Aprendizaje de máquina: proyecto final

Elizabeth Viveros Vergara y Juan B. Martínez Parente

24 de noviembre de 2018

Introducción

El objetivo de este trabajo es probar distintos modelos de aprendizaje de máquina con el fin de comparar resultados; además, consideraremos dos bases de datos de interés:

- Listado de propiedades en Airbnb para Berlín, París, Roma y Barcelona (descargadas en Inside Airbnb (http://insideairbnb.com/get-the-data.html)) con variables descriptivas de cada propiedad (como número de cuartos, número de baños, ubicación, tamaño, etc.), además de inforamción de los *hosts*, *ratings*, entre otros. Tomaremos como variable dependiente el precio por noche de cada propiedad (problema de regresión).
- Información recopilada de fuentes del Coneval (http://coneval.org.mx), INEGI (http://sc.inegi.org.mx/cobdem/contenido.jsp?rf=false&solicitud=), Conapo (http://www.conapo.gob.mx/es/CONAPO/Publicaciones) e INAFED (http://www.inafed.gob.mx/en/inafed/Socioeconomico_Municipal) a nivel municipio provenientes de diferentes encuestas de 2010 y relacionadas con el índice de marginación (rubros de salud, educación, vivienda, seguridad, economía). La variable dependiente será el grado de marginación como lo determina la CONAPO (problema de claseificación).

Utilizaremos R para programar los siguientes modelos:

- regresión lineal con regularización,
- *gradient boosting*
- árboles extremadamente aleatorizados (*extremely randomized trees*). La principal diferencia respecto a los bosques aleatorios consiste en que, en cada nodo, el corte que se realiza es totalmente aleatorizado.

Limpieza y exploración de los datos

Airbnb

Las bases en crudo de cada ciudad contienen las mismas variables. Se unieron todas las bases en una sola "base maestra" y se eliminaron

a. las variables con una proporción muy alta de valores nulos (como `square_feet`);
b. las variables categóricas con muchos niveles (en general, más de 30), como `neighbourhood_cleansed`, `neighborhood_overview`, `street`, `jurisdiction_names`, entre otras;
c. las variables de texto (`name`, `sumary`, `notes`, `host_name`, `host_about`, etc.;

ya que no están realmente relacionadas con el precio por una noche en las distintas propiedades. Algunas de ellas podrían resultar interesantes en otros tipo de anális (por ejemplo, *text mining*).

De las variables restantes, se tomaron las siguientes:

- `id` de la propiedad
- `country`
- `price` : precio por una noche (se filtró para propiedades de entre 25 y 200 euros),
- `review_scores_rating` evaluación promedio de los usuarios por cada propiedad (escala continua de 0 a 10),
- `room_type` : variable categórica que fue modificada para tomar los valores `Entire home/apt` , `Private/shared` ,
- `property_type` : variable categórica que fue filtrada para los valores`Hostel` , `Bed and breakfast` , `Loft` , `Condominium` , `House` y `Apartment` (los demás niveles ocurrían un número muy pequeño de veces, y algunos de ellos eran porpiedades “atípicas”, como castillos o barcos),
- `accommodates` : máximo número de personas que podrían caber en la propiedad` (se filtró para porpiedades de hasta 6 personas),
- `guests_included` : número de personas que incluye el precio (igual o menor a `accommodates` (se filtró para porpiedades de hasta 4 personas),
- `bathrooms` : número de baños (se filtró para propiedades de hasta 3 baños),
- `bedrooms` : número de habitaciones (se filtró para propiedades de hasta 3 habitaciones),
- `beds` : número de camas (se filtró para propiedades de hasta 4 camas),
- `bed_type` : tipo de cama (individual, compartida o sofá),
- `reviews_per_month` : número pomedio de reseñas por mes (se filtró para propiedades de hasta 4 reseñas por mes),
- `cleaning_fee` : variable binaria que vale 1 si el *host* cobra una tarifa de limpieza y 0 en otro caso,
- `security_deposit` : variable binaria que vale 1 si el *host* cobra un depósito de seguridad y 0 en otro caso,
- `latitude` y `longitude` : se redondearon para tener una precisión de aproximadamentre 1 kilómetro.

Cabe notar que las variables fueron filtradas para evitar que tomaran valores atípicos tanto hacia arriba como hacia abajo (en general, se tomaron por arriba del cuantil 0.05 y por debajo del cuantil 0.95).

```
library(tidyverse)
library(magrittr)
library(data.table)

archivos <- list.files('datos/airbnb/', pattern = 'csv', recursive = F)

listings_raw <- map(archivos, ~fread(paste0('datos/airbnb/', .),
                                     colClasses = c('id' = 'text',
                                                    'zipcode' = 'text'),
                                     encoding = 'UTF-8')) %>%
  bind_rows()

listings_raw %<>%
  select(-c(contains('url'), description, scrape_id, name, summary, space,
            neighborhood_overview, notes,
            transit, access, interaction, house_rules, host_name,
            host_location, host_about,
            host_neighbourhood, neighbourhood, neighbourhood_cleansed, city,
            state,  weekly_price, monthly_price, host_response_rate,
            market:country_code, amenities, host_verifications,
            requires_license,
            host_acceptance_rate, host_listings_count, square_feet,
            jurisdiction_names, last_scraped, host_has_profile_pic,
            street, neighbourhood_group_cleansed, license, calendar_updated,
            host_since, calendar_last_scraped, first_review, last_review,
            experiences_offered, has_availability,
            is_business_travel_ready)) %>%
  filter(property_type %in% c('Hostel', 'Bed and breakfast', 'Loft',
                              'Condominium', 'House', 'Apartment')) %>%
  mutate_at(vars(contains('price'), security_deposit, cleaning_fee,
                 extra_people),
            funs(parse_number)) %>%
  # mutate(zipcode = as.numeric(zipcode)) %>%
  mutate_at(vars(id, review_scores_rating, accommodates, bathrooms, bedrooms,
                 beds, guests_included, reviews_per_month),
            funs(as.numeric)) %>%
  dplyr::filter(!country %in% c('Vatican City', 'Switzerland'))

listing_mod <- listings_raw %>%
  dplyr::filter(property_type %in% c('Hostel', 'Bed and breakfast', 'Loft',
                                     'Condominium', 'House', 'Apartment'),
                accommodates <= 6,
                beds > 0, beds <= 4,
                bedrooms > 0, bedrooms <= 3,
                bathrooms > 0, bathrooms <= 2,
                guests_included <= 4,
                reviews_per_month <= 4,
                price > 25, price <= 200) %>%
  select(id, country, price, review_scores_rating, room_type, property_type,
         accommodates, bathrooms, beds, guests_included, bed_type,
         reviews_per_month, cleaning_fee, security_deposit, zipcode,
         bedrooms, latitude, longitude) %>%
  mutate(cleaning_fee = ifelse(is.na(cleaning_fee)|cleaning_fee == 0, '0', '1'),
```

```
        security_deposit = ifelse(is.na(security_deposit)|security_deposit == 0, '0', '1'),
        bed_type = ifelse(bed_type != 'Real Bed', 'Other', bed_type),
        room_type = ifelse(room_type != 'Entire home/apt', 'Private/shared',
                           room_type)) %>%
  na.omit() %>%
  mutate_if(is.character, funs(as.factor)) %>%
  mutate(longitude = round(as.numeric(as.character(longitude)), 2),
         latitude = round(as.numeric(as.character(latitude)), 2)) %>%
  mutate_at(vars(accommodates, bathrooms, beds, zipcode, bedrooms, guests_included, cleaning_fe
e),
            funs(as.factor)) %>%
  group_by(zipcode) %>%
  filter(n() >= 100, zipcode != '') %>%
  ungroup()
```

```
listing_mod %<>%
  mutate_if(is.numeric, funs(log))
```

Finalmente, separamos la muestra en entrenamiento y prueba (70% y 30%, respectivamente):

```
set.seed(124458)
entrena <- sample_frac(listing_mod, 0.7)
prueba <- listing_mod %>% dplyr::filter(!id %in% entrena$id) %>% select(-id)
entrena %<>% select(-id)

save(entrena, prueba, file = 'datos/bases_airbnb.RData')

entrena
```

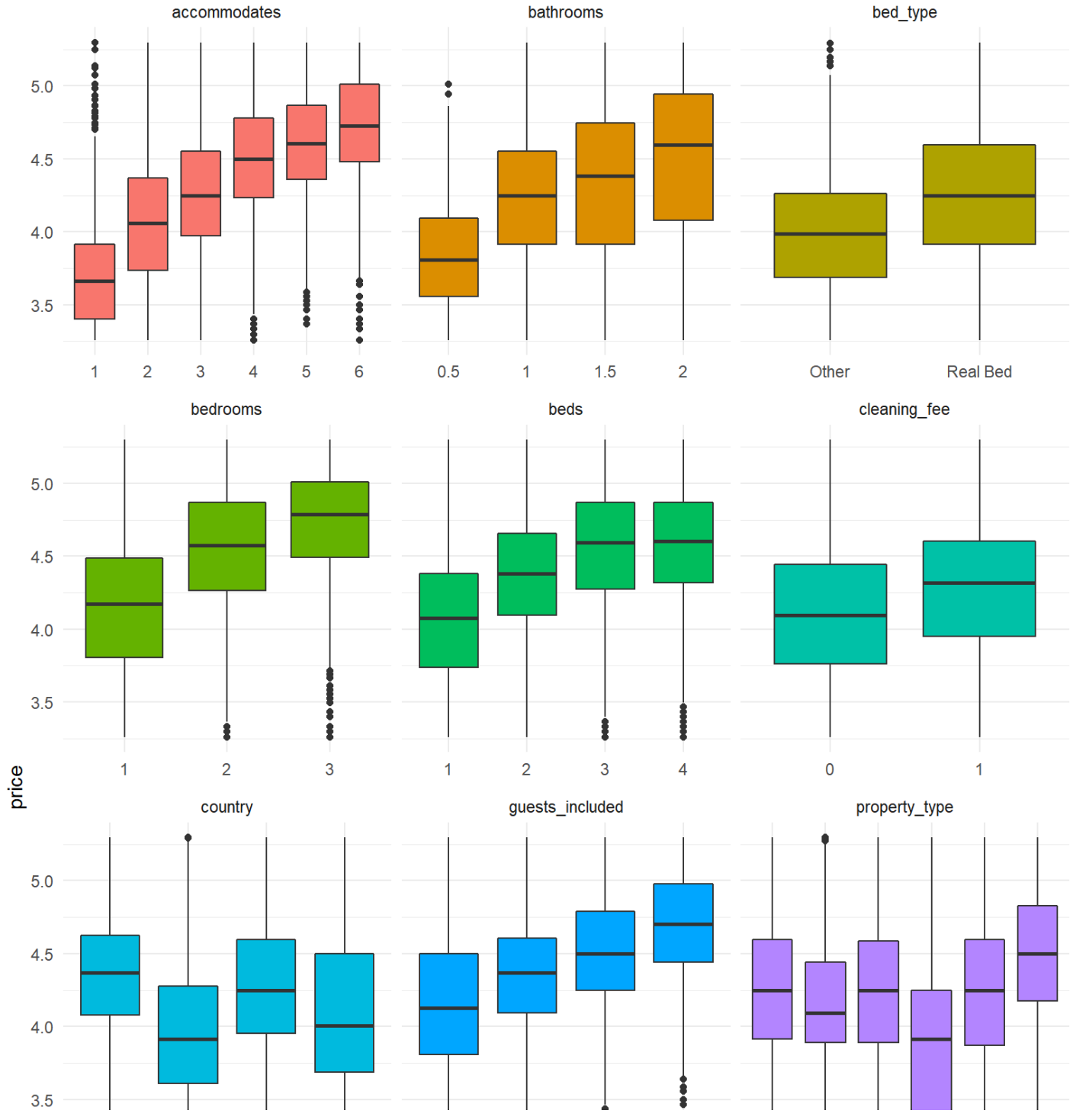
```
## # A tibble: 40,896 x 18
##    country price review_scores_r~ room_type property_type accommodates
##    <fct>   <dbl>            <dbl> <fct>     <fct>         <fct>
##  1 France   4.32             4.61 Entire h~ Apartment     2
##  2 France   4.23             4.57 Entire h~ Apartment     4
##  3 Italy    3.87             4.55 Private/~ Condominium   3
##  4 France   3.69             4.61 Private/~ Apartment     2
##  5 France   3.91             4.61 Entire h~ Loft          2
##  6 Italy    3.89             4.52 Entire h~ Loft          2
##  7 France   4.79             4.57 Entire h~ Apartment     4
##  8 France   4.13             4.54 Entire h~ Apartment     2
##  9 Italy    3.99             4.51 Private/~ Apartment     2
## 10 France   3.61             4.58 Private/~ Apartment     2
## # ... with 40,886 more rows, and 12 more variables: bathrooms <fct>,
## #   beds <fct>, guests_included <fct>, bed_type <fct>,
## #   reviews_per_month <dbl>, cleaning_fee <fct>, security_deposit <fct>,
## #   zipcode <fct>, bedrooms <fct>, latitude <dbl>, longitude <dbl>,
## #   price_gr <fct>
```

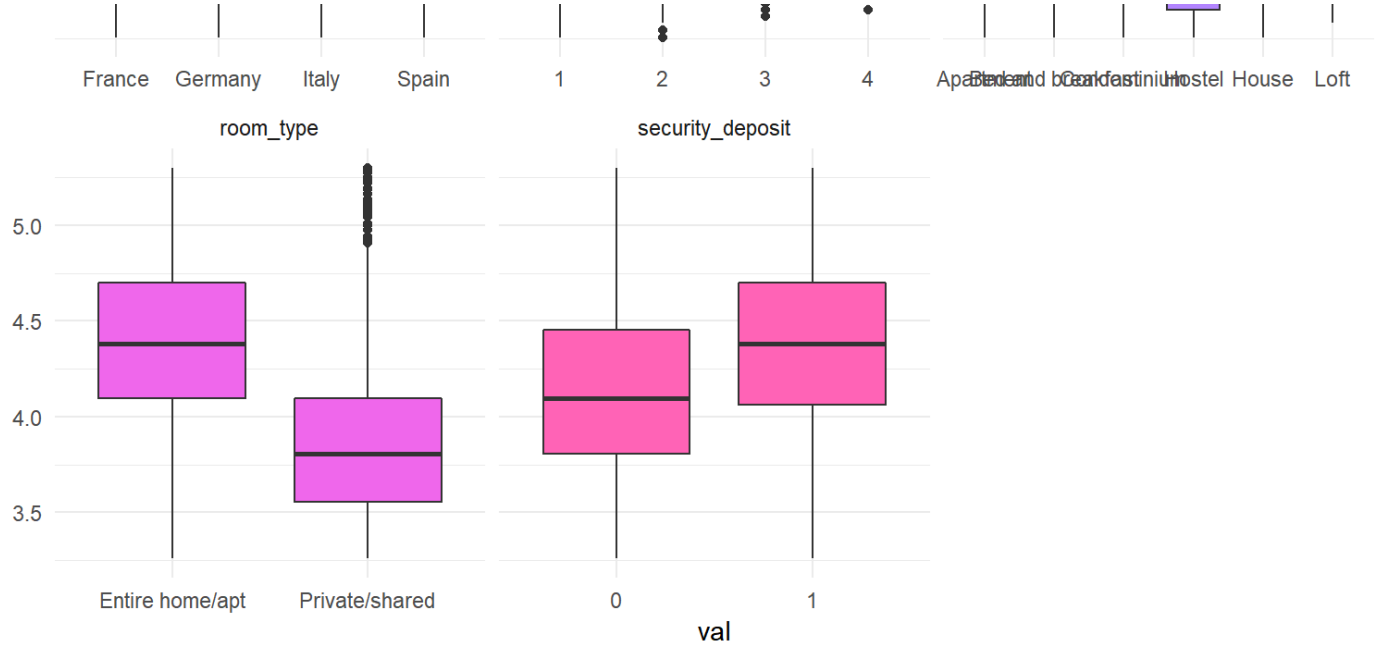
Exploratorio

Se hizo un breve análisis exploratorio de las variables seleccionadas. Se observa que el precio alcanza distintos niveles según los valores de las diferentes variables:

```
listing_mod %>%
  gather(var, val, c(country, room_type:bed_type, cleaning_fee,
                     security_deposit, bedrooms)) %>%
  ggplot(aes(val, price)) +
  facet_wrap(~var, scales = 'free_x', ncol = 3) +
  geom_boxplot(aes(fill = var)) +
  theme_minimal() +
  theme(legend.position = 'none')
```

```
## Warning: attributes are not identical across measure variables;
## they will be dropped
```





Grado de marginación

Se creo la base de datos del Índice de marginación de 2010 a nivel municipal a partir de las fuentes mencionadas arriba (INEGI, CONAPO, INAFED y Coneval) y se eligieron las variables que a continuación se mencionan para predecir el grado de marginación (muy bajo, bajo, medio, alto y muy alto), es decir, tratándose de un problema de clasificación:

- PIB per cápita
- Proporción de la población que vive en pobreza
- Proporción de la población que vive en pobreza extrema
- Proporción de la población que vive en pobreza moderada
- Proporción de la población con ingreso inferior a la línea de bienestar
- Proporción de la población con ingreso inferior a la línea de bienestar mínimo
- Proporción de la población con rezago educativo
- Proporción de la población sin acceso a los servicios básicos en la vivienda
- Proporción de la población con carencia por acceso a la alimentación
- Proporción de la población sin acceso a la seguridad social
- Proporción de la población sin acceso a los servicios de salud
- Proporción de la población con carencia por calidad y espacios de la vivienda
- Proporción de la población de 15 años o más analfabeta
- Proporción de la población en viviendas sin drenaje ni excusado
- Proporción de la población en viviendas sin energía eléctrica
- Proporción de la población en viviendas sin agua entubada
- Proporción de la población en viviendas con piso de tierra
- Tasa de mortalidad infantil
- Proporción de la población sin derechohabiencia
- Proporción de la población desocupada de 12 años y más

```
IM <- fread('datos/bae_IM.csv', encoding = 'UTF-8') %>%
  select(-predicted) %>%
  mutate(id = 1:nrow(.))

IM %>% names
```

```
##  [1] "Grado de marginación"
##  [2] "Índice de marginación"
##  [3] "PIB per cápita"
##  [4] "Pobreza (proporción)"
##  [5] "Pobreza extrema (proporción)"
##  [6] "Pobreza moderada (proporción)"
##  [7] "Población con ingreso inferior a la línea de bienestar (proporción)"
##  [8] "Población con ingreso inferior a la línea de bienestar mínimo (proporción)"
##  [9] "Rezago educativo (proporción)"
## [10] "Carencia por acceso a los servicios básicos en la vivienda (proporción)"
## [11] "Carencia por acceso a la alimentación (proporción)"
## [12] "Carencia por acceso a la seguridad social (proporción)"
## [13] "Carencia por acceso a los servicios de salud (proporción)"
## [14] "Carencia por calidad y espacios de la vivienda (proporción)"
## [15] "Mortalidad infantil (proporción)"
## [16] "Población sin derechohabiencia (proporción)"
## [17] "Población desocupada de 12 años y más (proporción)"
## [18] "id"
```

```
names(IM) <- c(
  'grado_IM',
  'indices_IM',
  'PIB_per_capita',
  'pobreza',
  'pobreza_extrema',
  'pobreza_moderada',
  'inf_bienestar',
  'inf_bienestar_min',
  'rezago_edu',
  'car_servicios_vivienda',
  'car_alimentacion',
  'car_seguridad',
  'car_salud',
  'car_espacios_vivienda',
  'mort_infantil',
  'sin_acceso_salud',
  'pob_desocupada',
  'id'
)

IM <- IM %>%
  mutate(grado_IM = ifelse(grado_IM == 0, '1 Muy bajo',
                           ifelse(grado_IM == 1, '2 Bajo',
                                  ifelse(grado_IM == 2, '3 Medio',
                                         ifelse(grado_IM == 3, '4 Alto', '5 Muy alto'))))) %>%
  mutate(grado_IM = factor(grado_IM))


set.seed(10585)
entrena <- sample_frac(IM, 0.7)
prueba <- IM %>% dplyr::filter(!id %in% entrena$id) %>% select(-id)
entrena %<>% select(-id)

x <- model.matrix(~., entrena[,-c(1, 2)])
y <- as.factor(entrena$grado_IM)

x.test <- model.matrix(~., prueba[,-c(1,2)])
y.test <- as.factor(prueba$grado_IM)

save(entrena, prueba, x, y, x.test, y.test, file = 'datos/bases_indice.RData')
```

Exploratorio

El análisis exploratorio está en el anexo ‘Análisis exploratorio’.

Modelos

Como se mecionó anteriormente, el objetivo del actual proyecto es comparar el desempeño de tres distintos tipos de modelos (regresión con regularización, *gradient boosting* y *extremely randomized trees*). A continuación presentamos los resultados para cada base de datos.

En todos los casos se hizo validación cruzada con la librería `caret` para elegir los parámetros que minimizan el error cuadrático medio (en el caso de regresión) y los que maximizan la precisión (en el caso de clasificación). Los parámetros de la validación cruzada fueron:

```
library(caret)

fitControl <- trainControl(
  method = "repeatedcv",
  number = 10,
  repeats = 10)
```

Airbnb

Regresión lineal con regularización Ridge y Lasso

Primero se hizo la validación cruzada para $\alpha \in \{0, 0.05, 0.1, \ldots, 0.95, 1\}$ y para $\lambda \in \{e^{-5}, e^{-4}, \ldots, e^{4}, e^{5}\}$.

```
library(glmnet)
x <- model.matrix(~., entrena[,-c(2,18)])
y <- entrena$price

# test_class_cv_model <- train(x, y,
#                              method = "glmnet",
#                              trControl = fitControl
#                              metric = "RMSE",
#                              tuneGrid = expand.grid(.alpha = seq(0, 1, 0.05),
#                                                     .lambda = exp(seq(-5,5,1))))

load('crossval/cv_reg_airbnb.RData')

test_class_cv_model$bestTune
```

```
##    alpha      lambda
## 12  0.05 0.006737947
```

Los valores que resultaron óptimos para el error cuadrático medio fueron $\alpha = 0.05$ y $\lambda = e^{-5}$. Corremos entonces el modelo y calculamos, para el conjunto de prueba, el error cuadrático medio y la asociación lineal (la correlación entre los valores observados y los valores predichos):

```
load('datos/bases_airbnb.RData')

x <- model.matrix(~., entrena[,-c(2,18)])
y <- entrena$price

mod_ridge <- glmnet(x, y,
                    alpha = 0.05,
                    family = 'gaussian',
                    intercept = T,
                    lambda = 0.006737947)

x.test <- model.matrix(~., prueba[,-c(2, 18)])
y.test <- prueba$price

df.reg <- data.frame(pred = predict(mod_ridge, x.test),
                     obs = prueba$price,
                     pred.exp = exp(predict(mod_ridge, x.test)),
                     obs.exp = exp(prueba$price),
                     country = prueba$country) %>%
  mutate(residuo = obs - s0)

rmse_reg <- sqrt(sum((df.reg$obs.exp - df.reg$s0.1)^2) / nrow(df.reg))
rmse_reg
```

```
## [1] 27.40256
```

```
as_lin_reg <- cor(df.reg$obs, df.reg$s0)^2
as_lin_reg
```

```
## [1] 0.5351157
```

```
library(broom)
tidy(mod_ridge) %>% select(term, estimate)
```

```
## # A tibble: 134 x 2
##    term                           estimate
##    <chr>                             <dbl>
##  1 (Intercept)                      2.75
##  2 countryGermany                  -0.0928
##  3 countrySpain                    -0.0813
##  4 review_scores_rating             0.170
##  5 room_typePrivate/shared         -0.313
##  6 property_typeBed and breakfast   0.299
##  7 property_typeCondominium         0.0202
##  8 property_typeHostel              0.129
##  9 property_typeHouse               0.0752
## 10 property_typeLoft                0.155
## # ... with 124 more rows
```

Gradient boosting

Hicimos validación cruzada para los parámetros `interaction.depth` en el conjunto $\{1, 2, 3\}$, `n.trees` en el conjunto $\{50, 100, 150\}$. `shrinkage` y `n.minobsinnode` se tomaron constantes con los valores respectivos de 0.1 y 10.

```
library(gbm)
```

```
## Loaded gbm 2.1.4
```

```
#gbmFit1 <- train(price ~ ., data = entrena,
#                 method = "gbm",
#                 trControl = fitControl,
#                 verbose = FALSE)

load('crossval/cv_gbm_airbnb.RData')

gbmFit1$bestTune
```

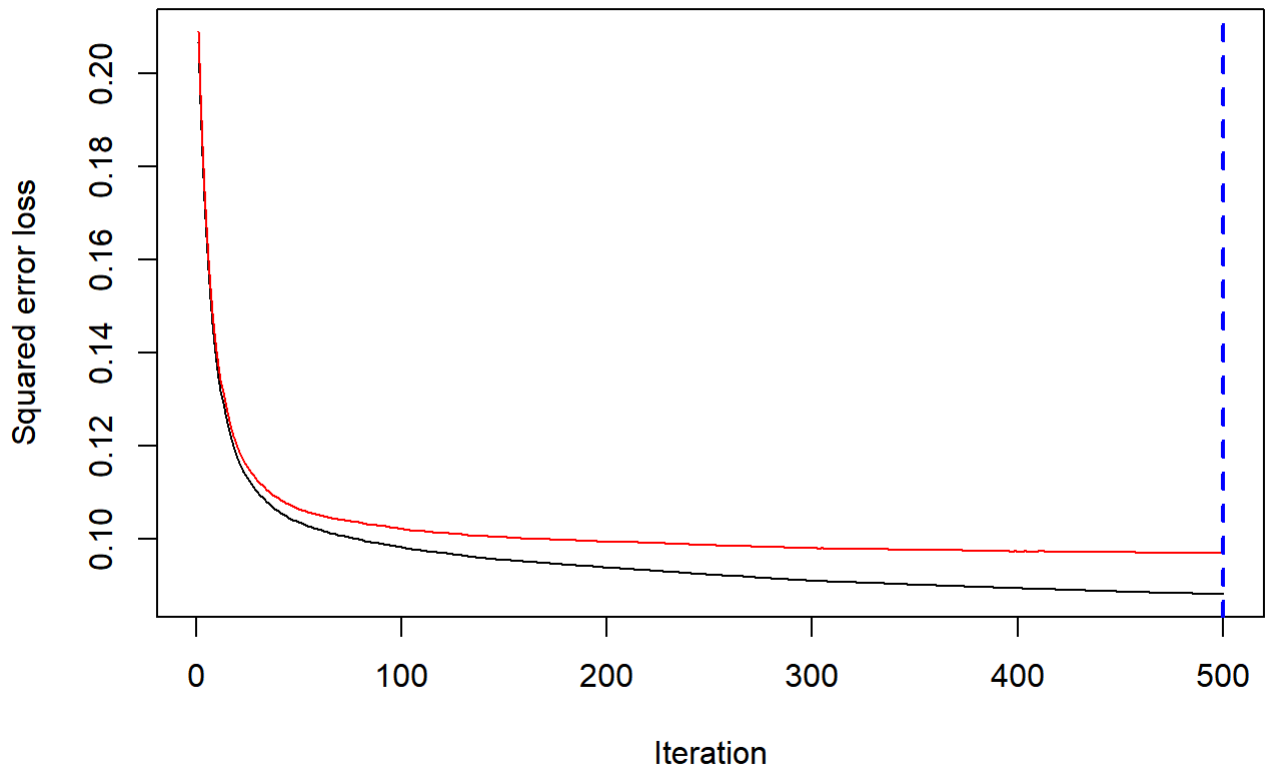
```
##   n.trees interaction.depth shrinkage n.minobsinnode
## 9     150                 3       0.1             10
```

Los valores que resultaron óptimos para el error cuadrático medio fueron `interaction.depth = 3` y `n.trees = 150`. Corremos entonces el modelo y calculamos, para el conjunto de prueba, el error cuadrático medio y la asociación lineal (la correlación entre los valores observados y los valores predichos):

```
mod_boosting <- gbm(price ~ .,
                    data = entrena %>% select(-c(price_gr)),
                    n.trees = 500,
                    interaction.depth = 3,
                    shrinkage = 0.1, # tasa de aprendizaje
                    n.minobsinnode = 10,
                    bag.fraction = 1,
                    train.fraction = 0.75)
```

```
## Distribution not specified, assuming gaussian ...
```

```
df.gbm <- data.frame(pred = predict(mod_boosting, prueba),
                     obs = prueba$price,
                     country = prueba$country) %>%
  mutate(pred.exp = exp(pred),
         obs.exp = exp(obs)) %>%
  mutate(residuo = obs - pred)
```




```
## Using 500 trees...
```

```
rmse_gbm <- sqrt(sum((df.gbm$obs.exp - df.gbm$pred.exp)^2) / nrow(df.gbm))
rmse_gbm
```

```
## [1] 26.19675
```

```
as_lin_gbm <- cor(df.gbm$obs, df.gbm$pred)^2
as_lin_gbm
```

```
## [1] 0.5752033
```

Extremely randomized trees

Hicimos validación cruzada para los parámetros `mtry` en el conjunto $\{4, 9, 16\}$, `numRandomCuts` en el conjunto $\{1, 2, 3\}$.

```
library(extraTrees)
```

```
## Loading required package: rJava
```

```
# gbmFit2 <- train(x, y,
#                  method = "extraTrees",
#                  trControl = fitControl)


load('crossval/cv_ET_airbnb.RData')

gbmFit2$bestTune
```

```
##   mtry numRandomCuts
## 1    5            18
```

Los valores que resultaron óptimos para el error cuadrático medio fueron `mtry = xxx` y `numRandomCuts = yyy`. Corremos entonces el modelo y calculamos, para el conjunto de prueba, el error cuadrático medio y la asociación lineal (la correlación entre los valores observados y los valores predichos):

```
# load('datos/mod_ET.RData')

mod_ET <- extraTrees(x, y, mtry = 5, numRandomCuts = 18)

df.ET <- data.frame(pred = predict(mod_ET, x.test),
                    obs = prueba$price,
                    country = prueba$country) %>%
  mutate(pred.exp = exp(pred),
         obs.exp = exp(obs)) %>%
  mutate(residuo = obs - pred)

rmse_ET <- sqrt(sum((df.ET$obs.exp - df.ET$pred.exp)^2) / nrow(df.ET))
rmse_ET
```

```
## [1] 26.76521
```

```
as_lin_ET <- cor(df.ET$obs, df.ET$pred)^2
as_lin_ET
```

```
## [1] 0.5505546
```

Comparación de los resultados

```
rmse_df <- data.frame(modelo = c('regularizacion', 'gbm', 'extra_trees'),
                      rmse = c(rmse_reg, rmse_gbm, rmse_ET),
                      asociacion_lineal = c(as_lin_reg, as_lin_gbm, as_lin_ET))
rmse_df
```

```
##           modelo     rmse asociacion_lineal
## 1 regularizacion 27.40256         0.5351157
## 2            gbm 26.19675         0.5752033
## 3    extra_trees 26.76521         0.5505546
```

```
library(Rmisc)

ajuste_reg <- df.reg %>%
  ggplot(aes(obs, s0)) +
  geom_jitter(aes(color = country), size = 0.5, width = 0.25) +
  geom_abline() +
  coord_equal() +
  theme_minimal() +
  theme(legend.position = 'bottom') +
  labs(title = 'Observados vs. ajustados (log) (regresión)')

ajuste_reg_exp <- df.reg %>%
  ggplot(aes(obs.exp, s0.1)) +
  geom_jitter(aes(color = country), size = 0.5, width = 1) +
  geom_abline() +
  coord_equal() +
  scale_x_continuous(labels = scales::comma) +
  scale_y_continuous(labels = scales::comma) +
  theme_minimal() +
  theme(legend.position = 'bottom') +
  labs(title = 'Observados vs. ajustados (regresión)')

residuales_reg <- df.reg %>%
  ggplot(aes(residuo)) +
  geom_density(color = 'red', size = 1, fill = 'red', alpha = 0.2) +
  geom_histogram(aes(y = ..density..), fill = 'royal blue', alpha = 0.75) +
  theme_minimal() +
  theme(legend.position = 'bottom') +
  labs(title = 'Residuales (regresión)')

ajuste_gbm <- df.gbm %>%
  ggplot(aes(obs, pred)) +
  geom_jitter(aes(color = country), size = 0.5, width = 0.25) +
  geom_abline() +
  coord_equal() +
  theme_minimal() +
  theme(legend.position = 'bottom') +
  labs(title = 'Observados vs. ajustados (log) (gbm)')

ajuste_gbm_exp <- df.gbm %>%
  ggplot(aes(obs.exp, pred.exp)) +
  geom_jitter(aes(color = country), size = 0.5, width = 1) +
  geom_abline() +
  coord_equal() +
  scale_x_continuous(labels = scales::comma) +
  scale_y_continuous(labels = scales::comma) +
  theme_minimal() +
  theme(legend.position = 'bottom') +
  labs(title = 'Observados vs. ajustados (gbm)')

residuales_gbm <- df.gbm %>%
  ggplot(aes(residuo)) +
  geom_density(color = 'red', size = 1, fill = 'red', alpha = 0.2) +
```

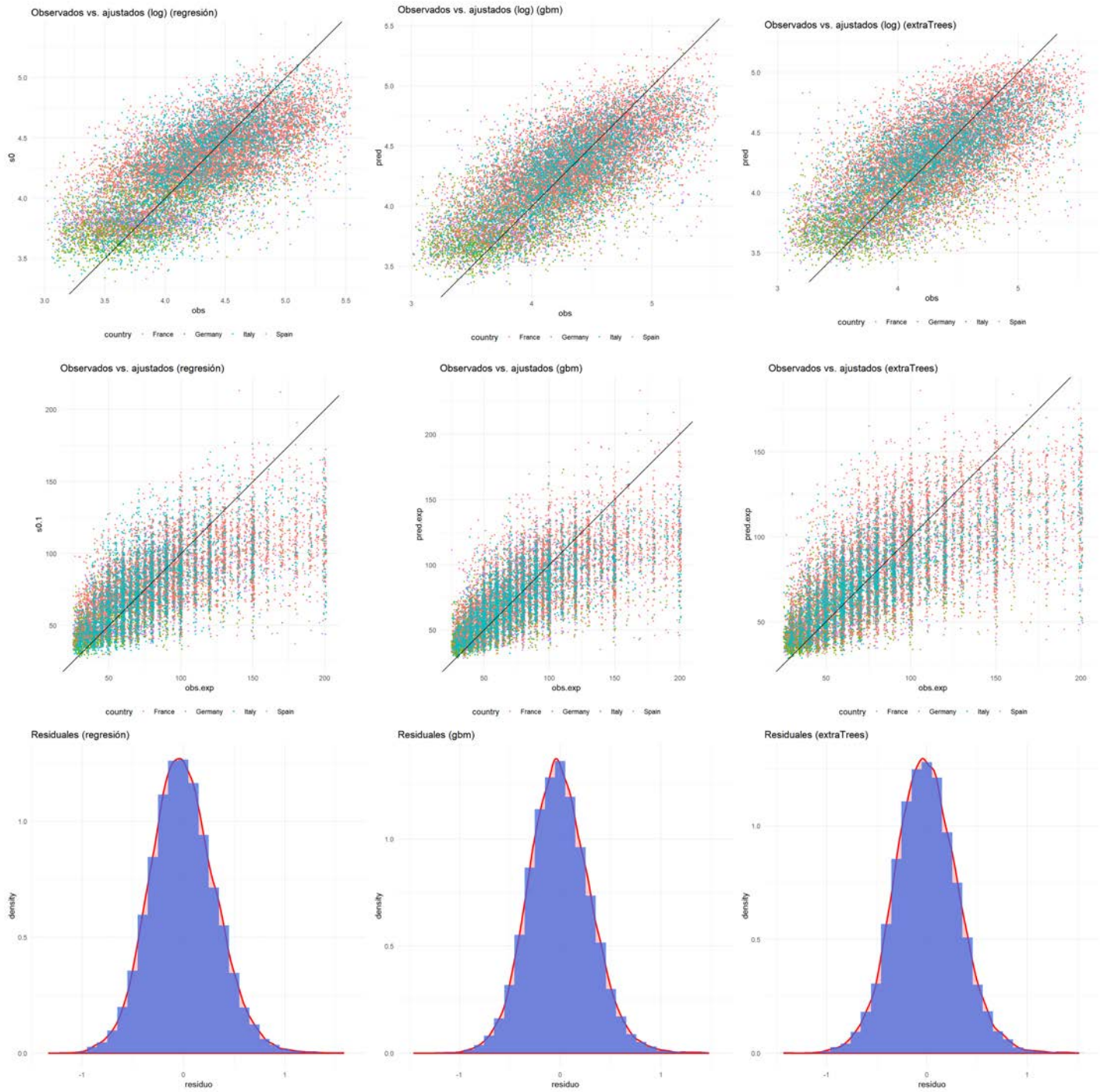
```
  geom_histogram(aes(y = ..density..), fill = 'royal blue', alpha = 0.75) +
  theme_minimal() +
  theme(legend.position = 'bottom') +
  labs(title = 'Residuales (gbm)')

ajuste_ET <- df.ET %>%
  ggplot(aes(obs, pred)) +
  geom_jitter(aes(color = country), size = 0.5, width = 0.25) +
  geom_abline() +
  coord_equal() +
  theme_minimal() +
  theme(legend.position = 'bottom') +
  labs(title = 'Observados vs. ajustados (log) (extraTrees)')

ajuste_ET_exp <- df.ET %>%
  ggplot(aes(obs.exp, pred.exp)) +
  geom_jitter(aes(color = country), size = 0.5, width = 1) +
  geom_abline() +
  coord_equal() +
  scale_x_continuous(labels = scales::comma) +
  scale_y_continuous(labels = scales::comma) +
  theme_minimal() +
  theme(legend.position = 'bottom') +
  labs(title = 'Observados vs. ajustados (extraTrees)')

residuales_ET <- df.ET %>%
  ggplot(aes(residuo)) +
  geom_density(color = 'red', size = 1, fill = 'red', alpha = 0.2) +
  geom_histogram(aes(y = ..density..), fill = 'royal blue', alpha = 0.75) +
  theme_minimal() +
  theme(legend.position = 'bottom') +
  labs(title = 'Residuales (extraTrees)')

multiplot(ajuste_reg, ajuste_reg_exp, residuales_reg,
          ajuste_gbm, ajuste_gbm_exp, residuales_gbm,
          ajuste_ET, ajuste_ET_exp, residuales_ET,
          cols = 3)
```



En general, los tres modelos se comportan de forma muy similar. El error cuadrático medio es muy cercano en cada caso (alrededor de 26 euros), y el ajuste lineal es bastante malo (entre 0.53 y 0.57). Sin emabrgo, si hubiera que elegir un modelo, la mejor elección sería el de *gradient boosting*.

La calidad del ajuste no se debe a algún problema en los modelos seleccionados, sino a los datos. No tiene sentido incorporar las variables que en un inicio se quitaron de la base, ya que, como se vio, no guardan relación con el precio. Valdría la pena, en un segundo ejercicio, buscar fuentes de información ajenas a Airbnb que complementen los datos y ayuden a mejorar el ajuste (quizá exista información sobre la edad promedio de los edificios por zona en las ciudades que tomamos o alguna variable similar).

Grado de marginación

Regresión lineal con regularización Ridge y Lasso

La validación cruzada arrojó los siguientes valores óptimos para el error cuadrático medio: $\alpha = 0.05$ y $\lambda = e^{-5}$.

```
library(glmnet)
x <- model.matrix(~., entrena[,-c(2,18)])
y <- entrena$price

# test_class_cv_model <- train(x, y,
#                              method = "glmnet",
#                              trControl = cctrlR,
#                              tuneGrid = expand.grid(.alpha = seq(0, 1, 0.05),
#                                                     .lambda = exp(seq(-5,5,1))))

load('crossval/cv_reg_indice.RData')

test_class_cv_model$bestTune
```

```
##    alpha      lambda
## 12  0.05 0.006737947
```

Corremos entonces el modelo y calculamos, para el conjunto de prueba, el error cuadrático medio y la asociación lineal (la correlación entre los valores observados y los valores predichos):

```
load('datos/bases_indice.RData')

mod_ridge <- glmnet(x, y,
                    family = 'multinomial',
                    alpha = 0.05,
                    intercept = T,
                    lambda = 0.006737947)

df.reg <- data.frame(pred = predict(mod_ridge, x.test, type = 'response')) %>%
  mutate(num = apply(., 1, which.max),
         # num = as.numeric(num),
         obs = y.test) %>%
  mutate(pred = ifelse(num == 1, '1 Muy bajo',
                       ifelse(num == 2, '2 Bajo',
                              ifelse(num == 3, '3 Medio',
                                     ifelse(num == 4, '4 Alto', '5 Muy alto')))))

acc_reg <- round(100 * sum(df.reg$obs == df.reg$pred) / nrow(df.reg), 1)
acc_reg
```

```
## [1] 72.7
```

```
conf_matrix_reg <- round(100 * prop.table(table(df.reg$obs, df.reg$pred), margin = 1), 1)
diag_reg <- diag(conf_matrix_reg)

conf_matrix_reg
```

```
##
##              1 Muy bajo 2 Bajo 3 Medio 4 Alto 5 Muy alto
##   1 Muy bajo       73.8   23.8     2.5    0.0        0.0
##   2 Bajo           11.1   54.0    34.9    0.0        0.0
##   3 Medio           0.7    5.8    85.9    6.5        1.0
##   4 Alto            0.0    0.0    31.4   50.8       17.8
##   5 Muy alto        0.0    0.0     1.6   17.2       81.1
```

```
conf_matrix_reg %<>%
  as.data.frame() %>%
  setNames(c('obs', 'pred', 'freq_reg'))

tidy(mod_ridge) %>% select(class, term, estimate) %>% spread(class, estimate)
```

```
## # A tibble: 16 x 6
##    term              `1 Muy bajo` `2 Bajo` `3 Medio` `4 Alto` `5 Muy alto`
##    <chr>                    <dbl>    <dbl>     <dbl>    <dbl>        <dbl>
##  1 ""                        8.08     2.77   -1.40      -4.08       -5.36
##  2 car_alimentacion          3.70    -0.408  -1.19      -1.69        0.0806
##  3 car_espacios_viv~        -4.16    -2.55   -0.230      2.24        5.02
##  4 car_salud                 5.31     2.07   -0.678     -2.10       -4.30
##  5 car_seguridad            -1.33     0.813   0.834     -1.22        0.623
##  6 car_servicios_vi~        -3.74    -1.53    1.09       1.95        2.05
##  7 inf_bienestar             0.119   -0.229  -0.529      1.03       -0.0914
##  8 inf_bienestar_min        -4.86    -4.05   -1.29       4.42        6.06
##  9 mort_infantil            -6.53    -1.49   -0.627      2.07        7.34
## 10 PIB_per_capita           -8.68     2.79    4.66       3.47       -3.17
## 11 pob_desocupada           24.5     16.5    NA        -15.4       -22.4
## 12 pobreza                  -3.52    -1.48    2.38       2.11        0.230
## 13 pobreza_extrema          -3.70    -5.01    0.167      3.27        4.99
## 14 pobreza_moderada         -0.326    7.22    5.60      -1.50      -10.6
## 15 rezago_edu              -11.1     -8.38    0.00128    6.66       12.3
## 16 sin_acceso_salud         -3.09    -0.149   0.734      0.778       1.43
```

Gradient boosting

Se usaron los mismos valores para la validación cruzada que para la base de Airbnb. De igual forma, los valores que resultaron ótpimos para el error cuadrático medio fueron `interaction.depth = 3` y `n.trees = 150`.

```
# gbmFit1 <- train(as.factor(grado_IM) ~ ., data = entrena %>% select(-indices_IM),
#                  method = "gbm",
#                  trControl = fitControl,
#                  verbose = FALSE)

load('crossval/cv_gbm_indice.RData')

gbmFit1$bestTune
```

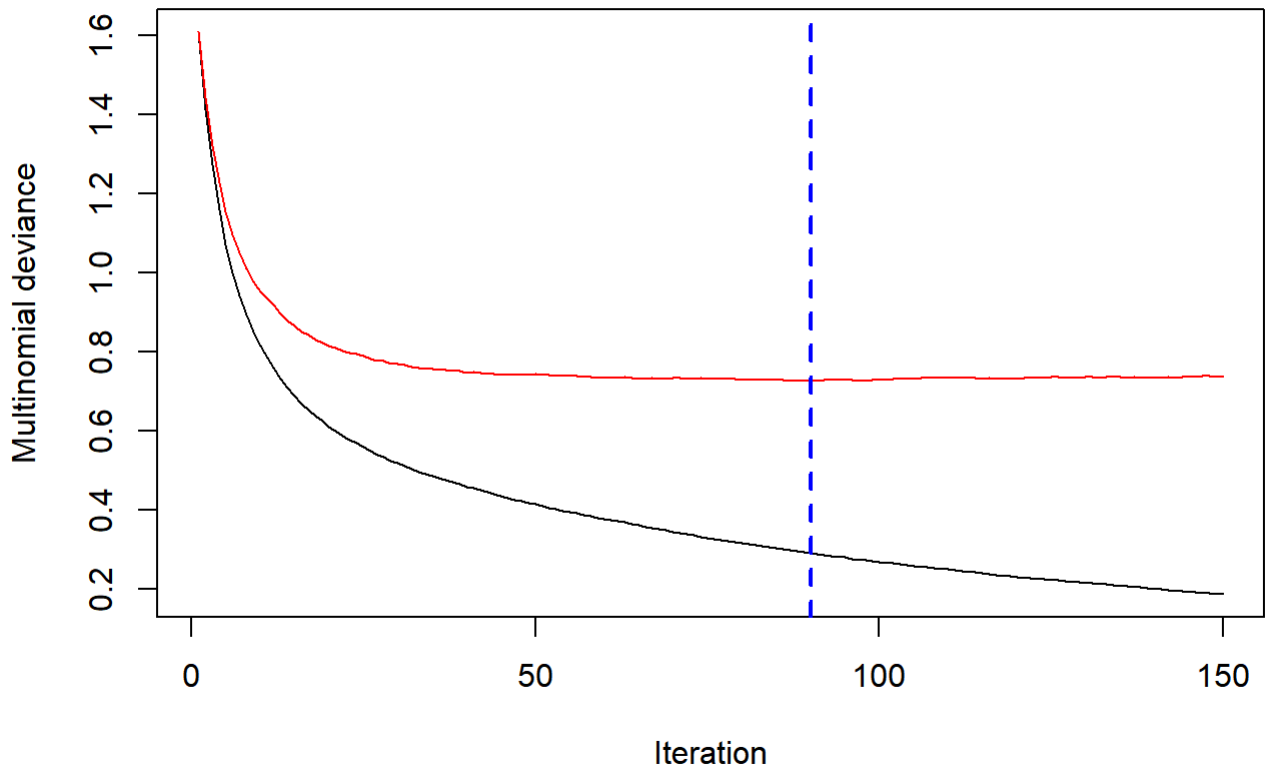
```
##   n.trees interaction.depth shrinkage n.minobsinnode
## 9     150                 3       0.1             10
```

Corremos entonces el modelo y calculamos, para el conjunto de prueba, el error cuadrático medio y la asociación lineal (la correlación entre los valores observados y los valores predichos):

```
mod_boosting <- gbm(factor(grado_IM) ~ .,
                    data = entrena %>% select(-indices_IM),
                    n.trees = 150,
                    interaction.depth = 3,
                    shrinkage = 0.1,
                    n.minobsinnode = 10,
                    bag.fraction = 1,
                    train.fraction = 0.75)
```

```
## Distribution not specified, assuming multinomial ...
```

```
df.gbm <- data.frame(pred = predict(mod_boosting, prueba %>% mutate(grado_IM = factor(grado_I
M)),
                                    type = 'response')) %>%
  mutate(num = apply(., 1, which.max),
         obs = as.character(prueba$grado_IM)) %>%
  mutate(pred = ifelse(num == 1, '1 Muy bajo',
                       ifelse(num == 2, '2 Bajo',
                              ifelse(num == 3, '3 Medio',
                                     ifelse(num == 4, '4 Alto', '5 Muy alto')))))
```



```
## Using 90 trees...
```

```
acc_gbm <- round(100 * sum(df.gbm$obs == df.gbm$pred) / nrow(df.gbm), 1)
acc_gbm
```

```
## [1] 75.3
```

```
conf_matrix_gbm <- round(100 * prop.table(table(df.gbm$obs, df.gbm$pred), margin = 1), 1)

diag_gbm <- diag(conf_matrix_gbm)
conf_matrix_gbm
```

```
##
##              1 Muy bajo 2 Bajo 3 Medio 4 Alto 5 Muy alto
##   1 Muy bajo       71.2   27.5     1.2    0.0        0.0
##   2 Bajo            7.1   59.5    33.3    0.0        0.0
##   3 Medio           0.0    4.1    85.6    8.9        1.4
##   4 Alto            0.0    0.0    18.6   59.3       22.0
##   5 Muy alto        0.0    0.0     0.8   13.9       85.2
```

```
conf_matrix_gbm %<>%
  as.data.frame() %>%
  setNames(c('obs', 'pred', 'freq_gbm'))
```

Extremely randomized trees

Hicimos validación cruzada para los parámetros `mtry = 2` en el conjunto $\{4, 9, 16\}$, `numRandomCuts = 3` en el conjunto $\{1, 2, 3\}$.

```
# gbmFit1 <- train(x, y,
#                  method = "extraTrees",
#                  trControl = fitControl)

load('crossval/cv_ET_indice.RData')

gbmFit1$bestTune
```

```
##   mtry numRandomCuts
## 3    2             3
```

Los valores que resultaron óptimos para el error cuadrático medio fueron `mtry = 2` y `numRandomCuts = 3`. Corremos entonces el modelo y calculamos, para el conjunto de prueba, el error cuadrático medio y la asociación lineal (la correlación entre los valores observados y los valores predichos):

```
mod_ET <- extraTrees(x, y, ntree = 1000, mtry = 2, numRandomCuts = 3)

df.ET <- data.frame(pred = predict(mod_ET, x.test),
                    obs = prueba$grado_IM)

acc_ET <- round(100 * sum(df.ET$obs == df.ET$pred) / nrow(df.ET), 1)
acc_ET
```

```
## [1] 75.3
```

```
conf_matrix_ET <- round(100 * prop.table(table(df.ET$obs, df.ET$pred), margin = 1), 1)

diag_ET <- diag(conf_matrix_ET)
conf_matrix_ET
```

```
##
##              1 Muy bajo 2 Bajo 3 Medio 4 Alto 5 Muy alto
##   1 Muy bajo       76.2   22.5     1.2    0.0        0.0
##   2 Bajo            7.1   61.9    31.0    0.0        0.0
##   3 Medio           0.0    3.8    85.9    9.3        1.0
##   4 Alto            0.0    0.0    25.4   53.4       21.2
##   5 Muy alto        0.0    0.0     0.8   14.8       84.4
```

```
conf_matrix_ET %<>%
  as.data.frame() %>%
  setNames(c('obs', 'pred', 'freq_ET'))
```

Comparación de los resultados

```
conf_reg <- conf_matrix_reg %>%
  ggplot(aes(pred, obs)) +
  geom_tile(aes(fill = freq_reg)) +
  geom_text(aes(pred, obs, label = freq_reg)) +
  scale_fill_continuous(low = 'white', high = 'dark green',
                        limits = c(0,100)) +
  theme_minimal() +
  theme(legend.position = 'none') +
  labs(title = 'Matriz de confusión (regresión)')

conf_gbm <- conf_matrix_gbm %>%
  ggplot(aes(pred, obs)) +
  geom_tile(aes(fill = freq_gbm)) +
  geom_text(aes(pred, obs, label = freq_gbm)) +
  scale_fill_continuous(low = 'white', high = 'dark green',
                        limits = c(0,100)) +
  theme_minimal() +
  theme(legend.position = 'none') +
  labs(title = 'Matriz de confusión (gbm)')

conf_ET <- conf_matrix_ET %>%
  ggplot(aes(pred, obs)) +
  geom_tile(aes(fill = freq_ET)) +
  geom_text(aes(pred, obs, label = freq_ET)) +
  scale_fill_continuous(low = 'white', high = 'dark green',
                        limits = c(0,100)) +
  theme_minimal() +
  theme(legend.position = 'none') +
  labs(title = 'Matriz de confusión (extraTrees)')

acc_df <- data.frame(modelo = c('regularizacion', 'gbm', 'extra_trees'),
             accuracy = c(acc_reg, acc_gbm, acc_ET))

acc_df
```

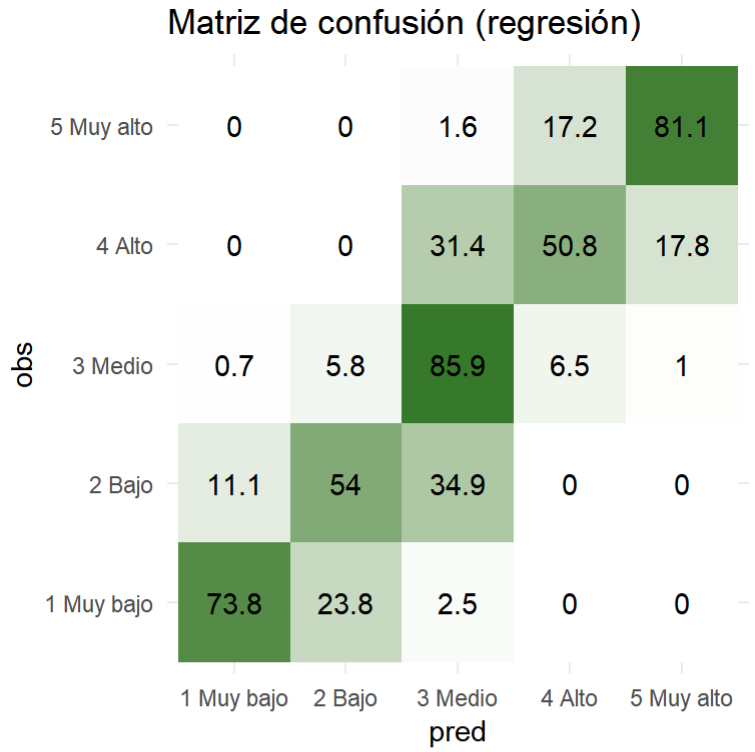
```
##           modelo accuracy
## 1 regularizacion     72.7
## 2            gbm     75.3
## 3    extra_trees     75.3
```

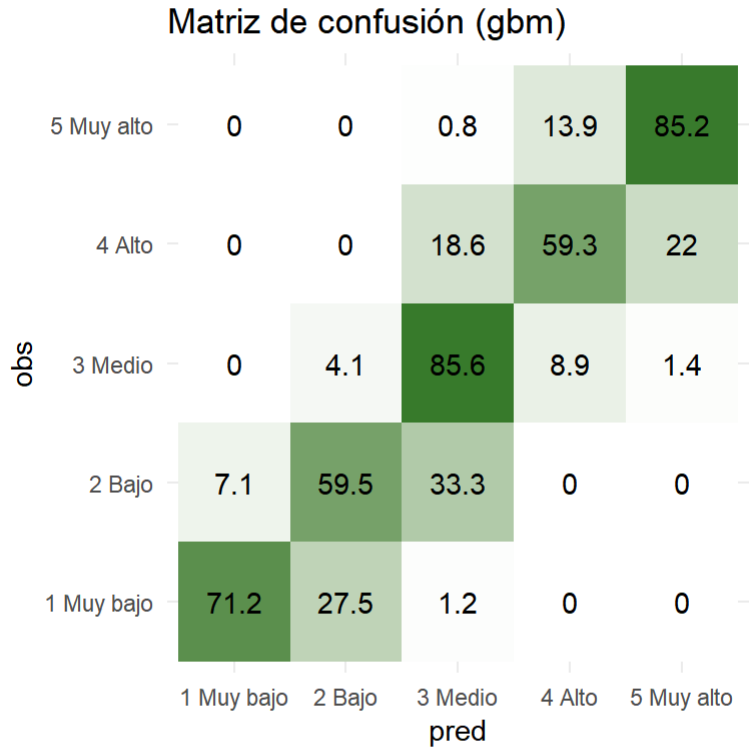
```
conf_matrices <- data.frame(reg = diag_reg,
                            gbm = diag_gbm,
                            ET = diag_ET)

conf_matrices
```

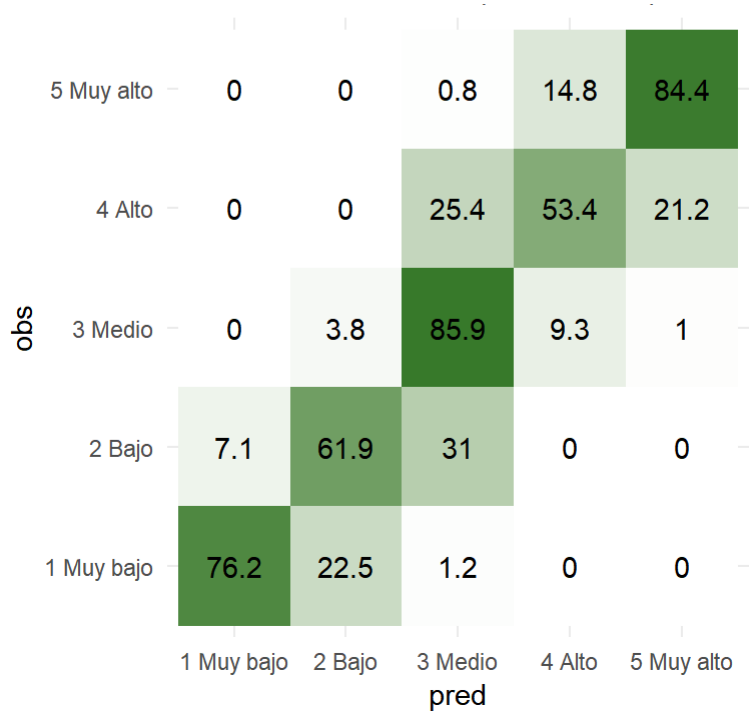
```
##            reg  gbm   ET
## 1 Muy bajo 73.8 71.2 76.2
## 2 Bajo     54.0 59.5 61.9
## 3 Medio    85.9 85.6 85.9
## 4 Alto     50.8 59.3 53.4
## 5 Muy alto 81.1 85.2 84.4
```

```
multiplot(conf_reg, conf_gbm, conf_ET)
```





Matriz de confusión (extraTrees)



Como ocurrió con los datos de Airbnb, el desempeño de los tres modelos es similar. La precisión está alrededor del 75%, siendo la mejor la de *extremely randomized trees*.

Si se observa los valores de las diagonales de las matrices confusión, vemos que *extremely randomized trees* no es el mejor prediciendo en todos los niveles del índice de marginación. La regresión con regularización es la que mjor predice el nivel de marginación medio, *gradient boosting* predice mejor los niveles alto y muy alto, y *extremely randomized trees* es el mejor prediciendo los niveles muy bajo y bajo.

Anexo. Análisis exploratorio del índice de marginación

In [1]:
```
import sklearn as sk
from sklearn import preprocessing
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns #Control figure
import numpy as np
from xgboost import XGBClassifier
matplotlib.style.use('ggplot')
%matplotlib inline
```

In [2]:
```
doc = 'C:/Users/Elizabeth/Desktop/Primer semestre MCD - ITAM/Aprendizaje de Ma
quina/Proyecto final/img_2.csv'
IM = pd.read_csv(doc, encoding="iso-8859-1") #index_col=0 Este es para que la
 columna clave aparezca como un índice y no como variable.
```

In [3]:
```
IM.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2456 entries, 0 to 2455
Data columns (total 26 columns):
Clave
     2456 non-null int64
Estado
     2456 non-null object
Municipio
     2456 non-null object
Índice de marginación escala
0 a 100                                              2456 non-null float64
Grado de marginación
     2456 non-null int64
Población total (2010)
     2456 non-null int64
PIB
Estatal
(2008=100)                                                          2456 non
-null float64
Pobreza (Miles de personas)
     2456 non-null int64
Pobreza extrema (Miles de personas)
     2456 non-null int64
Pobreza moderada (Miles de personas)
     2456 non-null int64
Población con ingreso inferior a la línea de bienestar
     2456 non-null int64
Población con ingreso inferior a la línea de bienestar mínimo
     2456 non-null int64
Rezago educativo
(Miles de personas)                                              2456 non-nul
l int64
Carencia por acceso a los servicios de salud
(Miles de personas)                  2456 non-null int64
Carencia por acceso a la seguridad social
(Miles de personas)                      2456 non-null int64
Carencia por calidad y espacios de la vivienda
(Miles de personas)                2456 non-null int64
Carencia por acceso a los servicios básicos en la vivienda
(Miles de personas)    2456 non-null int64
Carencia por acceso a la alimentación
(Miles de personas)                           2456 non-null int64
% de Población de 15 años o más analfabeta
     2456 non-null float64
% Ocupantes en viviendas sin drenaje ni excusado
     2456 non-null float64
% Ocupantes en viviendas sin energía eléctrica
     2456 non-null float64
% Ocupantes en viviendas sin agua entubada
     2456 non-null float64
% Ocupantes en viviendas con piso de tierra
     2456 non-null float64
Tasa de mortalidad infantil
     2456 non-null float64
Sin derechohabiencia
(miles de personas)                                          2456 non-null in
t64
```

```
Población desocupada de 12 años y más
(Miles de personas)                          2456 non-null int64
dtypes: float64(8), int64(16), object(2)
memory usage: 499.0+ KB
```

In [4]:
```
IM['Índice de marginación'] = IM['Índice de marginación escala \n0 a 100'] / 1
00
```

In [5]:
```
IM['PIB per cápita_1'] = IM['PIB \nEstatal\n(2008=100)'] / IM['Población total
 (2010)']
```

In [6]:
```
PIB = pd.DataFrame(IM, columns=['PIB per cápita_1'])
scaler = preprocessing.MinMaxScaler()
IM['PIB per cápita'] = [i[0] for i in scaler.fit_transform(PIB)]
```

In [7]:
```
IM['Pobreza (proporción)'] = IM['Pobreza (Miles de personas)'] / IM['Población
 total (2010)']
IM['Pobreza extrema (proporción)'] = IM['Pobreza extrema (Miles de personas)']
 / IM['Población total (2010)']
IM['Pobreza moderada (proporción)'] = IM['Pobreza moderada (Miles de persona
s)'] / IM['Población total (2010)']
```

In [8]:
```
IM['Población con ingreso inferior a la línea de bienestar (proporción)'] = IM
['Población con ingreso inferior a la línea de bienestar'] / IM['Población tot
al (2010)']
IM['Población con ingreso inferior a la línea de bienestar mínimo (proporció
n)'] = IM['Población con ingreso inferior a la línea de bienestar mínimo'] / I
M['Población total (2010)']
```

In [9]:
```
IM['Rezago educativo (proporción)'] = IM['Rezago educativo\n(Miles de persona
s)'] / IM['Población total (2010)']
IM['Carencia por acceso a los servicios básicos en la vivienda (proporción)']
= IM['Carencia por acceso a los servicios básicos en la vivienda\n(Miles de pe
rsonas)'] / IM['Población total (2010)']
IM['Carencia por acceso a la alimentación (proporción)'] = IM['Carencia por ac
ceso a la alimentación\n(Miles de personas)'] / IM['Población total (2010)']
IM['Carencia por acceso a la seguridad social (proporción)'] = IM['Carencia po
r acceso a la seguridad social\n(Miles de personas)'] / IM['Población total (2
010)']
IM['Carencia por acceso a los servicios de salud (proporción)'] = IM['Carencia
 por acceso a los servicios de salud\n(Miles de personas)'] / IM['Población to
tal (2010)']
IM['Carencia por calidad y espacios de la vivienda (proporción)'] = IM['Carenc
ia por calidad y espacios de la vivienda\n(Miles de personas)'] / IM['Població
n total (2010)']
```

In [10]:
```
IM['Población de 15 años o más analfabeta (proporción)'] = IM['% de Población
 de 15 años o más analfabeta'] / 100
IM['Ocupantes en viviendas sin drenaje ni excusado (proporción)'] = IM['% Ocup
antes en viviendas sin drenaje ni excusado'] / 100
IM['Ocupantes en viviendas sin energía eléctrica (proporción)'] = IM['% Ocupan
tes en viviendas sin energía eléctrica'] / 100
IM['Ocupantes en viviendas sin agua entubada (proporción)'] = IM['% Ocupantes
 en viviendas sin agua entubada'] / 100
IM['Ocupantes en viviendas con piso de tierra (proporción)'] = IM['% Ocupantes
 en viviendas con piso de tierra'] / 100
```

In [11]:
```
IM['Mortalidad infantil (proporción)'] = IM['Tasa de mortalidad infantil'] / 1
00
IM['Población sin derechohabiencia (proporción)'] = IM['Sin derechohabiencia\n
(miles de personas)'] / IM['Población total (2010)']
IM['Población desocupada de 12 años y más (proporción)'] = IM['Población desoc
upada de 12 años y más\n(Miles de personas)'] / IM['Población total (2010)']
```

In [12]:
```
IM = IM.drop(IM.columns[[0,1,2,3,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21
,22,23,24,25,27]], axis=1)
```

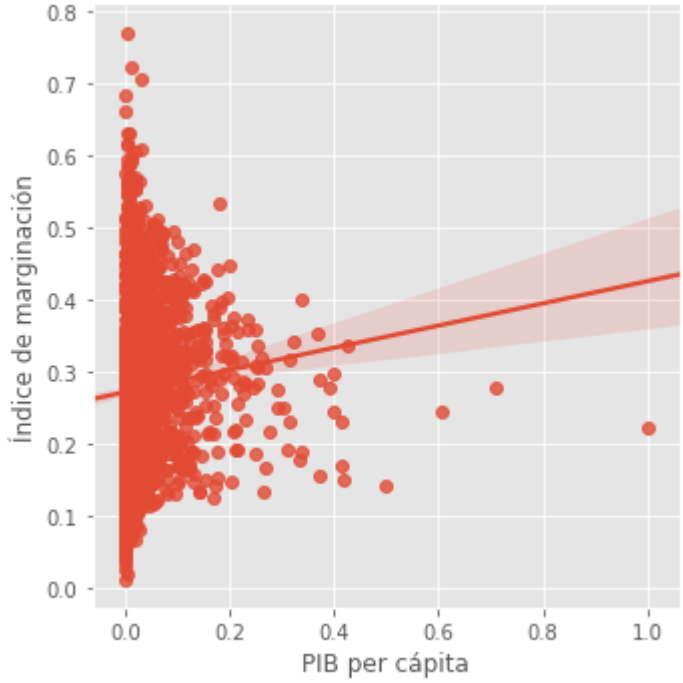
In [13]:
```
IM.info()
```

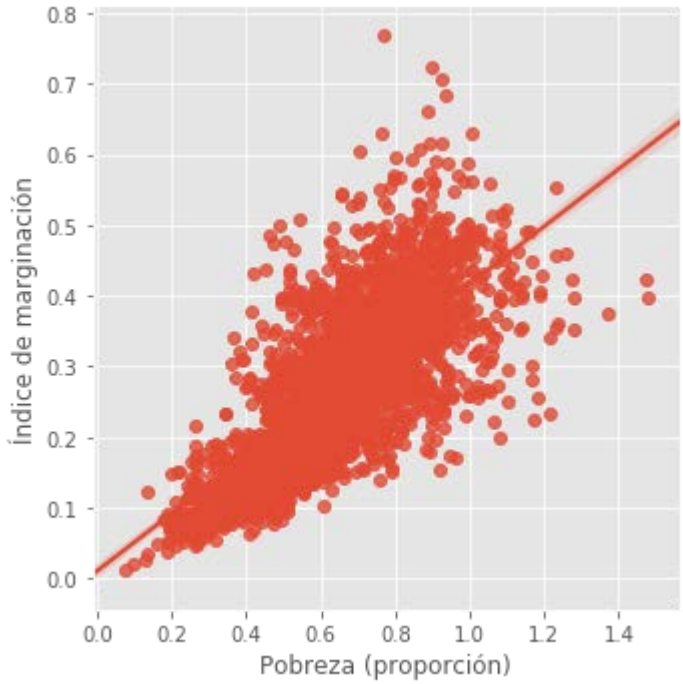
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2456 entries, 0 to 2455
Data columns (total 22 columns):
Grado de marginación
2456 non-null int64
Índice de marginación
2456 non-null float64
PIB per cápita
2456 non-null float64
Pobreza (proporción)
2456 non-null float64
Pobreza extrema (proporción)
2456 non-null float64
Pobreza moderada (proporción)
2456 non-null float64
Población con ingreso inferior a la línea de bienestar (proporción)
2456 non-null float64
Población con ingreso inferior a la línea de bienestar mínimo (proporción)
2456 non-null float64
Rezago educativo (proporción)
2456 non-null float64
Carencia por acceso a los servicios básicos en la vivienda (proporción)
2456 non-null float64
Carencia por acceso a la alimentación (proporción)
2456 non-null float64
Carencia por acceso a la seguridad social (proporción)
2456 non-null float64
Carencia por acceso a los servicios de salud (proporción)
2456 non-null float64
Carencia por calidad y espacios de la vivienda (proporción)
2456 non-null float64
Población de 15 años o más analfabeta (proporción)
2456 non-null float64
Ocupantes en viviendas sin drenaje ni excusado (proporción)
2456 non-null float64
Ocupantes en viviendas sin energía eléctrica (proporción)
2456 non-null float64
Ocupantes en viviendas sin agua entubada (proporción)
2456 non-null float64
Ocupantes en viviendas con piso de tierra (proporción)
2456 non-null float64
Mortalidad infantil (proporción)
2456 non-null float64
Población sin derechohabiencia (proporción)
2456 non-null float64
Población desocupada de 12 años y más (proporción)
2456 non-null float64
dtypes: float64(21), int64(1)
memory usage: 422.2 KB
```

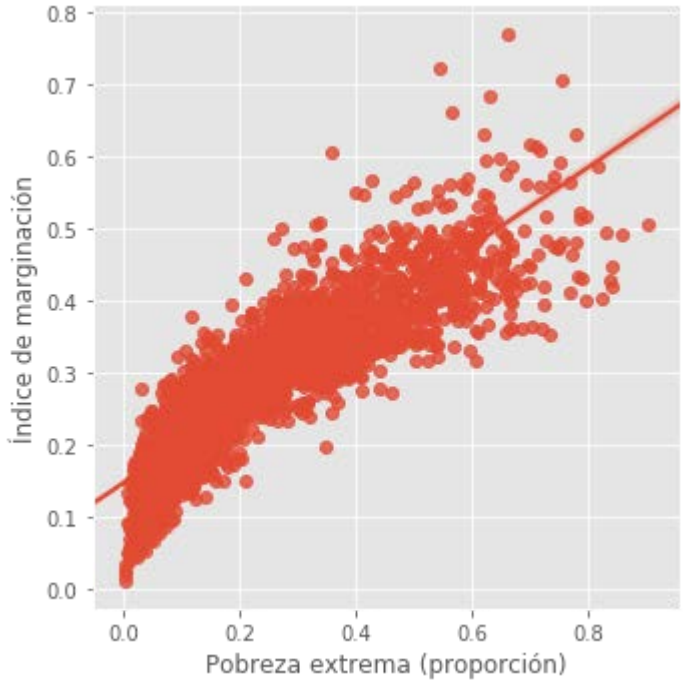
In [14]:
```
import seaborn as sns
```

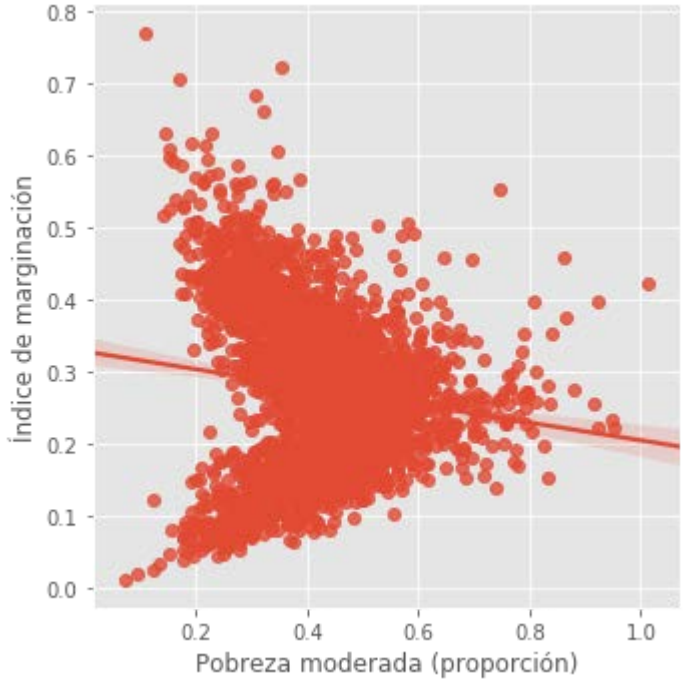
In [15]:
```
sns.lmplot(x="PIB per cápita", y="Índice de marginación", data=IM)
sns.lmplot(x="Pobreza (proporción)", y="Índice de marginación", data=IM)
sns.lmplot(x="Pobreza extrema (proporción)", y="Índice de marginación", data=I
M)
sns.lmplot(x="Pobreza moderada (proporción)", y="Índice de marginación", data=
IM)
sns.lmplot(x="Población con ingreso inferior a la línea de bienestar (proporci
ón)", y="Índice de marginación", data=IM)
sns.lmplot(x="Población con ingreso inferior a la línea de bienestar mínimo (p
roporción)", y="Índice de marginación", data=IM)
sns.lmplot(x="Rezago educativo (proporción)", y="Índice de marginación", data=
IM)
sns.lmplot(x="Carencia por acceso a los servicios básicos en la vivienda (prop
orción)", y="Índice de marginación", data=IM)
sns.lmplot(x="Carencia por acceso a la alimentación (proporción)", y="Índice d
e marginación", data=IM)
sns.lmplot(x="Carencia por acceso a la seguridad social (proporción)", y="Índi
ce de marginación", data=IM)
sns.lmplot(x="Carencia por acceso a los servicios de salud (proporción)", y="Í
ndice de marginación", data=IM)
sns.lmplot(x="Carencia por calidad y espacios de la vivienda (proporción)", y=
"Índice de marginación", data=IM)
sns.lmplot(x="Población de 15 años o más analfabeta (proporción)", y="Índice d
e marginación", data=IM)
sns.lmplot(x="Ocupantes en viviendas sin drenaje ni excusado (proporción)", y=
"Índice de marginación", data=IM)
sns.lmplot(x="Ocupantes en viviendas sin energía eléctrica (proporción)", y="Í
ndice de marginación", data=IM)
sns.lmplot(x="Ocupantes en viviendas sin agua entubada (proporción)", y="Índic
e de marginación", data=IM)
sns.lmplot(x="Ocupantes en viviendas con piso de tierra (proporción)", y="Índi
ce de marginación", data=IM)
sns.lmplot(x="Mortalidad infantil (proporción)", y="Índice de marginación", da
ta=IM)
sns.lmplot(x="Población sin derechohabiencia (proporción)", y="Índice de margi
nación", data=IM)
sns.lmplot(x="Población desocupada de 12 años y más (proporción)", y="Índice d
e marginación", data=IM)
```

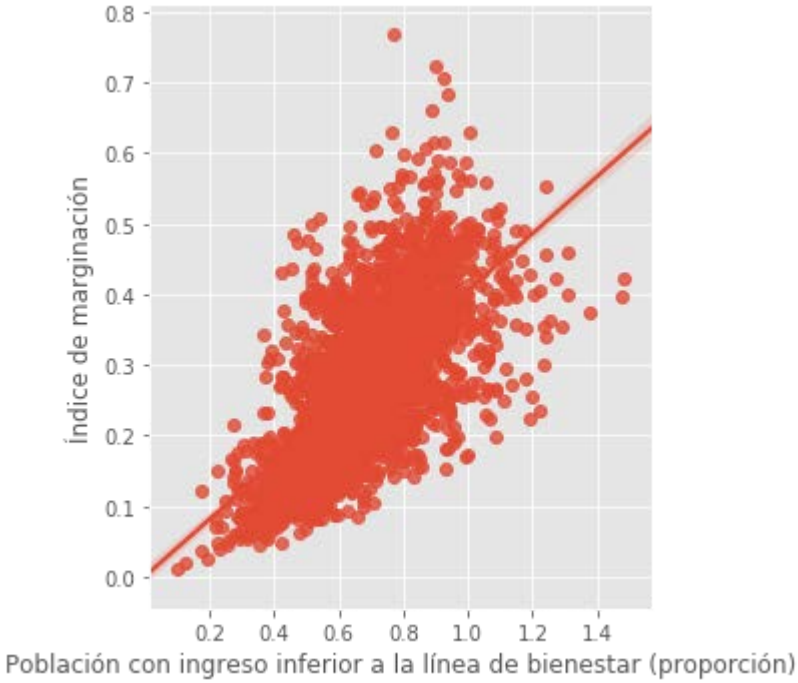
```
Out[15]: <seaborn.axisgrid.FacetGrid at 0x20b22ed8860>
```

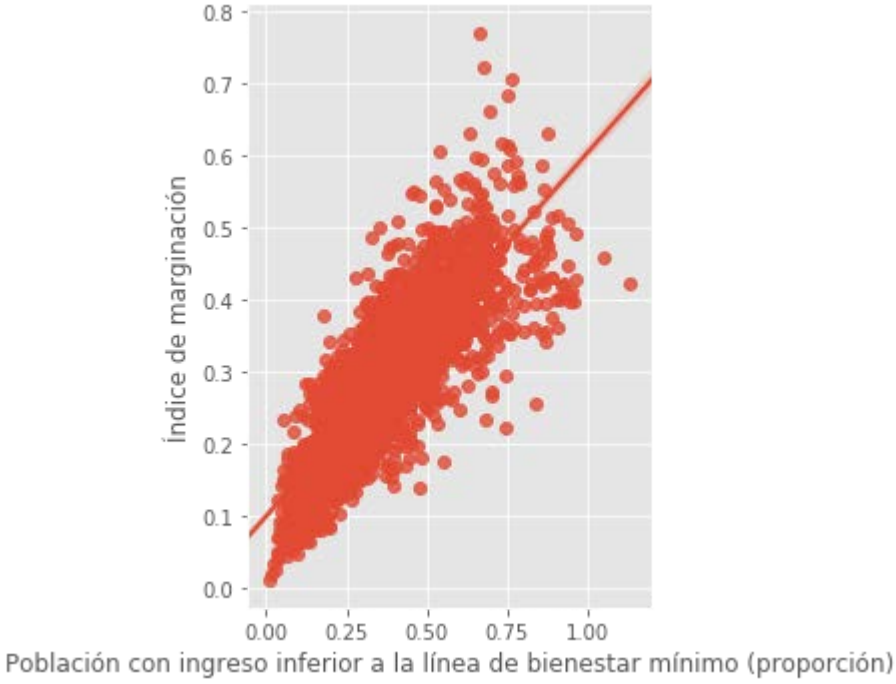


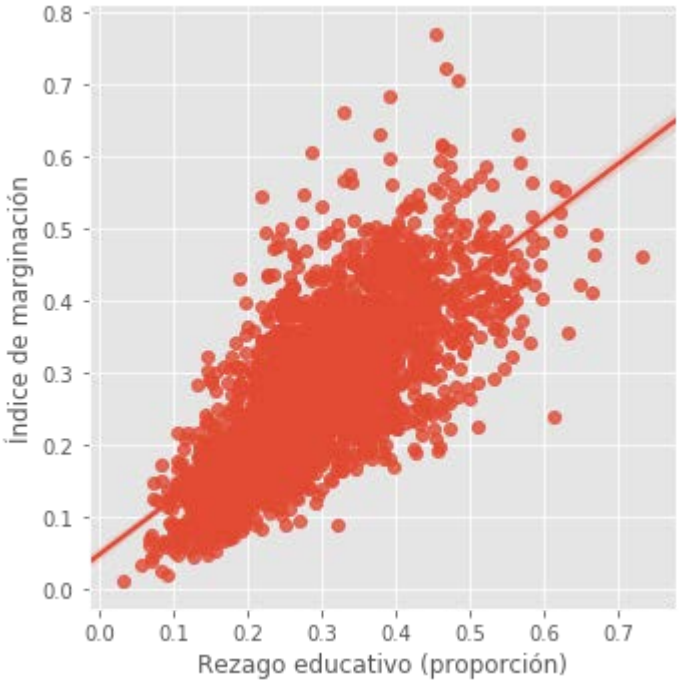


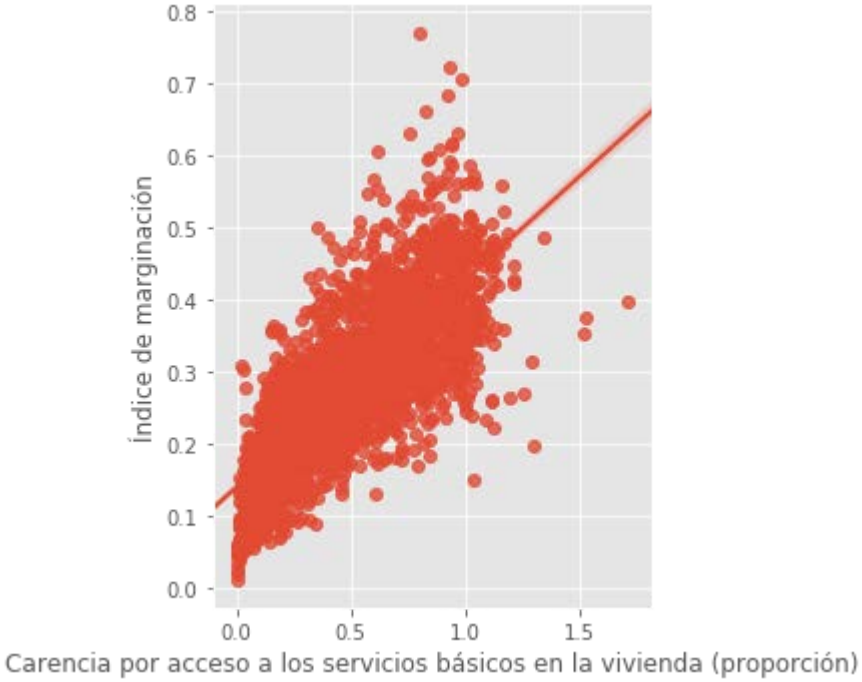

Índice de marginación
Pobreza extrema (proporción)
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.0
0.2
0.4
0.6
0.8

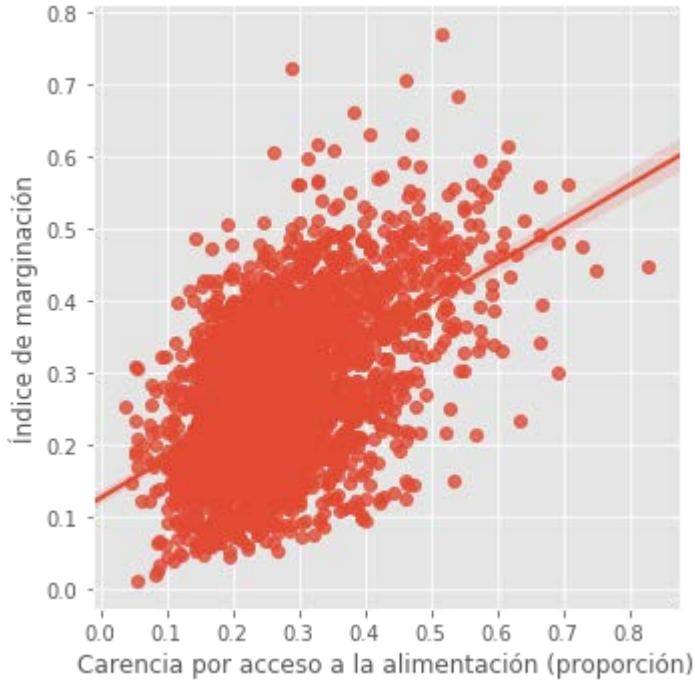

Índice de marginación
Pobreza moderada (proporción)
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.2
0.4
0.6
0.8
1.0

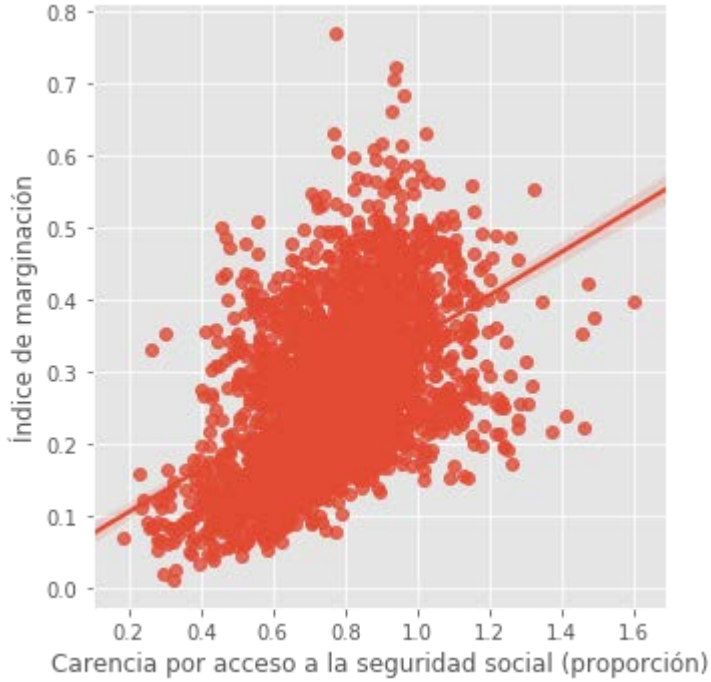

Índice de marginación
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.2
0.4
0.6
0.8
1.0
1.2
1.4
Población con ingreso inferior a la línea de bienestar (proporción)

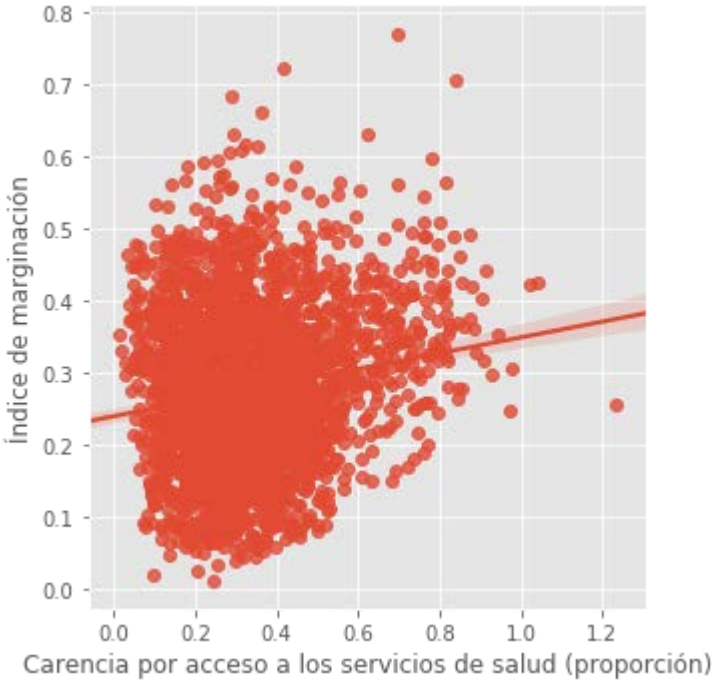

Índice de marginación
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.00
0.25
0.50
0.75
1.00
Población con ingreso inferior a la línea de bienestar mínimo (proporción)

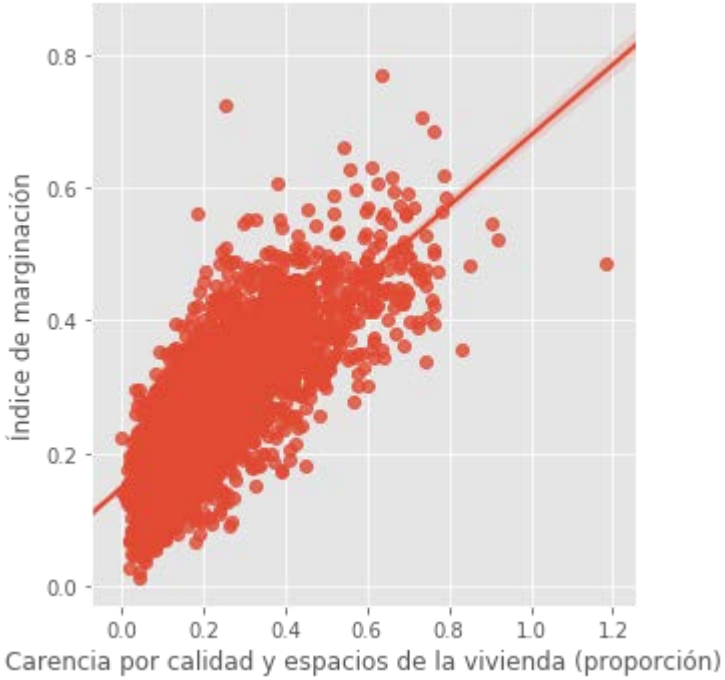

Índice de marginación
Rezago educativo (proporción)
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8

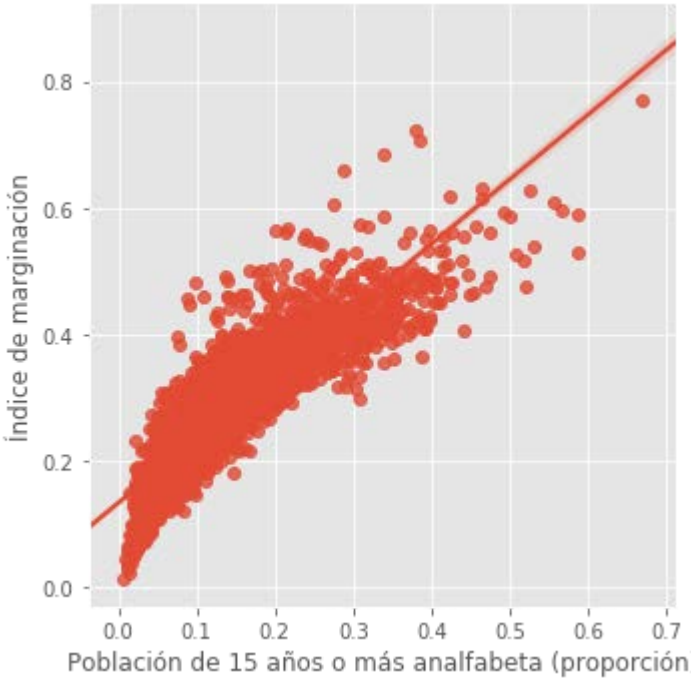

Índice de marginación
Carencia por acceso a los servicios básicos en la vivienda (proporción)
0.0
0.5
1.0
1.5
0.1
0.2
0.3
0.4
0.6
0.7
0.8

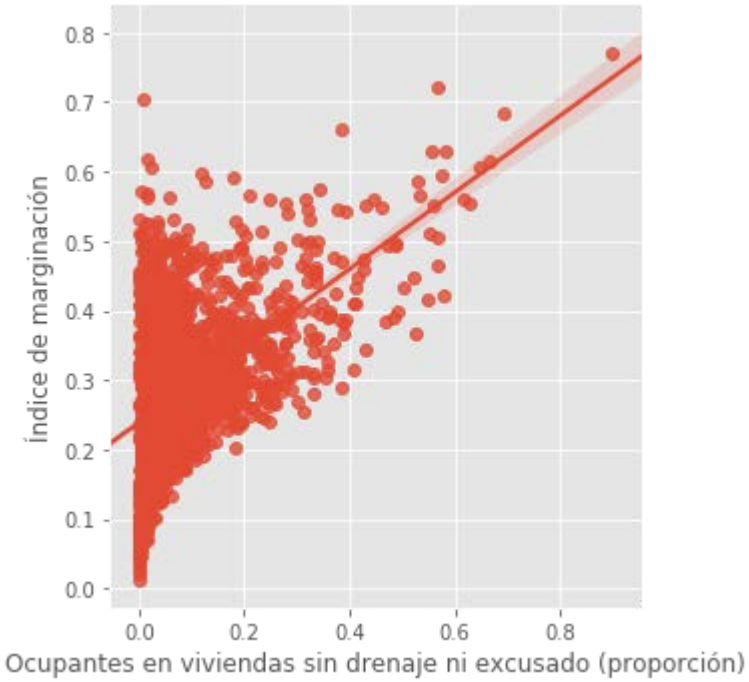

Índice de marginación
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
Carencia por acceso a la alimentación (proporción)

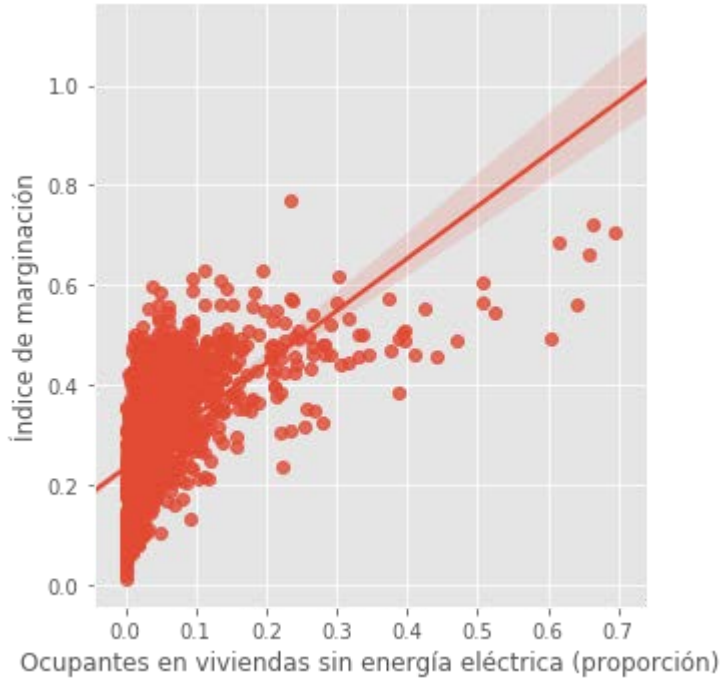

Índice de marginación
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
Carencia por acceso a la seguridad social (proporción)

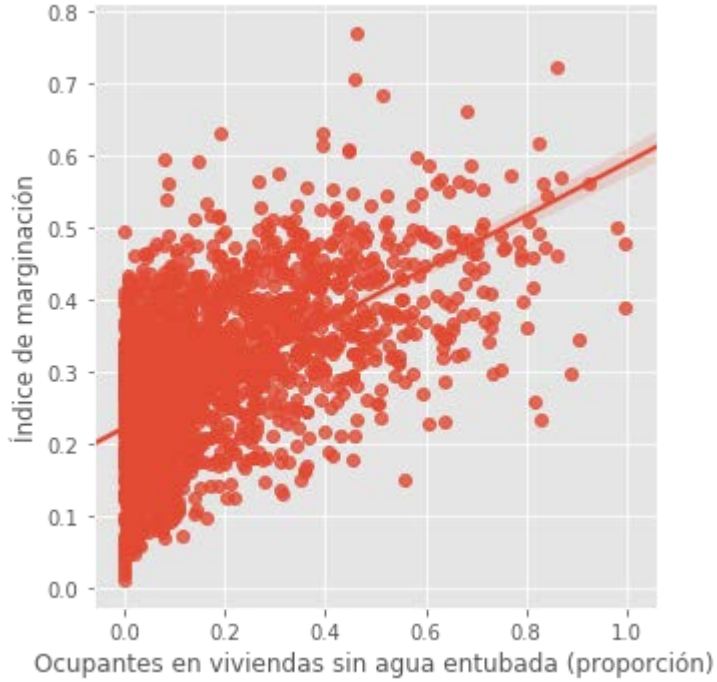

Índice de marginación
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.0
0.2
0.4
0.6
0.8
1.0
1.2
Carencia por acceso a los servicios de salud (proporción)

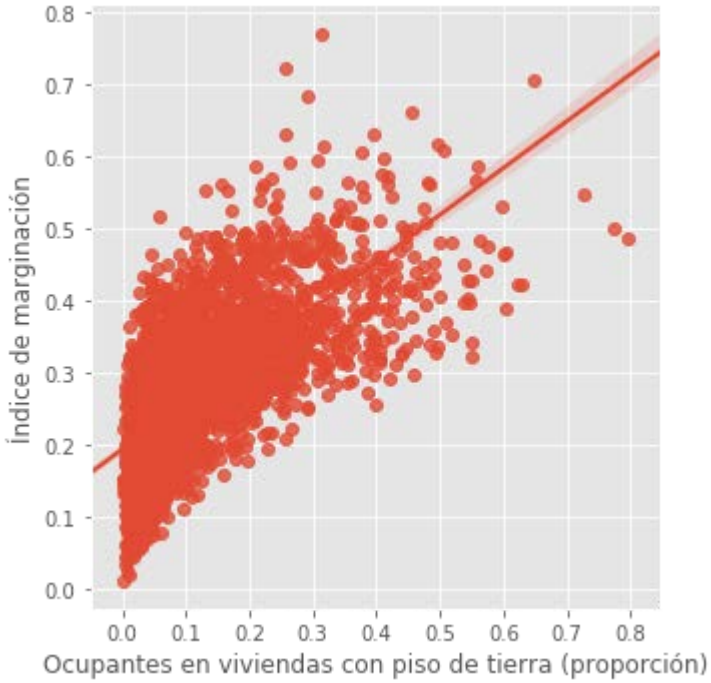

Índice de marginación
0.0
0.2
0.4
0.6
0.8
0.0
0.2
0.4
0.6
0.8
1.0
1.2
Carencia por calidad y espacios de la vivienda (proporción)

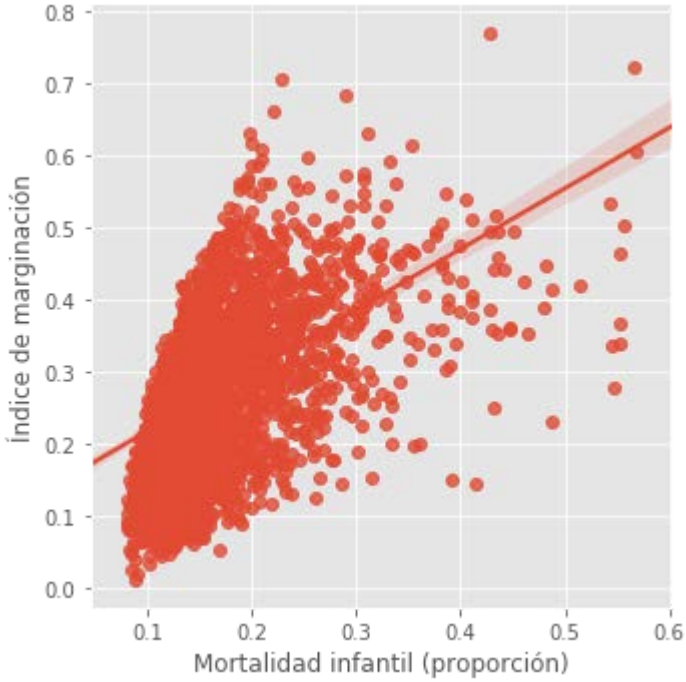

Índice de marginación
0.0
0.2
0.4
0.6
0.8
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Población de 15 años o más analfabeta (proporción)

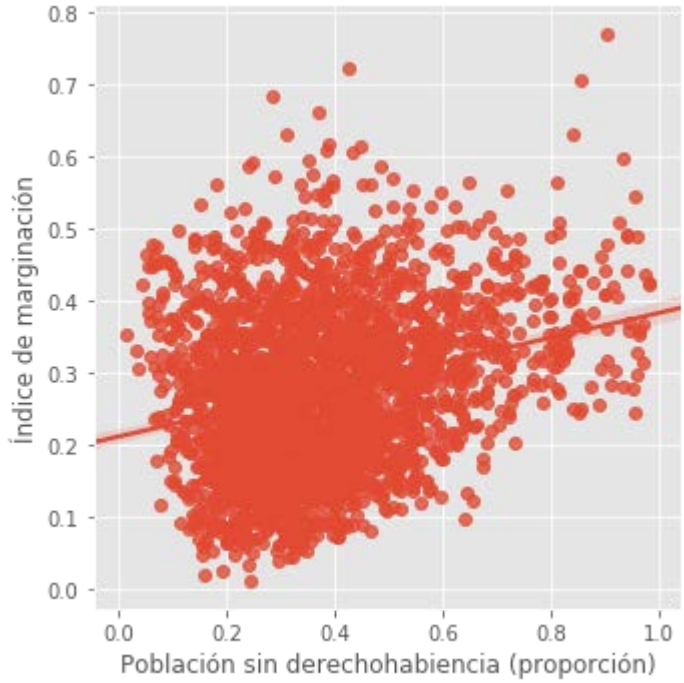

Índice de marginación
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.0
0.2
0.4
0.6
0.8
Ocupantes en viviendas sin drenaje ni excusado (proporción)

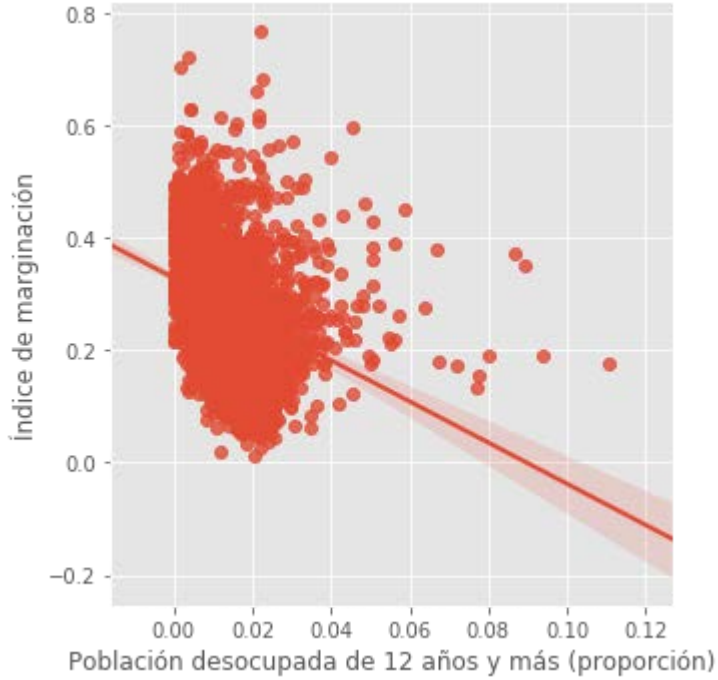

Índice de marginación
1.0
0.8
0.6
0.4
0.2
0.0
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7
Ocupantes en viviendas sin energía eléctrica (proporción)


Índice de marginación
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.0 0.2 0.4 0.6 0.8 1.0
Ocupantes en viviendas sin agua entubada (proporción)


Índice de marginación
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
Ocupantes en viviendas con piso de tierra (proporción)


Índice de marginación
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
Mortalidad infantil (proporción)





In [16]:
```
Y = IM[['Índice de marginación']]
```

In [17]:
```
IM[['Índice de marginación']].plot(alpha=0.5,figsize=(20,6))
```

Out[17]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x20b232bf898>
```