Modelo Lineal Multiple

Camilo Vega

Contents

Introduccion

Este documento presenta un análisis destinado a la construcción de un modelo lineal multiple utilizando los datos del dataframe "vivienda4" del paquete "paqueteMET". El objetivo es determinar el modelo más adecuado para explicar el precio de las viviendas en relación con los metros cuadrados zona de ubicación y tipo de vivienda.

Carga de librerias y funciones personalizadas

En este análisis, se utilizan las librería listadas en el siguiente código. Además, para facilitar la creación de visualizaciones, se emplean las funciones personalizadas, las cuales se encuentran en el archivo funciones_personalizadas.R.

```
# Carga de paquetes necesarios para el código
library(tidyverse) # Conjunto de paquetes para manipulación de datos
library(ggdist) # Extiende ggplot2 con gráficos de distribución
library(tidyquant) # Paquete de finanzas para análisis cuantitativo de datos
library(paqueteMET) # Paquete para el análisis de series temporales
library(knitr) # Paquete para creación de tablas en formato de salida
library(bestNormalize) # Busca la mejor transformación para noralización
library(rlang) # Conjunto de herramientas para programación en R
library(broom) # Convierte resultados de modelos en tablas y gráficos
library(qqplotr) # Paquete para gráficos QQ-plot
library(gridExtra) # Paquete para la combinación de gráficos
library(grid) # Paquete para la manipulación de grillas de gráficos
library(tidymodels) # Conjunto de paquetes para modelado estadístico
library(multilevelmod) # Paquete para ajuste de modelos de efectos mixtos
library(gee)
library(workboots)

# Se crea un objeto "datos_vivienda" que contiene los datos de vivienda4
datos_vivienda <- vivienda4

# Carga de funciones personalizadas
source("funciones_personalizadas.R")
```

Ingenieria de caracteristicas

Teniendo en cuenta las recomendaciones del análisis exploratorio inicial detallado en el documento `analisis_exploratorio.pdf`, se han realizado las siguientes transformaciones en los datos del dataframe `vivienda4` para construir un modelo lineal multiple adecuado.

Filtrado

Se ha llevado a cabo un filtrado de los datos del dataframe para incluir únicamente aquellos que corresponden a las Zonas Norte y Sur. De esta manera, se podrá trabajar exclusivamente con la información relevante para el análisis y construcción del modelo lineal mulple.

```r
#Filtrado zona, tipo
datos_vivienda_mod <- datos_vivienda |>
    filter(zona %in% c("Zona Sur", "Zona Norte"))
```

Transformación de datos

Para transformar los datos y lograr una aproximación lo más cercana posible a una distribución normal, se utilizará el paquete de R llamado `bestNormalize`. De esta manera, se podrá mejorar la calidad del análisis y la construcción del modelo lineal simple.

Transformación preciom

```r
# Fijar la semilla para reproducibilidad
set.seed(4321)

# Aplicar bestNormalize a los datos de precios de vivienda
norm_preciom <- bestNormalize(datos_vivienda_mod$preciom, allow_orderNorm = FALSE)

# Observar la mejor transformación
norm_preciom$chosen_transform
```

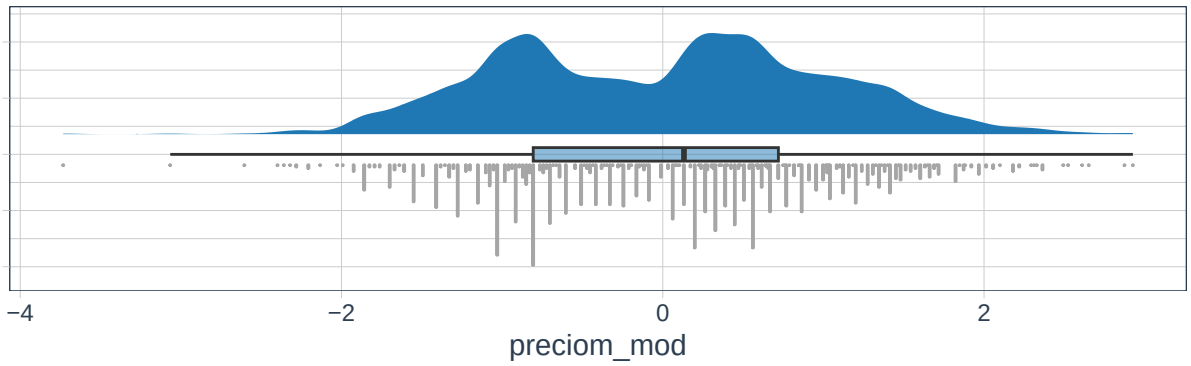
```
## Standardized Box Cox Transformation with 1632 nonmissing obs.:
##  Estimated statistics:
##  - lambda = -0.482184
##  - mean (before standardization) = 1.914907
##  - sd (before standardization) = 0.02539434
```

Luego de aplicar diferentes métodos de transformación en la variable `norm_preciom` se ha determinado que la mejor opción es el método de Box Cox, con un valor de λ de -0.482184. Esta información puede ser representada mediante la siguiente fórmula:

$$norm\_preciom\_mod = \frac{y^{-0.482184} - 1}{-0.482184}$$

Se procede a graficar la distribución resultante y a calcular las medidas de resumen correspondientes. De esta manera, se podrá visualizar y comprender mejor la distribución de los datos transformados.

```r
# Agregando preciom_mod al a los datos
datos_vivienda_mod <- datos_vivienda_mod |>
    mutate(preciom_mod = norm_preciom$x.t)

# Grafica de densidad y tabla de resumen
gg_rain_cloud(
    datos_vivienda_mod,
    preciom_mod); summary_table(
        datos_vivienda_mod,
        preciom_mod)
```



min	q1	median	mean	q3	max	skewness
-3.73	-0.81	0.13	0	0.72	2.93	0.03

Al observar las medidas de tendencia central de la distribución transformada, se puede apreciar que se acercan a cero y que la asimetría desaparece. Sin embargo, es importante mencionar que la transformación de Box Cox hace más evidente la presencia de una bimodalidad en los datos, lo que sugiere que hay variables que pueden estar afectando esta variable.

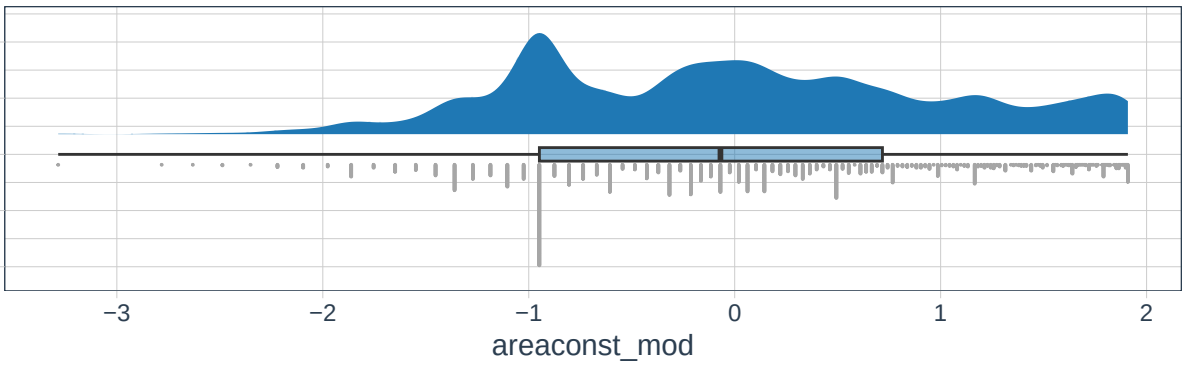
```
# Fijar la semilla para reproducibilidad
set.seed(8008)

# Aplicar bestNormalize a los datos de área de vivienda
norm_areaconst <- bestNormalize(datos_vivienda_mod$areaconst, allow_orderNorm = FALSE)

# Observar la mejor transformación
norm_areaconst$chosen_transform
```

```
## Standardized Yeo-Johnson Transformation with 1632 nonmissing obs.:
##  Estimated statistics:
##  - lambda = -1.227156
##  - mean (before standardization) = 0.810981
##  - sd (before standardization) = 0.001412117
```

Luego de aplicar diferentes métodos de transformación en la variable `areaconst` se ha determinado que la mejor opción es el método de Yeo-Johnson, con un valor de λ de -1.227156. Esta información puede ser representada mediante la siguiente fórmula:

$$areaconst\_mod = \frac{((y+1)^{-1.227156}) - 1}{-1.227156}$$

```
# Agregando datos_vivienda_mod al a los datos
datos_vivienda_mod <- datos_vivienda_mod |>
    mutate(areaconst_mod = norm_areaconst$x.t)

# Grafica de densidad y tabla de resumen
gg_rain_cloud(
    datos_vivienda_mod,
    areaconst_mod); summary_table(
        datos_vivienda_mod,
        areaconst_mod)
```



min	q1	median	mean	q3	max	skewness
-3.28	-0.95	-0.07	0	0.72	1.91	0.13

Al observar las medidas de tendencia central de la distribución transformada, se puede apreciar que se acercan a cero y que la asimetría desaparece. Sin embargo, es importante mencionar que la transformación de Yeo-Johnson hace más evidente la presencia de una multimodalidad en los datos, lo que sugiere que hay variables que pueden estar afectando la variable en cuestión.

Construcción de Modelos iniciales

Según los resultados de un modelo lineal simple descrito en el documento `Modelo-Lineal-Simple.pdf`, se ha observado que la relación entre el precio de la vivienda y el área es más fuerte cuando ambas variables están transformadas. Por esta razón, partiremos de este punto para construir un modelo que incluya también las variables `tipo` y `zona` en la ecuación lineal. Comenzaremos por el modelo más complejo, en el que todas las variables están relacionadas entre sí, y luego iremos simplificando el modelo hasta encontrar el que tenga el menor nivel de complejidad pero aún así ofrezca un buen ajuste a los datos.

Para cada una de los modelos se aplicara el siguente procedimiento:

1. Producción de objetos en R para creación de modelo analisis del mismo.
    - 1.A Creación del modelo lineal.
    - 1.B Creación tablas resumen del modelo (coeficientes, intervalos de confianza (95%), p-value coeficientes, p-value, R^2)
    - 1.C Creación graficas supuesto residuales.
    - 1.D Visualización del modelo.
2. Interpretación del modelo.

preciom_mod ~ areaconst_mod * tipo * zona

Creación Objetos R preciom_mod ~ areaconst_mod * tipo * zona

```
# Seleccionando datos y removiendo outliers
datos_multiple <- datos_vivienda_mod |>
    select(-estrato) |>
    remove_outliers(preciom_mod) |>
```

```
    remove_outliers(areaconst_mod)

# Creando modelo
lm_multiple_1 <- lm(preciom_mod ~ areaconst_mod*tipo*zona, datos_multiple)

# Tabla resumen modelo
tm1 <- glance(lm_multiple_1) |>
    mutate_all(~ round(.,4)) |>
    select(-10, -1) |>
    kable()

# Tabla resumen coeficientes
tm2 <- tidy(lm_multiple_1,
     conf.int = TRUE) |>
  mutate(across(where(is.numeric), ~ round(.,4))) |>
  kable()
# Agregando al data frame datos del modelo
datos_multiple_lm1_aug <- augment(lm_multiple_1) |>
  bind_cols(predict(lm_multiple_1, datos_multiple, interval = "prediction") |>
              as_tibble() |>
              select(-fit)) |>
  mutate(preciom_rev = predict(norm_preciom, preciom_mod, inverse = TRUE),
         .fitted_rev = predict(norm_preciom, .fitted, inverse = TRUE),
         areaconst_rev = predict(norm_areaconst, areaconst_mod, inverse = TRUE),
         lwr_rev = predict(norm_preciom, lwr, inverse = TRUE),
         upr_rev = predict(norm_preciom, upr, inverse = TRUE))

# Grafica modelo
pm1 <- gg_lm_plot(datos_multiple_lm1_aug,
                  areaconst_rev, preciom_rev, .fitted_rev, lwr_rev, upr_rev) +
    facet_grid(vars(tipo), vars(zona))
```

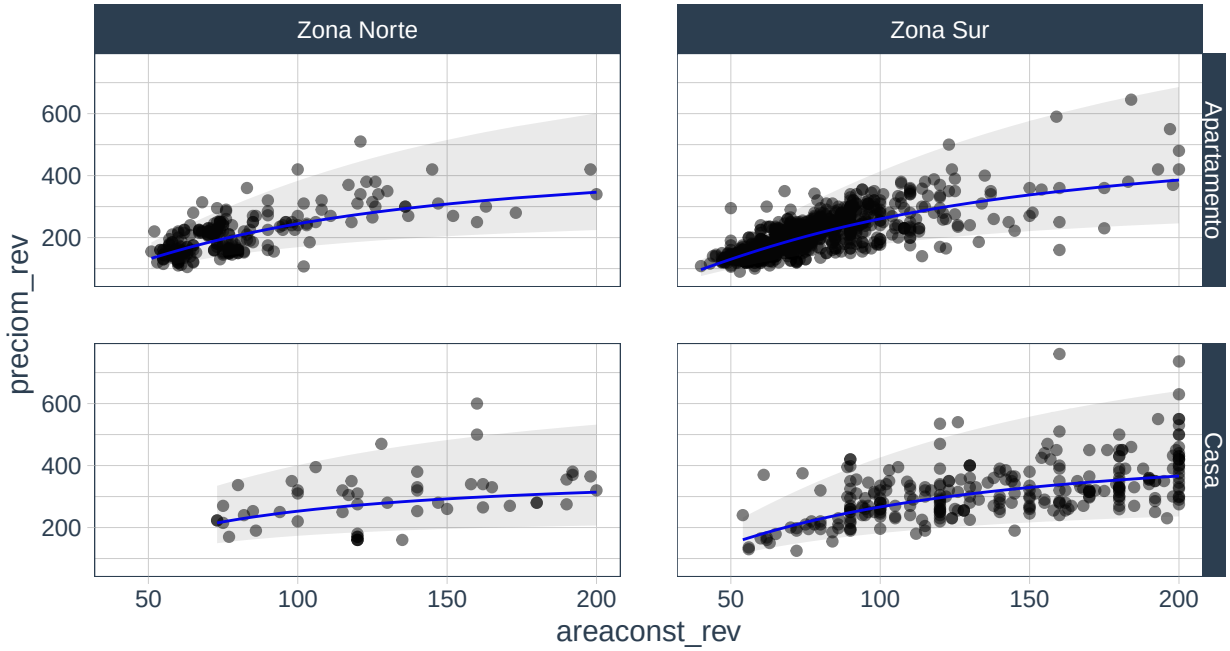
Tablas y Graficas preciom_mod ~ areaconst_mod * tipo * zona

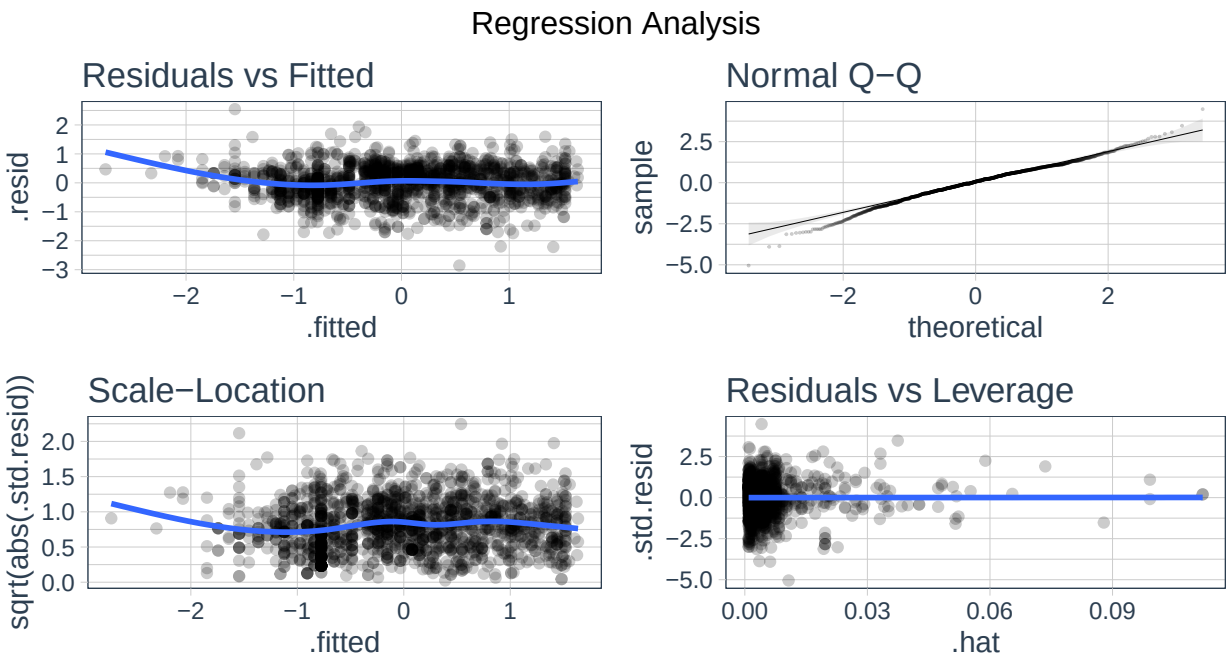
```
tm1; tm2; pm1; reg_analysis(
  datos_multiple_lm1_aug)
```

adj.r.squared	sigma	statistic	p.value	df	logLik	AIC	BIC	df.residual	nobs
0.6733	0.5693	480.8862	0	7	-1391.482	2800.964	2849.537	1623	1631

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	-0.0977	0.0378	-2.5877	0.0097	-0.1718	-0.0237
areaconst_mod	0.7797	0.0469	16.6299	0.0000	0.6877	0.8717
tipoCasa	0.3111	0.1752	1.7754	0.0760	-0.0326	0.6548
zonaZona Sur	0.1228	0.0422	2.9107	0.0037	0.0400	0.2055
areaconst_mod:tipoCasa	-0.2858	0.1408	-2.0294	0.0426	-0.5620	-0.0096
areaconst_mod:zonaZona Sur	0.0643	0.0513	1.2525	0.2106	-0.0364	0.1650
tipoCasa:zonaZona Sur	-0.1299	0.1895	-0.6858	0.4929	-0.5015	0.2417

term	estimate	std.error	statistic	p.value	conf.low	conf.high
areaconst_mod:tipoCasa:zonaZona Sur	0.1299	0.1510	0.8602	0.3898	-0.1663	0.4262





Interpretación modelo preciom_mod ~ areaconst_mod * tipo * zona

Al incluir las variables `tipo` y `zona`en el modelo, se ha observado una mejora significativa en comparación con la versión `box_box` descrita en el documento `Modelo-Lineal-Simple.pdf`. El modelo resultante presenta

un valor de R^2 más alto y se ha mejorado la normalidad de los residuos, así como disminuido su tendencia en relación al modelo `box_box`.

Sin embargo, al analizar los coeficientes del modelo, se ha detectado que no todas las variables parecen contribuir de manera significativa a la predicción del precio de la vivienda. Específicamente, se ha observado que los valores p asociados a las relaciones entre área/zona, tipo/zona y área/tipo/zona son bajos, lo que sugiere que estas variables tienen una baja correlación con la variable respuesta. Debido a esto, el siguiente modelo que se evaluará excluye estas relaciones.

preciom_mod ~ areaconst_mod * tipo + zona

Creación Objetos R preciom_mod ~ areaconst_mod * tipo + zona

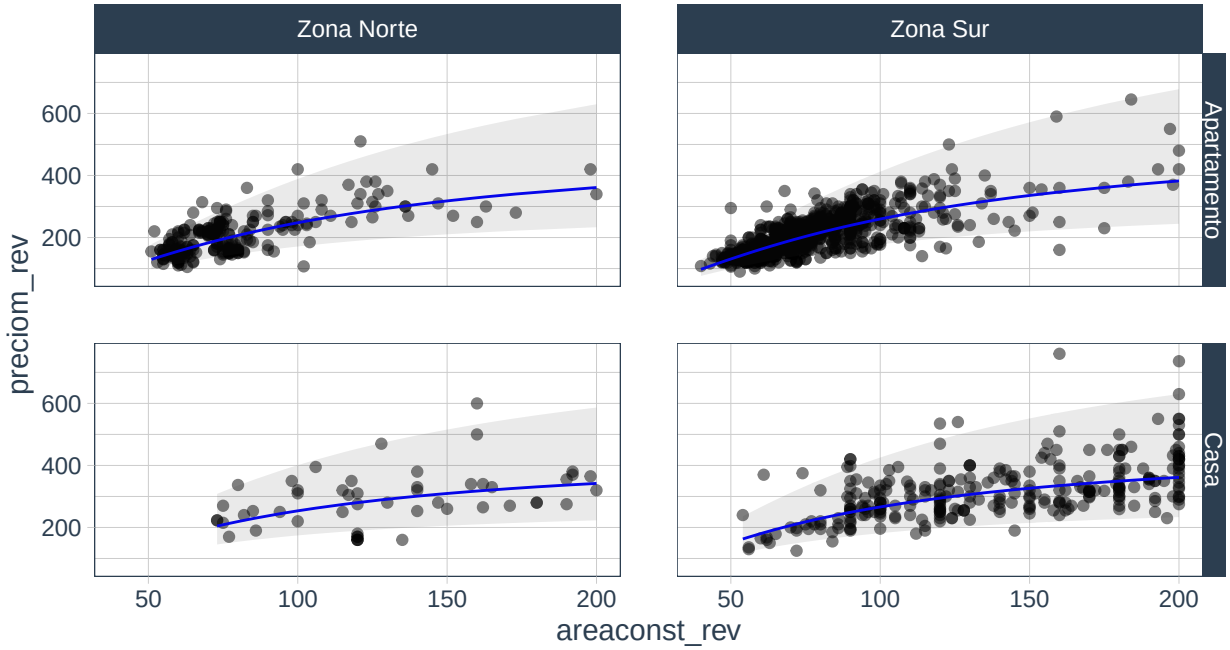
```
# Creando modelo
lm_multiple_2 <- lm(preciom_mod ~ areaconst_mod*tipo+zona, datos_multiple)

# Tabla resumen modelo
tm21 <- glance(lm_multiple_2) |>
    mutate_all(~ round(.,4)) |>
    select(-10, -1) |>
    kable()

# Tabla resumen coeficientes
tm22 <- tidy(lm_multiple_2,
     conf.int = TRUE) |>
  mutate(across(where(is.numeric), ~ round(.,4))) |>
  kable()
# Agregando al data frame datos del modelo
datos_multiple_lm2_aug <- augment(lm_multiple_2) |>
  bind_cols(predict(lm_multiple_2, datos_multiple, interval = "prediction") |>
              as_tibble() |>
              select(-fit)) |>
  mutate(preciom_rev = predict(norm_preciom, preciom_mod, inverse = TRUE),
         .fitted_rev = predict(norm_preciom, .fitted, inverse = TRUE),
         areaconst_rev = predict(norm_areaconst, areaconst_mod, inverse = TRUE),
         lwr_rev = predict(norm_preciom, lwr, inverse = TRUE),
         upr_rev = predict(norm_preciom, upr, inverse = TRUE))

# Grafica modelo
pm2 <- gg_lm_plot(datos_multiple_lm2_aug,
                  areaconst_rev, preciom_rev, .fitted_rev, lwr_rev, upr_rev) +
    facet_grid(vars(tipo), vars(zona))
```

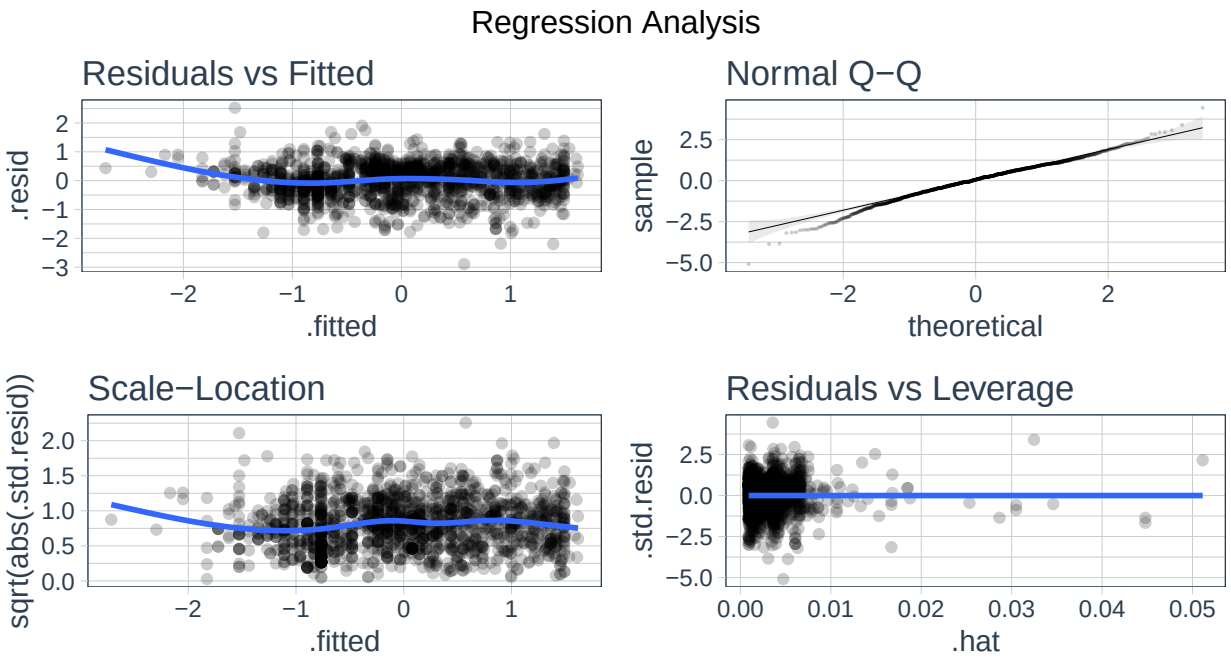
Tablas y Graficas preciom_mod ~ areaconst_mod * tipo + zona

```
tm21; tm22; pm2; reg_analysis(
  datos_multiple_lm2_aug
  )
```

adj.r.squared	sigma	statistic	p.value	df	logLik	AIC	BIC	df.residual	nobs
0.673	0.5696	839.5402	0	4	-1393.813	2799.626	2832.007	1626	1631

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	-0.1045	0.0343	-3.0434	0.0024	-0.1718	-0.0371
areaconst_mod	0.8340	0.0191	43.6333	0.0000	0.7965	0.8715
tipoCasa	0.1952	0.0665	2.9339	0.0034	0.0647	0.3257
zonaZona Sur	0.1297	0.0371	3.4958	0.0005	0.0569	0.2025
areaconst_mod:tipoCasa	-0.1688	0.0509	-3.3163	0.0009	-0.2686	-0.0690





Interpretación modelo preciom_mod ~ areaconst_mod * tipo + zona

Al remover las relaciones entre área/zona, tipo/zona y área/tipo/zona del modelo, se ha observado una ligera disminución en el valor de R^2 (0.3%). Sin embargo, los valores p asociados a los coeficientes han mejorado significativamente, siendo todos ellos menores a 0.005. Además, los supuestos sobre los residuos se mantienen estables, en niveles óptimos para valores altos y con pequeñas desviaciones en cuanto a normalidad y tendencia para valores bajos.

Al analizar los residuos, se ha observado que el área es la variable que tiene un mayor efecto en el precio de la vivienda. Además, las casas suelen tener precios más altos que los apartamentos y las viviendas en la zona sur suelen tener precios más elevados que las de la zona norte. Por último, se ha visto que la pendiente de los precios de los apartamentos es más pronunciada a medida que aumenta su área en comparación con el incremento en el área de las casas.

Dado que todos los coeficientes son estadísticamente significativos, no se considera necesario simplificar aún más el modelo.

Comparación y selección del modelo incial

De acuerdo con los resultados obtenidos, se puede apreciar que los dos modelos creados son similares. Para evaluar si existen diferencias significativas entre ellos, se aplicará un test de ANOVA.

```
# Comapración de modelos
anova(lm_multiple_1, lm_multiple_2) |>
    tidy() |>
    kable()
```

term	df.residual	rss	df	sumsq	statistic	p.value
preciom_mod ~ areaconst_mod * tipo * zona	1623	526.0335	NA	NA	NA	NA
preciom_mod ~ areaconst_mod * tipo + zona	1626	527.5390	-3	-1.505502	1.548336	0.2001543

Tras realizar un test de ANOVA y obtener un valor p de 0.2, no se ha encontrado evidencia suficiente para afirmar que existen diferencias significativas entre los dos modelos. Por lo tanto, se selecciona el modelo más sencillo para continuar con el análisis, que en este caso es `preciom_mod ~ areaconst_mod * tipo + zona`.

Construcción modelo para inferencia.

El modelo elegido, `preciom_mod ~ areaconst_mod * tipo + zona`, fue entrenado utilizando el conjunto completo de datos, lo que podría resultar en un sobreajuste y limitar su capacidad para hacer inferencias en nuevos datos. Por lo tanto, nuestro próximo paso será construir un modelo que pueda ser utilizado para inferir sobre nuevos datos. Para lograr esto, dividiremos nuestro conjunto de datos en un conjunto de entrenamiento y otro de prueba, y utilizaremos una validación cruzada con k = 10 pliegues en el conjunto de entrenamiento.

Utilizando el paquete `tidymodels`, vamos a modelar estos datos utilizando cuatro motores diferentes de regresión lineal: lm, stan, glenet y brulee. Para estos últimos dos, realizaremos una ajuste de parámetros adicional mediante una cuadrícula de 10 combinaciones. Posteriormente, seleccionaremos el mejor modelo en función de su coeficiente de determinación (R^2) en la validación cruzada.

División de los datos

Se dividirán los datos en un conjunto de entrenamiento y otro de prueba utilizando una proporción del 70% y 30%, respectivamente. Además, se aplicará una técnica de validación cruzada con 10 pliegues sobre el conjunto de datos de prueba. De esta manera, se podrá evaluar el rendimiento de los modelos de regresión lineal en datos no vistos y evitar sobreajuste.

```
# Fijar la semilla para reproducibilidad
set.seed(1234)

# Split 70/30
datos_lm2_split <- datos_multiple_lm2_aug |>
    select(preciom_mod, areaconst_mod, tipo, zona, areaconst_mod) |>
    initial_split(prop = 0.7,
                  strata = zona)

#Conjunto de entrenamiento, prueba y k=10 prlieges
datos_lm2_train <- training(datos_lm2_split)
datos_lm2_test <- testing(datos_lm2_split)
datos_lm2_folds <- vfold_cv(datos_lm2_train, strata = zona, v = 10)
```

Creación de modelos

Usando la función `recipe`, especificaremos la fórmula `preciom_mod ~ areaconst_mod * tipo + zona` para definir nuestra receta de transformación de datos. Dado que la transformación de datos ya se aplicó en la

fase inicial, no agregaremos pasos adicionales. Posteriormente, procederemos a definir los modelos que serán utilizados.

```
# Receta y definición del la formula
# Debido a como funciona recipe primero definimos la formula como preciom_mod ~ .
#que es el equivalente a preciom_mod ~ areaconst_mod + tipo + zona
lm2_rec <- recipe(preciom_mod ~ ., data = datos_lm2_train) |>
    # Luego agregamos la interaccion entre  areaconst_mod:tipo
    step_interact(~ areaconst_mod:tipo) |>
    # Pasando factores a variables indicadoras
    step_dummy(all_factor_predictors(), one_hot = TRUE)

# Iniciando los modelos a usar
models_lm2 <- list(
    lm_reg = linear_reg(),
    stan_reg = linear_reg(engine = "stan"),
    glmnet_reg = linear_reg(engine = "glmnet", penalty = tune(), mixture = tune()),
    brulee_reg = linear_reg(engine = "brulee", penalty = tune(), mixture = tune())
)

#
workflow_set_lm2 <- workflow_set(preproc = list(formula = lm2_rec),
                                 models = models_lm2)
```

Entremamiento modelos

Se entrenarán los modelos usando validación cruzada con k=10 y se guardarán los resultados en `grid_results`.

```
# Fijar la semilla para reproducibilidad
set.seed(1234)

# Definiendo que se guarden los pasos del flujo
grid_ctrl <-
   control_grid(
      save_pred = TRUE,
      parallel_over = "everything",
      save_workflow = TRUE
   )

# Entrenando los modelos con  k=10 pliegues y guardando los resultados
grid_results <-
   workflow_set_lm2 |>
   workflow_map(
      seed = 1234,
      resamples = datos_lm2_folds,
      grid = 10,
      control = grid_ctrl
   )
```

Selección del mejor modelo

Primero miraremos los resultados de los 10 mejores modelos segun su R^2.

```
grid_results |>
    rank_results() |>
    filter(.metric == "rsq") |>
    arrange(desc(mean)) |>
    mutate(rank = row_number()) |>
    select(-n, -preprocessor, -model) |>
    head (10) |>
    kable()
```

wflow_id	.config	.metric	mean	std_err	rank
formula_brulee_reg	Preprocessor1_Model10	rsq	0.6848331	0.0195983	1
formula_brulee_reg	Preprocessor1_Model01	rsq	0.6847267	0.0196114	2
formula_brulee_reg	Preprocessor1_Model03	rsq	0.6845215	0.0194125	3
formula_brulee_reg	Preprocessor1_Model02	rsq	0.6844500	0.0194401	4
formula_lm_reg	Preprocessor1_Model1	rsq	0.6843557	0.0193855	5
formula_brulee_reg	Preprocessor1_Model04	rsq	0.6843494	0.0195700	6
formula_glmnet_reg	Preprocessor1_Model10	rsq	0.6843015	0.0194471	7
formula_glmnet_reg	Preprocessor1_Model05	rsq	0.6843003	0.0194475	8
formula_glmnet_reg	Preprocessor1_Model09	rsq	0.6842999	0.0194484	9
formula_glmnet_reg	Preprocessor1_Model08	rsq	0.6842991	0.0194498	10

Vemos como los primeros puestos son ocupados por el motor brulee por lo cual usaremos el mejor modelo del mismo segun si R^2.

```
# Seleccionado mejor modelo
best_results <-
   grid_results |>
   extract_workflow_set_result("formula_brulee_reg") |>
   select_best(metric = "rsq")

# Reentrenando sobre el conjunto completo de datos
lm2_final_results <- grid_results |>
  extract_workflow("formula_brulee_reg") |>
  finalize_workflow(best_results) |>
  last_fit(split = datos_lm2_split)
```

Coeficientes y R^2 modelo inferencia final

Este es el R^2 y coeficientes del modelo.

```
collect_metrics(lm2_final_results) |>
  filter(.metric == "rsq") |>
  select(.metric, .estimate
         ) |>
  kable(); lm2_final_results |>
  extract_workflow() |>
  extract_model() |>
  coef() |>
  as.data.frame() |>
  rownames_to_column(var = "term") |>
```

```
  set_names(c("term", "estimate")) |>
  # Los coeficientes se muestran completos, se transformaran para mostrarlos en
  # los mismos terminos que lo lace lm
  pivot_wider(names_from = term, values_from = estimate) |>
  mutate(`(Intercept)` = `(Intercept)`+tipo_Apartamento+zona_Zona.Norte) |>
  select(-tipo_Apartamento,-zona_Zona.Norte) |>
  pivot_longer(cols = everything())|>
  set_names(c("term", "estimate")) |>

  kable()
```

.metric	.estimate
rsq	0.6384055

term	estimate
(Intercept)	-0.1226177
areaconst_mod	0.8480082
areaconst_mod_x_tipoCasa	-0.2353950
tipo_Casa	0.1986937
zona_Zona.Sur	0.0121117

Debido a que el motor Brulee está basado en la biblioteca de aprendizaje profundo `tensorflow`, no es posible obtener intervalos de confianza para los coeficientes.

Al observar los coeficientes del modelo ajustado mediante validación cruzada, podemos notar que son similares a los del modelo inicial `preciom_mod ~ areaconst_mod * tipo + zona`. Aunque el coeficiente de determinación (R^2) del modelo ajustado es menor, esto era de esperar debido a que la validación cruzada implica una mayor generalización del modelo, lo que puede resultar en una disminución del R^2. Sin embargo, la ventaja de este enfoque es que el modelo ajustado será menos propenso al sobreajuste y, por lo tanto, será más adecuado para predecir nuevos datos.

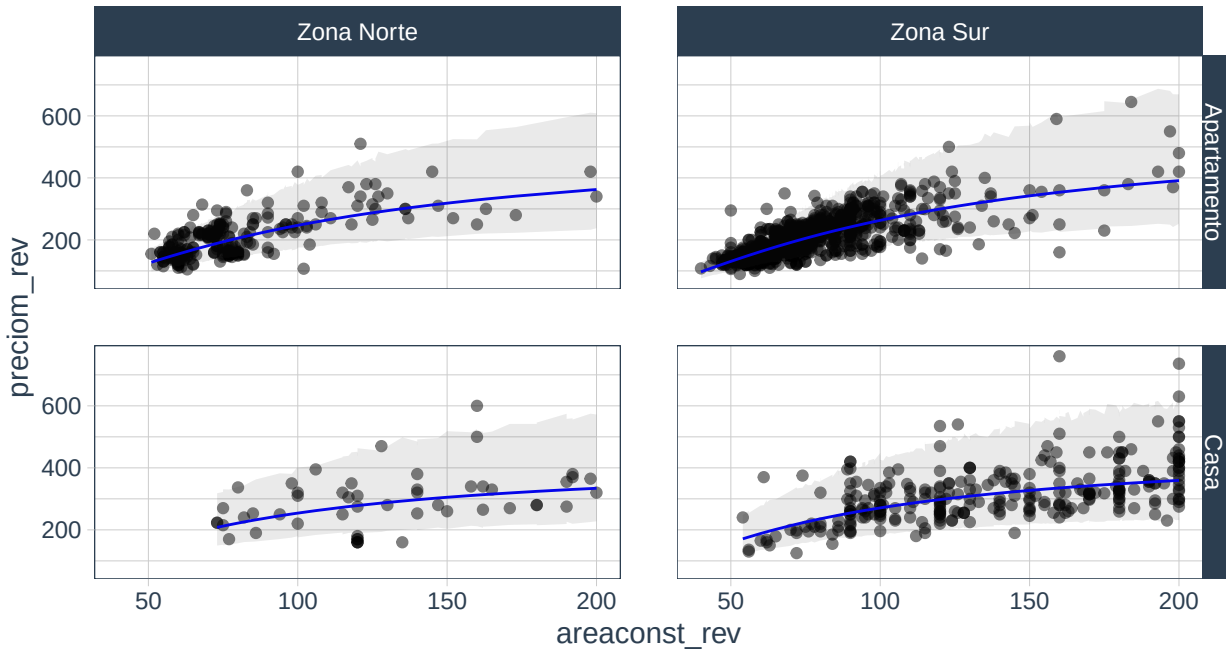
Gráfica modelo de inferencia final.

Como se mencionó previamente, el modelo Brulee no tiene la capacidad de generar intervalos de confianza para los modelos lineales, ya que está basado en `tensorflow`. Para superar esta limitación, utilizaremos una metodología de boostrap usando la libreria `workboots`. En esta técnica, simularemos 2000 conjuntos de datos basados en los datos de entrenamiento y reentrenaremos el modelo. De esta manera, nuestros intervalos de predicción se tomarán del 95% de confianza de las distribuciones de estas simulaciones.

A continuación, presentamos la representación gráfica del modelo evaluado sobre todos los datos de Apartamentos y Casas de las Zonas Sur y Norte de estrato 4.

```
# Fijar la semilla para reproducibilidad
set.seed(345)

# Boostrap sobre 2000 simulaciones de los datos de entrenamiento
lm_pred_int <-
  lm2_final_results |>
  extract_workflow()  |>
  predict_boots(
```

```
    n = 2000,
    training_data = datos_lm2_train,
    new_data = datos_multiple_lm2_aug |>
    select(preciom_mod, areaconst_mod, tipo, zona, areaconst_mod)
  )

# Extrayendo los intervalos de las predicciones
lm_int <- lm_pred_int |>
  summarise_predictions() |>
  select(-rowid, -.preds, -.pred)

datos_finales <- datos_multiple_lm2_aug |>
              select(preciom_mod, areaconst_mod, tipo, zona, areaconst_mod)

datos_para_modelo <- lm2_rec |>
  prep() |>
  bake(datos_finales)

datos_pred <- lm2_final_results |>
  extract_workflow() |>
  extract_model() |>
  predict(new_data = datos_para_modelo)

datos_para_modelo_aug <- datos_finales |>
  bind_cols(datos_pred, lm_int) |>
  mutate(preciom_rev = predict(norm_preciom, preciom_mod, inverse = TRUE),
         .fitted_rev = predict(norm_preciom, .pred, inverse = TRUE),
         areaconst_rev = predict(norm_areaconst, areaconst_mod, inverse = TRUE),
         lwr_rev = predict(norm_preciom, .pred_lower, inverse = TRUE),
         upr_rev = predict(norm_preciom, .pred_upper, inverse = TRUE))

ggplot(datos_para_modelo_aug, aes(areaconst_rev, preciom_rev,)) +
  geom_point(alpha = 0.5) +
  geom_line(data = datos_para_modelo_aug, aes(areaconst_rev, .fitted_rev), color = "blue") +
  geom_ribbon(aes(ymin = lwr_rev, ymax = upr_rev ),alpha = 0.1)+
    theme_tq()  +
    facet_grid(vars(tipo), vars(zona))
```



Grabando modelo final

Para su futura implementación, se guardará el modelo en un archivo de documento .RDS. Esto permitirá una fácil recuperación del modelo y su uso en diferentes contextos. El archivo .RDS contendrá el objeto del modelo guardado como un archivo binario, lo que asegurará que todas las características del modelo, como los coeficientes y los parámetros, se mantengan intactos. Además, la extensión .RDS es ampliamente reconocida en el ecosistema de R como un formato de archivo para la persistencia de objetos, lo que garantiza que el modelo pueda ser utilizado por otros usuarios sin problemas.

Además, se almacenan los objetos `norm_areaconst` y `norm_preciom` como complemento del modelo, con el fin de poder transformar y revertir la transformación de datos para el modelado. Igualmente guardaremos la simulación boostrap `lm_pred_int`, ya que la reproducción de la misma es costosa en cuanto a tiempo. De esta manera, se logra mostrar los datos en su escala original.

```
saveRDS(lm2_final_results |>
          extract_workflow() |>
          extract_model(),
        "modelo_lineal_multiple.rds")

saveRDS(norm_areaconst,
        "area_box_cox_multiple.rds")

saveRDS(norm_preciom,
        "precio_box_cox_multiple.rds")

saveRDS(lm_pred_int,
        "boostrap_modelo_multiple.rds")
```