TP Modelado: Parte 2

Grupo 12: Barragán, Rossi, Fontana Walser

2022-10-25

Carga de Librerias, Lectura del dataset y su estructura

```
require(tidyverse) # entorno tidy
```

```
## Loading required package: tidyverse
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──
## ✔ ggplot2 3.3.6     ✔ purrr   0.3.4
## ✔ tibble  3.1.8     ✔ dplyr   1.0.9
## ✔ tidyr   1.2.0     ✔ stringr 1.4.0
## ✔ readr   2.1.2     ✔ forcats 0.5.1
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

```
require(dplyr) # manejo de datos
require(GGally) # scatterplots multiples
```

```
## Loading required package: GGally
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
require(rje) # función powerSet
```

```
## Loading required package: rje
##
## Attaching package: 'rje'
##
## The following object is masked from 'package:dplyr':
##
##     last
```

```
dataset <- read_csv("datos_alquiler.csv")
```

```
## Rows: 997 Columns: 14
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr  (7): id, ad_type, l3, currency, price_period, property_type, operation_...
## dbl  (6): lat, lon, rooms, surface_total, surface_covered, price
## date (1): created_on
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Aplicando filtros

Nos quedamos con las variables de interés y filtramos los datos faltantes

```
datos <- dataset %>%
  mutate(fondo=surface_total-surface_covered) %>%
  select(property_type, surface_covered, fondo, rooms, price, created_on, lat, lon) %>%
  # filter(!is.na(rooms), !is.na(lat), !is.na(lon)) %>%
  # filter(fondo>=0) %>%
  arrange(created_on = as.Date(created_on, "%d-%m-%Y",origin = "2018-12-29")) %>%
  mutate(dia_semana = weekdays(created_on), mes = months(created_on)) # Nos va a ser útil para analizar las fecha
s de publicación
head(datos)
```

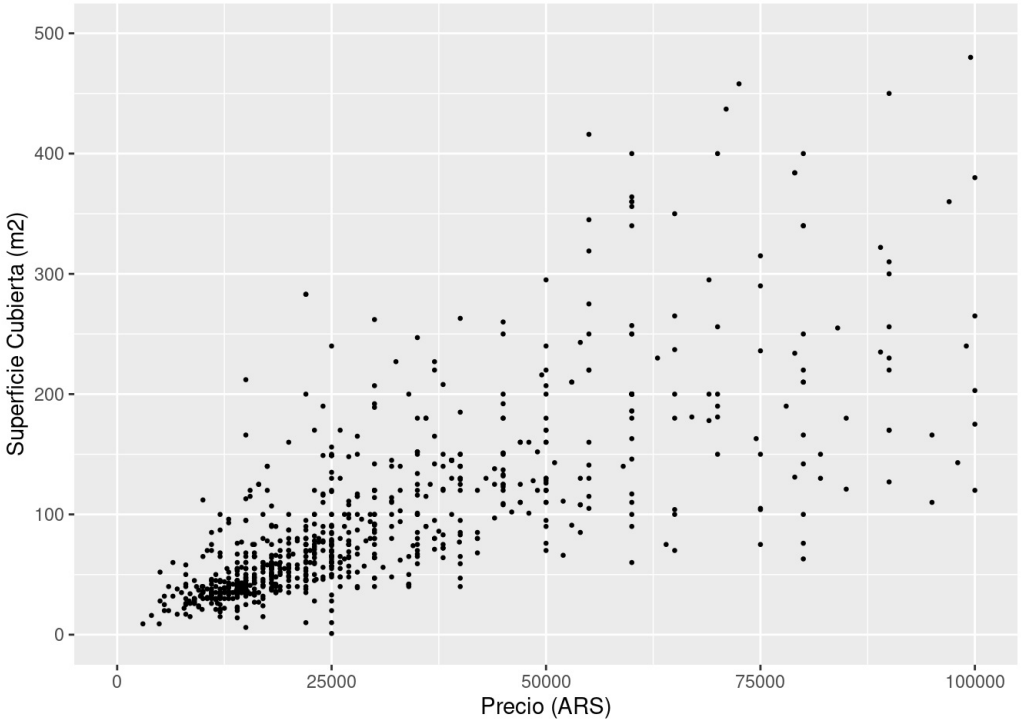
```
## # A tibble: 6 × 10
##   property_type   surfa…¹ fondo rooms price created_on   lat   lon dia_s…² mes
##   <chr>             <dbl> <dbl> <dbl> <dbl> <date>     <dbl> <dbl> <chr>   <chr>
## 1 Oficina             170     0    NA 90000 2018-12-29 -34.6 -58.4 sábado  dici…
## 2 PH                   38     0    NA 11000 2018-12-29 -34.6 -58.4 sábado  dici…
## 3 PH                   40     0     2 22000 2018-12-29 -34.6 -58.4 sábado  dici…
## 4 PH                   42    NA     3 18300 2018-12-29 -34.6 -58.5 sábado  dici…
## 5 Local comercial     100     0    NA 27000 2018-12-30 -34.6 -58.5 domingo dici…
## 6 Local comercial      68     0    NA 12000 2018-12-31 -34.6 -58.4 lunes   dici…
## # … with abbreviated variable names ¹surface_covered, ²dia_semana
```

1 - Análisis exploratorio de datos

Observemos cómo se distribuyen los valores de las variables que nos interesan. Veamos primero la relación entre el precio y superficie cubierta.

```
# Precio y superficie cubierta
datos1 <- datos %>% filter(!is.na(surface_covered))
ggplot(data = datos1,
  mapping = aes(
  x = price,
  y = surface_covered)) +
  geom_point(size = 0.5)+
  xlim(0,1e+05)+
  ylim(0,500) + xlab("Precio (ARS)") + ylab("Superficie Cubierta (m2)")
```

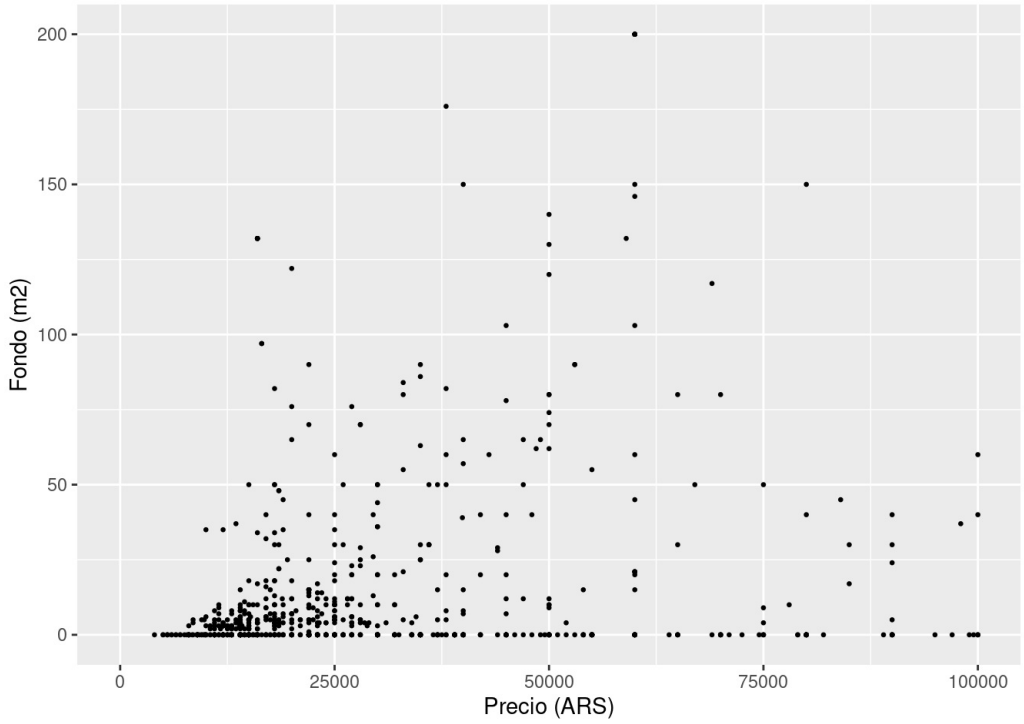
```
## Warning: Removed 89 rows containing missing values (geom_point).
```



Ahora miremos la relación entre el precio y el fondo de las propiedades.

```
# Precio y fondo
datos2 <- datos %>% filter(!is.na(fondo))
ggplot(data = datos2,
  mapping = aes(
  x = price,
  y = fondo)) +
  geom_point(size = 0.5)+
  xlim(0,1e+05)+
  ylim(0,200)  + xlab("Precio (ARS)") + ylab("Fondo (m2)")
```

```
## Warning: Removed 94 rows containing missing values (geom_point).
```

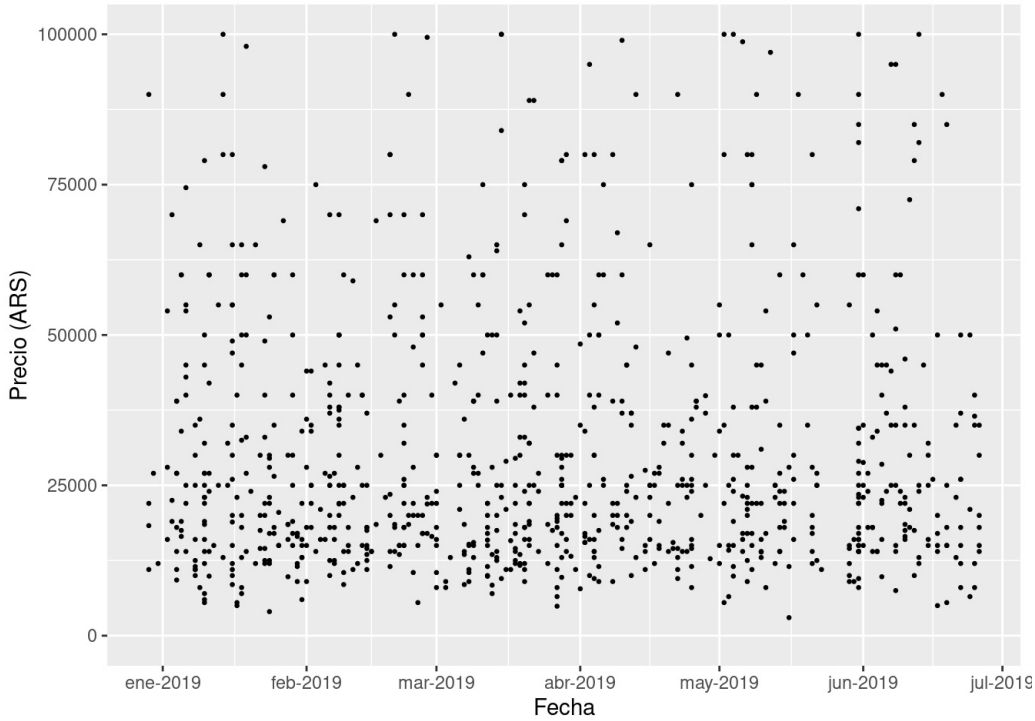


Podemos ver una relacion linealmente creciente entre el precio y estas dos variables, con la particularidad de que hay propiedades cuyo precio varía incluso sin poseer fondo. Esto se puede verificar en la linea horizontal inferior que se forma en el gráfico precio-fondo.

Veamos ahora la relacion entre el precio y la fecha de publicación de la propiedad.

```
#precio y fecha publicaión
datos3 <- datos %>% filter(!is.na(price))
ggplot(data = datos3,
  mapping = aes(
  y = price,
  x = created_on, ylab = "Precio (ARS)", xlab = "Fecha")) +
  geom_point(size = 0.5) +
  ylim(0,1e+05)  +
  scale_x_date(date_breaks = "1 month", date_labels = "%b-%Y") +

 xlab("Fecha") + ylab("Precio (ARS)")
```

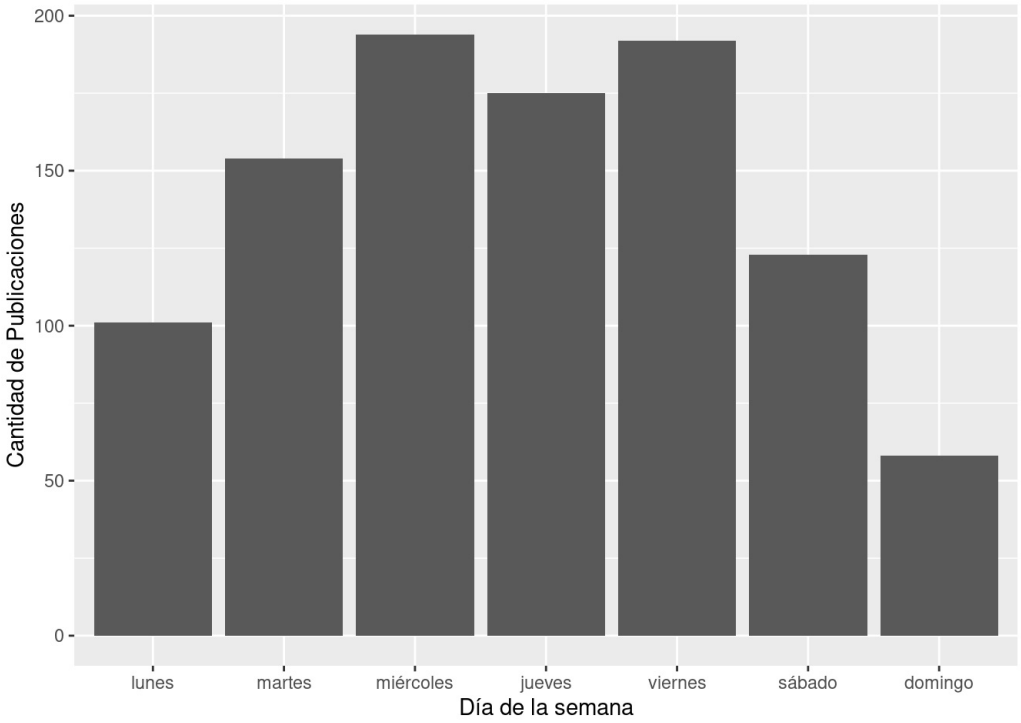
```
## Warning: Removed 95 rows containing missing values (geom_point).
```



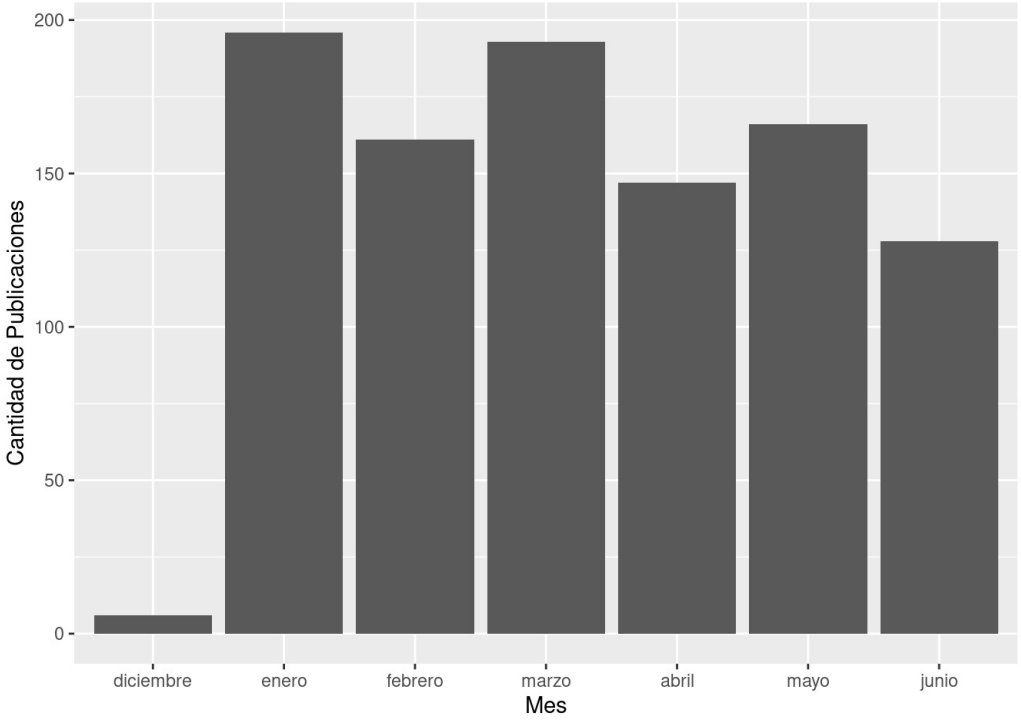
Podemos ver que las publicaciones de este dataset van desde diciembre de 2018 a junio de 2019.

Analicemos la cantidad de publicaciones según el día de la semana y el mes.

```
datos$dia_semana <- factor(datos$dia_semana,
                  levels = c("lunes", "martes", "miércoles", "jueves", "viernes","sábado","domingo"))
datos$mes <- factor(datos$mes, levels = c("diciembre", "enero", "febrero", "marzo", "abril", "mayo", "junio"))

datos %>%

  ggplot(aes(x = dia_semana)) +
    geom_bar() +
    xlab("Día de la semana") +
    ylab("Cantidad de Publicaciones")
```




```
datos %>%
  ggplot(aes(x = mes)) +
    geom_bar() +
    xlab("Mes") +
    ylab("Cantidad de Publicaciones")
```



Como se puede apreciar, la publicación de propiedades es mayor a mitad de semana, entre los días miércoles y viernes, y ha ido decreciendo a lo largo de los meses.

2 - Modelo constante del precio

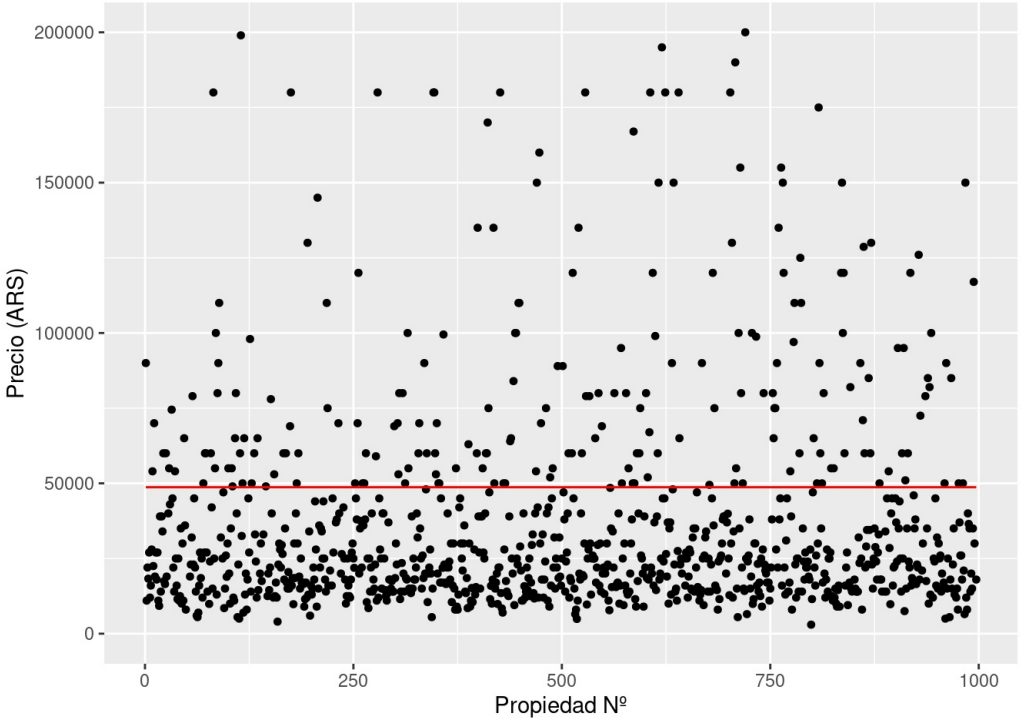
Ajustamos un modelo constante con el promedio de todos los precios.

```
ajusM0<-mean(datos3$price) # usamos dataset sin datos faltantes de precio
ajusM0
```

```
## [1] 48702.45
```

```
ggplot(datos3, aes(y=price, x = seq(1,length(price)))) +
  geom_point(size=2, shape=20) +
  ylim(0,200000) +
  ylab("Precio (ARS)") +
  xlab("Propiedad Nº") + # Revisar nombre del eje x, no queda claro
  geom_line(color= "red", aes(x=seq(1, length(price)), y= mean(price)))
```

```
## Warning: Removed 42 rows containing missing values (geom_point).
```



Claramente, este es el peor modelo ya que solo toma en cuenta el precio (la misma variable que intentamos entender) y no utiliza ninguna otra variable predictora. La recta representa el promedio del precio de las propiedades publicadas en el sitio de Properati.

3 - Modelo lineal del precio en función de la superficie cubierta

Comencemos ajustando el modelo.

```
datos4 <- datos %>% filter(!is.na(surface_covered), !is.na(fondo), !is.na(price))

# Ajuste
ajusM1<-lm(price~surface_covered,data=datos4) # modelo lineal
coeM1 <-coef(ajusM1) # coeficientes
coeM1
```

```
##     (Intercept) surface_covered
##      10819.5865        265.2695
```

Para determinar cuán bien ajusta el modelo a los datos procedemos a medir el error por validación cruzada.

```
n<-length(datos4$price)
predichos.oos<-rep(NA,n) # predichos out of sample

for (i in 1:n)
{
  ajus.cv<-lm(price~surface_covered,data=datos4[-i,])
  predichos.oos[i]<-predict(ajus.cv,newdata=datos4[i,])
}
# MAE
mean(abs(datos4$price-predichos.oos))
```

```
## [1] 19448.48
```

```
# PMAE
pmaeM2.cv<-mean(abs(datos4$price-predichos.oos))/mean(datos4$price)
pmaeM2.cv
```

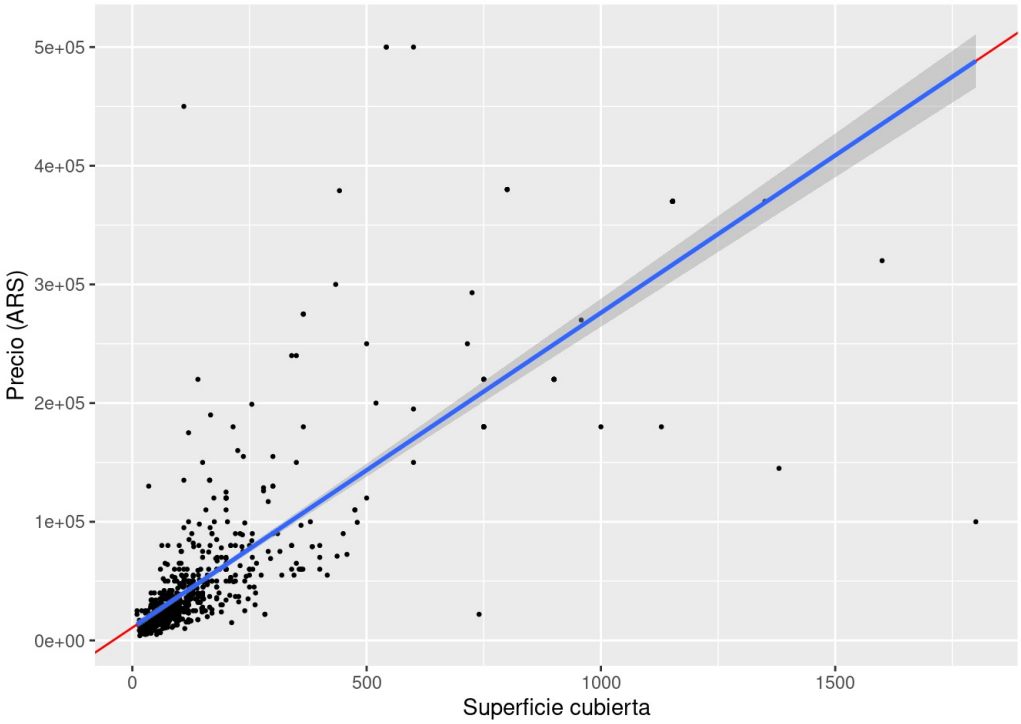
```
## [1] 0.3909503
```

Podemos ver que el MAE es cercano al 40% del precio promedio de todas las propiedades, y esto nos da una medida del error que estamos cometiendo con este modelo.

Veamos el gráfico del precio en función de la superficie cubierta incluyendo el modelo (lineal).

```
predicted_df <- data.frame(mpg_pred = predict(ajusM1, data=datos4$surface_covered), hp=datos4$price)
datos4 %>%
  ggplot(aes(x = surface_covered, y = price))  +
  geom_point(size = 0.5) +
  geom_abline(color="red", slope = coef(ajusM1)[2], intercept = coef(ajusM1)[1]) +
  geom_smooth(method="lm") +
  xlab("Superficie cubierta") +
  ylab("Precio (ARS)")
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
summary(ajusM1)
```

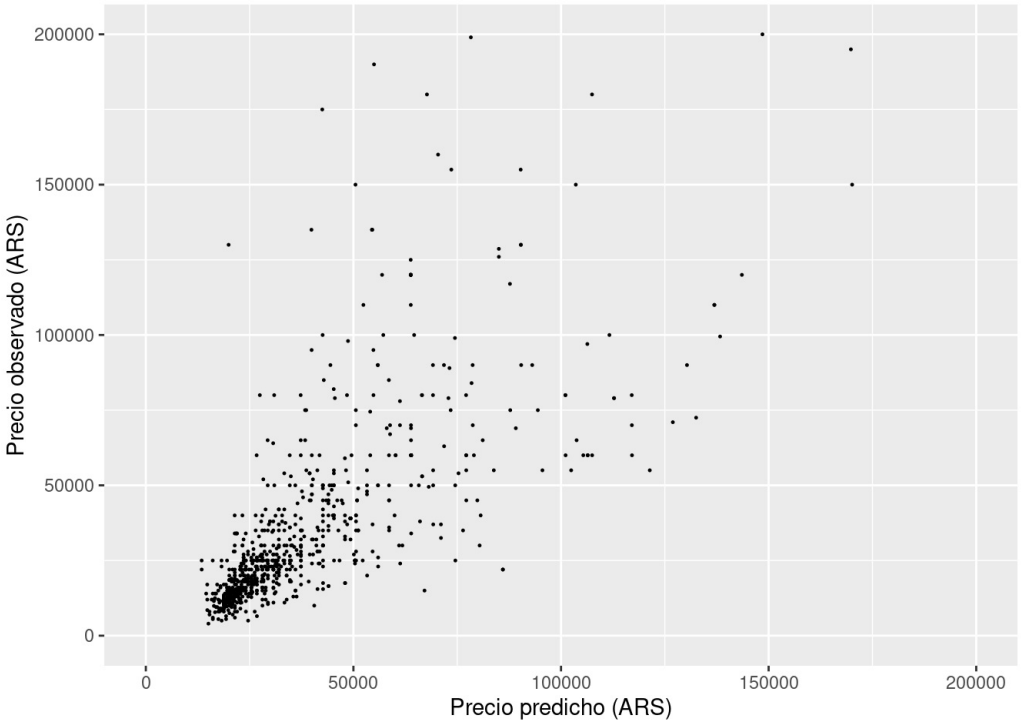
```
##
## Call:
## lm(formula = price ~ surface_covered, data = datos4)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -388305  -10762   -6288    1331  410001
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     10819.586   1748.517   6.188 9.38e-10 ***
## surface_covered   265.270      6.865  38.639  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 42200 on 870 degrees of freedom
## Multiple R-squared:  0.6318, Adjusted R-squared:  0.6314
## F-statistic:  1493 on 1 and 870 DF,  p-value: < 2.2e-16
```

La pendiente del ajuste representa el incremento del precio por metro cuadrado de superficie cubierta de una propiedad. Dado que la pendiente es positiva podemos ver que a mayor superficie cubierta, mayor es el precio de la propiedad en cuestión. En particular, al aumentar un metro cuadrado la superficie cubierta, el ajuste aumenta el precio en 265.27 ARS.

Ahora grafiquemos los valores predichos en función de los valores observados.

```
datos5 <- datos4 %>% mutate(predichos = predichos.oos)
ggplot(data = datos5, mapping = aes(x = predichos, y = price)) +
  geom_point(size = 0.2) +
  xlab("Precio predicho (ARS)") +
  ylab("Precio observado (ARS)") +
  xlim(0,2e+05) + # Nos sacamos de encima los outliers
  ylim(0,2e+05)
```

```
## Warning: Removed 52 rows containing missing values (geom_point).
```



El desempeño es mejor en el rango del grueso de los datos, esto lo vemos en que la variabilidad en los datos cerca del ajuste que nos indica geom_smooth es mínima allí, y en que en este último gráfico, el conjunto más denso de puntos parece comportarse más linealmente que en otras zonas. Fuera de dicho rango, los datos se dispersan mucho.

4 - Evolución del precio en función del tiempo y el impacto de la inflación

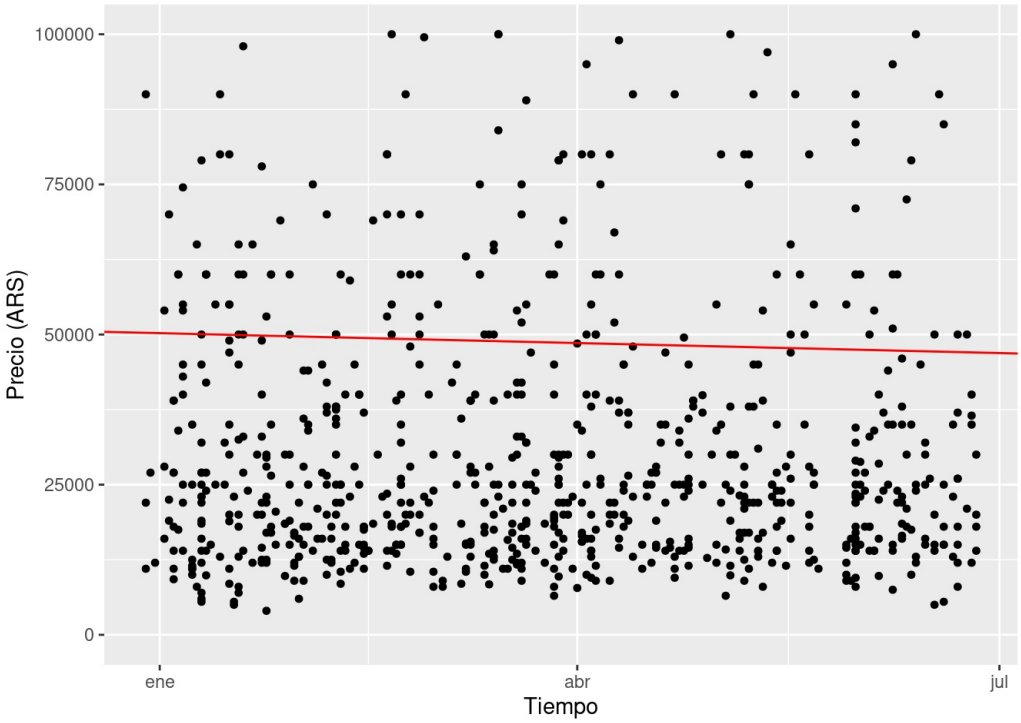
Miremos la evolución del precio con el paso del tiempo en un gráfico.

```
# Creamos un dataset limpiando los datos faltantes del precio y la fecha
datos6 <- datos %>% filter(!is.na(price), !is.na(created_on))

# Calculamos el modelo y graficamos
ajusM1_fecha <- lm(price~created_on, data = datos6) # modelo lineal
coeM1 <- coef(ajusM1_fecha) # coeficientes
coeM1
```

```
##  (Intercept)   created_on
## 377737.81292    -18.29993
```

```
datos4 %>%
  ggplot(aes(x = created_on, y = price)) +
  geom_point(size=2, shape=20) +
  ylab("Precio (ARS)") +
  xlab("Tiempo") +
  ylim(0,100000) +
  geom_abline(color = "red", slope = coeM1[2], intercept = coeM1[1])
```

```
## Warning: Removed 86 rows containing missing values (geom_point).
```



Al poner todas las propiedades en este gráfico no podemos ver la evolución del precio con claridad. Pensemos que estamos poniendo el valor de una casa y el de una cochera como si fueran iguales cuando claramente no lo son.

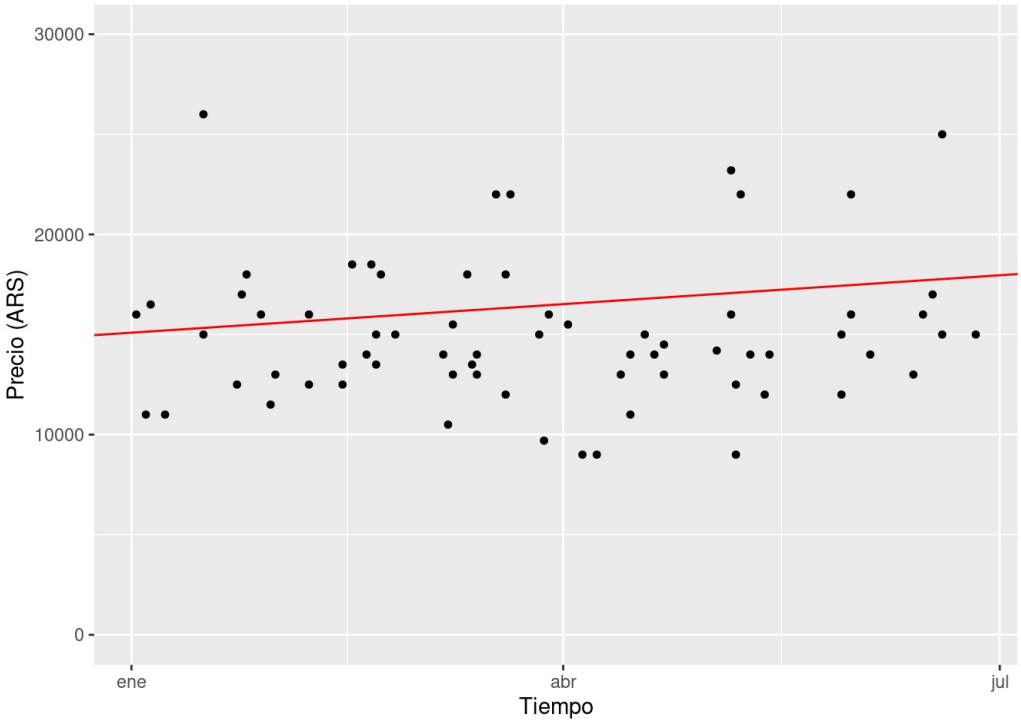
Veamos qué pasa si nos concentramos en el precio de los departamentos de 2 ambientes (rooms) en función del tiempo.

```
# Creamos un dataset donde nos quedamos con los departamentos de 2 ambientes y limpiamos los datos faltantes
datos7 <- datos %>% filter(property_type == "Departamento", rooms == 2, !is.na(price), !is.na(created_on))

# Calculamos el modelo y graficamos
ajusM1_fecha2 <- lm(price~created_on, data = datos7) # modelo lineal
coeM1 <- coef(ajusM1_fecha2) # coeficientes
coeM1
```

```
##   (Intercept)    created_on
## -268524.97404      15.84724
```

```
datos7 %>%
  ggplot(aes(x = created_on, y = price)) +
  geom_point(size=2, shape=20) +
  ylab("Precio (ARS)") +
  xlab("Tiempo") +
  ylim(0,30000) +
  geom_abline(color = "red", slope = coeM1[2], intercept = coeM1[1])
```

```
## Warning: Removed 5 rows containing missing values (geom_point).
```



Acá podemos ver el impacto de la inflación que produce el aumento del precio a medida que pasa el tiempo. Probablemente si tuviesemos más datos podríamos ver una suba más pronunciada.

5 - Precio y tipo de propiedad

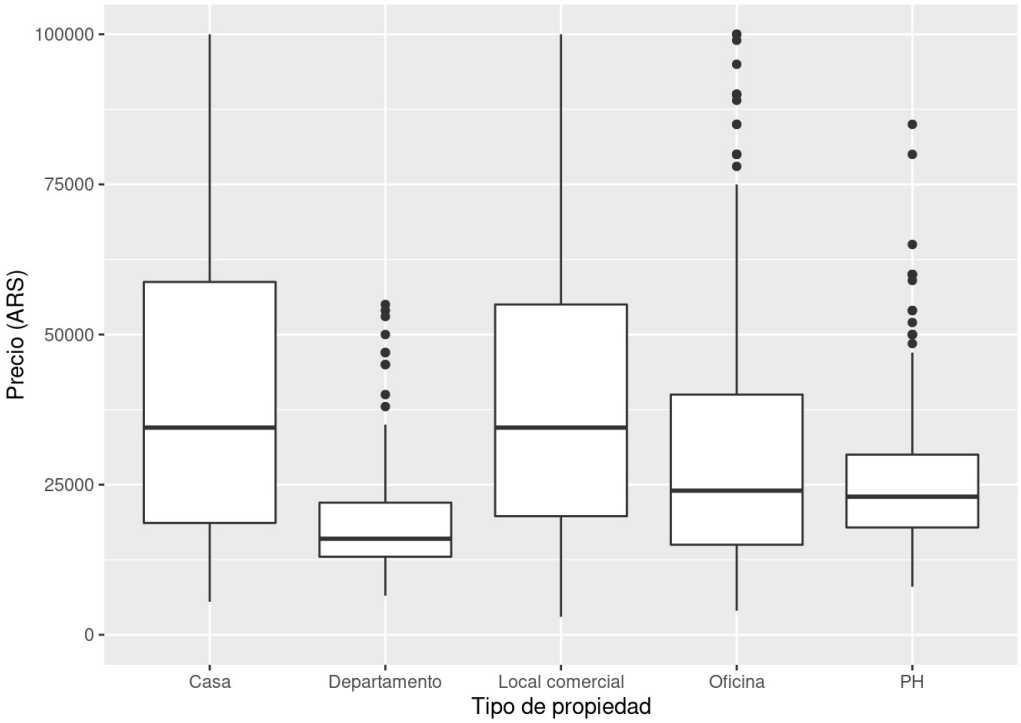
Al realizar un modelo lineal con variables categóricas debemos tener en cuenta que sólo una de ellas puede valer 1, las demás valdrán 0. Esto es porque no podría tener una propiedad que sea un PH y un departamento al mismo tiempo.

```
# Creamos un dataset sin los faltantes del precio y tipo de propiedad
datos8 <- datos %>% filter(!is.na(price), !is.na(property_type))

ajusM1_proptype <- lm(price~property_type, data = datos8)
coeM1_proptype <- coef(ajusM1_proptype)
coeM1_proptype
```

```
##                 (Intercept)    property_typeDepartamento
##                    84026.22                   -64838.47
## property_typeLocal comercial         property_typeOficina
##                    -11940.72                   -41579.47
##              property_typePH
##                    -57730.35
```

```
# Boxplot de los datos
datos8 %>%
  ggplot(aes(x=property_type, y=price)) +
  ylim(0,100000) +
  xlab("Tipo de propiedad") +
  ylab("Precio (ARS)") +
  geom_boxplot()
```

```
## Warning: Removed 95 rows containing non-finite values (stat_boxplot).
```



Ahora bien, ¿qué representa el intercept? ¿Por qué no tengo el coeficiente correspondiente a “Casa”? Si asumimos que todas nuestras variables valen 0, es decir la propiedad no es ni un departamento, local comercial, oficina ni PH, entonces el intercept de alguna manera representa el valor estimativo de la variable Casa. Así, cuando cada una de las demás variables toma el valor de 1, por ejemplo la variable Oficina, estamos comparando el valor estimativo de las Oficinas con el de la variable Casa. Con lo cual, el intercept se utiliza como valor de referencia.

Por ejemplo, podemos ver que todas las propiedades valen menos que una casa, porque el coeficiente asociado a cada una del resto de las propiedades es negativo. De hecho, podemos incluso ver que el local comercial parece ser el tipo de propiedad que más se parece en precio a una casa, pues su coeficiente es el de menor módulo. Por otro lado, el tipo de propiedad que menos se parece en precio a una casa es un departamento, razonando análogamente.

Ahora ajustemos dos modelos distintos del precio en función de la superficie cubierta y el tipo de propiedad.

```
# Creamos un dataset sin los faltantes del precio, tipo de propiedad y superficie cubierta
datos9 <- datos %>% filter(!is.na(price), !is.na(property_type), !is.na(surface_covered))

#sup + prop type
ajusM2_proptype_surface <- lm(price~property_type+surface_covered, data = datos9)
coeM2_proptype_surface <- coef(ajusM2_proptype_surface)
print("Modelo price~property_type+surface_covered")
```

```
## [1] "Modelo price~property_type+surface_covered"
```

```
coeM2_proptype_surface
```

```
##                  (Intercept)    property_typeDepartamento
##                    13925.6266                   -8605.0049
## property_typeLocal comercial         property_typeOficina
##                    14757.9873                   -8723.0395
##               property_typePH              surface_covered
##                    -8703.2363                    257.7406
```

```
#esto es para obtener los 5 primeros colores default de ggplot
library(scales)
```

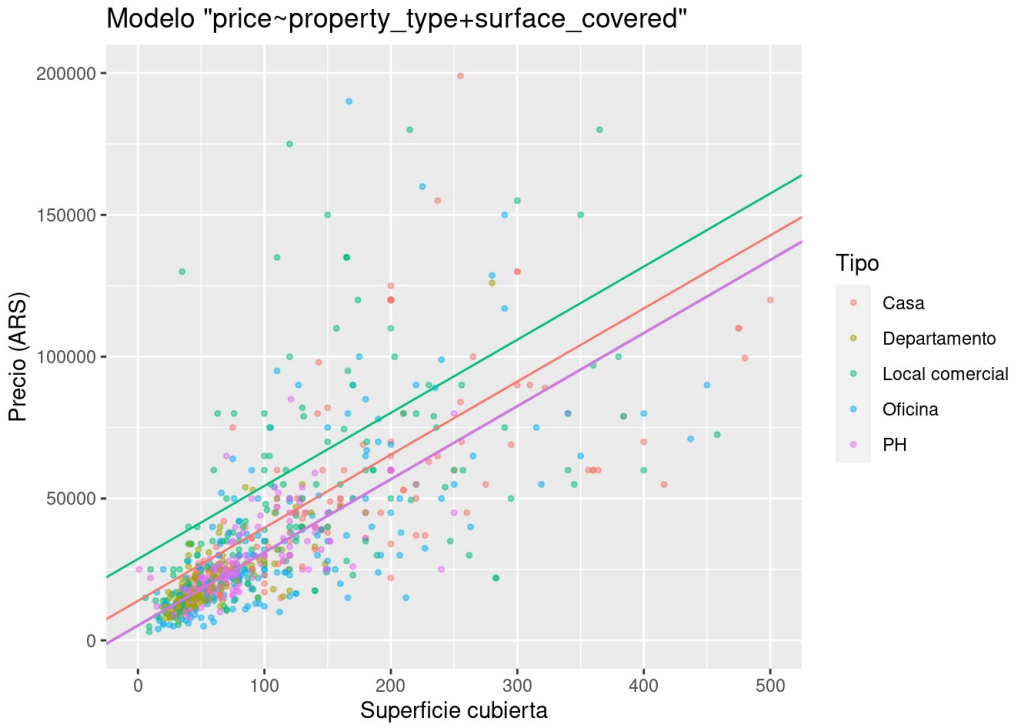
```
##
## Attaching package: 'scales'
```

```
## The following object is masked from 'package:purrr':
##
##     discard
```

```
## The following object is masked from 'package:readr':
##
##     col_factor
```

```
hex <- hue_pal()(5) #los codigos de los colores estan aca



ggplot(datos9) +
  aes(x = surface_covered, y = price, color = property_type) +
  geom_point(size = 0.9, alpha = 0.5) +
  xlim(0, 500) +
  ylim(0,2e+05) +
  xlab("Superficie cubierta") +
  ylab("Precio (ARS)") +
  labs(color = "Tipo")+
  ggtitle("Modelo \"price~property_type+surface_covered\"") +
  geom_abline(intercept = (coeM2_proptype_surface[1]), slope = coeM2_proptype_surface[6], color = hex[1]) +
  geom_abline(intercept = (coeM2_proptype_surface[1] + coeM2_proptype_surface[2]), slope = coeM2_proptype_surface
[6], color = hex[2] ) +
  geom_abline(intercept = (coeM2_proptype_surface[1] + coeM2_proptype_surface[3]), slope = coeM2_proptype_surface
[6], color = hex[3]) +
  geom_abline(intercept = (coeM2_proptype_surface[1] + coeM2_proptype_surface[4]), slope = coeM2_proptype_surface
[6], color = hex[4]) +
  geom_abline(intercept = (coeM2_proptype_surface[1] + coeM2_proptype_surface[5]), slope = coeM2_proptype_surface
[6], color = hex[5])
```

```
## Warning: Removed 55 rows containing missing values (geom_point).
```




```
#sup*prop type
ajusM2_proptypexsurface <- lm(price~property_type*surface_covered, data = datos9)
coeM2_proptypexsurface <- coef(ajusM2_proptypexsurface)
print(("Modelo price~property_type*surface_covered"))
```

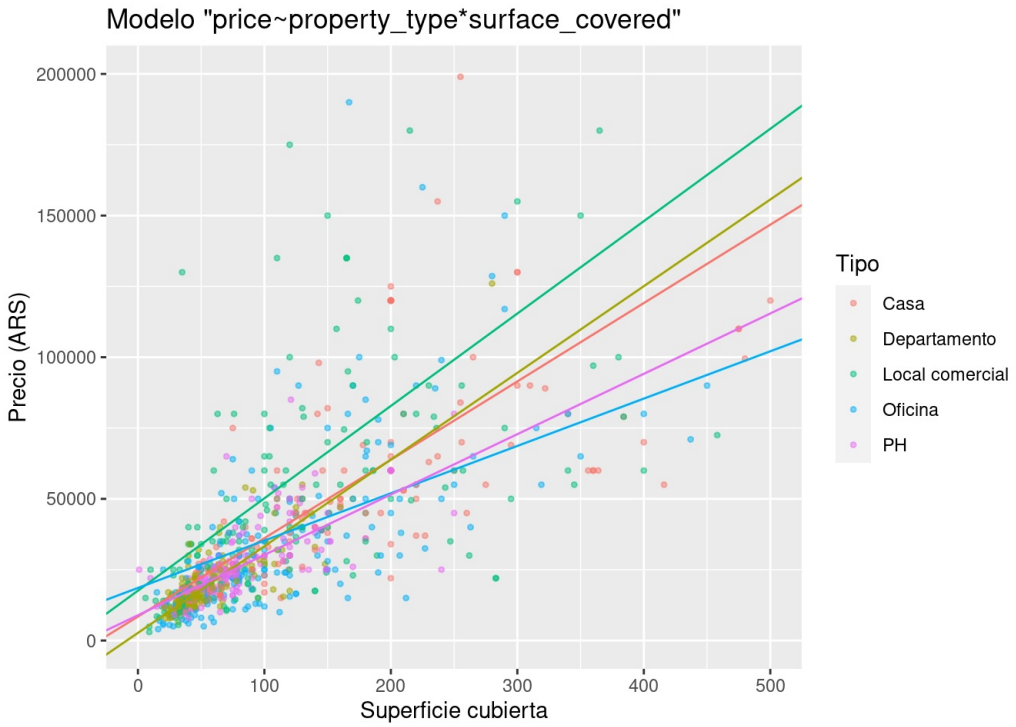
```
## [1] "Modelo price~property_type*surface_covered"
```

```
coeM2_proptypexsurface
```

```
##                                   (Intercept)
##                                    8450.22755
##                      property_typeDepartamento
##                                   -5766.10804
##                  property_typeLocal comercial
##                                    9142.61397
##                            property_typeOficina
##                                   10108.90852
##                                 property_typePH
##                                     473.42681
##                                 surface_covered
##                                     276.68134
##    property_typeDepartamento:surface_covered
##                                      29.31001
## property_typeLocal comercial:surface_covered
##                                      49.40454
##         property_typeOficina:surface_covered
##                                    -109.58951
##              property_typePH:surface_covered
##                                     -63.64312
```

```
ggplot(datos9) +
  aes(x = surface_covered, y = price, color = property_type) +
  geom_point(size = 0.9, alpha = 0.5) +
  xlim(0, 500) +
  ylim(0, 2e+05) +
  xlab("Superficie cubierta") +
  ylab("Precio (ARS)") +
  labs(color = "Tipo") +
  ggtitle("Modelo \"price~property_type*surface_covered\"") +
  geom_abline(intercept = (coeM2_proptypexsurface[1]), slope = coeM2_proptypexsurface[6], color = hex[1]) +
  geom_abline(intercept = (coeM2_proptypexsurface[1]+coeM2_proptypexsurface[2]), slope = (coeM2_proptypexsurface[
6]+coeM2_proptypexsurface[7]), color = hex[2]) +
  geom_abline(intercept = (coeM2_proptypexsurface[1]+coeM2_proptypexsurface[3]), slope = (coeM2_proptypexsurface[
6]+coeM2_proptypexsurface[8]), color = hex[3]) +
  geom_abline(intercept = (coeM2_proptypexsurface[1]+coeM2_proptypexsurface[4]), slope = (coeM2_proptypexsurface[
6]+coeM2_proptypexsurface[9]), color = hex[4]) +
  geom_abline(intercept = (coeM2_proptypexsurface[1]+coeM2_proptypexsurface[5]), slope = (coeM2_proptypexsurface[
6]+coeM2_proptypexsurface[10]), color = hex[5])
```

```
## Warning: Removed 55 rows containing missing values (geom_point).
```



Interpretando los resultados, el primer modelo (superficie+tipo_propiedad) nos muestra los coeficientes individuales de cada variable, sin interacción entre ellos, por esto es que las rectas de los diferentes ajustes son todas paralelas: la pendiente de cada uno de ellos es la misma que para el ajuste de las casas, y solo se modifica la ordenada de cada uno. (Aclaración: las rectas faltantes no se observan porque se solapan con la recta graficada en color violeta, dado que sus ordenadas son muy similares, y sus pendientes son iguales). Por otra parte, el segundo modelo (superficie*tipo_propiedad) no sólo muestra los coeficientes individuales de cada variable sino que también indica la relación entre cada uno de

los tipos de propiedad con la superficie, es decir, modifica tanto la ordenada como la pendiente de las rectas correspondientes a cada uno de los ajustes, y no se queda solo con la pendiente del ajuste para las casas como referencia. En este modelo estaremos viendo el efecto de la superficie, el efecto del tipo de propiedad, y además, el efecto de la superficie para una propiedad específica.

6 - Modelos alternativos

```
datos10 <- datos %>% filter(!is.na(price), !is.na(property_type), !is.na(lat), !is.na(lon), !is.na(rooms), !is.na
(fondo), !is.na(created_on))

#Creamos vectores con posibles modelos de 2 y 3 variables.
variables <- c("property_type", "lat", "lon", "rooms", "fondo", "created_on")

p2 <- powerSet(variables, 2)
p2 <- p2[lapply(p2, length) == 2]
p3 <- powerSet(variables, 3)
p3 <- p3[lapply(p3, length) == 3]

v2 <- c()
for(set in p2) {
  v2 <- c(v2, paste0("price~", set[1], "+", set[2]))
}

v3 <- c()
for(set in p3) {
  v3 <- c(v3, paste0("price~", set[1], "+", set[2], "+", set[3]))
}
```

Exploramos primero modelos de 2 variables.

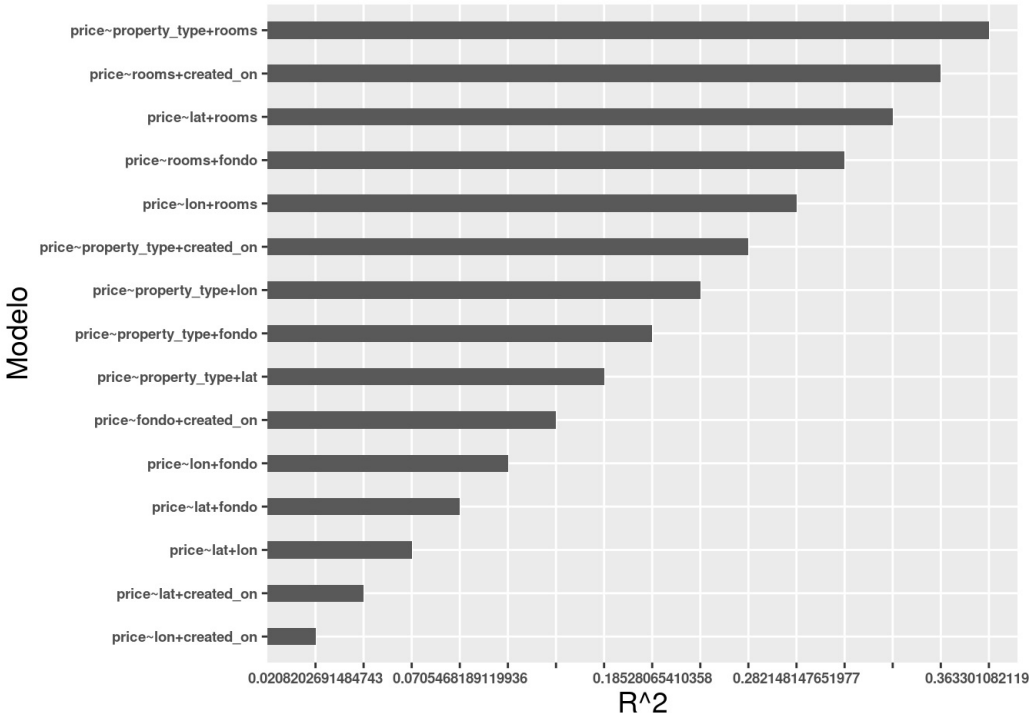
```
m2 <- matrix(data = NA, nrow = length(v2), ncol = 2)

for(i in 1:length(v2)) {
  modelo<-lm(formula(v2[i]),data=datos10)
  m2[i, ] <- c(v2[i], summary(modelo)$r.squared)
}
#convierto m2 en dataframe
m2 <- as.data.frame(m2)

#ordeno por r^2
m2$V1 <- factor(m2$V1, levels = m2$V1[order(m2$V2)])

#Imprimimos en consola el modelo de mayor R^2
print(m2[which.max(m2[,2]),])
```

```
##                          V1               V2
## 4 price~property_type+rooms 0.363301082119212
```

```
#Imprimimos la media de los R^2
print(mean(sapply(m2[,2], as.numeric)))
```

```
## [1] 8
```

```
ggplot(data = m2, aes(x = V1, y = V2)) +
  geom_bar(stat="identity", width=0.4, position = position_dodge(width=1)) +
  xlab("Modelo") +
  ylab("R^2") +
  scale_y_discrete(guide = guide_axis(check.overlap = TRUE)) +
  theme(axis.text=element_text(size=7,face="bold"),
        axis.title=element_text(size=14)) +
  coord_flip()
```



Analicemos qué pasa ahora con modelos de 3 variables.

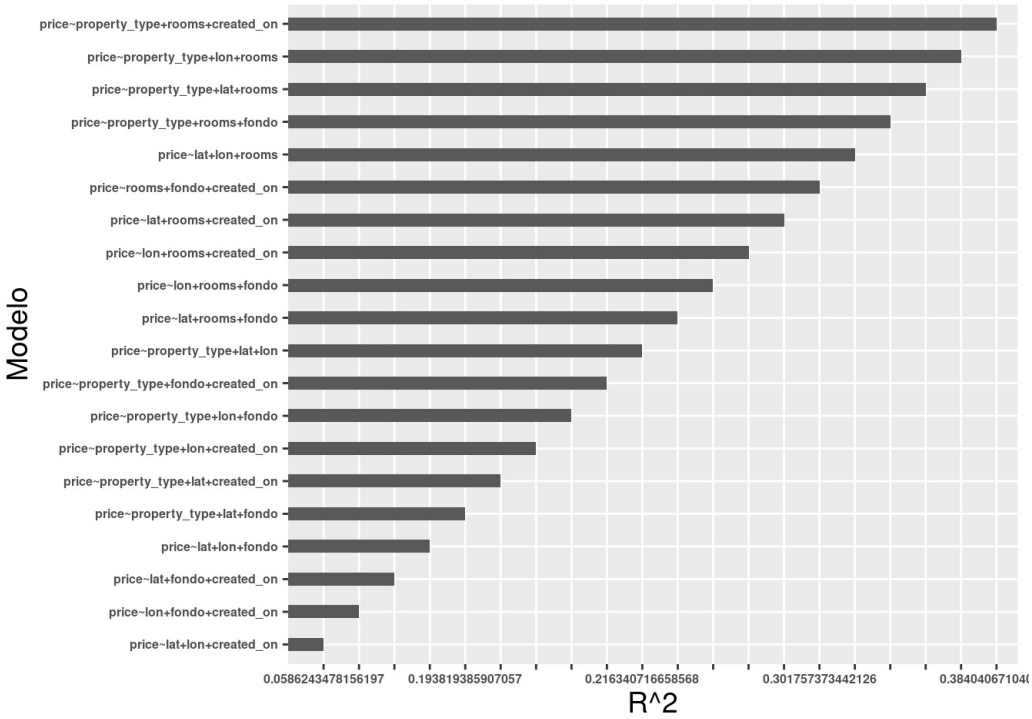
```
m3 <- matrix(data = NA, nrow = length(v3), ncol = 2)

for(i in 1:length(v3)) {
  modelo<-lm(formula(v3[i]),data=datos10)
  m3[i, ] <- c(v3[i], summary(modelo)$r.squared)
}

#convierto m2 en dataframe
m3 <- as.data.frame(m3)

#ordeno por r^2
m3$V1 <- factor(m3$V1, levels = m3$V1[order(m3$V2)])

#Imprimimos en consola el modelo de mayor R^2
print(m3[which.max(m3[,2]),])
```

```
##                                       V1                V2
## 14 price~property_type+rooms+created_on 0.384040671040095
```

```
#Imprimimos la media de los R^2
print(mean(sapply(m3[,2], as.numeric)))
```

```
## [1] 10.5
```

```
ggplot(data = m3, aes(x = V1, y = V2)) +
  geom_bar(stat="identity", width=0.4, position = position_dodge(width=1)) +
  xlab("Modelo") +
  ylab("R^2")+
  scale_y_discrete(guide = guide_axis(check.overlap = TRUE)) +
  theme(axis.text=element_text(size=6,face = "bold"),
        axis.title=element_text(size=14)) +
  coord_flip()
```



Podemos ver que los modelos de tres variables en general tienen menor error de ajuste que los de dos, ya que la media de los coeficientes R^2 correspondientes al conjunto de modelos de tres variables es mayor a la de modelos de dos variables. En particular, el modelo que tiene menor error de ajuste (mayor R^2) es el correspondiente a la fórmula “price~property_type+rooms+created_on”.

7 - Error promedio de predicción por validación cruzada

```
PMAE <- function(errores, datos, target) {
  return(mean(errores)/mean(datos[[target]]))
}

crossval <- function(datos, modelo, n_obs, fun_error, n_muestras=10){
  target <- strsplit(modelo, "[~]")[[1]][1]
  errores <- c()
  for(i in 1:n_muestras) {
    indicesEval <- sample(1:nrow(datos), size = n_obs)
    datosModelo <- datos[-indicesEval,]
    modeloLin <- lm(formula(modelo), data = datosModelo)
    abserrs <- c()
    for(o in indicesEval) {
      predicho <- predict(modeloLin, newdata = datos[o,])
      abserrs <- c(abserrs, abs(datos[[target]][o]-predicho))
    }
    errores <- c(errores, fun_error(abserrs, datos, target))
  }
  return(list(errores, mean(errores), var(errores), modelo, lm(formula(modelo), data = datos)))
}
```

Aca aplicamos la función para los modelos del punto anterior, creando una matriz con cada modelo y su error promedio de predicción en cada fila.

```
pmaes <- matrix(NA,nrow = length(v2) + length(v3), ncol = 2)
for(i in 1:length(v2)) {
   cv <- crossval(datos10, v2[i], 2, PMAE, 10)
   pmaes[i, 1] <- v2[i]
   pmaes[i, 2] <- as.numeric(cv[2])
}
for(i in 1:length(v3)) {
   cv <- crossval(datos10, v3[i], 2, PMAE, 10)
   pmaes[length(v2)+i, 1] <- v3[i]
   pmaes[length(v2)+i, 2] <- as.numeric(cv[2])
}
print(pmaes)
```

```
##       [,1]                                     [,2]
##  [1,] "price~property_type+lat"                "0.819011824676915"
##  [2,] "price~property_type+lon"                "0.482248871414114"
##  [3,] "price~lat+lon"                          "0.57637113480014"
##  [4,] "price~property_type+rooms"              "0.69024005624153"
##  [5,] "price~lat+rooms"                        "0.279376151683649"
##  [6,] "price~lon+rooms"                        "0.28617535886784"
##  [7,] "price~property_type+fondo"              "0.449746309814757"
##  [8,] "price~lat+fondo"                        "0.497657034785857"
##  [9,] "price~lon+fondo"                        "0.46810892442493"
## [10,] "price~rooms+fondo"                      "0.418849573584146"
## [11,] "price~property_type+created_on"         "0.593400724512697"
## [12,] "price~lat+created_on"                   "0.634159071767161"
## [13,] "price~lon+created_on"                   "0.733104313655431"
## [14,] "price~rooms+created_on"                 "0.78345373477572"
## [15,] "price~fondo+created_on"                 "0.440990618599565"
## [16,] "price~property_type+lat+lon"            "0.567015358958119"
## [17,] "price~property_type+lat+rooms"          "0.156435668358611"
## [18,] "price~property_type+lon+rooms"          "0.338356764509007"
## [19,] "price~lat+lon+rooms"                    "0.398209120394523"
## [20,] "price~property_type+lat+fondo"          "0.449109544067907"
## [21,] "price~property_type+lon+fondo"          "0.389478917348827"
## [22,] "price~lat+lon+fondo"                    "0.420699084653415"
## [23,] "price~property_type+rooms+fondo"        "0.748947110064838"
## [24,] "price~lat+rooms+fondo"                  "0.358913702481388"
## [25,] "price~lon+rooms+fondo"                  "0.241458628469773"
## [26,] "price~property_type+lat+created_on"     "0.582842500071975"
## [27,] "price~property_type+lon+created_on"     "0.786012626074918"
## [28,] "price~lat+lon+created_on"               "0.809132863730719"
## [29,] "price~property_type+rooms+created_on"   "0.483508975361268"
## [30,] "price~lat+rooms+created_on"             "0.618950688897839"
## [31,] "price~lon+rooms+created_on"             "0.721881756809629"
## [32,] "price~property_type+fondo+created_on"   "0.539020350755185"
## [33,] "price~lat+fondo+created_on"             "0.799115845590125"
## [34,] "price~lon+fondo+created_on"             "0.413373328726139"
## [35,] "price~rooms+fondo+created_on"           "0.675720716005601"
```

```
#Imprimimos el modelo de menor error de predicción.
print(pmaes[which.min(pmaes[,2]), ])
```

```
## [1] "price~property_type+lat+rooms" "0.156435668358611"
```

Se ve que el modelo de menor error de predicción no es necesariamente el modelo de menor error de ajuste.