12.- Exploratory Data Analysis_CU_18_20_infra_meteo_v_01

June 13, 2023

#

CU18_Infraestructuras_eventos

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_18_09.3_20_diario_infra
- 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

[5]:
```
iFile <- "CU_18_09.3_20_diario_infra.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_18_09.3_20_diario_infra.csv
```

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

[6]:
```
data <- read_csv(file_data)
```

```
Rows: 377727 Columns: 10
  Column specification

Delimiter: ","
dbl  (9): id_inf, capacidad, demanda, evento_infra, evento_zona, tmed,
prec,…
date (1): fecha
```

```
 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:

[7]:
```r
data |> glimpse()
```

```
Rows: 377,727
Columns: 10
$ id_inf       <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,
13, 14, 15, 16, 17…
$ fecha        <date> 2019-01-01, 2019-01-01, 2019-01-01,
2019-01-01, 2019-01-…
$ capacidad    <dbl> 993, 996, 1036, 1020, 992, 1026, 1007,
976, 1037, 972, 94…
$ demanda      <dbl> 883, 888, 922, 1134, 1103, 1139, 897,
1086, 1150, 861, 83…
$ evento_infra <dbl> 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0,
0, 0, 1, 0, 1, 1, …
$ evento_zona  <dbl> 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0,
1, 1, 1, 1, 0, 1, …
$ tmed         <dbl> 6.953211, 6.196420, 6.483569, 5.875797,
6.212680, 5.87854…
$ prec         <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, …
$ velmedia     <dbl> 0.6433886, 0.3417523, 0.4132169,
0.1820178, 0.2118110, 0.…
$ presMax      <dbl> 952.9357, 950.2191, 950.8051, 951.6768,
953.5118, 952.168…
```

Muestra de los primeros datos:

[8]:
```r
data |> slice_head(n = 5)
```

A spec_tbl_df: 5 × 10

id_inf <dbl>	fecha <date>	capacidad <dbl>	demanda <dbl>	evento_infra <dbl>	evento_zona <dbl>	tmed <dbl>	p <
1	2019-01-01	993	883	1	1	6.953211	0
2	2019-01-01	996	888	0	0	6.196420	0
3	2019-01-01	1036	922	0	0	6.483569	0
4	2019-01-01	1020	1134	1	0	5.875797	0
5	2019-01-01	992	1103	1	1	6.212680	0

Tamaño de Memoria de los datos

[9]:
```r
object.size(data)
```

30227008 bytes

Structure of non-numerical features

[10]:
```
# 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] "fecha"
```

Structure of numerical features

[11]:
```
# 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 [377,727 × 9] (S3: tbl_df/tbl/data.frame)
 $ id_inf      : num [1:377727] 1 2 3 4 5 6 7 8 9 10 …
 $ capacidad   : num [1:377727] 993 996 1036 1020 992 …
 $ demanda     : num [1:377727] 883 888 922 1134 1103 …
 $ evento_infra: num [1:377727] 1 0 0 1 1 0 0 1 1 0 …
 $ evento_zona : num [1:377727] 1 0 0 0 1 1 0 0 1 1 …
 $ tmed        : num [1:377727] 6.95 6.2 6.48 5.88 6.21 …
 $ prec        : num [1:377727] 0 0 0 0 0 0 0 0 0 0 …
 $ velmedia    : num [1:377727] 0.643 0.342 0.413 0.182 0.212 …
 $ presMax     : num [1:377727] 953 950 951 952 954 …
```

0.6 Data Types

Tipo de datos

[12]:
```
sapply(data, class)
glimpse(data)
```

id_inf 'numeric' **fecha** 'Date' **capacidad** 'numeric' **demanda** 'numeric' **evento_infra** 'numeric' **evento_zona** 'numeric' **tmed** 'numeric' **prec** 'numeric' **velmedia** 'numeric' **presMax** 'numeric'

```
Rows: 377,727
Columns: 10
```

```
$ id_inf       <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,
13, 14, 15, 16, 17…
$ fecha        <date> 2019-01-01, 2019-01-01, 2019-01-01,
2019-01-01, 2019-01-…
$ capacidad    <dbl> 993, 996, 1036, 1020, 992, 1026, 1007,
976, 1037, 972, 94…
$ demanda      <dbl> 883, 888, 922, 1134, 1103, 1139, 897,
1086, 1150, 861, 83…
$ evento_infra <dbl> 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0,
0, 0, 1, 0, 1, 1, …
$ evento_zona  <dbl> 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0,
1, 1, 1, 1, 0, 1, …
$ tmed         <dbl> 6.953211, 6.196420, 6.483569, 5.875797,
6.212680, 5.87854…
$ prec         <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, …
$ velmedia     <dbl> 0.6433886, 0.3417523, 0.4132169,
0.1820178, 0.2118110, 0.…
$ presMax      <dbl> 952.9357, 950.2191, 950.8051, 951.6768,
953.5118, 952.168…
```

0.7 Statistical Measures

[13]:

```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 id_inf :
Mean: 569.593
Median: 570
Standard Deviation: 328.4805
Minimum: 1
Maximum: 1138
25th Percentile: 285
50th Percentile (Median): 570
```

```
75th Percentile: 854

Statistical Measures for capacidad :
Mean: 999.8897
Median: 1000
Standard Deviation: 31.65462
Minimum: 851
Maximum: 1147
25th Percentile: 978
50th Percentile (Median): 1000
75th Percentile: 1021

Statistical Measures for demanda :
Mean: 999.921
Median: 999
Standard Deviation: 114.4421
Minimum: 757
Maximum: 1268
25th Percentile: 890
50th Percentile (Median): 999
75th Percentile: 1110

Statistical Measures for evento_infra :
Mean: 0.5002025
Median: 1
Standard Deviation: 0.5000006
Minimum: 0
Maximum: 1
25th Percentile: 0
50th Percentile (Median): 1
75th Percentile: 1

Statistical Measures for evento_zona :
Mean: 0.4996889
Median: 0
Standard Deviation: 0.5000006
Minimum: 0
Maximum: 1
25th Percentile: 0
50th Percentile (Median): 0
75th Percentile: 1

Statistical Measures for tmed :
Mean: 15.56518
Median: 14.56015
Standard Deviation: 7.680911
Minimum: -31.27333
Maximum: 48.0254
```

```
25th Percentile: 9.13574
50th Percentile (Median): 14.56015
75th Percentile: 21.60376

Statistical Measures for prec :
Mean: 1.120778
Median: 0
Standard Deviation: 4.047689
Minimum: -13.34451
Maximum: 75.77634
25th Percentile: 0
50th Percentile (Median): 0
75th Percentile: 0.1142545

Statistical Measures for velmedia :
Mean: 2.740693
Median: 2.366645
Standard Deviation: 1.639602
Minimum: -0.5362193
Maximum: 14.88656
25th Percentile: 1.600984
50th Percentile (Median): 2.366645
75th Percentile: 3.523233

Statistical Measures for presMax :
Mean: 935.2038
Median: 941.5102
Standard Deviation: 24.22767
Minimum: 813.7271
Maximum: 994.5682
25th Percentile: 933.9948
50th Percentile (Median): 941.5102
75th Percentile: 948.1539
```

0.8 Uniques values

[14]:

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

A tibble: 1 × 10

id_inf	fecha	capacidad	demanda	evento_infra	evento_zona	tmed	prec	velme
<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int
1138	332	265	483	2	2	376196	157761	37037

0.9 CrossTab

Select columns

Hacer los cruces que tengan sentido

[15]:
```
# data |> select(where(~ !is.numeric(.x))) |> colnames()
# Column1 <- "presMax"
# Column2 <- "velmedia"
```

Operation

[16]:
```
# Referencia cruzada de variables
# ctable(data[[Column1], data[[Column2]])
```

0.10 Analyzing Numerical Variables

0.10.1 Selecting continuous variables

[17]:
```
# Numeric colums
cdata <- data |> select(where(is.numeric))
```

0.10.2 Global view of the numerical variables

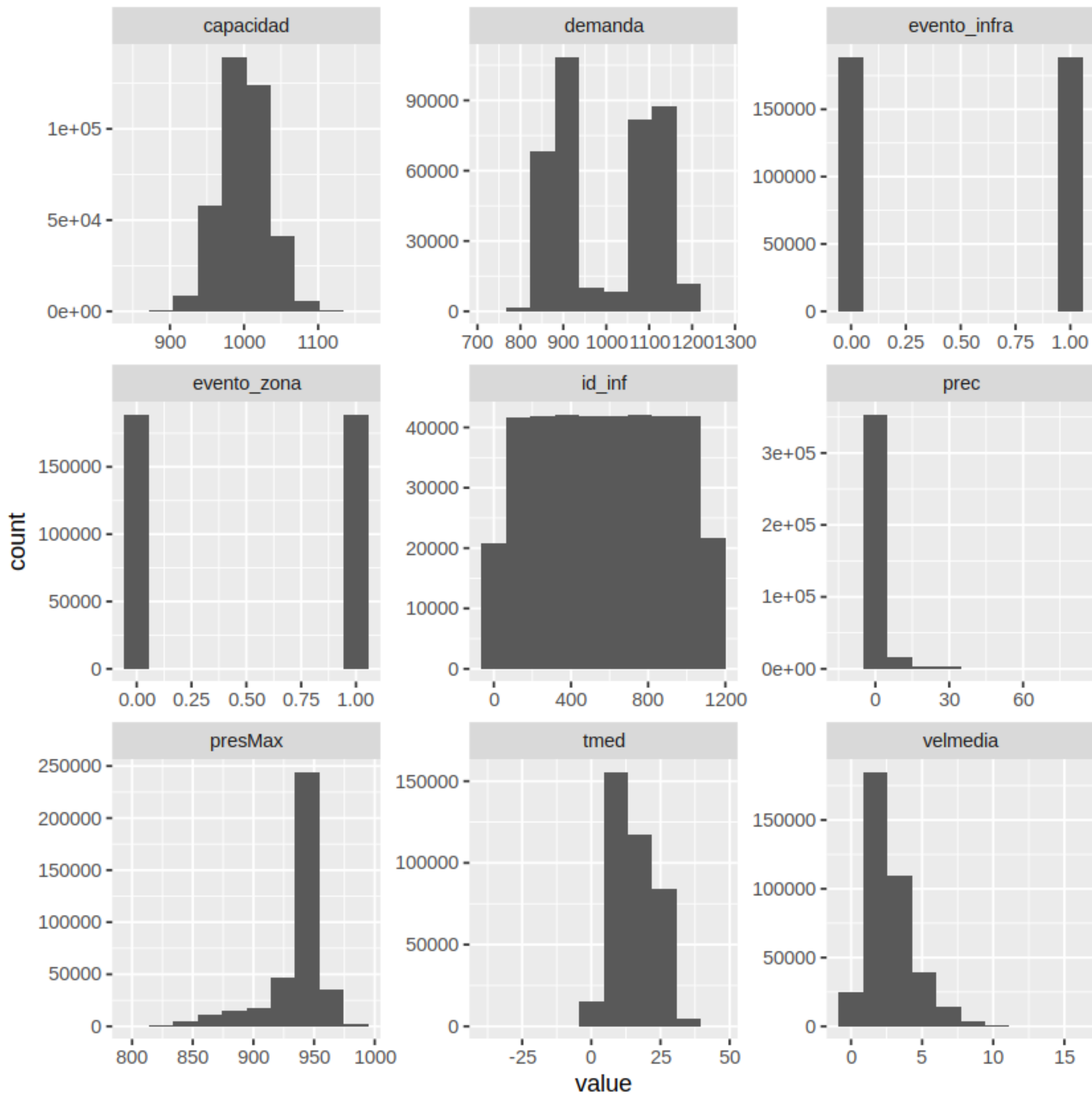
Global view on the dataset to identify some very unusual patterns.

NOTA: Esto puede tardar si hay muchas variables

[18]:
```
# pairs(cdata)
# cdata |> ggpairs()
```

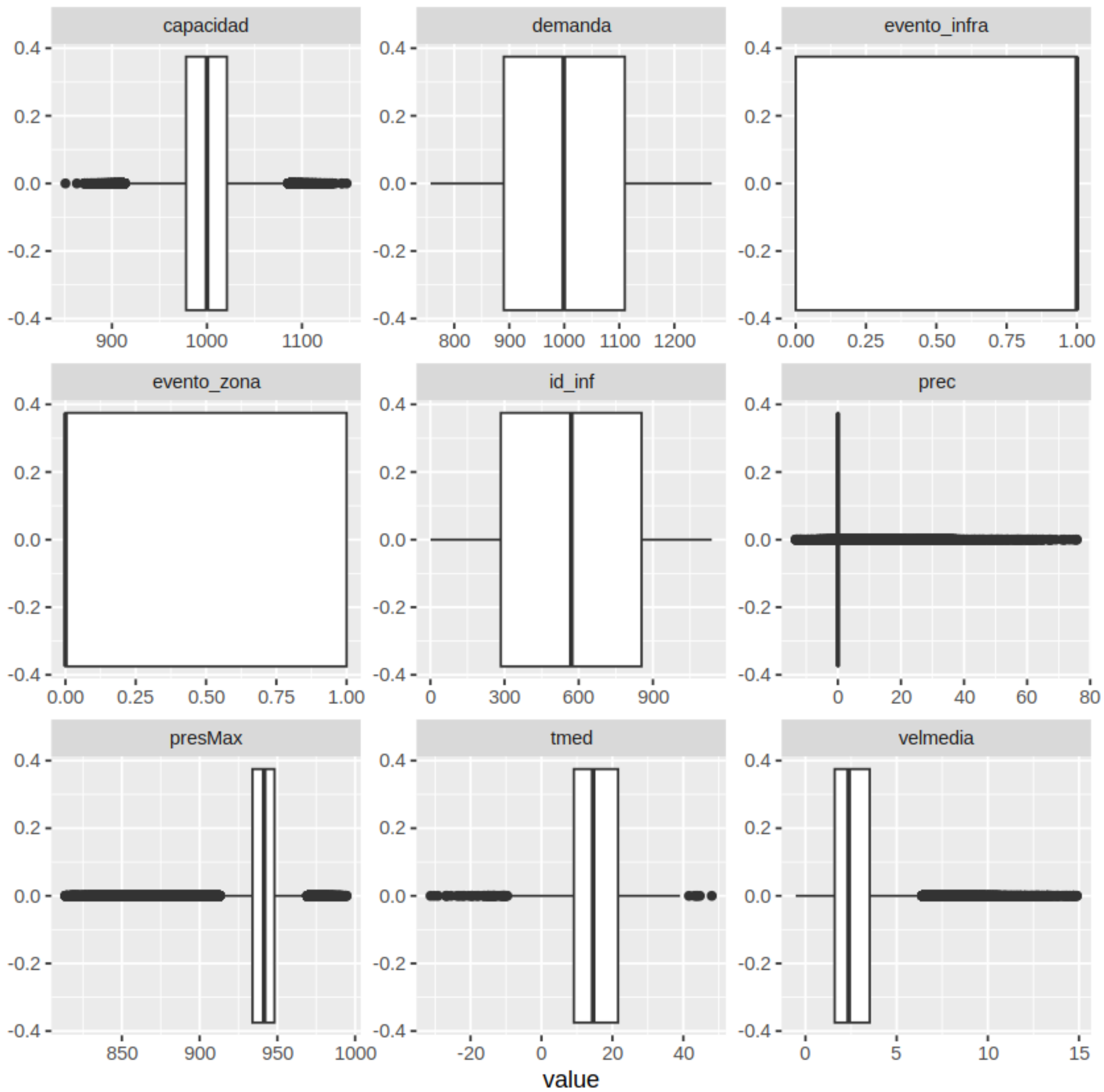
0.10.3 Histograms

[19]:
```
cdata |>
  pivot_longer(cols = everything()) |>
  ggplot(aes(x = value)) +
  geom_histogram(bins = 10) +
  facet_wrap(~name, scales = "free")
```



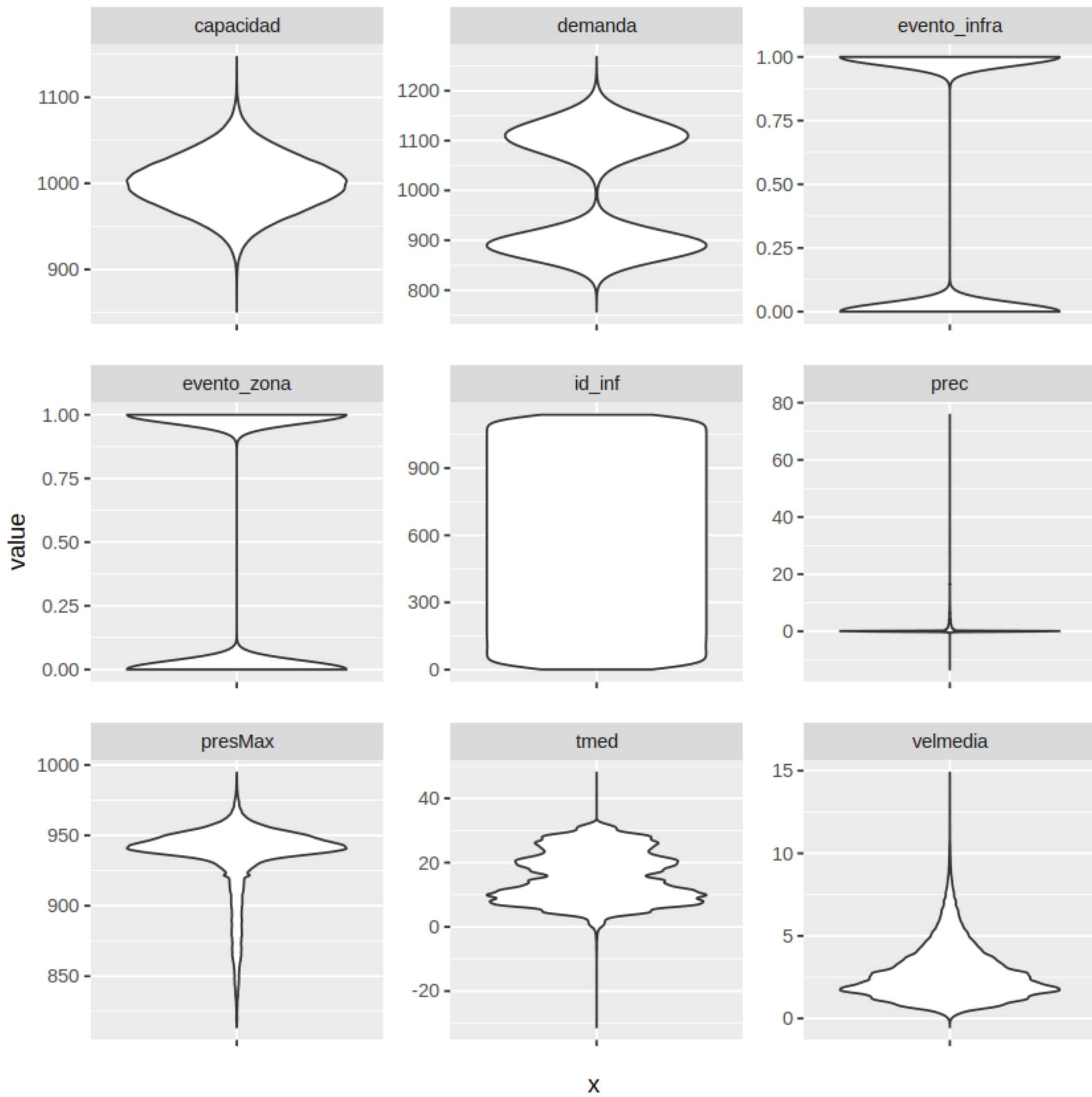
0.10.4 Box plot

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



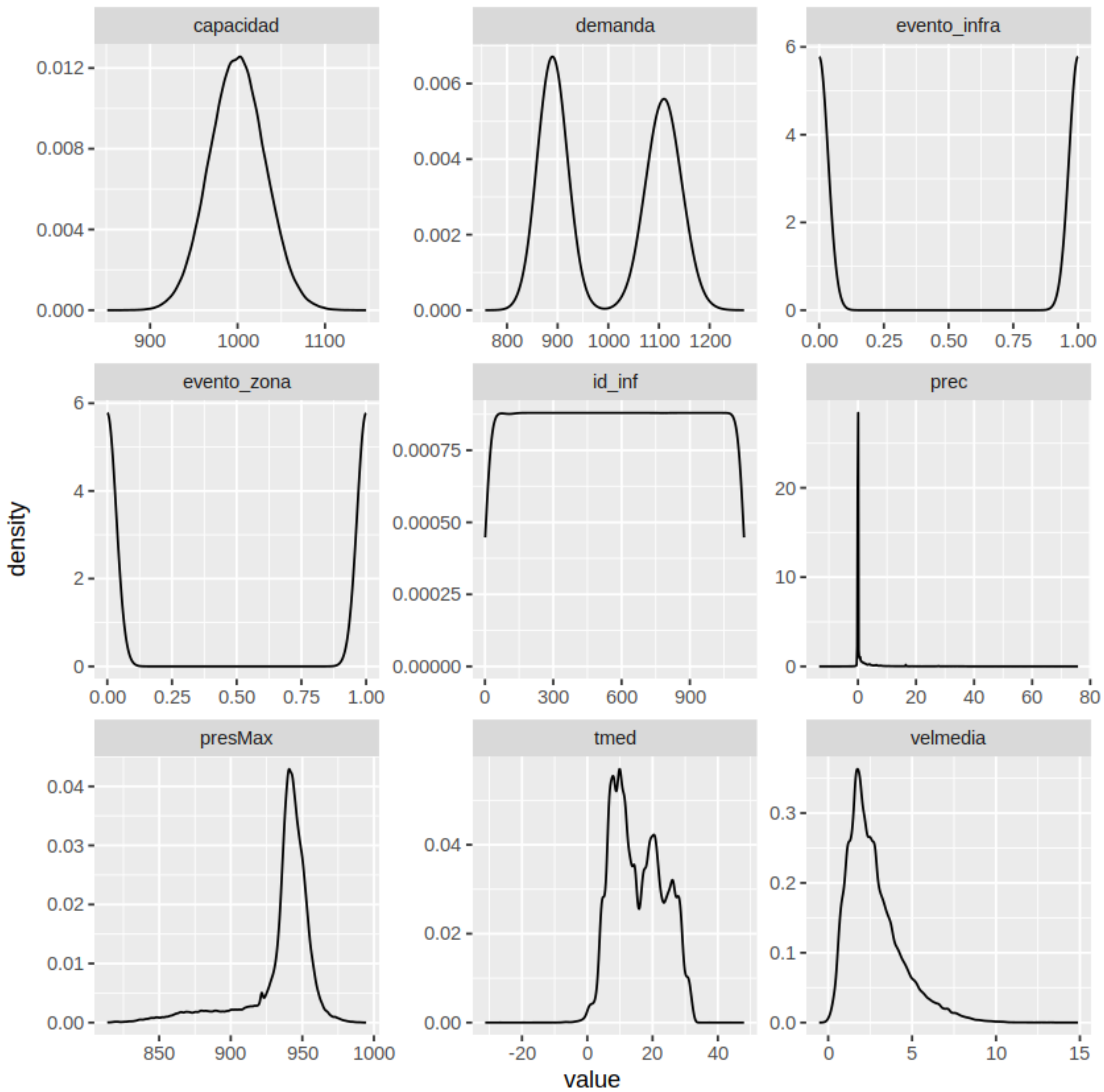
0.10.5 Violin plot

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



0.10.6 Distribution plot

[22]:
```
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

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

'fecha'

[24]:
```
# Category colums
char_data <- data |> select(where(~ !is.numeric(.x)))
char_data
```

A tibble: 377727 × 1

fecha <date>
2019-01-01
2019-01-01
2019-01-01
2019-01-01
2019-01-01
2019-01-01
2019-01-01
2019-01-01
2019-01-01
2019-01-01
2019-01-01
2019-01-01
2019-01-01
2019-01-01
2019-01-01
2019-01-01
2019-01-01
2019-01-01
2019-01-01
2019-01-01
2019-01-01
2019-01-01
2019-01-01
2019-01-01
2019-01-01
2019-01-01
2019-01-01
2019-01-01
2019-01-01
2019-01-01
2019-12-31
2019-12-31
2019-12-31
2019-12-31
2019-12-31
2019-12-31
2019-12-31
2019-12-31
2019-12-31
2019-12-31
2019-12-31
2019-12-31
2019-12-31
2019-12-31
2019-12-31
2019-12-31
2019-12-31
2019-12-31
2019-12-31
2019-12-31

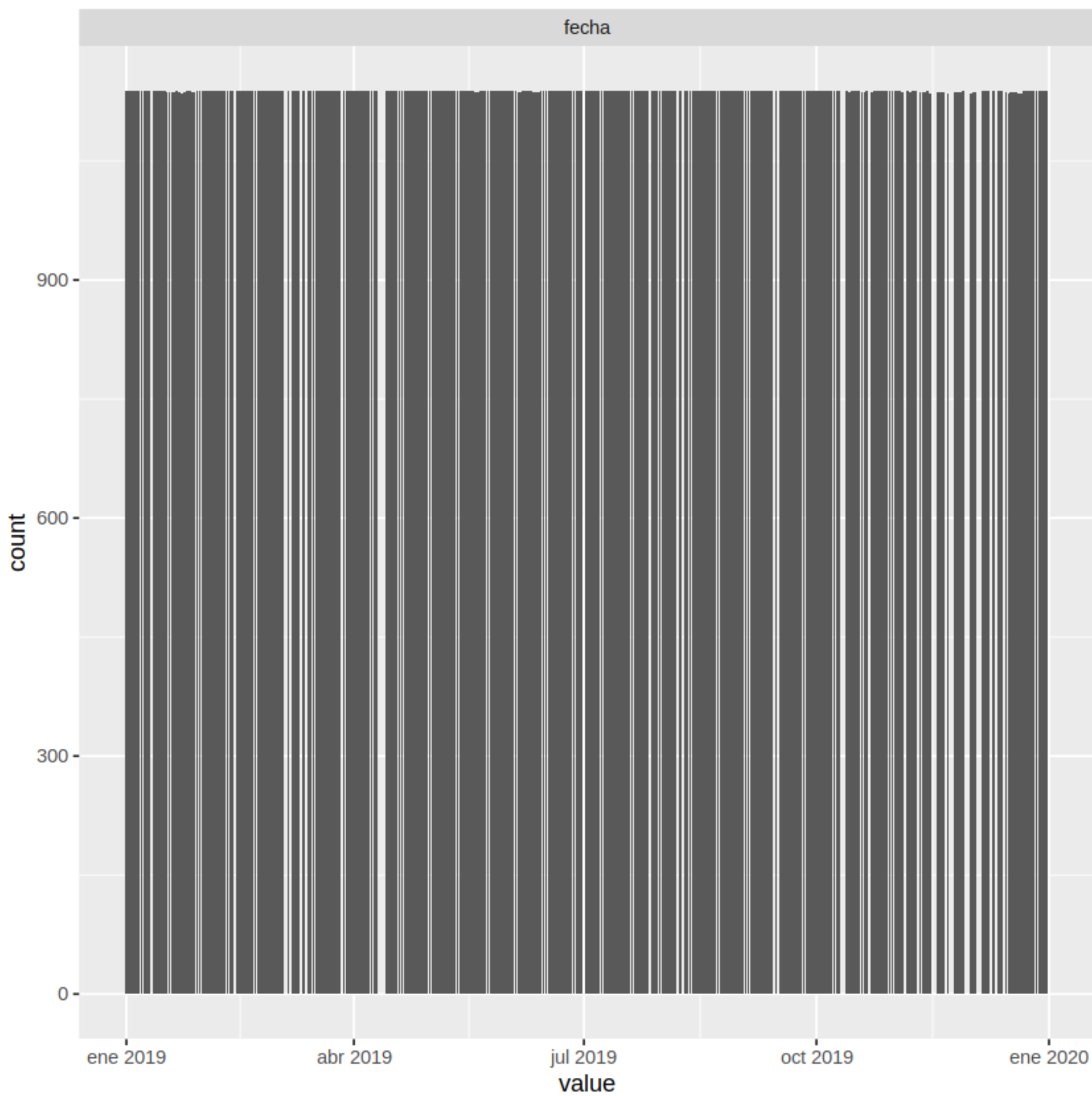
0.11.2 Most frequent entry

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

[25]:
```
# Calculate and visualizate the ratio of the most frequent entry for each␣
↪feature
```

0.11.3 Visualization of categorical variables

[26]:
```
# 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

[27]:
```
cdata_long <- cdata |>
  pivot_longer(cols = everything())
```

0.12.1 Test de Shapiro-Wilk

Si hay muchos datos este no se puede hacer

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

0.12.2 Test de Anderson-Darling

[30]:
```
tapply(cdata_long$value, cdata_long$name, ad.test)
```

```
$capacidad

        Anderson-Darling normality test

data:  X[[i]]
A = 19.676, p-value < 2.2e-16


$demanda

        Anderson-Darling normality test

data:  X[[i]]
A = 23950, p-value < 2.2e-16


$evento_infra

        Anderson-Darling normality test

data:  X[[i]]
A = 67855, p-value < 2.2e-16


$evento_zona

        Anderson-Darling normality test

data:  X[[i]]
```

```
A = 67855, p-value < 2.2e-16


$id_inf

        Anderson-Darling normality test

data:  X[[i]]
A = 4196.1, p-value < 2.2e-16


$prec

        Anderson-Darling normality test

data:  X[[i]]
A = 95062, p-value < 2.2e-16


$presMax

        Anderson-Darling normality test

data:  X[[i]]
A = 30902, p-value < 2.2e-16


$tmed

        Anderson-Darling normality test

data:  X[[i]]
A = 4371.6, p-value < 2.2e-16


$velmedia

        Anderson-Darling normality test

data:  X[[i]]
A = 8649.6, p-value < 2.2e-16
```

0.12.3 Test de Lilliefors

[31]:
```
tapply(cdata_long$value, cdata_long$name, lillie.test)
```

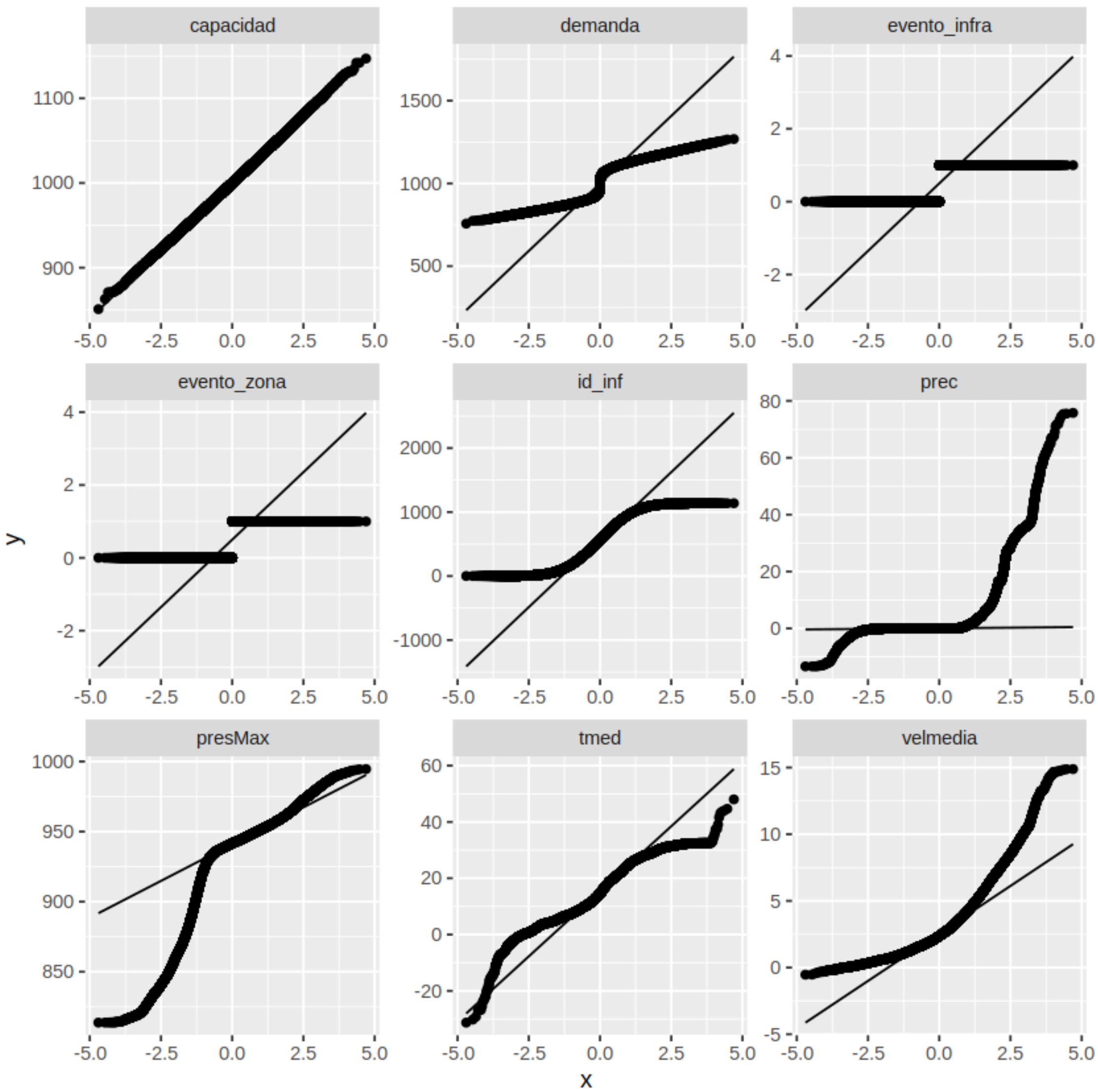
```
$capacidad

	Lilliefors (Kolmogorov-Smirnov) normality test

data:  X[[i]]
D = 0.0084785, p-value < 2.2e-16


$demanda

	Lilliefors (Kolmogorov-Smirnov) normality test

data:  X[[i]]
D = 0.18987, p-value < 2.2e-16


$evento_infra

	Lilliefors (Kolmogorov-Smirnov) normality test

data:  X[[i]]
D = 0.34145, p-value < 2.2e-16


$evento_zona

	Lilliefors (Kolmogorov-Smirnov) normality test

data:  X[[i]]
D = 0.3415, p-value < 2.2e-16


$id_inf

	Lilliefors (Kolmogorov-Smirnov) normality test

data:  X[[i]]
D = 0.05765, p-value < 2.2e-16


$prec

	Lilliefors (Kolmogorov-Smirnov) normality test
```

```
data:  X[[i]]
D = 0.36325, p-value < 2.2e-16


$presMax

        Lilliefors (Kolmogorov-Smirnov) normality test

data:  X[[i]]
D = 0.23022, p-value < 2.2e-16


$tmed

        Lilliefors (Kolmogorov-Smirnov) normality test

data:  X[[i]]
D = 0.084562, p-value < 2.2e-16


$velmedia

        Lilliefors (Kolmogorov-Smirnov) normality test

data:  X[[i]]
D = 0.10176, p-value < 2.2e-16
```

0.12.4 QQ-plots

[32]:

```
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

[33]:
```
cor(cdata, use = "pairwise.complete.obs")
```

A matrix: 9 × 9 of type dbl

	id_inf	capacidad	demanda	evento_infra	even
id_inf	1.000000e+00	0.0014906717	0.0008864349	9.826877e-06	-1.08
capacidad	1.490672e-03	1.0000000000	0.2743227533	-4.058876e-03	2.79
demanda	8.864349e-04	0.2743227533	1.0000000000	5.755948e-01	1.93
evento_infra	9.826877e-06	-0.0040588765	0.5755948394	1.000000e+00	1.21
evento_zona	-1.080539e-03	0.0027927308	0.1938094276	1.215669e-01	1.00
tmed	3.465725e-02	-0.0027198891	0.0006035635	1.815032e-03	3.94
prec	-1.387332e-02	0.0004568889	0.0009019087	7.719077e-04	-1.02
velmedia	-5.604063e-02	-0.0013727245	0.0012846329	1.756743e-03	8.90
presMax	1.335516e-01	0.0001555898	-0.0024259930	-1.983719e-03	1.37

0.14 Regression analysis

0.14.1 Modelo completo regresión lineal simple

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

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

0.14.2 Selección de variables

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

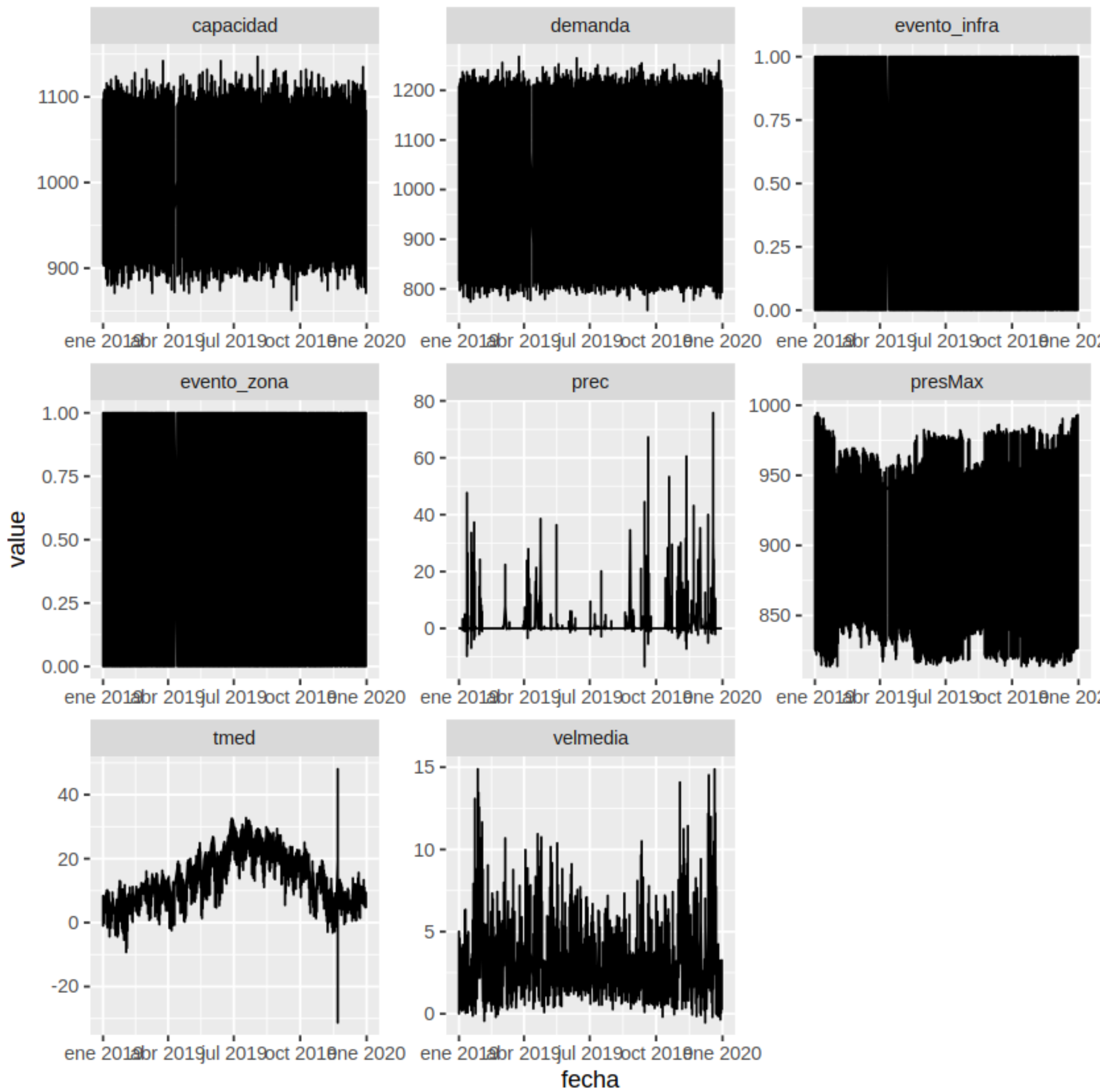
```
[ ]: # modelo2 <- step(modelo, trace = FALSE)
     # summary(modelo2)
```

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

```
[39]: data |>
        pivot_longer(cols = capacidad:presMax) |>
        ggplot(aes(x = fecha, y = value)) +
        geom_line() +
        facet_wrap(~name, scales = "free")
```



0.16 Data Save

- Solo si se han hecho cambios
- No aplica

Identificamos los datos a guardar

[40]:
```
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

[41]:
```
caso <- "CU_18"
proceso <- '_12'
tarea <- "_20"
archivo <- ""
proper <- "_diario_infra"
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

[42]:
```
# 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/.

[43]:
```
# 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

[44]:
```
# 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