Anexo 1: Guía con lineamientos y orientaciones metodológicas para la imputación de datos estadísticos Instituto Nacional de Estadísticas

Departamento de Metodologías e Innovación Estadística
Subdepartamento de Investigación Estadística
Instituto Nacional de Estadística

Índice

1 Anexo: Definiciones y aplicaciones en R

1.1 Deiniciones

En el lenguaje de programación R, utilizamos herramientas como `tidyverse` y el paquete R de `naniar` para enseñar a manejar y analizar datos faltantes de manera efectiva. El paquete `naniar` es una herramienta muy útil para explorar, visualizar y manejar valores faltantes en R.

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.3     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.3     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## 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 errors
```

```
library(naniar)
```

La estadística Gertrude Mary Cox (@monroe1980memoriam) dijo una vez: "Lo mejor que se puede hacer con los datos faltantes es no tener ninguno". Si bien esto es cierto, no es el mundo en el que vivimos. Trabajar con datos del mundo real significa trabajar con datos faltantes. Para ser un gran analista, necesitamos saber cómo lidiar con los valores faltantes. Comprender cómo funcionan los datos faltantes es importante, ya que pueden tener efectos inesperados en tu análisis. Por ejemplo, ajustar un modelo lineal en datos con valores faltantes elimina fragmentos de datos. Esto significa que tus decisiones no se basan en la evidencia correcta. Reemplazar los valores faltantes, lo que se llama imputación, debe hacerse con mucho cuidado, ya que insertar solo la media puede llevar a estimaciones y decisiones deficientes.

En este apartado aprenderemos sobre qué son los valores faltantes, cómo encontrar datos faltantes, cómo manipular y limpiar datos faltantes, por qué faltan datos y cómo imputar valores faltantes. Por lo tanto, asumiremos que tenemoss experiencia básica a intermedia con R, experiencia en la creación de gráficos utilizando `ggplot2`, experiencia en el uso de `dplyr` para manipular datos y experiencia en ajustar modelos lineales en R.

¿Qué son los valores faltantes?

Antes de comenzar, debemos definir los valores faltantes. Los valores faltantes son valores que deberían haberse registrado, pero no lo fueron. Pensemos en esto de esta manera: puedes no haber registrado accidentalmente que viste un pájaro, esto es un valor faltante. Esto es diferente a registrar que no se observaron pájaros. `R` almacena los valores faltantes como `NA`, que significa no disponible.

¿Cómo puedo verificar si tengo valores faltantes?

Los valores faltantes no saltan y gritan "¡Estoy aquí!". Por lo general, están ocultos, como una aguja en un pajar. Para detectar valores faltantes, usaresmo `any_na`, que devuelve `TRUE` si hay valores faltantes y `FALSE` si no los hay. `are_na` pregunta "¿son estos `NA`?" y devuelve `TRUE/FALSE` para cada valor. are_na nos muestra 3 valores `TRUE`, que corresponden a 3 valores faltantes. Para evitar contar cada `TRUE` manualmente, `n_miss`

cuenta el número de valores faltantes. Y `prop_miss` proporciona la proporción de valores faltantes, lo que proporciona un contexto importante: Por ejemplo, el 50% de los datos está faltando.

¿Qué sucede cuando mezclamos valores faltantes con nuestros cálculos? Necesitamos saber qué sucede para poder estar preparados para encontrar estos casos. La regla general es: Los cálculos con `NA` devuelven `NA`. Digamos que tienes la altura de tres amigos: Sophie, Dan y Fred. La suma de sus alturas devuelve `NA`, esto se debe a que no conocemos la suma de un número y `NA`.

Hay algunas "trampas" que debes tener en cuenta al trabajar con datos faltantes: Por ejemplo, `NaN` significa "Not a Number" (No es un número) y se obtiene de operaciones como la raíz cuadrada de -1. **R** interpreta `NaN` como un valor faltante. `NULL` es un valor vacío pero no es faltante. Esto es sutilmente diferente de los valores faltantes: un cubo vacío no tiene agua faltante. `Inf` es un valor infinito, y se obtiene de ecuaciones como 10 dividido por 0 y no es faltante.

Por último, debeos tener cuidado con las declaraciones condicionales con valores faltantes. Por ejemplo, `NA` o `TRUE` es `TRUE`. `NA` o `FALSE` es `NA`. `NA + NaN` es `NA`. `NaN + NA` es `NaN`.

1.2 Usando y encontrando valores faltantes

Al trabajar con datos faltantes, hay algunos comandos con los que deberías estar familiarizado - en primer lugar, debes poder identificar si hay valores faltantes y dónde se encuentran.

Usando las herramientas `any_na()` y `are_na()`, identifica qué valores faltan.

```
# creamos x, un vector, con valores NA, NaN, Inf, ".", y "missing"
x <- c(NA, NaN,Inf, ".", "missing")

# Usamos any_na() y are_na() para explorar los valores missings
any_na(x)
```

```
## [1] TRUE
```

```
are_na(x)
```

```
## [1]  TRUE FALSE FALSE FALSE FALSE
```

¿Cuántos valores faltantes hay?

Una de las primeras cosas que desearás comprobar en un nuevo conjunto de datos es si existen valores faltantes y cuántos hay.

Podrías usar `are_na()` y contar los valores faltantes, pero la forma más eficiente de contarlos es usar la función `n_miss()`. Esto te dirá el número total de valores faltantes en los datos.

Luego puedes encontrar el porcentaje de valores faltantes en los datos con la función `pct_miss`. Esto te dirá el porcentaje de valores faltantes en los datos.

También puedes encontrar el complemento de estos valores, cuántos valores completos hay, usando `n_complete` y `pct_complete`.

```
# Usando el dataframe de ejemplo de estatuta (heights) y pesos (weights) dat_hw
dat_hw <- read.table("dealing/dat_hw.txt", h=T,  dec=".")
```

```
# Usa n_miss() para contar el numero total de valores missing en dat_hw
naniar::n_miss(dat_hw)
```

```
## [1] 30
```

```
# Usa n_miss() en dat_hw$weight para contar el numero total de valores missing
naniar::n_miss(dat_hw$weight)
```

```
## [1] 15
```

```
# Usa n_complete() en dat_hw para contar el numero total de valores completos
n_complete(dat_hw)
```

```
## [1] 170
```

```
# Usa n_complete() en dat_hw$weight para contar el numero total de valores completos
n_complete(dat_hw$weight)
```

```
## [1] 85
```

```
# Usamos prop_miss() y prop_complete() en dat_hw para contar el numero total de valores missing en cada
prop_miss(dat_hw)
```

```
## [1] 0.15
```

```
prop_complete(dat_hw)
```

```
## [1] 0.85
```

Resumiendo la ausencia de datos

Ahora que comprendes el comportamiento de los valores faltantes en R y cómo contarlos, escalaremos nuestros resúmenes para casos (filas) y variables, utilizando `miss_var_summary()` y `miss_case_summary()`, y también exploraremos cómo se pueden aplicar a grupos en un dataframe utilizando la función `group_by` de `dplyr`.

```
# Summarise missingness in each variable of the `airquality` dataset
miss_var_summary(airquality)
```

```
## # A tibble: 6 x 3
##   variable n_miss pct_miss
##   <chr>     <int>    <dbl>
## 1 Ozone        37    24.2
## 2 Solar.R       7     4.58
## 3 Wind          0     0
## 4 Temp          0     0
## 5 Month         0     0
## 6 Day           0     0
```

```
# Summarise missingness in each case of the `airquality` dataset
miss_case_summary(airquality)
```

```
## # A tibble: 153 x 3
##     case n_miss pct_miss
##    <int>  <int>    <dbl>
##  1     5      2     33.3
##  2    27      2     33.3
##  3     6      1     16.7
##  4    10      1     16.7
##  5    11      1     16.7
##  6    25      1     16.7
##  7    26      1     16.7
##  8    32      1     16.7
##  9    33      1     16.7
## 10    34      1     16.7
## # i 143 more rows
```

```
# Return the summary of missingness in each variable, grouped by Month, in the `airquality` dataset
airquality %>% group_by(Month) %>% miss_var_summary()
```

```
## # A tibble: 25 x 4
## # Groups:   Month [5]
##    Month variable n_miss pct_miss
##    <int> <chr>     <int>    <dbl>
##  1     5 Ozone         5     16.1
##  2     5 Solar.R       4     12.9
##  3     5 Wind          0      0
##  4     5 Temp          0      0
##  5     5 Day           0      0
##  6     6 Ozone        21     70
##  7     6 Solar.R       0      0
##  8     6 Wind          0      0
##  9     6 Temp          0      0
## 10     6 Day           0      0
## # i 15 more rows
```

```
# Return the summary of missingness in each case, grouped by Month, in the `airquality` dataset
airquality %>% group_by(Month) %>% miss_case_summary()
```

```
## # A tibble: 153 x 4
## # Groups:   Month [5]
##    Month  case n_miss pct_miss
##    <int> <int>  <int>    <dbl>
##  1     5     5      2       40
##  2     5    27      2       40
##  3     5     6      1       20
##  4     5    10      1       20
##  5     5    11      1       20
##  6     5    25      1       20
##  7     5    26      1       20
##  8     5     1      0        0
##  9     5     2      0        0
## 10     5     3      0        0
## # i 143 more rows
```

1.3 Evaluación de la no respuesta

1.3.1 Actividad

Considerando el siguien set de datos.

```
library(dplyr)
library(ggplot2)
biopics <- read.csv("curso_imputacion/biopics.csv")
```

Muestra las primeras 10 observaciones de los datos `biopics` y familiarízate con las variables.

```
# Muestra las primeras 10 observaciones
head(biopics, 10)
```

```
##    country year earnings sub_num sub_type sub_race non_white sub_sex
## 1       UK 1971       NA       1 Criminal     <NA>         0    Male
## 2    US/UK 2013   56.700       1    Other  African         1    Male
## 3    US/UK 2010   18.300       1  Athlete     <NA>         0    Male
## 4   Canada 2014       NA       1    Other    White         0    Male
## 5       US 1998    0.537       1    Other     <NA>         0    Male
## 6       US 2008   81.200       1    Other    other         1    Male
```

```
## 7       UK 2002    1.130       1 Musician    White         0    Male
## 8       US 2013   95.000       1  Athlete  African         1    Male
## 9       US 1994   19.600       1  Athlete     <NA>         0    Male
## 10   US/UK 1987    1.080       2   Author     <NA>         0    Male
```

```
# Obtiene el numero de valores perdidos por variable
biopics %>%
    is.na() %>%
    colSums()
```

```
##   country      year  earnings   sub_num  sub_type  sub_race non_white   sub_sex
##         0         0       324         0         0       197         0         0
```

1.3.2 Reconociendo los mecanismos de datos faltantes

En este ejercicio, se presentarán seis escenarios diferentes en los que faltan algunos datos. Intenta asignar a cada uno de ellos el mecanismo de datos faltantes más probable. Como recordatorio, aquí hay algunas pautas generales:

Si la razón de la falta de datos es puramente aleatoria, es MCAR. Si la razón de la falta de datos puede explicarse por otra variable, es MAR. Si la razón de la falta de datos depende del valor faltante en sí mismo, es MNAR.

Pérdida Completamente Aleatoria (MCAR)	Pérdida Aleatoria (MAR)	Pérdida no Aleatoria (NMAR)
Mientras se etiquetaba manualmente los datos, el etiquetador accidentalmente dejó algunas entradas sin etiquetar.	En una encuesta de salud, se observan datos faltantes en el peso. Se sospecha que los valores de la variable de peso faltan más para un género que para otro.	Es común que los simpatizantes de la extrema derecha no lo admitan en las encuestas electorales.
En un conjunto de datos que contiene resultados de exámenes escolares, algunos niños no tienen el resultado porque estaban enfermos y no asistieron al examen.	Está realizando el seguimiento de las ubicaciones de los visitantes de un sitio web. Si están utilizando una VPN (lo cual se sabe), el seguimiento no es confiable y a menudo se registran valores faltantes.	En las encuestas, las personas ricas tienen más probabilidades de no revelar sus ingresos.

Figura 1: alt text

1.3.2.0.1 Prueba t para perdida MAR Preparación de los datos

De los tres, MAR es posiblemente el más importante de detectar, ya que muchos métodos de imputación asumen que los datos son MAR. En este ejercicio práctico con `R`, buscaremos identificar si el patrón de pérdida es MAR.

Trabajaremos con los datos de la base `biopics`. El objetivo es probar si el número de valores faltantes en `earnings` difiere por género del sujeto. En este ejercicio, solo se preparan los datos para aplicar una *prueba t*. Primero, se crea una variable ficticia que indica la falta de datos en `earnings`. Luego, se divide por género filtrando los datos para mantener uno de los géneros y luego sacando la variable ficticia. Para filtrar, puede ser útil imprimir el `head()` de `biopics` en la consola y examinar la variable de género.

```
# Crea una variable dummy para la perdida en el gasto
biopics <- biopics %>%
  mutate(missing_earnings = is.na(earnings))

# Obtiene la perdida del gasto para hombres
missing_earnings_males <- biopics %>%
  filter(sub_sex == "Male") %>%
```

```
  pull(missing_earnings)

# Obtiene la perdida del gasto para mujeres
missing_earnings_females <- biopics %>%
  filter(sub_sex == "Female") %>%
  pull(missing_earnings)
```

Interpretación

En el ejercicio anterior, hemos preparado dos vectores con los valores faltantes de ingresos para cada sexo: `perdidos_gastos_hombres` y `perdidos_gastos_mujeres`. Ambos están disponibles en tu espacio de trabajo. Ahora es posible realizar la **prueba t** para verificar si sus medias difieren significativamente entre sí con el siguiente script

```
# Ejecuta el t-test
t.test(missing_earnings_males, missing_earnings_females)
```

```
##
##  Welch Two Sample t-test
##
## data:  missing_earnings_males and missing_earnings_females
## t = 1.1116, df = 294.39, p-value = 0.2672
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.03606549  0.12969214
## sample estimates:
## mean of x mean of y
## 0.4366438 0.3898305
```

El resultado muestra que no existe diferencia estadísticamente significativa ($\alpha > 0.05$) entre ambos grupos. Por lo tanto, se concluye que la perdida es MAR.

1.3.3 Aggregation plot

El gráfico de agregación proporciona la respuesta a la pregunta básica que uno puede hacer sobre un conjunto de datos incompleto: ¿en qué combinaciones de variables faltan datos y con qué frecuencia? Es muy útil para obtener una visión general de alto nivel de los patrones de ausencia de datos. Por ejemplo, hace visible inmediatamente si hay alguna combinación de variables que faltan juntas con frecuencia, lo que podría sugerir alguna relación entre ellas.

En este ejercicio, primero aplicaremos el gráfico de agregación para los datos de `biopics` y luego practicarás sacando conclusiones basadas en él. ¡Vamos a hacer algunos gráficos!

```
# Load the VIM package
library(VIM)
```

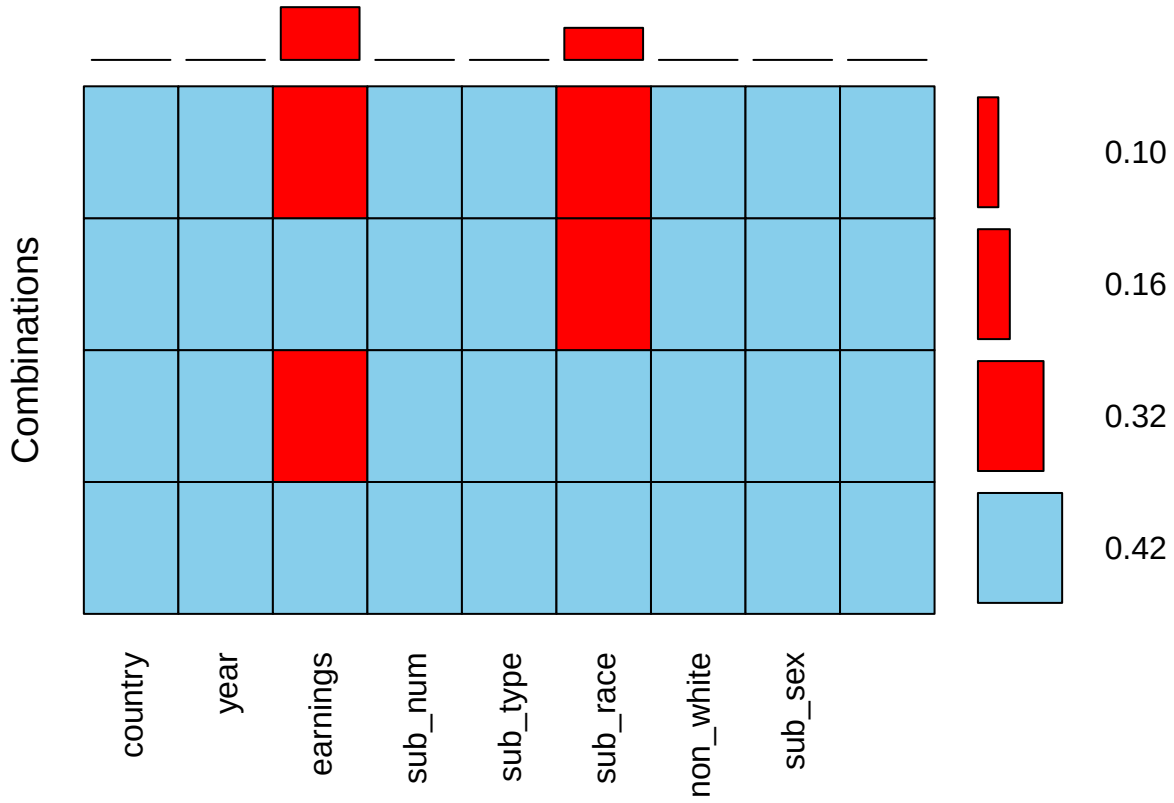
```
## Loading required package: colorspace
```

```
## Loading required package: grid
```

```
## The legacy packages maptools, rgdal, and rgeos, underpinning the sp package,
## which was just loaded, will retire in October 2023.
## Please refer to R-spatial evolution reports for details, especially
## https://r-spatial.org/r/2023/05/15/evolution4.html.
## It may be desirable to make the sf package available;
## package maintainers should consider adding sf to Suggests:.
## The sp package is now running under evolution status 2
##      (status 2 uses the sf package in place of rgdal)
```

```
## VIM is ready to use.
## Suggestions and bug-reports can be submitted at: https://github.com/statistikat/VIM/issues
##
## Attaching package: 'VIM'
## The following object is masked from 'package:datasets':
##
##     sleep
```

```
# Draw an aggregation plot of biopics
biopics %>%
    aggr(combined = TRUE, numbers = TRUE)
```



1.3.4 Cuestiones aclaratorias

Basado en el gráfico de agregación que acaba de crear, ¿cuál de las siguientes afirmaciones es falsa?

Posibles respuestas:

a. El 10% de las observaciones tienen valores faltantes tanto en `earnings` como en `sub_race`.

b. Hay más valores faltantes en `sub_race` que en `earnings`.

c. El 42% de las observaciones no tiene entradas faltantes.

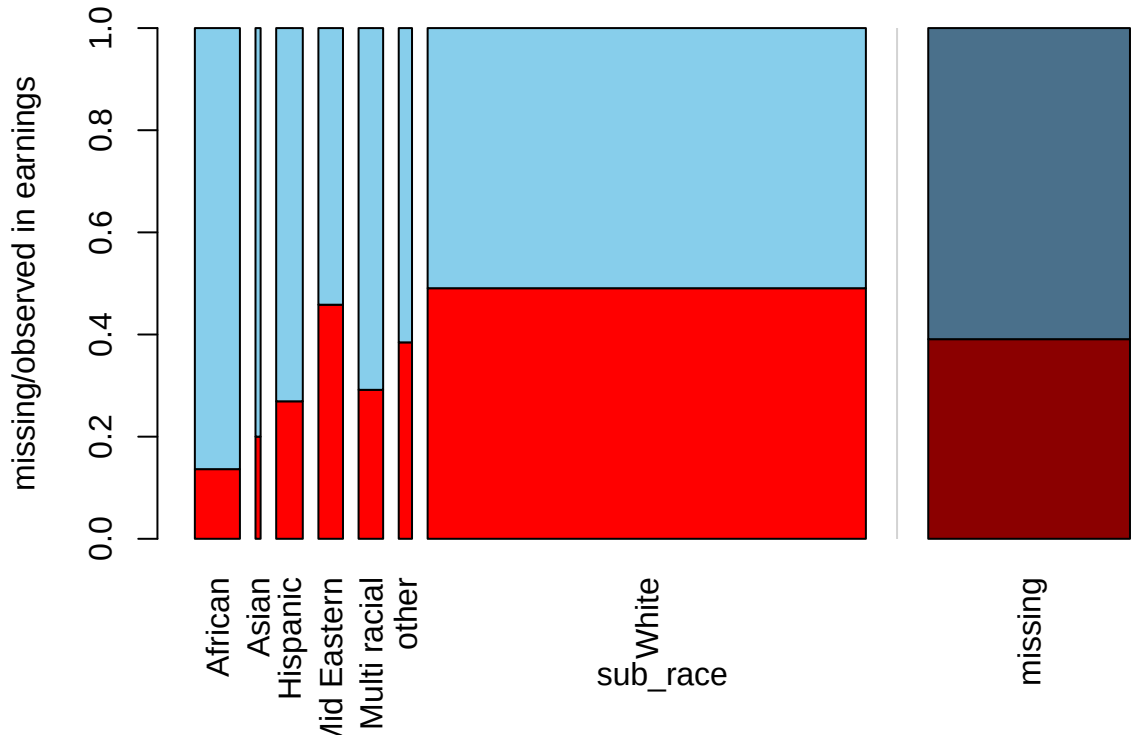
d. Exactamente dos variables en los datos `biopics` tienen valores faltantes.

1.3.5 gráfico de Mosaico

El gráfico de agregación que has dibujado en el ejercicio anterior te dio una visión general de alto nivel de los datos faltantes. Si estás interesado en la interacción entre variables específicas, un gráfico de Mosaico es el camino a seguir. Te permite estudiar el porcentaje de valores faltantes en una variable para diferentes valores de la otra, lo cual es conceptualmente muy similar a los test t que has estado realizando en la lección anterior.

En este ejercicio, dibujarás un gráfico de Mosaico para investigar el porcentaje de datos faltantes en `earnings` para diferentes categorías de `sub_race`. ¿Hay más datos faltantes en `earnings` para algunas razas específicas del personaje principal de la película? ¡Vamos a descubrirlo! El paquete `VIM` ya ha sido cargado para ti.

```
# Draw a spine plot to analyse missing values in earnings by sub_race

biopics %>%
    dplyr::select(sub_race,earnings) %>%
    spineMiss()
```



Cuestiones aclaratorias

Basándose en la gráfica de Mosaico que acabas de crear, ¿cuál de las siguientes afirmaciones es falsa?

Opciones de respuesta:

a. En la gran mayoría de las películas, el personaje principal es blanco.

b. Cuando el sujeto principal es africano, es más probable que tengamos información completa sobre las ganancias.

c. En lo que respecta a las ganancias y la subraza, los datos parecen ser MAR.

d. La raza que aparece con menos frecuencia en los datos tiene alrededor del 40% de las ganancias faltantes.

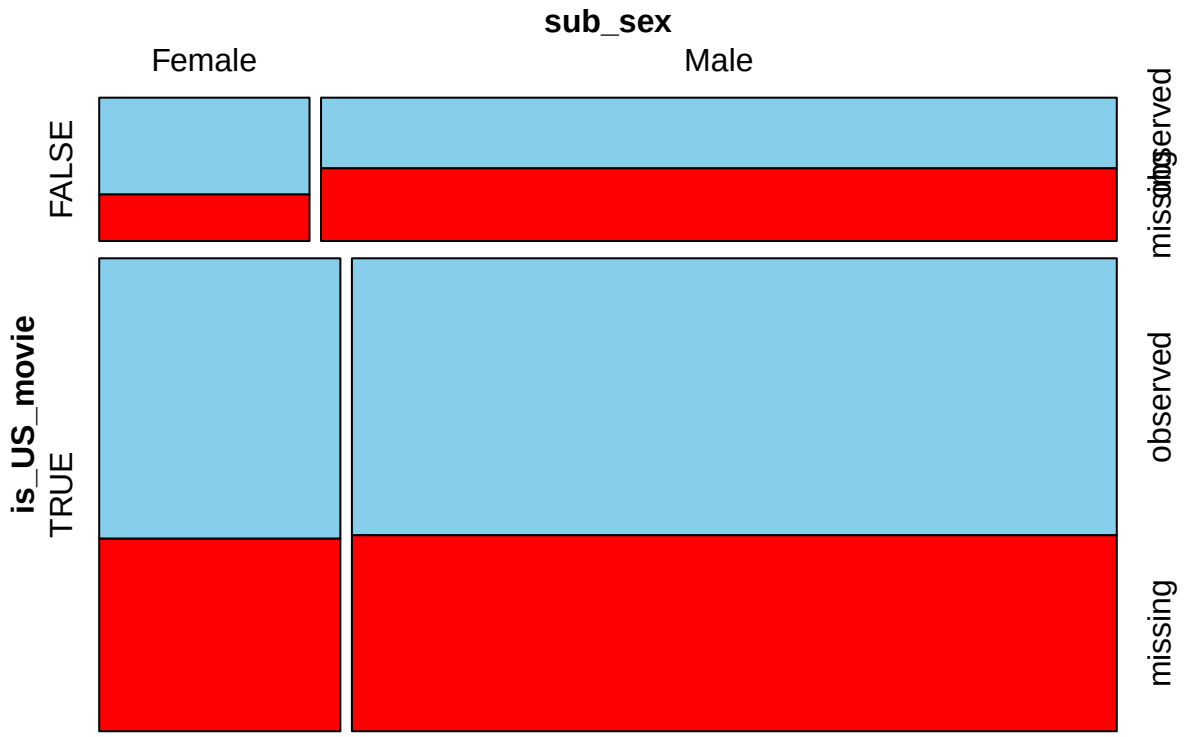
(incorrecta)

1.3.6 Mosaic plot

La gráfica de Mosaico que hemos creado en el ejercicio anterior permite estudiar los patrones de datos faltantes entre dos variables a la vez. Esta idea se generaliza a más variables en forma de un gráfico de mosaico.

En este ejercicio, comenzarás por crear una variable ficticia que indique si Estados Unidos participó en la producción de cada película. Para hacer esto, utilizarás la función `grepl()`, que verifica si la cadena pasada como su primer argumento está presente en el objeto pasado como su segundo argumento. Luego, crearemos un gráfico de mosaico para ver si el género del sujeto se correlaciona con la cantidad de datos faltantes en `earnings` tanto para películas estadounidenses como no estadounidenses.

```
# Prepare data for plotting and draw a mosaic plot
biopics %>%
    # Create a dummy variable for US-produced movies
    mutate(is_US_movie = grepl("US", country)) %>%
    # Draw mosaic plot
    mosaicMiss(highlight = "earnings",
               plotvars = c("is_US_movie", "sub_sex"))
```



1.3.7 Olfateando el peligro de la imputación por la media

Uno de los métodos de imputación más populares es la imputación por media, en la cual los valores faltantes en una variable se reemplazan con la media de los valores observados en esa variable. Sin embargo, en muchos casos, este enfoque simple es una mala elección. A veces, una mirada rápida a los datos puede alertarnos sobre los peligros de la imputación por la media.

En esta etapa, trabajaremos con una submuestra de los datos del proyecto de *Atmósfera Tropical Oceánica* (`tao`). El conjunto de datos consiste en mediciones atmosféricas tomadas en dos períodos de tiempo diferentes en cinco ubicaciones distintas. Los datos vienen con el paquete `VIM`.

En este ejercicio, nos familiarizaremos con los datos y realizaremos un análisis simple que indicará cuáles podrían ser las consecuencias de la imputación por la media.

```
data(tao, package = "VIM")
names(tao)<-tolower(names(tao))
names(tao)<-sub("[.]", "_", names(tao))
names(tao)<-sub("[.]", "_", names(tao))

# Imprime las primeras 10 observaciones
head(tao, 10)
```

```
##    year latitude longitude sea_surface_temp air_temp humidity uwind vwind
## 1  1997        0      -110            27.59    27.15     79.6  -6.4   5.4
## 2  1997        0      -110            27.55    27.02     75.8  -5.3   5.3
## 3  1997        0      -110            27.57    27.00     76.5  -5.1   4.5
## 4  1997        0      -110            27.62    26.93     76.2  -4.9   2.5
## 5  1997        0      -110            27.65    26.84     76.4  -3.5   4.1
## 6  1997        0      -110            27.83    26.94     76.7  -4.4   1.6
## 7  1997        0      -110            28.01    27.04     76.5  -2.0   3.5
## 8  1997        0      -110            28.04    27.11     78.3  -3.7   4.5
## 9  1997        0      -110            28.02    27.21     78.6  -4.2   5.0
## 10 1997        0      -110            28.05    27.25     76.9  -3.6   3.5
```

```
# Obtiene el numero de valores perdidos por columna
tao %>%
  is.na() %>%
  colSums()
```

```
##             year         latitude        longitude sea_surface_temp
##                0                0                0                3
##         air_temp         humidity            uwind            vwind
##               81               93                0                0
```

```
# Calculate the number of missing values in air_temp per year
tao %>%
  group_by(year) %>%
  summarize(num_miss = sum(is.na(air_temp)))
```

```
## # A tibble: 2 x 2
##    year num_miss
##   <int>    <int>
## 1  1993        4
## 2  1997       77
```

1.3.8 Imputación de la media en temperatura

Imputar la media en la temperatura puede ser arriesgado. Si la variable que se está imputando está correlacionada con otras variables, esta correlación podría ser destruida por los valores imputados. Lo viste en el ejercicio anterior cuando analizaste la variable `air_temp`.

Para averiguar si estas preocupaciones son válidas, en este ejercicio realizarás una imputación de la media en `air_temp`, creando también un indicador binario para mostrar dónde se imputan los valores. Será útil en el siguiente ejercicio, cuando evaluarás el desempeño de tu imputación. ¡Vamos a completar esos valores faltantes!

```r
tao_imp <- tao %>%
  # Create a binary indicator for missing values in air_temp
  mutate(air_temp_imp = ifelse(is.na(air_temp), TRUE, FALSE)) %>%
  # Impute air_temp with its mean
  mutate(air_temp = ifelse(is.na(air_temp), mean(air_temp, na.rm = TRUE), air_temp))

# Print the first 10 rows of tao_imp
head(tao_imp, 10)
```

```
##    year latitude longitude sea_surface_temp air_temp humidity uwind vwind
## 1  1997        0      -110            27.59    27.15     79.6  -6.4   5.4
## 2  1997        0      -110            27.55    27.02     75.8  -5.3   5.3
## 3  1997        0      -110            27.57    27.00     76.5  -5.1   4.5
## 4  1997        0      -110            27.62    26.93     76.2  -4.9   2.5
## 5  1997        0      -110            27.65    26.84     76.4  -3.5   4.1
## 6  1997        0      -110            27.83    26.94     76.7  -4.4   1.6
## 7  1997        0      -110            28.01    27.04     76.5  -2.0   3.5
## 8  1997        0      -110            28.04    27.11     78.3  -3.7   4.5
## 9  1997        0      -110            28.02    27.21     78.6  -4.2   5.0
## 10 1997        0      -110            28.05    27.25     76.9  -3.6   3.5
##    air_temp_imp
## 1         FALSE
## 2         FALSE
## 3         FALSE
## 4         FALSE
## 5         FALSE
## 6         FALSE
## 7         FALSE
## 8         FALSE
## 9         FALSE
## 10        FALSE
```

```r
head(filter(tao_imp, air_temp_imp==TRUE))
```

```
##   year latitude longitude sea_surface_temp air_temp humidity uwind vwind
## 1 1997        0       -95            27.69 25.02925     79.8  -0.6   4.2
## 2 1997        0       -95            27.63 25.02925     74.5  -3.9   5.8
## 3 1997        0       -95            27.51 25.02925     76.3  -2.8   5.3
## 4 1997        0       -95            27.54 25.02925     81.6  -2.3   5.6
## 5 1997        0       -95            27.47 25.02925     81.9  -4.6   6.1
## 6 1997        0       -95            27.44 25.02925     74.2  -4.4   6.5
##   air_temp_imp
## 1         TRUE
## 2         TRUE
## 3         TRUE
## 4         TRUE
## 5         TRUE
## 6         TRUE
```

Nos damos cuenta que no tiene mucho sentido imputar por la media, ya que puede agregar inconsistencias entre las variables correlacionadas.

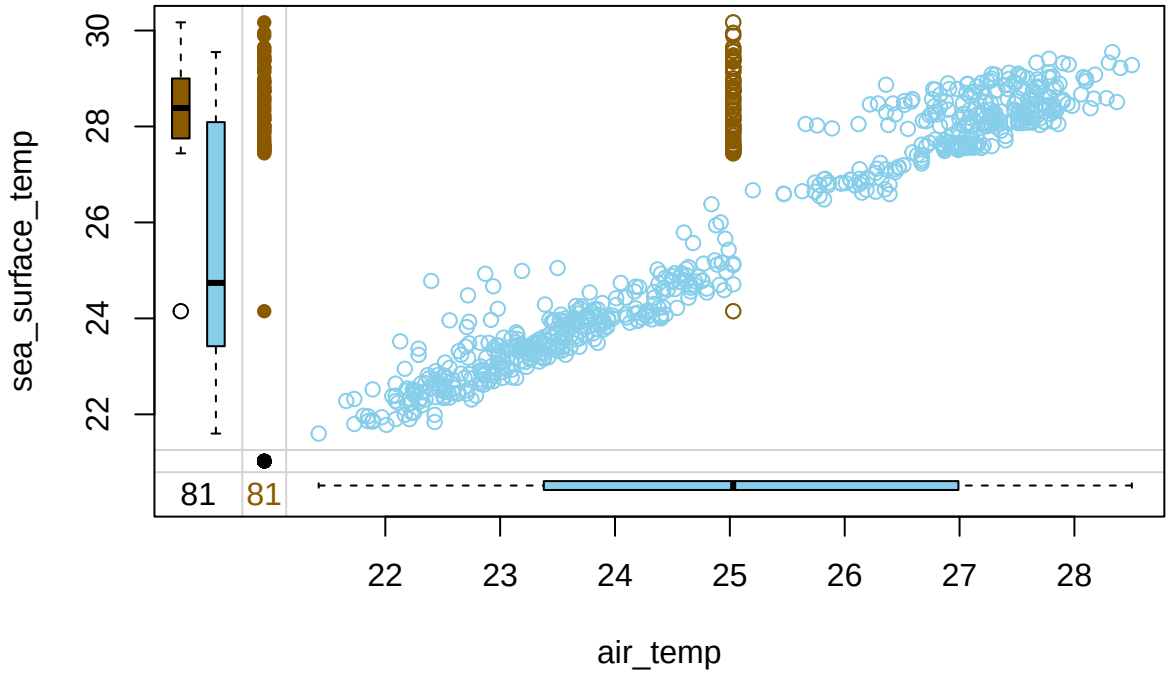
1.3.9 Evaluar la calidad de la imputación con un marginplot

En el último ejercicio, hemos imputado la media de `air_temp` y hemos agregado una variable indicadora para denotar cuáles valores fueron imputados, llamada `air_temp_imp`. Ahora es momento de ver qué tan bien funciona esto.

Al examinar los datos de `tao`, podríamos haber notado que también contiene una variable llamada `sea_surface_temp`, que razonablemente se esperaría que esté positivamente correlacionada con `air_temp`. Si ese es el caso, esperaríamos que estas dos temperaturas sean altas o bajas al mismo tiempo. Imputar la temperatura media del aire cuando la temperatura del mar es alta o baja rompería esta relación.

Para averiguarlo, en este ejercicio seleccionaremos las dos variables de temperatura y la variable indicadora y las usaremos para crear un `marginplot`.

```
# Creamos un marginplot de air_temp vs sea_surface_temp
tao_imp %>%
  select(air_temp, sea_surface_temp, air_temp_imp) %>%
  marginplot(delimiter = "imp")
```



1.3.10 Imputación por hot-deck

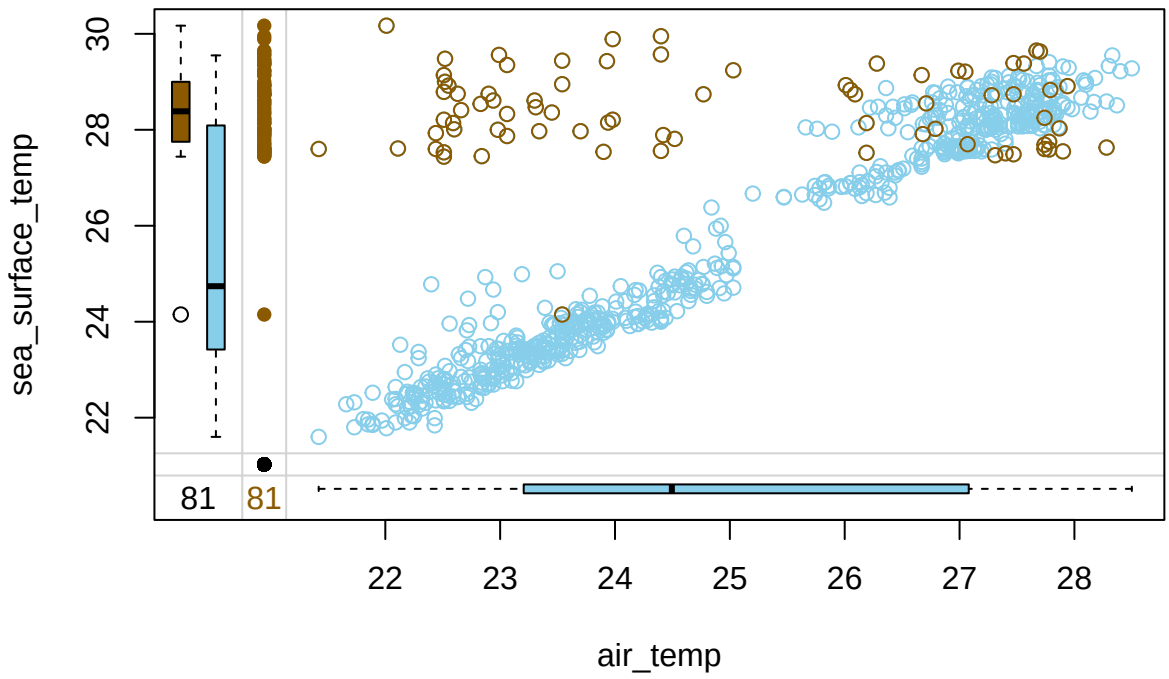
La imputación por hot-deck es un método simple que reemplaza cada valor faltante en una variable por el último valor observado en esa variable. Es muy rápido, ya que solo se necesita una revisión por los datos, pero en su forma más simple, hot-deck a veces puede romper las relaciones entre las variables.

En este ejemplo, lo probaremos en el conjunto de datos `tao`. Imputaremos los valores faltantes en la columna de temperatura del aire `air_temp` por hot-deck y luego visualizaremos un gráfico de margen (`marginplot`) para analizar la relación entre los valores imputados y la columna de temperatura de la superficie del mar `sea_surface_temp`.

```
# Load VIM package
library(VIM)

# Impute air_temp in tao with hot-deck imputation
tao_imp <- hotdeck(tao, variable = "air_temp")

# Check the number of missing values in each variable
tao_imp %>%
    is.na() %>%
    colSums()
```

```
##             year         latitude        longitude sea_surface_temp
##                0                0                0                3
##         air_temp         humidity            uwind            vwind
##                0               93                0                0
##     air_temp_imp
##                0
```

```
# Draw a margin plot of air_temp vs sea_surface_temp
tao_imp %>%
    select(air_temp, sea_surface_temp, air_temp_imp) %>%
    marginplot(delimiter = "imp")
```



¿Se ve bien la imputación? Observa las observaciones en la parte superior izquierda del gráfico con los datos de `air_temp` imputados y los valores altos en `sea_surface_temp`. Estas observaciones deben haber sido precedidas por observaciones con bajos valores de `air_temp` en el `data frame`, y por lo tanto, después de la imputación hot-deck, terminaron siendo valores atípicos con `air_temp` bajos y `sea_surface_temp` altos.

1.3.11 Hot-deck trucos y consejos I: imputando dentro de dominios

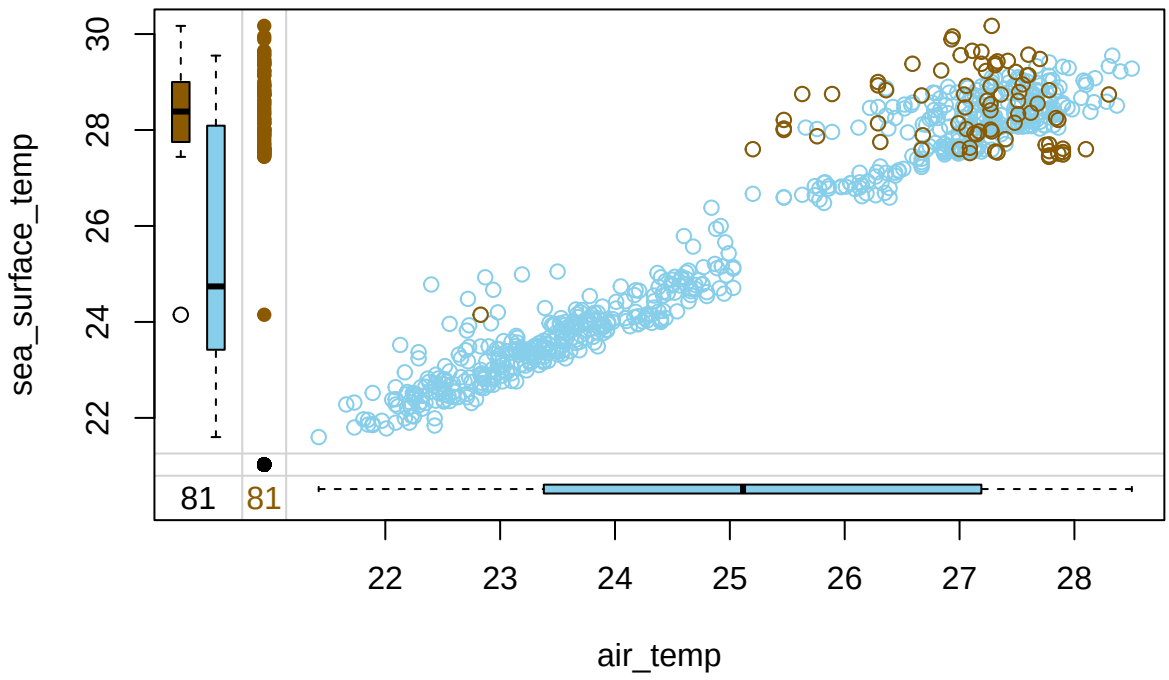
Un truco que puede ayudar cuando la imputación por hot-deck rompe las relaciones entre las variables es imputar dentro de los dominios. Esto significa que si la variable a imputar está correlacionada con otra variable categórica, se puede ejecutar hot-deck por separado para cada una de sus categorías.

Por ejemplo, se podría esperar que la temperatura del aire dependa del tiempo, ya que estamos viendo que las temperaturas promedio aumentan debido al calentamiento global. El indicador de tiempo que tenemos disponible en los datos de `tao` es una variable categórica, `year`. Primero, comprobaremos si la temperatura media del aire es diferente en cada uno de los dos años estudiados y luego ejecutaremos hot-deck dentro de los dominios de los años. Finalmente, volveremos a crear el `marginplot` para evaluar el rendimiento de la imputación.

```
# Calculate mean air_temp per year
tao %>%
    group_by(year) %>%
    summarize(average_air_temp = mean(air_temp, na.rm = TRUE))
```

```
## # A tibble: 2 x 2
##    year average_air_temp
##   <int>            <dbl>
## 1  1993             23.4
## 2  1997             27.1
```

```
# Hot-deck-impute air_temp in tao by year domain
tao_imp <- hotdeck(tao, variable = "air_temp", domain_var = "year")

# Draw a margin plot of air_temp vs sea_surface_temp
tao_imp %>%
    select(air_temp, sea_surface_temp, air_temp_imp) %>%
    marginplot(delimiter = "imp")
```



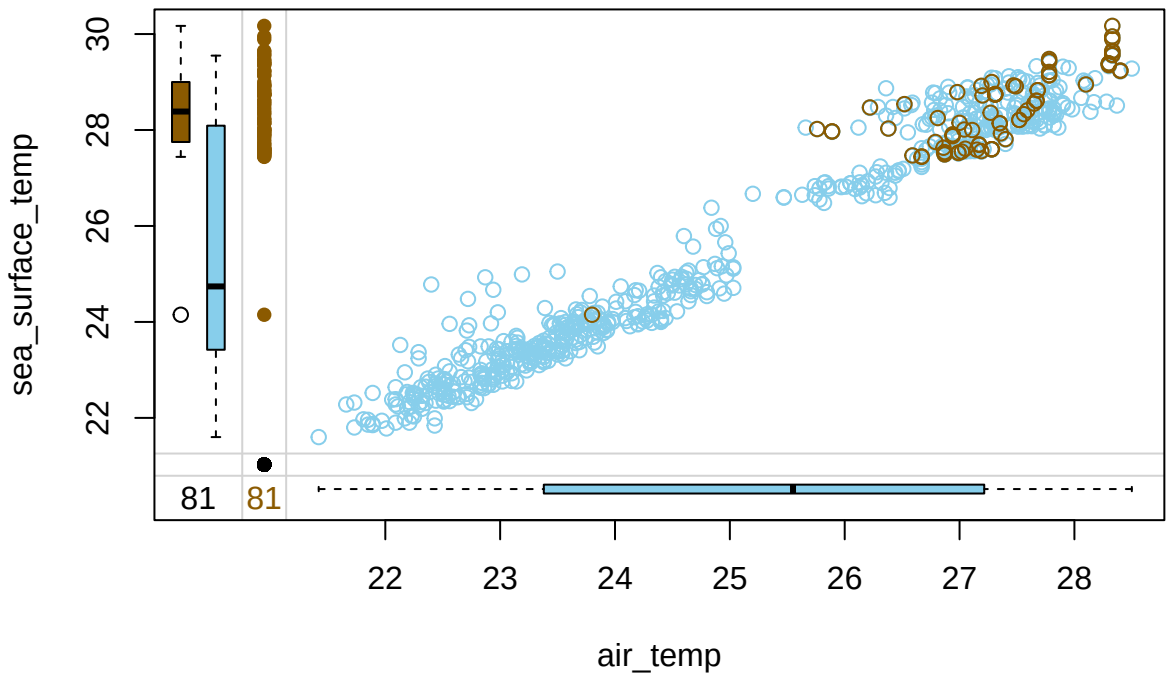
Los resultados se ven mucho mejor esta vez. Sin embargo, si observas la esquina superior derecha del gráfico, verás que la varianza en los valores imputados (naranja) es algo mayor que entre los valores observados (azul). Veamos si podemos mejorar aún más en el próximo ejercicio.

1.3.12 Hot-deck trucos y consejos II: ordenando por variables correlacionadas

Otro truco que puede mejorar el rendimiento de la imputación hot-deck es ordenar los datos por variables correlacionadas con la que queremos imputar.

Por ejemplo, en todos los `marginplot` que hemos estado usando recientemente, se ha visto que la temperatura del aire está fuertemente correlacionada con la temperatura de la superficie del mar, lo cual tiene mucho sentido. Podemos aprovechar este conocimiento para mejorar la imputación hot-deck. Si primero ordenamos los datos por `sea_surface_temp`, entonces cada valor imputado de `air_temp` vendrá de un donante con una `sea_surface_temp` similar.

```
# Hot-deck-impute air_temp in tao ordering by sea_surface_temp
tao_imp <- hotdeck(tao, variable = "air_temp", ord_var = "sea_surface_temp")

# Draw a margin plot of air_temp vs sea_surface_temp
tao_imp %>%
    select(air_temp, sea_surface_temp, air_temp_imp) %>%
    marginplot(delimiter = "imp")
```



Esta vez la imputación parece no afectar la relación entre las temperaturas del aire y la superficie del mar: si no fuera por los colores, probablemente no sabriamos cuáles son los valores imputados. La imputación hot-deck, posiblemente mejorada con la imputación por dominios o el ordenamiento, es un método rápido y sencillo que puede funcionar bien en muchas situaciones. Sin embargo, a veces puede ser necesario un enfoque más complejo.

1.3.13 Elegir el número de vecinos

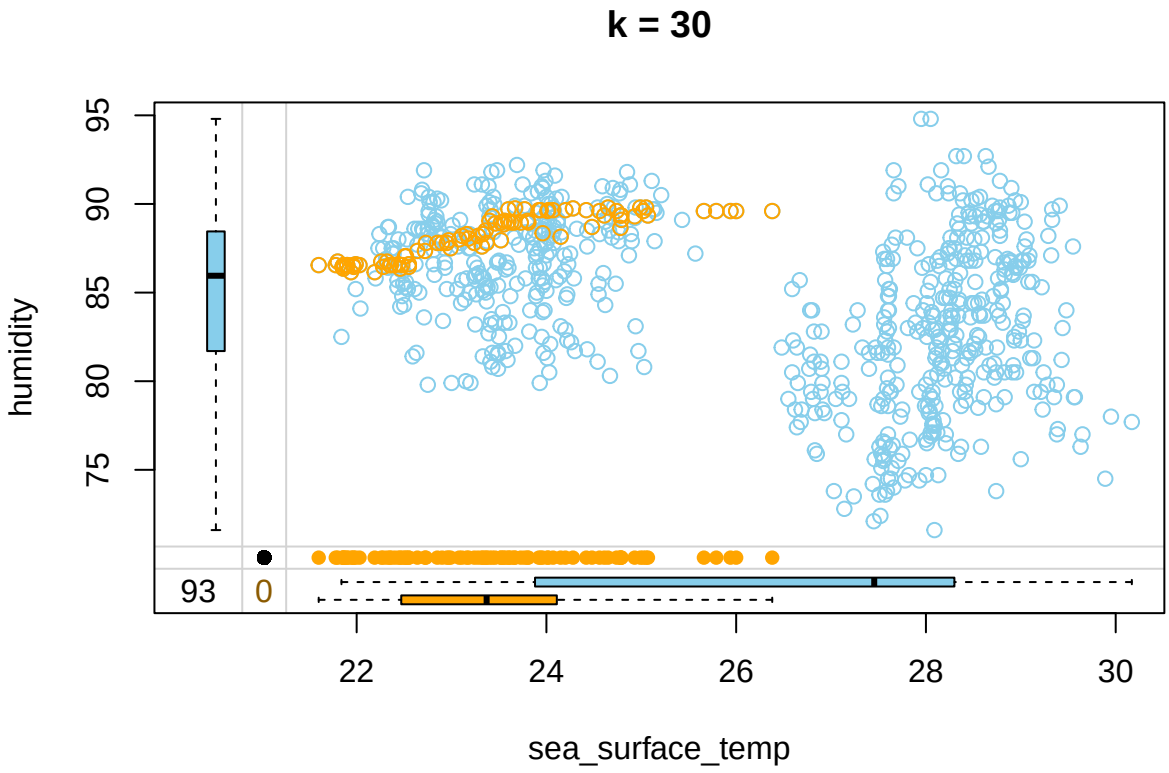
La imputación de k-Nearest-Neighbors (o kNN) imputa los valores faltantes en una observación en función de los valores que provienen de las k otras observaciones más similares a ella. El número de estas observaciones similares, llamadas vecinos, se consideran que es un parámetro que debe elegirse de antemano.

¿Cómo elegir k? Una forma es probar diferentes valores y ver cómo afectan las relaciones entre los datos imputados y observados.

Intentemos imputar `humidity` en los datos de `tao` utilizando tres valores diferentes de k y ver cómo se ajustan los valores imputados a la relación entre `humidity` y `sea_surface_temp`.

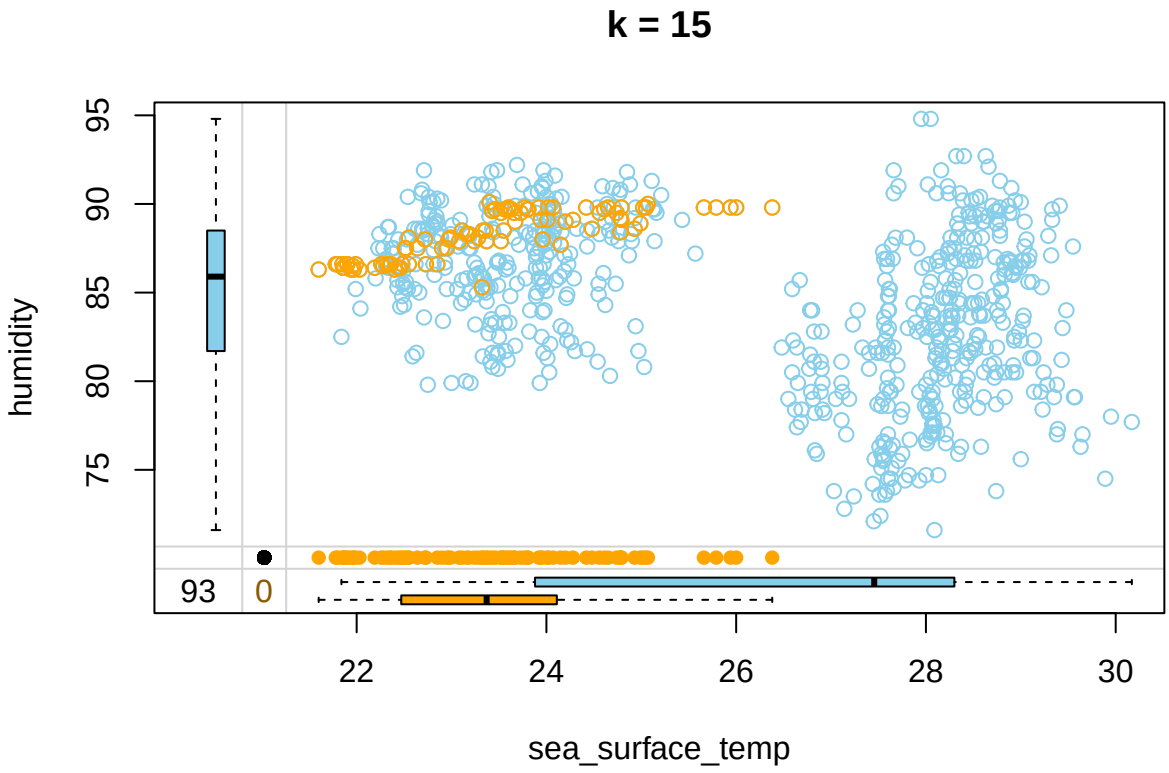
Imputamos `humidity` con la imputación de kNN usando 30 vecinos y visualizadolo mediante un `marginplot()` de `sea_surface_temp` vs `humidity`.

```
# Impute humidity using 30 neighbors
tao_imp <- kNN(tao, k = 30, variable = "humidity")

# Draw a margin plot of sea_surface_temp vs humidity
tao_imp %>%
    select(sea_surface_temp, humidity, humidity_imp) %>%
    marginplot(delimiter = "imp", main = "k = 30")
```



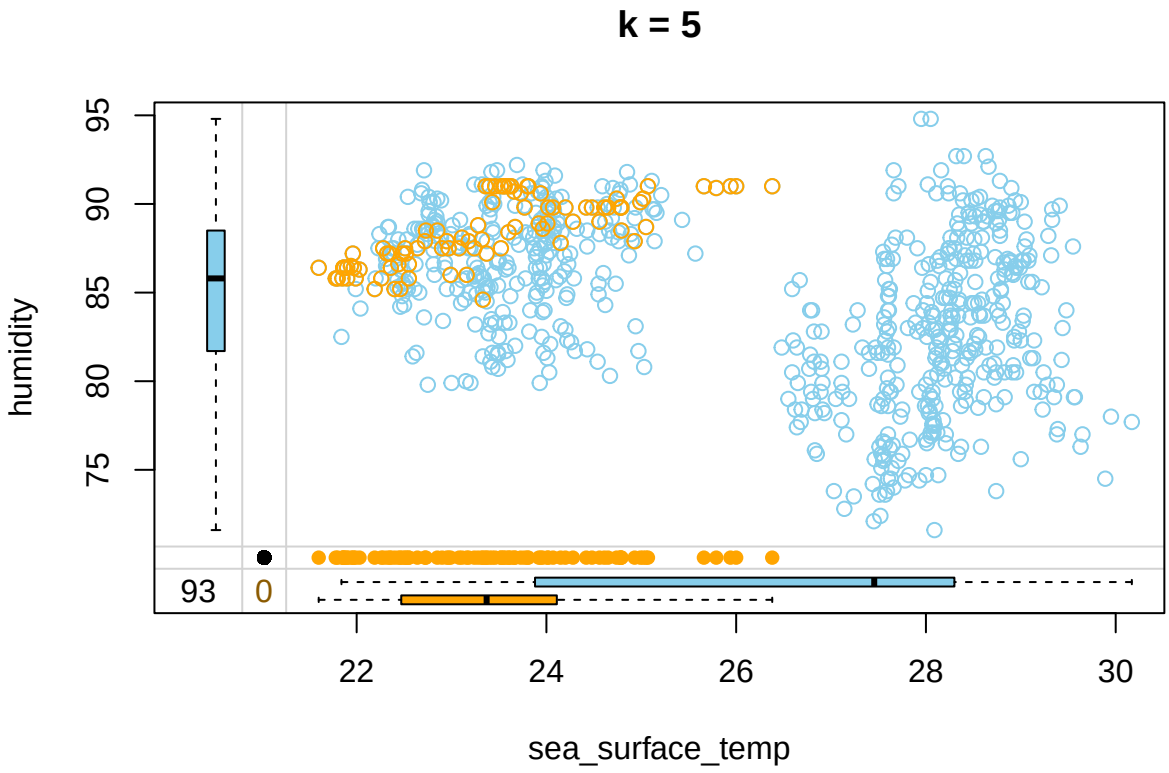
Ahora, imputamos `humidity` con imputación kNN usando 15 vecinos y vemos mediante el `marginplot` de `sea_surface_temp` vs `humidity`.

```
# Impute humidity using 15 neighbors
tao_imp <- kNN(tao, k = 15, variable = "humidity")

# Draw a margin plot of sea_surface_temp vs humidity
tao_imp %>%
    select(sea_surface_temp, humidity, humidity_imp) %>%
    marginplot(delimiter = "imp", main = "k = 15")
```



Finalmente, imputamos `humidity` con imputación kNN usando 5 vecinos y visualizando los resulados mediante `marginplot` de `sea_surface_temp` vs `humidity`.

```
# Impute humidity using 5 neighbors
tao_imp <- kNN(tao, k = 5, variable = "humidity")

# Draw a margin plot of sea_surface_temp vs humidity
tao_imp %>%
    select(sea_surface_temp, humidity, humidity_imp) %>%
    marginplot(delimiter = "imp", main = "k = 5")
```

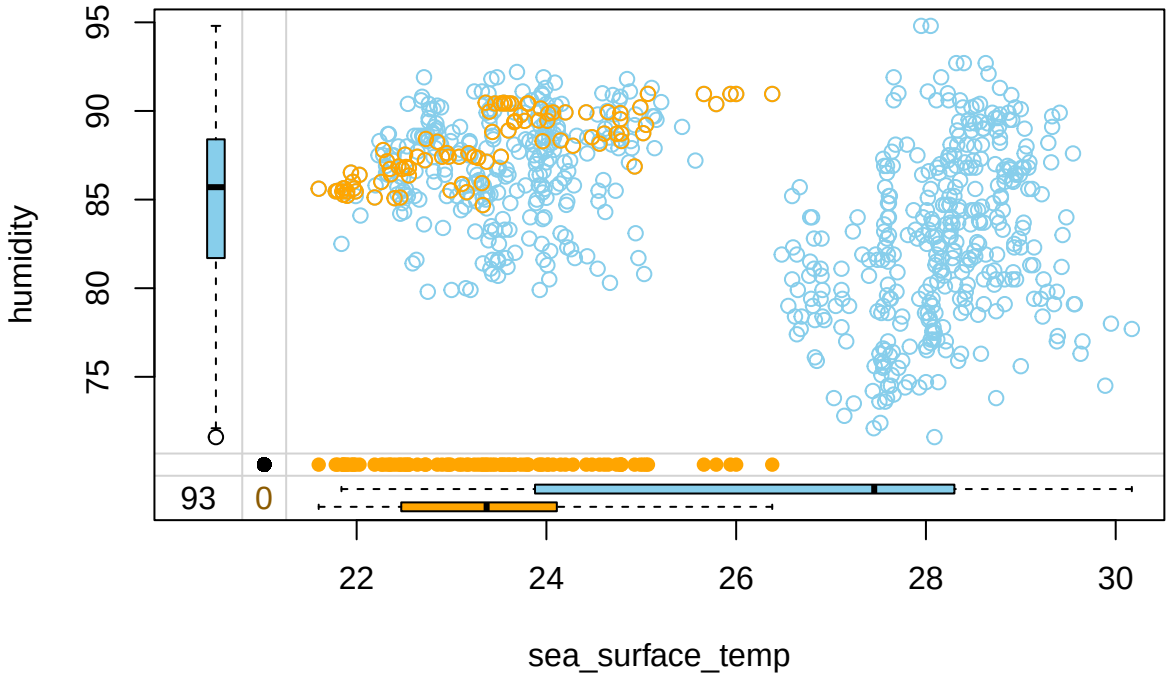


1.3.14 kNN trucos y consejos I: ponderando los donantes

Una variación de la imputación `kNN` que se aplica con frecuencia utiliza la llamada agregación ponderada por distancia. Lo que esto significa es que cuando agregamos los valores de los vecinos para obtener un reemplazo para un valor faltante, lo hacemos usando la media ponderada y las ponderaciones son las distancias invertidas de cada vecino. Como resultado, los vecinos más cercanos tienen más impacto en el valor imputado.

En este ejercicio, aplicamos la agregación ponderada por distancia mientras imputamos los datos de `tao`. Esto solo requerirá dar dos argumentos adicionales a la función `kNN()`.

```
# Load the VIM package
library(VIM)

# Impute humidity with kNN using distance-weighted mean
tao_imp <- kNN(tao,
               k = 5,
               variable = "humidity",
               numFun = weighted.mean,
               weightDist = TRUE)

tao_imp %>%
    select(sea_surface_temp, humidity, humidity_imp) %>%
    marginplot(delimiter = "imp")
```



Trucos y consejos de kNN II: ordenar variables

Mientras el algoritmo de k-Nearest Neighbors recorre las variables en los datos para imputarlos, calcula las distancias entre observaciones utilizando otras variables, algunas de las cuales ya han sido imputadas en los pasos anteriores. Esto significa que si las variables ubicadas al principio de los datos tienen muchas valores faltantes, entonces el cálculo de la distancia posterior se basa en muchos valores imputados. Esto introduce ruido en el cálculo de la distancia.

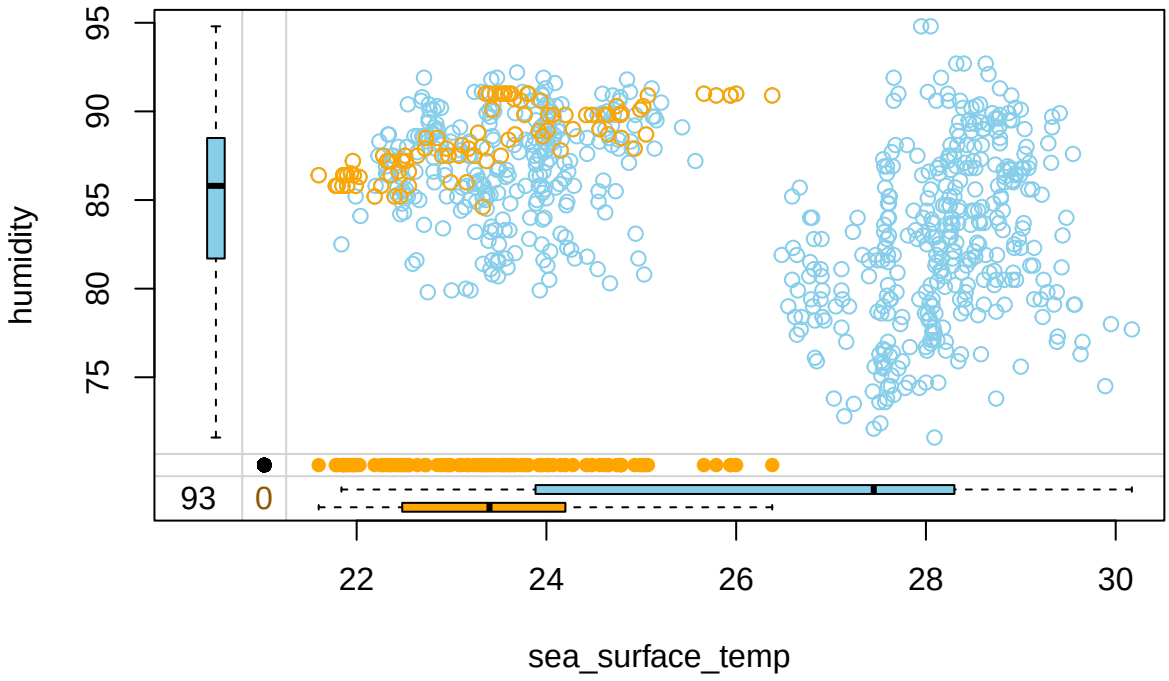
Por esta razón, una buena práctica es ordenar las variables por el número de valores faltantes antes de realizar la imputación kNN. De esta manera, cada cálculo de distancia se basa en tantos datos observados y tan pocos datos imputados como sea posible.

```
# Get tao variable names sorted by number of NAs
vars_by_NAs <- tao %>%
  is.na() %>%
  colSums() %>%
  sort(decreasing = FALSE) %>%
  names()

# Sort tao variables and feed it to kNN imputation
tao_imp <- tao %>%
  select(vars_by_NAs) %>%
  kNN(k= 5)
```

```
## Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
## i Please use ‘all_of()‘ or ‘any_of()‘ instead.
##   # Was:
##   data %>% select(vars_by_NAs)
##
```

```
##   # Now:
##   data %>% select(all_of(vars_by_NAs))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
tao_imp %>%
    select(sea_surface_temp, humidity, humidity_imp) %>%
    marginplot(delimiter = "imp")
```



El kNN que acabamos de programar debería ser más preciso y resistente a imputaciones defectuosas, así que recordemos ordenar las variables primero antes de realizar la imputación con kNN.

1.3.15 Imputación con regresión lineal

A veces, se puede utilizar el conocimiento del dominio, la investigación previa o simplemente el sentido común para describir las relaciones entre las variables en sus datos. En tales casos, la imputación basada en modelos es una gran solución, ya que permite imputar cada variable de acuerdo con un modelo estadístico que puede especificar uno mismo, teniendo en cuenta cualquier suposición que pueda tener sobre cómo las variables impactan entre sí.

Para variables continuas, una elección de modelo popular es la regresión lineal. Siempre puede incluir un cuadrado o un logaritmo de una variable en los predictores. En este caso, trabajaremos con el paquete `simputation` para ejecutar una sola imputación de regresión lineal en los datos `tao` y analizar los resultados.

```
# Lee la libreria simputation
library(simputation)
```

```
# Imputa air_temp y humidity con una regresion lineal
formula <- air_temp + humidity ~ year + latitude + sea_surface_temp
tao_imp <- impute_lm(tao, formula)
```

```
# Obtenemos el numero de valores missing por columna
tao_imp %>%
  is.na() %>%
  colSums()
```

```
##             year         latitude        longitude sea_surface_temp
##                0                0                0                3
##         air_temp         humidity            uwind            vwind
##                3                2                0                0
```

```
# Imprime las celdas de tao_imp en donde air_temp o humidity siguen missing
tao_imp %>%
  filter(is.na(air_temp) | is.na(humidity))
```

```
##   year latitude longitude sea_surface_temp air_temp humidity uwind vwind
## 1 1993        0       -95               NA       NA       NA  -5.6   3.1
## 2 1993        0       -95               NA       NA       NA  -6.3   0.5
## 3 1993       -2       -95               NA       NA     89.9  -3.4   2.4
```

La regresión lineal falla cuando al menos uno de los predictores está ausente. En este caso, fue `sea_surface_temp`. En el próximo ejercicio, lo solucionaremos inicializando los valores faltantes antes de ejecutar `impute_lm()`.

1.3.16 Inicialización de valores perdidos e iteración sobre variables

Como acabamos de ver, la ejecución de `impute_lm()` podría no llenar todos los valores perdidos. Para asegurarte de imputar todos ellos, debemos inicializar los valores perdidos con un método simple, como la imputación de hot-deck que de la sección anterior, que simplemente retroalimenta el último valor observado.

Además, una sola imputación generalmente no es suficiente. Se basa en los valores iniciales básicos y podría estar sesgada. Un enfoque adecuado es iterar sobre las variables, imputándolas una a la vez en las ubicaciones donde originalmente faltan.

En este ejercicio, primero inicializaremos los valores perdidos con la imputación de hot-deck y luego iteraremos cinco veces sobre `air_temp` y `humidity` de los datos `tao` para imputarlos con la regresión lineal.

```
# Inicializa los valores missing con hot-deck
tao_imp <- hotdeck(tao)

# Crea un indicador booleano desde donde air_temp y humidity son missing
missing_air_temp <- tao_imp$air_temp_imp
missing_humidity <- tao_imp$humidity_imp

for (i in 1:5) {
  # Define air_temp como NA en los lugares donde faltaban originalmente y re-imputa
  tao_imp$air_temp[missing_air_temp] <- NA
  tao_imp <- impute_lm(tao_imp, air_temp ~ year + latitude + sea_surface_temp + humidity)
  # Define humidity como NA en los lugares donde faltan originalmente y  re-imputa
  tao_imp$humidity[missing_humidity] <- NA
  tao_imp <- impute_lm(tao_imp, humidity ~ year + latitude + sea_surface_temp + air_temp)
}
```

Esa es una aproximación apropiada a la imputación basada en modelos que acabamos de codificar, pero,

¿cómo sabemos que 5 es el número adecuado de iteraciones para ejecutar?.

1.3.17 Detectando convergencia

¿Cuántas iteraciones son necesarias? Cuando los valores imputados no cambian con la nueva iteración, podemos detenernos.

Ahora extenderás nuestro código para calcular las diferencias entre las variables imputadas en las iteraciones subsiguientes. Para hacer esto, usaremos la función de cambio porcentual promedio absoluto (`mapc`), definida de la siguiente manera:

```
mapc <- function(a, b) {   mean(abs(b - a) / a, na.rm = TRUE) }
```

`mapc()` es una función que devuelve un solo número que te dice cuánto difiere b de a. La usaremos para verificar cuánto cambian las variables imputadas en las iteraciones siguientes. En base a esto, decidiremos cuántas iteraciones son necesarias.

Los indicadores booleanos `missing_air_temp` y `missing_humidity` son usados aquí, al igual que los datos de `tao_imp` inicializados con hot-deck.

```
mapc<- function(a, b) {
  mean(abs(b - a) / a, na.rm = TRUE)
}
```

```
diff_air_temp <- c()
diff_humidity <- c()

for (i in 1:5) {
  # Asigna el resultado de la iteración anterior (o inicialización) a prev_iter
  prev_iter <- tao_imp
  # Imputa air_temp y humidity en las ubicaciones que originalmente faltaban
  tao_imp$air_temp[missing_air_temp] <- NA
  tao_imp <- impute_lm(tao_imp, air_temp ~ year + latitude + sea_surface_temp + humidity)
  tao_imp$humidity[missing_humidity] <- NA
  tao_imp <- impute_lm(tao_imp, humidity ~ year + latitude + sea_surface_temp + air_temp)
  # Calcula MAPC para air_temp y humidity y los incluye a la iteración anterior de MAPC
  diff_air_temp <- c(diff_air_temp, mapc(prev_iter$air_temp, tao_imp$air_temp))
  diff_humidity <- c(diff_humidity, mapc(prev_iter$humidity, tao_imp$humidity))
}
```

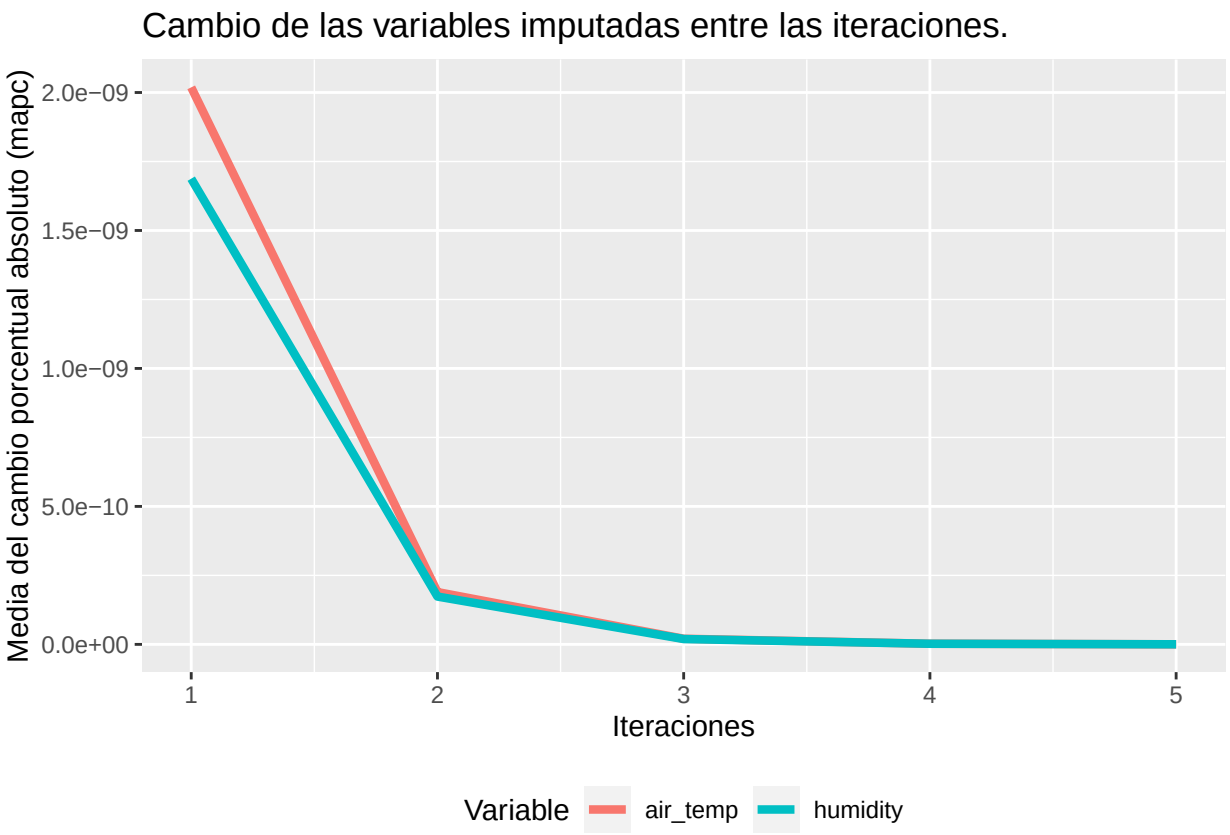
¿Cuál es un número suficiente de iteraciones para ejecutar, según las diferencias almacenadas en `diff_air_temp` y `diff_humidity`?

Para responder a esta pregunta, puedemos imprimir los dos vectores en la consola y analizar los números, o trazarlos usando la función proporcionada: simplemente ejecutamos `plot_diffs(diff_air_temp, diff_humidity)` en la consola.

```
plot_diffs <- function(a, b) {
  data.frame("mapc" = c(a, b),
             "Variable" = c(rep("air_temp", length(a)),
                            rep("humidity", length(b))),
             "Iteraciones" = c(1:length(a), 1:length(b))) %>%
    ggplot(aes(Iteraciones, mapc, color = Variable)) +
    geom_line(size = 1.5) +
    ylab("Media del cambio porcentual absoluto (mapc)") +
    ggtitle("Cambio de las variables imputadas entre las iteraciones.") +
    theme(legend.position = "bottom")
}
```

```
plot_diffs(diff_air_temp, diff_humidity)
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```



1.3.18 Imputación por regresión logística

Una opción popular para imputar variables binarias es la regresión logística. Desafortunadamente, no hay una función similar a `impute_lm()` que lo haga. Por eso crearemos una función para ello.

Llamemos a la función `impute_logreg()`. Su primer argumento será un data frame `df`, cuyos valores faltantes se han inicializado y solo contiene valores faltantes en la columna a imputar. El segundo argumento será una fórmula para el modelo de regresión logística.

La función hará lo siguiente:

1. Mantendrá las ubicaciones de los valores faltantes.
2. Construirá el modelo.
3. Realizará predicciones.
4. Reemplazará los valores faltantes con las predicciones.

No te preocupes por la línea que crea `imp_var` - esto es solo una forma de extraer el nombre de la columna a imputar de la fórmula.

```r
impute_logreg <- function(df, formula) {
  # Extrae el nombre de la variable respuesta
  imp_var <- as.character(formula[2])
  # Guarda los lugares donde la respuesta es missing
  missing_imp_var <- is.na(df[imp_var])
  # Ajusta una regresion del modo logistica
  logreg_model <- glm(formula, data = df, family = binomial)
  # Predice la respuesta y la convierte 0s y 1s
  preds <- predict(logreg_model, type = "response")
  preds <- ifelse(preds >= 0.5, 1, 0)
  # Imputa los valores missing con las predicciones
  df[missing_imp_var, imp_var] <- preds[missing_imp_var]
  return(df)
}
```

La función está completamente operativa y se puede enchufar en el bucle sobre las variables que viste en la sección previa, al igual que `impute_lm()` del paquete `simputation`. Pronto, combinaremos estos dos para imputar tanto variables continuas como binarias. Pero antes, mejoraremos `impute_logreg()` para que reproduzca mejor la variabilidad en los datos imputados.

1.3.19 Crear una distribución condicional

Simplemente llamar a `predict()` en un modelo siempre devolverá el mismo valor para los mismos valores de los predictores. Esto da como resultado una pequeña variabilidad en los datos imputados. Para aumentarla y que la imputación replique la variabilidad de los datos originales, que podemos extraer de la distribución condicional. Esto significa que en lugar de siempre predecir 1 cuando el modelo devuelve una probabilidad mayor que `0.5` , podemos extraer la predicción de una distribución binomial descrita por la probabilidad devuelta por el modelo.

Trabajaremos en el código del ejercicio anterior. La siguiente línea fue eliminada:

```r
preds <- ifelse(preds >= 0.5, 1, 0)
```

Nuestra tarea es llenar su lugar con la creación de una distribución binomial.

```r
impute_logreg <- function(df, formula) {
  # Extrae el nombre de la variable respuesta
  imp_var <- as.character(formula[2])
  # Guarda las posiciones donde la respuesta es missing
  missing_imp_var <- is.na(df[imp_var])
  # Ajusta una regresion del modo logistico
  logreg_model <- glm(formula, data = df, family = binomial)
  # Predice la respuesta
  preds <- predict(logreg_model, type = "response")
  # Toma una muestra de las predicciones de la distribución binomial
  # preds <- ifelse(preds >= 0.5, 1, 0)
  preds <- rbinom(length(preds), size = 1, prob = preds)
  # Imputa los valores missing con las predicciones
  df[missing_imp_var, imp_var] <- preds[missing_imp_var]
  return(df)
}
```

Crear la distribución condicional hará que la variabilidad de los datos imputados sea mucho parecida a la del conjunto de datos observados originales. Con esta potente función en nuestras manos, ahora podemos diseñar un flujo de imputación basado en modelos que se encargue tanto de variables continuas como binarias.

1.3.20 Imputación basada en modelos con varios tipos de variables

En este ejercicio, combinaremos lo que hemos aplicado hasta ahora sobre imputación basada en modelos para imputar diferentes tipos de variables en los datos de `tao`.

Nuestra tarea es iterar sobre las variables como lo hemos hecho previamente e imputar dos variables:

`is_hot`, una nueva variable binaria que se creó a partir de `air_temp`, que es 1 si `air_temp` está a 26 grados o más y 0 de lo contrario; `humidity`, una variable continua con la que ya estamos familiarizados.

Tendremos que utilizar la función de regresión lineal que aprendimos antes, así como la función para la regresión logística.

```
tao$is_hot<-ifelse(tao$air_temp>= 26, 1,0)
```

```
# Inicializamos los valores missing con hot-deck
tao_imp <- hotdeck(tao)

# Creamos el indicador booleano desde donde is_hot y humidity son missing
missing_is_hot <- tao_imp$is_hot_imp
missing_humidity <- tao_imp$humidity_imp

for (i in 1:3) {
  # Define is_hot como NA en los lugares donde fue originalmente missing y re-imputa
  tao_imp$is_hot[missing_is_hot] <- NA
  tao_imp <- impute_logreg(tao_imp, is_hot ~ sea_surface_temp)
  # Define humidity como NA en los lugares donde fue originalmente missing y re-imputa
  tao_imp$humidity[missing_humidity] <- NA
  tao_imp <- impute_lm(tao_imp, humidity ~ sea_surface_temp + air_temp)
}
```

1.3.21 Imputación con bosques aleatorios

Un enfoque de aprendizaje automático para la imputación puede ser más preciso y más fácil de implementar en comparación con modelos estadísticos tradicionales. Primero, no requiere que especifiques relaciones entre variables. Además, los modelos de aprendizaje automático como los *random forest* son capaces de descubrir relaciones altamente complejas y no lineales y explotarlas para predecir valores faltantes.

En este ejercicio, usaremos el paquete `missForest`, que construye un bosque aleatorio separado para predecir valores faltantes para cada variable, uno por uno. Llamaremos a la función de imputación sobre los datos de películas, `biopics`, con los que hemos trabajado anteriormente y luego extraeremos los datos completos, así como los errores de imputación estimados.

```
# leemos nuevamente los datos
biopics <-read.csv("curso_imputacion/biopics.csv")
# cargamos la libreria
library(missForest)

# trasformación de character a factor
biopics <- type.convert(biopics, as.is=FALSE)


# imputa los datos de biopics usando missForest
imp_res <- missForest(biopics)

# Extrae los datos imputados y revisa por valores missing
imp_data <- imp_res$ximp
print(sum(is.na(imp_data)))
```

```
## [1] 0
```

```
# Extrae e imprime los errores de imputacion
imp_err <- imp_res$OOBerror
print(imp_err)
```

```
##      NRMSE        PFC
## 0.02039904 0.04565603
```

En el ejercicio anterior hemos extraído los errores de imputación estimados a partir de la salida de `missForest`. Esto te dio dos números:

el error cuadrático medio raíz normalizado (NRMSE) para todas las variables continuas; la proporción de entradas falsamente clasificadas (PFC) para todas las variables categóricas.

Sin embargo, podría darse el caso de que el modelo de imputación funcione muy bien para una variable continua y muy mal para otra. Para diagnosticar tales casos, basta con decirle a `missForest` que produzca estimaciones de error por variable. Esto se hace estableciendo el argumento `variablewise` en `TRUE`.

```
# Imputa los datos de biopics con missForest calculando los errores por variable
imp_res <- missForest(biopics, variablewise = TRUE)

# Extrae e imprime los errores de imputacion
per_variable_errors <- imp_res$OOBerror
print(per_variable_errors)
```

```
##          PFC          MSE          MSE          MSE          PFC          PFC
##    0.0000000    0.0000000 1356.5367508    0.0000000    0.0000000    0.1719858
##          MSE          PFC
##    0.0000000    0.0000000
```

```
# Renombra las columnas para incluir el nombre de las variables
names(per_variable_errors) <- paste(names(biopics),
                                    names(per_variable_errors),
                                    sep = "_")

# Imprime los errores renombrados
print(per_variable_errors)
```

```
##   country_PFC      year_MSE  earnings_MSE   sub_num_MSE  sub_type_PFC
##     0.0000000     0.0000000  1356.5367508     0.0000000     0.0000000
##  sub_race_PFC non_white_MSE   sub_sex_PFC
##     0.1719858     0.0000000     0.0000000
```

Observa cómo produjimos una serie de medidas de error en lugar de las dos por defecto que habiamos visto antes. Ahora podemos evaluar la calidad de imputación para cada variable por separado. Esto es útil cuando necesitamos saber cómo se desempeña el modelo para una variable en particular que deseamos modelar o analizar más a fondo.

1.3.22 Trade-off velocidad-precisión

En este sentido existen dos paramétros que podemos ajustar para influir en el rendimiento de los bosques aleatorios (*random forest*):

. Número de árboles de decisión en cada bosque. . Número de variables utilizadas para la división dentro de los árboles de decisión.

Aumentar cada uno de ellos puede mejorar la precisión del modelo de imputación, pero también requerirá más tiempo para ejecutarse. En este ejercicio, exploraremos estas ideas ajustando `missForest()` a los datos

de `biopics` dos veces con diferentes configuraciones. Mientras seguimos estos pasos, pongamos atención a los errores que imprimiremos y al tiempo que tomará la ejecución del código.

```
# Determina el tiempo inicial del primer enfoque
 t <- proc.time()

# Define el numero de arboles a 5 y el numero de variables usadas para dividir en 2
imp_res <- missForest(biopics, mtry = 2, ntree = 5)
tiempo1<-proc.time() - t
# Imprime los resultados de los errores de la imputacion
print(imp_res$OOBerror)
```

```
##      NRMSE        PFC
## 0.02424906 0.08053360
```

```
# Determina el tiempo inicial del segundo enfoque
 t <- proc.time()
# Define el numero de arboles a 50 y el numero de variables usadas para dividir en 6
imp_res <- missForest(biopics, mtry = 6, ntree = 50)
tiempo2<-proc.time() - t
# Imprime los errores resultantes de la imputacion
print(imp_res$OOBerror)
```

```
##      NRMSE        PFC
## 0.02138087 0.04654255
```

Compara los errores y los tiempos de ejecución de los dos modelos de imputación. ¿Puedes ver una relación? Como dicen, "no hay nada gratuito". Para obtener una imputación más precisa, tuvimos que invertir más tiempo de computación.

```
tiempo1
```

```
##    user  system elapsed
##    0.09    0.00    0.10
```

```
tiempo2
```

```
##    user  system elapsed
##    2.89    0.05    2.94
```

1.3.23 La imputación y el modelado en una función

Siempre que realice cualquier análisis o modelado en datos imputados, debe tener en cuenta la incertidumbre de la imputación. Ejecutar un modelo en un conjunto de datos imputados, se ignora el hecho de que la imputación estima los valores faltantes con incertidumbre. Los errores estándar de dicho modelo tienden a ser demasiado pequeños. La solución a esto es la imputación múltiple y una forma de implementarla es mediante bootstrap.

Trabajaremos con los datos `biopics`. El objetivo es utilizar la imputación múltiple mediante bootstrap y la regresión lineal para ver si, en función de los datos disponibles, las películas biográficas con mujeres ganan menos que las de hombres.

Comencemos escribiendo una función que construya una muestra de bootstrap, la impute y ajuste un modelo de regresión lineal.

```
calc_gender_coef <- function(data, indices) {
  # Obtener una muestra bootstrap
  data_boot <- data[indices, ]
  # Imputa con imputacion kNN
```

```
  data_imp <- kNN(data_boot, k = 5)
  # Ajusta una regresion lineal
  linear_model <- lm(earnings ~ sub_sex + sub_type + year, data = data_imp)
  # Extrae y calcula coeficiente para gender
  gender_coefficient <- coef(linear_model)[2]
  return(gender_coefficient)
}
```

La función `calc_gender_coef()` toma los datos y los índices de bootstrap como entradas, y produce nuestra estadística de interés: el impacto del género en las ganancias de la regresión lineal. Ahora podemos usar esta función en el algoritmo de bootstrapping.

1.3.24 Ejecutando bootstrap

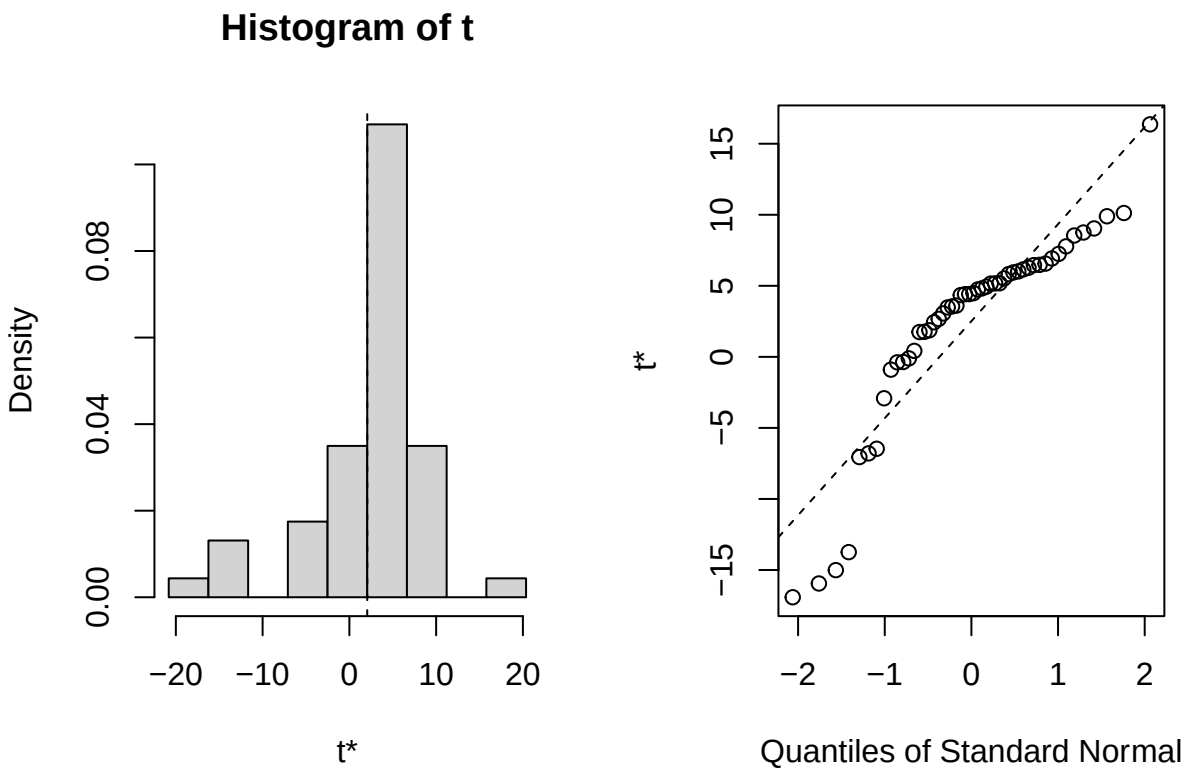
Esta función crea una muestra de bootstrap, la imputa y produce el coeficiente de regresión lineal que describe el impacto de que el tema de la película sea femenino en las ganancias de la película.

En este ejercicio, usarás el paquete `boot` para obtener una distribución de bootstrap de estos coeficientes. La propagación de esta distribución capturará la incertidumbre de la imputación. También verás cómo la distribución de bootstrap difiere de una imputación y regresión.

```
# Carga la libreria boot
library(boot)

# Ejecuta bootstrap sobre los datos biopics
boot_results <- boot(biopics, statistic = calc_gender_coef, R = 50)

# Imprime y grafica los resultados del bootstrapping
print(boot_results)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = biopics, statistic = calc_gender_coef, R = 50)
##
##
## Bootstrap Statistics :
##     original    bias    std. error
## t1* 2.060346 0.4512541    6.828914
```

```
plot(boot_results)
```



Si hubieramos ejecutado la imputación `kNN` y el análisis de regresión en los datos de `biopics` solo una vez, habríamos obtenido un coeficiente de `-1.45` para las películas sobre mujeres (llamado "original" en la salida de la consola), lo que sugiere que las películas sobre mujeres ganan menos. Sin embargo, al corregir la incertidumbre de la imputación, hemos obtenido una distribución que cubre tanto valores negativos como positivos.

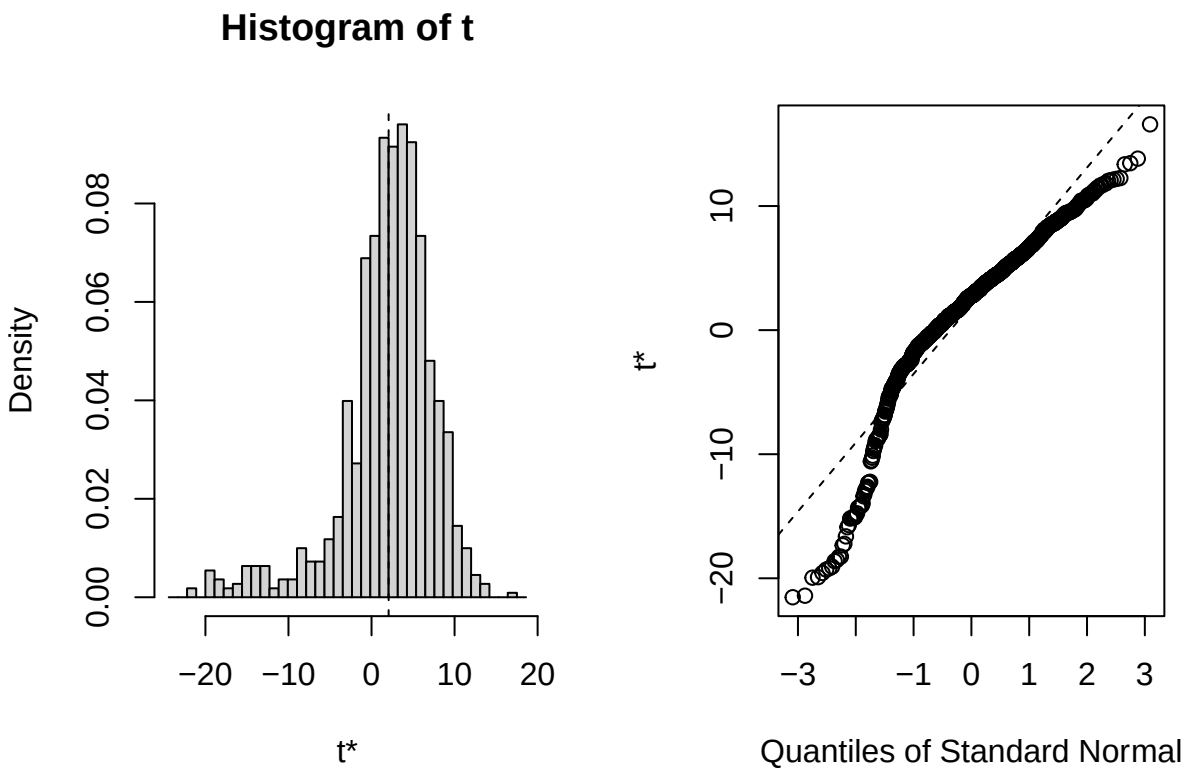
1.3.25 Bootstrapping para intervalos de confianza

Después de haber generado la distribución del coeficiente del efecto femenino en el último ejercicio, ahora podemos usarla para estimar un intervalo de confianza. Esto permitirá hacer la siguiente evaluación sobre los datos: "Dada la incertidumbre de la imputación, estamos 95% seguros de que el efecto femenino en las ganancias se encuentra entre `a` y `b`", donde `a` y `b` son los límites inferior y superior del intervalo.

En el último ejercicio, ejecutamos la técnica de bootstrapping con `R = 50` réplicas. Sin embargo, en la mayoría de las aplicaciones esto no es suficiente. En este ejercicio, puedes utilizar los `boot_results` que se prepararon utilizando 1000 réplicas. Primero, verás si la distribución de bootstrapping parece normal. Si es así, entonces podrás confiar en la distribución normal para calcular el intervalo de confianza.

```
# Ejecuta bootstrap sobre los datos biopics y mide el tiempo de ejecucion
boot_results <- boot(biopics, statistic = calc_gender_coef, R = 1000)
```

```
# Plot and print boot_results
plot(boot_results)
```




```
print(boot_results)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = biopics, statistic = calc_gender_coef, R = 1000)
##
##
## Bootstrap Statistics :
##     original      bias    std. error
## t1* 2.060346 -0.04651124    5.542859
```

```
# Calculate and print confidence interval
boot_ci <- boot.ci(boot_results, conf = 0.95, type = "norm")
print(boot_ci)
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot_results, conf = 0.95, type = "norm")
##
## Intervals :
## Level      Normal
## 95%   (-8.757, 12.971 )
```

Calculations and Intervals on Original Scale

A pesar de que la tendencia general parece ser una relación negativa, las réplicas de bootstrap muestran que algunas películas con protagonistas femeninas en realidad ganan más. Al tener en cuenta la incertidumbre de la imputación, no se puede estar al 100% seguro acerca de la dirección de esta relación, aunque un análisis único sugiera lo contrario.

1.3.26 El flujo de MICE: mice - with - pool

El flujo de `MICE` (imputación múltiple por ecuaciones encadenadas) nos permite estimar la incertidumbre de la imputación mediante la imputación de un conjunto de datos varias veces mediante la imputación basada en modelos, mientras se extrae de las distribuciones condicionales. De esta manera, cada conjunto de datos imputados es ligeramente diferente. Luego, se realiza un análisis en cada uno de ellos y se combinan los resultados, obteniendo las cantidades de interés junto con sus intervalos de confianza que reflejan la incertidumbre de la imputación.

En este ejercicio, practicaremos el flujo típico de la imputación con `MICE`: `mice() - with() - pool()`. Realizaremos un análisis de regresión en los datos de `biopics` para ver qué tipo de ocupación de sujeto, `sub_type`, está asociada con mayores ingresos en las películas.

```
# Carga el paquete mice
library(mice)
```

```
##
## Attaching package: 'mice'
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

```
# Imputa biopics con mice usando 5 imputaciones
biopics_multiimp <- mice(biopics, m = 5, seed = 3108)
```

```
##
##  iter imp variable
##   1   1  earnings  sub_race
##   1   2  earnings  sub_race
##   1   3  earnings  sub_race
##   1   4  earnings  sub_race
##   1   5  earnings  sub_race
##   2   1  earnings  sub_race
##   2   2  earnings  sub_race
##   2   3  earnings  sub_race
##   2   4  earnings  sub_race
##   2   5  earnings  sub_race
##   3   1  earnings  sub_race
##   3   2  earnings  sub_race
##   3   3  earnings  sub_race
##   3   4  earnings  sub_race
##   3   5  earnings  sub_race
##   4   1  earnings  sub_race
##   4   2  earnings  sub_race
##   4   3  earnings  sub_race
##   4   4  earnings  sub_race
```

```
##   4   5  earnings  sub_race
##   5   1  earnings  sub_race
##   5   2  earnings  sub_race
##   5   3  earnings  sub_race
##   5   4  earnings  sub_race
##   5   5  earnings  sub_race
```

```
## Warning: Number of logged events: 50
```

```
# Ajusta una regresion lineal para cada set de datos imputados
lm_multiimp <- with(biopics_multiimp, lm(earnings ~ year + sub_type))

# Combina las estimaciones por las reglas de Rubin (pool)
lm_pooled <- pool(lm_multiimp)
summary(lm_pooled, conf.int = TRUE, conf.level = 0.95)
```

```
##                              term     estimate  std.error   statistic
## 1                     (Intercept) -287.8866750 490.062824 -0.58744851
## 2                            year    0.1612176   0.239277  0.67376974
## 3  sub_typeAcademic (Philosopher)  -32.8423364  39.593926 -0.82947915
## 4                sub_typeActivist  -17.4281787  16.052579 -1.08569339
## 5                   sub_typeActor  -30.7740287  19.378762 -1.58802862
## 6                 sub_typeActress  -27.3033456  21.249448 -1.28489670
## 7      sub_typeActress / activist   16.0962907  39.192849  0.41069458
## 8                  sub_typeArtist  -27.4779956  18.148136 -1.51409465
## 9                 sub_typeAthlete   -4.2336944  12.503822 -0.33859202
## 10     sub_typeAthlete / military   79.1943734  38.055447  2.08102595
## 11                 sub_typeAuthor  -23.6540827  18.471621 -1.28056347
## 12          sub_typeAuthor (poet)  -23.4506193  19.985477 -1.17338304
## 13               sub_typeComedian  -22.9330598  21.576033 -1.06289508
## 14               sub_typeCriminal   -2.6478096  16.754147 -0.15803906
## 15             sub_typeGovernment   -5.0221576  21.879188 -0.22954040
## 16             sub_typeHistorical   -9.3734433  19.183957 -0.48860845
## 17             sub_typeJournalist  -26.5071032  29.005345 -0.91386960
## 18                  sub_typeMedia  -11.5302020  26.461566 -0.43573393
## 19               sub_typeMedicine    4.7788761  15.006237  0.31845932
## 20               sub_typeMilitary   10.9417685  19.325322  0.56618816
## 21    sub_typeMilitary / activist   37.2248141  39.752579  0.93641257
## 22               sub_typeMusician  -19.7931797  17.320867 -1.14273611
## 23                  sub_typeOther  -15.5895605  16.044509 -0.97164460
## 24             sub_typePolitician  -13.6751859  39.752579 -0.34400752
## 25                 sub_typeSinger    0.6677979  17.844592  0.03742298
## 26                sub_typeTeacher   51.1575084  39.268925  1.30274786
## 27           sub_typeWorld leader    5.9976436  16.882980  0.35524793
##            df    p.value        2.5 %       97.5 %
## 1    3.983993 0.58858265 -1650.677928 1074.9045785
## 2    4.030421 0.53712441    -0.501149    0.8235843
## 3  139.785256 0.40824758  -111.122704   45.4380311
## 4   10.576004 0.30174143   -52.933253   18.0768961
## 5   10.847500 0.14097812   -73.499669   11.9516115
## 6    7.876739 0.23532207   -76.438456   21.8317642
## 7  169.869958 0.68181408   -61.271473   93.4640548
## 8    8.280382 0.16719720   -69.082290   14.1262992
## 9   15.246283 0.73953513   -30.847513   22.3801244
## 10 336.810947 0.03818668     4.338081  154.0506662
```

```
## 11   7.047583 0.24087818   -67.272814   19.9646489
## 12  10.612273 0.26630514   -67.635233   20.7339945
## 13  16.686775 0.30297041   -68.519689   22.6535693
## 14   7.275709 0.87872330   -41.962758   36.6671385
## 15  81.573759 0.81902349   -48.550237   38.5059216
## 16   6.172715 0.64199292   -55.998343   37.2514567
## 17 113.284343 0.36272632   -83.970362   30.9561559
## 18   6.128774 0.67795894   -75.950965   52.8905609
## 19 247.938063 0.75040464   -24.777080   34.3348320
## 20   6.645835 0.58986247   -35.253693   57.1372305
## 21 130.043524 0.35079655   -41.420661  115.8702894
## 22   6.996299 0.29074124   -60.754913   21.1685537
## 23   7.168854 0.36286255   -53.348394   22.1692725
## 24 130.043524 0.73139625   -92.320661   64.9702894
## 25  10.377953 0.97085786   -38.897026   40.2326216
## 26 163.415623 0.19449369   -26.382404  128.6974204
## 27  14.000056 0.72769899   -30.212733   42.2080205
```

En este caso, hemos seguido el flujo "mice-with-pool" para imputar, modelar y agrupar los resultados. Ahora, echemos un vistazo a la salida en la consola: algunos `sub_types` tienen un impacto positivo en las ganancias. Sin embargo, al tener en cuenta la incertidumbre de la imputación con una confianza del 95%, nunca estamos seguros de estos efectos, ya que los límites inferiores son negativos. Con una excepción: para `sub_typeAthlete / military`, tanto los límites inferiores como los superiores son positivos. Lo que podemos decir con seguridad es que las películas sobre atletas militares son populares.

1.3.26.1 Selección de modelos por defecto MICE crea un modelo de imputación separado para cada variable en los datos. El tipo de modelo depende del tipo de variable en cuestión. Una forma popular de especificar los tipos de modelos que queremos usar es establecer un modelo predeterminado para cada uno de los cuatro tipos de variables.

Podemos hacer esto usando el argumento defaultMethod en la función `mice()`, que debe ser un vector de longitud 4 que contenga los métodos de imputación predeterminados para:

Variables continuas, Variables binarias, Variables categóricas (factores no ordenados), Variables factoriales (factores ordenados).

En este caso, aprovecharemos la documentación de `mice` para ver la lista de métodos disponibles y seleccionar los deseados para que el algoritmo los use.

```
# Imputa biopics usando los metodos especificados en la instruccion
biopics_multiimp <- mice(biopics, m = 20,
                         defaultMethod = c("cart", "lda", "pmm", "polr"))
```

```
##
##  iter imp variable
##   1   1  earnings  sub_race
##   1   2  earnings  sub_race
##   1   3  earnings  sub_race
##   1   4  earnings  sub_race
##   1   5  earnings  sub_race
##   1   6  earnings  sub_race
##   1   7  earnings  sub_race
##   1   8  earnings  sub_race
##   1   9  earnings  sub_race
##   1   10  earnings  sub_race
##   1   11  earnings  sub_race
##   1   12  earnings  sub_race
```

```
##   1   13  earnings  sub_race
##   1   14  earnings  sub_race
##   1   15  earnings  sub_race
##   1   16  earnings  sub_race
##   1   17  earnings  sub_race
##   1   18  earnings  sub_race
##   1   19  earnings  sub_race
##   1   20  earnings  sub_race
##   2   1  earnings  sub_race
##   2   2  earnings  sub_race
##   2   3  earnings  sub_race
##   2   4  earnings  sub_race
##   2   5  earnings  sub_race
##   2   6  earnings  sub_race
##   2   7  earnings  sub_race
##   2   8  earnings  sub_race
##   2   9  earnings  sub_race
##   2   10  earnings  sub_race
##   2   11  earnings  sub_race
##   2   12  earnings  sub_race
##   2   13  earnings  sub_race
##   2   14  earnings  sub_race
##   2   15  earnings  sub_race
##   2   16  earnings  sub_race
##   2   17  earnings  sub_race
##   2   18  earnings  sub_race
##   2   19  earnings  sub_race
##   2   20  earnings  sub_race
##   3   1  earnings  sub_race
##   3   2  earnings  sub_race
##   3   3  earnings  sub_race
##   3   4  earnings  sub_race
##   3   5  earnings  sub_race
##   3   6  earnings  sub_race
##   3   7  earnings  sub_race
##   3   8  earnings  sub_race
##   3   9  earnings  sub_race
##   3   10  earnings  sub_race
##   3   11  earnings  sub_race
##   3   12  earnings  sub_race
##   3   13  earnings  sub_race
##   3   14  earnings  sub_race
##   3   15  earnings  sub_race
##   3   16  earnings  sub_race
##   3   17  earnings  sub_race
##   3   18  earnings  sub_race
##   3   19  earnings  sub_race
##   3   20  earnings  sub_race
##   4   1  earnings  sub_race
##   4   2  earnings  sub_race
##   4   3  earnings  sub_race
##   4   4  earnings  sub_race
##   4   5  earnings  sub_race
##   4   6  earnings  sub_race
```

```
##   4   7  earnings  sub_race
##   4   8  earnings  sub_race
##   4   9  earnings  sub_race
##   4   10  earnings  sub_race
##   4   11  earnings  sub_race
##   4   12  earnings  sub_race
##   4   13  earnings  sub_race
##   4   14  earnings  sub_race
##   4   15  earnings  sub_race
##   4   16  earnings  sub_race
##   4   17  earnings  sub_race
##   4   18  earnings  sub_race
##   4   19  earnings  sub_race
##   4   20  earnings  sub_race
##   5   1  earnings  sub_race
##   5   2  earnings  sub_race
##   5   3  earnings  sub_race
##   5   4  earnings  sub_race
##   5   5  earnings  sub_race
##   5   6  earnings  sub_race
##   5   7  earnings  sub_race
##   5   8  earnings  sub_race
##   5   9  earnings  sub_race
##   5   10  earnings  sub_race
##   5   11  earnings  sub_race
##   5   12  earnings  sub_race
##   5   13  earnings  sub_race
##   5   14  earnings  sub_race
##   5   15  earnings  sub_race
##   5   16  earnings  sub_race
##   5   17  earnings  sub_race
##   5   18  earnings  sub_race
##   5   19  earnings  sub_race
##   5   20  earnings  sub_race
```

```
## Warning: Number of logged events: 200
```

```
# Imprime biopics_multiimp
print(biopics_multiimp)
```

```
## Class: mids
## Number of multiple imputations:  20
## Imputation methods:
##   country      year  earnings   sub_num  sub_type  sub_race non_white   sub_sex
##        ""        ""    "cart"        ""        ""     "pmm"        ""        ""
## PredictorMatrix:
##          country year earnings sub_num sub_type sub_race non_white sub_sex
## country        0    1        1       1        1        1         1       1
## year           1    0        1       1        1        1         1       1
## earnings       1    1        0       1        1        1         1       1
## sub_num        1    1        1       0        1        1         1       1
## sub_type       1    1        1       1        0        1         1       1
## sub_race       1    1        1       1        1        0         1       1
## Number of logged events:  200
##   it im      dep meth                          out
```

```
## 1  1  1 earnings cart sub_typeAcademic (Philosopher)
## 2  1  1 sub_race  pmm    sub_typeMilitary / activist
## 3  1  2 earnings cart sub_typeAcademic (Philosopher)
## 4  1  2 sub_race  pmm    sub_typeMilitary / activist
## 5  1  3 earnings cart sub_typeAcademic (Philosopher)
## 6  1  3 sub_race  pmm    sub_typeMilitary / activist
```

La capacidad de especificar modelos de imputación puede resultar útil cuando se observa que algunos métodos específicos no funcionan bien. Otro factor que influye en cómo funcionan los métodos de imputación es el conjunto de predictores que utilizan. En el siguiente ejercicio, veremos cómo establecer estos predictores.

1.3.26.2 Usando una matriz predictora Se trata de tomar decisiones importantes cuando se utiliza la imputación basada en modelos, como por ejemplo, qué variables deben incluirse como predictores y en qué modelos. En `mice()`, esto se rige por la matriz de predictores y, por defecto, todas las variables se utilizan para imputar todas las demás.

En caso de tener muchas variables en los datos o poco tiempo para realizar una selección adecuada del modelo, puede utilizar la funcionalidad de `mice` para crear una matriz de predictores basada en las correlaciones entre las variables. Esta matriz se puede incorporar a `mice()`. En este ejercicio, practicaremos exactamente esto: primero construiremos una matriz de predictores de modo que cada variable se imputa utilizando las variables más correlacionadas con ella; luego, usará una matriz de predictores con la función de imputación.

```
# Crea una matriz predictora con correlacion minima de 0.1
pred_mat <- quickpred(biopics, mincor = 0.1)

# Imputa biopics con mice
biopics_multiimp <- mice(biopics,
                         m = 10,
                         predictorMatrix = pred_mat,
                         seed = 3108)
```

```
##
##  iter imp variable
##   1   1  earnings  sub_race
##   1   2  earnings  sub_race
##   1   3  earnings  sub_race
##   1   4  earnings  sub_race
##   1   5  earnings  sub_race
##   1   6  earnings  sub_race
##   1   7  earnings  sub_race
##   1   8  earnings  sub_race
##   1   9  earnings  sub_race
##   1   10  earnings  sub_race
##   2   1  earnings  sub_race
##   2   2  earnings  sub_race
##   2   3  earnings  sub_race
##   2   4  earnings  sub_race
##   2   5  earnings  sub_race
##   2   6  earnings  sub_race
##   2   7  earnings  sub_race
##   2   8  earnings  sub_race
##   2   9  earnings  sub_race
##   2   10  earnings  sub_race
##   3   1  earnings  sub_race
##   3   2  earnings  sub_race
##   3   3  earnings  sub_race
```

```
##   3   4  earnings  sub_race
##   3   5  earnings  sub_race
##   3   6  earnings  sub_race
##   3   7  earnings  sub_race
##   3   8  earnings  sub_race
##   3   9  earnings  sub_race
##   3   10  earnings  sub_race
##   4   1  earnings  sub_race
##   4   2  earnings  sub_race
##   4   3  earnings  sub_race
##   4   4  earnings  sub_race
##   4   5  earnings  sub_race
##   4   6  earnings  sub_race
##   4   7  earnings  sub_race
##   4   8  earnings  sub_race
##   4   9  earnings  sub_race
##   4   10  earnings  sub_race
##   5   1  earnings  sub_race
##   5   2  earnings  sub_race
##   5   3  earnings  sub_race
##   5   4  earnings  sub_race
##   5   5  earnings  sub_race
##   5   6  earnings  sub_race
##   5   7  earnings  sub_race
##   5   8  earnings  sub_race
##   5   9  earnings  sub_race
##   5   10  earnings  sub_race
```

```
## Warning: Number of logged events: 50
```

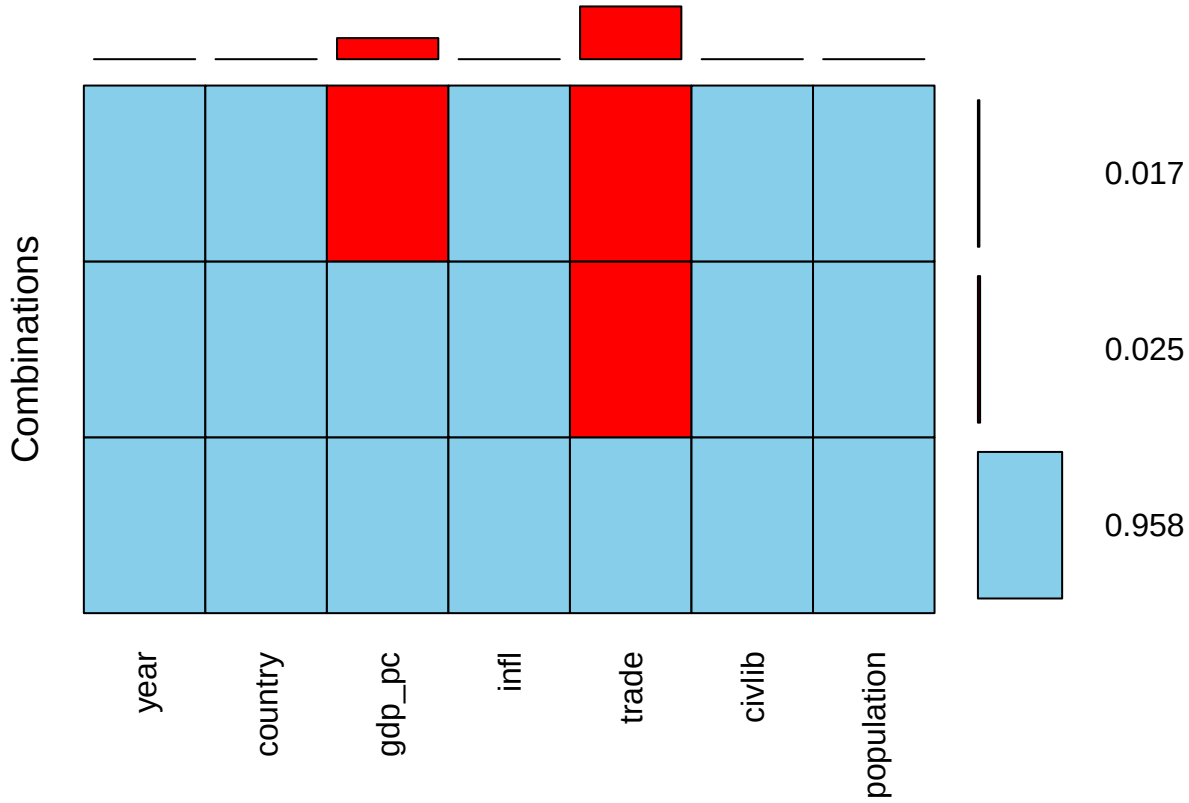
```
# Imprime biopics_multiimp
# print(biopics_multiimp)
```

1.3.27 Analizando los patrones de datos faltantes

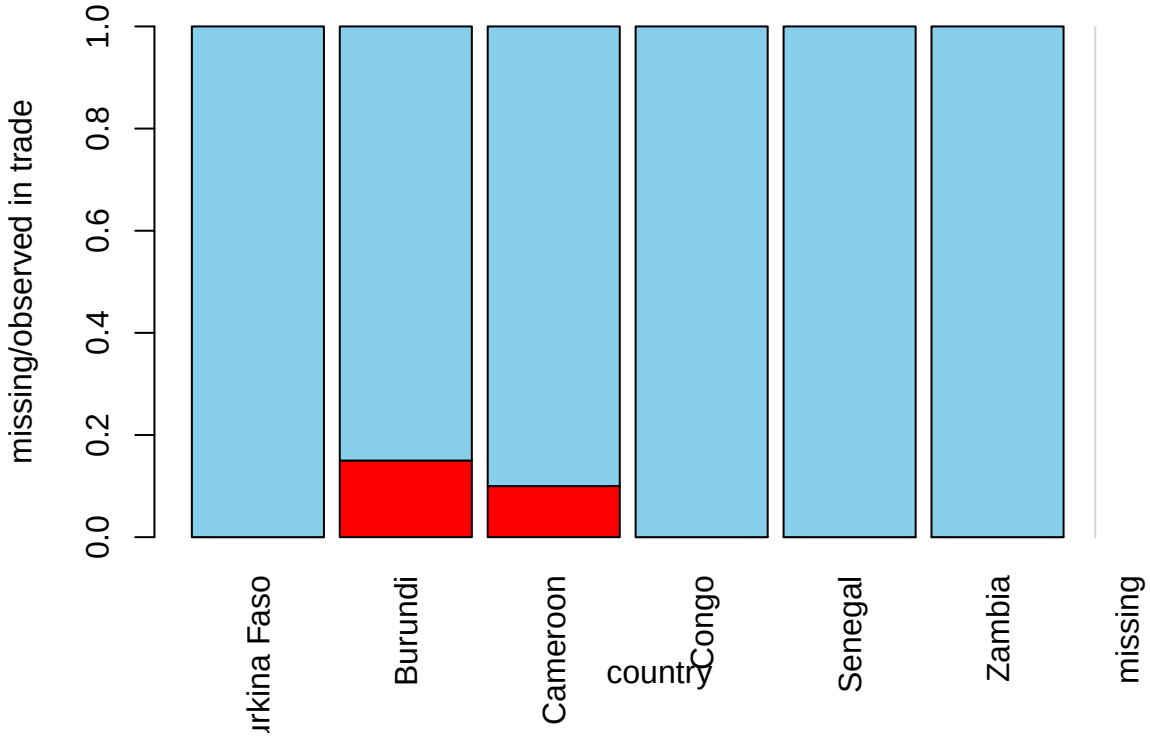
El primer paso para trabajar con datos incompletos es obtener información sobre los patrones de ausencia de datos, y una buena manera de hacerlo es mediante visualizaciones. Comenzarás tu análisis de los datos de `África` empleando el paquete `VIM` para crear dos visualizaciones: el gráfico de agregación y el gráfico de Mosaico. Te dirán cuántos datos faltan, en qué variables y configuraciones, y si podemos decir algo sobre el mecanismo de ausencia de datos. ¡Comencemos con algunas gráficas!

```
africa <- read.csv("handing/africa.csv", sep = ";")
```

```
# carga el paquete VIM
library(VIM)



# Crea un grafico de agregación combinada del set de datos africa
africa %>%
  aggr(combined = TRUE, numbers = TRUE)
```




```
# Crea un grafico spine plot de pais vs trade
africa %>%
  select(country, trade) %>%
  spineMiss()
```



Observamos que no hay tantos valores faltantes. Además, observe en el gráfico de Mosaico para los datos de `africa` parecen ser MAR - al menos con respecto al PIB y al país, lo que significa que se pueden imputar.

1.3.28 Imputando e inspeccionando resultados

Hemos descubierto que hay algunos datos faltantes en el PIB, `gdp_pc`, y en `trade` como porcentaje del PIB. Además, se sospecha que los datos son MAR, por lo que es posible que sean imputados. En este caso, haremos uso de la imputación múltiple del paquete `mice` para imputar los datos de `africa`. Luego, crearemos un gráfico para `gdp_pc` vs `trade` para ver si los datos imputados no rompen la relación entre estas variables.

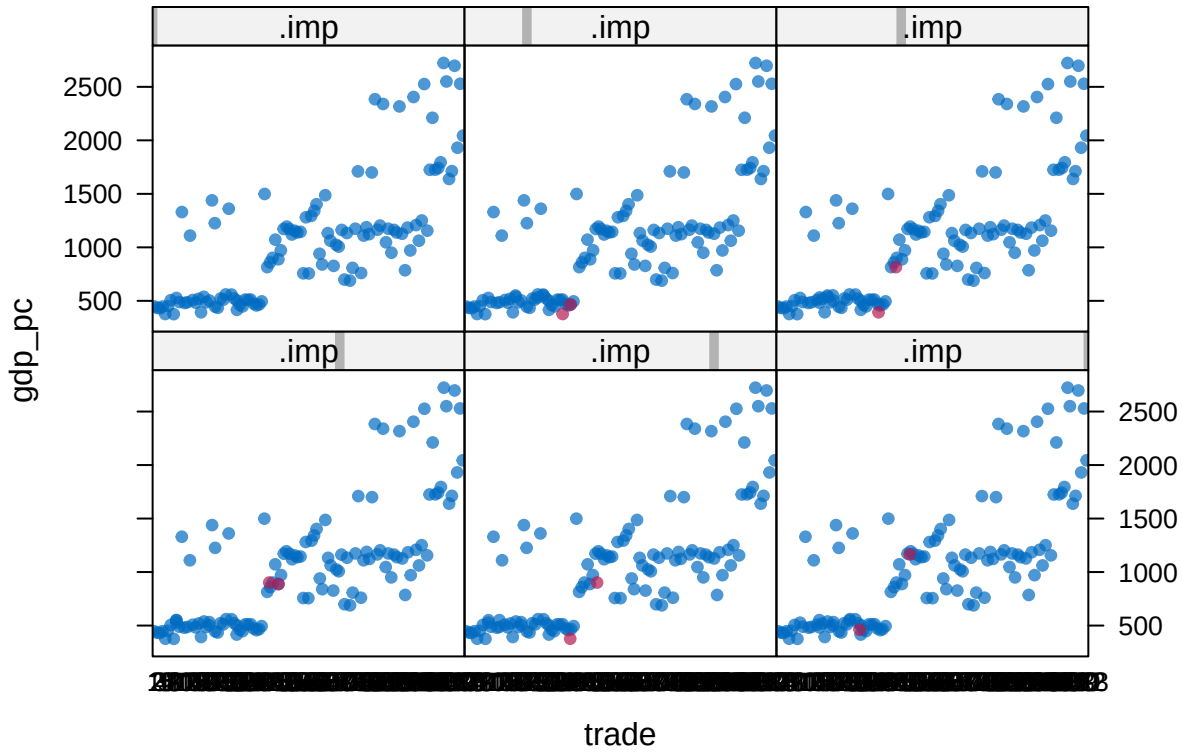
```
# Carga mice
library(mice)

# Imputa africa con mice
africa_multiimp <- mice(africa, m = 5, defaultMethod = "cart", seed = 3108)
```

```
##
##  iter imp variable
##   1   1  gdp_pc  trade
##   1   2  gdp_pc  trade
##   1   3  gdp_pc  trade
##   1   4  gdp_pc  trade
##   1   5  gdp_pc  trade
##   2   1  gdp_pc  trade
##   2   2  gdp_pc  trade
##   2   3  gdp_pc  trade
##   2   4  gdp_pc  trade
```

```
##    2   5  gdp_pc  trade
##    3   1  gdp_pc  trade
##    3   2  gdp_pc  trade
##    3   3  gdp_pc  trade
##    3   4  gdp_pc  trade
##    3   5  gdp_pc  trade
##    4   1  gdp_pc  trade
##    4   2  gdp_pc  trade
##    4   3  gdp_pc  trade
##    4   4  gdp_pc  trade
##    4   5  gdp_pc  trade
##    5   1  gdp_pc  trade
##    5   2  gdp_pc  trade
##    5   3  gdp_pc  trade
##    5   4  gdp_pc  trade
##    5   5  gdp_pc  trade
```

```
## Warning: Number of logged events: 1
```

```
# Crea un stripplot of gdp_pc versus trade
stripplot(africa_multiimp, gdp_pc ~ trade | .imp, pch = 20, cex = 1)
```



Se observa que la imputación funciona bien: hay pequeños grupos en los gráficos de dispersión, que probablemente corresponden a diferentes países. Cada punto de datos imputado encaja en uno de los grupos, en lugar de ser un valor atípico en algún lugar entre los grupos. Después de haber realizado la imputación, podemos proceder con el modelado.

1.3.28.1 Inferencia con datos imputados En este último caso, hemos utilizado `mice` para imputar los datos de `africa`. En este, implementaremos los otros dos pasos del flujo de "mice - with - pool" que hemos usado anteriormente. El modelo de interés es una regresión lineal que explica el PIB, `gdp_pc`, con otras variables. Nos interesa particularmente el coeficiente de libertades civiles, `civlib`. ¿Está asociado tener valores más altos en `civlib` en función a un mayor crecimiento económico una vez que incorporamos la incertidumbre de la imputación?

```
# Ajusta im regresion lineal a cada data set  imputado
lm_multiimp <- with(africa_multiimp, lm(gdp_pc ~ country + year + trade + infl + civlib))

# Combina las estimaciones por las reglas de Rubin (pool)
lm_pooled <- pool(lm_multiimp)

# Summarize pooled results
summary(lm_pooled, conf.int = TRUE, conf.level = 0.9)
```

```
##              term      estimate   std.error  statistic        df      p.value
## 1     (Intercept) -31703.576601 6031.455164 -5.2563728  92.53952 9.387968e-07
## 2   countryBurundi     63.739453   65.610134  0.9714879 103.18378 3.335772e-01
## 3  countryCameroon    622.343351   66.226798  9.3971530  56.28736 3.941785e-13
## 4     countryCongo   1303.940067  119.821885 10.8823197 101.65641 9.508463e-19
## 5   countrySenegal    516.442634   82.535746  6.2571995 106.90957 8.275032e-09
## 6    countryZambia    396.326840   87.729106  4.5176209 106.82235 1.620019e-05
## 7             year     16.156955    3.036095  5.3216230  92.06641 7.199791e-07
## 8            trade      5.468655    1.663523  3.2873939 105.04858 1.375742e-03
## 9             infl     -4.389418    1.031399 -4.2557883 107.91316 4.455825e-05
## 10          civlib    -84.852311  147.925470 -0.5736153  66.50333 5.681630e-01
##               5 %          95 %
## 1  -41724.760108 -21682.393094
## 2     -45.157324    172.636231
## 3     511.587065    733.099637
## 4    1105.037965   1502.842168
## 5     379.496718    653.388550
## 6     250.762899    541.890781
## 7      11.112260     21.201650
## 8       2.708058      8.229252
## 9      -6.100609     -2.678226
## 10   -331.605424    161.900803
```

Basándose en el resumen de los resultados de la regresión agrupada que acabamos de imprimir. Podemos decir que, dado que los límites inferior y superior tienen signos diferentes, no podemos estar seguros de la dirección del efecto.