14.- Feature Data
Transform_04_06_turismo_origen_completo_v_01

June 11, 2023

#

CU45_Planificación y promoción del destino en base a los patrones en origen de los turistas

Citizenlab Data Science Methodology > III - Feature Engineering Domain *** > # 14.- Feature Data Transform

Feature Data Transform is the process that allows change (if is required) the type and/or distribution of data features (e.g. scaling, normalizing o standardizing data features).

0.1 Tasks

Perform Basic Data Transforms

Perform Categorical Variable Transformation
- Encode Transformation
- One-hot encoding
- Ordinal encoding
- Dummy encoding
- Evaluate a Logistic Regression model
- Consider Embedding if text mining context

Perform Numeric Variable Transformation
- Scale Transformation
- Normalization
- Standardization
- IQR Robust Scaler Transform
- Evaluate a KNN model
- Distribution Transformation
- Discretization
- Uniform
- Clustered(k-Means)
- Quantile
- Normal Quantile
- Uniform Quantile
- Evaluate a KNN model
- Evaluate a KNN model
- Power transforms (Make Distributions More Gaussian)
- Box-Cox Transform

- Yeo-Johnson Transform
- Evaluate a KNN model

0.2 Consideraciones casos CitizenLab programados en R

- Algunas de las tareas de este proceso se han realizado en los notebooks del proceso 05 Data Collection porque eran necesarias para las tareas ETL. En esos casos, en este notebook se referencia al notebook del proceso 05 correspondiente
- Otras tareas típicas de este proceso se realizan en los notebooks del dominio IV al ser más eficiente realizarlas en el propio pipeline de modelización.
- Por tanto en los notebooks de este proceso de manera general se incluyen las comprobaciones necesarias, y comentarios si procede
- Las tareas del proceso se van a aplicar solo a los archivos que forman parte del despliegue, ya que hay muchos archivos intermedios que no procede pasar por este proceso
- El nombre de archivo del notebook hace referencia al nombre de archivo del proceso 05 al que se aplica este proceso, por eso pueden no ser correlativa la numeración
- Las comprobaciones se van a realizar teniendo en cuenta que el lenguaje utilizado en el despliegue de este caso es R

0.3 File

- Input File: CU_45_08_03_turismo_receptor.csv
- Sampled Input File: CU_45_07_03_turismo_receptor.csv
- Output File: No aplica

0.3.1 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")
```

```
Warning message in Sys.setlocale(category = "LC_ALL", locale = "es_ES.UTF-8"):
"OS reports request to set locale to "es_ES.UTF-8" cannot be honored"
```

”

0.4 Settings

0.4.1 Libraries to use

[2]:
```
library(readr)
library(dplyr)
library(tidyr)
library(forcats)
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


Attaching package: 'lubridate'


The following objects are masked from 'package:base':

    date, intersect, setdiff, union
```

0.4.2 Paths

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

0.5 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_45_08_03_turismo_receptor.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_45_08_03_turismo_receptor.csv
```

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

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

```
Rows: 50294 Columns: 9
  Column specification



Delimiter: ","
chr (5): mes, pais_orig_cod, pais_orig, mun_dest, CMUN
dbl (3): mun_dest_cod, turistas, Target
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.
```

Estructura de los datos:

[10]:
```
data |> glimpse()
```

```
Rows: 50,294
Columns: 9
$ mes           <chr> "2019-08", "2021-07", "2021-07",
"2022-01", "2019-08", "…
$ pais_orig_cod <chr> "110", "010", "010", "000", "128",
"000", "011", "126", …
$ pais_orig     <chr> "Francia", "Total Europa", "Total
Europa", "Total", "Rum…
$ mun_dest_cod  <dbl> 28161, 28176, 28132, 28141, 28130,
28126, 28075, 28005, …
$ mun_dest      <chr> "Valdemoro", "Villanueva de la Cañada",
"San Martín de l…
$ turistas      <dbl> 466, 1375, 465, 54, 135, 30, 285, 768,
31, 1646, 116, 36…
$ CMUN          <chr> "161", "176", "132", "141", "130",
"126", "075", "005", …
$ Target        <dbl> 466, 1375, 465, 54, 135, 30, 285, 768,
31, 1646, 116, 36…
$ is_train      <lgl> TRUE, TRUE, TRUE, TRUE, TRUE, TRUE,
TRUE, TRUE, TRUE, TR…
```

Muestra de los primeros datos:

[11]:
```
data |> slice_head(n = 5)
```

	mes <chr>	pais_orig_cod <chr>	pais_orig <chr>	mun_dest_cod <dbl>	mun_dest <chr>	t <
	2019-08	110	Francia	28161	Valdemoro	4
A spec_tbl_df: 5 × 9	2021-07	010	Total Europa	28176	Villanueva de la Cañada	1
	2021-07	010	Total Europa	28132	San Martín de la Vega	4
	2022-01	000	Total	28141	Sevilla la Nueva	5
	2019-08	128	Rumania	28130	San Fernando de Henares	1

0.6 Basic Data Transforms

0.6.1 Data Selecting

[12]:
```
data |> select(1)
```

mes
<chr>
2019-08
2021-07
2021-07
2022-01
2019-08
2022-07
2022-08
2022-03
2020-01
2021-09
2020-05
2020-06
2021-05
2021-02
2020-01
2021-08
2021-07
2019-12
2020-04
2022-10
2021-03
2019-11
2020-08
2021-07
2021-07
2022-04
2019-07
2020-05
2022-03
2019-11

A tibble: 50294 × 1

2022-10
2022-10
2022-10
2022-10
2022-10
2022-10
2022-10
2022-10
2022-10
2022-10
2022-10
2022-10
2022-10
2022-10
2022-10
2022-10
2022-10
2022-10
2022-10
2022-10
2022-10
2022-10

0.6.2 Data Filtering

[15]:
```
data |> filter(pais_orig == "Francia")
```

	mes	pais_orig_cod	pais_orig	mun_dest_cod	mun_dest
	<chr>	<chr>	<chr>	<dbl>	<chr>
	2019-08	110	Francia	28161	Valdemoro
	2021-09	110	Francia	28005	Alcalá de Henares
	2022-10	110	Francia	28066	Griñón
	2022-03	110	Francia	28903	Tres Cantos
	2020-03	110	Francia	28058	Fuenlabrada
	2020-12	110	Francia	28068	Guadarrama
	2020-07	110	Francia	28053	Daganzo de Arriba
	2020-05	110	Francia	28054	Escorial, El
	2022-08	110	Francia	28009	Algete
	2022-01	110	Francia	28030	Cabrera, La
	2021-01	110	Francia	28047	Collado Villalba
	2020-09	110	Francia	28054	Escorial, El
	2022-08	110	Francia	28075	Loeches
	2020-08	110	Francia	28109	Pelayos de la Presa
	2021-07	110	Francia	28058	Fuenlabrada
	2021-04	110	Francia	28075	Loeches
	2020-10	110	Francia	28083	Meco
	2020-08	110	Francia	28901	Lozoyuela-Navas-Sieteiglesias
	2020-04	110	Francia	28065	Getafe
	2021-03	110	Francia	28006	Alcobendas
	2021-03	110	Francia	28127	Rozas de Madrid, Las
	2020-01	110	Francia	28137	Santos de la Humosa, Los
	2021-05	110	Francia	28026	Brunete
	2021-05	110	Francia	28032	Camarma de Esteruelas
	2022-06	110	Francia	28095	Navalagamella
	2022-07	110	Francia	28144	Soto del Real
	2019-07	110	Francia	28073	Humanes de Madrid
	2021-12	110	Francia	28161	Valdemoro
	2022-03	110	Francia	28015	Arroyomolinos
A spec_tbl_df: 3396 × 9	2020-01	110	Francia	28154	Torres de la Alameda
	2022-09	110	Francia	28007	Alcorcón
	2022-09	110	Francia	28009	Algete
	2022-09	110	Francia	28026	Brunete
	2022-09	110	Francia	28027	Buitrago del Lozoya
	2022-09	110	Francia	28046	Collado Mediano
	2022-09	110	Francia	28047	Collado Villalba
	2022-09	110	Francia	28049	Coslada
	2022-09	110	Francia	28053	Daganzo de Arriba
	2022-09	110	Francia	28065	Getafe
	2022-09	110	Francia	28073	Humanes de Madrid
	2022-09	110	Francia	28079	Madrid
	2022-09	110	Francia	28113	Pinto
	2022-09	110	Francia	28116	Pozuelo del Rey
	2022-09	110	Francia	28129	San Agustín del Guadalix
	2022-09	110	Francia	28144	Soto del Real
	2022-09	110	Francia	28154	Torres de la Alameda
	2022-09	110	Francia	28168	Vellón, El
	2022-09	110	Francia	28171	Villa del Prado
	2022-09	110	Francia	28181	Villaviciosa de Odón
	2022-09	110	Francia	28903	Tres Cantos

0.6.3 Insert New Column

[16]:
```
data |>
    mutate(x = TRUE)
```

	mes	pais_orig_cod	pais_orig	mun_dest_cod	mun_dest
	<chr>	<chr>	<chr>	<dbl>	<chr>
	2019-08	110	Francia	28161	Valdemoro
	2021-07	010	Total Europa	28176	Villanueva de la Ca
	2021-07	010	Total Europa	28132	San Martín de la V
	2022-01	000	Total	28141	Sevilla la Nueva
	2019-08	128	Rumania	28130	San Fernando de H
	2022-07	000	Total	28126	Robregordo
	2022-08	011	Total Unión Europea	28075	Loeches
	2022-03	126	Alemania	28005	Alcalá de Henares
	2020-01	121	Países Bajos	28066	Griñón
	2021-09	110	Francia	28005	Alcalá de Henares
	2020-05	121	Países Bajos	28047	Collado Villalba
	2020-06	351	Venezuela	28065	Getafe
	2021-05	010	Total Europa	28100	Nuevo Baztán
	2021-02	123	Portugal	28113	Pinto
	2020-01	010	Total Europa	28005	Alcalá de Henares
	2021-08	131	Suecia	28074	Leganés
	2021-07	000	Total	28034	Canencia
	2019-12	302	EE.UU.	28047	Collado Villalba
	2020-04	213	Egipto	28079	Madrid
	2022-10	110	Francia	28066	Griñón
	2021-03	102	Austria	28007	Alcorcón
	2019-11	126	Alemania	28005	Alcalá de Henares
	2020-08	128	Rumania	28148	Torrejón de Ardoz
	2021-07	117	Luxemburgo	28080	Majadahonda
	2021-07	407	China	28007	Alcorcón
	2022-04	340	Argentina	28134	San Sebastián de lo
	2019-07	228	Marruecos	28005	Alcalá de Henares
	2020-05	117	Luxemburgo	28080	Majadahonda
	2022-03	110	Francia	28903	Tres Cantos
A tibble: 50294 × 10	2019-11	343	Colombia	28130	San Fernando de H
	2022-10	000	Total	28162	Valdeolmos-Alalpar
	2022-10	010	Total Europa	28162	Valdeolmos-Alalpar
	2022-10	000	Total	28164	Valdetorres de Jara
	2022-10	011	Total Unión Europea	28164	Valdetorres de Jara
	2022-10	000	Total	28167	Velilla de San Anto
	2022-10	000	Total	28168	Vellón, El
	2022-10	126	Alemania	28168	Vellón, El
	2022-10	000	Total	28169	Venturada
	2022-10	011	Total Unión Europea	28169	Venturada
	2022-10	000	Total	28171	Villa del Prado
	2022-10	011	Total Unión Europea	28171	Villa del Prado
	2022-10	121	Países Bajos	28171	Villa del Prado
	2022-10	010	Total Europa	28172	Villalbilla
	2022-10	000	Total	28174	Villamanta
	2022-10	011	Total Unión Europea	28174	Villamanta
	2022-10	011	Total Unión Europea	28176	Villanueva de la Ca
	2022-10	020	Total África	28176	Villanueva de la Ca
	2022-10	033	Total Sudamérica	28176	Villanueva de la Ca
	2022-10	102	Austria	28176	Villanueva de la Ca
	2022-10	115	Italia	28176	Villanueva de la Ca

0.6.4 Delete Column

[17]:
```
col <- "pais_orig"
```

[18]:
```
data %>% select(all_of(col))
```

pais_orig <chr>
Francia
Total Europa
Total Europa
Total
Rumania
Total
Total Unión Europea
Alemania
Países Bajos
Francia
Países Bajos
Venezuela
Total Europa
Portugal
Total Europa
Suecia
Total
EE.UU.
Egipto
Francia
Austria
Alemania
Rumania
Luxemburgo
China
Argentina
Marruecos
Luxemburgo
Francia
Colombia
Total
Total Europa
Total
Total Unión Europea
Total
Total
Alemania
Total
Total Unión Europea
Total
Total Unión Europea
Países Bajos
Total Europa
Total
Total Unión Europea
Total Unión Europea
Total África
Total Sudamérica
Austria
Italia

A tibble: 50294 × 1

0.6.5 Rank Data

Operation

[19]:
```
data |> mutate(rank = order(Target))
```

	mes	pais_orig_cod	pais_orig	mun_dest_cod	mun_dest
	<chr>	<chr>	<chr>	<dbl>	<chr>
	2019-08	110	Francia	28161	Valdemoro
	2021-07	010	Total Europa	28176	Villanueva de la Ca
	2021-07	010	Total Europa	28132	San Martín de la V
	2022-01	000	Total	28141	Sevilla la Nueva
	2019-08	128	Rumania	28130	San Fernando de H
	2022-07	000	Total	28126	Robregordo
	2022-08	011	Total Unión Europea	28075	Loeches
	2022-03	126	Alemania	28005	Alcalá de Henares
	2020-01	121	Países Bajos	28066	Griñón
	2021-09	110	Francia	28005	Alcalá de Henares
	2020-05	121	Países Bajos	28047	Collado Villalba
	2020-06	351	Venezuela	28065	Getafe
	2021-05	010	Total Europa	28100	Nuevo Baztán
	2021-02	123	Portugal	28113	Pinto
	2020-01	010	Total Europa	28005	Alcalá de Henares
	2021-08	131	Suecia	28074	Leganés
	2021-07	000	Total	28034	Canencia
	2019-12	302	EE.UU.	28047	Collado Villalba
	2020-04	213	Egipto	28079	Madrid
	2022-10	110	Francia	28066	Griñón
	2021-03	102	Austria	28007	Alcorcón
	2019-11	126	Alemania	28005	Alcalá de Henares
	2020-08	128	Rumania	28148	Torrejón de Ardoz
	2021-07	117	Luxemburgo	28080	Majadahonda
	2021-07	407	China	28007	Alcorcón
	2022-04	340	Argentina	28134	San Sebastián de lo
	2019-07	228	Marruecos	28005	Alcalá de Henares
	2020-05	117	Luxemburgo	28080	Majadahonda
	2022-03	110	Francia	28903	Tres Cantos
A tibble: 50294 × 10	2019-11	343	Colombia	28130	San Fernando de H
	2022-10	000	Total	28162	Valdeolmos-Alalpar
	2022-10	010	Total Europa	28162	Valdeolmos-Alalpar
	2022-10	000	Total	28164	Valdetorres de Jara
	2022-10	011	Total Unión Europea	28164	Valdetorres de Jara
	2022-10	000	Total	28167	Velilla de San Anto
	2022-10	000	Total	28168	Vellón, El
	2022-10	126	Alemania	28168	Vellón, El
	2022-10	000	Total	28169	Venturada
	2022-10	011	Total Unión Europea	28169	Venturada
	2022-10	000	Total	28171	Villa del Prado
	2022-10	011	Total Unión Europea	28171	Villa del Prado
	2022-10	121	Países Bajos	28171	Villa del Prado
	2022-10	010	Total Europa	28172	Villalbilla
	2022-10	000	Total	28174	Villamanta
	2022-10	011	Total Unión Europea	28174	Villamanta
	2022-10	011	Total Unión Europea	28176	Villanueva de la Ca
	2022-10	020	Total África	28176	Villanueva de la Ca
	2022-10	033	Total Sudamérica	28176	Villanueva de la Ca
	2022-10	102	Austria	28176	Villanueva de la Ca
	2022-10	115	Italia	28176	Villanueva de la Ca

0.7 Numeric Variable Transformation: Scale

0.7.1 Normalization Transform

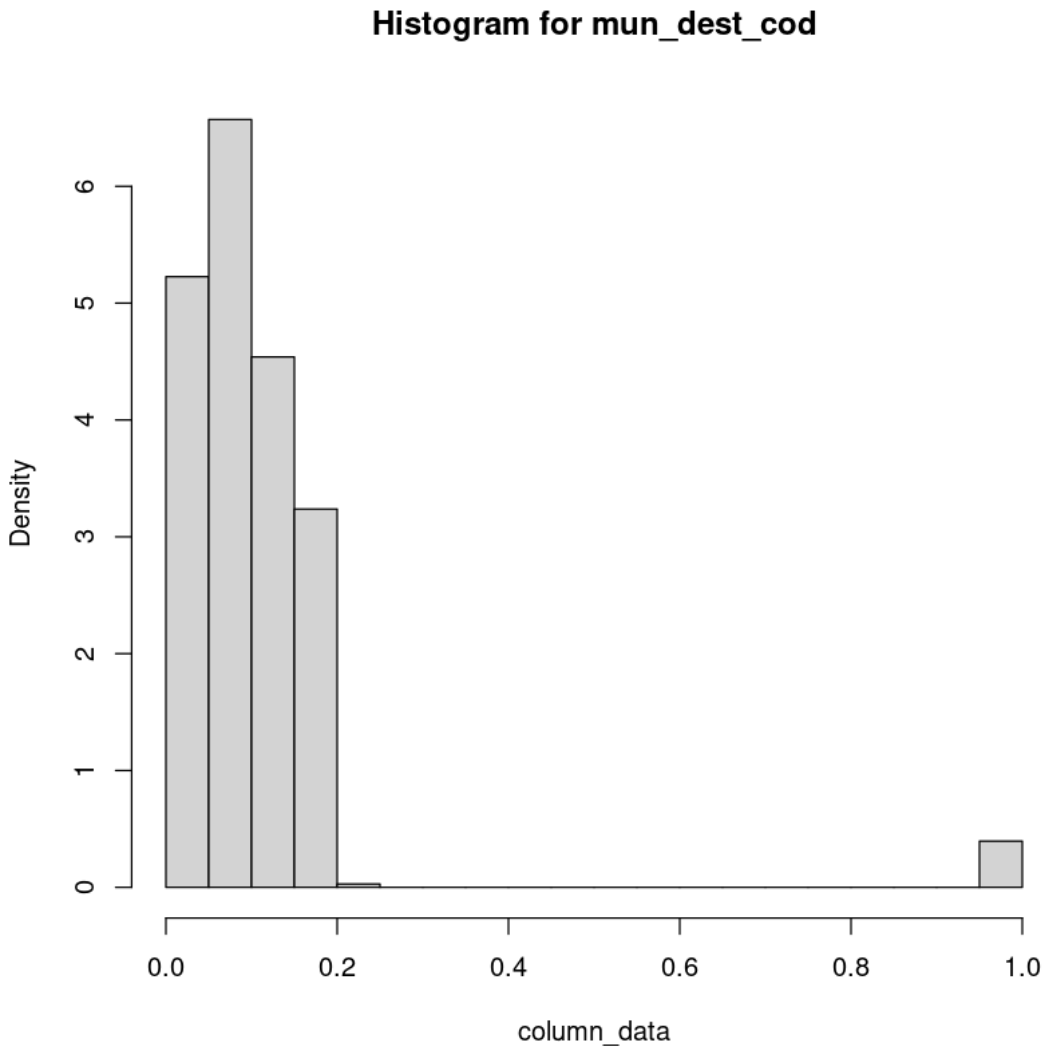
Select columns

[25]:
```
numeric_cols <- sapply(data, is.numeric)
```

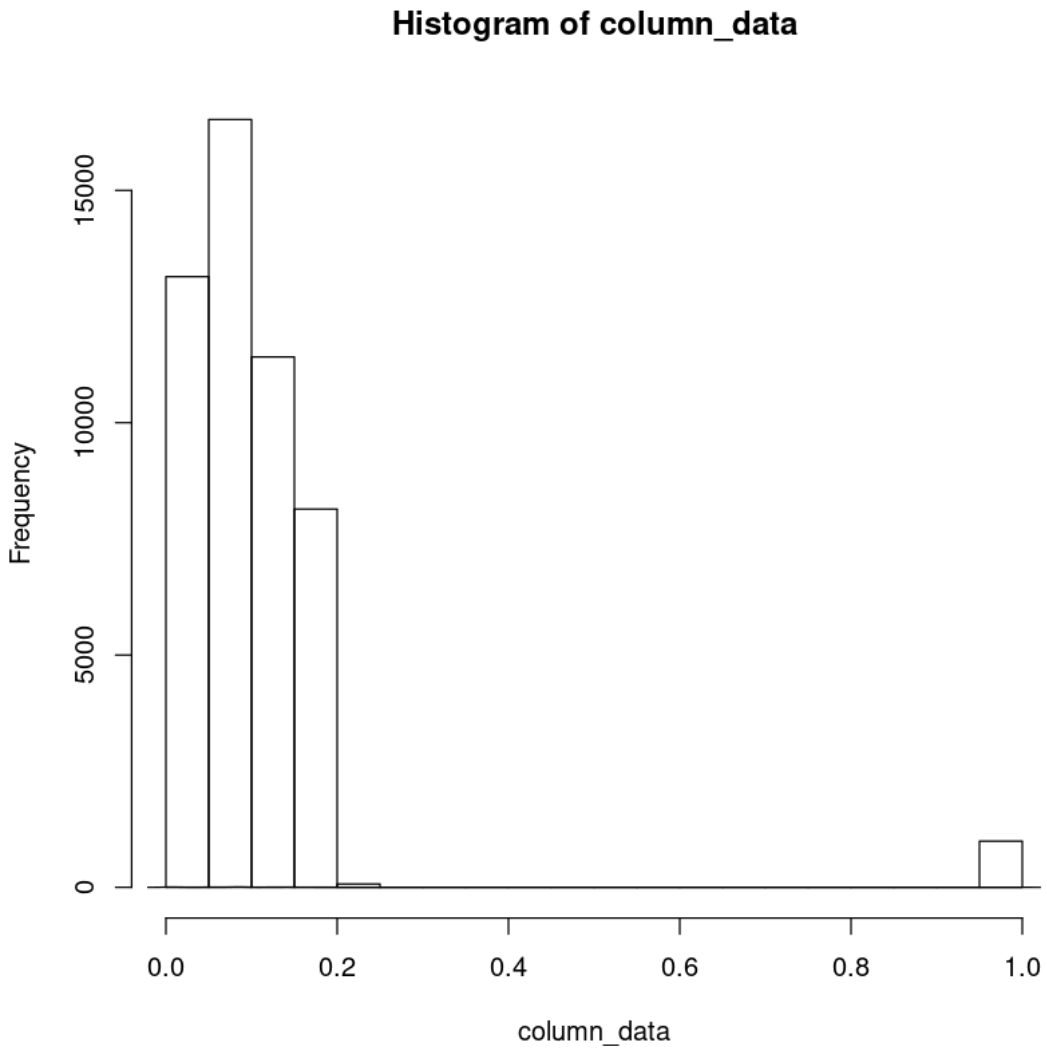
Operation

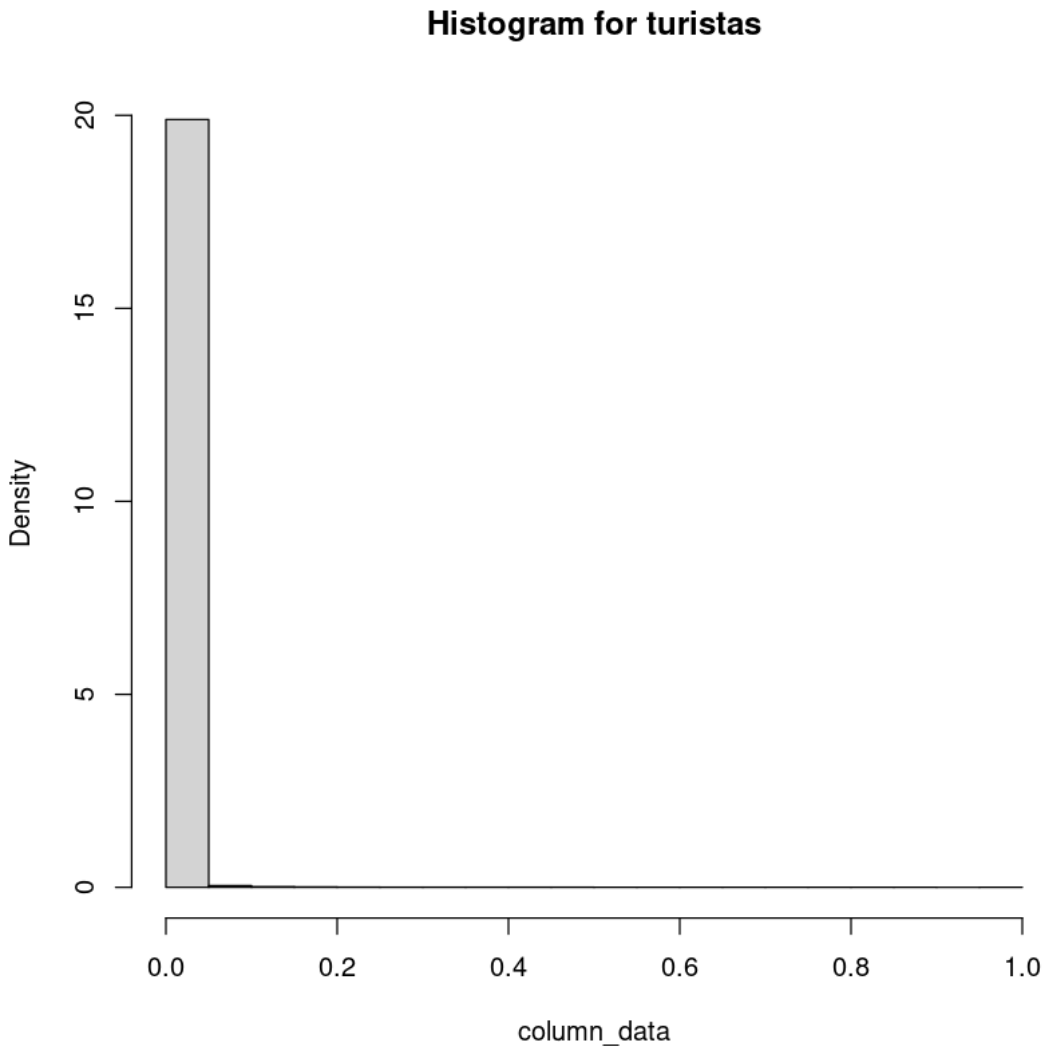
[26]:
```
data_normalized <- data
for (col_name in names(numeric_cols)[numeric_cols]) {
  data_normalized[[col_name]] <- (data[[col_name]] - min(data[[col_name]])) /
                                 (max(data[[col_name]]) - min(data[[col_name]]))
}

for (col_name in names(cols)[cols]) {
    cat("Processing column:", col_name, "\n")
    column_data <- data_normalized[[col_name]]
    hist_plot <- hist(column_data, freq = FALSE, main = paste("Histogram for",␣
 ↪col_name))
    print(plot(hist_plot))
    lines(density(column_data))
}
```

```
Processing column: mun_dest_cod
```

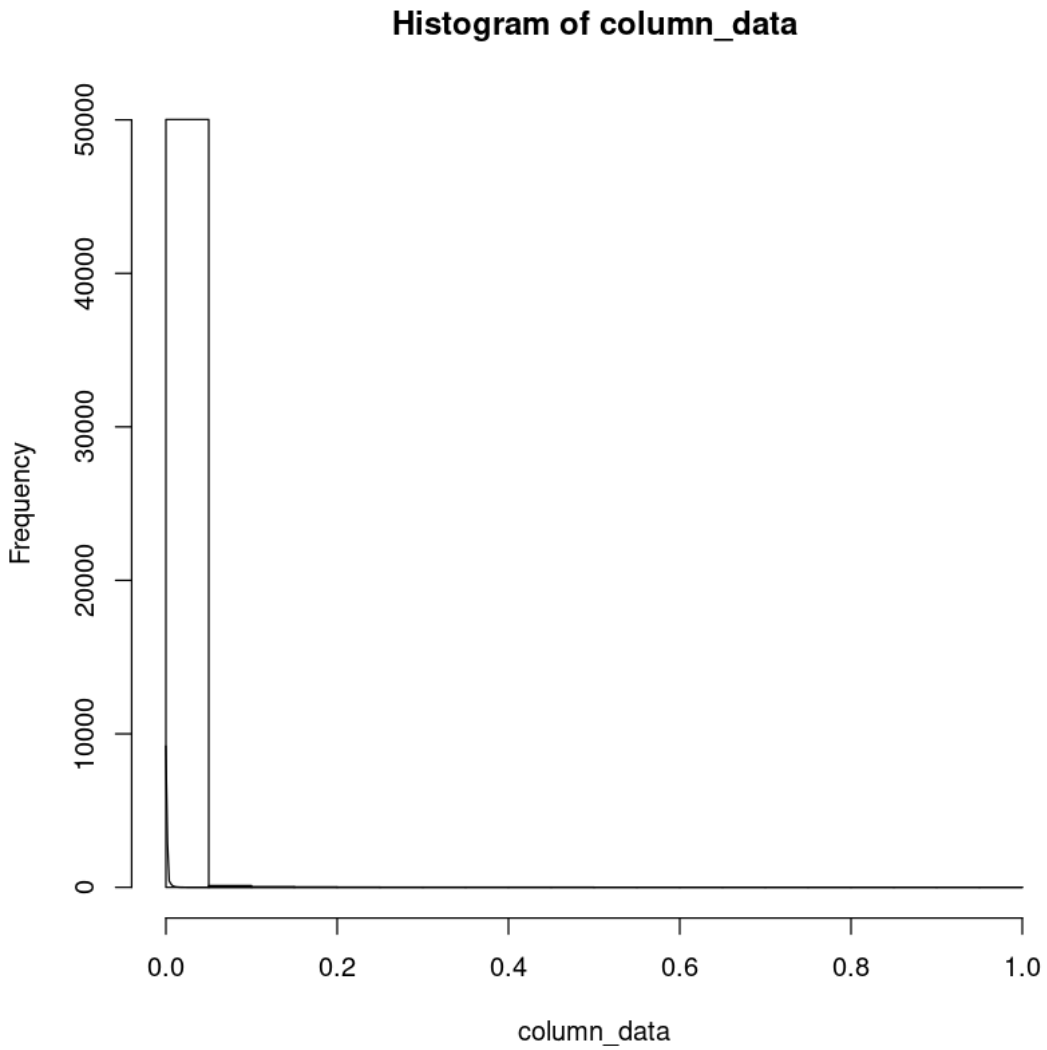


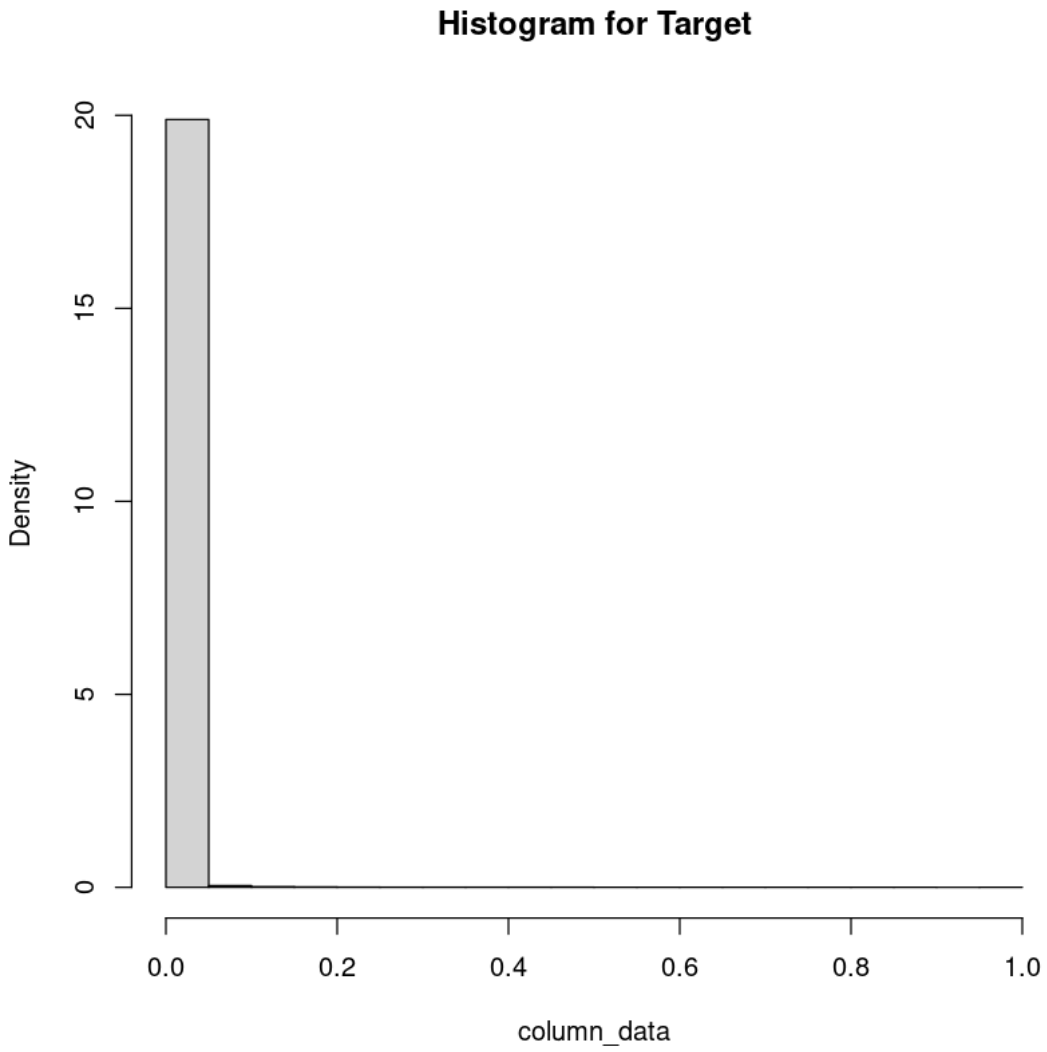

```
NULL
Processing column: turistas
```


Histogram of column_data
Frequency
0
5000
10000
15000
0.0
0.2
0.4
0.6
0.8
1.0
column_data

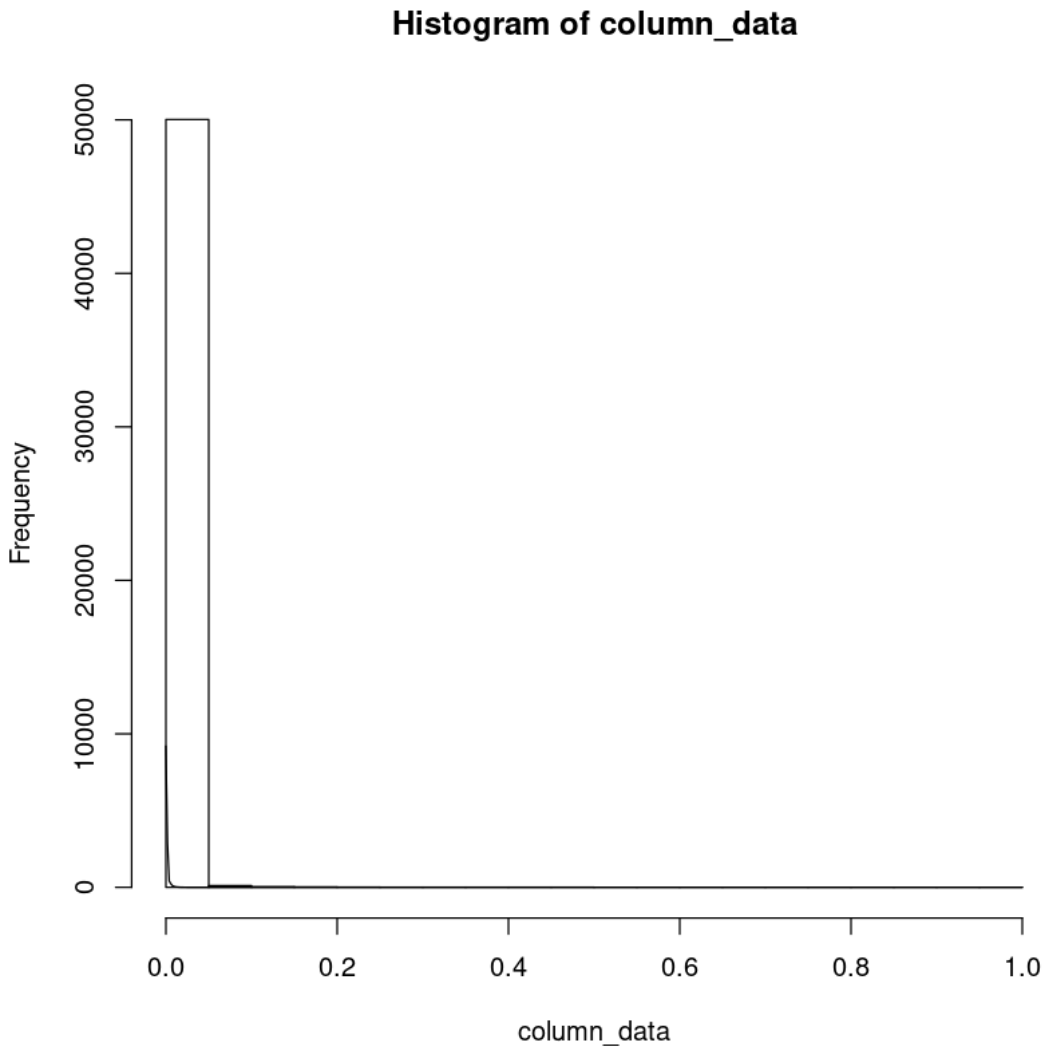



```
NULL
Processing column: Target
```


Histogram of column_data
Frequency
0
10000
20000
30000
40000
50000
0.0
0.2
0.4
0.6
0.8
1.0
column_data


Histogram for Target
Density
0
5
10
15
20
0.0
0.2
0.4
0.6
0.8
1.0
column_data

```
NULL
```



[ ]:

0.7.2 Standarization Transform

Select columns

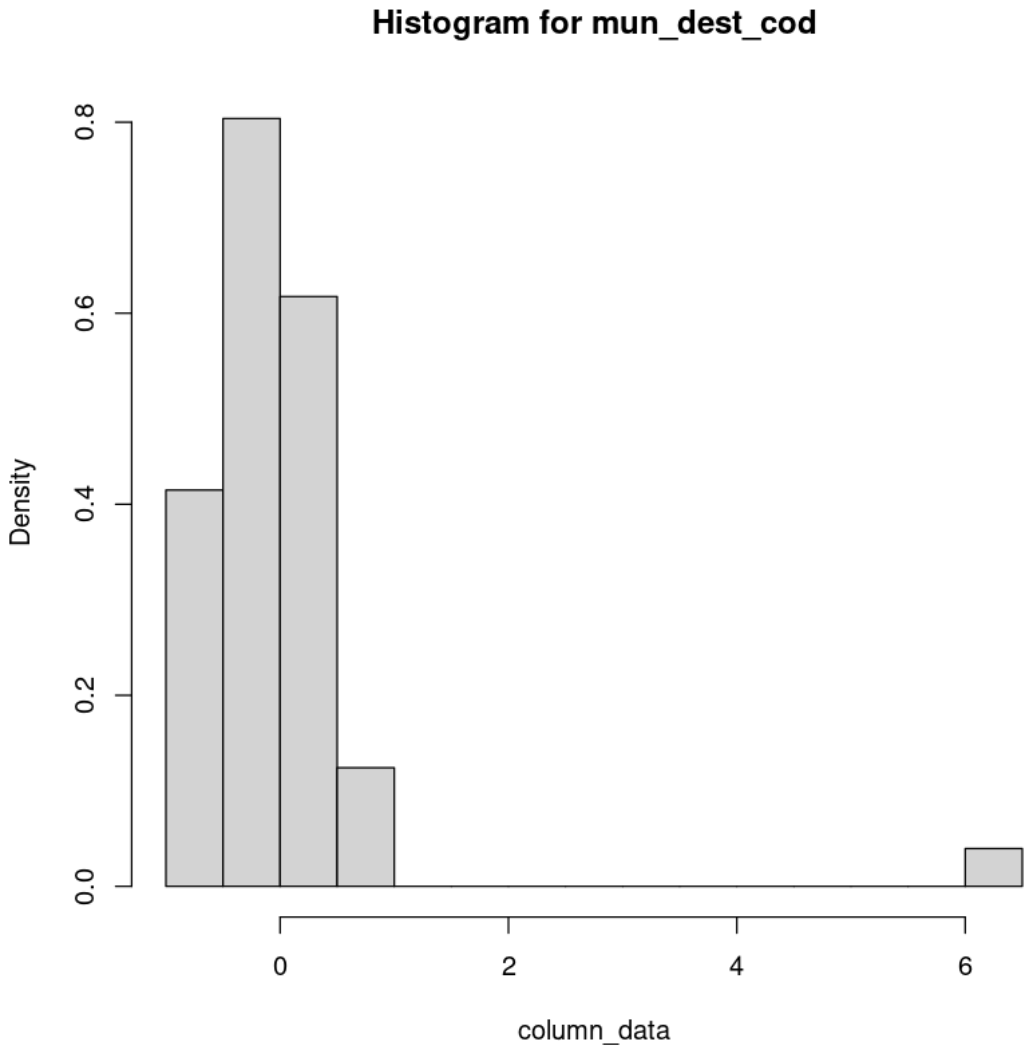
[27]:
```
cols <- sapply(data, is.numeric)
```

Operation

[28]:
```
data_standardized <- data

for (col_name in names(numeric_cols)[numeric_cols]) {
  data_standardized[[col_name]] <- scale(data[[col_name]])
```

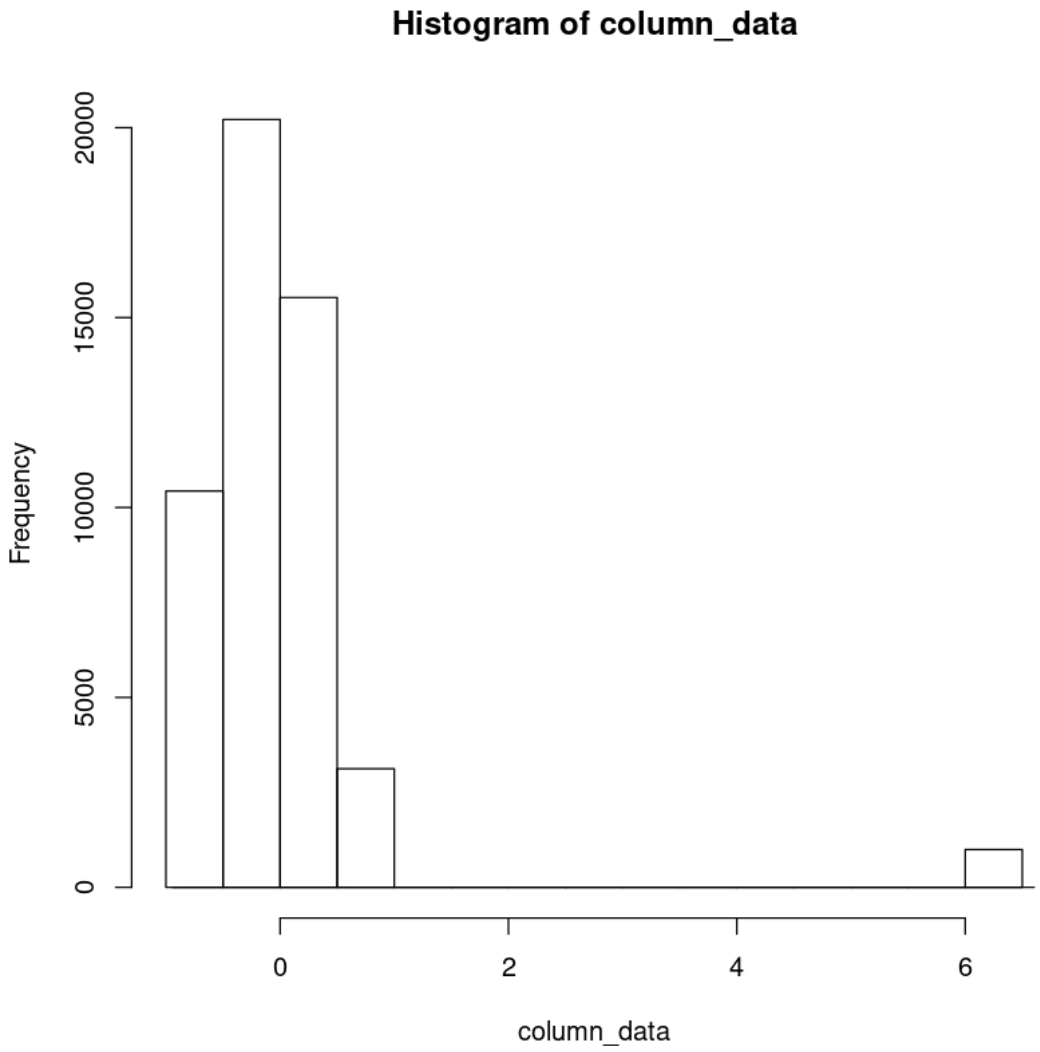
```
}
for (col_name in names(cols)[cols]) {
    cat("Processing column:", col_name, "\n")
    column_data <- data_standardized[[col_name]]
    hist_plot <- hist(column_data, freq = FALSE, main = paste("Histogram for",␣
 ↪col_name))
    print(plot(hist_plot))
    lines(density(column_data))
}
```

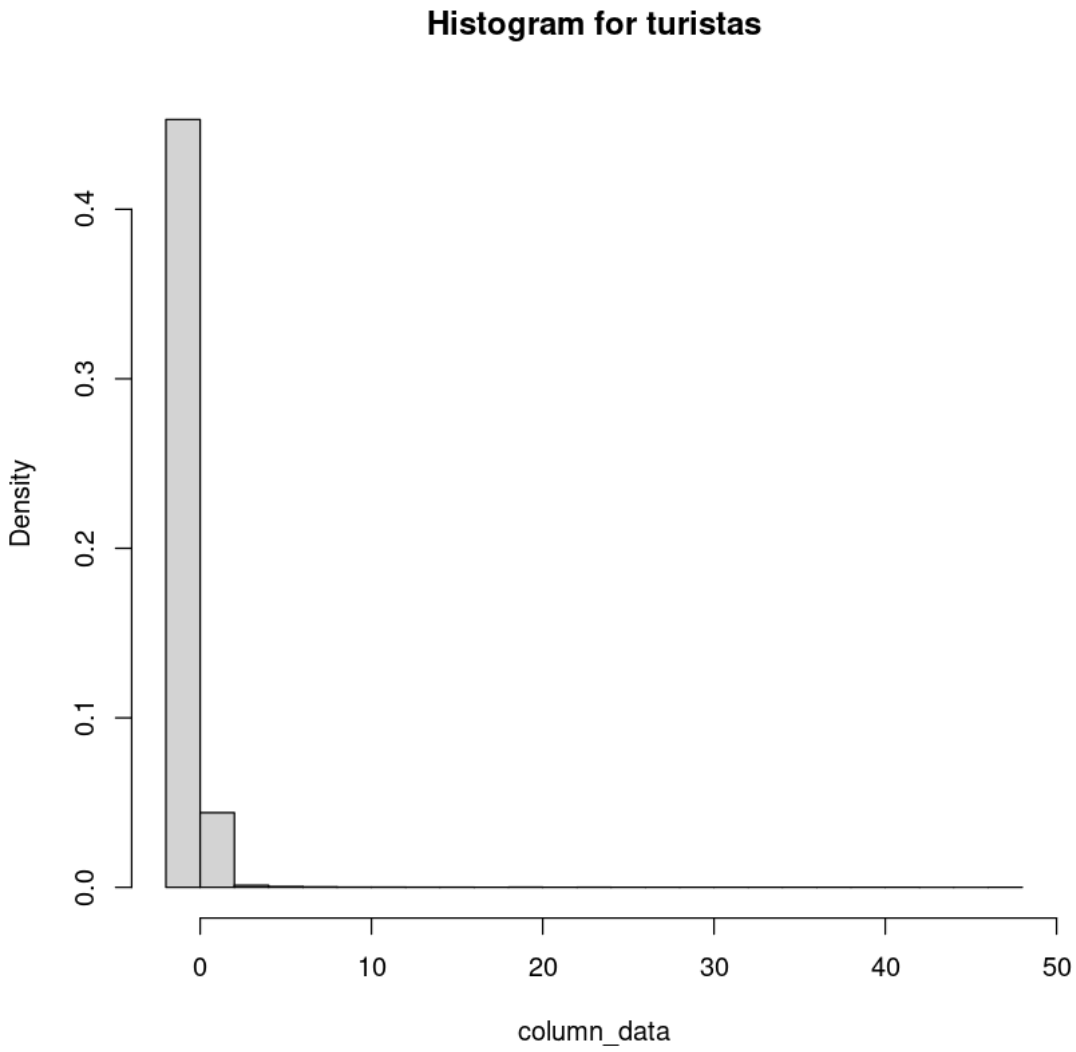
```
Processing column: mun_dest_cod
```



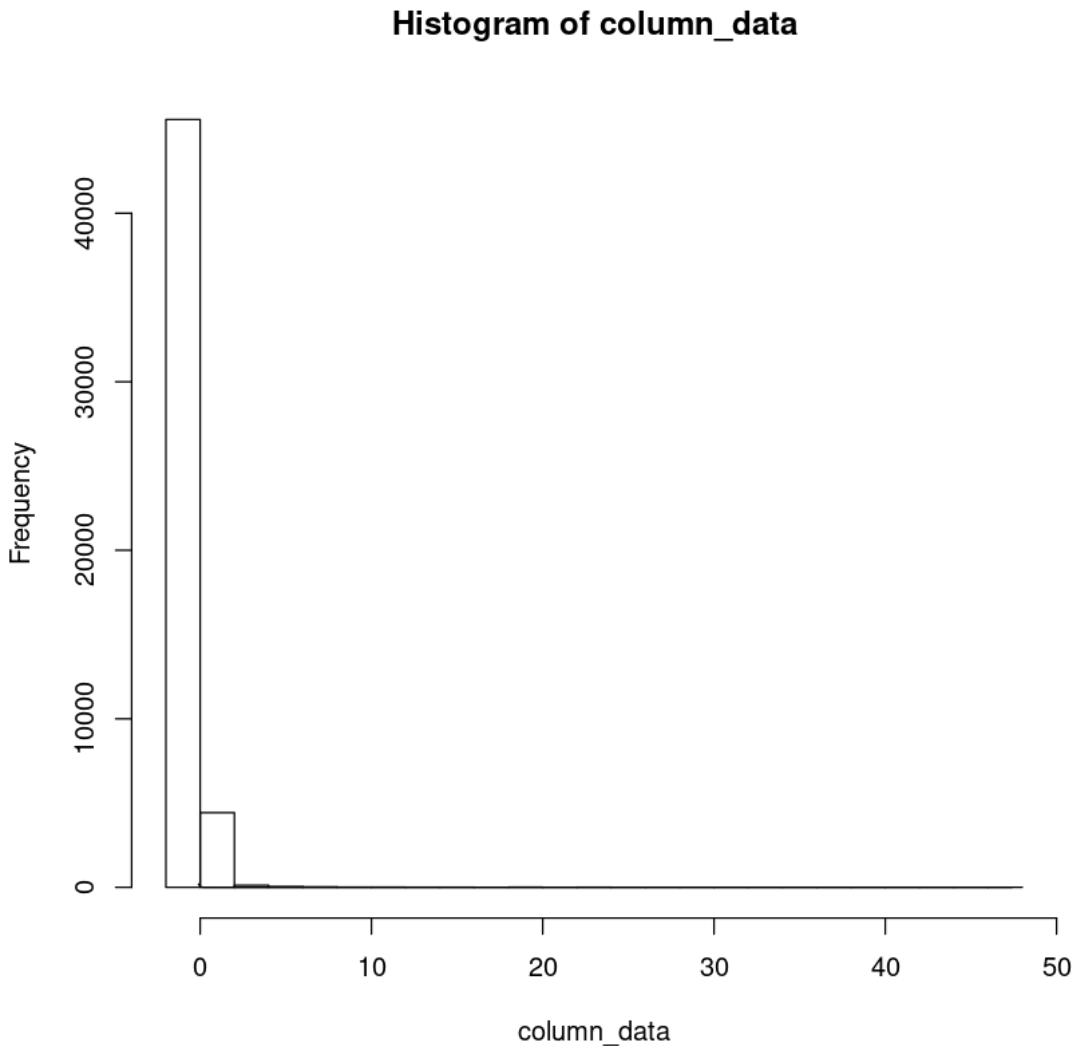

```
NULL
```

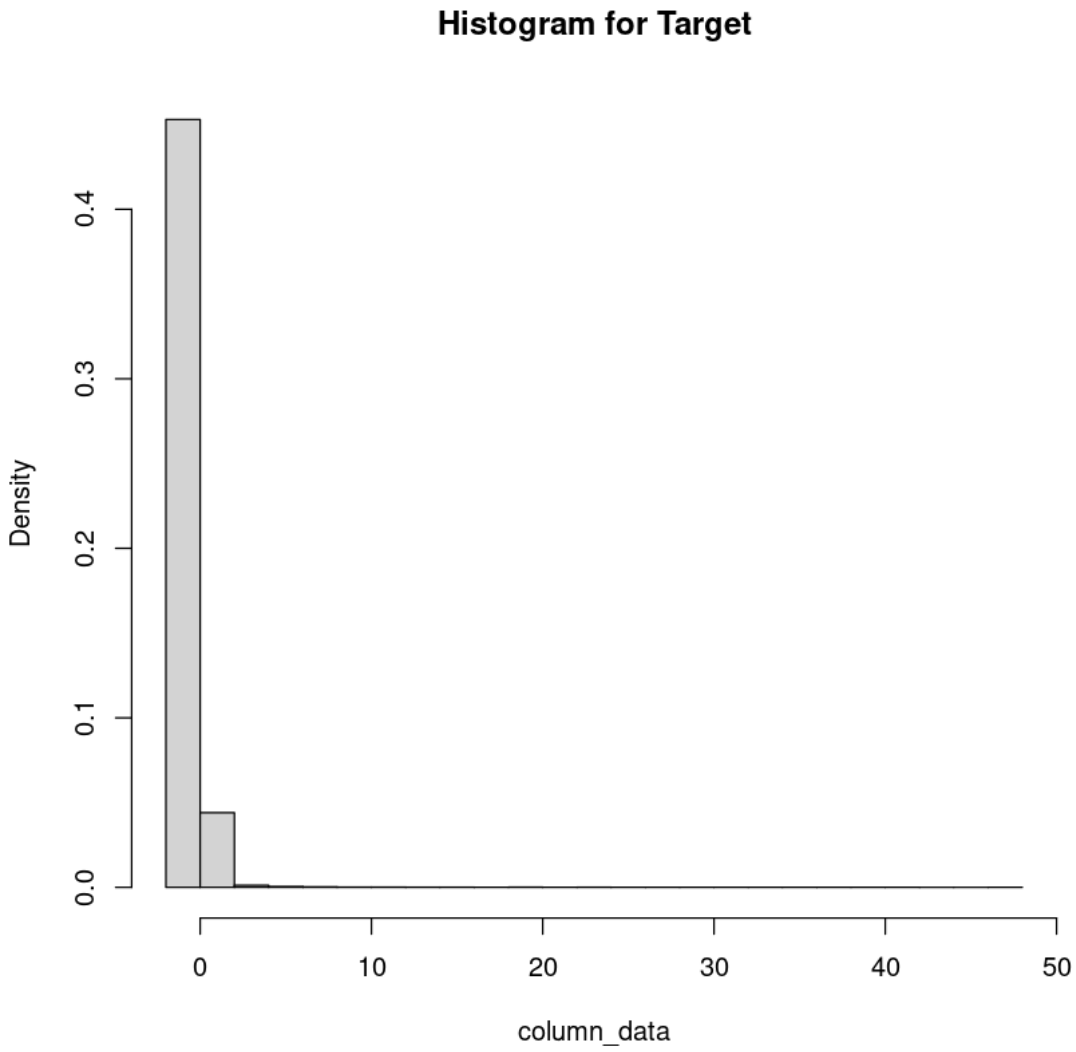
```
Processing column: turistas
```



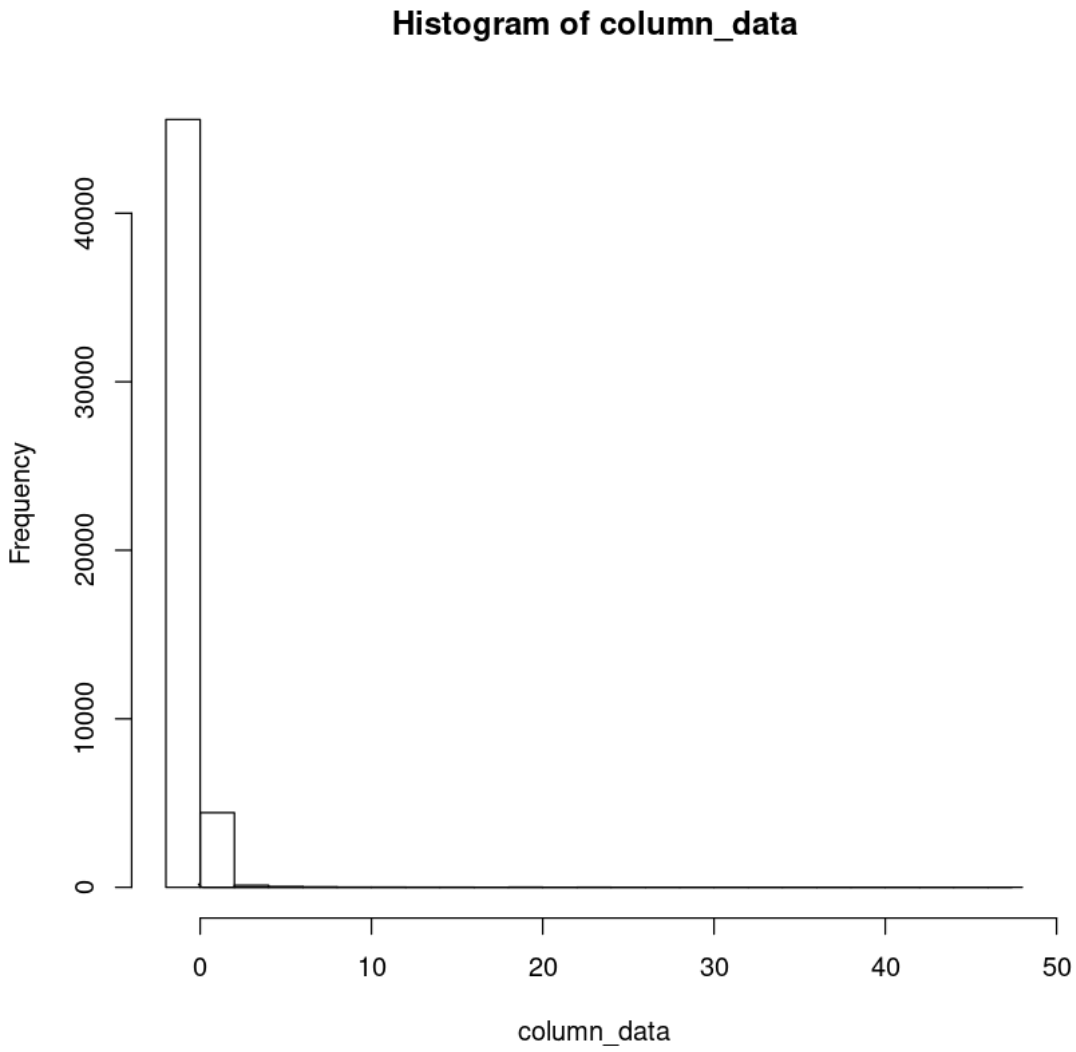



```
NULL
Processing column: Target
```


Histogram of column_data
Frequency
0
10000
20000
30000
40000
0
10
20
30
40
50
column_data


Histogram for Target
Density
0.0
0.1
0.2
0.3
0.4
0
10
20
30
40
50
column_data

```
NULL
```



0.8 Numeric Variable Transformation: Distribution

0.8.1 Discretization Transform

Evaluating Discretization Transformations

Uniform Discretization Transform Select columns

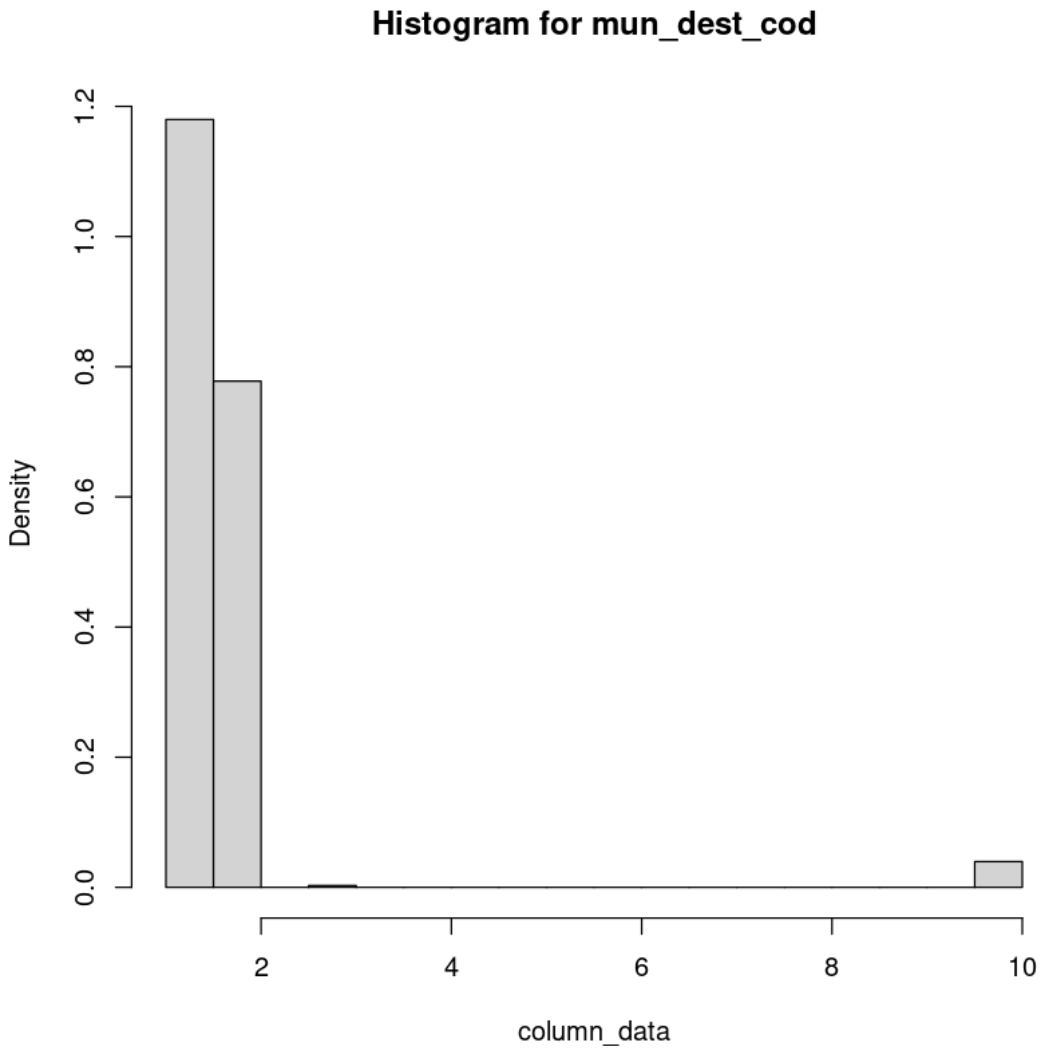
[29]:
```
cols <- sapply(data, is.numeric)
```

[ ]:
```

```

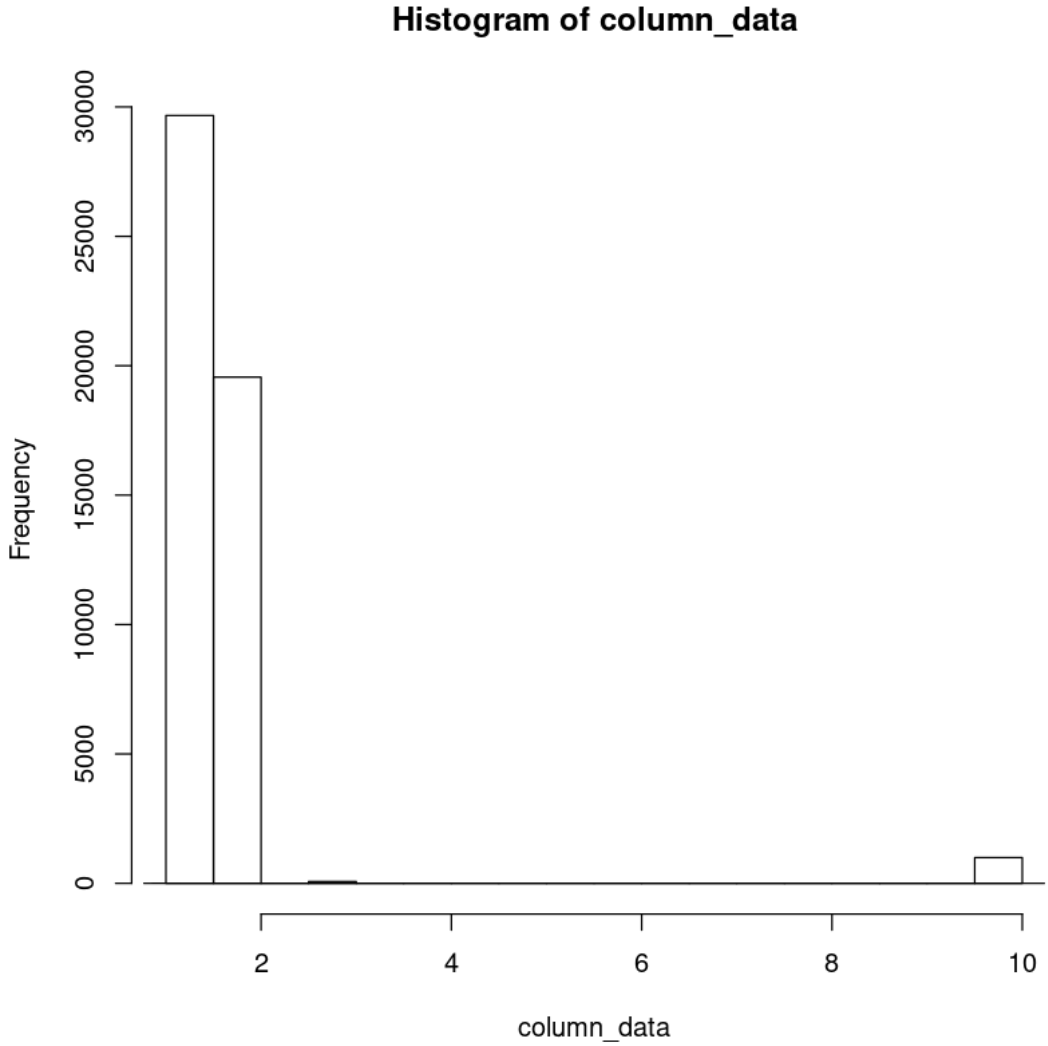
Operation

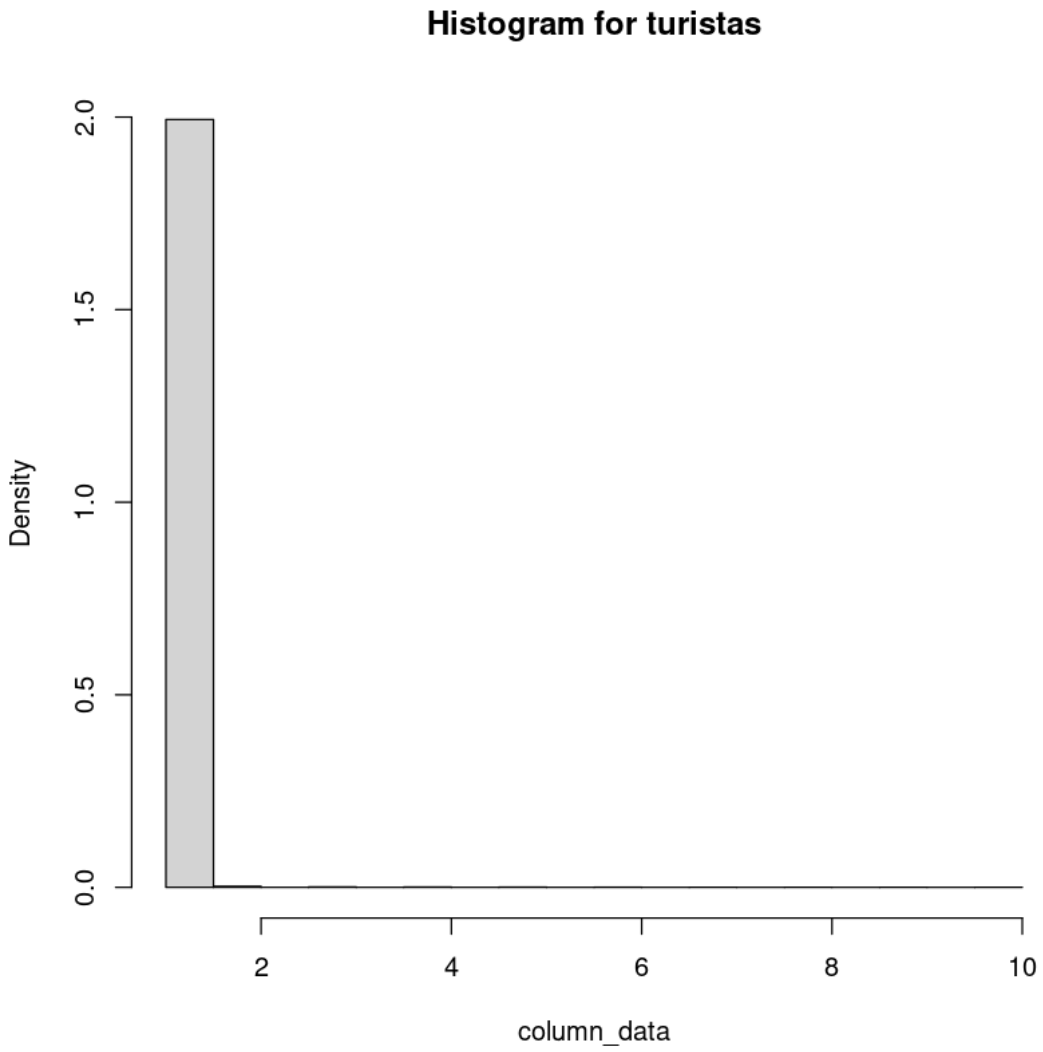
[30]:
```r
# create a copy of the original data frame
data_discretized <- data

# number of intervals
k <- 10  # change this value according to your needs

# discretize each numeric column
for (col_name in names(numeric_cols)[numeric_cols]) {
  data_discretized[[col_name]] <- cut(data[[col_name]], breaks = k, labels =␣
 ↪FALSE)
}
for (col_name in names(cols)[cols]) {
    cat("Processing column:", col_name, "\n")
    column_data <- data_discretized[[col_name]]
    hist_plot <- hist(column_data, freq = FALSE, main = paste("Histogram for",␣
 ↪col_name))
    print(plot(hist_plot))
    lines(density(column_data))
}
```

```
Processing column: mun_dest_cod
```

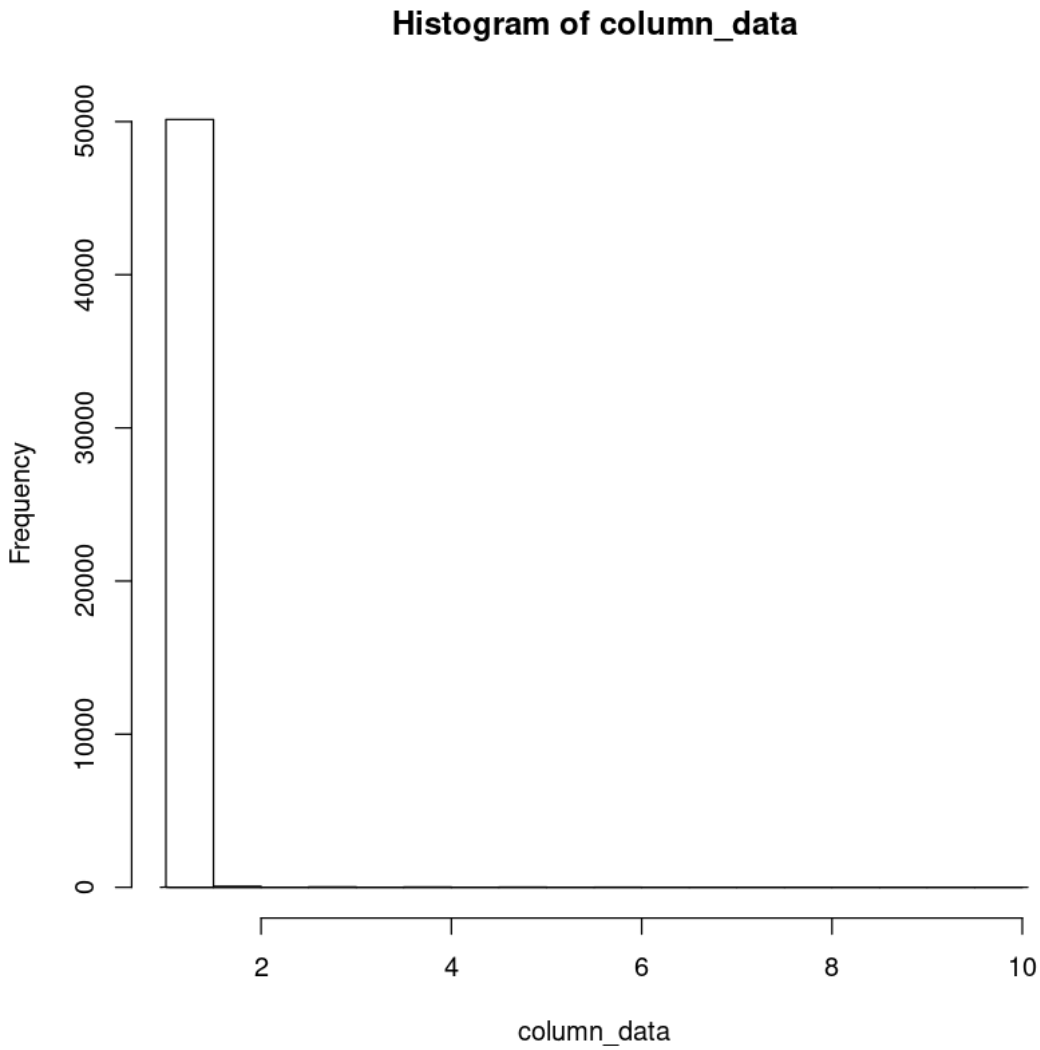


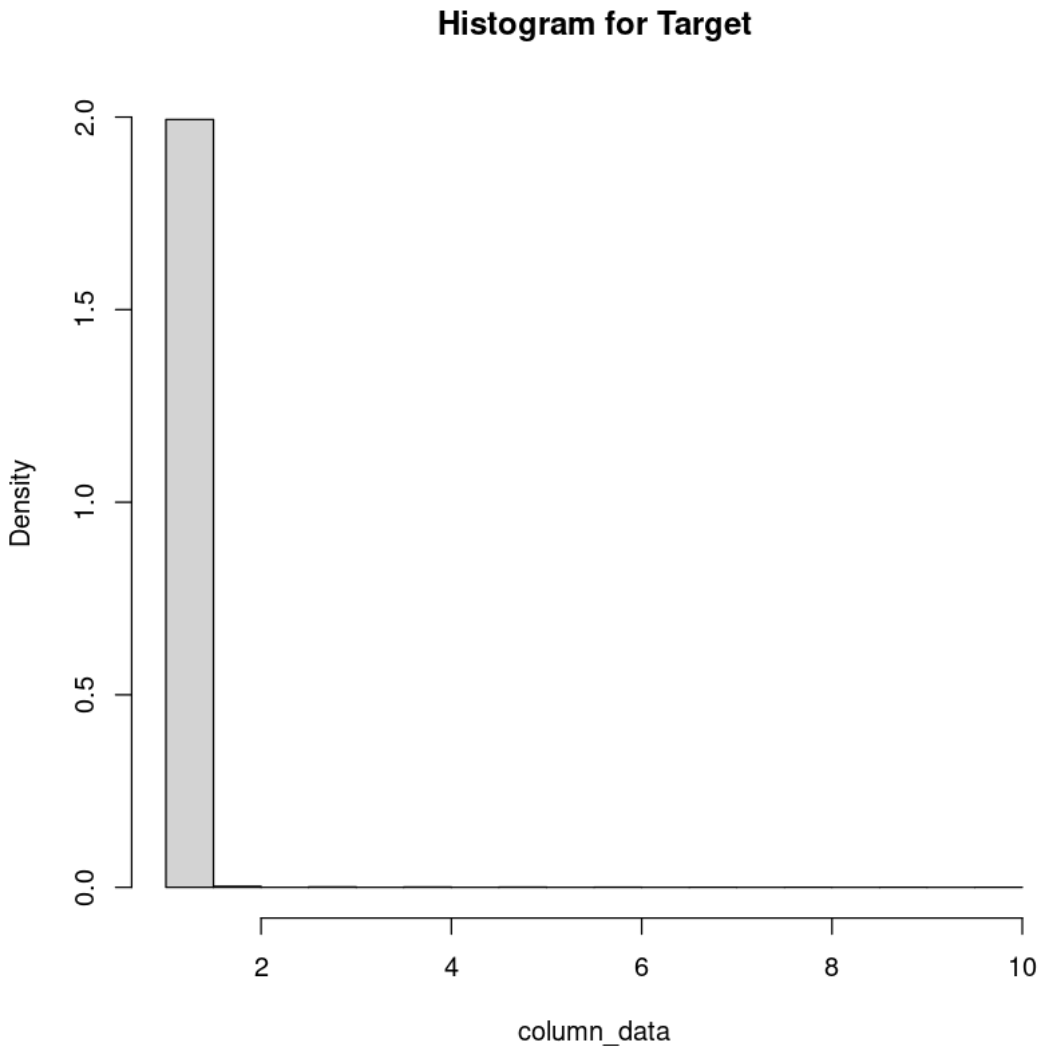

```
NULL
Processing column: turistas
```


Histogram of column_data
Frequency
0
5000
10000
15000
20000
25000
30000
2
4
6
8
10
column_data

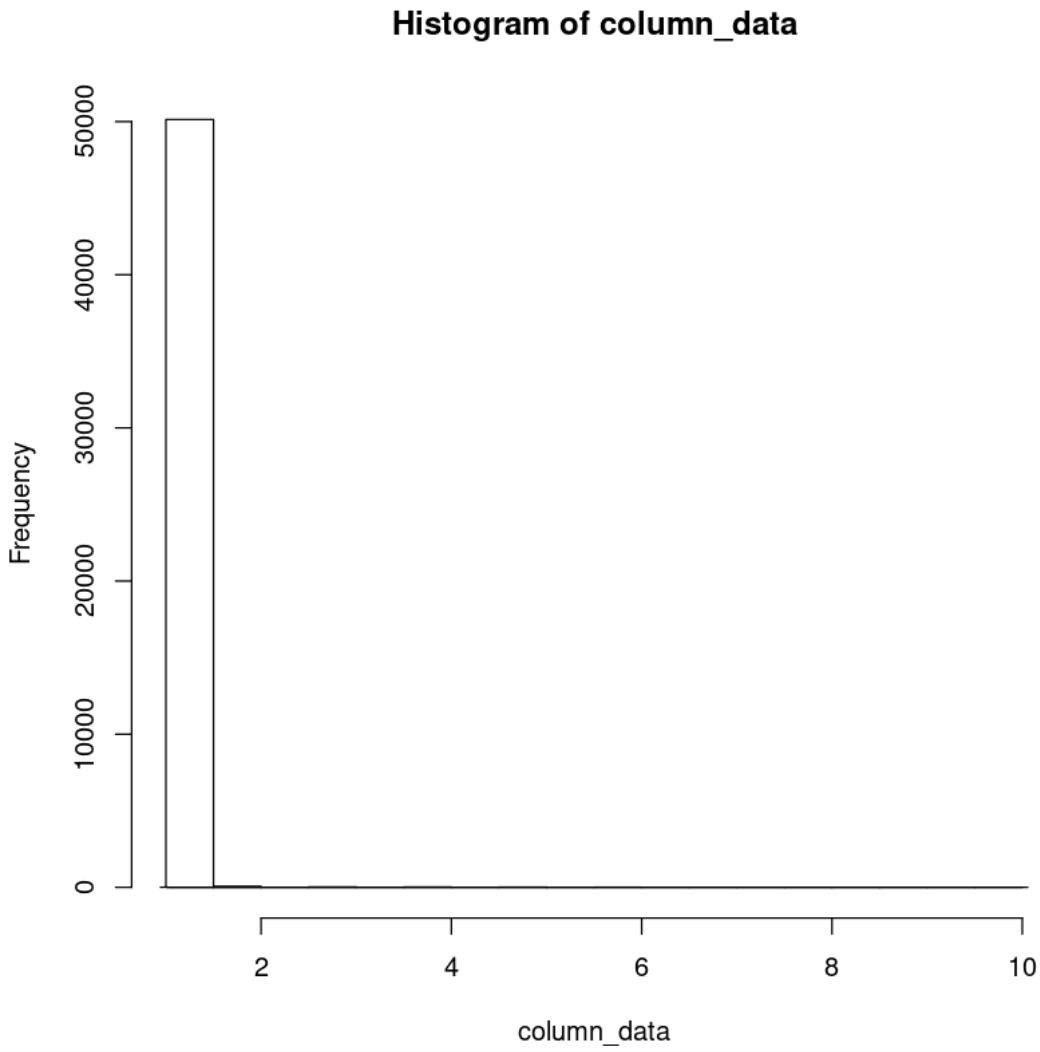



```
NULL
Processing column: Target
```


Histogram of column_data
Frequency
0
10000
20000
30000
40000
50000
2
4
6
8
10
column_data


Histogram for Target
Density
0.0
0.5
1.0
1.5
2.0
2
4
6
8
10
column_data

```
NULL
```



0.8.2 Power Transform

Data to Transform

Evaluating Yeo-Johnson tranform

Yeo-Johnson Transform Select columns

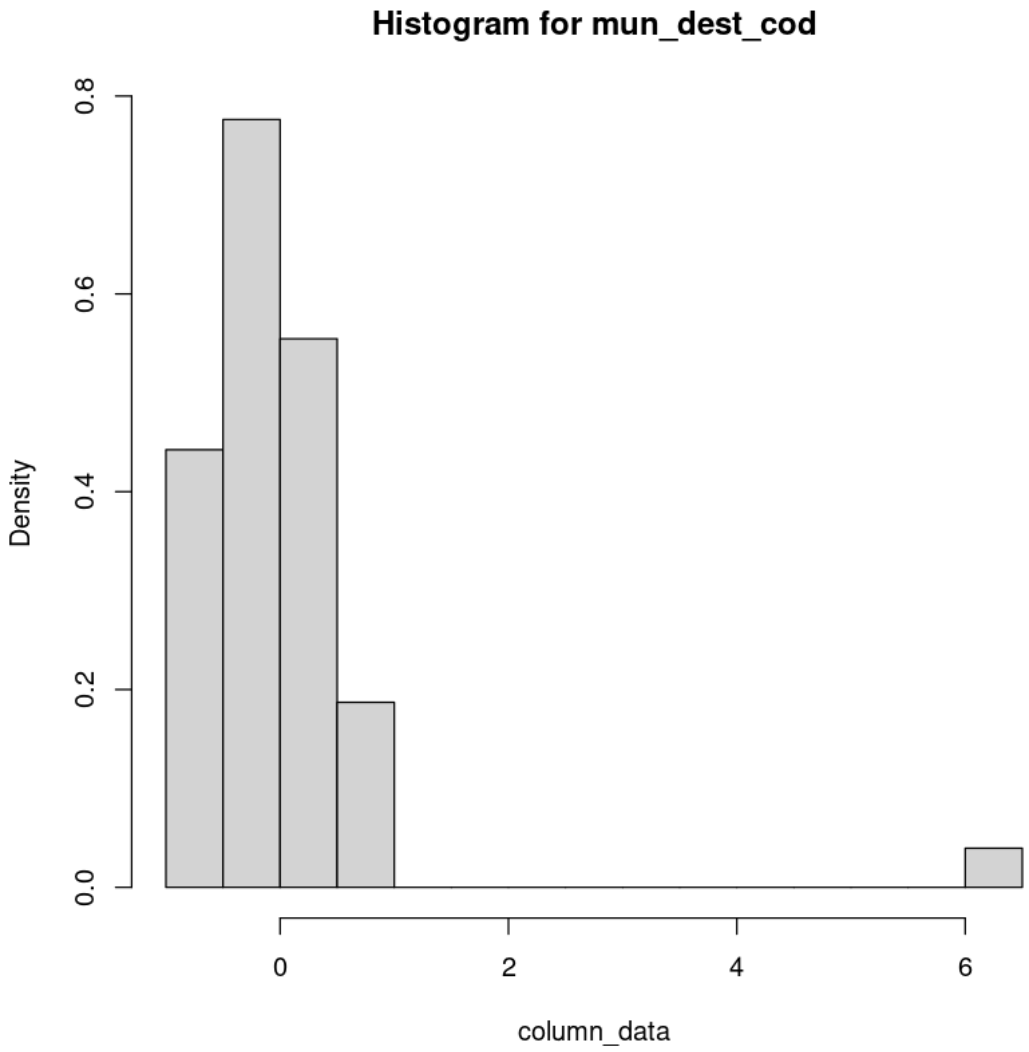
[31]:
```
cols <- sapply(data, is.numeric)
```

Operation

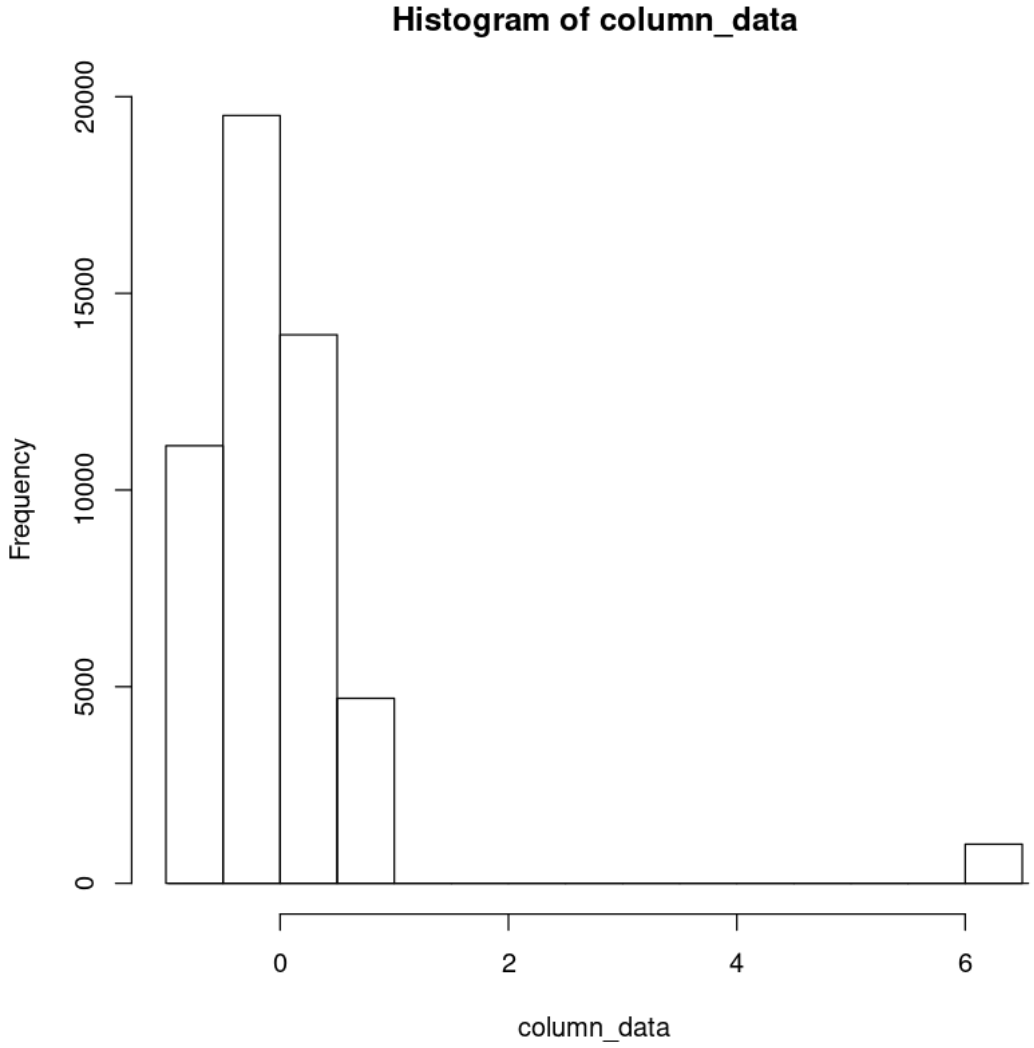
[32]:
```r
if (!require(bestNormalize)) {
  install.packages('bestNormalize')
}

# load the bestNormalize package
library(bestNormalize)

# assuming 'data' is your data frame

# identify the numeric columns
numeric_cols <- sapply(data, is.numeric)

# create a copy of the original data frame
data_yeojohnson <- data

# apply the Yeo-Johnson transformation to each numeric column
for (col_name in names(numeric_cols)[numeric_cols]) {
  yj <- yeojohnson(data[[col_name]])
  data_yeojohnson[[col_name]] <- yj$x.t
}
for (col_name in names(cols)[cols]) {
    cat("Processing column:", col_name, "\n")
    column_data <- data_yeojohnson[[col_name]]
    hist_plot <- hist(column_data, freq = FALSE, main = paste("Histogram for",␣
 ↪col_name))
    print(plot(hist_plot))
    lines(density(column_data))
}
```

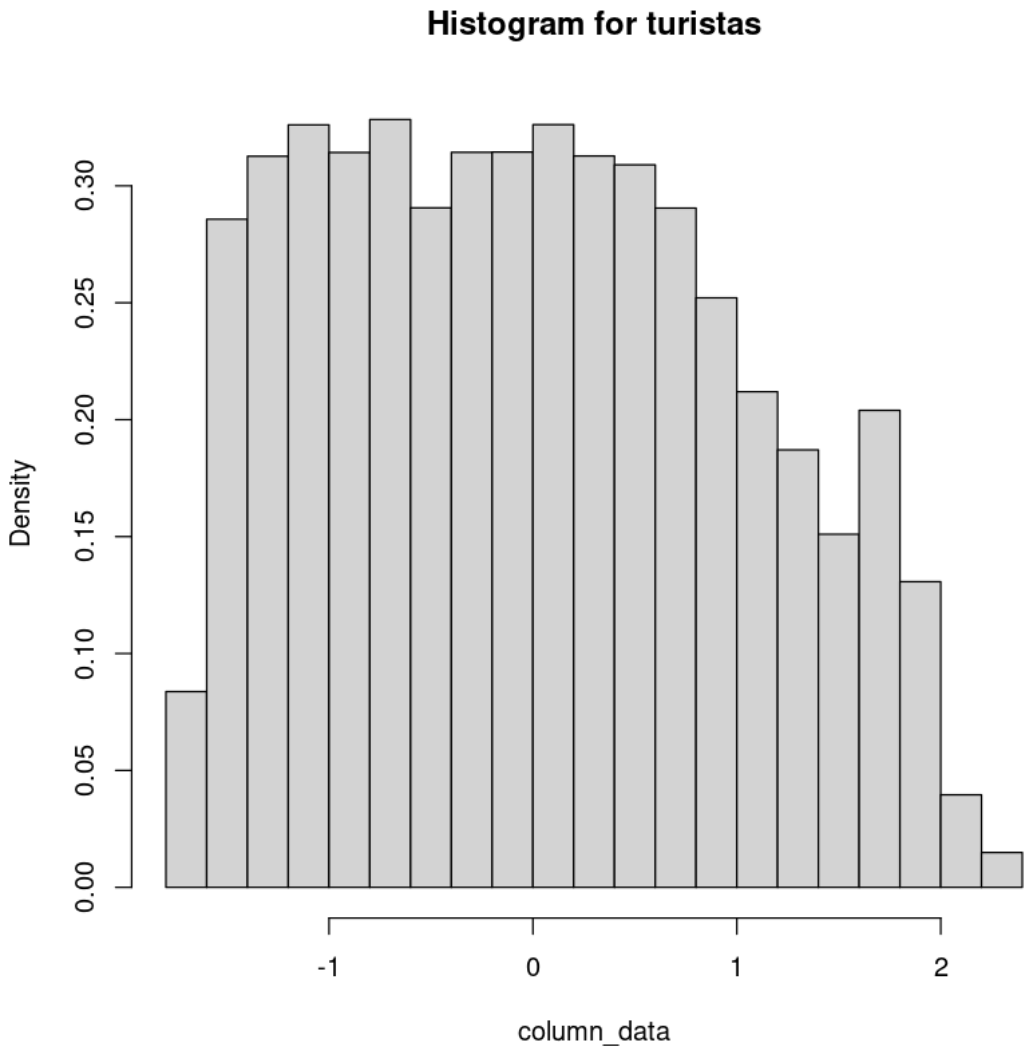
```
Loading required package: bestNormalize
```

```
Processing column: mun_dest_cod
```

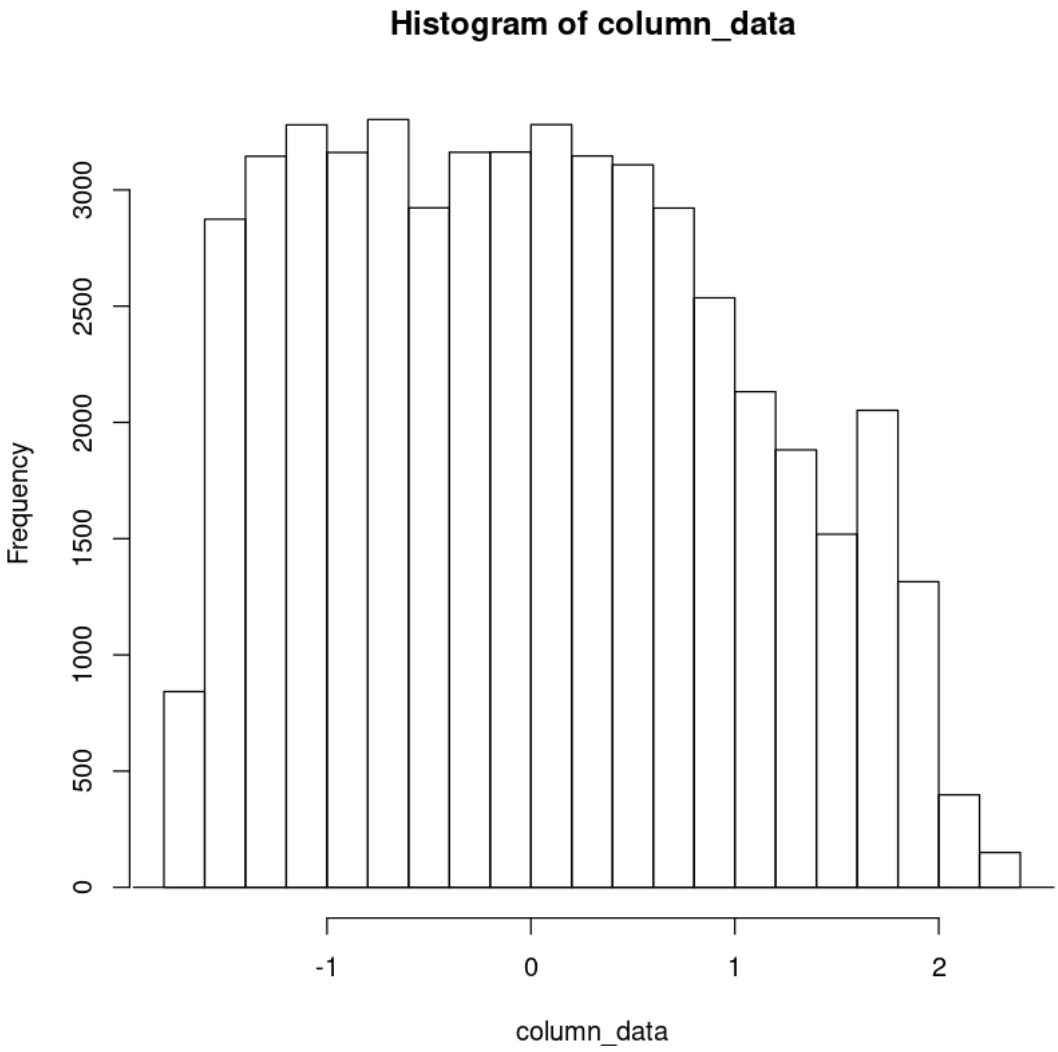


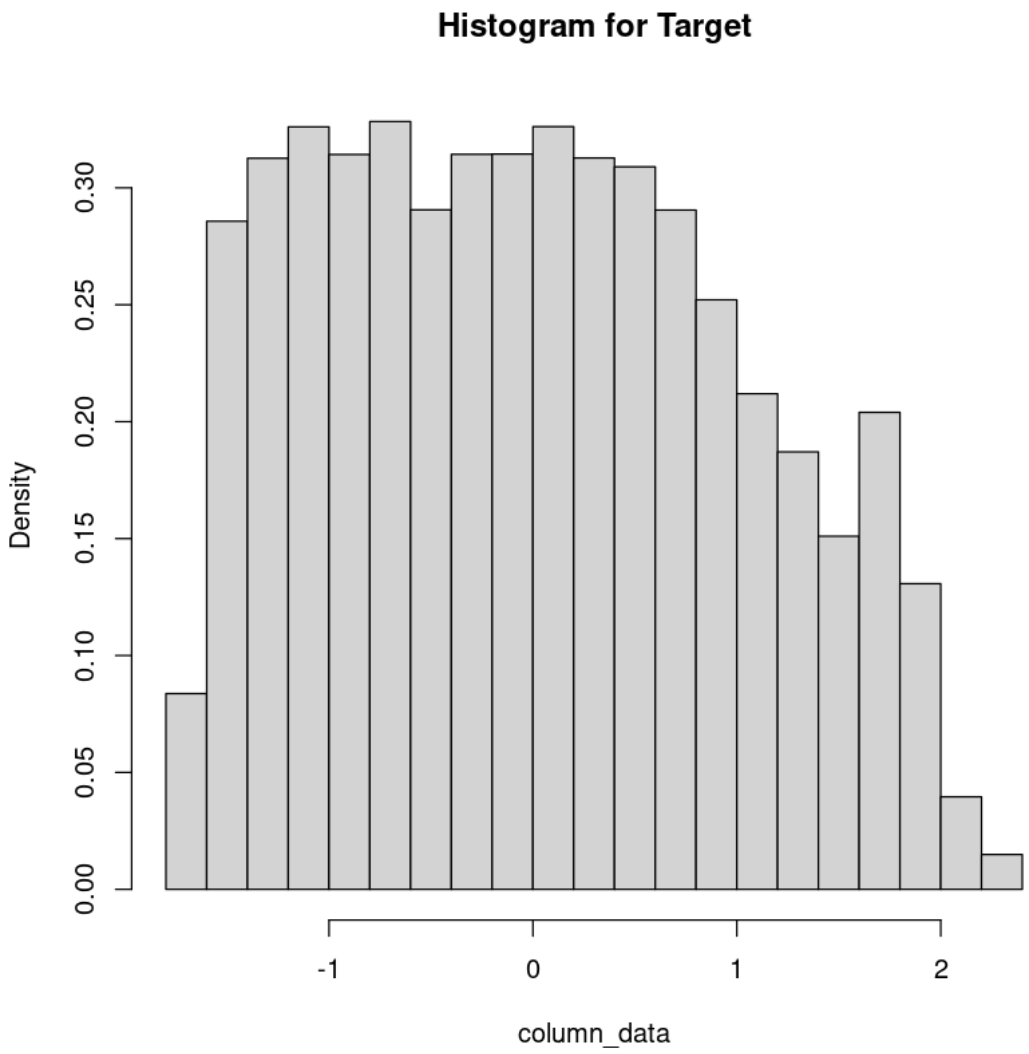

```
NULL
Processing column: turistas
```


Histogram of column_data
Frequency
0
5000
10000
15000
20000
0
2
4
6
column_data

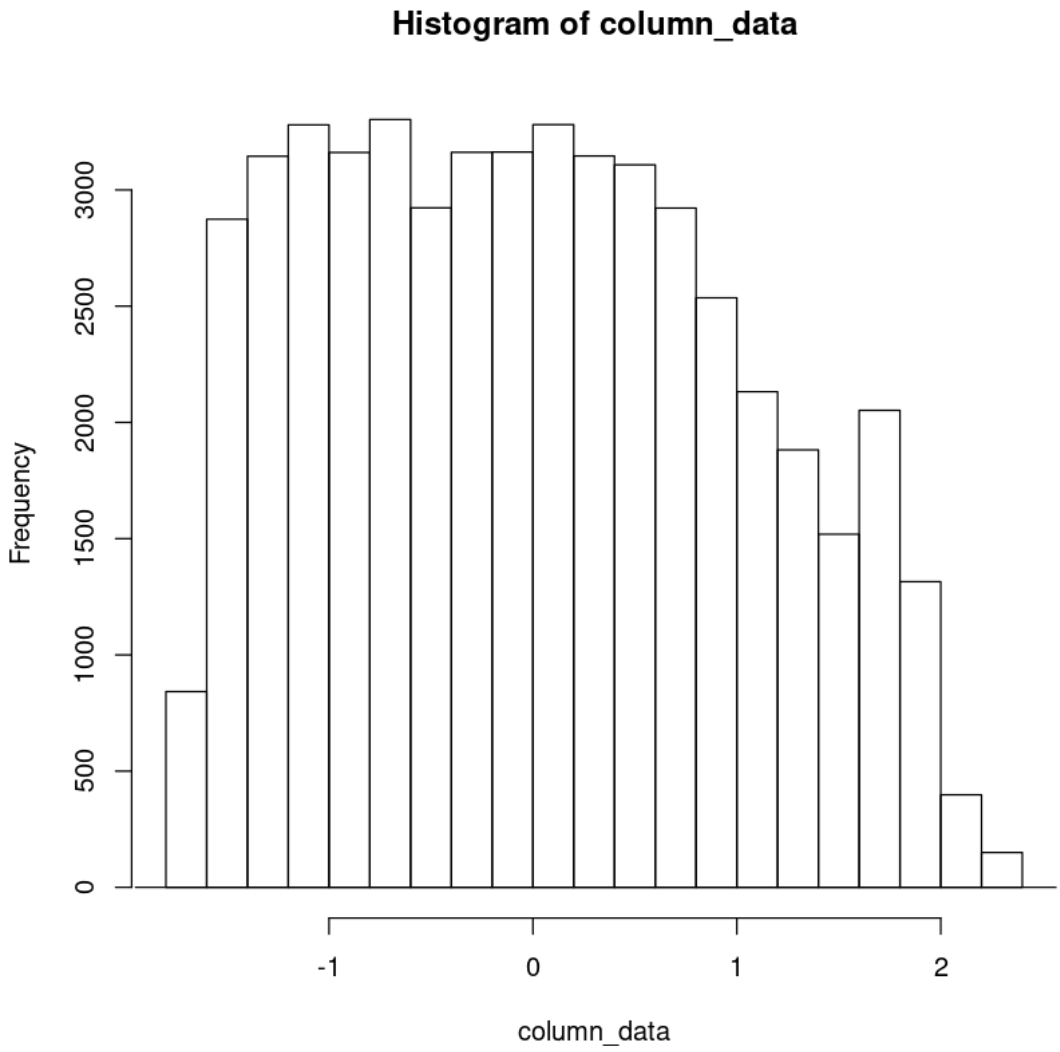



```
NULL
Processing column: Target
```


Histogram of column_data
Frequency
0
500
1000
1500
2000
2500
3000
-1
0
1
2
column_data


Histogram for Target
Density
0.00
0.05
0.10
0.15
0.20
0.25
0.30
-1
0
1
2
column_data

```
NULL
```


Histogram of column_data
Frequency
0
500
1000
1500
2000
2500
3000
-1
0
1
2
column_data