12.- Exploratory Data Analysis_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 *** > # 12.- EDA - Exploratory Data Analysis

0.1 Tasks

- Univariate Analysis
  - Data Structure Analysis
  - Data Types Analysis
  - Statistical Measures
  - Uniques Values
  - Continuous Variables Analysis
  - Categorical Variables analysis
    - Most frequent entry
    - Number of occurrences
- Normaluty Analysis
  - Data Distribution Analysis
    - Skew and Kurtosis
    - Omnibus K-squared test
    - Jarque-Bera tests
  - Visual Normality Checks
    - Histogram Plot
    - Quantile-Quantile Plot
  - Statistical Normality Tests
    - Shapiro-Wilk Test
    - D'Agostino's K^2 Test
    - Anderson-Darling Test
  - Transformations
    - Log
    - Square Root
    - Box-Cox
- Bi-variate Analysis
  - Continuous & Continuous variables analysis
    - Scatter plots
    - Correlation coefficients
      - Pearson
      - Kendall Tau
      - Spearman
    - Pairplot Visualization
  - Categorical & Continuous variables analysis
    - Categorical & Continuous
      - ANOVA
    - Continuous & Categorical
      - Box plots
      - Violin plots
      - Logistic Regression
  - Categorical & Categorical variables analysis
    - Contingency table
    - Pearson's Chi-Squared Test
- Hypothesis Test
  - z-test
  - t-test
- Regression Analysis
- Homogeneity Analysis
  - Chi-square test
- Stationary Analysis

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

[2]:
```
library(readr)
library(dplyr)
# library(sf)
library(tidyr)
library(ggplot2)
# library(summarytools)
library(GGally)
library(nortest)
library(lubridate)
```

```
Attaching package: ‘dplyr’


The following objects are masked from ‘package:stats’:

    filter, lag


The following objects are masked from ‘package:base’:

    intersect, setdiff, setequal, union


Registered S3 method overwritten by 'GGally':
  method from
  +.gg   ggplot2
```

```
Attaching package: ‘lubridate’


The following objects are masked from ‘package:base’:

    date, intersect, setdiff, union
```

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.
```

0.5 Data Structure

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

Tamaño de Memoria de los datos

[8]:
```
object.size(data)
```

```
296728 bytes
```

Structure of non-numerical features

[9]:
```
# Display non-numerical features
# Identify non-numerical columns
non_numeric_cols <- sapply(data, function(x) !is.numeric(x))

# Get the names of non-numerical columns
non_numeric_cols <- names(data)[non_numeric_cols]

# Print the non-numerical columns
print(non_numeric_cols)
```

```
[1] "is_train"
```

Structure of numerical features

[10]:
```
# Identify numerical columns
numeric_cols <- sapply(data, is.numeric)

# Subset the dataframe to include only numerical columns
numeric_data <- data[, numeric_cols]

# Display the structure of numerical features
str(numeric_data)
```

```
tibble [2,028 × 17] (S3: tbl_df/tbl/data.frame)
 $ rank_score_spi: num [1:2028] 80 97 46 84 99 150 74 105 36 143 …
 $ score_spi     : num [1:2028] 67.6 60.1 74 62.9 61.4 …
 $ score_bhn     : num [1:2028] 79.2 74.6 81.9 79.4 77.8 …
 $ score_fow     : num [1:2028] 65.4 51.2 70.7 61.2 57.6 …
```

```
 $ score_opp      : num [1:2028] 58.2 54.5 69.3 47.9 48.8 …
 $ score_nbmc     : num [1:2028] 86.7 72.9 86.3 83.9 87.7 …
 $ score_ws       : num [1:2028] 86.4 83.3 88.1 77.7 78.2 …
 $ score_sh       : num [1:2028] 87.7 77.2 89.6 85.1 86.6 …
 $ score_ps       : num [1:2028] 55.8 64.8 63.5 71.1 58.9 …
 $ score_abk      : num [1:2028] 74.2 47 89.1 65.2 55.8 …
 $ score_aic      : num [1:2028] 74.2 37.2 68.1 51.2 78.2 …
 $ score_hw       : num [1:2028] 53.5 64.6 61.4 62 45.3 …
 $ score_eq       : num [1:2028] 59.7 56.2 64.1 66.5 51.2 …
 $ score_pr       : num [1:2028] 81.6 71.1 90.3 61.6 60.4 …
 $ score_pfc      : num [1:2028] 60.3 64.8 67.7 56.5 58.6 …
 $ score_incl     : num [1:2028] 40.2 56.1 68.5 48.7 35.6 …
 $ score_aae      : num [1:2028] 50.7 26 50.9 24.9 40.7 …
```

0.6 Data Types

Tipo de datos

[11]:
```
sapply(data, class)
glimpse(data)
```

rank_score_spi 'numeric' **score_spi** 'numeric' **score_bhn** 'numeric' **score_fow** 'numeric' **score_opp** 'numeric' **score_nbmc** 'numeric' **score_ws** 'numeric' **score_sh** 'numeric' **score_ps** 'numeric' **score_abk** 'numeric' **score_aic** 'numeric' **score_hw** 'numeric' **score_eq** 'numeric' **score_pr** 'numeric' **score_pfc** 'numeric' **score_incl** 'numeric' **score_aae** 'numeric' **is_train** 'logical'

```
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…
```

0.7 Statistical Measures

[12]:
```r
# Identify numeric columns
numeric_cols <- sapply(data, is.numeric)

# Loop through numeric columns and calculate statistics
for (column in names(data)[numeric_cols]) {
  cat("Statistical Measures for", column, ":\n")
  cat("Mean:", mean(data[[column]], na.rm = TRUE), "\n")
  cat("Median:", median(data[[column]], na.rm = TRUE), "\n")
  cat("Standard Deviation:", sd(data[[column]], na.rm = TRUE), "\n")
  cat("Minimum:", min(data[[column]], na.rm = TRUE), "\n")
  cat("Maximum:", max(data[[column]], na.rm = TRUE), "\n")
  cat("25th Percentile:", quantile(data[[column]], 0.25, na.rm = TRUE), "\n")
  cat("50th Percentile (Median):", quantile(data[[column]], 0.5, na.rm = TRUE),
 ↪"\n")
  cat("75th Percentile:", quantile(data[[column]], 0.75, na.rm = TRUE), "\n\n")
}
```

```
Statistical Measures for rank_score_spi :
Mean: 85
Median: 85
Standard Deviation: 48.79728
Minimum: 1
Maximum: 169
25th Percentile: 43
50th Percentile (Median): 85
75th Percentile: 127

Statistical Measures for score_spi :
Mean: 63.94841
```

```
Median: 64.92
Standard Deviation: 15.53372
Minimum: 27.5
Maximum: 90.85
25th Percentile: 51.0325
50th Percentile (Median): 64.92
75th Percentile: 76.1775

Statistical Measures for score_bhn :
Mean: 72.42971
Median: 79.16
Standard Deviation: 16.42544
Minimum: 25.4
Maximum: 93.35
25th Percentile: 58.71
50th Percentile (Median): 79.16
75th Percentile: 85.52

Statistical Measures for score_fow :
Mean: 61.89
Median: 62.455
Standard Deviation: 15.85997
Minimum: 25.56
Maximum: 91.26
25th Percentile: 49.5
50th Percentile (Median): 62.455
75th Percentile: 73.705

Statistical Measures for score_opp :
Mean: 57.52548
Median: 55.52
Standard Deviation: 17.23898
Minimum: 17.52
Maximum: 90.42
25th Percentile: 44.93
50th Percentile (Median): 55.52
75th Percentile: 70.62

Statistical Measures for score_nbmc :
Mean: 79.92303
Median: 86.325
Standard Deviation: 15.27208
Minimum: 30.44
Maximum: 97.91
25th Percentile: 66.5175
50th Percentile (Median): 86.325
75th Percentile: 92.92
```

```
Statistical Measures for score_ws :
Mean: 75.58688
Median: 83.35
Standard Deviation: 20.10349
Minimum: 14.32
Maximum: 99.26
25th Percentile: 61.785
50th Percentile (Median): 83.35
75th Percentile: 91.11

Statistical Measures for score_sh :
Mean: 72.95862
Median: 85.005
Standard Deviation: 21.91298
Minimum: 14.96
Maximum: 97.05
25th Percentile: 55.4025
50th Percentile (Median): 85.005
75th Percentile: 89.76

Statistical Measures for score_ps :
Mean: 61.25034
Median: 61.365
Standard Deviation: 11.95775
Minimum: 29.05
Maximum: 83.54
25th Percentile: 52.76
50th Percentile (Median): 61.365
75th Percentile: 71.6225

Statistical Measures for score_abk :
Mean: 73.41254
Median: 77.72
Standard Deviation: 20.41318
Minimum: 15.62
Maximum: 99.53
25th Percentile: 59.5575
50th Percentile (Median): 77.72
75th Percentile: 91.32

Statistical Measures for score_aic :
Mean: 59.80123
Median: 61.56
Standard Deviation: 22.08924
Minimum: 2.56
Maximum: 98.87
25th Percentile: 43.22
50th Percentile (Median): 61.56
```

```
75th Percentile: 78.3775

Statistical Measures for score_hw :
Mean: 56.54411
Median: 56.28
Standard Deviation: 17.14786
Minimum: 16.83
Maximum: 90.84
25th Percentile: 42.975
50th Percentile (Median): 56.28
75th Percentile: 68.5625

Statistical Measures for score_eq :
Mean: 57.80218
Median: 58.02
Standard Deviation: 12.73231
Minimum: 17.36
Maximum: 85.14
25th Percentile: 50.3575
50th Percentile (Median): 58.02
75th Percentile: 66.395

Statistical Measures for score_pr :
Mean: 70.5549
Median: 75.02
Standard Deviation: 22.13732
Minimum: 10.47
Maximum: 98.57
25th Percentile: 54.3975
50th Percentile (Median): 75.02
75th Percentile: 89.8825

Statistical Measures for score_pfc :
Mean: 62.94486
Median: 62.405
Standard Deviation: 14.99453
Minimum: 25.27
Maximum: 91.97
25th Percentile: 52.8775
50th Percentile (Median): 62.405
75th Percentile: 72.4825

Statistical Measures for score_incl :
Mean: 48.50657
Median: 46.65
Standard Deviation: 20.0328
Minimum: 4.38
Maximum: 92.29
```

```
25th Percentile: 33.195
50th Percentile (Median): 46.65
75th Percentile: 61.6125

Statistical Measures for score_aae :
Mean: 48.0955
Median: 46.635
Standard Deviation: 19.19248
Minimum: 10.46
Maximum: 88.41
25th Percentile: 31.15
50th Percentile (Median): 46.635
75th Percentile: 63.44
```

0.8 Uniques values

[13]:
```
# Rthe number of unique values in each column.
data |> summarise(across(everything(), n_distinct))
```

A tibble: 1 × 18

rank_score_spi <int>	score_spi <int>	score_bhn <int>	score_fow <int>	score_opp <int>	score_nbmc <int>	score_ws <int>	s <
169	1699	1585	1680	1703	1493	1651	1

0.9 CrossTab

Select columns

Hacer los cruces que tengan sentido

[14]:
```
# data |> select(where(~ !is.numeric(.x))) |> colnames()
# Column1 <- "presMax"
# Column2 <- "velmedia"
```

Operation

[15]:
```
# Referencia cruzada de variables
# ctable(data[[Column1], data[[Column2]])
```

0.10 Analyzing Numerical Variables

0.10.1 Selecting continuous variables

[16]:
```
# Numeric colums
cdata <- data |> select(where(is.numeric))
```

0.10.2 Global view of the numerical variables

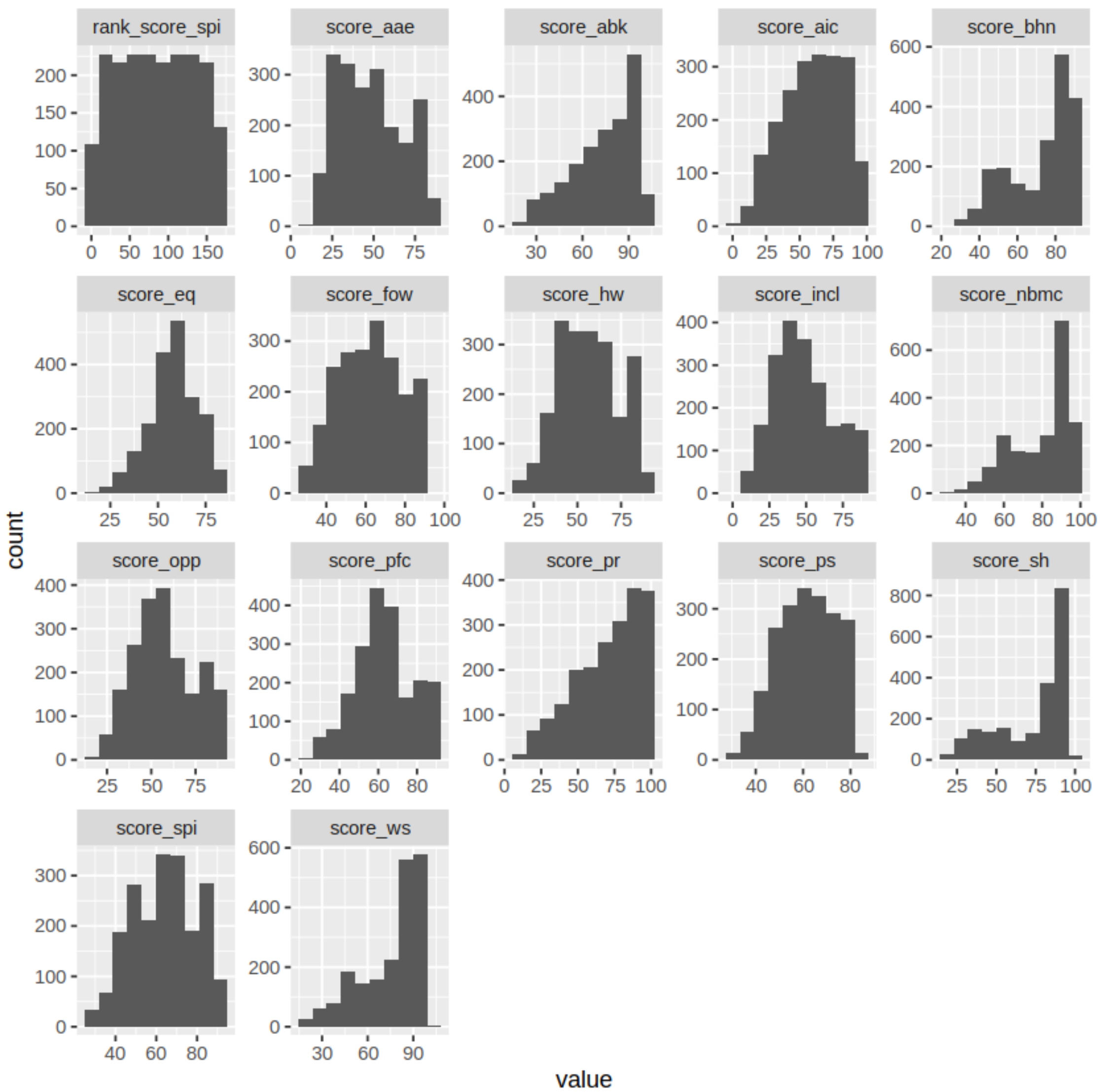
Global view on the dataset to identify some very unusual patterns.

NOTA: Esto puede tardar si hay muchas variables

[17]:
```r
# pairs(cdata)
# cdata |> ggpairs()
```

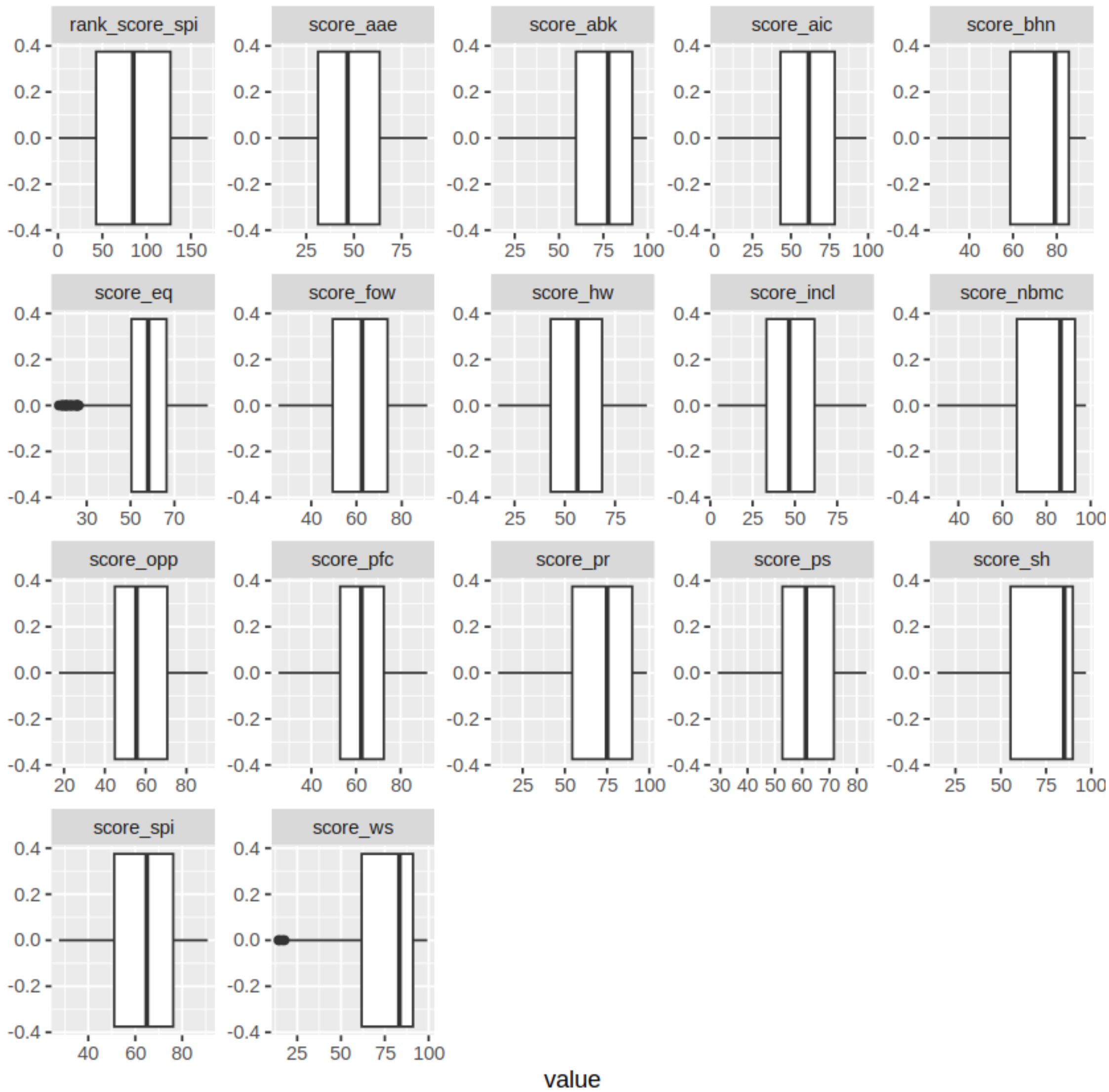
0.10.3 Histograms

[18]:
```r
cdata |>
  pivot_longer(cols = everything()) |>
  ggplot(aes(x = value)) +
  geom_histogram(bins = 10) +
  facet_wrap(~name, scales = "free")
```



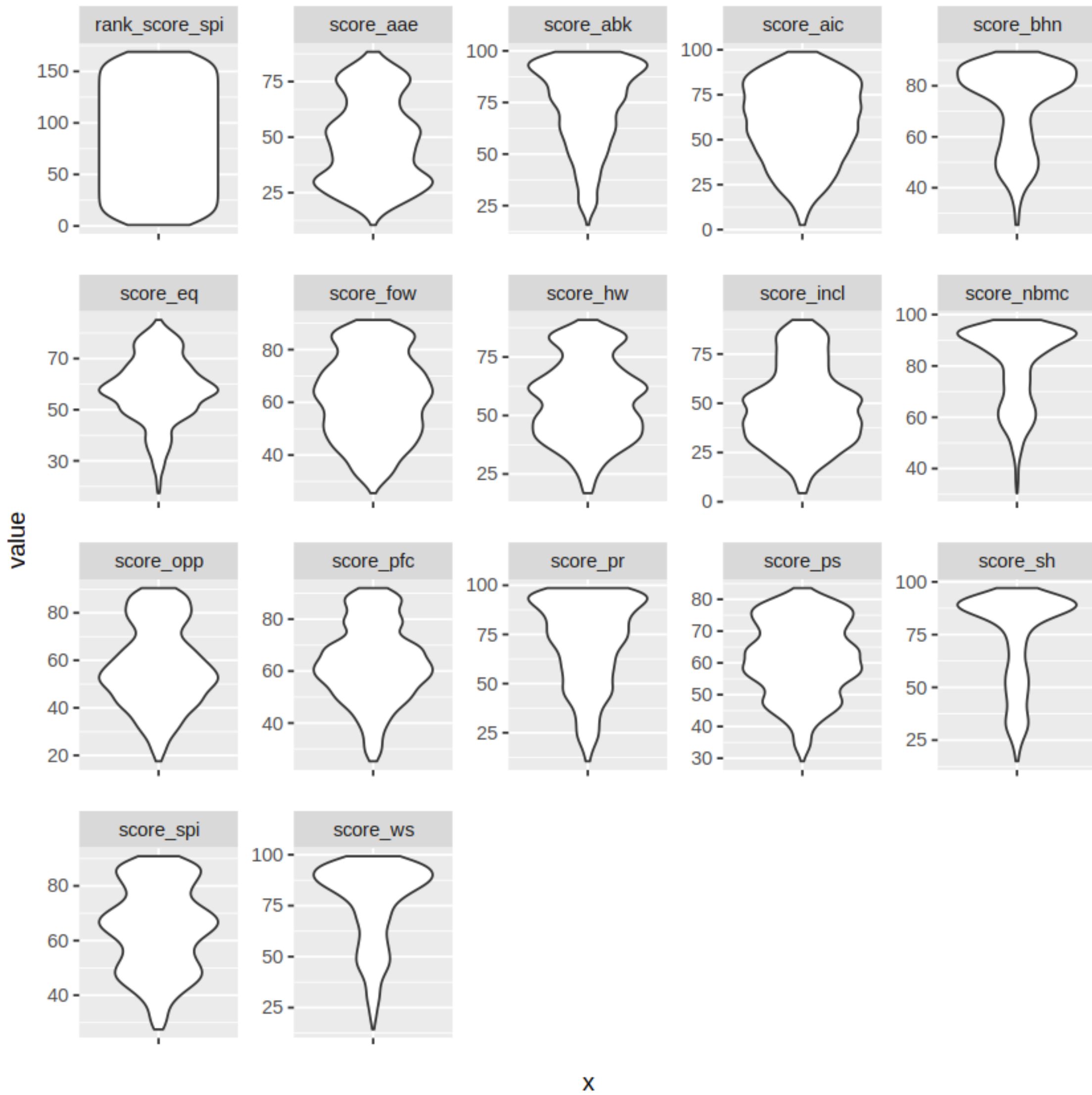
0.10.4 Box plot

[19]:
```
cdata |>
  pivot_longer(cols = everything()) |>
  ggplot(aes(x = value)) +
  geom_boxplot() +
  facet_wrap(~name, scales = "free")
```



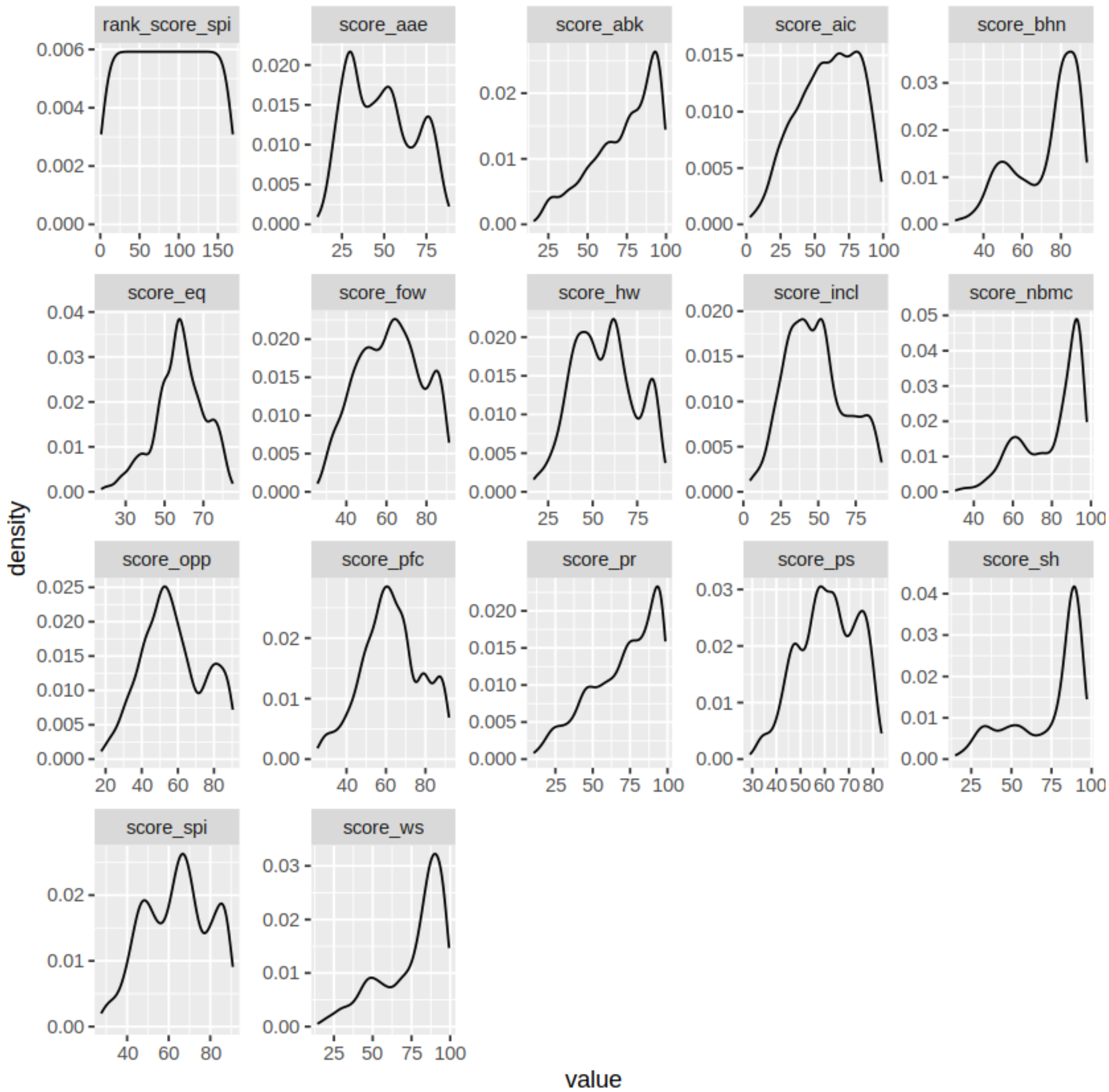
0.10.5 Violin plot

[20]:
```
cdata |>
  pivot_longer(cols = everything()) |>
  ggplot(aes(x = "", y = value)) +
  geom_violin() +
  facet_wrap(~name, scales = "free")
```



0.10.6 Distribution plot

[21]:
```
cdata |>
  pivot_longer(cols = everything()) |>
  ggplot(aes(x = value)) +
  geom_density() +
  facet_wrap(~name, scales = "free")
```



0.11 Analyzing Categorical Variables

0.11.1 Selecting categorical variables

[22]:
```
# Category colums
char_cols <- data |> select(where(~ !is.numeric(.x))) |> colnames()
char_cols
```

'is_train'

[23]:
```
# Category colums
char_data <- data |> select(where(~ !is.numeric(.x)))
char_data
```

A tibble: 2028 × 1

is_train
<lgl>
TRUE
TRUE
TRUE
TRUE
TRUE
TRUE
TRUE
TRUE
TRUE
TRUE
TRUE
TRUE
TRUE
TRUE
TRUE
TRUE
TRUE
TRUE
TRUE
TRUE
TRUE
TRUE
TRUE
TRUE
TRUE
TRUE
TRUE
TRUE
TRUE
TRUE
FALSE
FALSE
FALSE
FALSE
FALSE
FALSE
FALSE
FALSE
FALSE
FALSE
FALSE
FALSE
FALSE
FALSE
FALSE
FALSE
FALSE
FALSE
FALSE
FALSE

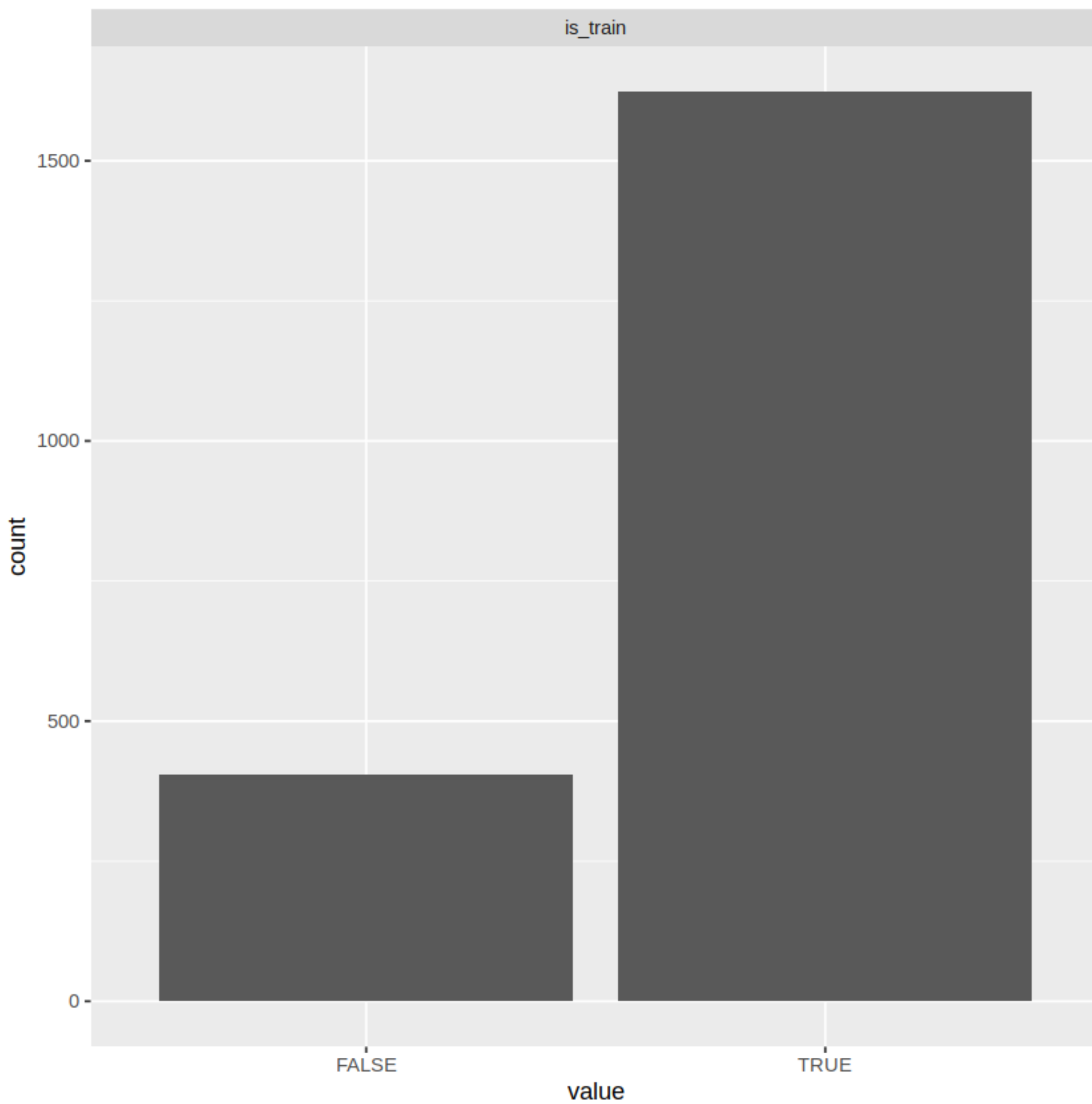
0.11.2 Most frequent entry

- Ver salida de `summarytools::freq()` arriba

[24]:
```
# Calculate and visualizate the ratio of the most frequent entry for each␣
↪feature
```

0.11.3 Visualization of categorical variables

[25]:
```
# returns a visualization of the number and frequency of categorical features
char_data |>
  pivot_longer(cols = everything()) |>
  ggplot(aes(x = value)) +
  geom_bar() +
  facet_wrap(~name, scales = "free")
```



0.12 Statistical Normality Tests

[26]:
```
cdata_long <- cdata |>
  pivot_longer(cols = everything())
```

0.12.1 Test de Shapiro-Wilk

Si hay muchos datos este no se puede hacer

[27]:
```
# tapply(cdata_long$value, cdata_long$name, shapiro.test)
```

0.12.2 Test de Anderson-Darling

[28]:
```
tapply(cdata_long$value, cdata_long$name, ad.test)
```

```
$rank_score_spi

        Anderson-Darling normality test

data:  X[[i]]
A = 22.558, p-value < 2.2e-16


$score_aae

        Anderson-Darling normality test

data:  X[[i]]
A = 27.001, p-value < 2.2e-16


$score_abk

        Anderson-Darling normality test

data:  X[[i]]
A = 46.657, p-value < 2.2e-16


$score_aic

        Anderson-Darling normality test

data:  X[[i]]
```

```
A = 13.554, p-value < 2.2e-16


$score_bhn

        Anderson-Darling normality test

data:  X[[i]]
A = 89.089, p-value < 2.2e-16


$score_eq

        Anderson-Darling normality test

data:  X[[i]]
A = 6.3578, p-value = 1.316e-15


$score_fow

        Anderson-Darling normality test

data:  X[[i]]
A = 9.4974, p-value < 2.2e-16


$score_hw

        Anderson-Darling normality test

data:  X[[i]]
A = 11.178, p-value < 2.2e-16


$score_incl

        Anderson-Darling normality test

data:  X[[i]]
A = 13.688, p-value < 2.2e-16


$score_nbmc

        Anderson-Darling normality test

data:  X[[i]]
```

```
A = 99.596, p-value < 2.2e-16


$score_opp

        Anderson-Darling normality test

data:  X[[i]]
A = 14.412, p-value < 2.2e-16


$score_pfc

        Anderson-Darling normality test

data:  X[[i]]
A = 5.5012, p-value = 1.451e-13


$score_pr

        Anderson-Darling normality test

data:  X[[i]]
A = 43.508, p-value < 2.2e-16


$score_ps

        Anderson-Darling normality test

data:  X[[i]]
A = 10.406, p-value < 2.2e-16


$score_sh

        Anderson-Darling normality test

data:  X[[i]]
A = 146.26, p-value < 2.2e-16


$score_spi

        Anderson-Darling normality test

data:  X[[i]]
```

```
A = 12.693, p-value < 2.2e-16


$score_ws

	Anderson-Darling normality test

data:  X[[i]]
A = 87.713, p-value < 2.2e-16
```

0.12.3 Test de Lilliefors

[29]:
```
tapply(cdata_long$value, cdata_long$name, lillie.test)
```

```
$rank_score_spi

	Lilliefors (Kolmogorov-Smirnov) normality test

data:  X[[i]]
D = 0.060109, p-value < 2.2e-16


$score_aae

	Lilliefors (Kolmogorov-Smirnov) normality test

data:  X[[i]]
D = 0.094213, p-value < 2.2e-16


$score_abk

	Lilliefors (Kolmogorov-Smirnov) normality test

data:  X[[i]]
D = 0.10047, p-value < 2.2e-16


$score_aic

	Lilliefors (Kolmogorov-Smirnov) normality test

data:  X[[i]]
D = 0.056094, p-value = 2.643e-16
```

```
$score_bhn

        Lilliefors (Kolmogorov-Smirnov) normality test

data:  X[[i]]
D = 0.17145, p-value < 2.2e-16


$score_eq

        Lilliefors (Kolmogorov-Smirnov) normality test

data:  X[[i]]
D = 0.046923, p-value = 3.475e-11


$score_fow

        Lilliefors (Kolmogorov-Smirnov) normality test

data:  X[[i]]
D = 0.044876, p-value = 3.515e-10


$score_hw

        Lilliefors (Kolmogorov-Smirnov) normality test

data:  X[[i]]
D = 0.054995, p-value = 1.227e-15


$score_incl

        Lilliefors (Kolmogorov-Smirnov) normality test

data:  X[[i]]
D = 0.056503, p-value < 2.2e-16


$score_nbmc

        Lilliefors (Kolmogorov-Smirnov) normality test

data:  X[[i]]
D = 0.17365, p-value < 2.2e-16
```

```
$score_opp

        Lilliefors (Kolmogorov-Smirnov) normality test

data:  X[[i]]
D = 0.062292, p-value < 2.2e-16


$score_pfc

        Lilliefors (Kolmogorov-Smirnov) normality test

data:  X[[i]]
D = 0.043894, p-value = 1.024e-09


$score_pr

        Lilliefors (Kolmogorov-Smirnov) normality test

data:  X[[i]]
D = 0.10357, p-value < 2.2e-16


$score_ps

        Lilliefors (Kolmogorov-Smirnov) normality test

data:  X[[i]]
D = 0.059325, p-value < 2.2e-16


$score_sh

        Lilliefors (Kolmogorov-Smirnov) normality test

data:  X[[i]]
D = 0.22424, p-value < 2.2e-16


$score_spi

        Lilliefors (Kolmogorov-Smirnov) normality test

data:  X[[i]]
D = 0.053873, p-value = 5.682e-15
```

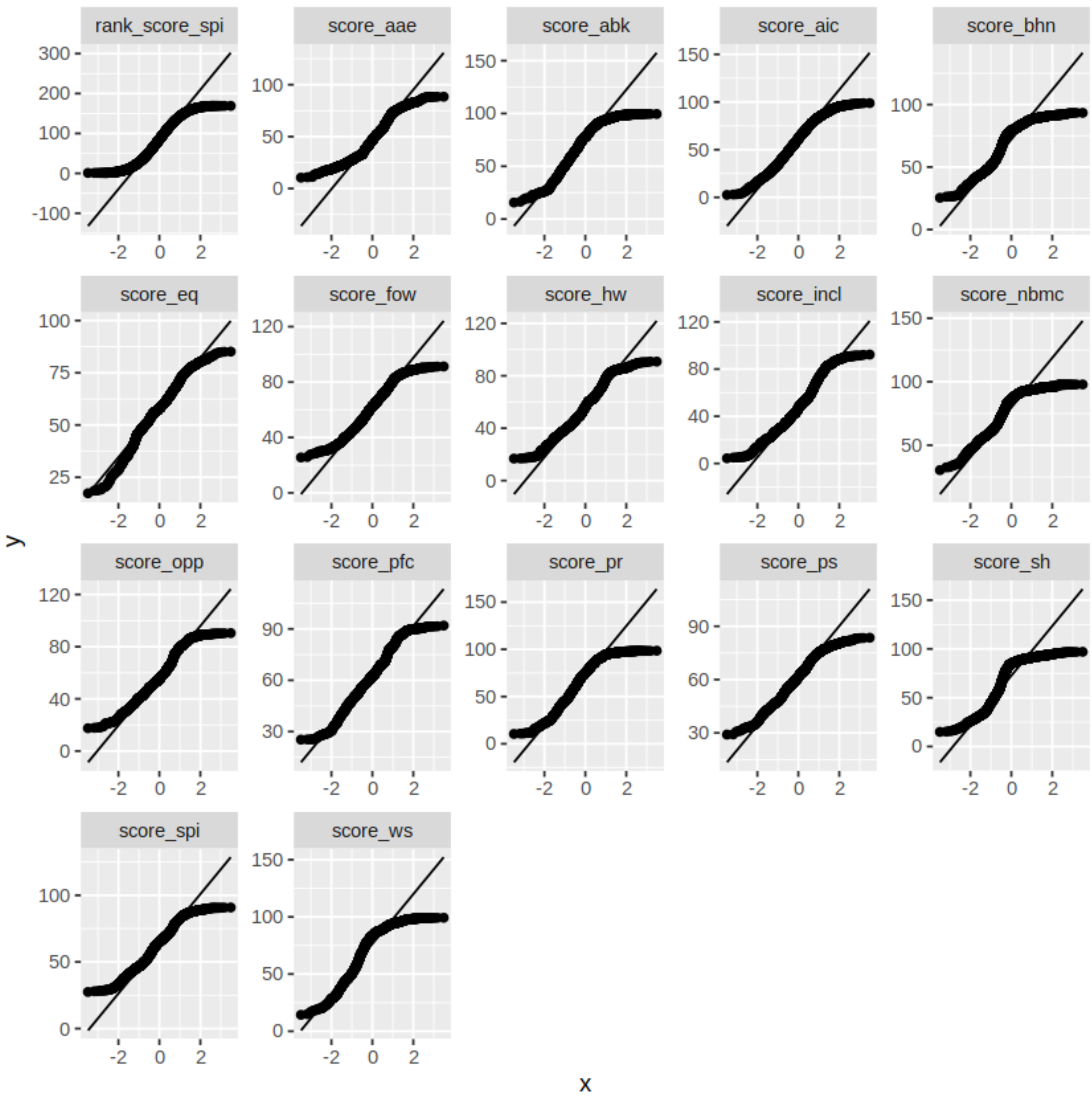
```
$score_ws

	Lilliefors (Kolmogorov-Smirnov) normality test

data:  X[[i]]
D = 0.16049, p-value < 2.2e-16
```

0.12.4 QQ-plots

[30]:
```
cdata |>
  pivot_longer(cols = everything()) |>
  ggplot(aes(sample = value)) +
  geom_qq() +
  geom_qq_line() +
  facet_wrap(~name, scales = "free")
```



0.13 Bivariate analysis

- Ver gráficos de dispersión y ggpairs arriba
- Completar si es necesario con alguna comparación específica (gráfico de dispersión o boxplot por grupos)

Correlaciones

[31]:
```
cor(cdata, use = "pairwise.complete.obs")
```

A matrix: 17 × 17 of type dbl

	rank_score_spi	score_spi	score_bhn	score_fow	score_
rank_score_spi	1.0000000	-0.9878106	-0.9021323	-0.9620901	-0.9256
score_spi	-0.9878106	1.0000000	0.9185031	0.9790282	0.92738
score_bhn	-0.9021323	0.9185031	1.0000000	0.8780754	0.72230
score_fow	-0.9620901	0.9790282	0.8780754	1.0000000	0.88991
score_opp	-0.9256091	0.9273803	0.7223005	0.8899141	1.00000
score_nbmc	-0.8576526	0.8739073	0.9693135	0.8397381	0.66625
score_ws	-0.8678145	0.8877350	0.9707138	0.8438714	0.69849
score_sh	-0.8390627	0.8556774	0.9670387	0.8232191	0.63434
score_ps	-0.8647591	0.8700078	0.8523566	0.8247451	0.78095
score_abk	-0.8808735	0.8908200	0.8634562	0.8986400	0.75864
score_aic	-0.8618812	0.9035901	0.8302835	0.9233468	0.80203
score_hw	-0.9014910	0.9079536	0.8530819	0.9161940	0.79869
score_eq	-0.6720218	0.6594053	0.4013381	0.7059798	0.75063
score_pr	-0.7155381	0.7096524	0.4552058	0.6494782	0.88712
score_pfc	-0.9172505	0.9306336	0.8096624	0.9171944	0.90045
score_incl	-0.8174928	0.8181303	0.5832981	0.7812701	0.93706
score_aae	-0.9303603	0.9323883	0.8286763	0.9161582	0.88803

0.14 Regression analysis

0.14.1 Modelo completo regresión lineal simple

[32]:
```
# modelo <- lm(xxxx ~ ., data = cdata)
# summary(modelo)
```

[33]:
```
# plot(modelo)
```

0.14.2 Selección de variables

Puede que dé error por la estructura de los datos, en ese caso dejarlo indicado

[40]:
```
# modelo2 <- step(modelo, trace = FALSE)
# summary(modelo2)
```

0.15 Stationary analysis

- Si hay una variable fecha, usarla
- Si hay mes, o semana, convertir a fecha

Todas las series, probablemente habría que filtrar por geografía

[ ]:
```

```

0.16 Data Save

- Solo si se han hecho cambios
- No aplica

Identificamos los datos a guardar

[41]:
```
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

0.16.1 Proceso 12

[42]:
```
caso <- "CU_53"
proceso <- '_12'
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

[43]:
```
# file_save <- paste0(caso, proceso, tarea, tcltk::tkgetSaveFile(), proper,␣
↪extension)
# path_out <- paste0(oPath, file_save)
# write_csv(data_to_save_xxxxx, 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/.

[44]:
```
# file_save <- paste0(caso, proceso, tarea, archivo, proper, extension)
# path_out <- paste0(oPath, file_save)
# write_csv(data_to_save_xxxxx, 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

[45]:
```
# path_in <- paste0(iPath, file_save)
# file.copy(path_out, path_in, overwrite = TRUE)
```

0.17 REPORT

A continuación se realizará un informe de las acciones realizadas

0.18 Main Actions Carried Out

- Se ha realizado exploratorio de los datos del caso de uso

0.19 Main Conclusions

- Los datos son adecuados para el caso de uso

0.20 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