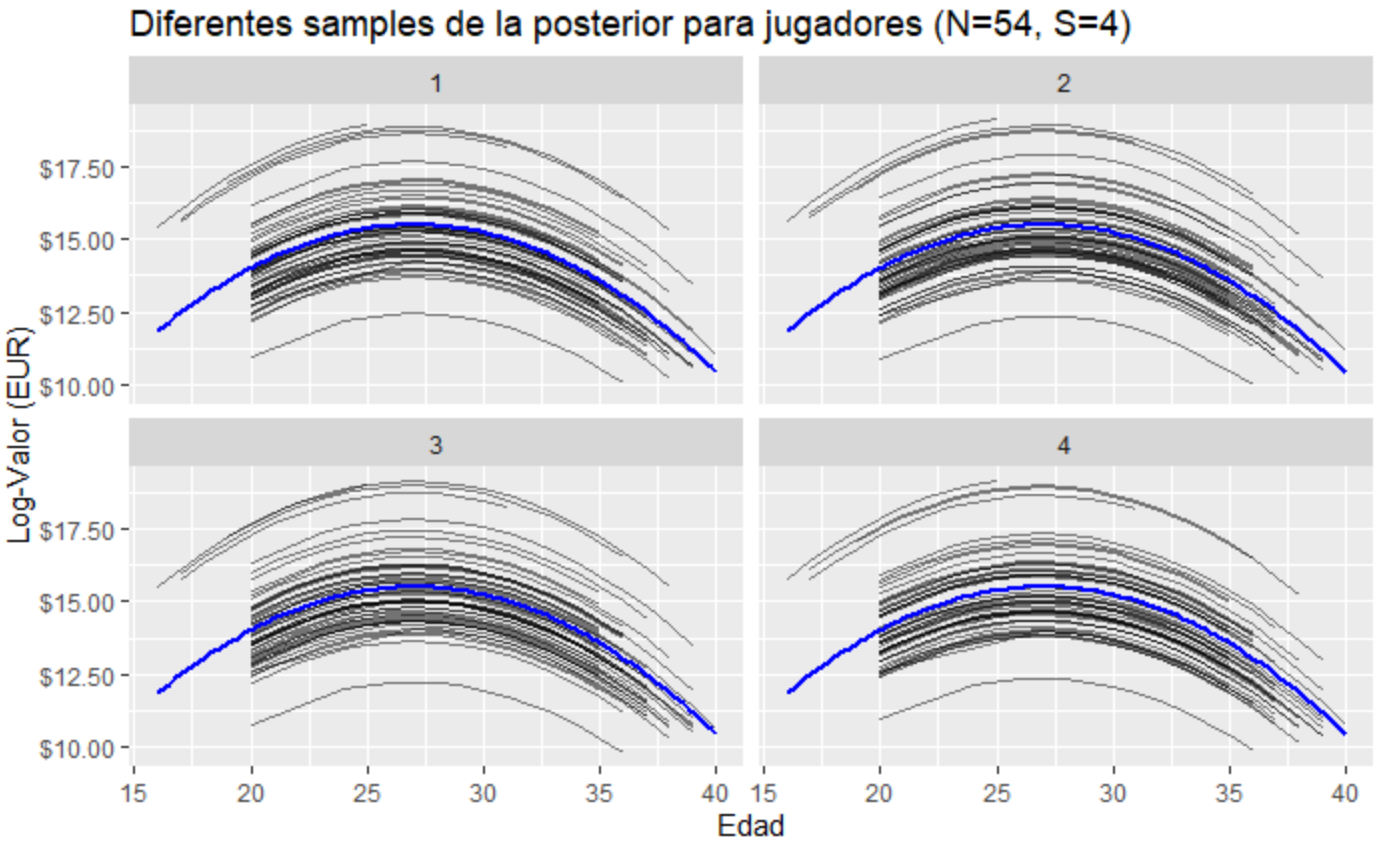
Hide

```
set.seed(314159)

cracks %>%
  add_linpred_draws(model_1, ndraws = 20) %>%
  ggplot(aes(x = age, y = log_market_value_in_eur)) +
  geom_line(aes(y = .linpred, group = paste(player_id, .draw)), alpha = 0.1) +
  geom_line(data = data.frame(age = seq(min(cracks$age), max(cracks$age), length.out = 100)),
            aes(x = age, y = quadratic_fun(age, tidy_summary_1$estimate[3],   tidy_summary_1$est
imate[2], tidy_summary_1$estimate[1])),
            color = "blue", size = 1) +
  ggtitle("Diferentes samples de la posterior para jugadores (N=54, S=20)") +
  labs(x="Edad", y="Log-Valor (EUR)") +
  scale_y_continuous(labels = scales::label_dollar())
```



Hide

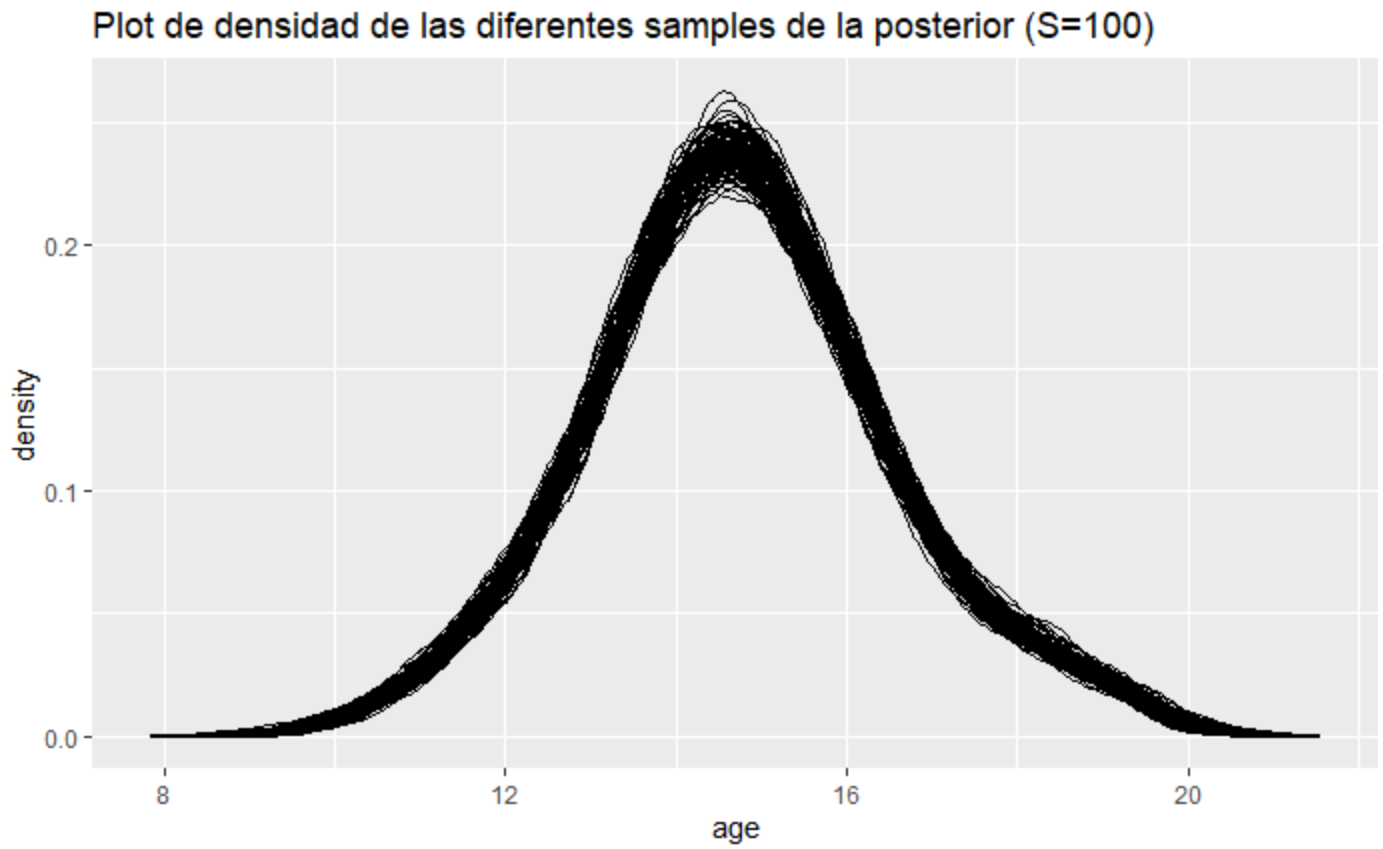
```
cracks %>%
  add_linpred_draws(model_1, ndraws = 4) %>%
  ggplot(aes(x = age, y=log_market_value_in_eur)) +
  geom_line(aes(y = .linpred, group = paste(player_id, .draw)), alpha = 0.5) +
    geom_line(data = data.frame(age = seq(min(cracks$age), max(cracks$age), length.out = 100)),
              aes(x = age, y = quadratic_fun(age, tidy_summary_1$estimate[3],   tidy_summary_1$est
imate[2], tidy_summary_1$estimate[1])),
              color = "blue", size = 1) +
  facet_wrap(~ .draw) +
  ggtitle("Diferentes samples de la posterior para jugadores (N=54, S=4)") +
  labs(x="Edad", y="Log-Valor (EUR)") +
  scale_y_continuous(labels = scales::label_dollar())
```



Vemos diferentes samples de la posterior para nuestros jugadores y observamos como la mayoria de estas estan en el medio, cerca de la regresion para el jugador tipico (linea azul).

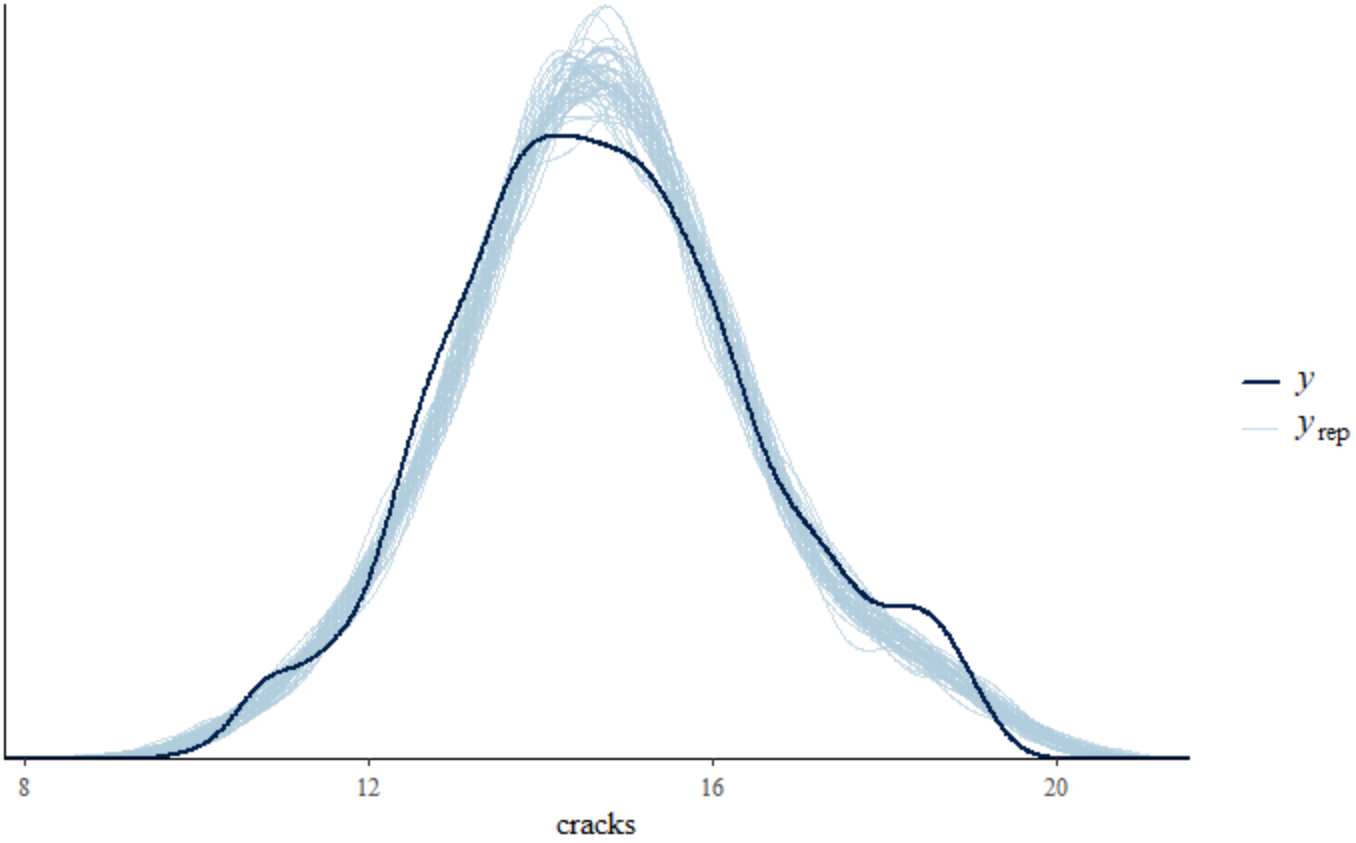
Hide

```
set.seed(314159)

cracks %>%
    add_predicted_draws(model_1, ndraws = 100) %>%
    ggplot(aes(x = age)) +
      geom_density(aes(x = .prediction, group = .draw), alpha = 0.3) +
      ggtitle("Plot de densidad de las diferentes samples de la posterior (S=100)")
```



Hide

```
pp_check(model_1, nreps = 50) +
     xlab("cracks")
```



Vemos que la densidad de los modelos es bastante similar a la de los datos.

Veamos los distintos intercepts para algunos jugadores

Hide

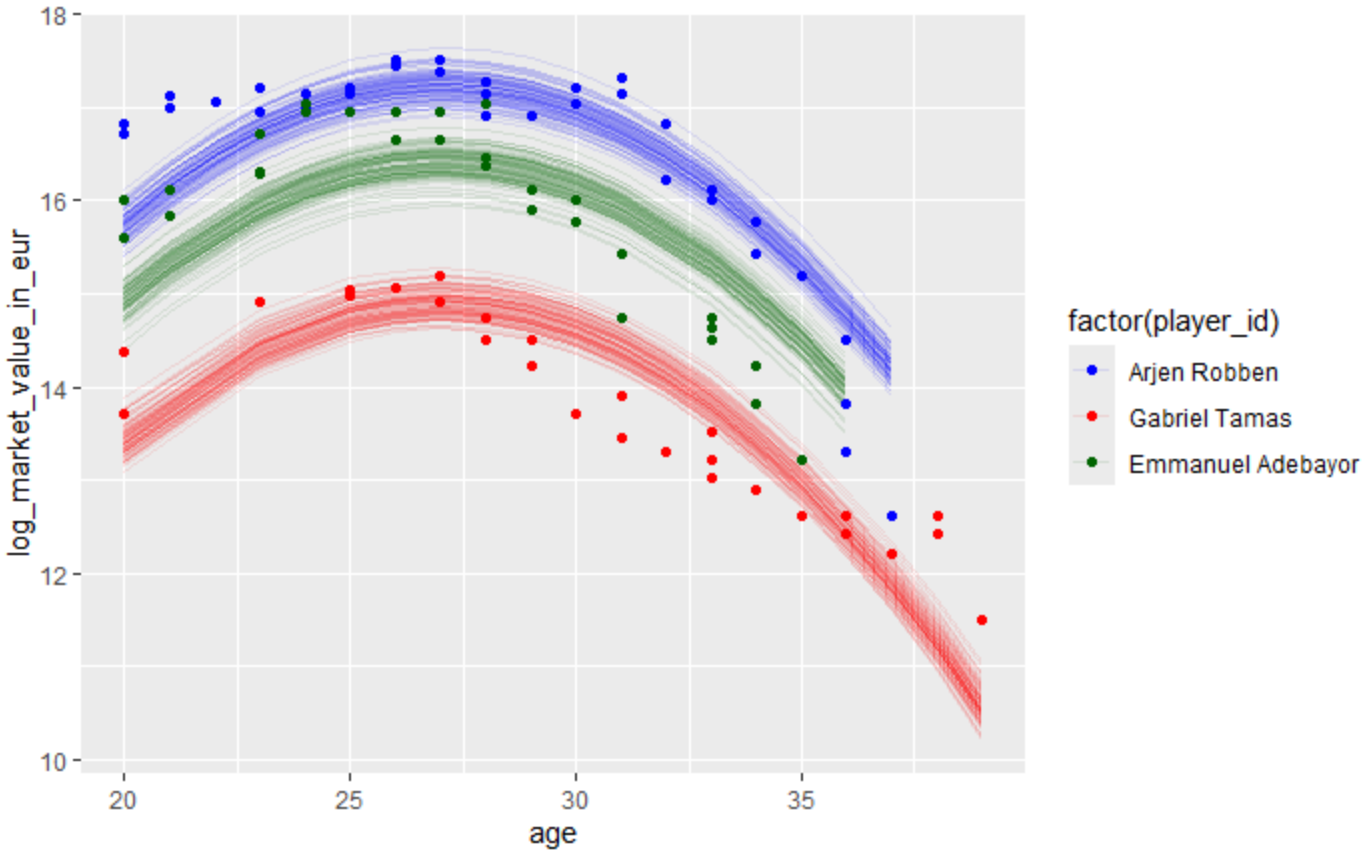
```
player_summaries_1 <- model_1 %>%
  spread_draws(`(Intercept)`, b[,player_id]) %>%
  mutate(player_intercept = `(Intercept)` + b) %>%
  select(-`(Intercept)`, -b) %>%
  median_qi(.width = 0.80) %>%
  select(player_id, player_intercept, .lower, .upper)

player_summaries_1 %>%
  filter(player_id %in% c("player_id:4360", "player_id:6893", "player_id:7767"))
```

player_id <chr>	player_intercept <dbl>	.lower <dbl>	.upper <dbl>
player_id:4360	-4.819589	-5.547573	-4.091937
player_id:6893	-7.158221	-7.890850	-6.429816
player_id:7767	-4.964635	-5.705237	-4.219584

3 rows

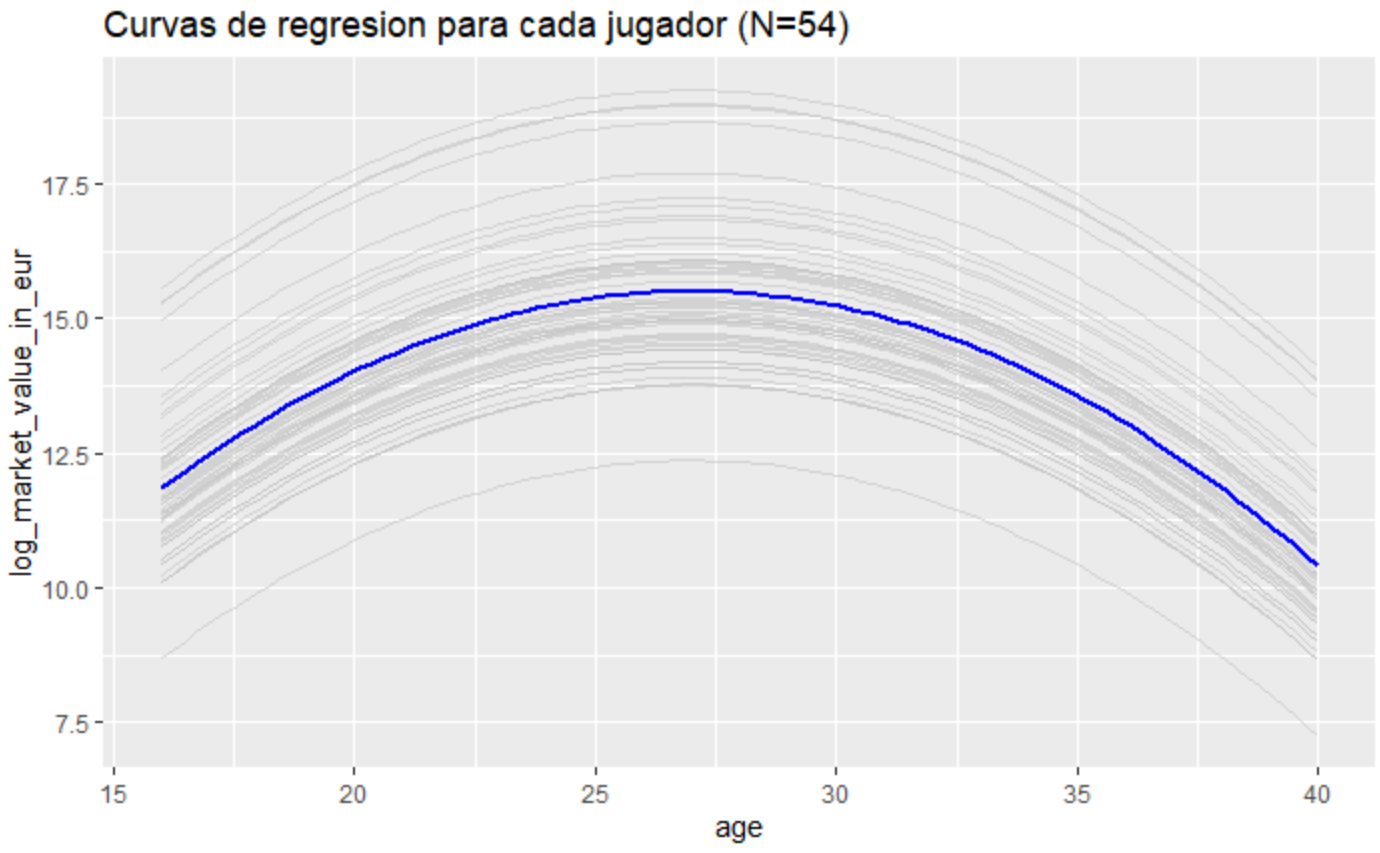
Hide

```
cracks %>%
  filter(player_id %in% c(4360, 8883, 6893)) %>%
  add_linpred_draws(model_1, ndraws = 100) %>%
  ggplot(aes(x = age, y = log_market_value_in_eur)) +
  geom_line(
    aes(y = .linpred, group = paste(player_id, .draw), color = factor(player_id)),
    alpha = 0.1) +
  geom_point(aes(color = factor(player_id))) +
  scale_color_manual(values = c("4360" = "blue", "6893" = "red", "8883" = "darkgreen"), labels =
c("4360" = "Arjen Robben", "6893" = "Gabriel Tamas", "8883" = 'Emmanuel Adebayor'))
```



Hide

```
quadratic_data <- player_summaries_1 %>%
  mutate(age = list(seq(min(cracks$age), max(cracks$age), length.out = 100))) %>%
  unnest(age) %>%
  mutate(log_market_value_in_eur = mapply(quadratic_fun, age, tidy_summary_1$estimate[3], tidy_s
ummary_1$estimate[2],player_intercept))

ggplot(cracks, aes(x = age, y = log_market_value_in_eur)) +
  ggtitle("Curvas de regresion para cada jugador (N=54)") +
  geom_line(data = quadratic_data,
            aes(x = age, y = log_market_value_in_eur, group = player_id),
            color = "gray",
            alpha=0.5
            ) +
  geom_line(data = data.frame(age = seq(min(cracks$age), max(cracks$age), length.out = 100)),
            aes(x = age, y = quadratic_fun(age, tidy_summary_1$estimate[3], tidy_summary_1$estim
ate[2], tidy_summary_1$estimate[1])),
            color = "blue",
            size = 1
            )
```

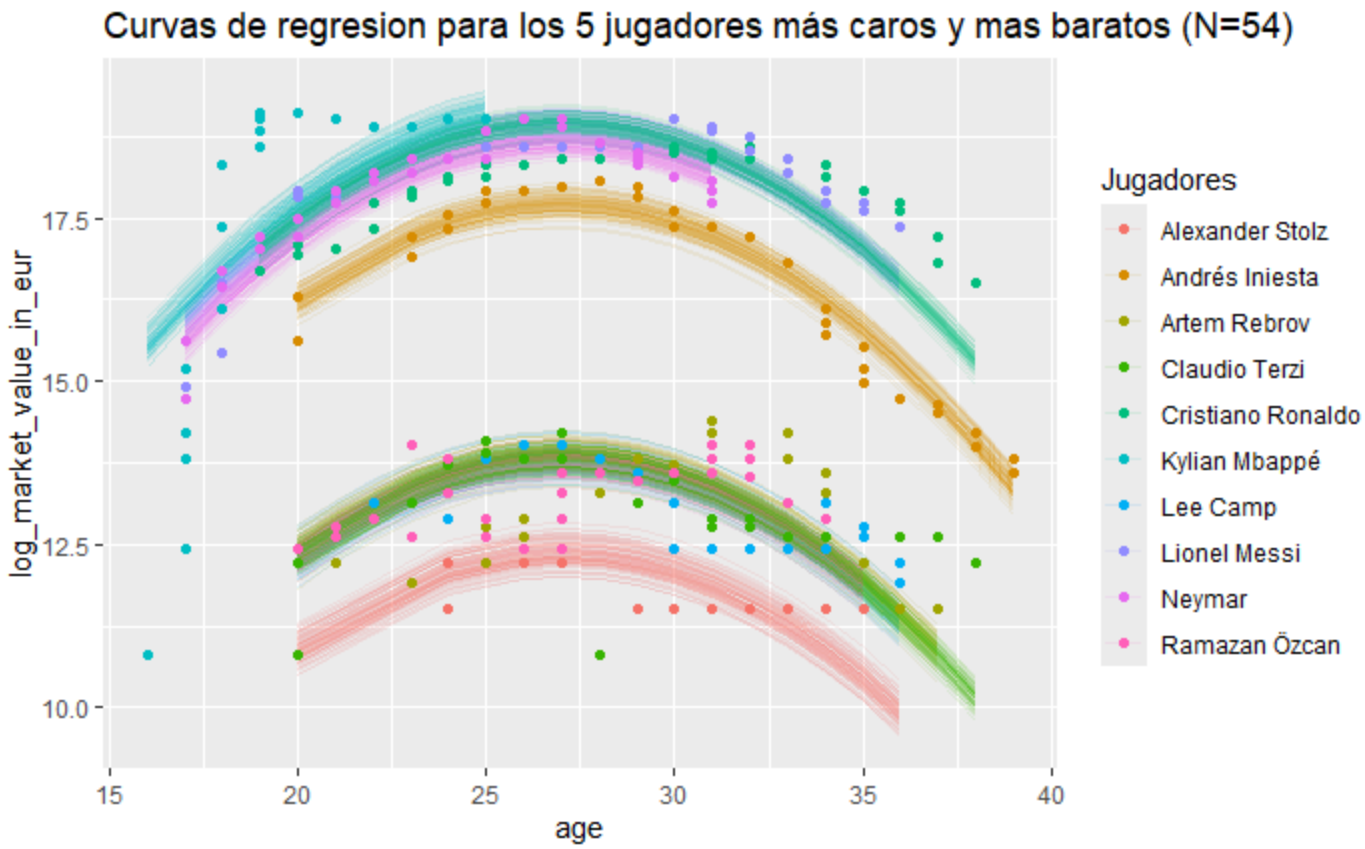


Hide

NA

Hide

```
expensive_playes_ids <- player_summaries_1 %>%
  arrange(desc(player_intercept)) %>%
  head(5) %>%
  select(player_id) %>%
  mutate(player_id = as.integer(sub("player_id:", "", player_id)))

cheap_player_ids <- player_summaries_1 %>%
  arrange(player_intercept) %>%
  head(5) %>%
  select(player_id) %>%
  mutate(player_id = as.integer(sub("player_id:", "", player_id)))

player_ids <- rbind(expensive_playes_ids, cheap_player_ids)$player_id

player_names <- cracks %>%
  filter(player_id %in% player_ids) %>%
  select(player_id, name) %>%
  distinct()

cracks %>%
  filter(player_id %in% player_ids) %>%
  add_linpred_draws(model_1, ndraws = 100) %>%
  ggplot(aes(x = age, y = log_market_value_in_eur)) +
  ggtitle("Curvas de regresion para los 5 jugadores más caros y mas baratos (N=54)") +
  geom_line( aes(y = .linpred, group = paste(player_id, .draw), color = name), alpha = 0.1) +
  geom_point(aes(color = name)) +
  guides(color = guide_legend(title = "Jugadores")) +
  theme(legend.position = "right")
```



Veamos la variabilidad explicada por la diferencia entre los jugadores vs diferencias de cada jugador con el modelo

Hide

```
tidy_sigma <- tidy(model_1, effects = "ran_pars")
tidy_sigma
```

term <chr>	group <chr>	estimate <dbl>
sd_(Intercept).player_id	player_id	1.4352971
sd_Observation.Residual	Residual	0.7498305

2 rows

Hide

```
sigma_0 <- tidy_sigma[1,3]
sigma_y <- tidy_sigma[2,3]
sigma_0^2 / (sigma_0^2 + sigma_y^2)
```

estimate <dbl>
0.7855925

1 row

Hide

```
sigma_y^2 / (sigma_0^2 + sigma_y^2)
```

estimate <dbl>
0.2144075

1 row

Vemos que la varianza de σ_0 es del 78% y la de σ_y es del 21%, esto nos indica que una mayor parte de la varianza se explica por la variabilidad del precio entre los diferentes jugadores que la variabilidad de un jugador con su curva de regresion

Posterior Predictions

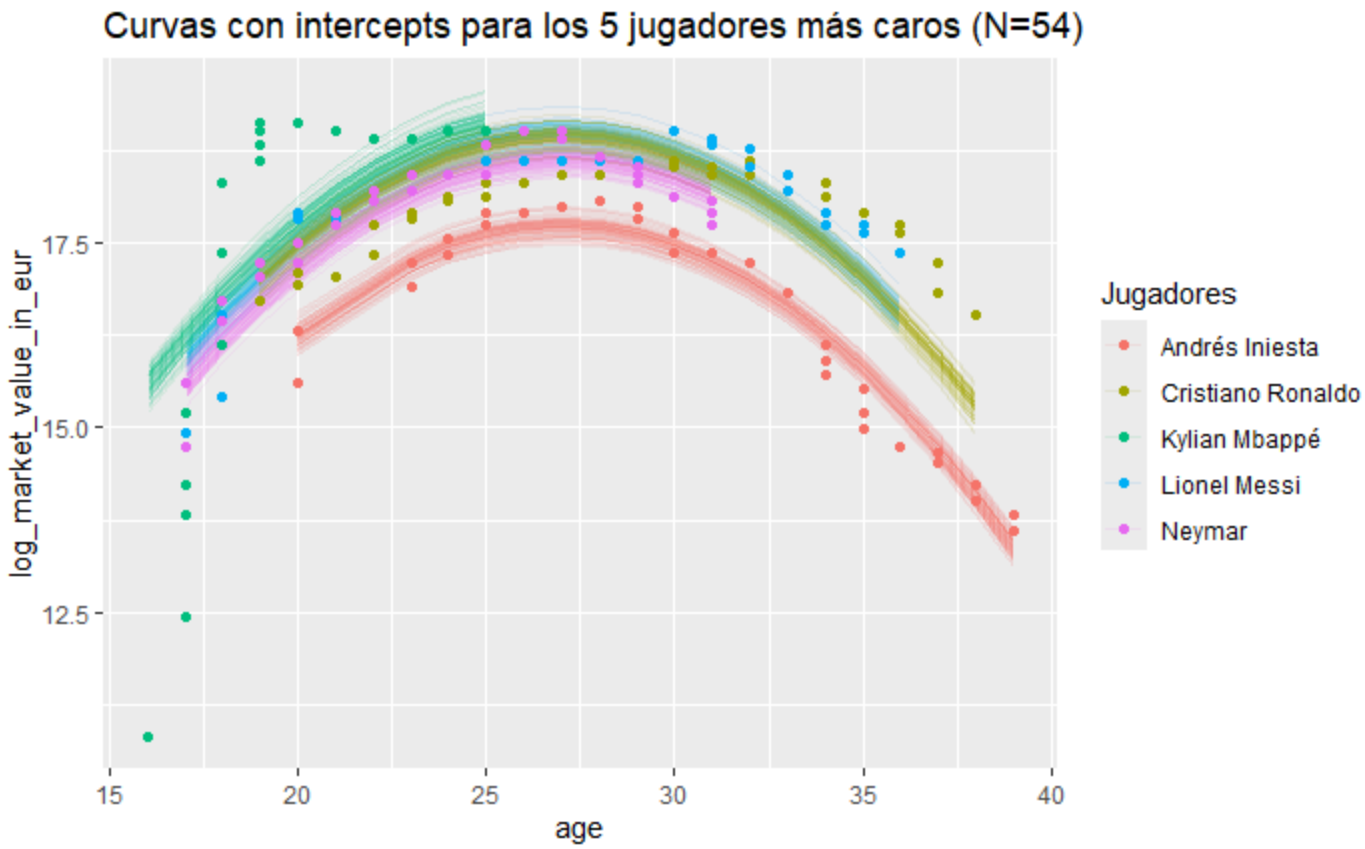
Ahora respondamos algunas preguntas utilizando el modelo

Quien es el jugador mas caro?

Como profesionales serios que somos vamos a responder a esta pregunta utilizando usando modelos estadisticos. Vamos a comparar los intercepts entre varios jugadores y luego hacer predicciones, lo bueno de nuestro modelo es que podemos extrapolar datos ya que no todos los jugadores tienen la misma edad

Hide

```
player_ids <- expensive_playes_ids$player_id

player_names <- cracks %>%
  filter(player_id %in% player_ids) %>%
  select(player_id, name) %>%
  distinct()

cracks %>%
  filter(player_id %in% player_ids) %>%
  add_linpred_draws(model_1, ndraws = 100) %>%
  ggplot(aes(x = age, y = log_market_value_in_eur)) +
  ggtitle("Curvas con intercepts para los 5 jugadores más caros (N=54)") +
  geom_line( aes(y = .linpred, group = paste(player_id, .draw), color = name), alpha = 0.1) +
  geom_point(aes(color = name)) +
  guides(color = guide_legend(title = "Jugadores")) +
  theme(legend.position = "right")
```



Hide

```
player_summaries_1 %>%
  filter(player_id %in% sapply(expensive_playes_ids, function(x) paste0("player_id:", x)))
```

player_id <chr>	player_intercept <dbl>	.lower <dbl>	.upper <dbl>
player_id:28003	-3.081597	-3.801650	-2.361795
player_id:342229	-2.816329	-3.501765	-2.127315
player_id:68290	-3.411538	-4.118220	-2.711131
player_id:7600	-4.350130	-5.075241	-3.622491
player_id:8198	-3.106805	-3.826090	-2.383596

5 rows

Luego de ver los intervalos de credibilidad del 80% en los que estan los intercepts de cada jugador podemos afirmar que:

Hay un 80% de probabilidad de que iniesta (player_id = 7600) es mas barato que el resto de jugadores ya que su intervalo esta mas abajo que el del resto y no se solapa.

Como los otros jugadores sus intervalos de credibilidad se solapan, no tenemos evidencia significativa de que alguno es mas caro que otro

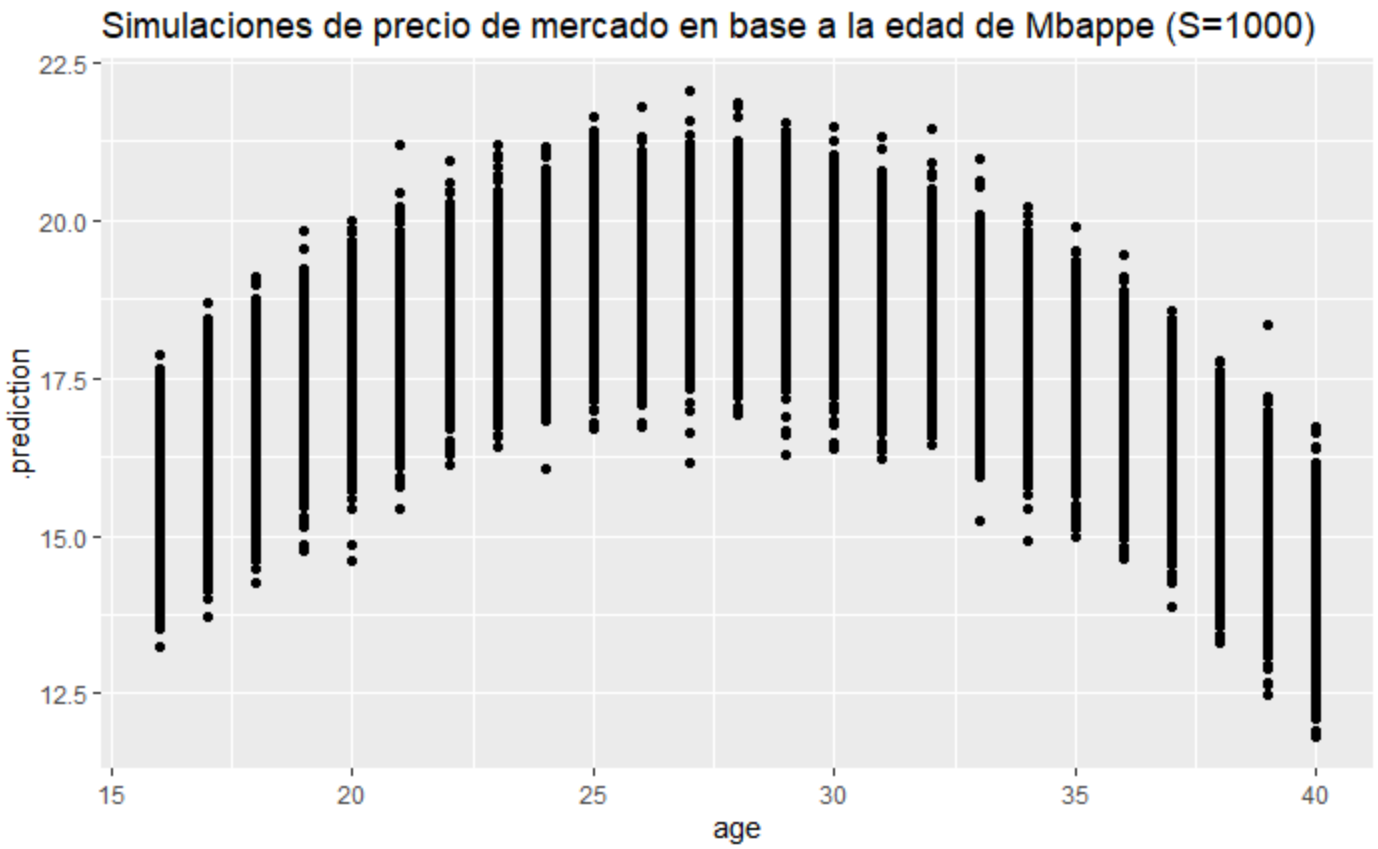
A que edad Mbappe alcanzara su precio mas alto y cual sera ese precio?

Para responder esta pregunta podemos simular posibles precios de mercado historicos para Mbappe usando la posterior y viendo a que edad alcanzara su precio maximo.

Vamos a tomar 1000 samples de la posterior y con cada uno generar un historial de precios de mercado para las edades entre 16 y 40. Luego nos vamos a quedar con la edad en el que tenga el pico de precio y vamos a hacer un histograma de las edades con picos maximos.

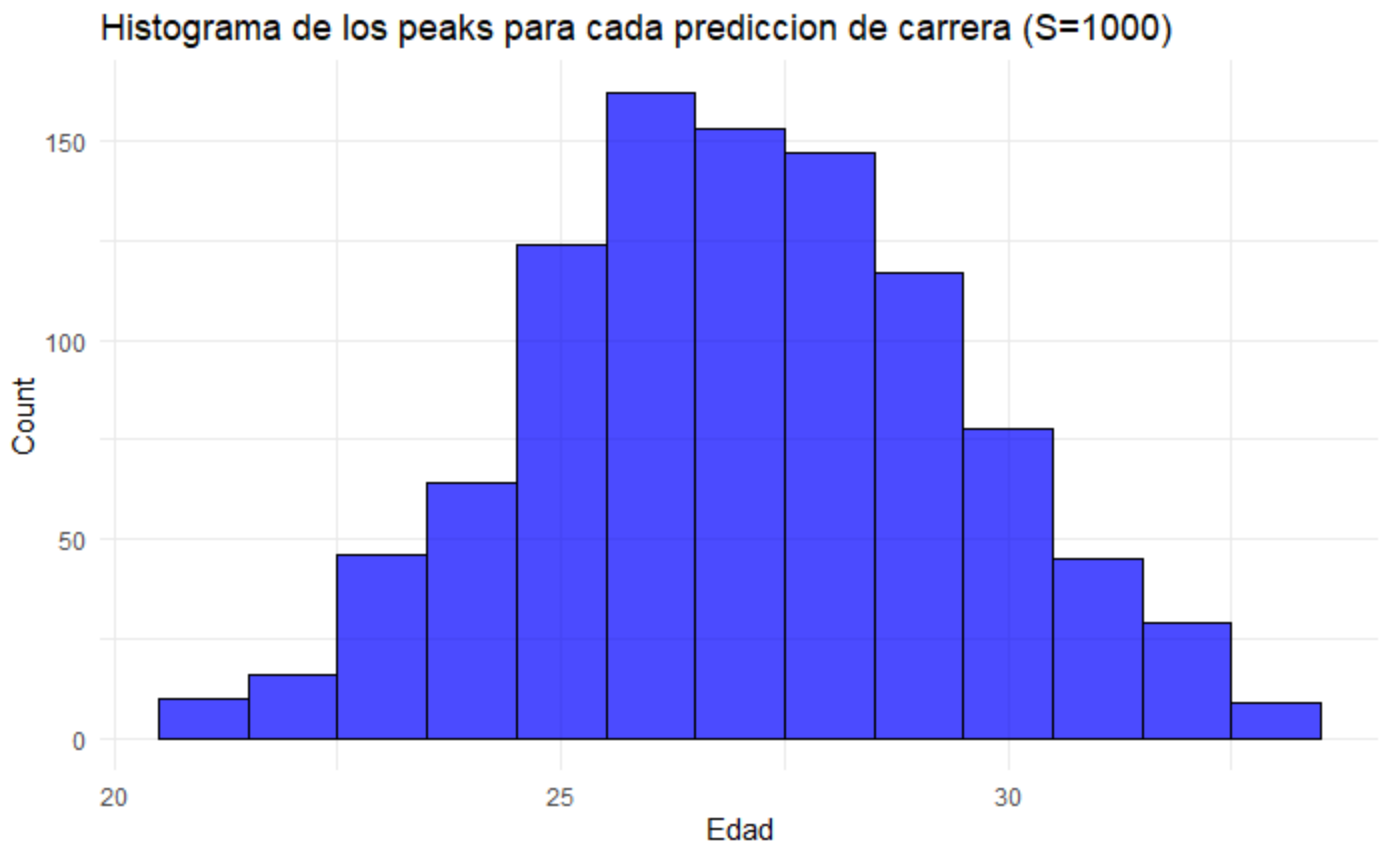
Hide

```
set.seed(314159)

mbappe_pred = cracks %>%
    filter(player_id == 342229) %>%
    group_by(age) %>%
    slice_max(order_by = market_value_in_eur, n = 1) %>%
    distinct() %>%
    select(-market_value_in_eur, -log_market_value_in_eur) %>%
    bind_rows(
        data.frame(
          player_id = rep(342229, 15),
          name = rep("Kylian Mbappé", 15),
          age = 26:40,
          age_2 = (26:40)^2
        ) %>%
      mutate(player_id = factor(player_id))
    ) %>%
  add_predicted_draws(model_1, ndraws = 1000)

mbappe_pred %>%
  ggplot(aes(x = age, y = .prediction)) +
  ggtitle("Simulaciones de precio de mercado en base a la edad de Mbappe (S=1000)") +
  geom_point(aes(y = .prediction, group = .draw))
```



Hide

```
mbappe_pred %>%
  group_by(.draw) %>%               # Group by the .draw column
  slice_max(order_by = .prediction, n = 1) %>%
    ggplot(aes(x = age)) +         # Map 'age' column to the x-axis
    geom_histogram(binwidth = 1,     # Control the width of the bins
                   fill = "blue",    # Set the color of the bars
                   color = "black",  # Set the color of the borders of the bars
                   alpha = 0.7) +    # Make the bars semi-transparent
    labs(title = "Histograma de los peaks para cada prediccion de carrera (S=1000)", x = "Edad",
y = "Count") + # Add title and axis labels
    theme_minimal()
```



Hide

```
(mbappe_pred %>%
  group_by(.draw) %>%
  slice_max(order_by = .prediction, n = 1) %>%
  filter(age == 26) %>%
  nrow()) /
  (mbappe_pred %>%
  group_by(.draw) %>%
  slice_max(order_by = .prediction, n = 1) %>%
  nrow())
```

```
[1] 0.162
```

La probabilidad de que haga peak entre los 26 es de 16%

Hide

```
(mbappe_pred %>%
  group_by(.draw) %>%
  slice_max(order_by = .prediction, n = 1) %>%
  filter(age %in% c(26, 27, 28)) %>%
  nrow()) /
  (mbappe_pred %>%
  group_by(.draw) %>%
  slice_max(order_by = .prediction, n = 1) %>%
  nrow())
```

```
[1] 0.462
```

La probabilidad de que haga peak entre los 26 y los 28 es de 46%

Hide

```
(mbappe_pred %>%
  group_by(.draw) %>%
  slice_max(order_by = .prediction, n = 1) %>%
  filter(age %in% 22:33) %>%
  nrow()) /
  (mbappe_pred %>%
  group_by(.draw) %>%
  slice_max(order_by = .prediction, n = 1) %>%
  nrow())
```

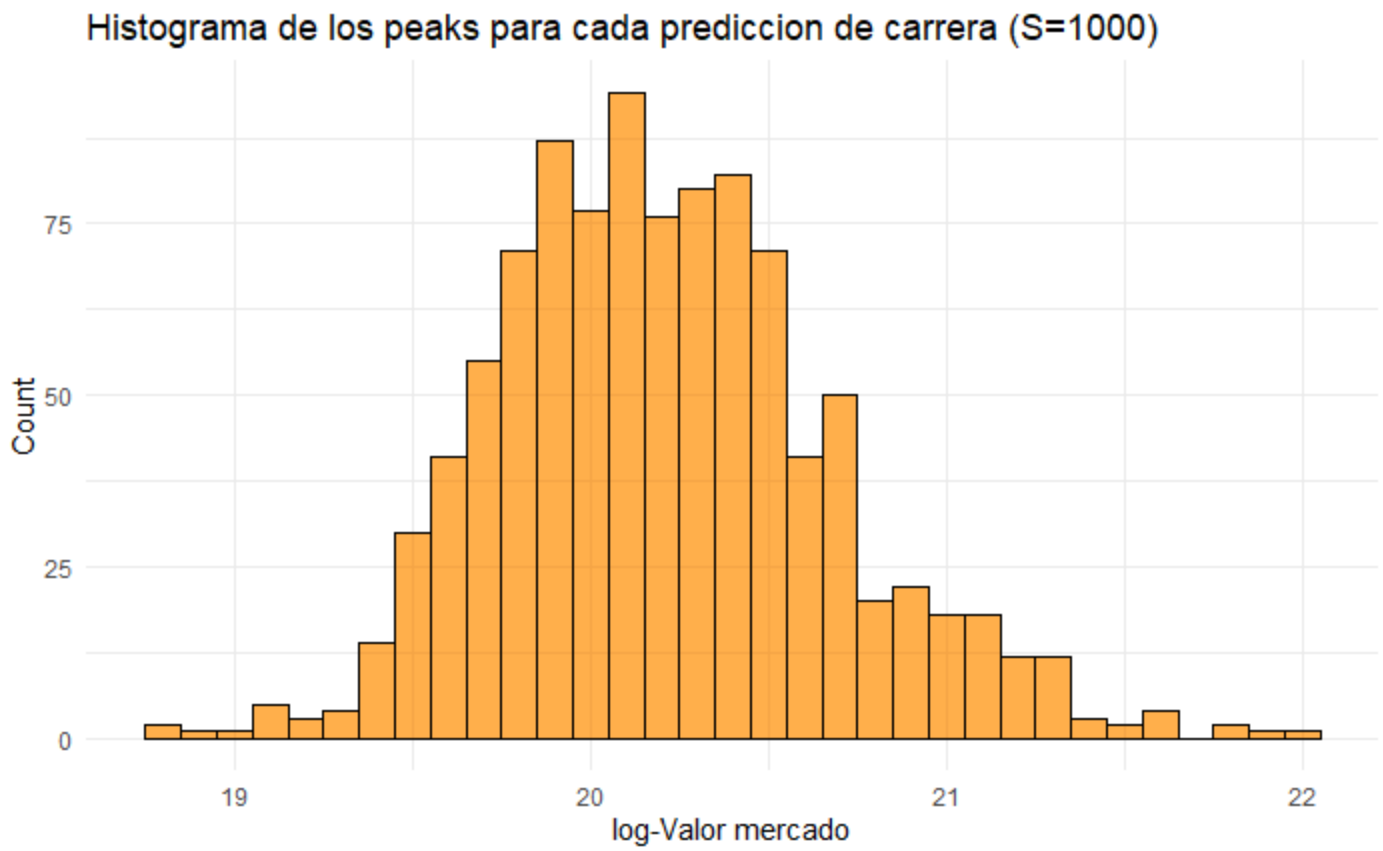
```
[1] 0.99
```

La probabilidad de que haga peak entre los 22 y los 33 es de 99%

Luego vamos a hacer exactamente lo mismo pero en vez de quedarnos con la edad del pico maximo, nos vamos a quedar con el valor maximo en si.

Hide

```
mbappe_pred %>%
  group_by(.draw) %>%               # Group by the .draw column
  slice_max(order_by = .prediction, n = 1) %>%
    ggplot(aes(x = .prediction)) +         # Map 'age' column to the x-axis
    geom_histogram(binwidth = 0.1,     # Control the width of the bins
                   fill = "darkorange",    # Set the color of the bars
                   color = "black",  # Set the color of the borders of the bars
                   alpha = 0.7) +    # Make the bars semi-transparent
    labs(title = "Histograma de los peaks para cada prediccion de carrera (S=1000)", x = "log-Va
lor mercado", y = "Count") + # Add title and axis labels
    theme_minimal()
```



Hide

```
(mbappe_pred %>%
  group_by(.draw) %>%
  slice_max(order_by = .prediction, n = 1) %>%
  filter(.prediction > 19 & .prediction < 20.5) %>%
  nrow()) /
  (mbappe_pred %>%
  group_by(.draw) %>%
  slice_max(order_by = .prediction, n = 1) %>%
  nrow())
```

```
[1] 0.759
```

Hay un 75% de probabilidad de que en su peak, Mbappe va a cotizar entre 19 y 20.5 log-EUR que transformandolos serian entre 178.482.301 y 799.902.177 EUR

Conclusiones

En este TP pudimos usar modelos jerarquicos bayesianos para modelar la relacion entre la edad y el precio de mercado de varios jugadores viendo las diferencias entre los mismos y su evolucion individual.

El enfoque bayesiano tiene como ventaja que podemos realizar simulaciones para respondernos preguntas del modelo de manera simple sampleando de la posterior y realizando simulaciones con los parametros obtenidos, esto nos permite tener en cuenta la incerteza de los valores de los parametros a la hora de sacar conclusiones. A su vez tambien es flexible permitiendo modelar nuestro conocimiento de dominio del ploblema y nuestra confianza o restricciones al comporamiento del modelo mediante los priors.

Futuros pasos

Como bien vimos en el informe, el modelo propuesto no permite que cada jugador tenga su propia curva de progreso de precios, con lo cual hace que todos tengan la misma tendencia, esto se podria solucionar haciendo que cada jugador tenga sus coeficiente β_0 y β_1

Tambien se podria agregar una capa mas al modelo haciendo una agrupacion de los datos por liga ademas de por jugador.