11.- Causal Anaysis_CU_53_02_spi_v_01

June 13, 2023

#

CU53_impacto de las políticas de inversión en sanidad, infraestructuras y promoción turística en el SPI

Citizenlab Data Science Methodology > II - Data Processing Domain *** > # 11.- ECA - Exploratory Causal Analysis

Exploratory causal analysis (ECA) is the process of discovering the root causes of problems in order to identify appropriate solutions.

0.1 Tasks

Define the key challenge or setback

Determine the causes and effects of the key challenge

Use a diagram or graph to organize information

Formulate a response to the primary causes of your challenge

Review your process and address new causes and effects

0.2 File

- Input File: CU_53_09.2_02_spi
- Output File: No aplica

0.2.1 Encoding

Con la siguiente expresión se evitan problemas con el encoding al ejecutar el notebook. Es posible que deba ser eliminada o adaptada a la máquina en la que se ejecute el código.

[1]:
```
Sys.setlocale(category = "LC_ALL", locale = "es_ES.UTF-8")
```

'LC_CTYPE=es_ES.UTF-8;LC_NUMERIC=C;LC_TIME=es_ES.UTF-8;LC_COLLATE=es_ES.UTF-8;LC_MONETARY=es_ES.UTF-8;LC_MESSAGES=en_US.UTF-8;LC_PAPER=es_ES.UTF-8;LC_NAME=C;LC_ADDRESS=C;LC_TELEPHONE=C;LC_MEASUREMENT 8;LC_IDENTIFICATION=C'

0.3 Settings

0.3.1 Libraries to use

[15]:
```
library(readr)
library(dplyr)
# library(sf)
library(tidyr)
library(stringr)
library(ggplot2)
```

0.3.2 Paths

[3]:
```
iPath <- "Data/Input/"
oPath <- "Data/Output/"
```

0.4 Data Load

OPCION A: Seleccionar fichero en ventana para mayor comodidad

Data load using the {tcltk} package. Ucomment the line if using this option

[4]:
```
# file_data <- tcltk::tk_choose.files(multi = FALSE)
```

OPCION B: Especificar el nombre de archivo

[4]:
```
iFile <- "CU_53_09.2_02_spi.csv"
file_data <- paste0(iPath, iFile)

if(file.exists(file_data)){
    cat("Se leerán datos del archivo: ", file_data)
} else{
    warning("Cuidado: el archivo no existe.")
}
```

```
Se leerán datos del archivo:  Data/Input/CU_53_09.2_02_spi.csv
```

Data file to dataframe Usar la función adecuada según el formato de entrada (xlsx, csv, json, ...)

[5]:
```
data <- read_csv(file_data)
```

```
Rows: 2028 Columns: 18
  Column specification



Delimiter: ","
dbl (17): rank_score_spi, score_spi, score_bhn, score_fow, score_opp,
```

```
score_…
lgl  (1): is_train

 Use `spec()` to retrieve the full column specification for this
data.
 Specify the column types or set `show_col_types = FALSE` to quiet
this message.
```

Visualizo los datos.

Estructura de los datos:

[6]:
```
data |> glimpse()
```

```
Rows: 2,028
Columns: 18
$ rank_score_spi <dbl> 80, 97, 46, 84, 99, 150, 74, 105, 36,
143, 154, 69, 168…
$ score_spi      <dbl> 67.59, 60.10, 73.96, 62.86, 61.43,
45.57, 66.56, 59.45,…
$ score_bhn      <dbl> 79.16, 74.55, 81.88, 79.45, 77.84,
47.15, 80.41, 66.16,…
$ score_fow      <dbl> 65.40, 51.25, 70.69, 61.22, 57.63,
45.21, 62.82, 54.62,…
$ score_opp      <dbl> 58.22, 54.49, 69.32, 47.92, 48.83,
44.34, 56.46, 57.56,…
$ score_nbmc     <dbl> 86.67, 72.88, 86.33, 83.91, 87.72,
54.66, 92.38, 72.21,…
$ score_ws       <dbl> 86.44, 83.35, 88.07, 77.71, 78.15,
47.82, 78.47, 66.32,…
$ score_sh       <dbl> 87.69, 77.17, 89.59, 85.11, 86.61,
36.59, 85.21, 75.91,…
$ score_ps       <dbl> 55.85, 64.81, 63.55, 71.08, 58.87,
49.53, 65.57, 50.21,…
$ score_abk      <dbl> 74.20, 47.04, 89.07, 65.15, 55.79,
50.36, 81.61, 68.71,…
$ score_aic      <dbl> 74.19, 37.15, 68.14, 51.25, 78.17,
33.84, 61.95, 56.61,…
$ score_hw       <dbl> 53.55, 64.58, 61.41, 62.00, 45.35,
36.99, 61.64, 41.87,…
$ score_eq       <dbl> 59.66, 56.22, 64.13, 66.47, 51.22,
59.66, 46.07, 51.28,…
$ score_pr       <dbl> 81.60, 71.05, 90.28, 61.56, 60.41,
69.20, 70.02, 74.13,…
$ score_pfc      <dbl> 60.29, 64.77, 67.65, 56.51, 58.62,
40.61, 62.49, 59.83,…
$ score_incl     <dbl> 40.24, 56.12, 68.48, 48.70, 35.57,
41.81, 36.89, 55.73,…
$ score_aae      <dbl> 50.73, 26.03, 50.87, 24.90, 40.72,
```

```
25.72, 56.45, 40.54,…
$ is_train       <lgl> TRUE, TRUE, TRUE, TRUE, TRUE, TRUE,
TRUE, TRUE, TRUE, T…
```

Muestra de los primeros datos:

[7]:
```
data |> slice_head(n = 5)
```

A spec_tbl_df: 5 × 18

rank_score_spi <dbl>	score_spi <dbl>	score_bhn <dbl>	score_fow <dbl>	score_opp <dbl>	score_nbmc <dbl>	score_ <dbl>
80	67.59	79.16	65.40	58.22	86.67	86.44
97	60.10	74.55	51.25	54.49	72.88	83.35
46	73.96	81.88	70.69	69.32	86.33	88.07
84	62.86	79.45	61.22	47.92	83.91	77.71
99	61.43	77.84	57.63	48.83	87.72	78.15

0.5 Exploratory causal analysis

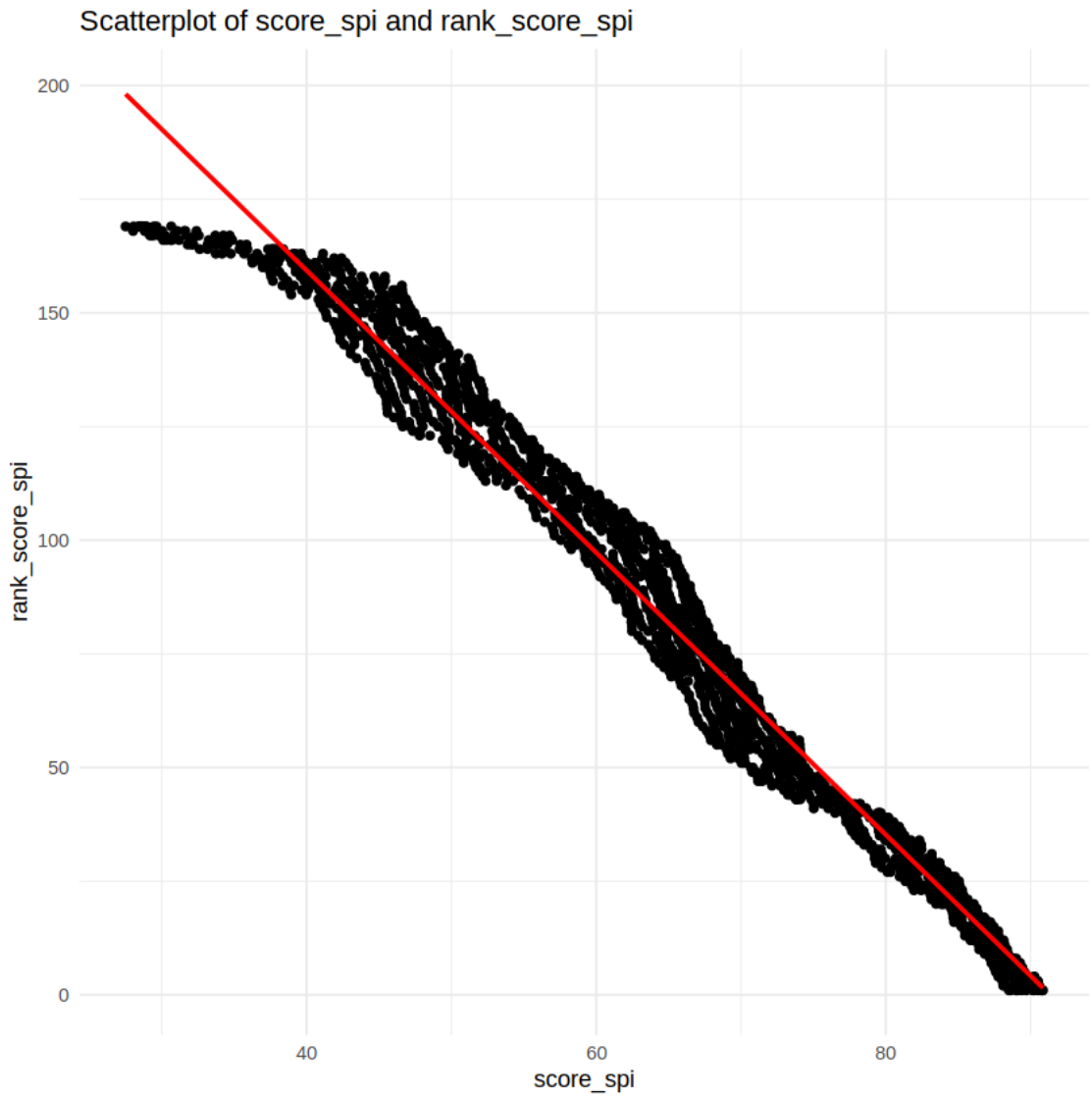
REFERENCE https://bookdown.org/paul/applied-causal-analysis/

Select columns

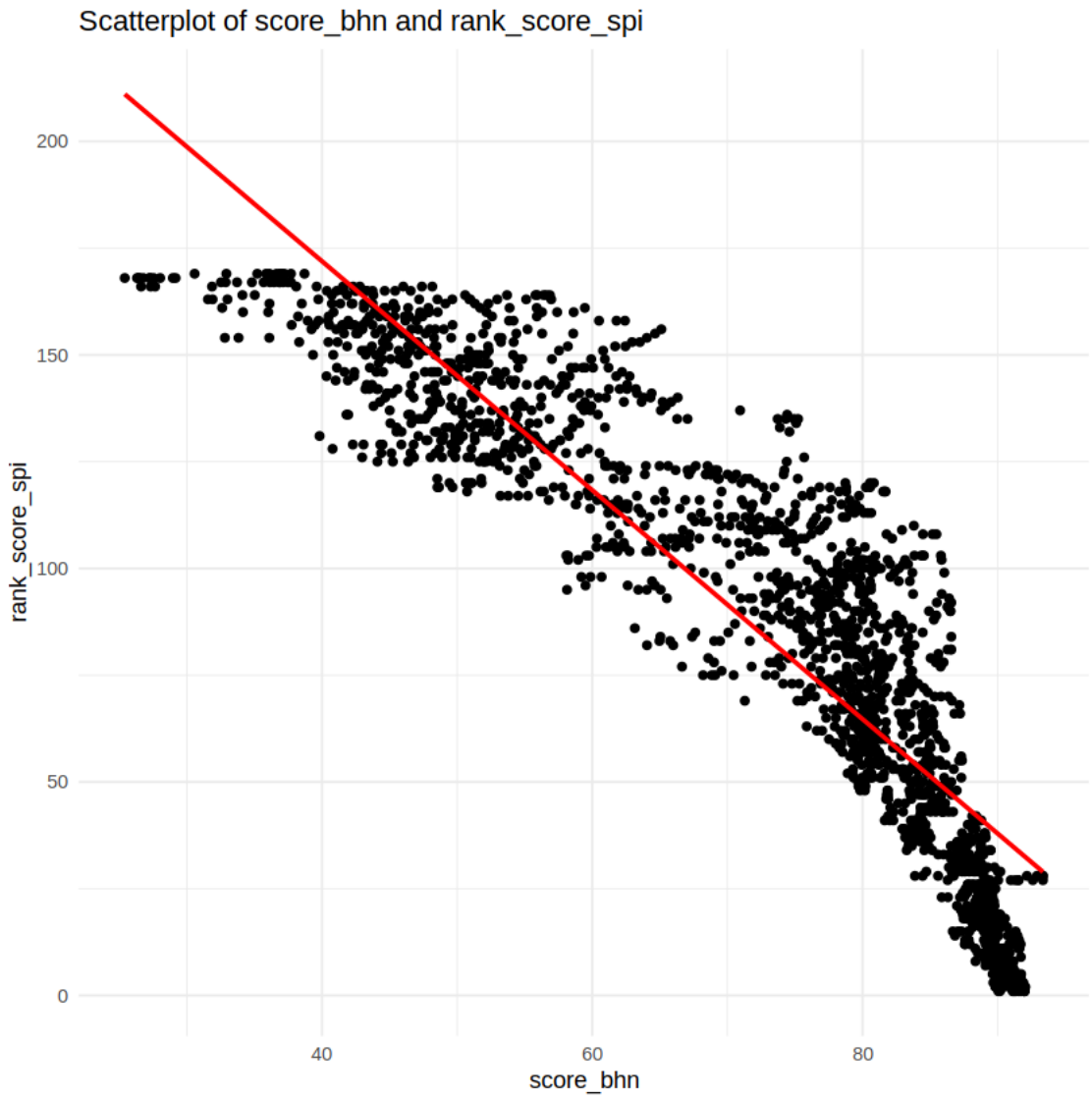
[16]:
```
# Seleccionamos las variables a analizar.
cols <- names(data[, -1])
```

[17]:
```
# Create scatterplots
for (col in cols) {
  if (is.numeric(data[[col]])) {
    p <- ggplot(data, aes_string(x = col, y = 'rank_score_spi')) +
      geom_point() +
      geom_smooth(method = "lm", se = FALSE, color = "red") +
      theme_minimal() +
      ggtitle(paste("Scatterplot of", col, "and rank_score_spi"))
    print(p)
  }
}
```

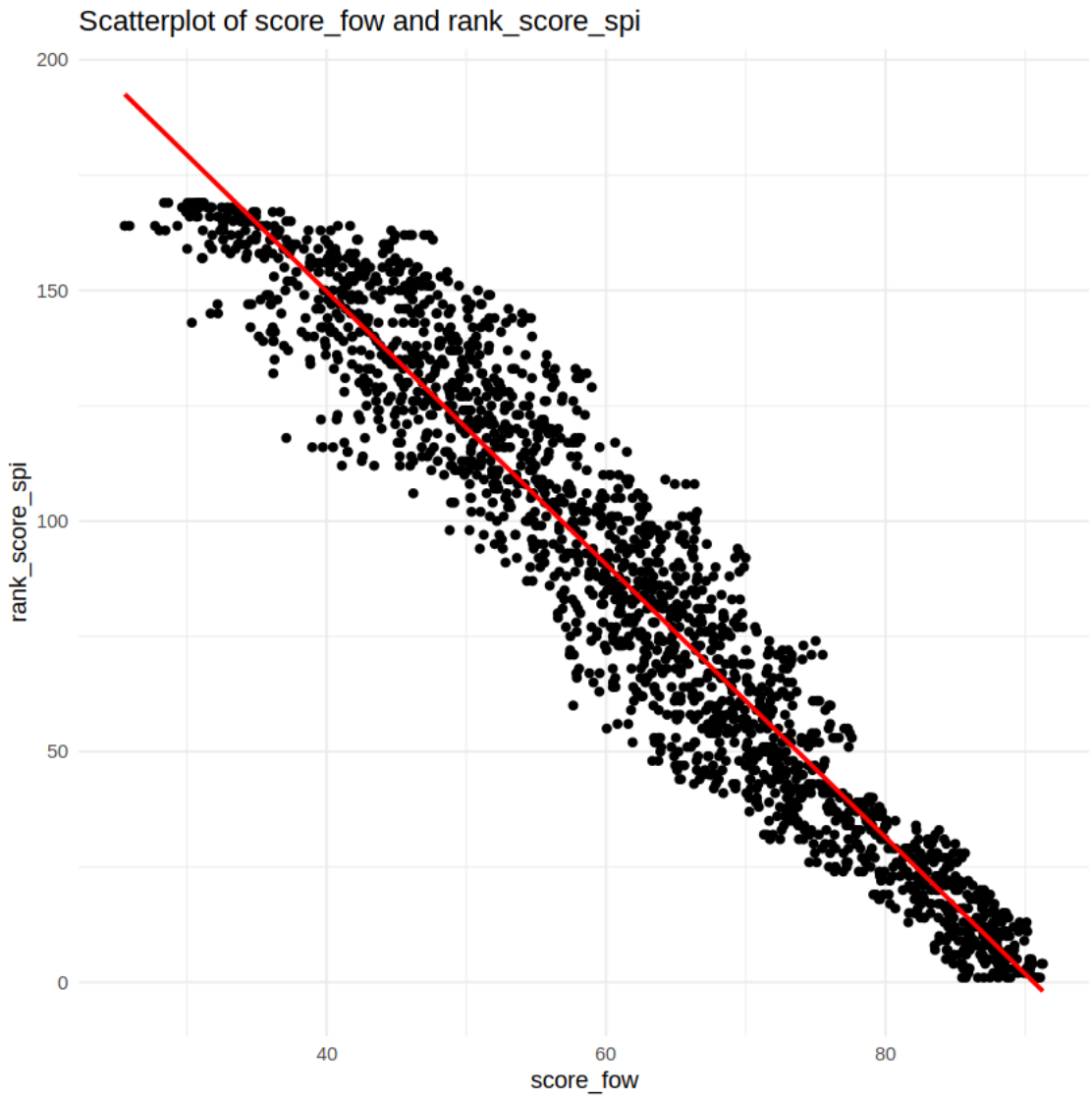
```
Warning message:
"`aes_string()` was deprecated in ggplot2 3.0.0.
 Please use tidy evaluation idioms with `aes()`.
 See also `vignette("ggplot2-in-packages")` for more information."
`geom_smooth()` using formula = 'y ~ x'
`geom_smooth()` using formula = 'y ~ x'
```



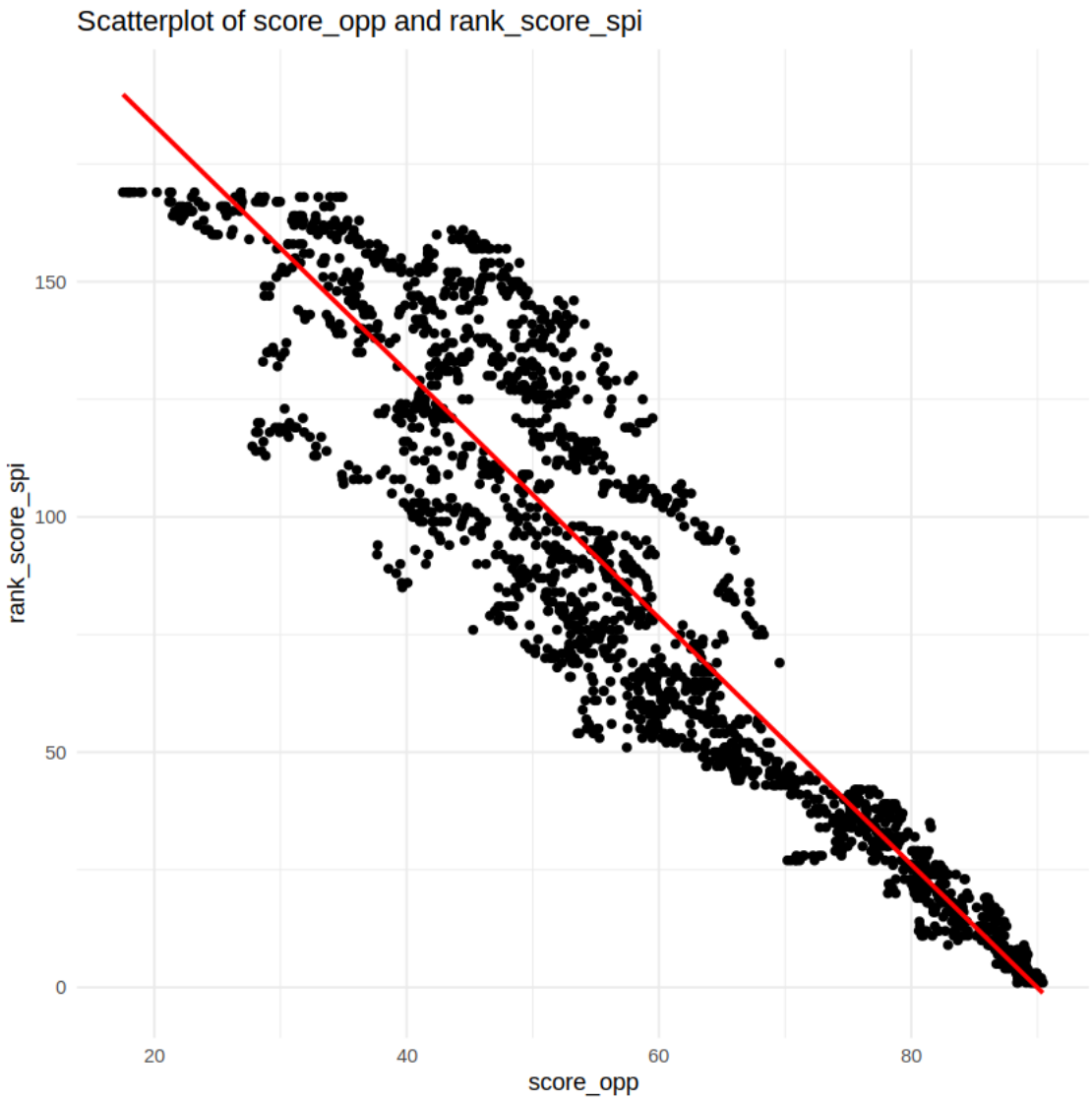

```
`geom_smooth()` using formula = 'y ~ x'
```



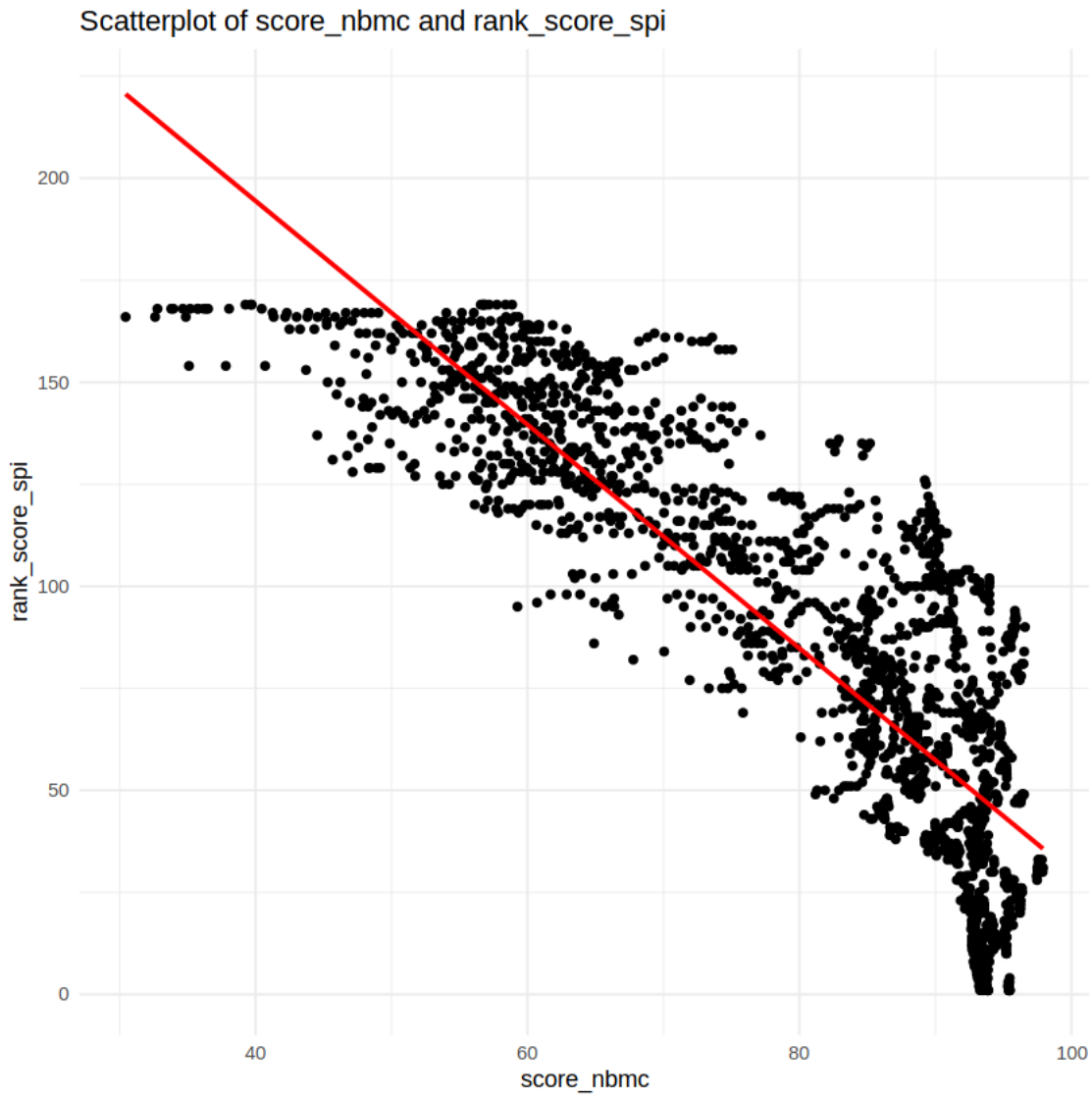

```
`geom_smooth()` using formula = 'y ~ x'
```



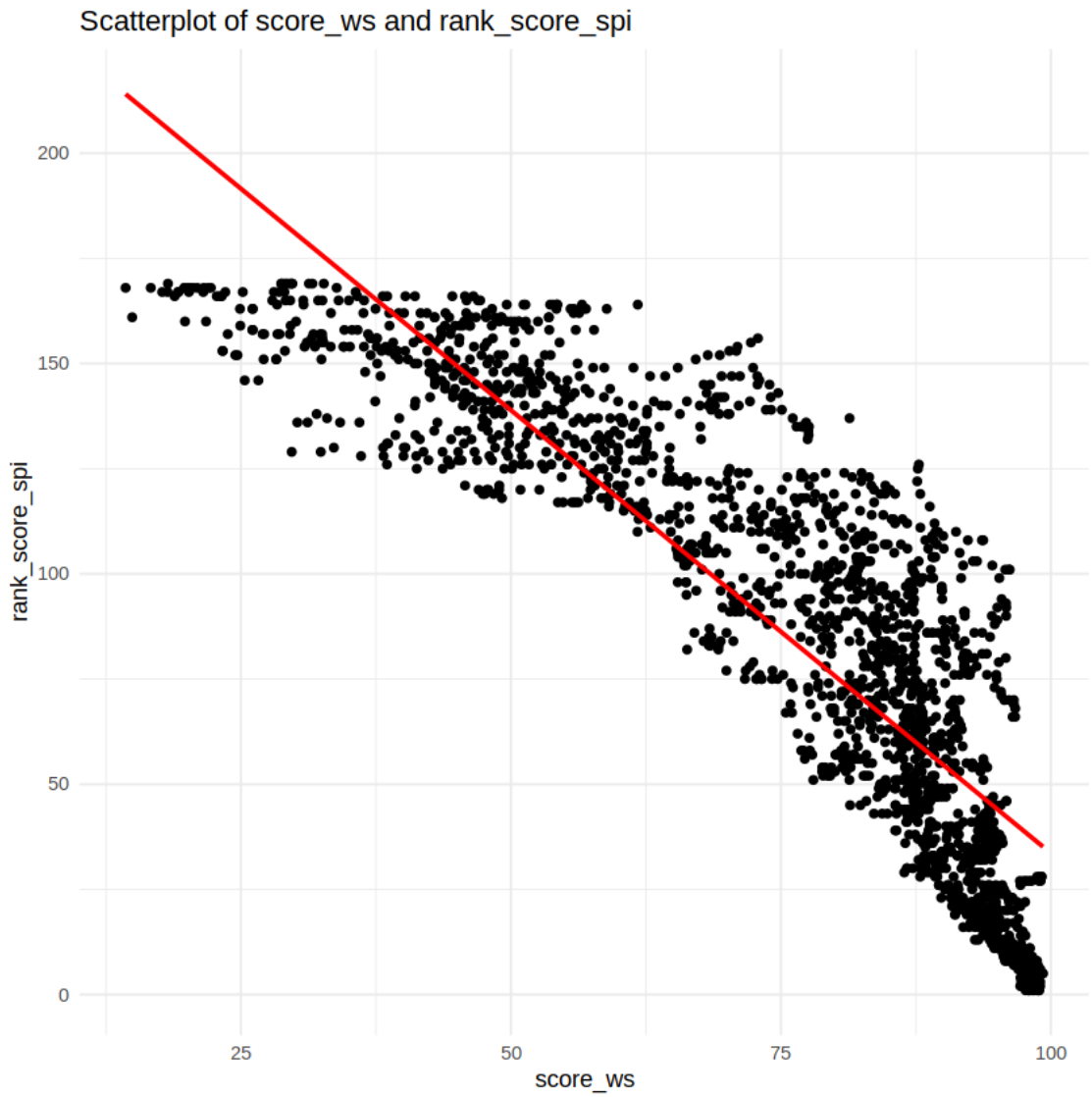

```
`geom_smooth()` using formula = 'y ~ x'
```



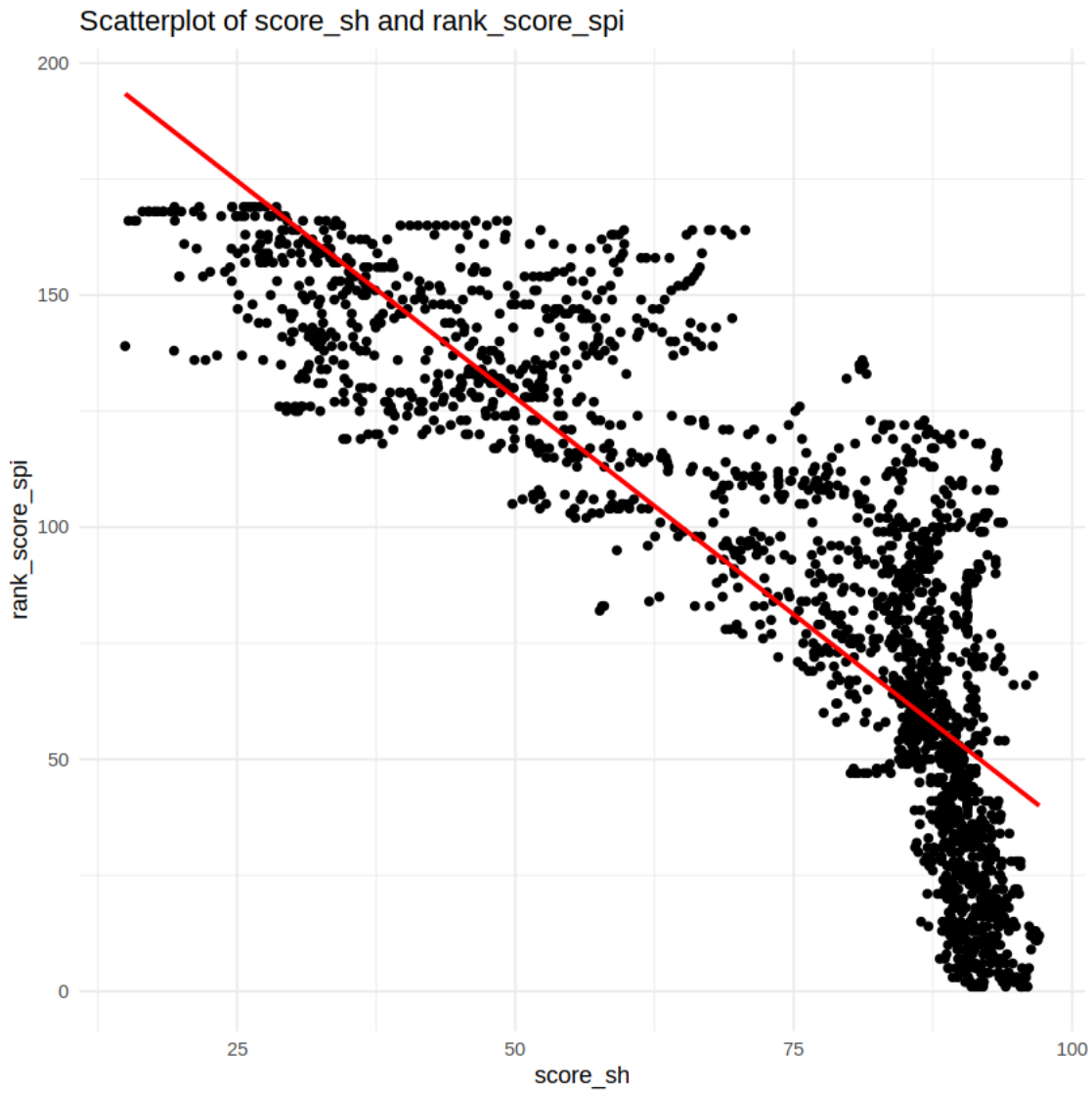

```
`geom_smooth()` using formula = 'y ~ x'
```



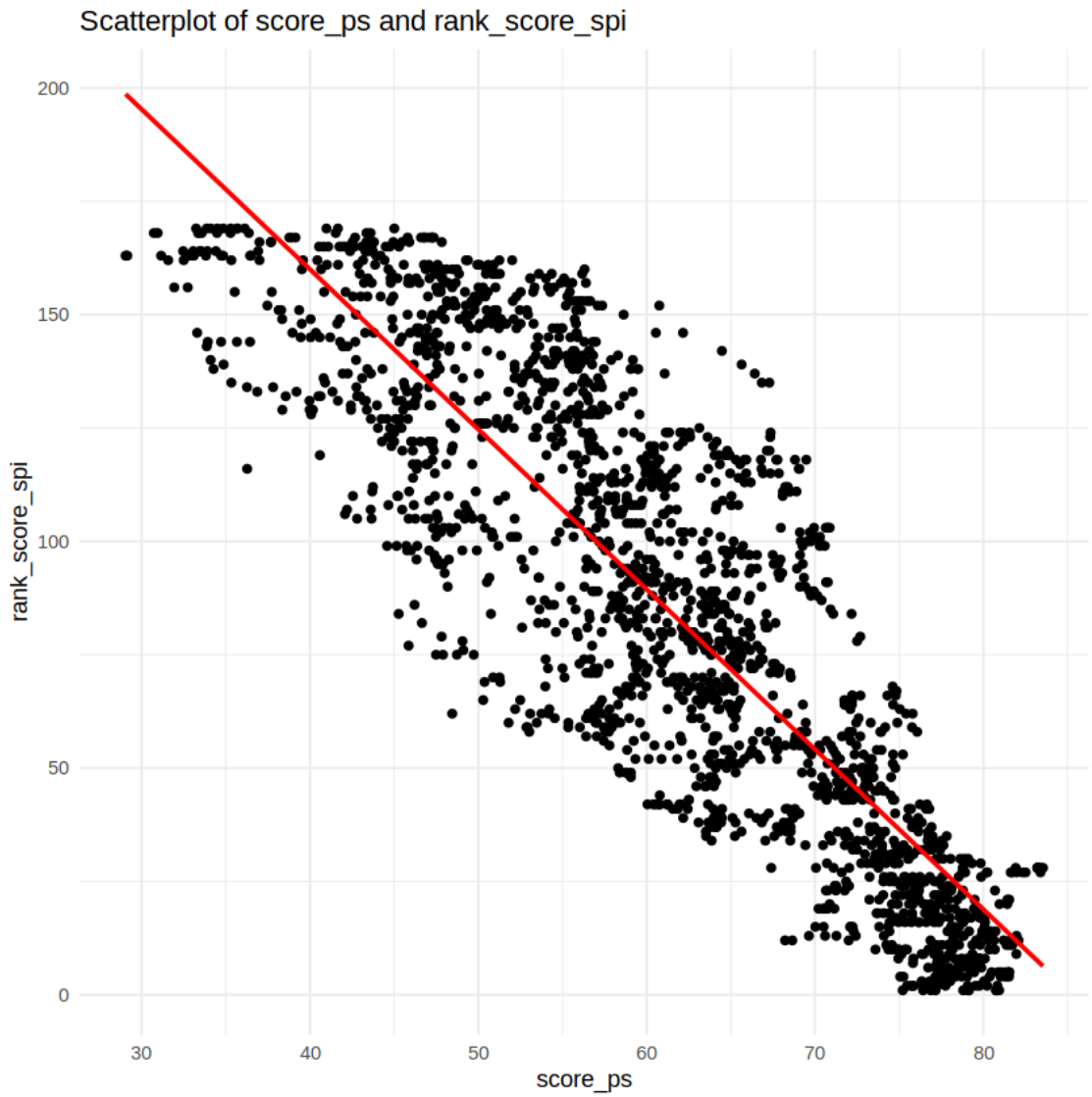

```
`geom_smooth()` using formula = 'y ~ x'
```



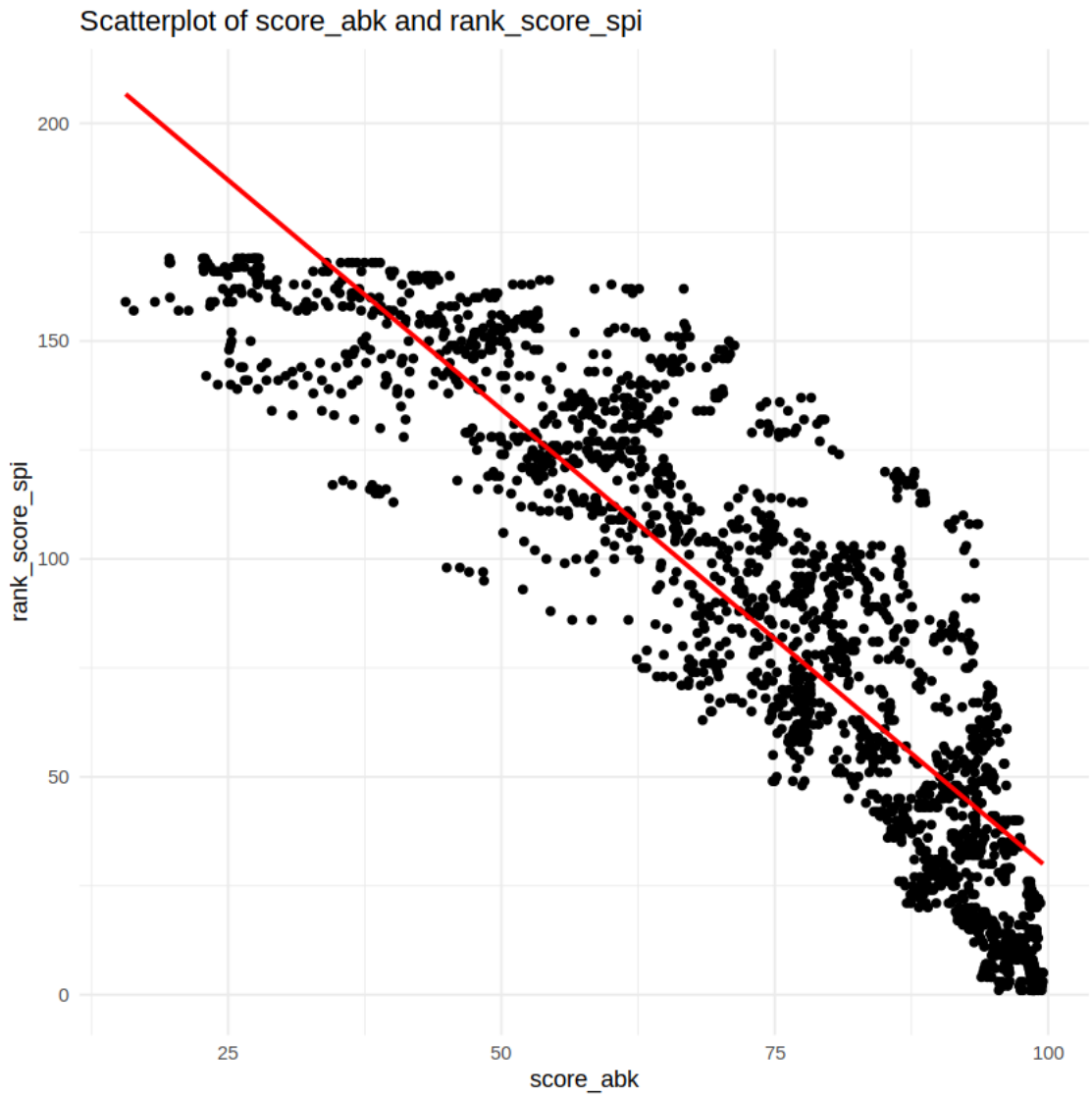

```
`geom_smooth()` using formula = 'y ~ x'
```



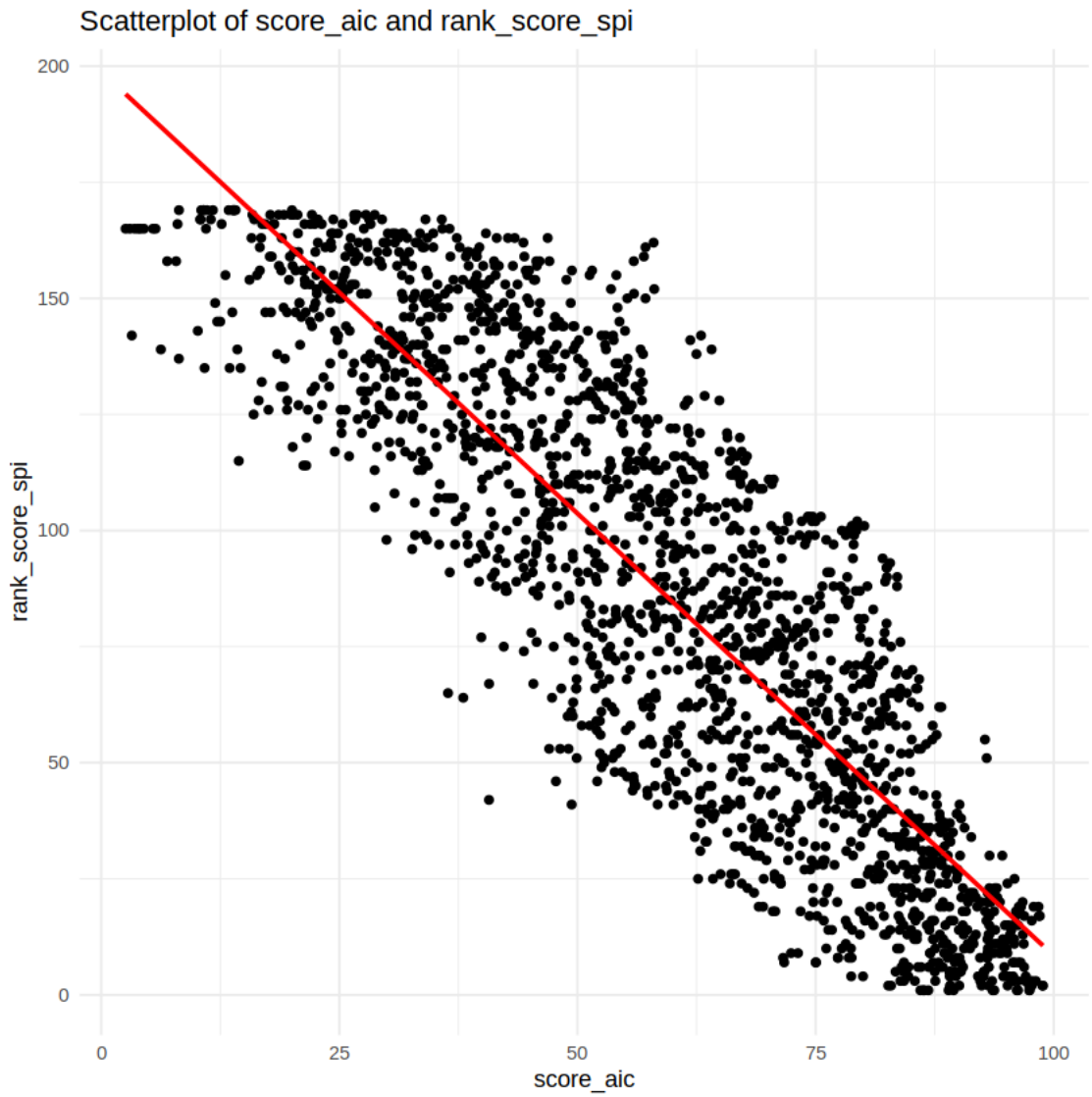

```
`geom_smooth()` using formula = 'y ~ x'
```



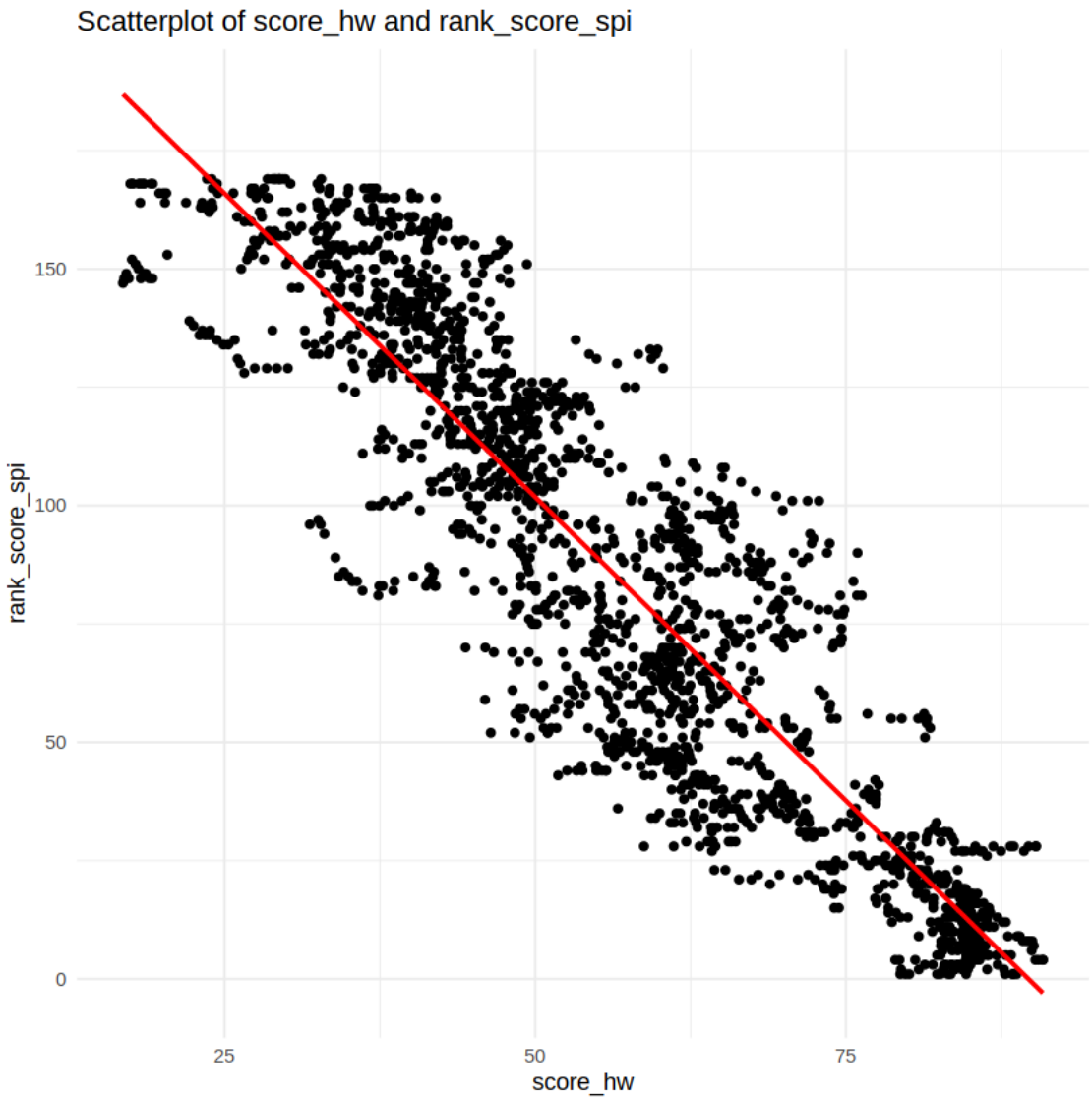

```
`geom_smooth()` using formula = 'y ~ x'
```



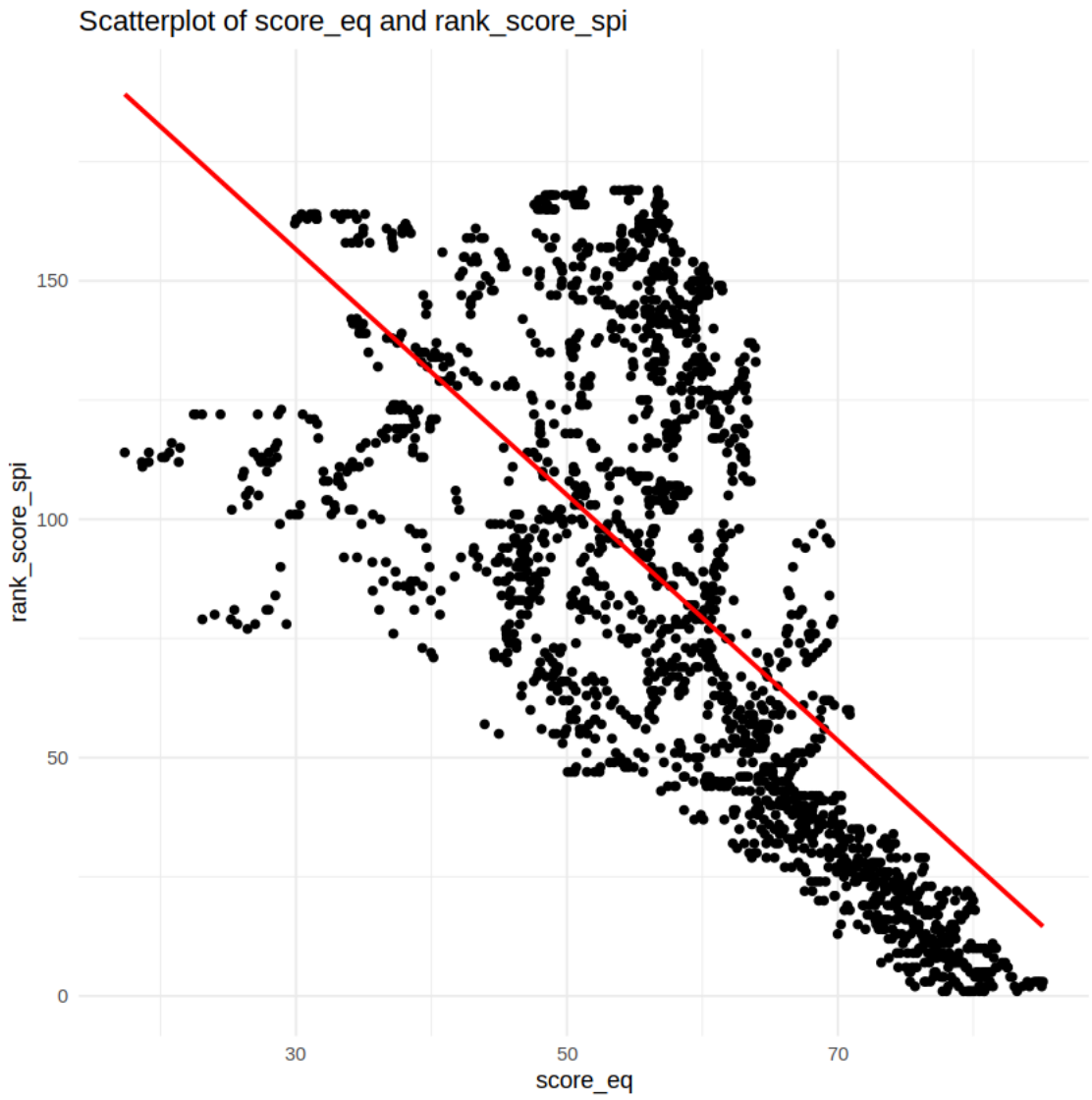

```
`geom_smooth()` using formula = 'y ~ x'
```



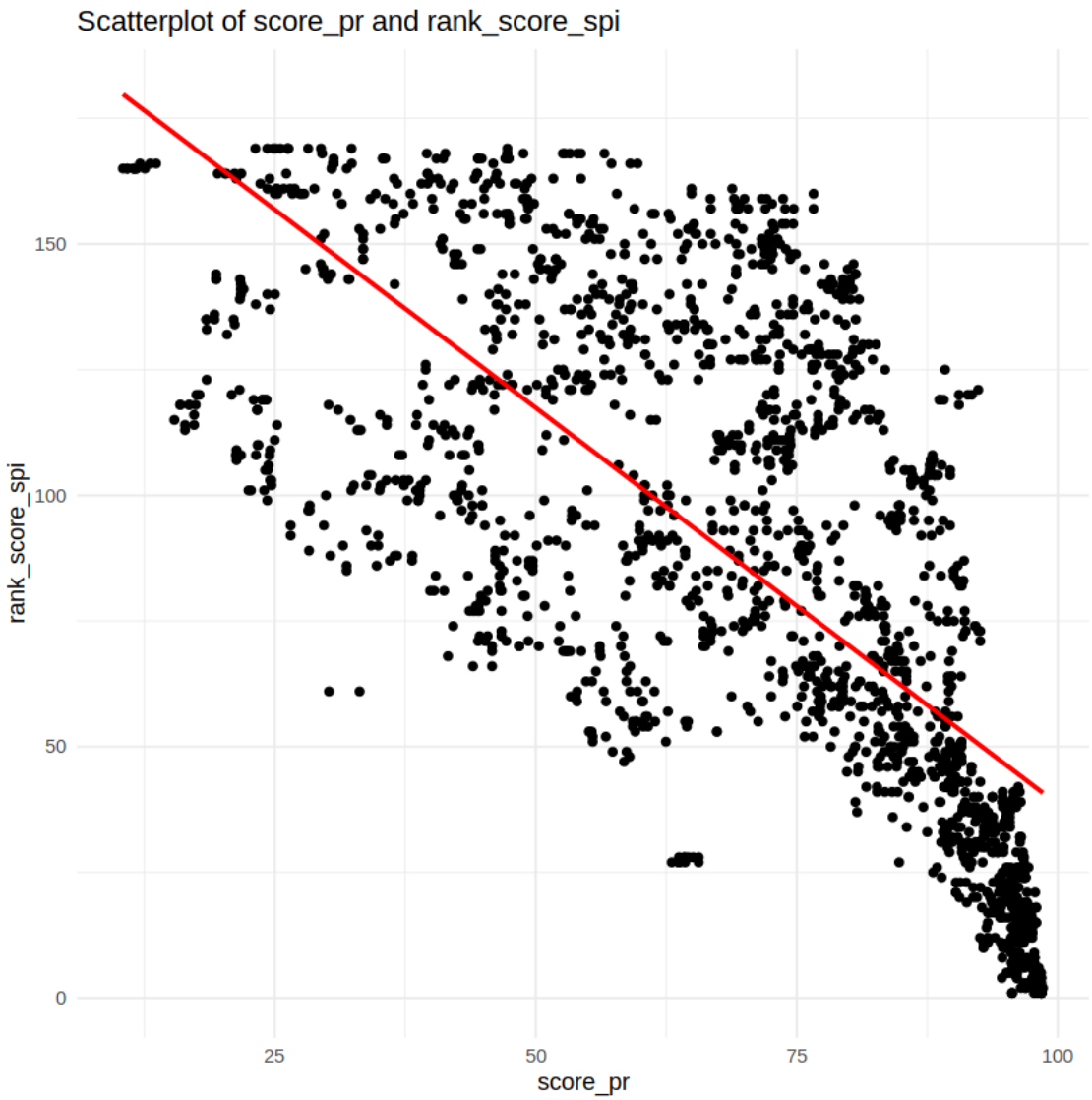

```
`geom_smooth()` using formula = 'y ~ x'
```



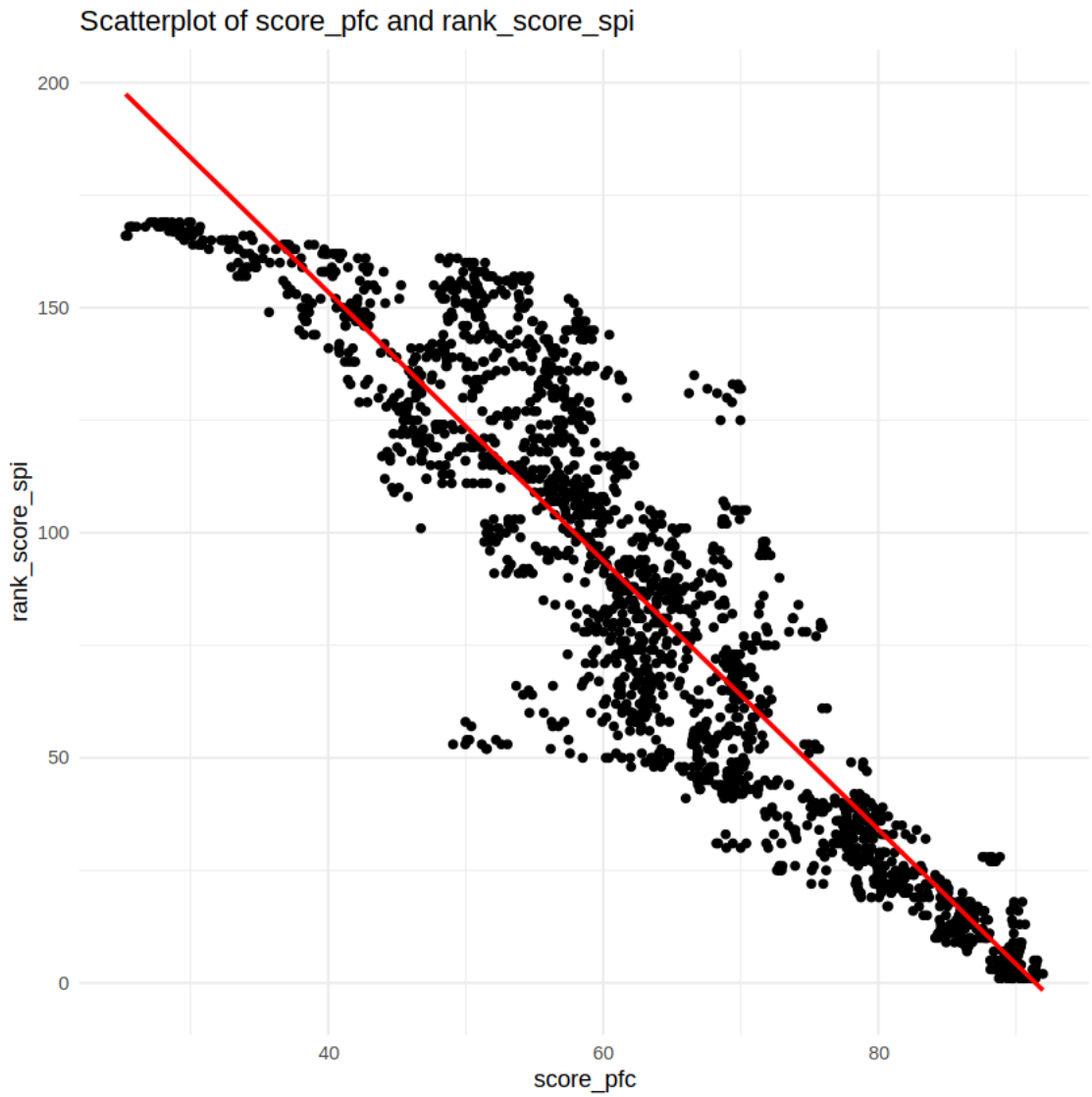

```
`geom_smooth()` using formula = 'y ~ x'
```



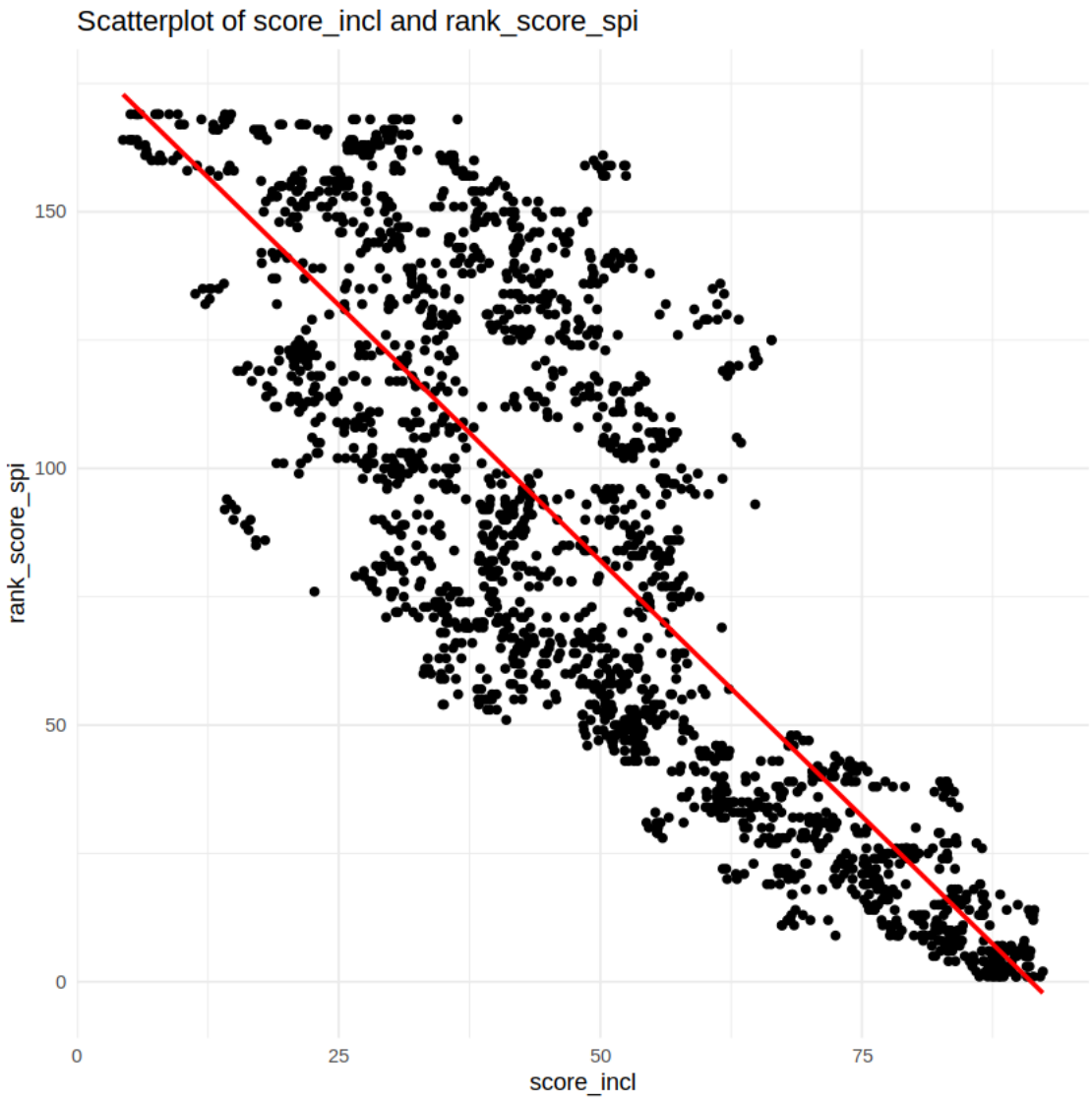

```
`geom_smooth()` using formula = 'y ~ x'
```

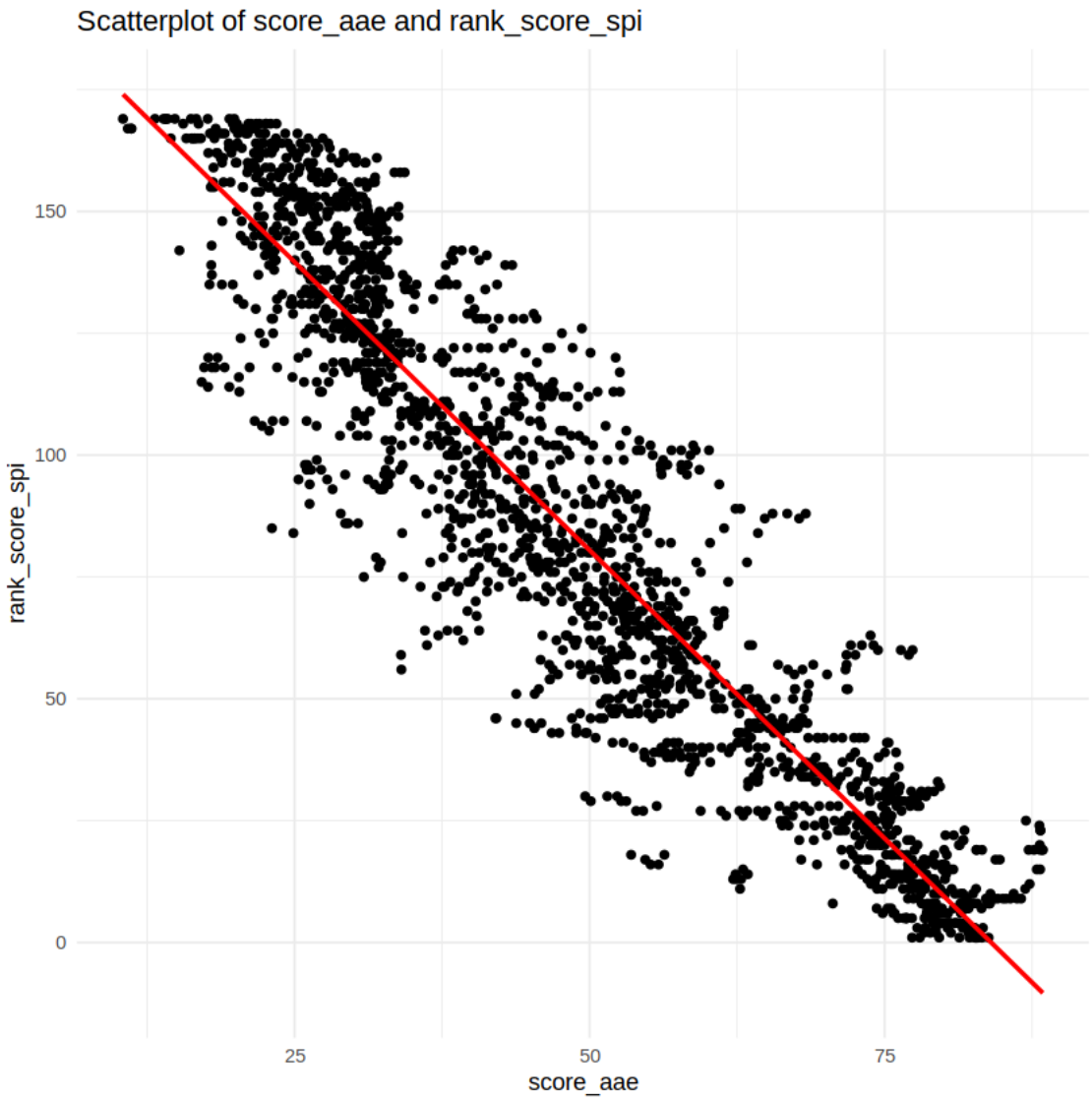



```
`geom_smooth()` using formula = 'y ~ x'
```




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


Scatterplot of score_incl and rank_score_spi
rank_score_spi
150
100
50
0
0
25
50
75
score_incl



0.6 Data Save

- Solo si se han hecho cambios
- No aplica

Identificamos los datos a guardar

[18]:
```
data_to_save <- data
```

Estructura de nombre de archivos:

- Código del caso de uso, por ejemplo “CU_04”
- Número del proceso que lo genera, por ejemplo ”_06”.
- Resto del nombre del archivo de entrada

- Extensión del archivo

Ejemplo: ”CU_04_06_01_01_zonasgeo.json, primer fichero que se genera en la tarea 01 del proceso 05 (Data Collection) para el caso de uso 04 (vacunas) y que se ha transformado en el proceso 06

Importante mantener los guiones bajos antes de proceso, tarea, archivo y nombre

0.6.1 Proceso 11

[19]:
```
caso <- "CU_53"
proceso <- '_11'
tarea <- "_02"
archivo <- ""
proper <- "_spi"
extension <- ".csv"
```

OPCION A: Uso del paquete “tcltk” para mayor comodidad

- Buscar carpeta, escribir nombre de archivo SIN extensión (se especifica en el código)
- Especificar sufijo2 si es necesario
- Cambiar datos por datos_xx si es necesario

[20]:
```
# file_save <- paste0(caso, proceso, tarea, tcltk::tkgetSaveFile(), proper,␣
 ↪extension)
# path_out <- paste0(oPath, file_save)
# write_csv(data_to_save_xxxx, path_out)

# cat('File saved as: ')
# path_out
```

OPCION B: Especificar el nombre de archivo

- Los ficheros de salida del proceso van siempre a Data/Output/.

[21]:
```
# file_save <- paste0(caso, proceso, tarea, archivo, proper, extension)
# path_out <- paste0(oPath, file_save)
# write_csv(data_to_save_xxxx, path_out)

# cat('File saved as: ')
# path_out
```

Copia del fichero a Input Si el archivo se va a usar en otros notebooks, copiar a la carpeta Input

[22]:
```
# path_in <- paste0(iPath, file_save)
# file.copy(path_out, path_in, overwrite = TRUE)
```

0.7 REPORT

A continuación se realizará un informe de las acciones realizadas

0.8 Main Actions Carried Out

- Al no haber target no se realiza análisis causal

0.9 Main Conclusions

- Los datos son adecuados para los modelos que se preveen

0.10 CODE TO DEPLOY (PILOT)

A continuación se incluirá el código que deba ser llevado a despliegue para producción, dado que se entiende efectúa operaciones necesarias sobre los datos en la ejecución del prototipo

Description

- No hay nada que desplegar en el piloto, ya que estos datos son estáticos o en todo caso cambian con muy poca frecuencia, altamente improbable durante el proyecto.

CODE

[ ]: