# 6.-Vis._matriz_correlacion_y_regresión_lineal

Abel_Sánchez

2024-06-28

```r
# Configurar el directorio de trabajo
if (.Platform$OS.type == "windows") {
  setwd("C:/Users/FX506/Documents/Portafolio")
} else {
  setwd("~/Portafolio")
}

# Limpiar el entorno
rm(list = ls())
gc()
```

```
##          used (Mb) gc trigger (Mb) max used (Mb)
## Ncells 492958 26.4    1065221 56.9   686460 36.7
## Vcells 922659  7.1    8388608 64.0  1876652 14.4
```

```r
# Cargar las librerías
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```r
library(lmtest)
```

```
## Cargando paquete requerido: zoo
##
## Adjuntando el paquete: 'zoo'
##
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```r
library(gridExtra)
```

```
##
## Adjuntando el paquete: 'gridExtra'
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
# Definir la ruta de los archivos
files_folder <- "data/Fitbit/Processed_Files"
output_folder <- "output/fitbit"

# Crear el directorio de salida si no existe
if (!dir.exists(output_folder)) {
  dir.create(output_folder)
}

# Lista de archivos y sus respectivas columnas
file_info <- list(
  list(name = "filtered_dailyActivity_merged.csv", columns = c("Date_time", "TotalSteps")),
  list(name = "filtered_heartrate_seconds_merged.csv", columns = c("Date_time", "Value")),
  list(name = "filtered_weightLogInfo_merged.csv", columns = c("Date_time", "WeightKg")),
  list(name = "filtered_hourlyCalories_merged.csv", columns = c("Date_time", "Calories"))
)

# Leer cada archivo por separado, seleccionar columnas relevantes y resumir para evitar duplicados
data_list <- lapply(file_info, function(info) {
  read_csv(file.path(files_folder, info$name)) %>%
    select(any_of(info$columns)) %>%
    filter(!is.na(Date_time)) %>%
    group_by(Date_time) %>%
    summarise(across(everything(), mean, na.rm = TRUE)) %>%
    ungroup()
})
```

```
## Rows: 1397 Columns: 15
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl  (14): Id, TotalSteps, TotalDistance, TrackerDistance, LoggedActivitiesD...
## date  (1): Date_time
##
## 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.
```

```
## Warning: There was 1 warning in `summarise()`.
## i In argument: `across(everything(), mean, na.rm = TRUE)`.
## i In group 1: `Date_time = 2016-03-12`.
## Caused by warning:
## ! The ‘...‘ argument of ‘across()‘ is deprecated as of dplyr 1.1.0.
## Supply arguments directly to ‘.fns‘ through an anonymous function instead.
##
##   # Previously
##   across(a:b, mean, na.rm = TRUE)
##
##   # Now
##   across(a:b, \(x) mean(x, na.rm = TRUE))

## Rows: 3638339 Columns: 3
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl  (2): Id, Value
## date (1): Date_time
##
## 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.
## Rows: 100 Columns: 8
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl  (6): Id, WeightKg, WeightPounds, Fat, BMI, LogId
## lgl  (1): IsManualReport
## date (1): Date_time
##
## 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.
## Rows: 46183 Columns: 3
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl  (2): Id, Calories
## date (1): Date_time
##
## 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.
```

```r
# Eliminar elementos nulos de la lista de datos
data_list <- Filter(Negate(is.null), data_list)

# Verificar si hay datos en la lista
if (length(data_list) == 0) {
  stop("No se encontraron datos válidos para visualizar.")
}

# Fusionar los conjuntos de datos por Date_time
combined_data <- Reduce(function(x, y) full_join(x, y, by = "Date_time"), data_list)

# Verificar y eliminar filas con valores no finitos
combined_data <- combined_data %>%
  filter(across(everything(), is.finite))
```

```
## Warning: Using ‘across()‘ in ‘filter()‘ was deprecated in dplyr 1.0.8.
## i Please use ‘if_any()‘ or ‘if_all()‘ instead.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

```
# Normalización de los datos
normalize <- function(x) {
  return((x - min(x)) / (max(x) - min(x)))
}

combined_data <- combined_data %>%
  mutate(TotalSteps = normalize(TotalSteps),
         Calories = normalize(Calories),
         Value = normalize(Value),
         WeightKg = normalize(WeightKg))

# Segmentar datos por niveles de actividad de pasos
combined_data <- combined_data %>%
  mutate(ActivityLevel = case_when(
    TotalSteps < 0.33 ~ "Low Activity",
    TotalSteps >= 0.33 & TotalSteps < 0.66 ~ "Moderate Activity",
    TotalSteps >= 0.66 ~ "High Activity"
  ))

# Función para calcular correlación dentro de cada segmento
calculate_correlation <- function(data, segment) {
  segment_data <- data %>% filter(ActivityLevel == segment)
  return(cor(segment_data[, c("TotalSteps", "Calories", "Value", "WeightKg")], use = "complete.obs"))
}

# Calcular matriz de correlación para cada segmento
correlation_matrices <- lapply(unique(combined_data$ActivityLevel), function(segment) {
  calculate_correlation(combined_data, segment)
})

names(correlation_matrices) <- unique(combined_data$ActivityLevel)

# Imprimir matrices de correlación para cada segmento
print(correlation_matrices)
```

```
## $‘Moderate Activity‘
##             TotalSteps   Calories      Value   WeightKg
## TotalSteps   1.0000000 -0.2807578 -0.3272243  0.1181105
## Calories    -0.2807578  1.0000000 -0.1816402 -0.3995118
## Value       -0.3272243 -0.1816402  1.0000000  0.4526398
## WeightKg     0.1181105 -0.3995118  0.4526398  1.0000000
##
## $‘High Activity‘
##              TotalSteps   Calories       Value    WeightKg
## TotalSteps   1.00000000  0.5994484  0.05593782 -0.09474442
## Calories     0.59944838  1.0000000  0.27810540 -0.28966930
## Value        0.05593782  0.2781054  1.00000000 -0.15252065
## WeightKg    -0.09474442 -0.2896693 -0.15252065  1.00000000
##
## $‘Low Activity‘
##             TotalSteps Calories Value WeightKg
## TotalSteps          NA       NA    NA       NA
## Calories            NA       NA    NA       NA
## Value               NA       NA    NA       NA
## WeightKg            NA       NA    NA       NA
```

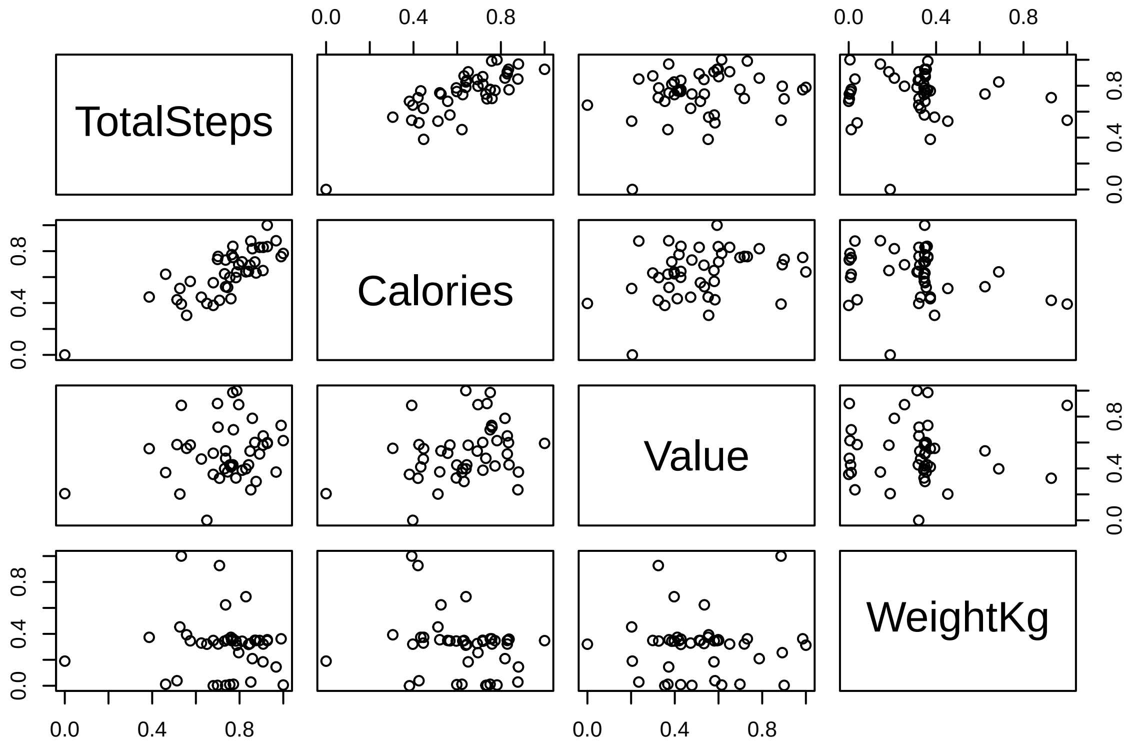
```
# Función para realizar regresión dentro de cada segmento
perform_regression <- function(data, segment) {
  segment_data <- data %>% filter(ActivityLevel == segment)
  lm_model <- lm(Calories ~ TotalSteps, data = segment_data)
  return(summary(lm_model))
}

# Realizar regresión para cada segmento
regression_results <- lapply(unique(combined_data$ActivityLevel), function(segment) {
  perform_regression(combined_data, segment)
})

names(regression_results) <- unique(combined_data$ActivityLevel)

# Imprimir resultados de regresión para cada segmento
print(regression_results)
```

```
## $`Moderate Activity`
##
## Call:
## lm(formula = Calories ~ TotalSteps, data = segment_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.14439 -0.06097 -0.02163  0.05186  0.13975
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.6379     0.2363   2.699   0.0307 *
## TotalSteps   -0.3376     0.4361  -0.774   0.4643
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.09894 on 7 degrees of freedom
## Multiple R-squared:  0.07882,    Adjusted R-squared:  -0.05277
## F-statistic: 0.599 on 1 and 7 DF,  p-value: 0.4643
##
##
## $`High Activity`
##
## Call:
## lm(formula = Calories ~ TotalSteps, data = segment_data)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.211629 -0.084382 -0.002691  0.091761  0.199456
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.06658    0.18085  -0.368 0.715198
## TotalSteps   0.93577    0.22088   4.237 0.000179 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1135 on 32 degrees of freedom
## Multiple R-squared:  0.3593, Adjusted R-squared:  0.3393
## F-statistic: 17.95 on 1 and 32 DF,  p-value: 0.0001794
##
##
## $`Low Activity`
##
## Call:
## lm(formula = Calories ~ TotalSteps, data = segment_data)
##
## Residuals:
## ALL 1 residuals are 0: no residual degrees of freedom!
##
## Coefficients: (1 not defined because of singularities)
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)        0        NaN     NaN      NaN
## TotalSteps        NA         NA      NA       NA
##
## Residual standard error: NaN on 0 degrees of freedom
```

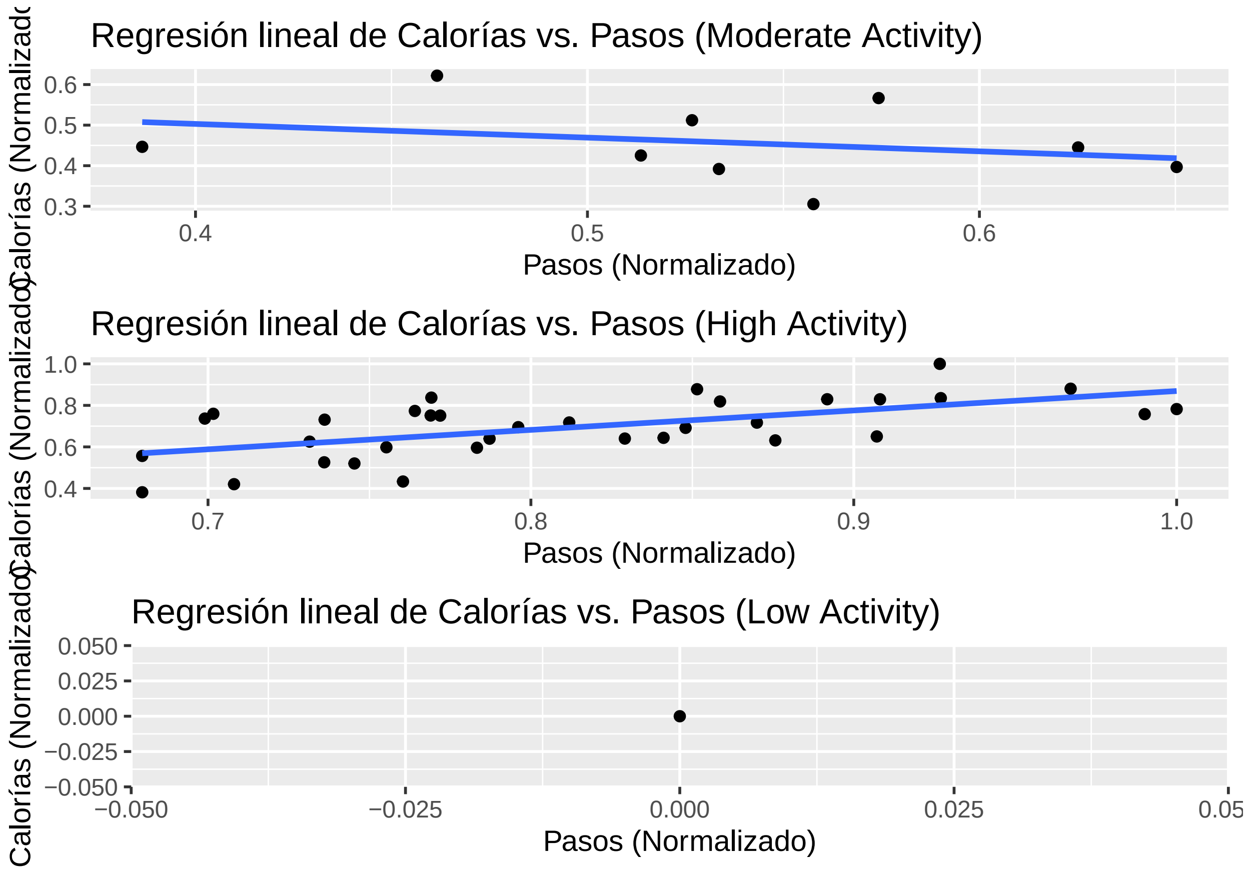
```
# Guardar los resultados de correlación y regresión en archivos separados
saveRDS(correlation_matrices, file = file.path(output_folder, "correlation_matrices.rds"))
saveRDS(regression_results, file = file.path(output_folder, "regression_results.rds"))

summary(cars)
```

```
##      speed           dist
##  Min.   : 4.0   Min.   :  2.00
##  1st Qu.:12.0   1st Qu.: 26.00
##  Median :15.0   Median : 36.00
##  Mean   :15.4   Mean   : 42.98
##  3rd Qu.:19.0   3rd Qu.: 56.00
##  Max.   :25.0   Max.   :120.00
```

## Including Plots



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


Regresión lineal de Calorías vs. Pasos (Moderate Activity)
Calorías (Normalizado)
0.3
0.4
0.5
0.6
0.4
0.5
0.6
Pasos (Normalizado)
Regresión lineal de Calorías vs. Pasos (High Activity)
Calorías (Normalizado)
0.4
0.6
0.8
1.0
0.7
0.8
0.9
1.0
Pasos (Normalizado)
Regresión lineal de Calorías vs. Pasos (Low Activity)
Calorías (Normalizado)
0.050
0.025
0.000
−0.025
−0.050
−0.050
−0.025
0.000
0.025
0.05
Pasos (Normalizado)