# Alexander_Barrantes

January 24, 2025

[ ]:
```r
#Título: "Proyecto Final - Curso 'Introducción a R"
#Autor: 'Alexander Barrantes Herrera'
#Fecha: "11 de Diciembre de 2024"


##0.Cargar Librerías y DataSet
rm(list=ls())
gc()

library(readxl)
library(dplyr)
library(tidyverse)
library(readr)
library(BiodiversityR)
library(tidymodels)

library(GGally)
library(ggplot2)
library(gplots)
library(ggthemes)

library(knitr)
library(rmarkdown)
library(pander)

#install.packages("dplyr")
#install.packages("tidyverse")
#install.packages("devtools")
#install.packages("tidymodels")


# Se carga data set Ifri el cual corresponde a muestreo de árboles
datos <- data(ifri)
datos <- as.data.frame(ifri)

##############################################################################################
## 1. Objetivo del Análisis
```

```
  "Describir el tipo de bosque con mayor frecuencia y predecir el volumen de␣
 ↪basal más aproximado mediante el mejor modelo predictivo."


##############################################################################################
## 2. Método - Descripción de los datos y análisis realizado.
  "Se describen los datos de especies, tipos de bosque, volumen de basal y␣
 ↪conteo con la ayuda de herramientas de estadística descriptiva"

  # Ver especies con más tipos de bosque
  datos |>
      group_by(forest) |>
      distinct(species) |>
      count()

  # Gráfico de Puntos y Lineas
  datos |>
    ggplot(aes(x = count,
               y = basal)) +
    geom_point() +
    geom_line()


  # Barras por Forest
  datos |>
    ggplot(aes(x = forest)) +
    geom_bar()

  # Barras por Species
  datos |>
    ggplot(aes(x = species)) +
    geom_bar()

  # Boxplot por Forest y Basal
  datos |>
    ggplot(aes(x = forest,
               y = basal)) +
    geom_boxplot()

  # Histograma
  datos |>
    ggplot(aes(x = basal)) +
    geom_histogram(bins = 50)

##############################################################################################
## 3. Resultados - Descripción de los principales resultados.
```

```
  "Se presenta los siguientes modelamientos de variables, donde se tomó como␣
↪referencia el tipo de bosque 'LOT' quien cuenta con mayor frencuencia de␣
↪registros"

 # Modelamiento de Variables
    # Se filta por Tipo de Bosque LOT y los campos cantidad y basal
    dfbpq <- datos |>
      filter(forest == "LOT") |>
      select(count, basal)

    # Modelo descriptivo
    modAltDiam <- linear_reg() |>
      fit(count ~ basal, data = dfbpq)
    modAltDiam

    tidy(modAltDiam)



  # 1° Modelo (Modelo Lineal)
  # Modelo Lineal entre las variables conteo y basal para los tipos de bosque␣
↪"LOT"
  dfbpq |>
    ggplot(aes(x = count,
               y = basal)) +
    geom_point() +
    geom_smooth(method = lm,
                se = FALSE,
                col = "firebrick2") +
    cowplot::theme_cowplot()

  # Segmentación de Data Sets para Preparación de Modelos
    # Asegurar reproducibilidad con selección aleatoria
      set.seed(5)
      nrow(dfbpq)

      split_datos <- initial_split(dfbpq, prop = 0.75)

    # Crea dataframe para los 2 sets
      train_data <- training(split_datos)
      test_data  <- testing(split_datos)

    # Crear receta. Define variables dependientes y variables independientes.

      receta <- recipe(count ~ basal, data = train_data)

        # Tipo de Modelo (Lineal General)
```

```
        mod_rl <- linear_reg() |>  set_engine("glm")

        # Crear flujo de trabajo. Agregar modelo y receta a un flujo de␣
↪trabajo.

        altdiam_wflow <- workflow() |>
          add_model(mod_rl) |> add_recipe(receta)

        #Ajustar Modelo
        altdiam_fit <-
          altdiam_wflow |> fit(data = train_data)

        #Ver Modelo
        altdiam_fit |>
          extract_fit_parsnip() |> tidy()

        # Probar modelo y sacar datos
        predict(altdiam_fit, test_data)

        #Predecir y comparar contra original usando los datos de test
        test_preds <- altdiam_fit |> augment(test_data)
        test_preds

        #Hacerlo para datos de entrenamiento
        train_preds <- altdiam_fit |> augment(train_data)
        train_preds

        #Seleccionar métricas de evaluación
        eval_metrics <- metric_set(rmse,rsq_trad)

        test_preds |>
          eval_metrics(truth = basal,
                       estimate = .pred)

        #Comparar para datos de entrenamiento
        train_preds |> eval_metrics(truth = basal, estimate = .pred)



  # 2° Modelo (Modelo Exponencial)
  # Modelo Exponencial entre las variables conteo y basal para los tipos de␣
↪bosque "LOT"
    dfbpq |>
      mutate(across(c(count, basal), ~log(.x+3))) |>
      ggplot(aes(x = count,
                 y = basal)) +
```

```
       geom_point() +
       geom_smooth(method = lm,
                   col = "firebrick1",
                   se = FALSE) +
       cowplot::theme_cowplot()

       # Crear receta. En este caso, se agrega la transformación de las␣
↪variables numéricas a log (base natural) + 3 (para evitar valores negativos␣
↪y ceros en log).
       receta2 <- recipe(basal ~ count,
                         data = train_data) |> step_mutate_at(all_numeric(),␣
↪fn = ~log(.x+3))

       # Visualiza  los datos después de receta
       receta2 |> prep(data = train_data) |>
         bake(new_data = train_data)

           # Tipo de modelo exponencial
           show_engines("linear_reg")

           mod_rl2 <- linear_reg() |>set_engine("glm")

           #Crear flujo de trabajo
           altdiam_wflow2 <- workflow() |>  add_model(mod_rl2) |>
             add_recipe(receta2)

           #Ajustar modelo
           altdiam_fit2 <- altdiam_wflow2 |>
             fit(data = train_data)

           # Ver modelo
           altdiam_fit2 |> extract_fit_parsnip() |>
                        tidy()

           #Predecir y comparar contra original en datos de verificación
           test_preds2 <- altdiam_fit2 |> augment(test_data)
           test_preds2

           #Predecir y comparar contra original en datos de entrenamiento
           train_preds2 <- altdiam_fit2 |>
             augment(train_data)
           train_preds2

           #Seleccionar métricas de evaluación
           eval_metrics <- metric_set(rmse,rsq_trad)

           test_preds2 |>
```

```
            mutate(across(.pred, ~exp(.x)-3)) |> eval_metrics(truth = basal,
                        estimate = .pred)

          train_preds2 |> mutate(across(.pred, ~exp(.x)-3)) |>
            eval_metrics(truth = basal, estimate = .pred)

   #Resultado/ Se identifica que el modelo Lineal presenta RMSE de 2119 y␣
↪RSQ_TRAD de -0.735.
   #   Por otra parte, el modelo exponencial presenta RMSE de 1392 y RSQ_TRAD␣
↪de 0.252.

##############################################################################################
## 4. Discusión - Patrones observados en los resultados.
 "Se presenta los siguientes comparativos de modelos, donde se tomó como␣
↪referencia el modelo lineal y modelo exponencial para identificar cual de␣
↪estos permite tener mejor predicción de Volumen de Basal"

 # Comparacion de los dos modelos
     # Resultado del Modelo 1.
     test_preds |> eval_metrics(truth = basal, estimate = .pred)

     # Resultado del Modelo 2.
     test_preds2 |>
       mutate(across(.pred, ~exp(.x)-3)) |> eval_metrics(truth = basal,␣
↪estimate = .pred)

     #Comparación de los gráficos
     coeficientes <- altdiam_fit |> extract_fit_parsnip() |>
       tidy()

     coeficientes2 <- altdiam_fit2 |> extract_fit_parsnip() |>
       tidy()

         # Grafico Modelo 1 (Lineal)
         test_preds |> ggplot(aes(x = count,
                         y = basal)) +
                        geom_point() +
                        geom_abline(intercept = coeficientes$estimate[1],
                                    slope = coeficientes$estimate[2],
                                    col = "red", lwd = 2)


         # Grafico Modelo 2 (Exponencial)
         test_preds2 |>
           # .pred ya está con la transformación de log
           # mutate(across(.pred, ~exp(.x)-3)) |>
           mutate(across(c(count,basal), ~log(.x+3))) |>
```

```
          ggplot(aes(x = count,
                     y = basal)) +
          geom_point() +
          geom_abline(intercept = coeficientes2$estimate[1],
                      slope = coeficientes2$estimate[2],
                      col = "red",
                      lwd = 2)


    # Observados vs predichos
        #Grafico Modelo 1 (Lineal)
        test_preds |> ggplot(aes(x = count, y = .pred)) +
                    geom_point() +
                    geom_abline(intercept = 0,
                                slope = 1,
                                col = "red",
                                lwd = 2) +
                    labs(x = "Conteo obs", y = "Basal pred")

        # Grafico Modelo 2 (Exponencial)
        test_preds2 |> ggplot(aes(x = count, y = exp(.pred)-3)) +
                      geom_point() +
                      geom_abline(intercept = 0,
                                  slope = 1,
                                  col = "red",
                                  lwd = 2)+
                      labs(x = "Conteo obs", y = "Basal pred")


  # Validación cruzada: Comparación de modelos
      # Asegurar reproducilidad
        set.seed(10)
        df3 <- dfbpq |> dplyr::select(basal, count)

        df_split <- initial_split(df3, prop = 0.6)
        df_train <- training(df_split)
        df_test <- testing(df_split)
        df_r <- vfold_cv(df_train,v = 10)

        df_r$splits

      # Ver resultados por fold.
        resuls <- workflow_set(preproc = list(simple = receta, log =␣
↪receta2), models = list(lm = mod_rl)) |>
          workflow_map(fn = "fit_resamples",
               # Opciones de `workflow_map()`:
               seed = 1101,
```

```
                verbose = TRUE,
                # Opciones de `fit_resamples()`:
                resamples = df_r)

      #Ver resultados de los flujos
        resuls

      # Ver resultados desagregados
        resuls |> filter(wflow_id == "simple_lm") |>
          select(result) |> unnest(result)

      #Ver resultados resumidos, calcula el promedio y error estándar.
        mfits <- resuls |> collect_metrics()
        mfits

      # Sacar mejor modelo de regresión simple. Finalizar el flujo de trabajo␣
↪y ajustar a datos de entrenamiento (ya sin la validación cruzada)y evaluar␣
↪sobre datos de entrenamiento.
        mejor <- resuls |> extract_workflow_set_result("simple_lm") %>%␣
↪select_best(metric = "rmse")

        mod <- resuls |>  extract_workflow("simple_lm") %>%
                          finalize_workflow(mejor) %>%
                          fit(data = df_train)

        mod |> augment(df_train) |>
              eval_metrics(truth = basal,estimate = .pred)


      # Ver el modelo 1 (Lineal)
      #Last_fit entrena sobre entrenamiento y evalua sobre test.
        mejor <- resuls |> extract_workflow_set_result("simple_lm") %>%␣
↪select_best(metric = "rmse")

        mod <- resuls |>extract_workflow("simple_lm") %>%␣
↪finalize_workflow(mejor) %>%
                        last_fit(split = df_split)

        test_preds <- mod  %>% collect_predictions()

        test_preds |> eval_metrics(truth = basal, estimate = .pred)


      # Ver el modelo 2 (Lineal)
        mejor <- resuls |> extract_workflow_set_result("log_lm") %>%␣
↪select_best(metric = "rmse")
```

```r
        mod <- resuls |> extract_workflow("log_lm") %>%␣
↪finalize_workflow(mejor) %>%
                        last_fit(split = df_split)

        test_preds <- mod  %>% collect_predictions()

        test_preds |> mutate(across(c(basal, .pred), ~ exp(.x)-3)) |>
                      eval_metrics(truth = basal, estimate = .pred)


 # Resultado: Como resultado final, es mejor utilizar el modelo de Regresión␣
↪Exponencial para predecir los datos de Basal,
 #           ya que presenta un RMSE de 2452 y RSQ_TRAD de -0.112.
```