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

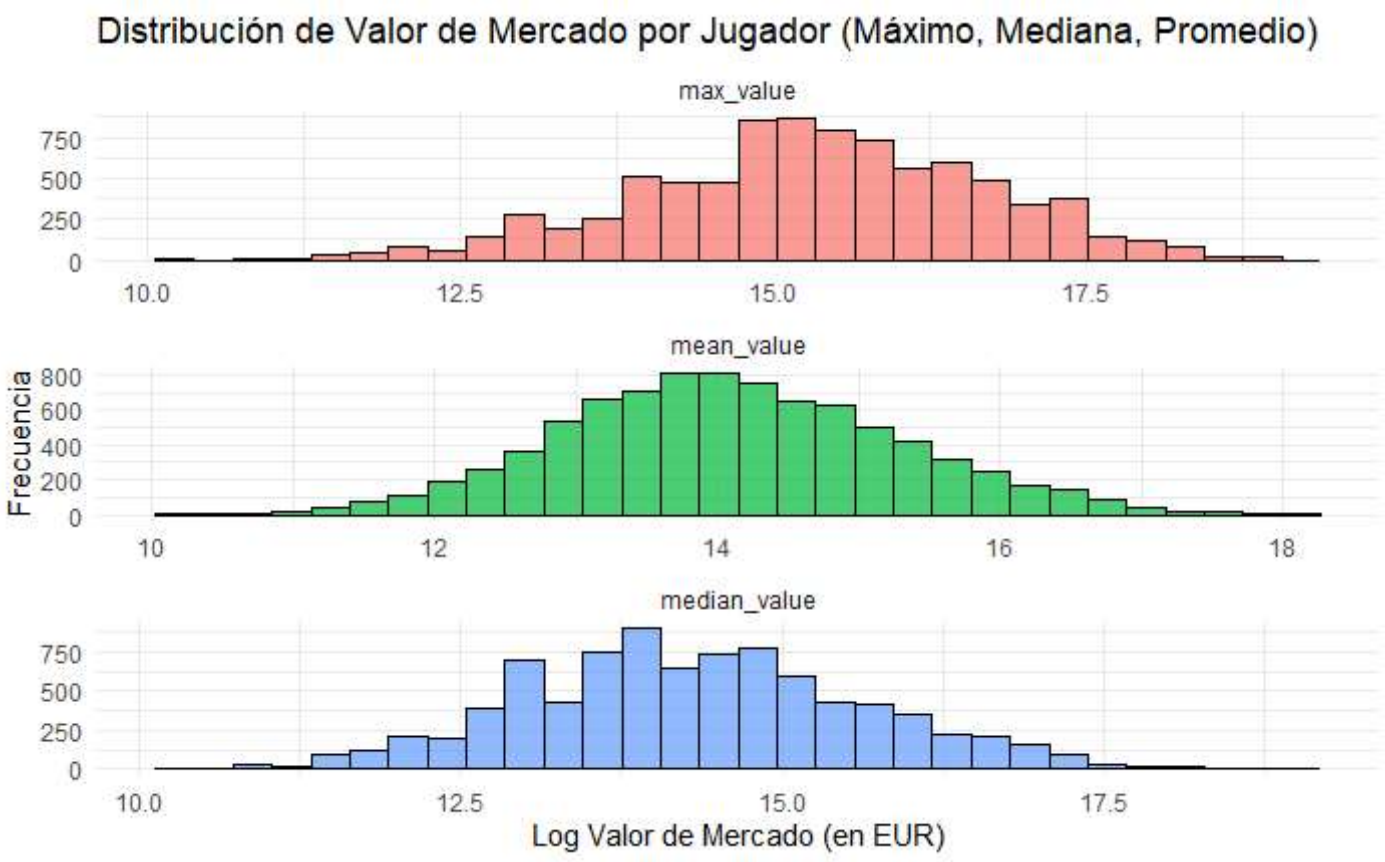
6 rows

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

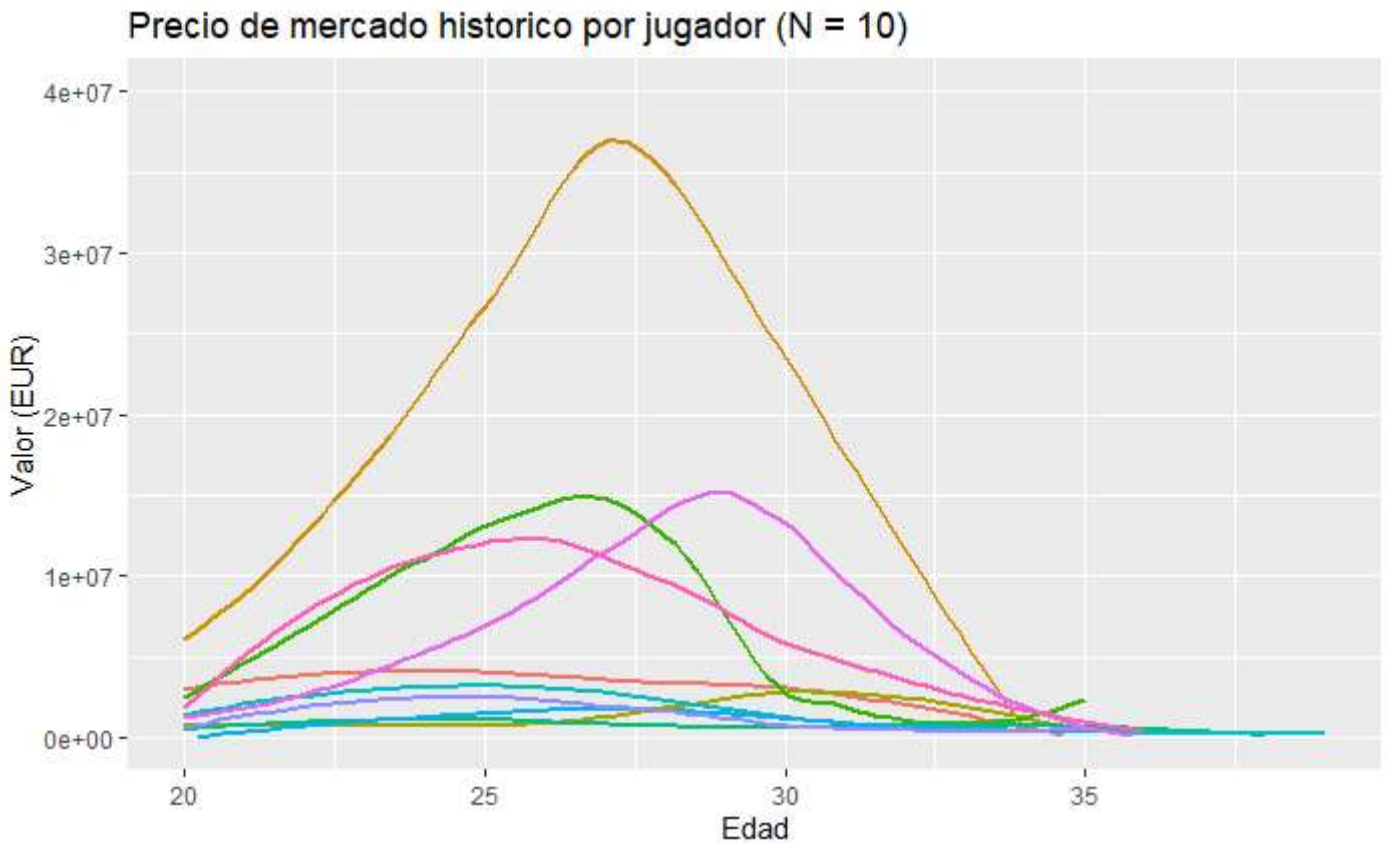
Show



Esto nos da un buen insight de que el logaritmo del valor de cada jugador se distribuye de una forma normal.

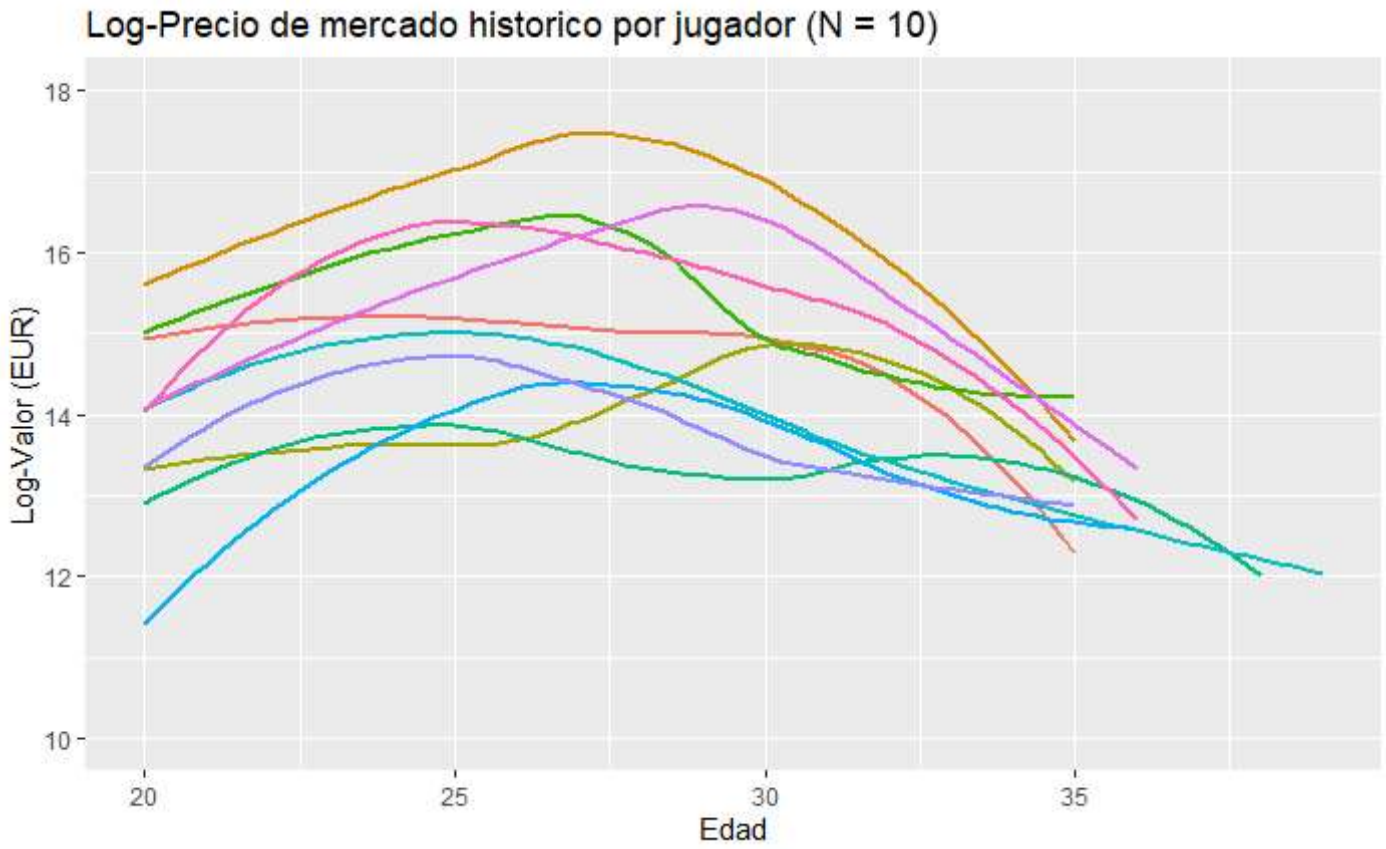
Show

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

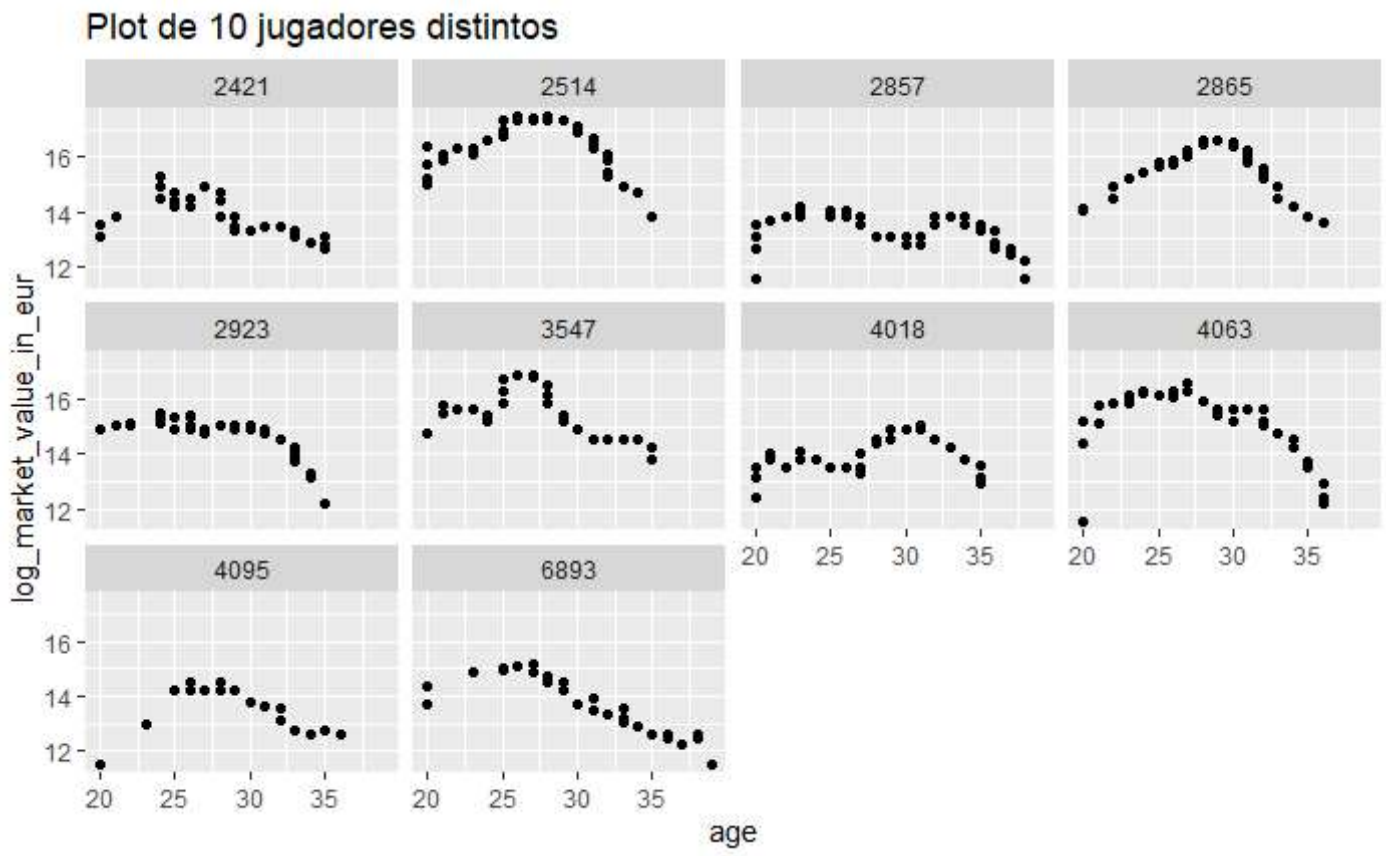


Show

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



Show

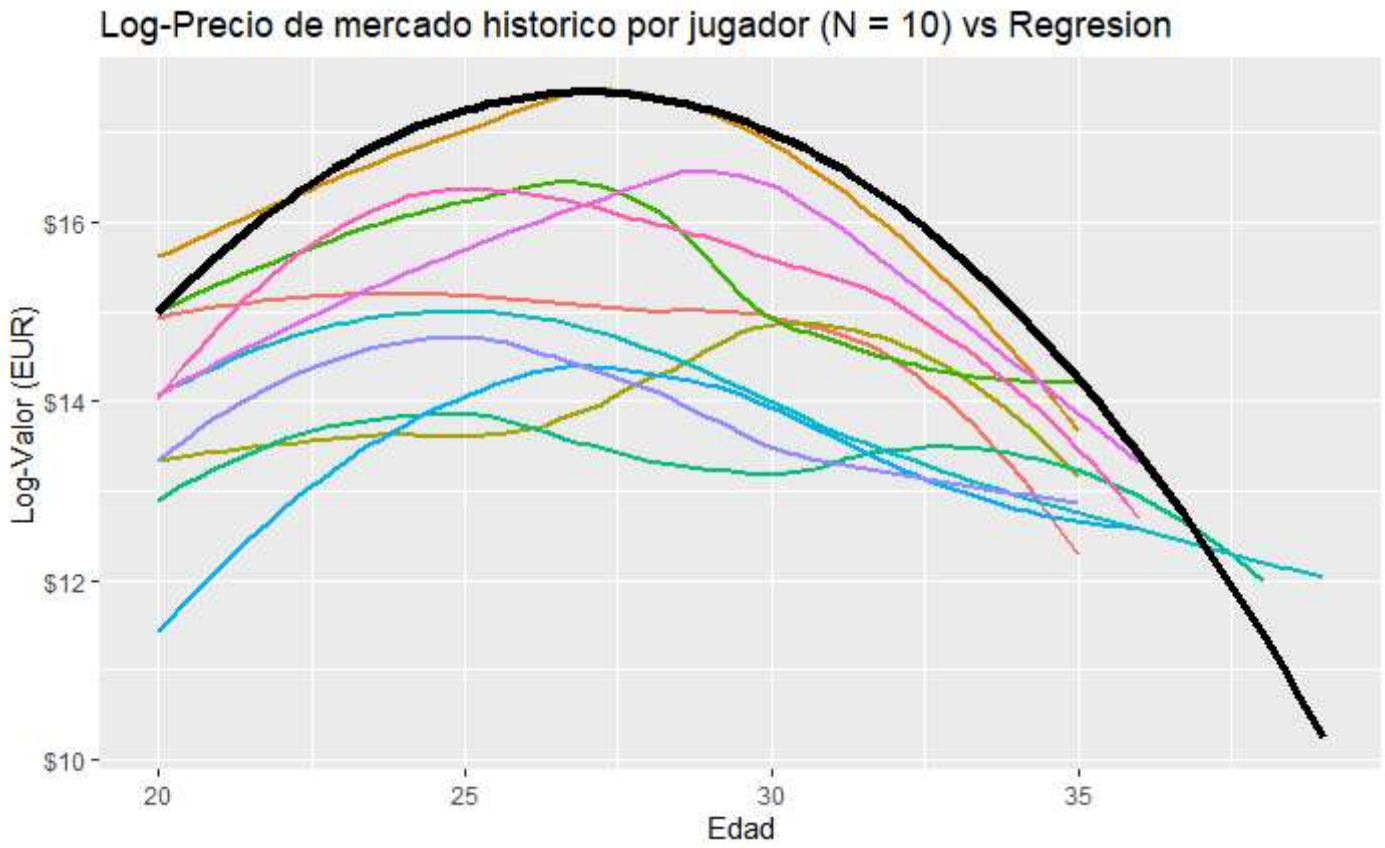


Modelado

Dada la forma de los datos pensamos es poder predecir el logaritmo del precio de mercado de un jugador en base solamente 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.

Show

```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
Please use `linewidth` instead.
```



Vemos la funcion cuadratica que vemos que puede llegar a aproximar bien a los datos
$y = -19 + 2.7x - 0.05x^2$

Definicion:

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

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

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

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

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

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

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

Show

```
SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.000703 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 7.03 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: 56.6 seconds (Warm-up)
Chain 1:                27.222 seconds (Sampling)
Chain 1:                83.822 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 0.000104 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 1.04 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: 69.985 seconds (Warm-up)
Chain 2:                26.884 seconds (Sampling)
Chain 2:                96.869 seconds (Total)
Chain 2:

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

```
Chain 3:
Chain 3: Gradient evaluation took 0.000115 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 1.15 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: 59.747 seconds (Warm-up)
Chain 3:                27.677 seconds (Sampling)
Chain 3:                87.424 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 9.8e-05 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.98 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: 60.84 seconds (Warm-up)
Chain 4:                26.773 seconds (Sampling)
Chain 4:                87.613 seconds (Total)
Chain 4:
```

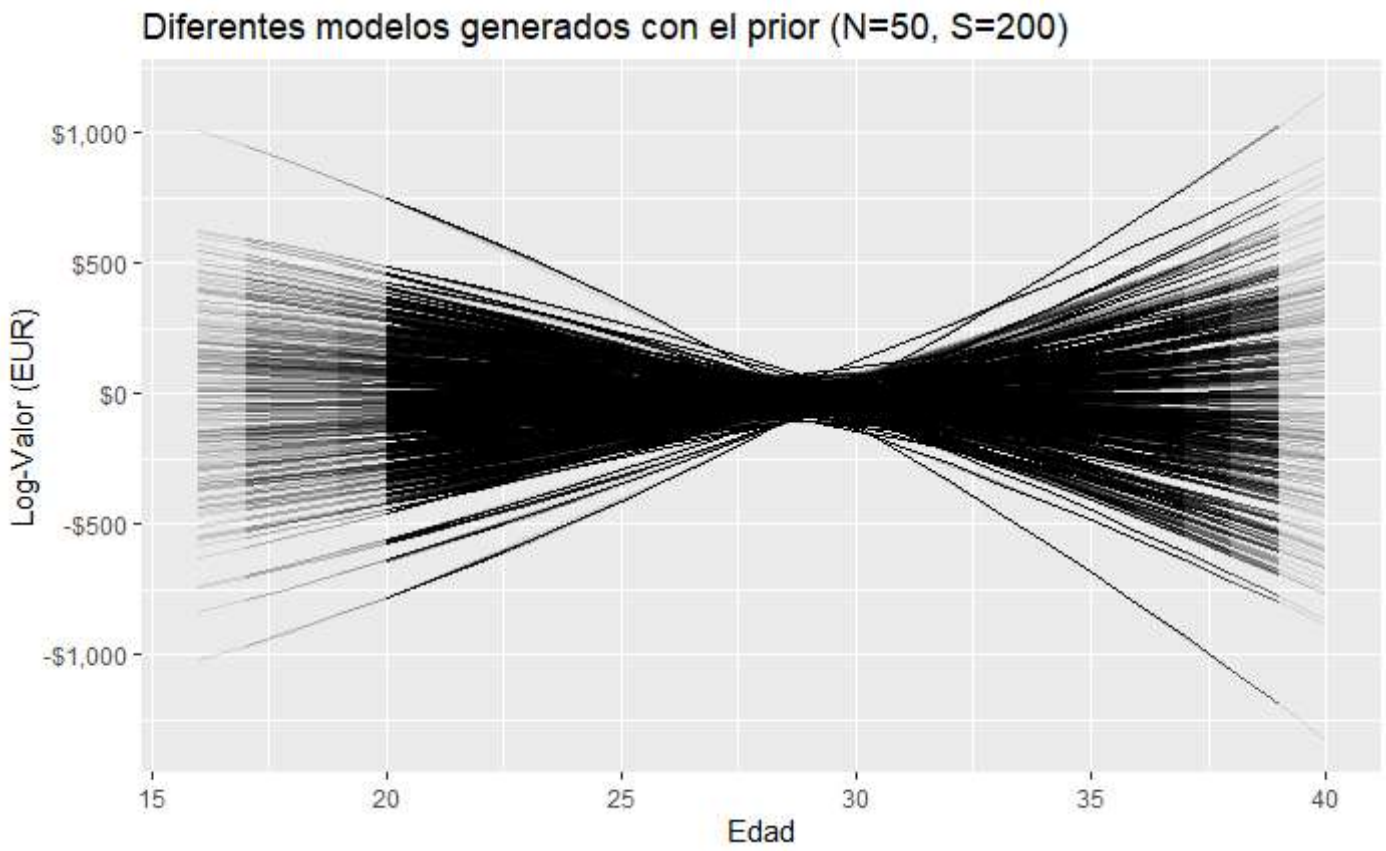
Vamos a ver si los priors que definimos generar modelos y datos coherentes con lo que esperamos

Show

```
SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 2e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.2 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: 0.895 seconds (Warm-up)
Chain 1:                0.847 seconds (Sampling)
Chain 1:                1.742 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 2.1e-05 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.21 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: 0.848 seconds (Warm-up)
Chain 2:                0.835 seconds (Sampling)
Chain 2:                1.683 seconds (Total)
Chain 2:

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

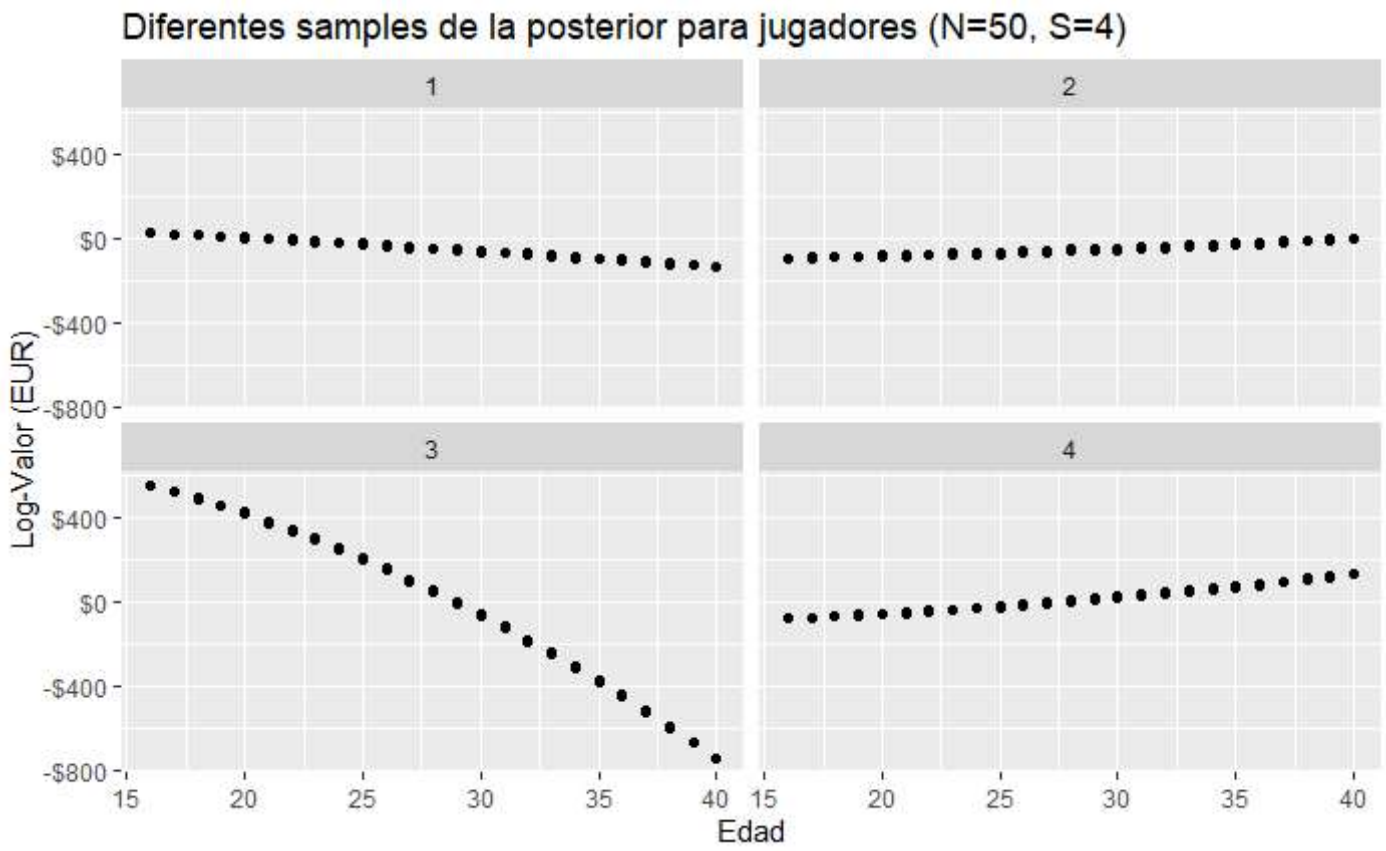
```
Chain 3:
Chain 3: Gradient evaluation took 1.9e-05 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.19 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:    1 / 10000 [  0%]  (Warmup)
Chain 3: Iteration: 1000 / 10000 [ 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: 0.826 seconds (Warm-up)
Chain 3:                0.861 seconds (Sampling)
Chain 3:                1.687 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 2e-05 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.2 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: 0.808 seconds (Warm-up)
Chain 4:                0.826 seconds (Sampling)
Chain 4:                1.634 seconds (Total)
Chain 4:
```

Show



Show

```
Warning:
In add_predicted_draws(): The `n` argument is a deprecated alias for `ndraws`.
Use the `ndraws` argument instead.
See help("tidybayes-deprecated").
```



Ahora vamos a ver los resultados de nuestro modelo

Show

```
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
```

Show

term <chr>	estimate <dbl>	std.error <dbl>	conf.low <dbl>	conf.high <dbl>
(Intercept)	-6.5533460	0.5832947216	-7.30638418	-5.80663058
age	1.6336574	0.0393001097	1.58274109	1.68434716
age_2	-0.0302452	0.0006840102	-0.03112876	-0.02935047

3 rows

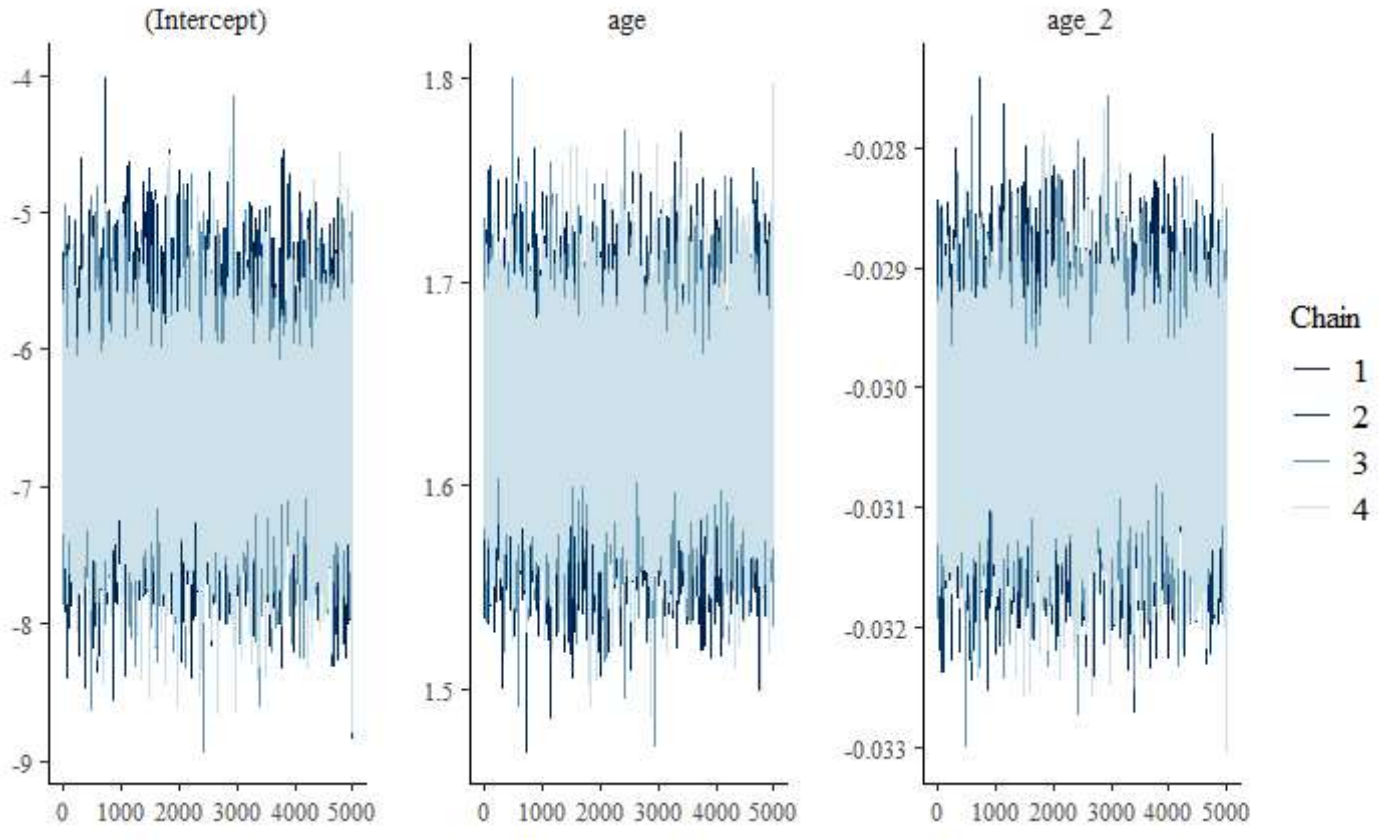
Vemos que con una confianza del 80% podemos ver que: el intercept promedio esta entre -7.30 y -5.80 el coeficiente ‘b’ lineal de la regresion este entre 1.58 y 1.68 el coeficiente ‘a’ cuadratico de la regresion este entre -0.03 y -0.029

estos valores son razonables ya que son cercanos a la realidad que vemos en los datos.

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.

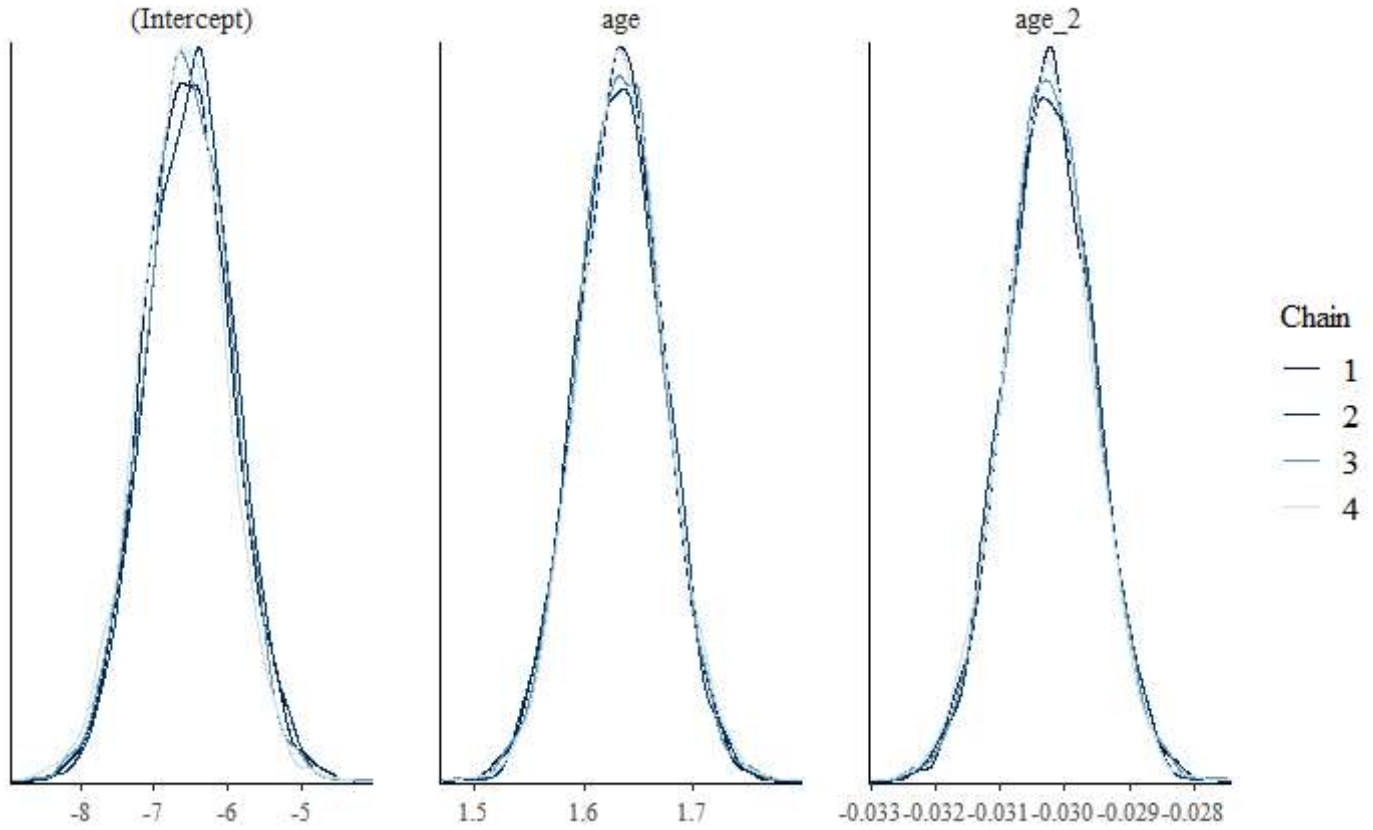
Hide

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



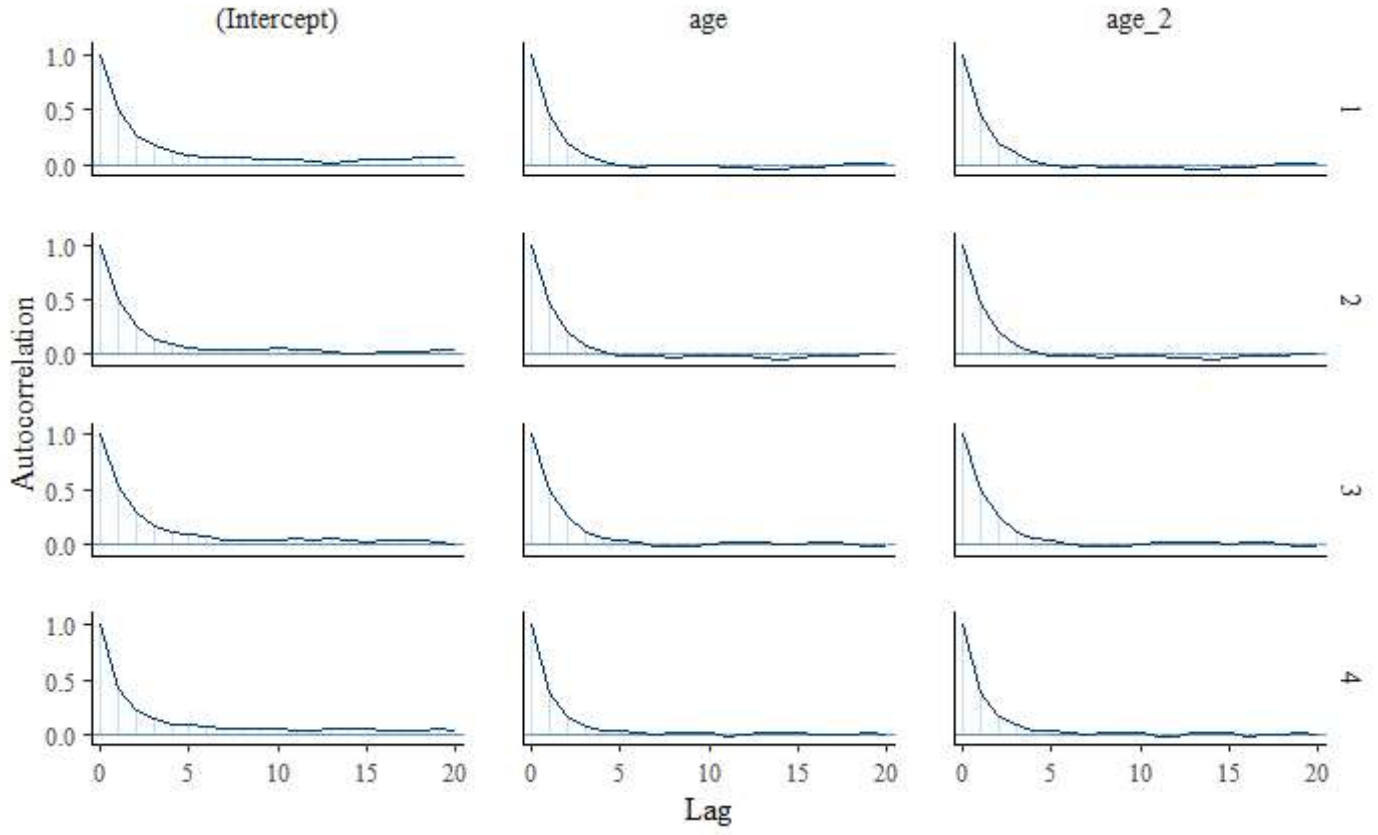
Hide

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



Hide

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



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
```

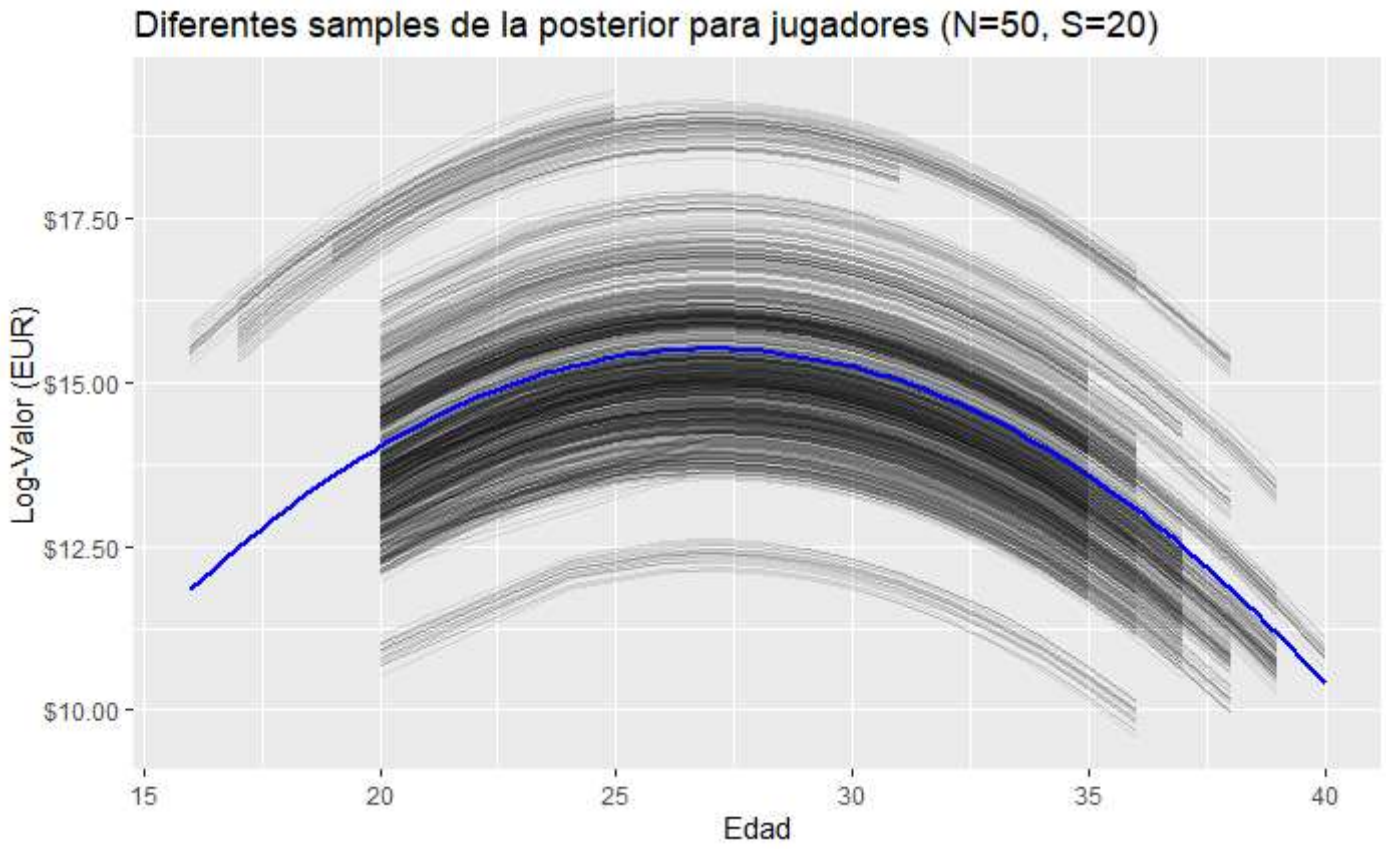
Hide

```
rhat(model_1)
```

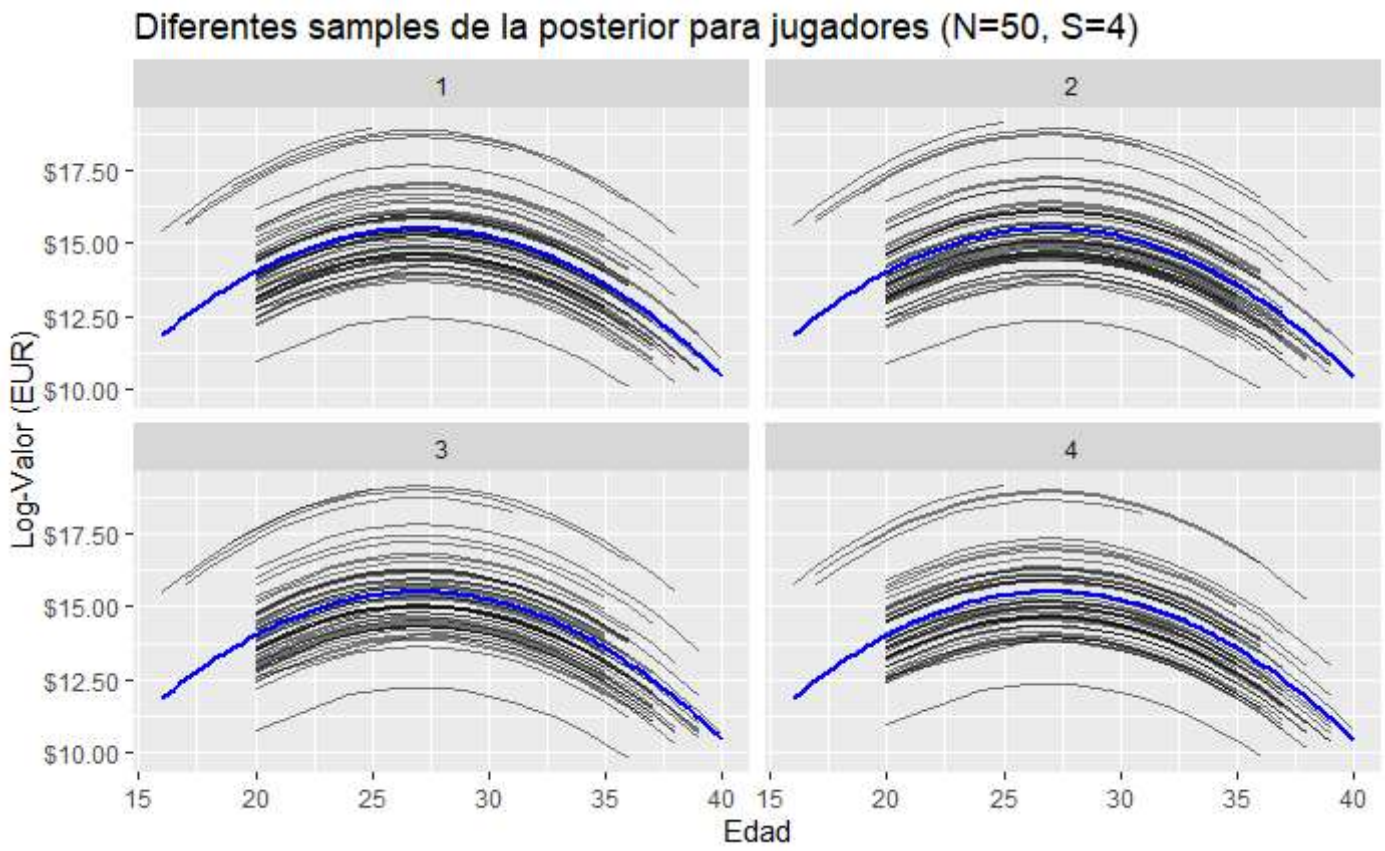
(Intercept)	age
1.0033054	1.0006558
age_2	b[(Intercept) player_id:10003]
1.0006348	1.0138140
b[(Intercept) player_id:10255]	b[(Intercept) player_id:10329]
1.0152135	1.0129196
b[(Intercept) player_id:12124]	b[(Intercept) player_id:12704]
1.0157299	1.0153341
b[(Intercept) player_id:13217]	b[(Intercept) 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
```

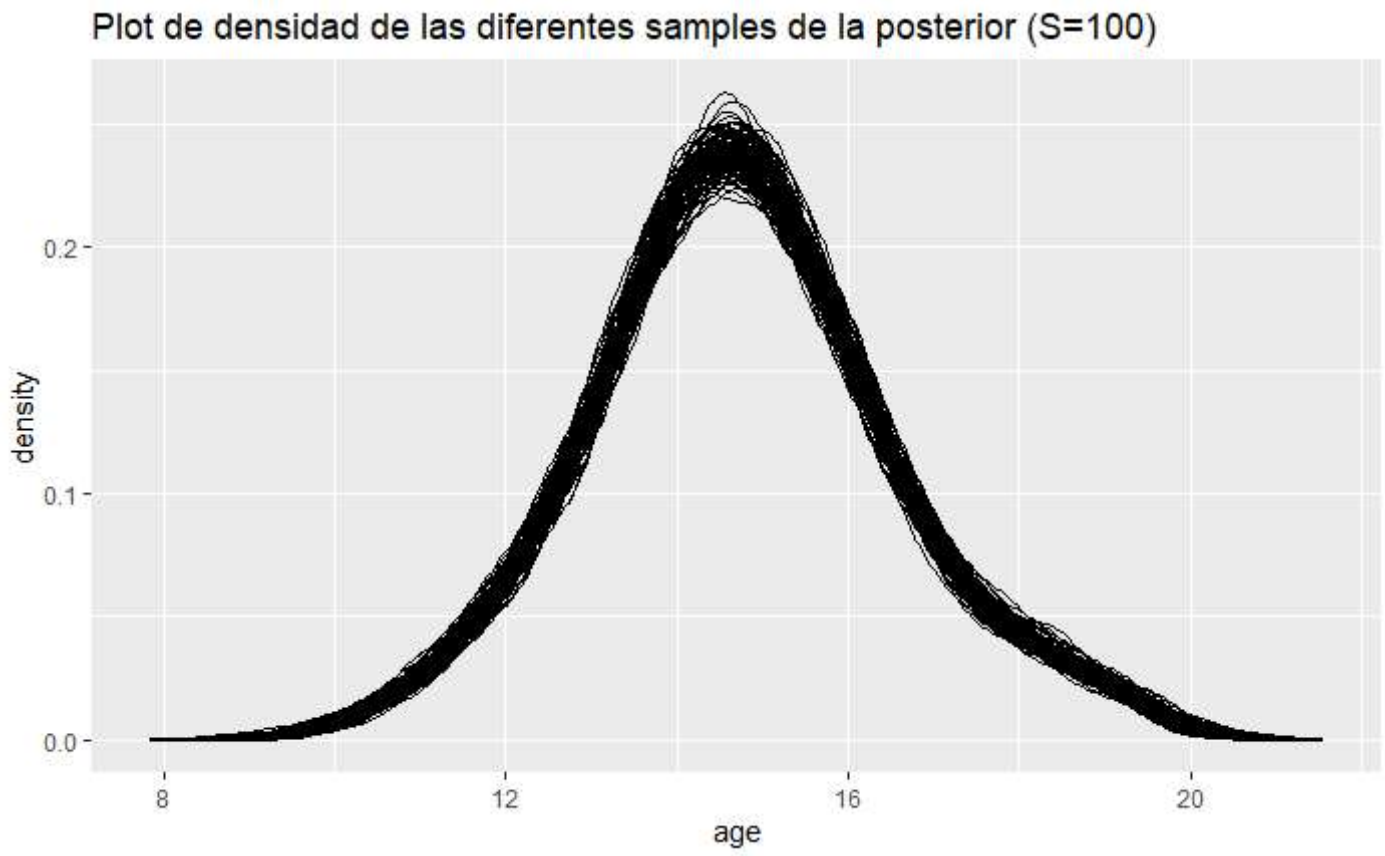
Show



Show

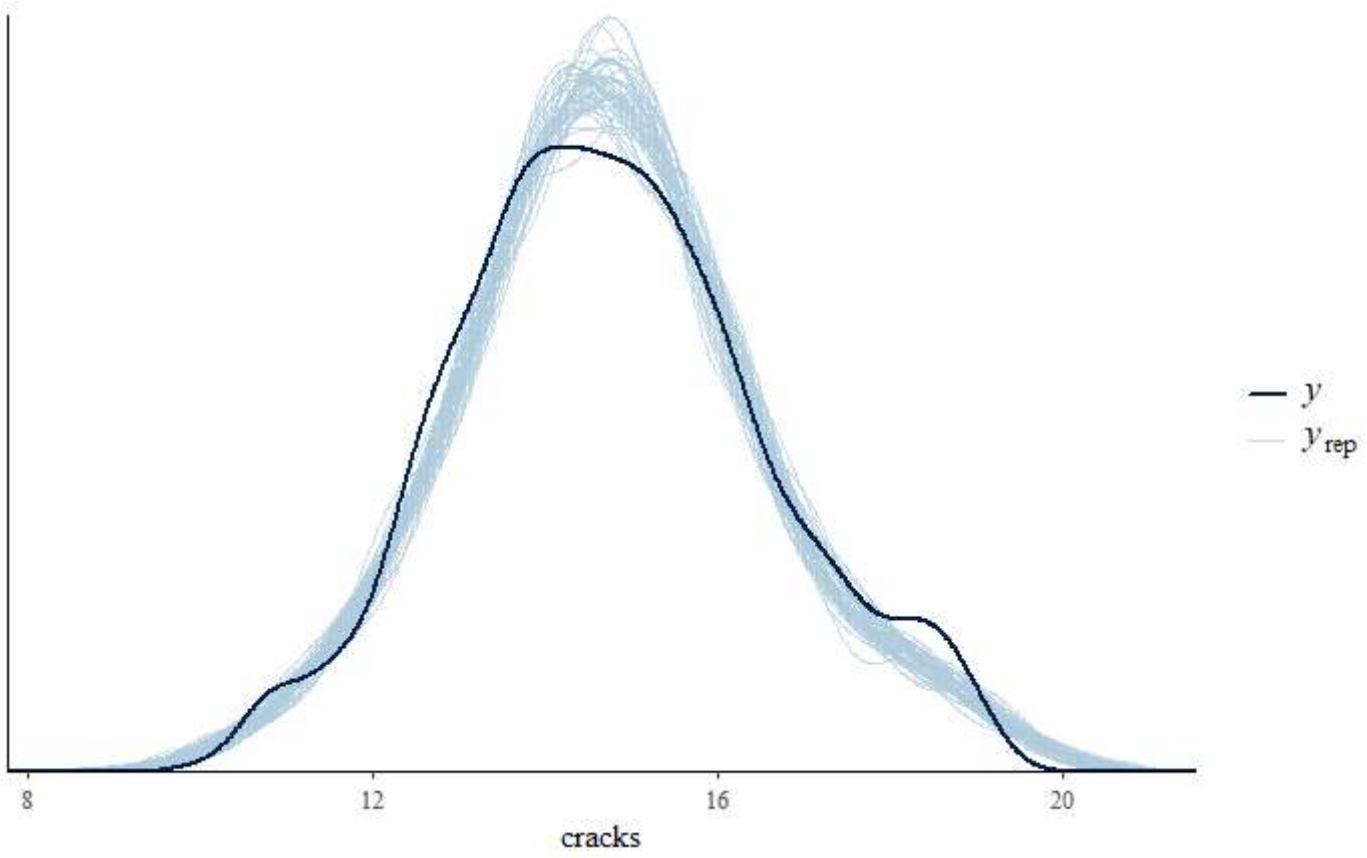

Diferentes samples de la posterior para jugadores (N=50, S=4)
1
2
3
4
Log-Valor (EUR)
$17.50
$15.00
$12.50
$10.00
15
20
25
30
35
40
Edad

Show


Plot de densidad de las diferentes samples de la posterior (S=100)
density
0.2
0.1
0.0
8
12
16
20
age

Hide

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



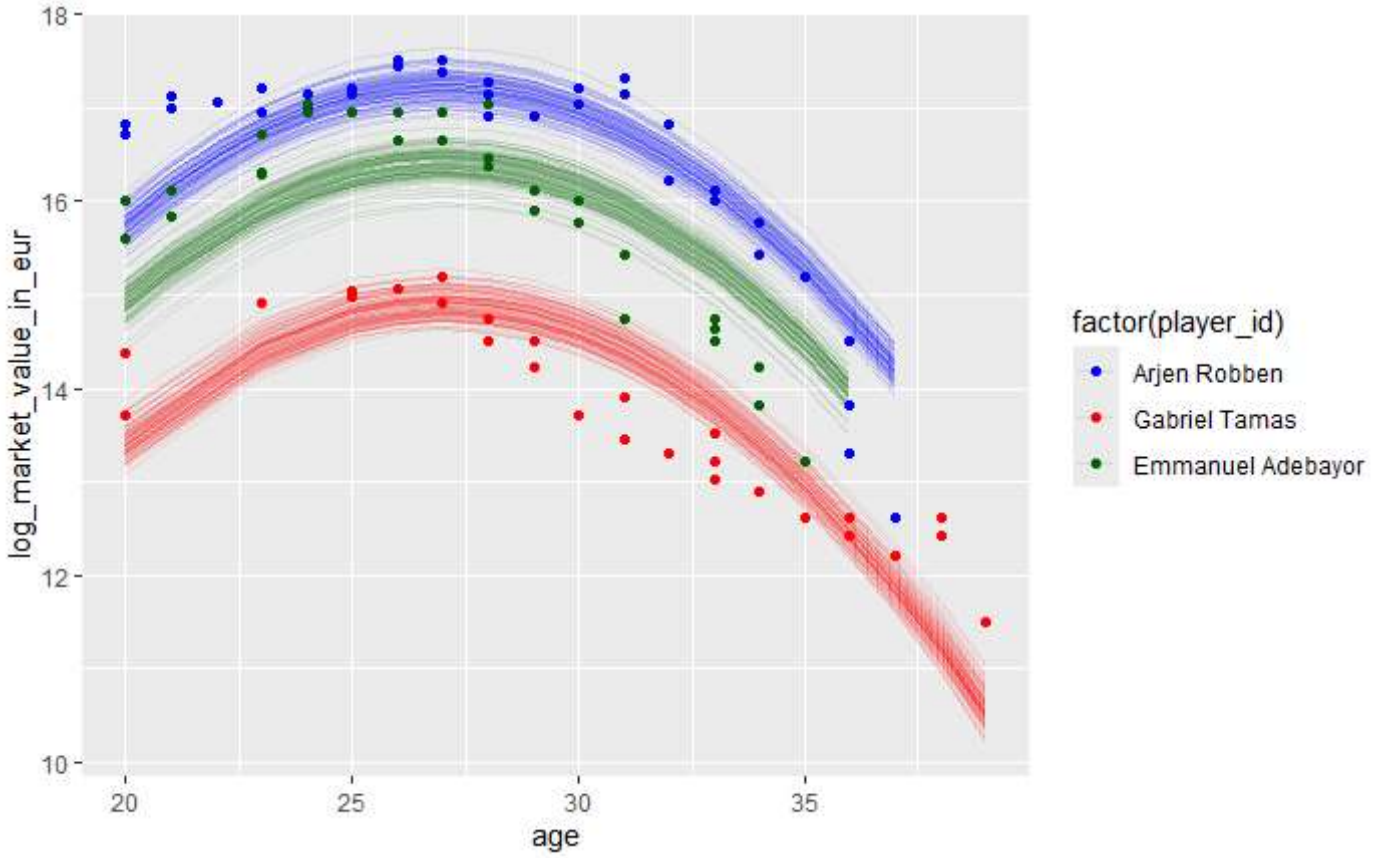
Veamos los distintos intercepts para algunos jugadores

Show

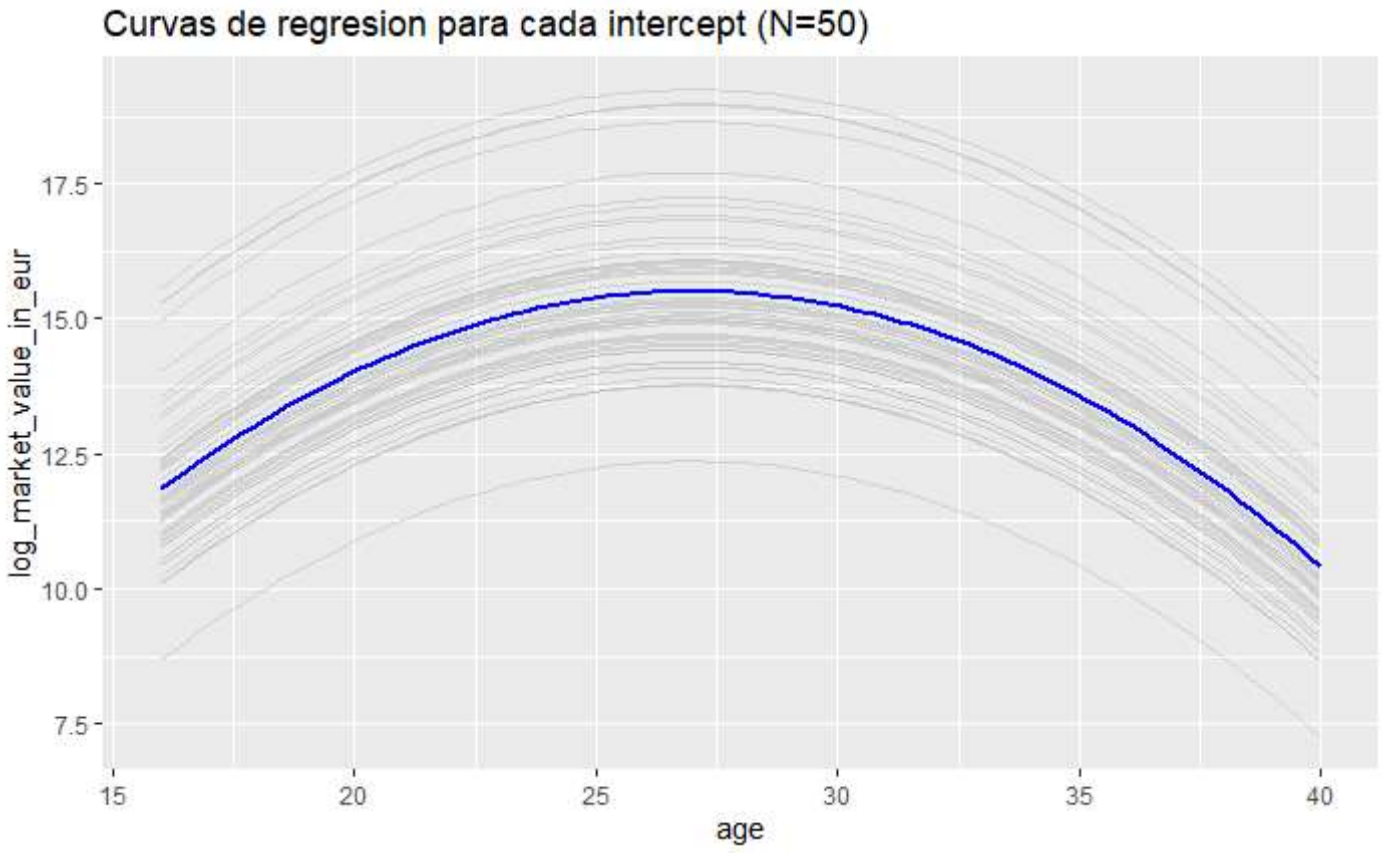
player_id <chr>	player_intercept <dbl>	.lower <dbl>	.upper <dbl>
player_id:4360	-4.819589	-5.547573	-4.091937
player_id:6893	-7.158221	-7.890850	-6.429816
player_id:7767	-4.964635	-5.705237	-4.219584

3 rows

Show

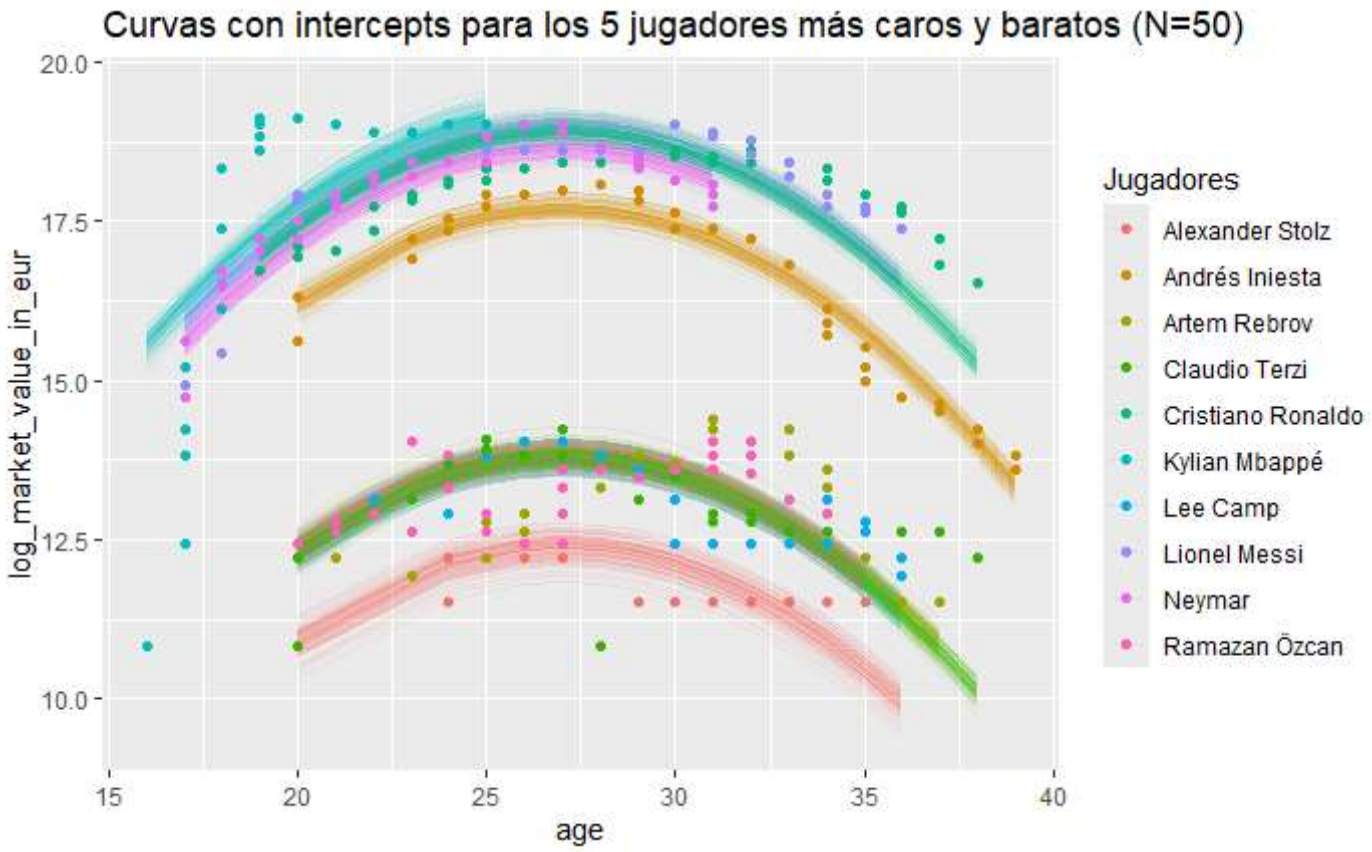


Show



Show

Show



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 hay mas variabilidad entre el precio de los jugadores que las fluctuaciones para un jugador individual

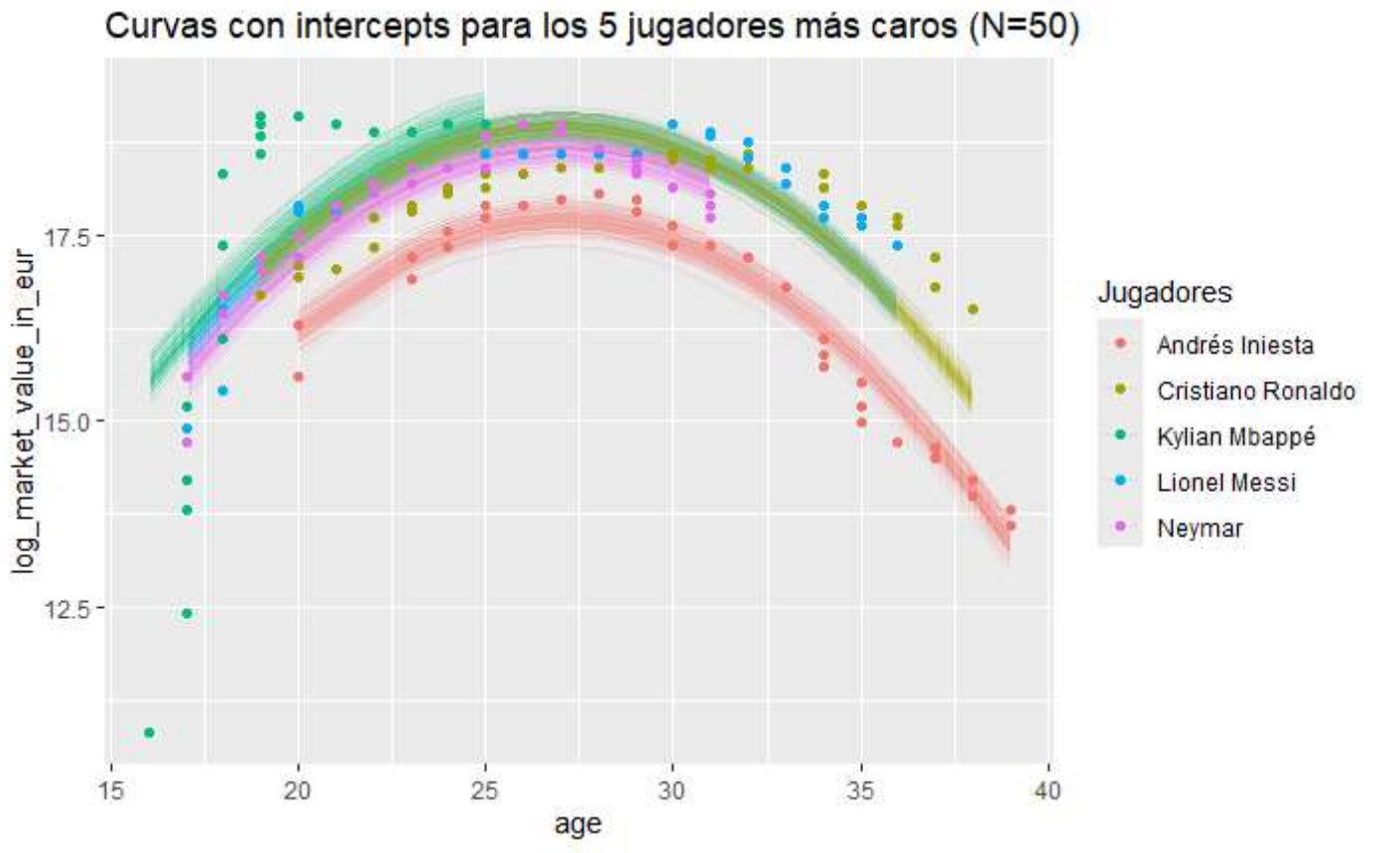
Posterior Predictions

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

Show



Show

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

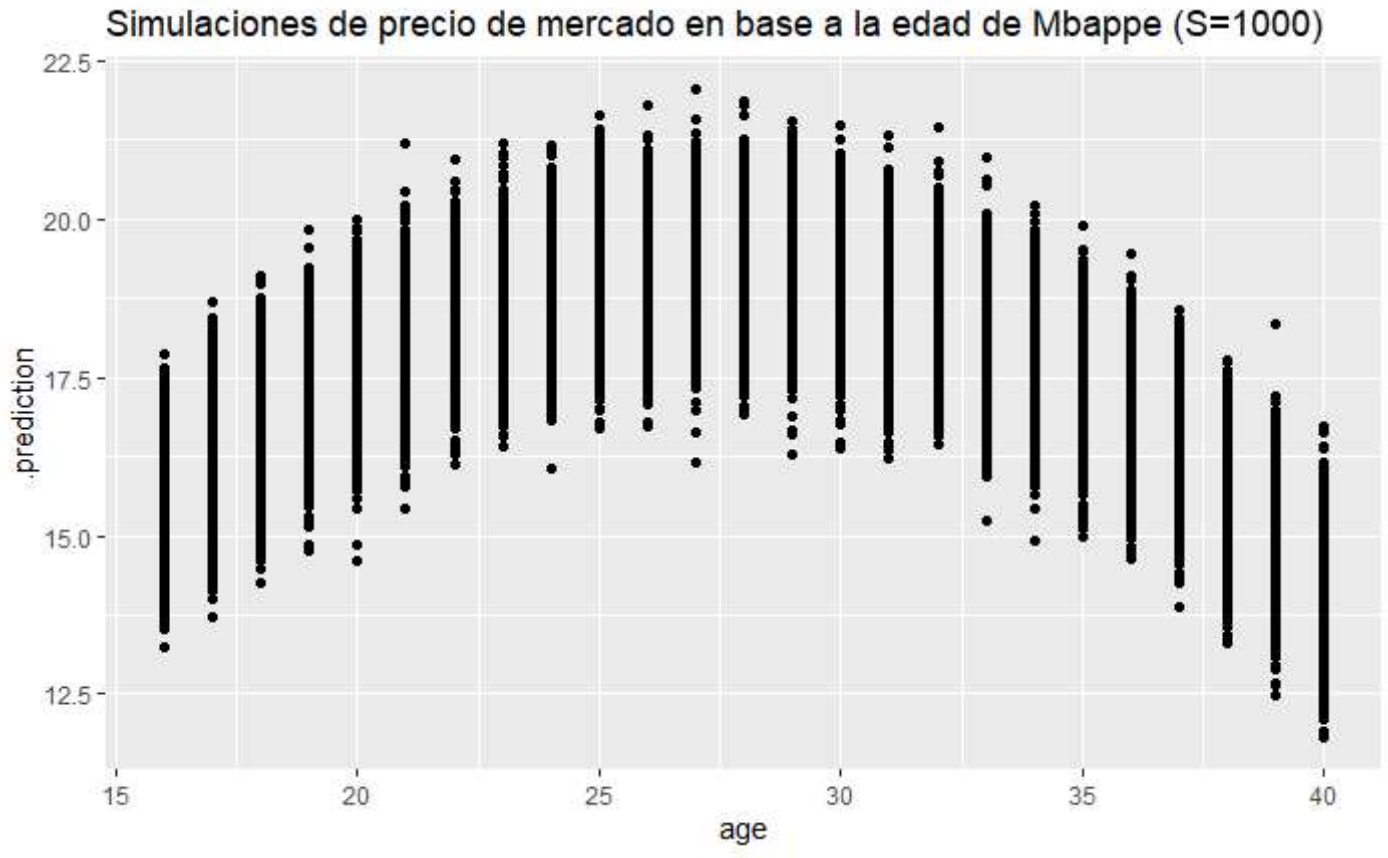
5 rows

- Podemos estar seguros con nivel de confianza alto de que iniesta (7600) es mas barato que el resto de jugadores.
- No podemos estar seguros con buen nivel de confianza que Mbappe(342229) es mas caro que algun otro jugador

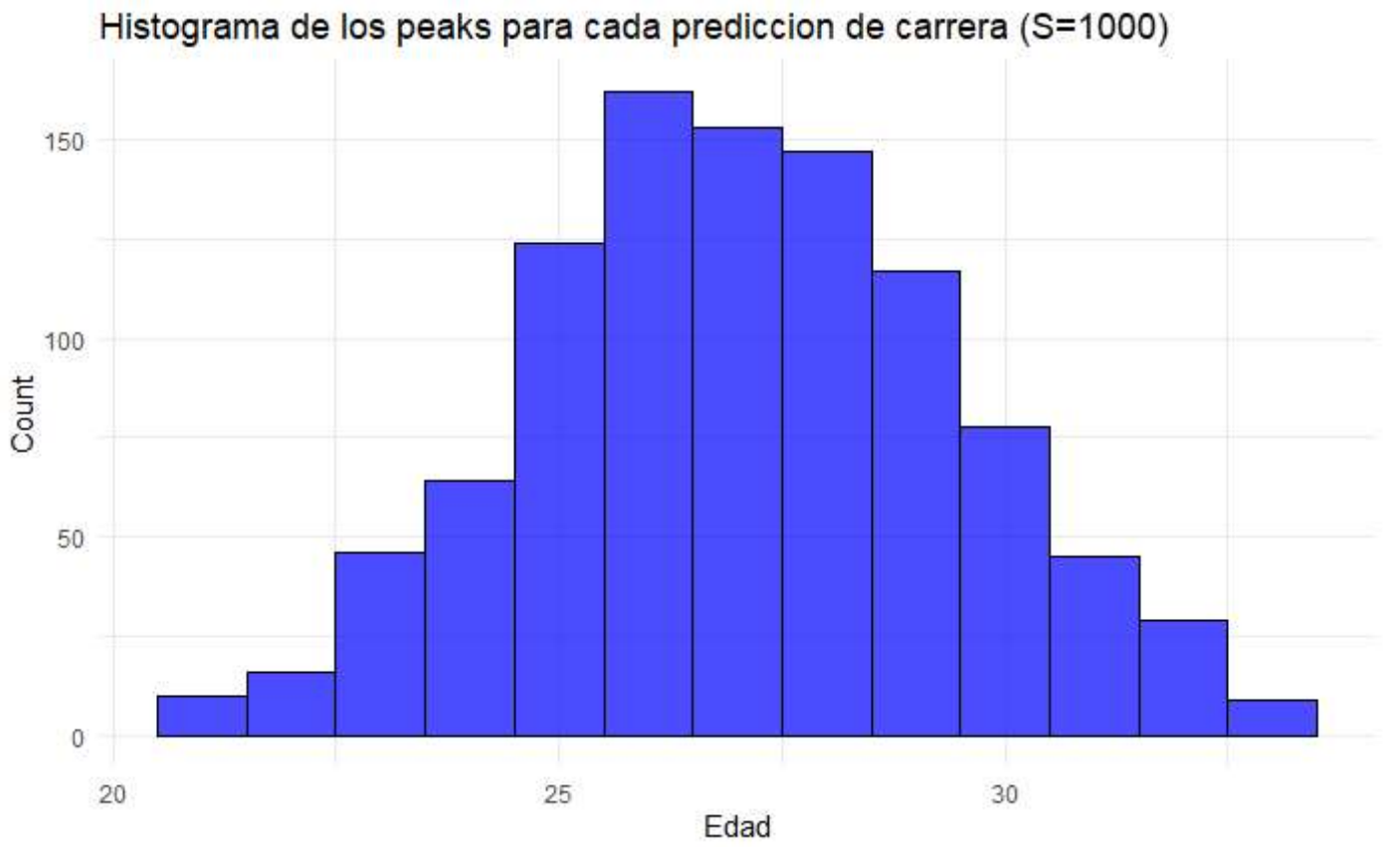
A que edad Mbappe alcanzara su precio mas alto y cual sera ese precio?

Para responder esta pregunta podemos simular posibles precios de mercado para Mbappe usando la posterior y viendo que valor tendra su pico.

Show



Show



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%

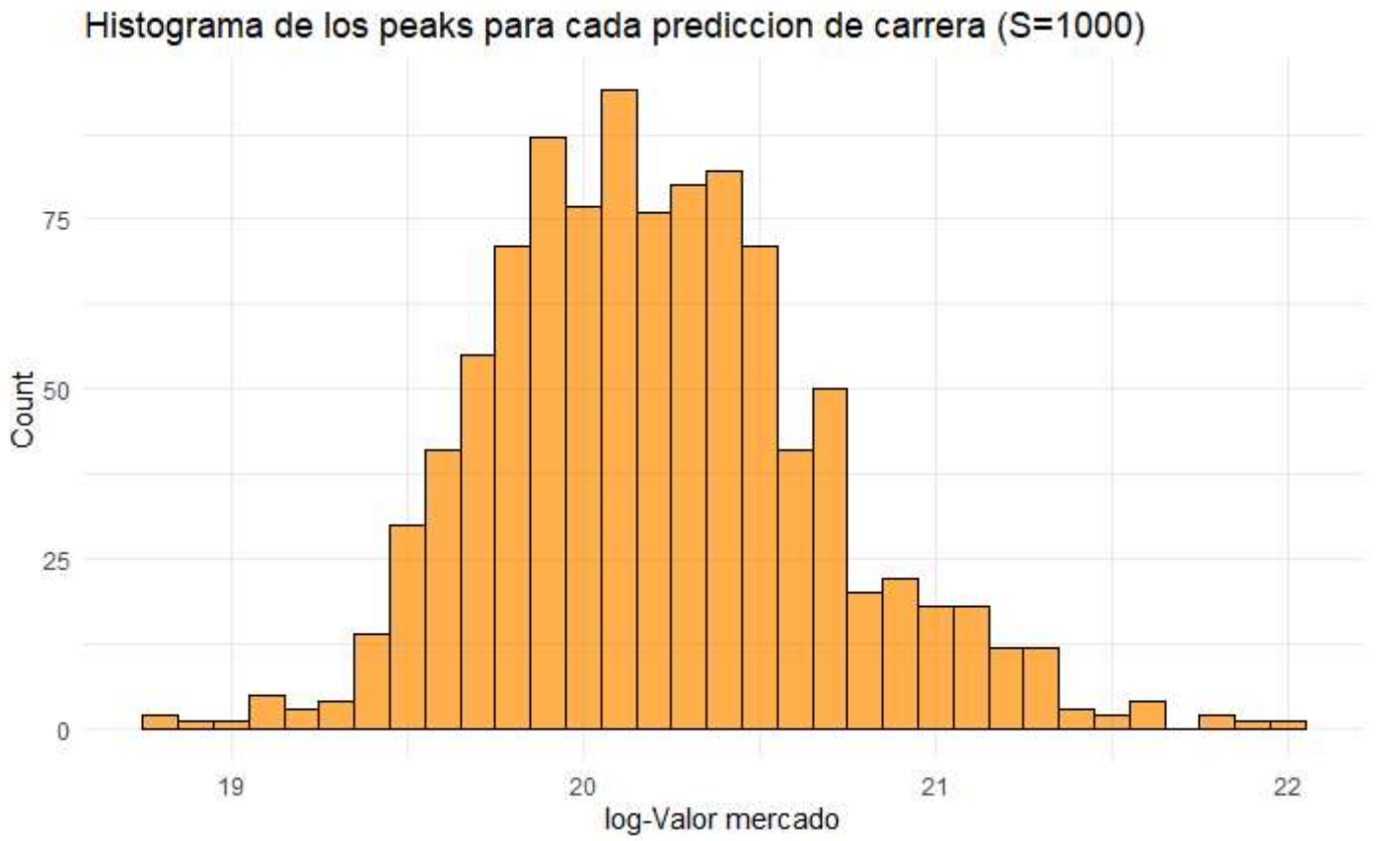
Hide

```
(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%

Show



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
```

Podemos estar seguros con una confianza del 75% 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