Minería de datos: PRA1 - Selección y preparación de un juego de datos

Autor: Juan Luis Acebal Rico

Noviembre 2024

Contents

Enunciado

Todo estudio analítico debe nacer de una necesidad por parte del **negocio** o de una voluntad de dotarle de un conocimiento contenido en los datos y que solo podremos obtener a través de una colección de buenas prácticas basadas en la Minería de Datos.

El mundo de la analítica de datos se sustenta en 3 ejes:

A. Uno de ellos es el profundo **conocimiento** que deberíamos tener **del negocio** al que tratamos de dar respuestas mediante los estudios analíticos.

B. El otro gran eje es sin duda las **capacidades analíticas** que seamos capaces de desplegar y en este sentido, las dos prácticas de esta asignatura pretenden que el estudiante realice un recorrido sólido por este segundo eje.

C. El tercer eje son los **Datos**. Las necesidades del Negocio deben concretarse con preguntas analíticas que a su vez sean viables responder a partir de los datos de que disponemos. La tarea de analizar los datos es sin duda importante, pero la tarea de identificarlos y obtenerlos va a ser para un analista un reto permanente.

Como **primera parte** del estudio analítico que nos disponemos a realizar, se pide al estudiante que complete los siguientes pasos:

1. Plantear un problema de analítica de datos detallando los objetivos analíticos y explica una metodología para resolverlos de acuerdo con lo que se ha practicado en las PEC y lo que se ha aprendido en el material didáctico.
2. Seleccionar un juego de datos y justificar su elección. El juego de datos **deberá tener capacidades** para que se le puedan aplicar **algoritmos supervisados** y **algoritmos no supervisados** en la PRA2 y deberá estar alineado con el problema analítico planteado en el paso anterior.

Requisito mínimo: El juego de datos deberá tener como mínimo 500 observaciones con un mínimo de 5 variables numéricas, 2 categóricas y 1 binaria. Además **debe ser distinto**, es importante que no sea un dataset usado en las PEC anteriores.

Adjuntamos aquí una lista de portales de datos abiertos para seleccionar el juego de datos. Se pueden usar otras fuentes para obtener vuestro juego de datos, pero recordad de citarlas:

- **Datos abiertos**
  - Google Dataset Search
  - Datos abiertos España
  - Datos abiertos Madrid
  - Datos abiertos Barcelona
  - Datos abiertos Londres
  - Datos abiertos New York
- **Conjuntos de datos para aprendizaje automático e investigación**
  - UCI Machine Learning
  - Datasets for machine-learning research (Wikipedia)
  - Kaggle datasets

3. Realizar un análisis exploratorio del juego de datos seleccionado.
4. Realizar tareas de limpieza y acondicionado para poder ser usado en procesos de modelado.
5. Realizar métodos de discretización
6. Aplicar un estudio PCA sobre el juego de datos. A pesar de no estar explicado en el material didáctico, se valorará si en lugar de PCA investigáis por vuestra cuenta y aplicáis SVD (Single Value Decomposition).

- **Algunos recursos**
  - PCA para reducción de dimensiones
  - SVD Singular Value Decomposition

Recordad que para todas las PRA es **necesario documentar** en cada apartado del ejercicio práctico que se ha hecho, por qué se ha hecho y cómo se ha hecho. Asimismo, todas las decisiones y conclusiones deberán ser presentados de forma razonada y clara, **contextualizando los resultados**, es decir, especificando todos y cada uno de los pasos que se hayan llevado a cabo para su resolución. Por último, incluid una **conclusión final** resumiendo los resultados obtenidos en la práctica e indicad eventuales **citaciones bibliográficas**, fuentes internas/externas y materiales de investigación.

Recursos de programación

- Incluimos en este apartado una lista de recursos de programación para minería de datos donde podréis encontrar ejemplos, ideas e inspiración:
  - Material adicional del libro: Minería de datos Modelos y Algoritmos
  - Espacio de recursos UOC para ciencia de datos

- Buscador de código R
- Colección de cheatsheets en R

El dataset ha sido sacado de aqui: https://www.kaggle.com/datasets/die9origephit/amazon-data-science-books

```
#!/bin/bash
#kaggle datasets download die9origephit/amazon-data-science-books
```

```
#unzip amazon-data-science-books.zip
```

```
library(reticulate)
py_config()
```

```
## python:         /Users/juan/.virtualenvs/r-reticulate/bin/python
## libpython:      /Users/juan/.pyenv/versions/3.8.17/lib/libpython3.8.dylib
## pythonhome:     /Users/juan/.virtualenvs/r-reticulate:/Users/juan/.virtualenvs/r-reticulate
## version:        3.8.17 (default, Sep 19 2024, 09:05:54)  [Clang 15.0.0 (clang-1500.3.9.4)]
## numpy:          /Users/juan/.virtualenvs/r-reticulate/lib/python3.8/site-packages/numpy
## numpy_version:  1.24.4
##
## NOTE: Python version was forced by VIRTUAL_ENV
```

```
py_install("pandas")
```

```
## Using virtual environment '/Users/juan/.virtualenvs/r-reticulate' ...

## + /Users/juan/.virtualenvs/r-reticulate/bin/python -m pip install --upgrade --no-user pandas
```

```
py_install("matplotlib")
```

```
## Using virtual environment '/Users/juan/.virtualenvs/r-reticulate' ...

## + /Users/juan/.virtualenvs/r-reticulate/bin/python -m pip install --upgrade --no-user matplotlib
```

```
py_install("seaborn")
```

```
## Using virtual environment '/Users/juan/.virtualenvs/r-reticulate' ...

## + /Users/juan/.virtualenvs/r-reticulate/bin/python -m pip install --upgrade --no-user seaborn
```

```
py_install("Jinja2")
```

```
## Using virtual environment '/Users/juan/.virtualenvs/r-reticulate' ...

## + /Users/juan/.virtualenvs/r-reticulate/bin/python -m pip install --upgrade --no-user Jinja2
```

```
py_install("scikit-learn")
```

```
## Using virtual environment '/Users/juan/.virtualenvs/r-reticulate' ...
```

```
## + /Users/juan/.virtualenvs/r-reticulate/bin/python -m pip install --upgrade --no-user scikit-learn
```

```
import pandas as pd
import numpy as np
df = pd.read_csv("final_book_dataset_kaggle2.csv")
```

```
df.head().to_csv('head_df.csv')
```

```
head_df <- read.csv('head_df.csv')
print(head_df)
```

```
##   X
## 1 0
## 2 1
## 3 2
## 4 3
## 5 4
##                                                                                 title
## 1                    Data Analysis Using R (Low Priced Edition): A Primer for Data Scientist
## 2 Head First Data Analysis: A learner's guide to big numbers, statistics, and good decisions
## 3   Guerrilla Data Analysis Using Microsoft Excel: Overcoming Crap Data and Excel Skirmishes
## 4                   Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython
## 5                              Excel Data Analysis For Dummies (For Dummies (Computer/Tech))
##                              author price price..including.used.books. pages
## 1                [ Dr Dhaval Maheta]  6.75                          6.75   500
## 2                                    33.72              21.49 - 33.72    484
## 3 [ Oz du Soleil,  and , Bill Jelen] 32.07                       32.07   274
## 4                [ William McKinney] 53.99                       53.99   547
## 5                  [ Paul McFedries] 24.49                       24.49   368
##   avg_reviews n_reviews star5 star4 star3 star2 star1               dimensions
## 1         4.4        23   55%   39%    6%                 8.5 x 1.01 x 11 inches
## 2         4.3       124   61%   20%    9%    4%    6%    8 x 0.98 x 9.25 inches
## 3         4.7        10   87%   13%                    8.25 x 0.6 x 10.75 inches
## 4         4.6     1,686   75%   16%    5%    2%    2%    7 x 1.11 x 9.19 inches
## 5         3.9        12   52%   17%   10%   10%   10% 7.38 x 0.83 x 9.25 inches
##        weight language                                          publisher
## 1 2.53 pounds  English    Notion Press Media Pvt Ltd (November 22, 2021)
## 2 1.96 pounds  English     O'Reilly Media; 1st edition (August 18, 2009)
## 3  1.4 pounds  English Holy Macro! Books; Third edition (August 1, 2022)
## 4 1.47 pounds  English   O'Reilly Media; 2nd edition (November 14, 2017)
## 5  1.3 pounds  English       For Dummies; 5th edition (February 3, 2022)
##          ISBN_13
## 1 978-1685549596
## 2 978-0596153939
## 3 978-1615470747
## 4 978-1491957660
## 5 978-1119844426
```

```
##
## 1           /Data-Analysis-Using-Low-Priced/dp/1685549594/ref=sr_1_16?keywords=data+analysis&qid=167
## 2      /Head-First-Data-Analysis-statistics/dp/0596153937/ref=sr_1_15?keywords=data+analysis&qid=167
## 3 /Guerrilla-Analysis-Using-Microsoft-Excel/dp/1615470743/ref=sr_1_14?keywords=data+analysis&qid=167
## 4   /Python-Data-Analysis-Wrangling-IPython/dp/1491957662/ref=sr_1_11?keywords=data+analysis&qid=167
## 5     /Excel-Data-Analysis-Dummies-Computer/dp/1119844428/ref=sr_1_12?keywords=data+analysis&qid=167
##
## 1           https://www.amazon.com/Data-Analysis-Using-Low-Priced/dp/1685549594/ref=sr_1_16?keywords=
## 2      https://www.amazon.com/Head-First-Data-Analysis-statistics/dp/0596153937/ref=sr_1_15?keywords=
## 3 https://www.amazon.com/Guerrilla-Analysis-Using-Microsoft-Excel/dp/1615470743/ref=sr_1_14?keywords=
## 4   https://www.amazon.com/Python-Data-Analysis-Wrangling-IPython/dp/1491957662/ref=sr_1_11?keywords=
## 5     https://www.amazon.com/Excel-Data-Analysis-Dummies-Computer/dp/1119844428/ref=sr_1_12?keywords=
```

```
df.info()
```

```
## <class 'pandas.core.frame.DataFrame'>
## RangeIndex: 830 entries, 0 to 829
## Data columns (total 19 columns):
##  #   Column                         Non-Null Count  Dtype
## ---  ------                         --------------  -----
##  0   title                          830 non-null    object
##  1   author                         657 non-null    object
##  2   price                          722 non-null    float64
##  3   price (including used books)   722 non-null    object
##  4   pages                          745 non-null    object
##  5   avg_reviews                    702 non-null    float64
##  6   n_reviews                      702 non-null    object
##  7   star5                          702 non-null    object
##  8   star4                          635 non-null    object
##  9   star3                          554 non-null    object
##  10  star2                          451 non-null    object
##  11  star1                          328 non-null    object
##  12  dimensions                     644 non-null    object
##  13  weight                         651 non-null    object
##  14  language                       759 non-null    object
##  15  publisher                      714 non-null    object
##  16  ISBN_13                        665 non-null    object
##  17  link                           830 non-null    object
##  18  complete_link                  830 non-null    object
## dtypes: float64(2), object(17)
## memory usage: 123.3+ KB
```

Tenemos 830 observaciones y 18 variables. 5 variables numéricas, 6 categóricas y 1 binaria.

```
col_a_fillna = ['star5', 'star4', 'star3', 'star2', 'star1']
df[col_a_fillna] = df[col_a_fillna].fillna(0)
df[col_a_fillna] = df[col_a_fillna].replace('[^\d]', '', regex=True).astype(int)
df[col_a_fillna] = df[col_a_fillna].astype(int)

print(df.isnull().sum())
```

```
## title                          0
## author                       173
```

```
## price                            108
## price (including used books)     108
## pages                             85
## avg_reviews                      128
## n_reviews                        128
## star5                              0
## star4                              0
## star3                              0
## star2                              0
## star1                              0
## dimensions                       186
## weight                           179
## language                          71
## publisher                        116
## ISBN_13                          165
## link                               0
## complete_link                      0
## dtype: int64
```

```
df=df[(df['star5'] + df['star4'] + df['star3'] + df['star2'] + df['star1'] <= 101) &
                (df['star5'] + df['star4'] + df['star3'] + df['star2'] + df['star1'] >= 99)]
```

Registros que la suma de las variables 'star5', 'star4', 'star3', 'star2', 'star1' esta entre 99 y 101, para eliminar los registros que no cumplan con esta condición ya que seria nulo (101 y 99 lo permito ya que por el redondeo puede ser posible)

```
df.rename(columns={'language': 'ingles'}, inplace=True)
df['ingles'] = df['ingles'].apply(lambda x: 1 if x == 'English' else 0)
```

```
#
pd.crosstab(index=df['ingles'], columns="count")
```

```
## col_0   count
## ingles
## 0          50
## 1         652
```

Tabla de contingencia de la variable ingles

```
df['Peso (g)'] = df['weight'].str.extract('(\d+\.?\d*)').astype(float) * 453.592
df.drop(columns=['weight'], inplace=True)
```

```
df.describe().to_csv('df.describe.csv')
```

```
df.describe <- read.csv('df.describe.csv')
print(df.describe)
```

```
##       X     price avg_reviews     star5      star4      star3      star2
## 1 count 637.00000 702.0000000 702.00000 702.000000 702.000000 702.000000
## 2  mean  42.03755   4.4720798  72.95299  15.484330   6.831909   2.756410
## 3   std  31.61785   0.4096078  14.72145   9.736714   6.166515   3.382709
```

```
## 4   min   0.99000   1.0000000  20.00000   0.000000   0.000000   0.000000
## 5   25%  22.49000   4.3250000  64.00000  10.250000   3.000000   0.000000
## 6   50%  37.49000   4.5000000  73.00000  15.000000   6.000000   2.000000
## 7   75%  49.99000   4.7000000  81.00000  20.000000   9.000000   4.000000
## 8   max 287.14000   5.0000000 100.00000  64.000000  49.000000  22.000000
##        star1      ingles    Peso..g.
## 1 702.000000 702.0000000   583.00000
## 2   1.972934   0.9287749  2031.46585
## 3   3.152448   0.2573838  2129.58665
## 4   0.000000   0.0000000    15.87572
## 5   0.000000   1.0000000   666.78024
## 6   0.000000   1.0000000   943.47136
## 7   3.000000   1.0000000  2358.67840
## 8  29.000000   1.0000000 13698.47840
```

```
#df.info()
```

```
df = df.dropna(subset=['price'])
```

```
#df.info()
```

```
#print(df[['price', 'avg_reviews', 'pages', 'Peso (g)']].head())
#df[['price', 'avg_reviews', 'pages', 'Peso (g)']].describe()
```

```
df.info()
```

```
## <class 'pandas.core.frame.DataFrame'>
## Index: 637 entries, 0 to 829
## Data columns (total 19 columns):
##  #   Column                          Non-Null Count  Dtype
## ---  ------                          --------------  -----
##  0   title                           637 non-null    object
##  1   author                          528 non-null    object
##  2   price                           637 non-null    float64
##  3   price (including used books)    637 non-null    object
##  4   pages                           621 non-null    object
##  5   avg_reviews                     637 non-null    float64
##  6   n_reviews                       637 non-null    object
##  7   star5                           637 non-null    int64
##  8   star4                           637 non-null    int64
##  9   star3                           637 non-null    int64
##  10  star2                           637 non-null    int64
##  11  star1                           637 non-null    int64
##  12  dimensions                      588 non-null    object
##  13  ingles                          637 non-null    int64
##  14  publisher                       617 non-null    object
##  15  ISBN_13                         590 non-null    object
##  16  link                            637 non-null    object
##  17  complete_link                   637 non-null    object
##  18  Peso (g)                        581 non-null    float64
## dtypes: float64(3), int64(6), object(10)
## memory usage: 99.5+ KB
```

```
df['author'] = df['author'].fillna('N/A')
df['publisher'] = df['publisher'].fillna('N/A')
df['ISBN_13'] = df['ISBN_13'].fillna('N/A')

col_a_eliminar = ['link', 'complete_link']
df = df.drop(columns=col_a_eliminar)
```

```
df.info()
```

```
## <class 'pandas.core.frame.DataFrame'>
## Index: 637 entries, 0 to 829
## Data columns (total 17 columns):
##  #   Column                          Non-Null Count  Dtype
## ---  ------                          --------------  -----
##  0   title                           637 non-null    object
##  1   author                          637 non-null    object
##  2   price                           637 non-null    float64
##  3   price (including used books)    637 non-null    object
##  4   pages                           621 non-null    object
##  5   avg_reviews                     637 non-null    float64
##  6   n_reviews                       637 non-null    object
##  7   star5                           637 non-null    int64
##  8   star4                           637 non-null    int64
##  9   star3                           637 non-null    int64
##  10  star2                           637 non-null    int64
##  11  star1                           637 non-null    int64
##  12  dimensions                      588 non-null    object
##  13  ingles                          637 non-null    int64
##  14  publisher                       637 non-null    object
##  15  ISBN_13                         637 non-null    object
##  16  Peso (g)                        581 non-null    float64
## dtypes: float64(3), int64(6), object(8)
## memory usage: 89.6+ KB
```

```
from sklearn.impute import KNNImputer


df['pages'] = pd.to_numeric(df['pages'], errors='coerce')
df['n_reviews'] = pd.to_numeric(df['n_reviews'], errors='coerce')


variables_para_imputar = ['price', 'avg_reviews', 'pages', 'Peso (g)', 'n_reviews']

for col in variables_para_imputar:
    df[col] = pd.to_numeric(df[col], errors='coerce')

df_imputar = df[variables_para_imputar]
imputador = KNNImputer(n_neighbors=5)
df_imputado = pd.DataFrame(imputador.fit_transform(df_imputar), columns=variables_para_imputar)

for col in variables_para_imputar:
    df[col] = df_imputado[col]
```

```
#print(df.isnull().sum())
```

```
df.info()
```

```
## <class 'pandas.core.frame.DataFrame'>
## Index: 637 entries, 0 to 829
## Data columns (total 17 columns):
##  #   Column                          Non-Null Count  Dtype
## ---  ------                          --------------  -----
##  0   title                           637 non-null    object
##  1   author                          637 non-null    object
##  2   price                           502 non-null    float64
##  3   price (including used books)    637 non-null    object
##  4   pages                           502 non-null    float64
##  5   avg_reviews                     502 non-null    float64
##  6   n_reviews                       502 non-null    float64
##  7   star5                           637 non-null    int64
##  8   star4                           637 non-null    int64
##  9   star3                           637 non-null    int64
##  10  star2                           637 non-null    int64
##  11  star1                           637 non-null    int64
##  12  dimensions                      588 non-null    object
##  13  ingles                          637 non-null    int64
##  14  publisher                       637 non-null    object
##  15  ISBN_13                         637 non-null    object
##  16  Peso (g)                        502 non-null    float64
## dtypes: float64(5), int64(6), object(6)
## memory usage: 89.6+ KB
```

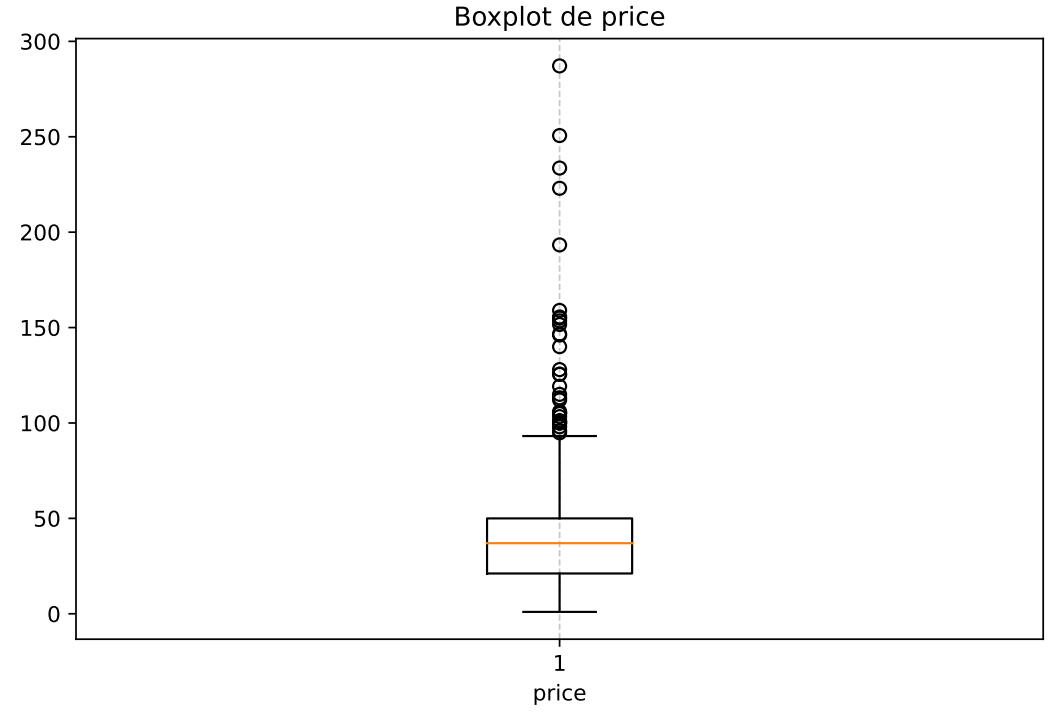
```
print(df.isnull().sum())
```

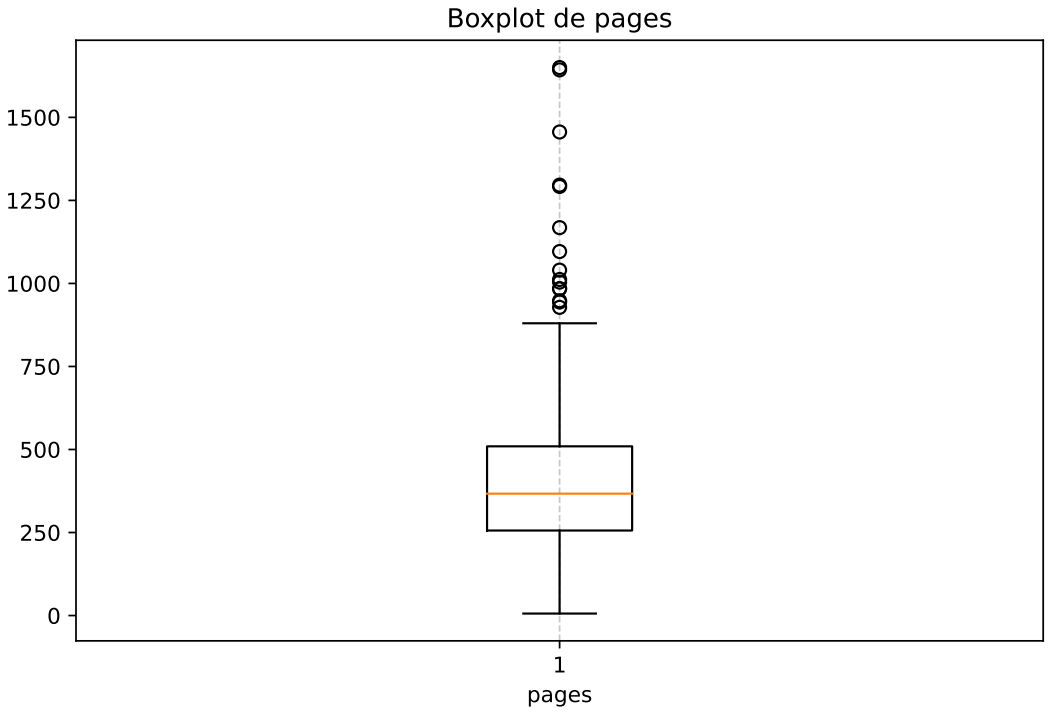
```
## title                             0
## author                            0
## price                           135
## price (including used books)      0
## pages                           135
## avg_reviews                     135
## n_reviews                       135
## star5                             0
## star4                             0
## star3                             0
## star2                             0
## star1                             0
## dimensions                       49
## ingles                            0
## publisher                         0
## ISBN_13                           0
## Peso (g)                        135
## dtype: int64
```

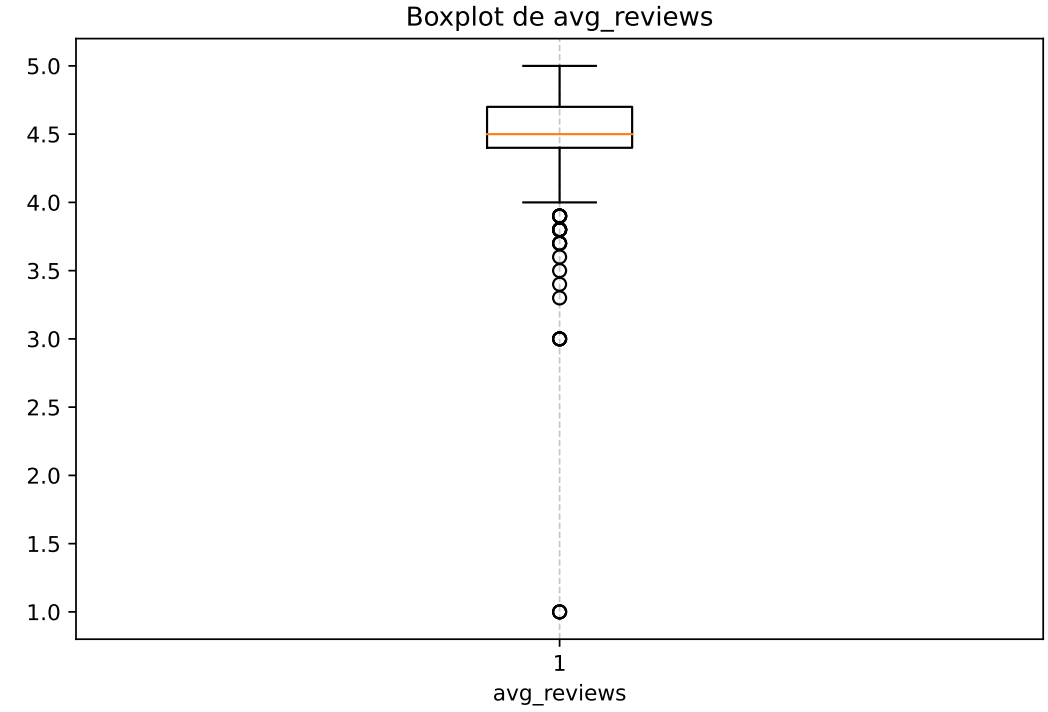
```
df = df.drop(columns=['dimensions', 'ISBN_13'])
df = df.dropna()
print(df.isnull().sum())
```

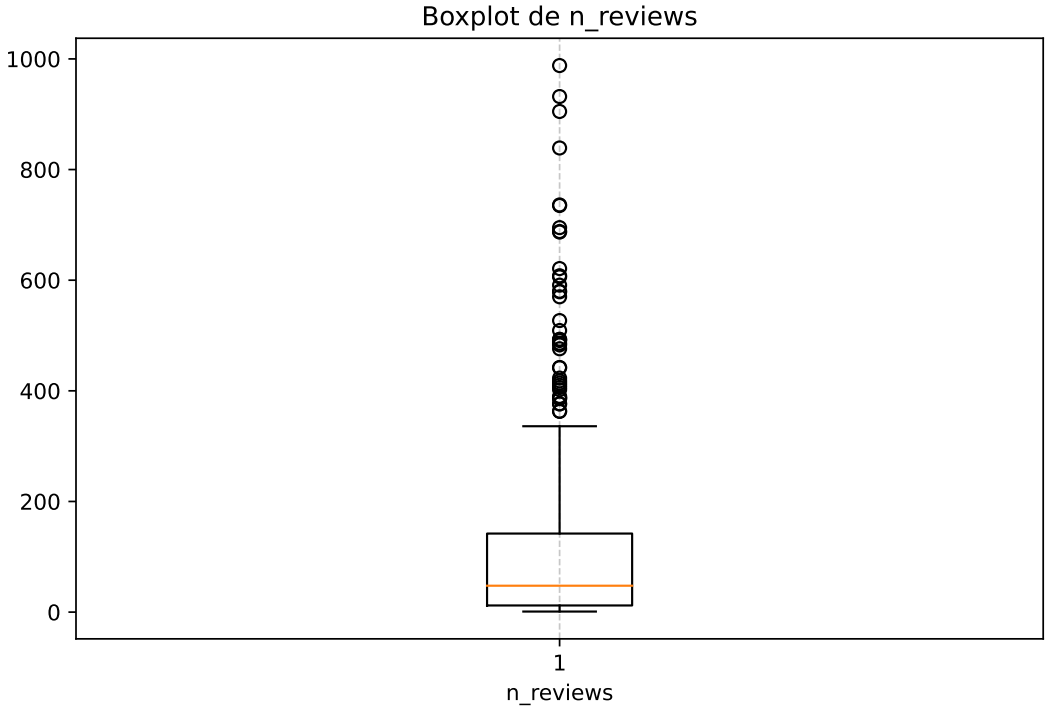
```
## title                          0
## author                         0
## price                          0
## price (including used books)   0
## pages                          0
## avg_reviews                    0
## n_reviews                      0
## star5                          0
## star4                          0
## star3                          0
## star2                          0
## star1                          0
## ingles                         0
## publisher                      0
## Peso (g)                       0
## dtype: int64
```

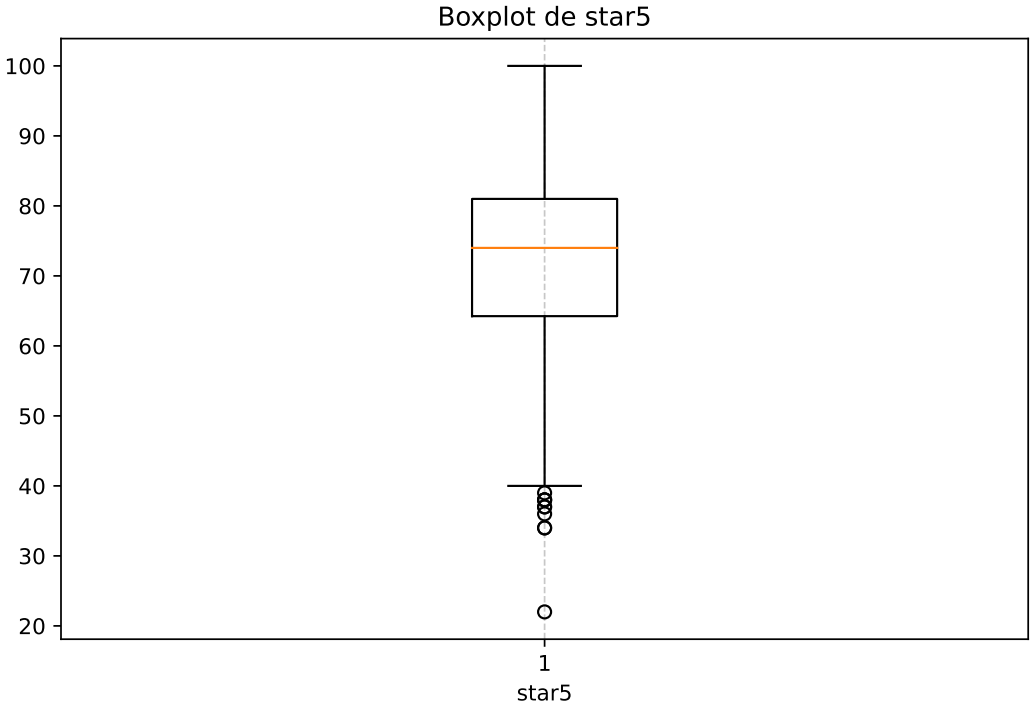
```
import matplotlib.pyplot as plt

for variable in df.select_dtypes(include=['number']).columns:
    plt.figure(figsize=(8, 5))
    plt.boxplot(df[variable].dropna())
    plt.title(f'Boxplot de {variable}')
    plt.xlabel(variable)
    plt.grid(axis='x', linestyle='--', alpha=0.7)
    plt.show()
    plt.close()
```

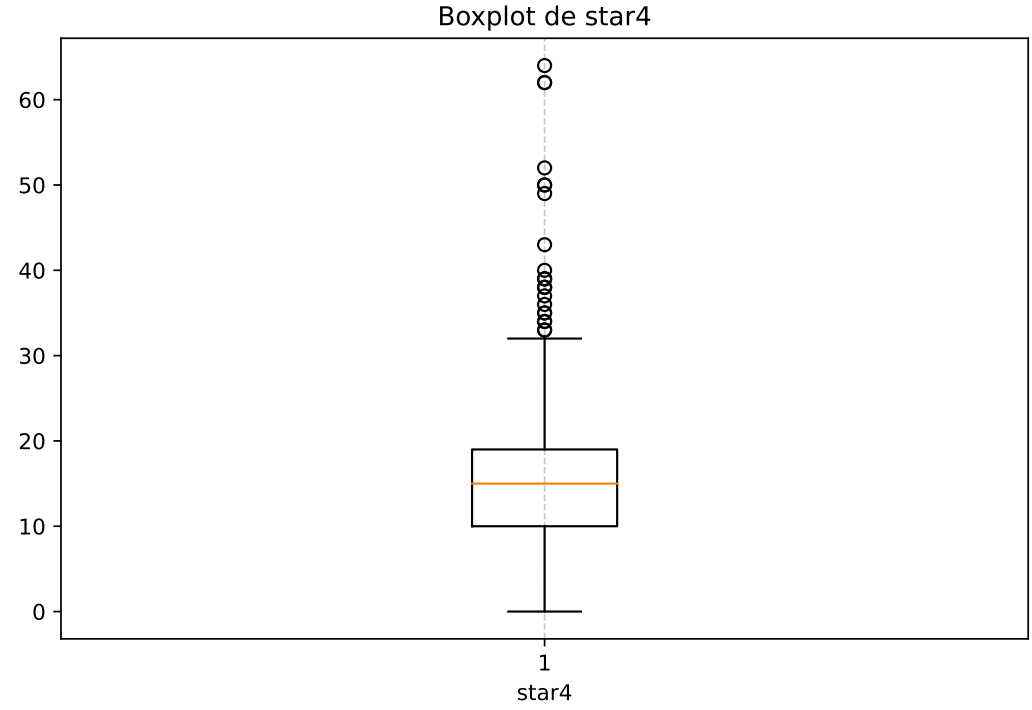

Boxplot de price
300
250
200
150
100
50
0
1
price

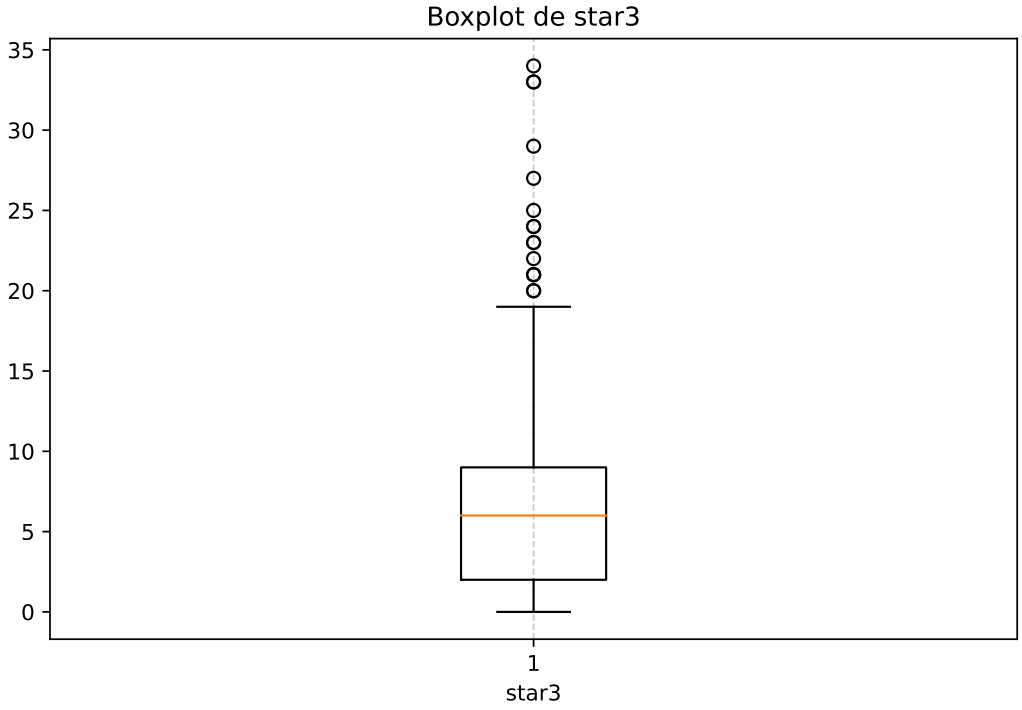

Boxplot de pages
1500
1250
1000
750
500
250
0
1
pages

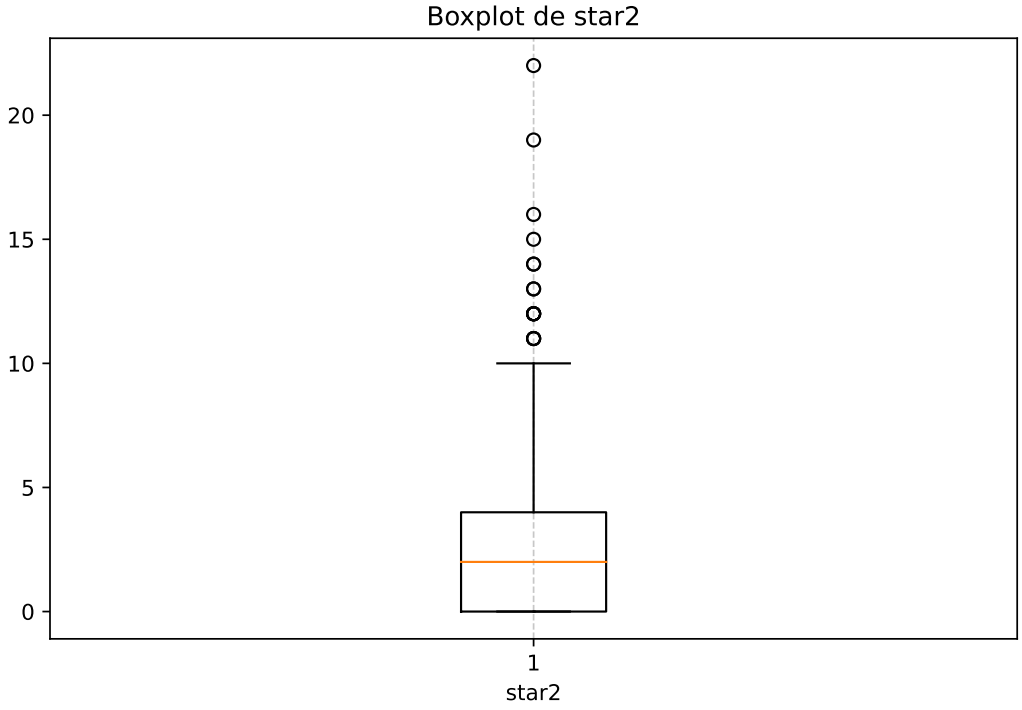

Boxplot de avg_reviews
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
1
avg_reviews

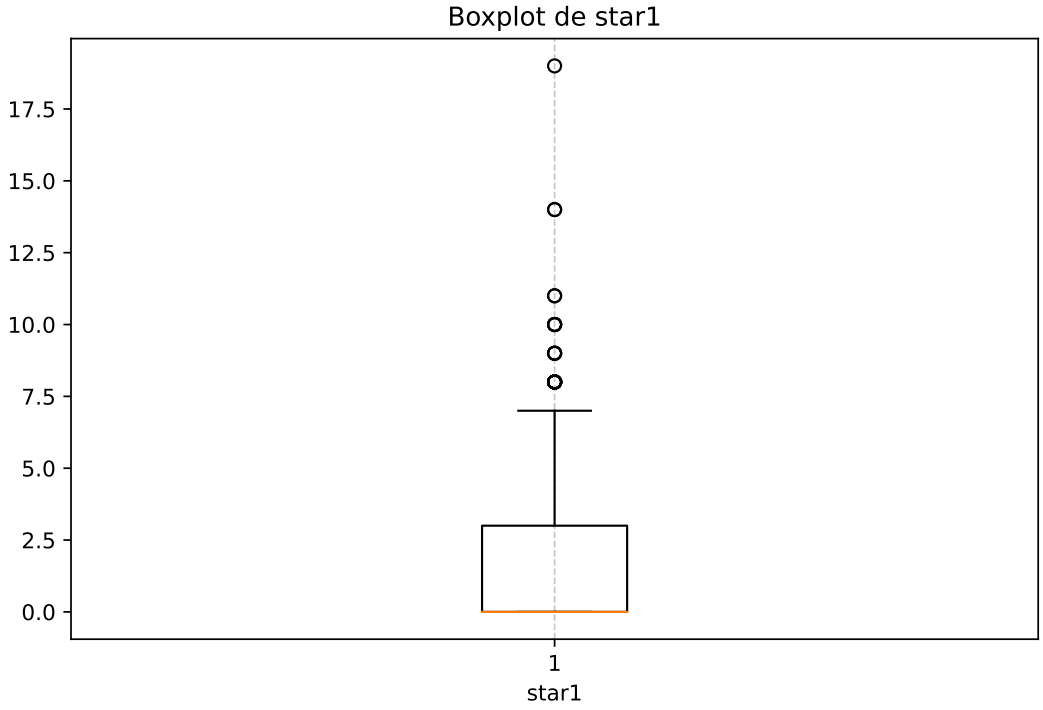

Boxplot de n_reviews
1000
800
600
400
200
0
1
n_reviews

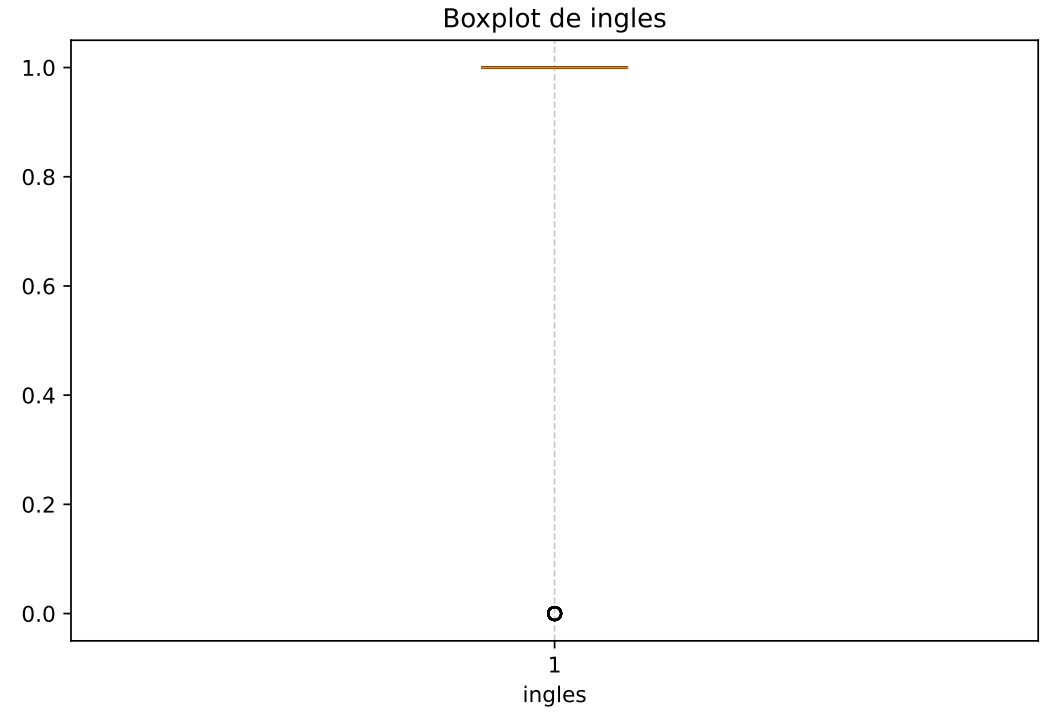

Boxplot de star5
100
90
80
70
60
50
40
30
20
1
star5

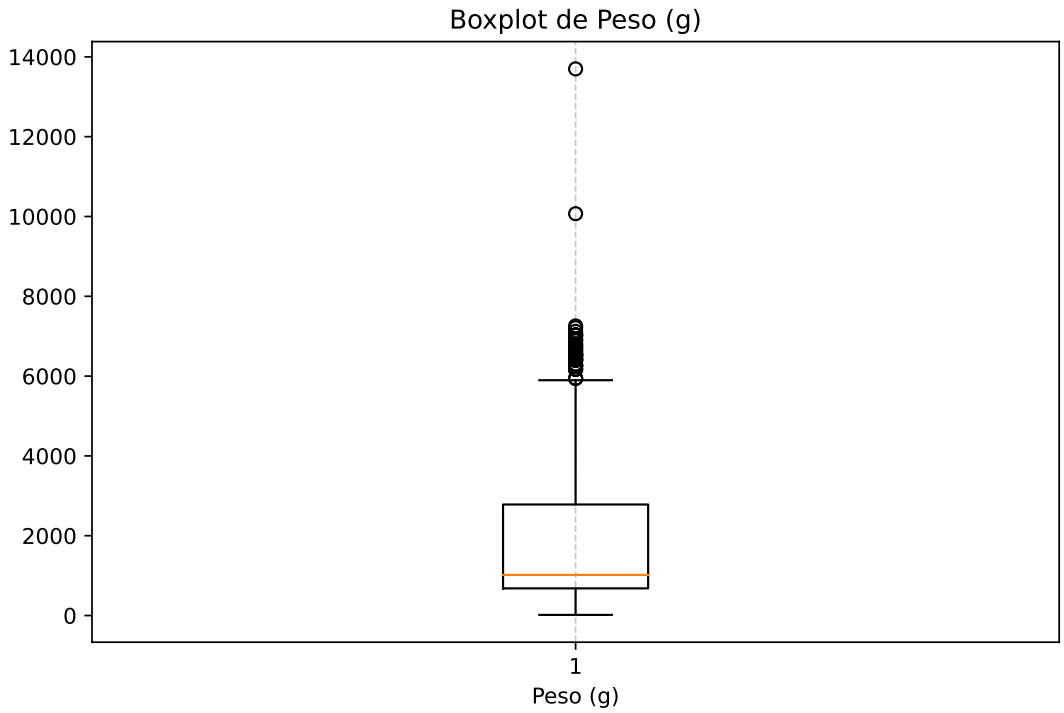

Boxplot de star4
60
50
40
30
20
10
0
1
star4


Boxplot de star3
35
30
25
20
15
10
5
0
1
star3

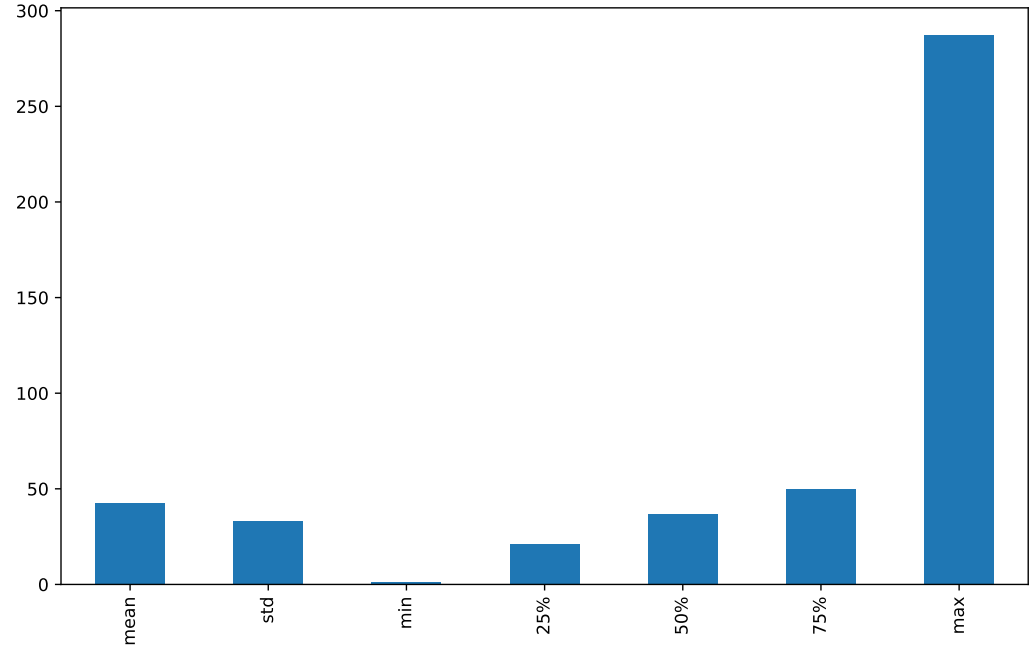

Boxplot de star2
20
15
10
5
0
1
star2

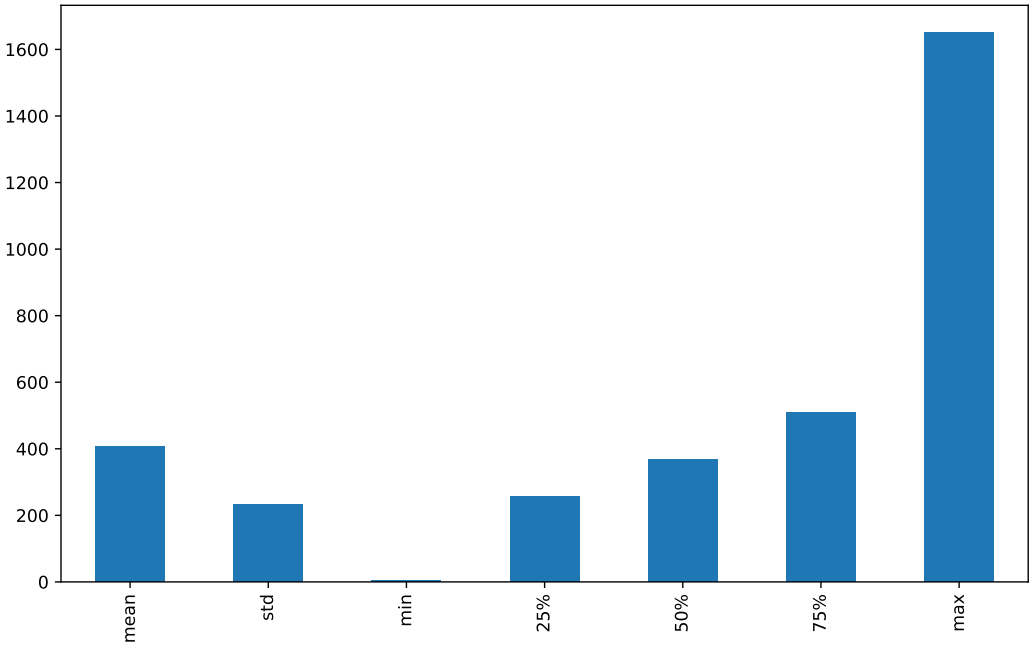

Boxplot de star1
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
1
star1

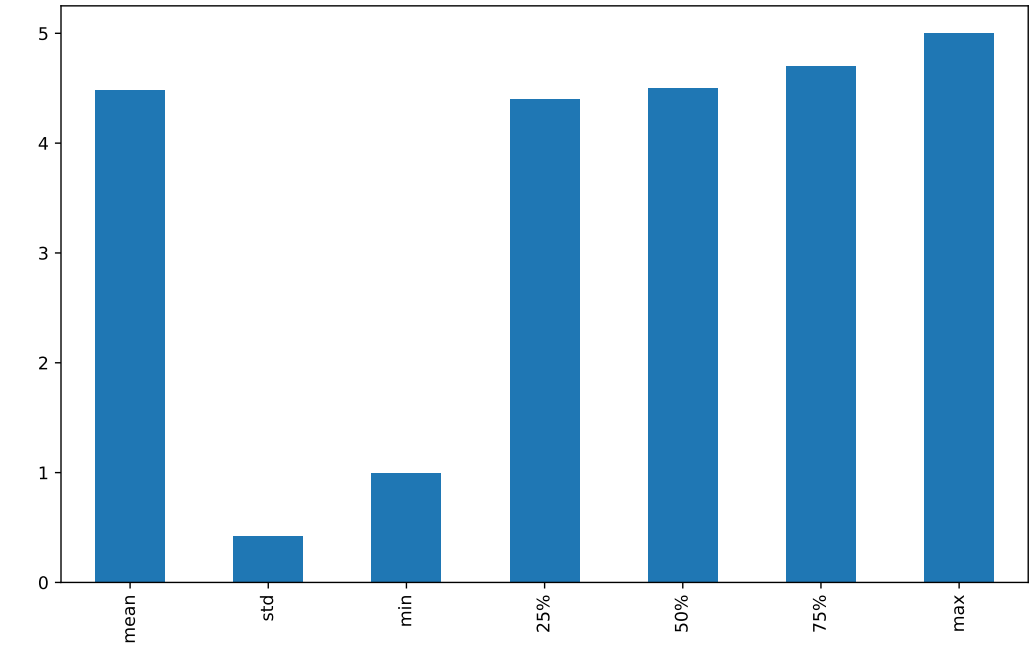

Boxplot de ingles
1.0
0.8
0.6
0.4
0.2
0.0
1
ingles

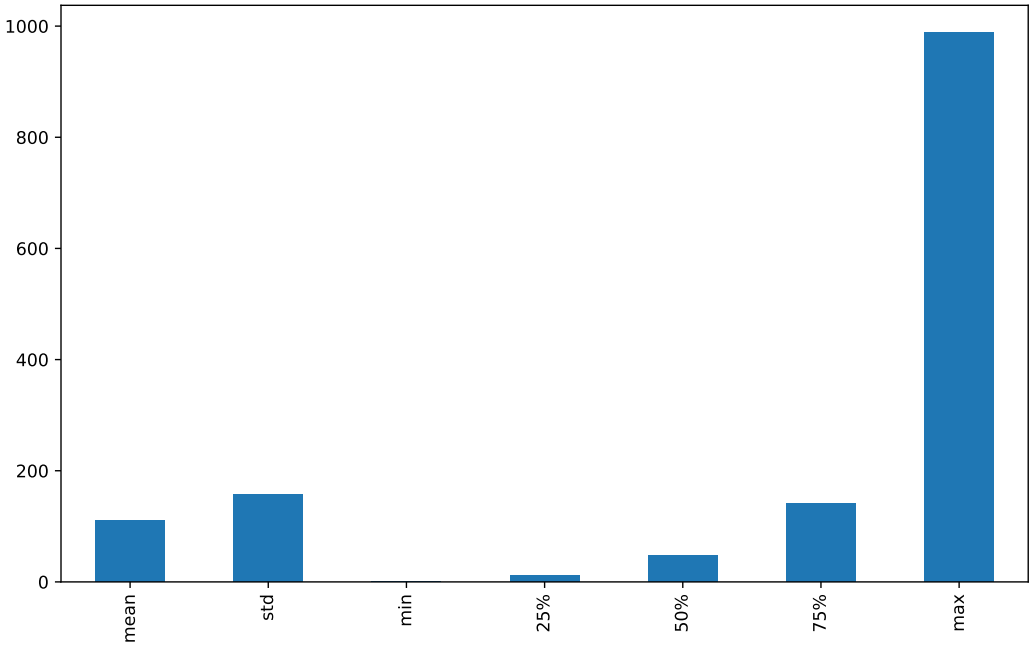


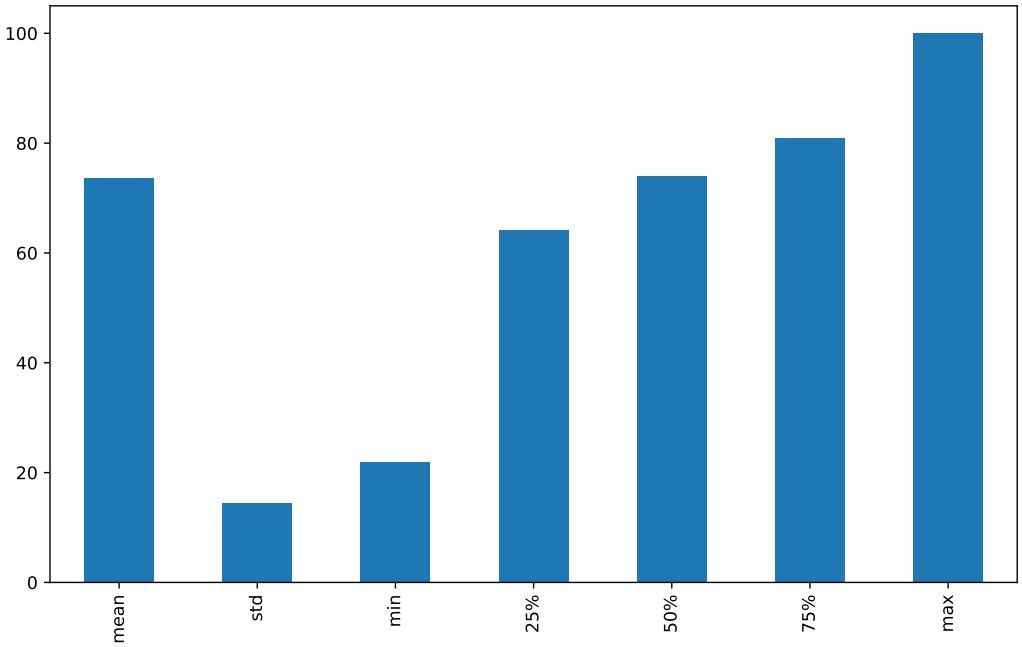

```
import seaborn as sns
import matplotlib.pyplot as plt
describ = df.describe().transpose()



for fila in describ.index:
    linea = describ.loc[fila, ['mean', 'std', 'min', '25%', '50%', '75%', 'max']]
    linea.plot(kind='bar', figsize=(10, 6))
    plt.show()
    plt.close()
```

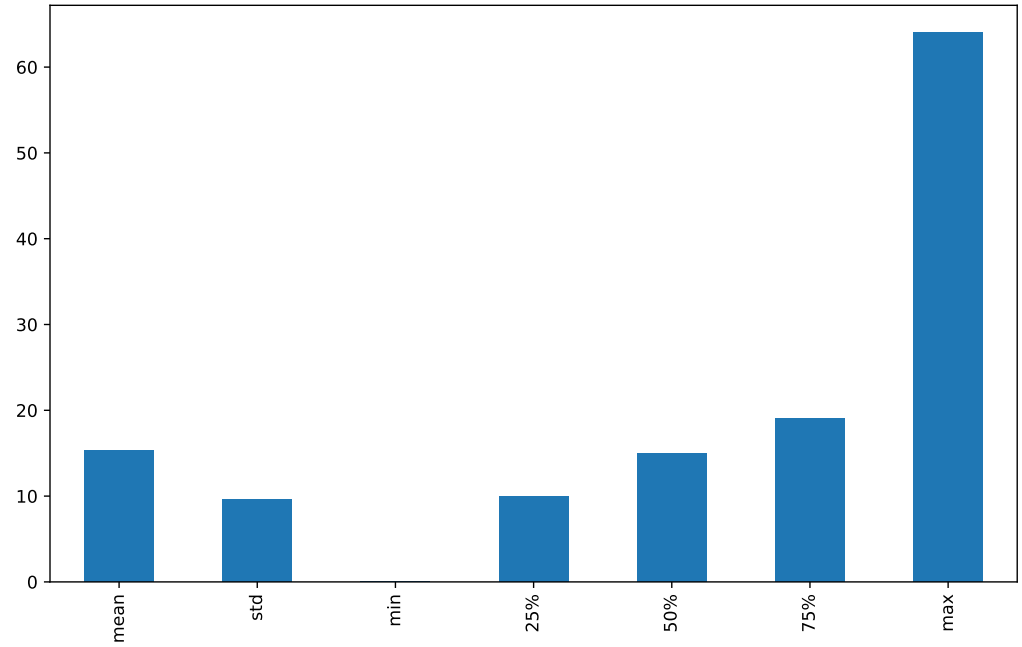

300
250
200
150
100
50
0
mean
std
min
25%
50%
75%
max

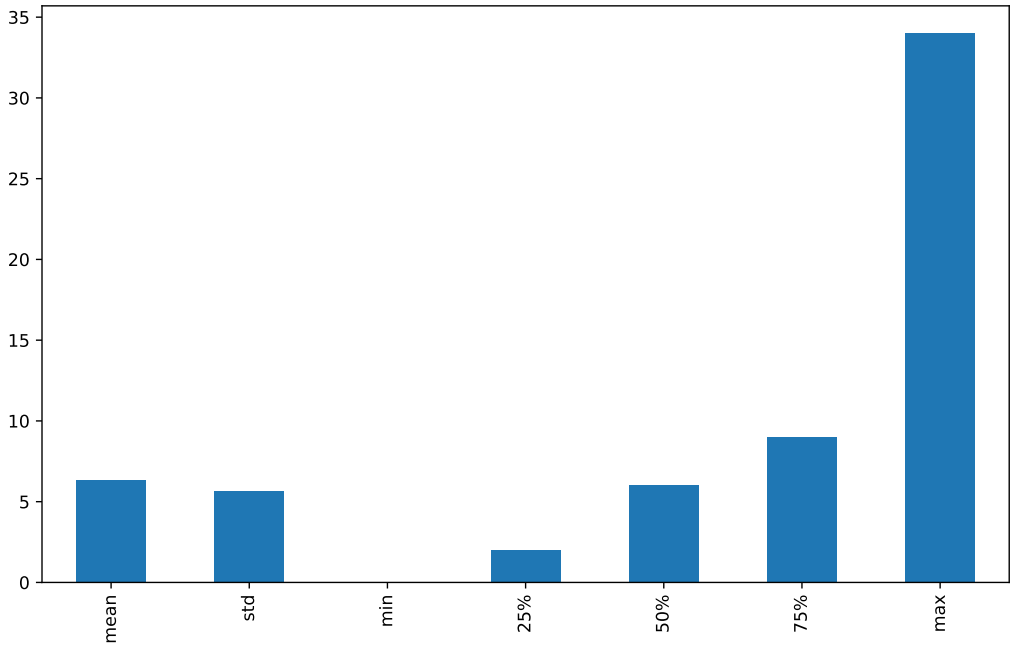

1600
1400
1200
1000
800
600
400
200
0
mean
std
min
25%
50%
75%
max

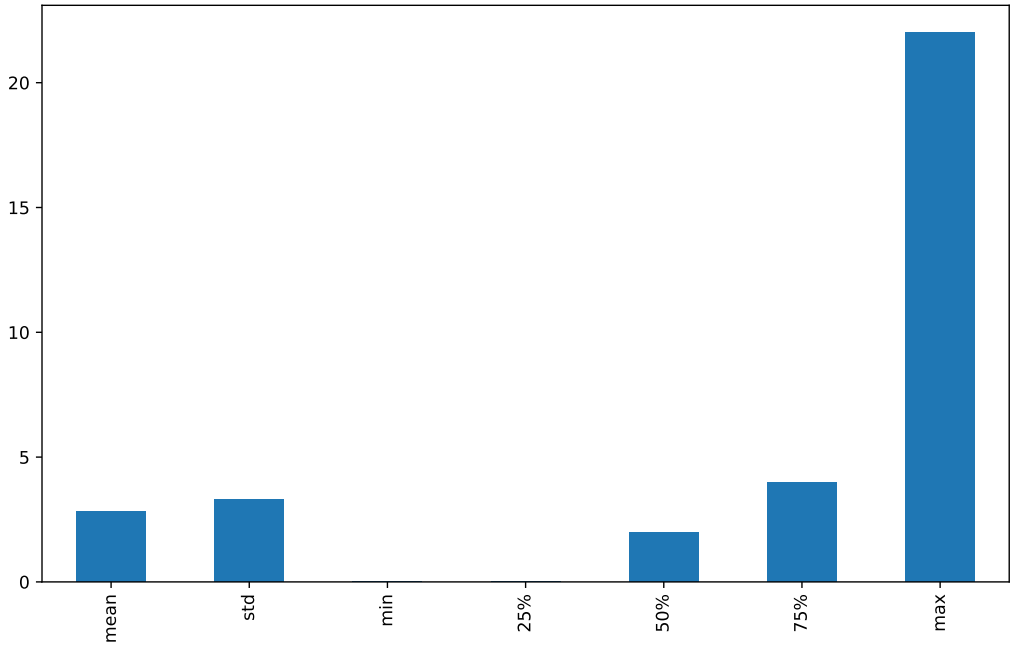

5
4
3
2
1
0
mean
std
min
25%
50%
75%
max

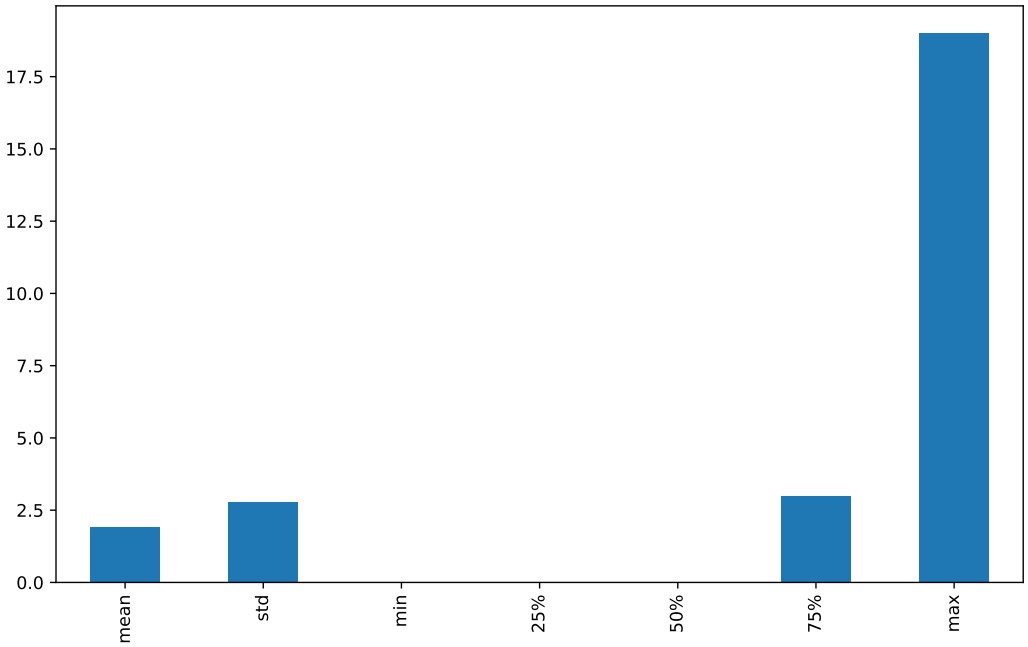

1000
800
600
400
200
0
mean
std
min
25%
50%
75%
max

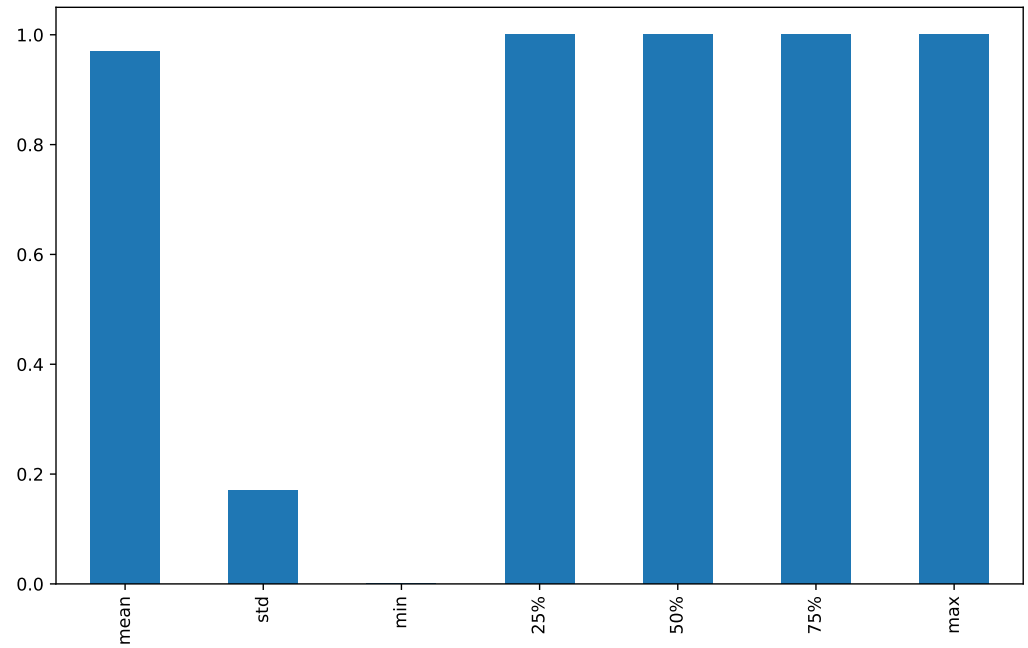

100
80
60
40
20
0
mean
std
min
25%
50%
75%
max

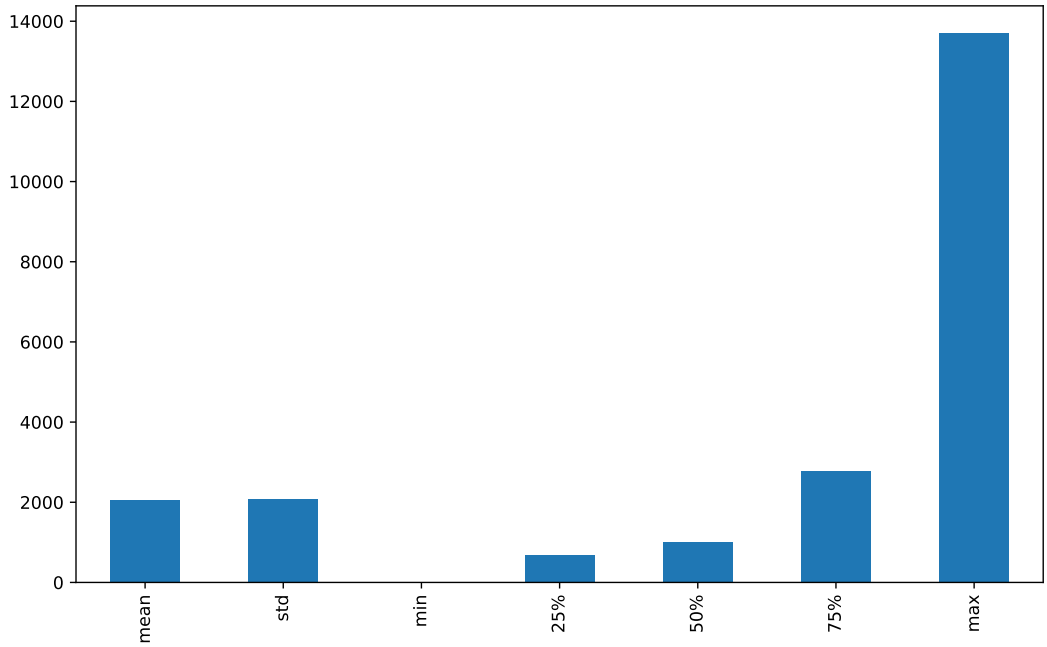

60
50
40
30
20
10
0
mean
std
min
25%
50%
75%
max


35
30
25
20
15
10
5
0
mean
std
min
25%
50%
75%
max


20
15
10
5
0
mean
std
min
25%
50%
75%
max


17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
mean
std
min
25%
50%
75%
max


1.0
0.8
0.6
0.4
0.2
0.0
mean
std
min
25%
50%
75%
max




```python
def categorizar_paginas(pages):

    if isinstance(pages, str) and pages.startswith("Menos de"):
        return pages

    if pd.isnull(pages):
        return "No conocido"

    try:
        pages = int(pages)
    except ValueError:
        return "No conocido"

    if pages < 250:
        return "Menos de 250"
    elif pages < 500:
        return "Menos de 500"
    elif pages < 750:
        return "Menos de 750"
    elif pages < 1000:
        return "Menos de 1000"
    elif pages < 1250:
        return "Menos de 1250"
    elif pages < 1500:
        return "Menos de 1500"
    elif pages < 1750:
        return "Menos de 1750"
    elif pages < 2000:
        return "Menos de 2000"
```

```
    else:
        return "2000 o más"
```

```
df['pages'] = df['pages'].apply(categorizar_paginas)
df['pages'].info()
```

```
## <class 'pandas.core.series.Series'>
## Index: 502 entries, 0 to 636
## Series name: pages
## Non-Null Count  Dtype
## --------------  -----
## 502 non-null    object
## dtypes: object(1)
## memory usage: 7.8+ KB
```

```
pd.crosstab(index=df['pages'], columns="count")
```

```
## col_0          count
## pages
## Menos de 1000     26
## Menos de 1250      5
## Menos de 1500      3
## Menos de 1750      2
## Menos de 250     118
## Menos de 500     251
## Menos de 750      97
```

Tabla de contingencia de la variable pages

```
columnas = df.select_dtypes(include=['float64', 'int64'])
tabla_correlacion = columnas.corr()
tabla_correlacion.to_csv('tabla_correlacion.csv', index=True)
```

```
correlacion <- read.csv('tabla_correlacion.csv')
print(correlacion)
```

```
##              X       price  avg_reviews    n_reviews       star5        star4
## 1        price  1.00000000 -0.037447947 -0.151332390 -0.01276644 -0.003307460
## 2  avg_reviews -0.03744795  1.000000000  0.119146868  0.02485325 -0.013744726
## 3    n_reviews -0.15133239  0.119146868  1.000000000  0.02382599 -0.007448468
## 4        star5 -0.01276644  0.024853246  0.023825992  1.00000000 -0.804588825
## 5        star4 -0.00330746 -0.013744726 -0.007448468 -0.80458882  1.000000000
## 6        star3  0.01922052 -0.054862419 -0.047751883 -0.69407243  0.267734136
## 7        star2  0.05616970  0.020762513 -0.041069353 -0.54472248  0.139050645
## 8        star1 -0.01968884  0.006135908  0.028296080 -0.37979651  0.037796185
## 9       ingles  0.08527829 -0.033360737  0.001483131 -0.05054941  0.056244390
## 10    Peso (g) -0.17547379  0.010113411  0.017273497 -0.06429064  0.100655618
##           star3       star2        star1       ingles     Peso..g.
## 1   0.019220522  0.05616970 -0.019688836  0.085278286 -0.175473792
## 2  -0.054862419  0.02076251  0.006135908 -0.033360737  0.010113411
```

```
## 3  -0.047751883 -0.04106935  0.028296080  0.001483131  0.017273497
## 4  -0.694072433 -0.54472248 -0.379796513 -0.050549408 -0.064290639
## 5   0.267734136  0.13905065  0.037796185  0.056244390  0.100655618
## 6   1.000000000  0.39654430  0.205222482  0.019311861  0.001986655
## 7   0.396544302  1.00000000  0.365906498  0.018389288 -0.025260628
## 8   0.205222482  0.36590650  1.000000000  0.024972770  0.003302209
## 9   0.019311861  0.01838929  0.024972770  1.000000000 -0.044667374
## 10  0.001986655 -0.02526063  0.003302209 -0.044667374  1.000000000
```

Bueno, parece que cuando el precio es mas alto, tiene menos correlaciones y que el peso afecta al precio tambien. La media de reviews tiene una pequena correlacion con el numero de reviews. Hay una fuerte correlacion negativa, entre 4 y 5 estrellas, lo que viene a decir, que un producto esta en una u otra proporcion de estrellas, pero no en ambas. Esto es interesante para saber cuales son los mejores productos y ver su distribucion no solamente cuando es 5 estrellas, sino que pasa cuando son 3, 2 o 1 estrellas, por ejemplo.

```
df.to_csv('df.csv', index=True)
```

Cambioa R, ya que tengo problemas con la visualizacion

```
df <- read.csv('df.csv')
```

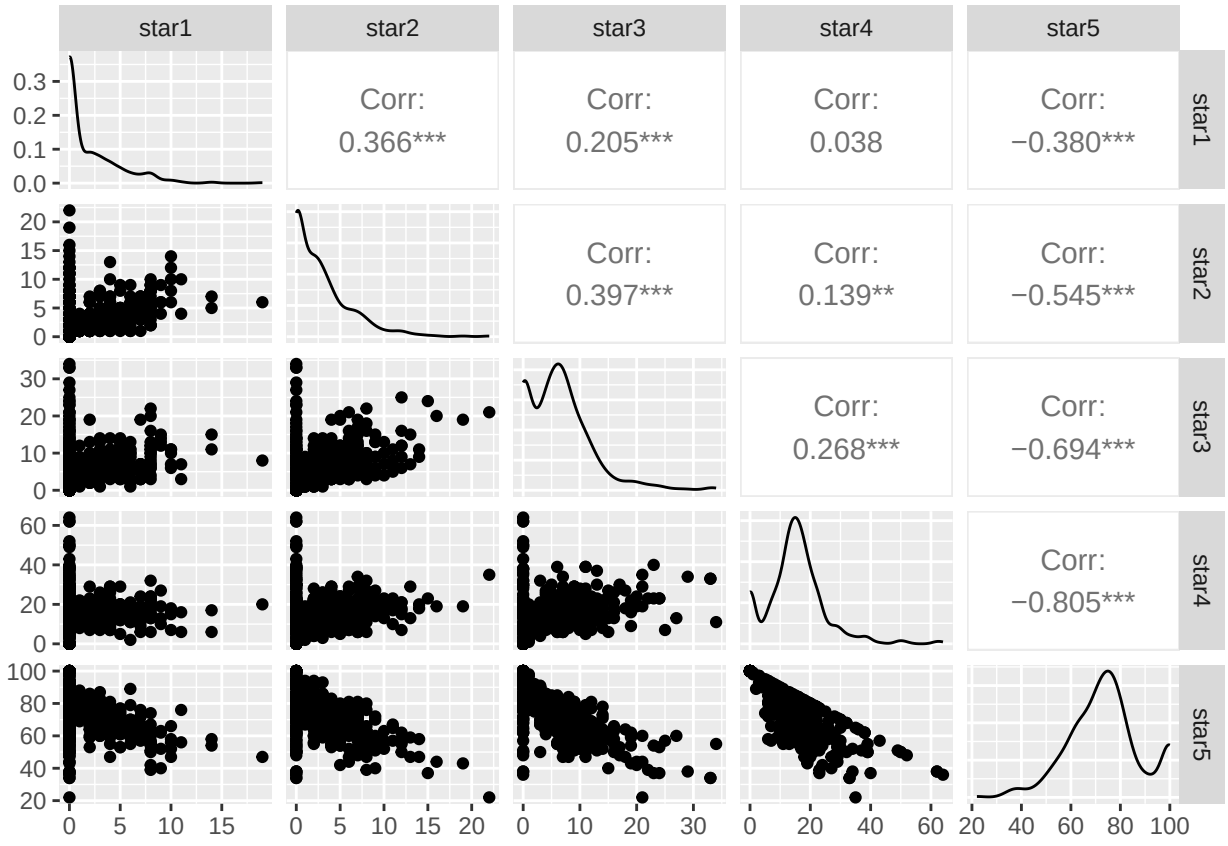
```
if (!requireNamespace("GGally", quietly = TRUE)) {
  install.packages("GGally")
}
```

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

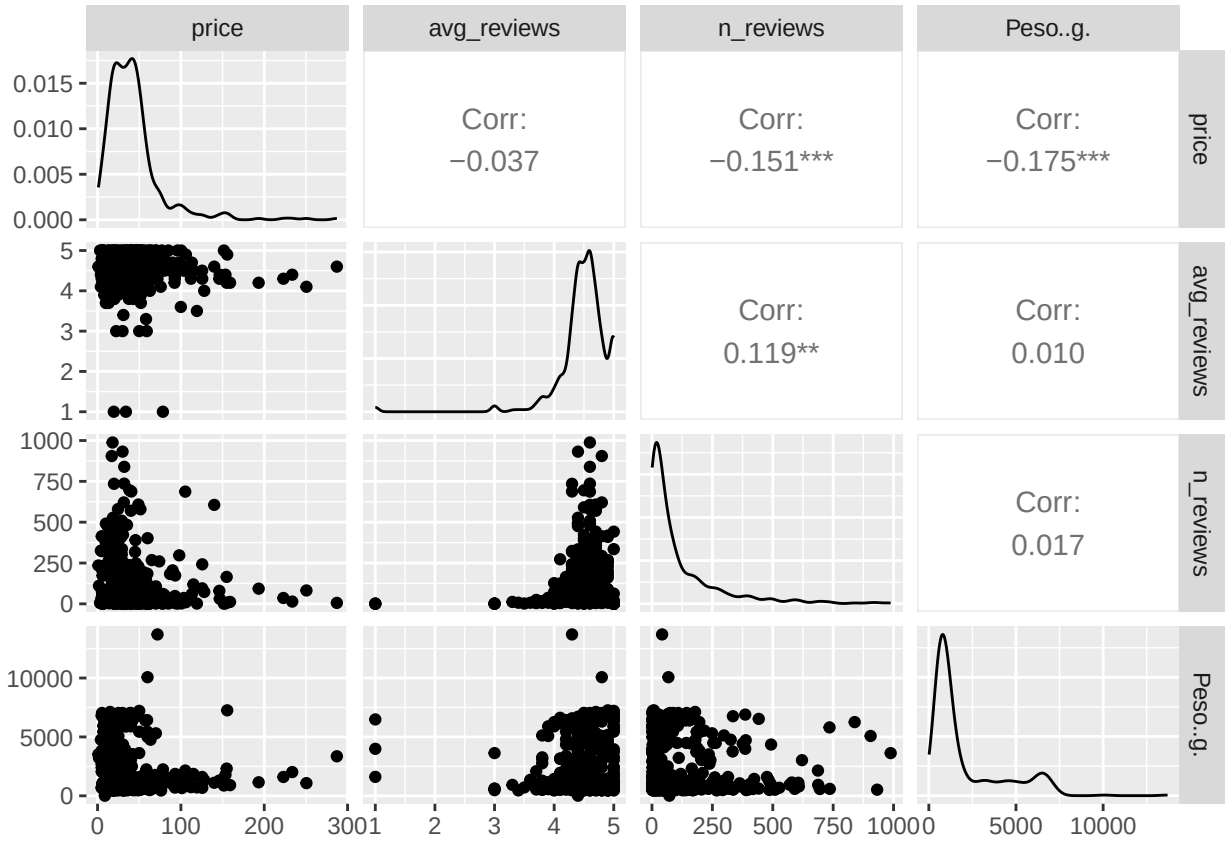
```
library(GGally)
```

```
## Loading required package: ggplot2
```

```
library(ggplot2)
star <- df[, c("star1", "star2", "star3", "star4", "star5")]

ggpairs(star)
```




```
otras <- df[, c("price", "avg_reviews", "n_reviews", "Peso..g.")]

ggpairs(otras)
```



Ahora voy a aplicar kmeans, primero escalo y busco el numero de clusters optimo

```
summary(df)
```

```
##        X            title              author             price
##  Min.   :  0.0   Length:502         Length:502         Min.   :  0.99
##  1st Qu.:152.2   Class :character   Class :character   1st Qu.: 21.12
##  Median :297.5   Mode  :character   Mode  :character   Median : 36.99
##  Mean   :307.7                                         Mean   : 42.32
##  3rd Qu.:468.5                                         3rd Qu.: 49.99
##  Max.   :636.0                                         Max.   :287.14
##  price..including.used.books.    pages           avg_reviews
##  Length:502                   Length:502         Min.   :1.000
##  Class :character             Class :character   1st Qu.:4.400
##  Mode  :character             Mode  :character   Median :4.500
##                                                  Mean   :4.486
##                                                  3rd Qu.:4.700
##                                                  Max.   :5.000
##    n_reviews          star5            star4           star3
##  Min.   :  1.0   Min.   : 22.00   Min.   : 0.00   Min.   : 0.000
##  1st Qu.: 12.0   1st Qu.: 64.25   1st Qu.:10.00   1st Qu.: 2.000
##  Median : 47.7   Median : 74.00   Median :15.00   Median : 6.000
##  Mean   :109.8   Mean   : 73.62   Mean   :15.28   Mean   : 6.353
##  3rd Qu.:142.0   3rd Qu.: 81.00   3rd Qu.:19.00   3rd Qu.: 9.000
##  Max.   :988.0   Max.   :100.00   Max.   :64.00   Max.   :34.000
##      star2            star1           ingles        publisher
```

```
##  Min.   : 0.000   Min.   : 0.00   Min.   :0.0000   Length:502
##  1st Qu.: 0.000   1st Qu.: 0.00   1st Qu.:1.0000   Class :character
##  Median : 2.000   Median : 0.00   Median :1.0000   Mode  :character
##  Mean   : 2.813   Mean   : 1.93   Mean   :0.9701
##  3rd Qu.: 4.000   3rd Qu.: 3.00   3rd Qu.:1.0000
##  Max.   :22.000   Max.   :19.00   Max.   :1.0000
##     Peso..g.
##  Min.   :   15.88
##  1st Qu.:  680.39
##  Median : 1018.31
##  Mean   : 2047.91
##  3rd Qu.: 2782.56
##  Max.   :13698.48
```

```
columnas_numericas <- (df[sapply(df, is.numeric)])
datos_escalados <- as.data.frame(scale(columnas_numericas))
colnames(datos_escalados) <- colnames(columnas_numericas)
head(datos_escalados)
```

```
##           X      price avg_reviews   n_reviews       star5       star4
## 1 -1.685786 -1.0696769  -0.2046753 -0.55367240 -1.28307343  2.46484387
## 2 -1.680307 -0.2587316  -0.4414196  0.09065168 -0.86970118  0.49040188
## 3 -1.674827 -0.3083445   0.5055575 -0.63660520  0.92157857 -0.23702412
## 4 -1.669348  0.3507553   0.2688132 -0.35335779  0.09483407  0.07472988
## 5 -1.663869 -0.5362631  -1.3883968 -0.62384631 -1.48975955  0.17864788
## 6 -1.652910  1.4335192   0.5055575  0.60738683  0.09483407  0.28256588
##        star3      star2       star1    ingles   Peso..g.
## 1 -0.0625906 -0.8507507 -0.69126041 0.1753267 -0.4313758
## 2  0.4699599  0.3590987  1.45742521 0.1753267 -0.5552543
## 3 -1.1276916 -0.8507507 -0.69126041 0.1753267 -0.6769595
## 4 -0.2401074 -0.2458260  0.02496813 0.1753267 -0.6617464
## 5  0.6474767  2.1738729  2.88988229 0.1753267 -0.6986926
## 6 -0.7726579 -0.2458260  0.02496813 0.1753267 -0.5139615
```

```
quitar <- grep("star", names(datos_escalados))
datos_escalados <- datos_escalados[, -quitar]
head(datos_escalados)
```

```
##           X      price avg_reviews   n_reviews    ingles   Peso..g.
## 1 -1.685786 -1.0696769  -0.2046753 -0.55367240 0.1753267 -0.4313758
## 2 -1.680307 -0.2587316  -0.4414196  0.09065168 0.1753267 -0.5552543
## 3 -1.674827 -0.3083445   0.5055575 -0.63660520 0.1753267 -0.6769595
## 4 -1.669348  0.3507553   0.2688132 -0.35335779 0.1753267 -0.6617464
## 5 -1.663869 -0.5362631  -1.3883968 -0.62384631 0.1753267 -0.6986926
## 6 -1.652910  1.4335192   0.5055575  0.60738683 0.1753267 -0.5139615
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

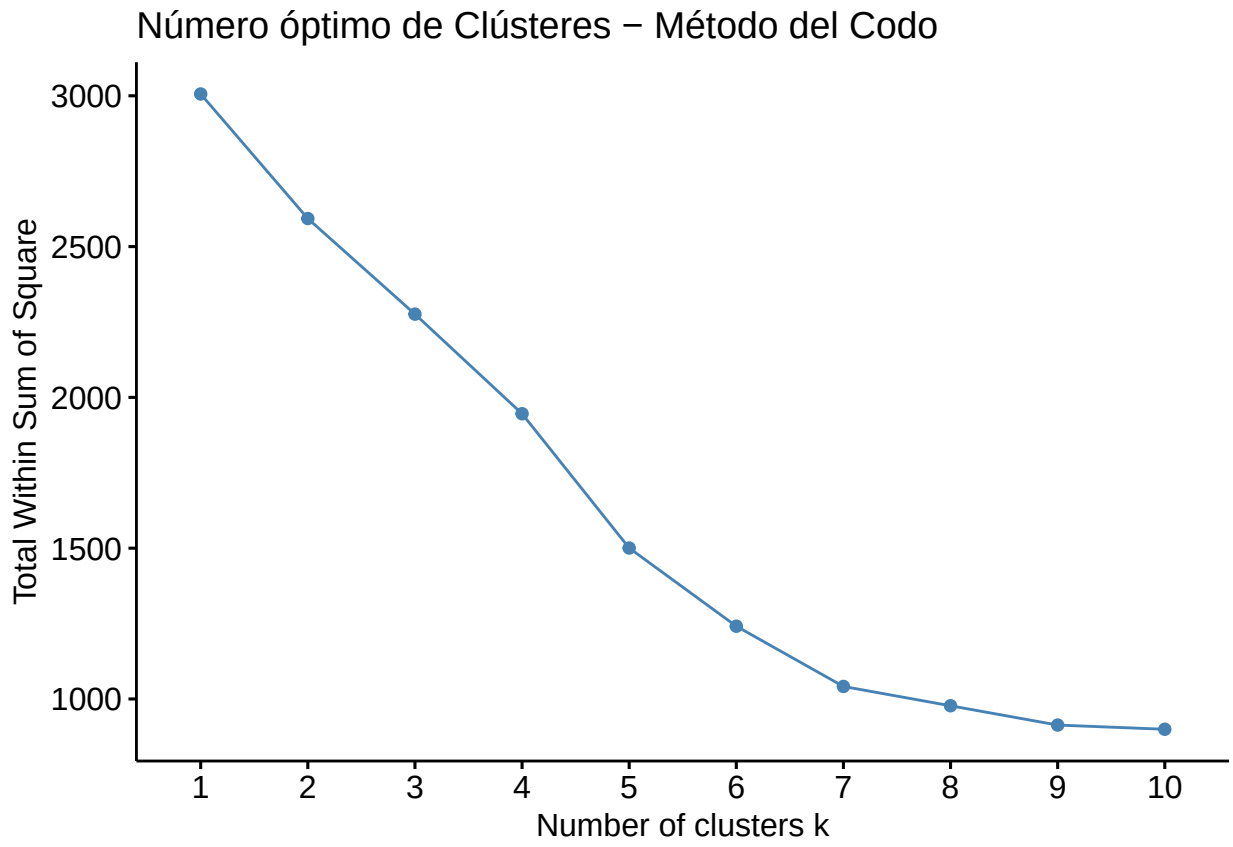
```
## The following objects are masked from 'package:stats':
```

```
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

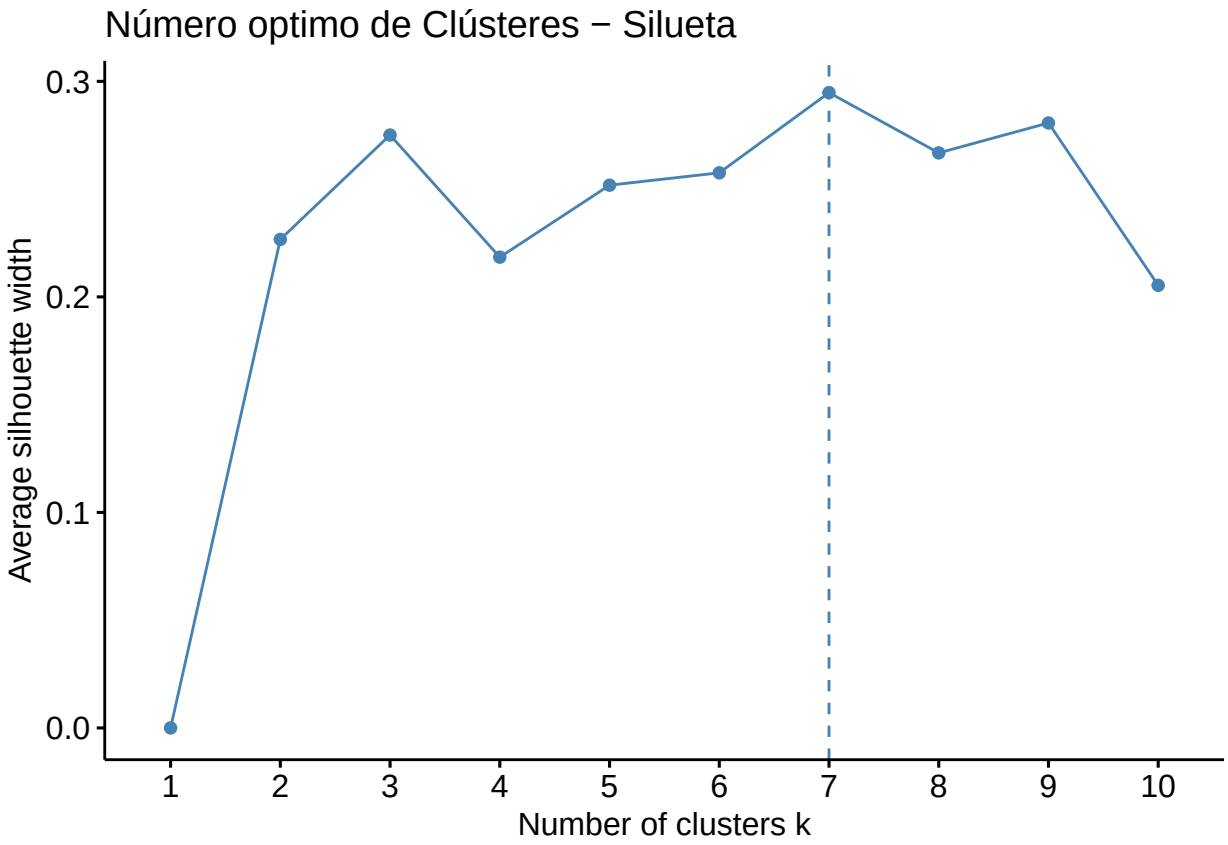
```
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
fviz_nbclust(datos_escalados, kmeans, method = "wss") +
  ggtitle("Número óptimo de Clústeres - Método del Codo")
```

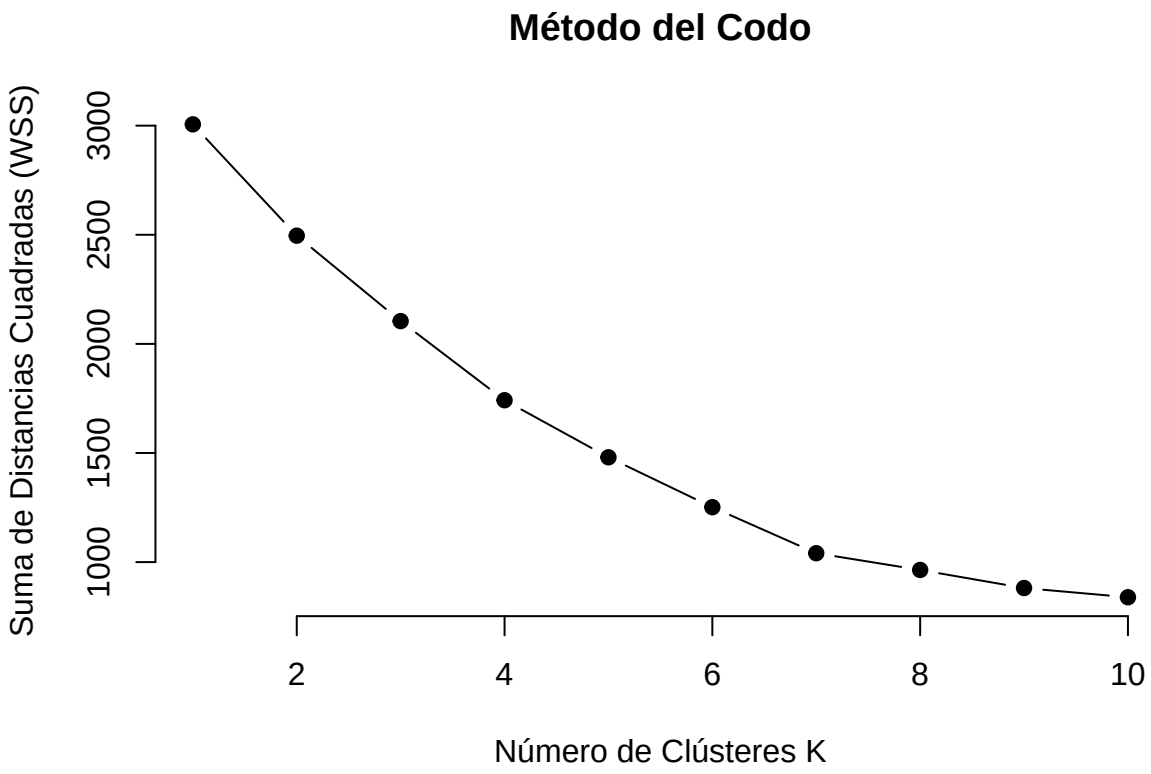



```
fviz_nbclust(datos_escalados, kmeans, method = "silhouette") +
  ggtitle("Número optimo de Clústeres - Silueta")
```



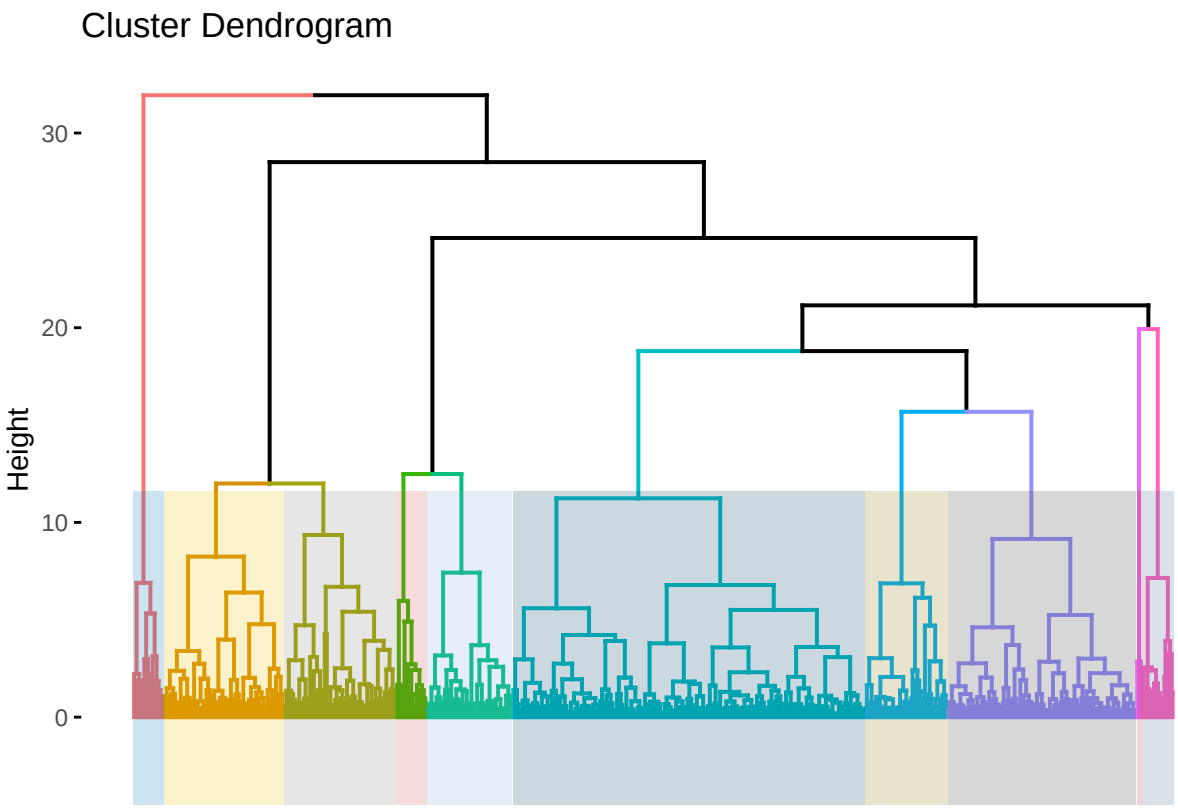
En el metodo siluette esta marcado y recomienda 7, en el codo es mas confuso pero podriamos seleccionar 3 o 4, o a partir de 7

```
wss <- sapply(1:10, function(k) {
  kmeans(datos_escalados, centers = k, nstart = 10)$tot.withinss
})

plot(1:10, wss, type = "b", pch = 19, frame = FALSE,
     xlab = "Número de Clústeres K",
     ylab = "Suma de Distancias Cuadradas (WSS)",
     main = "Método del Codo")
```



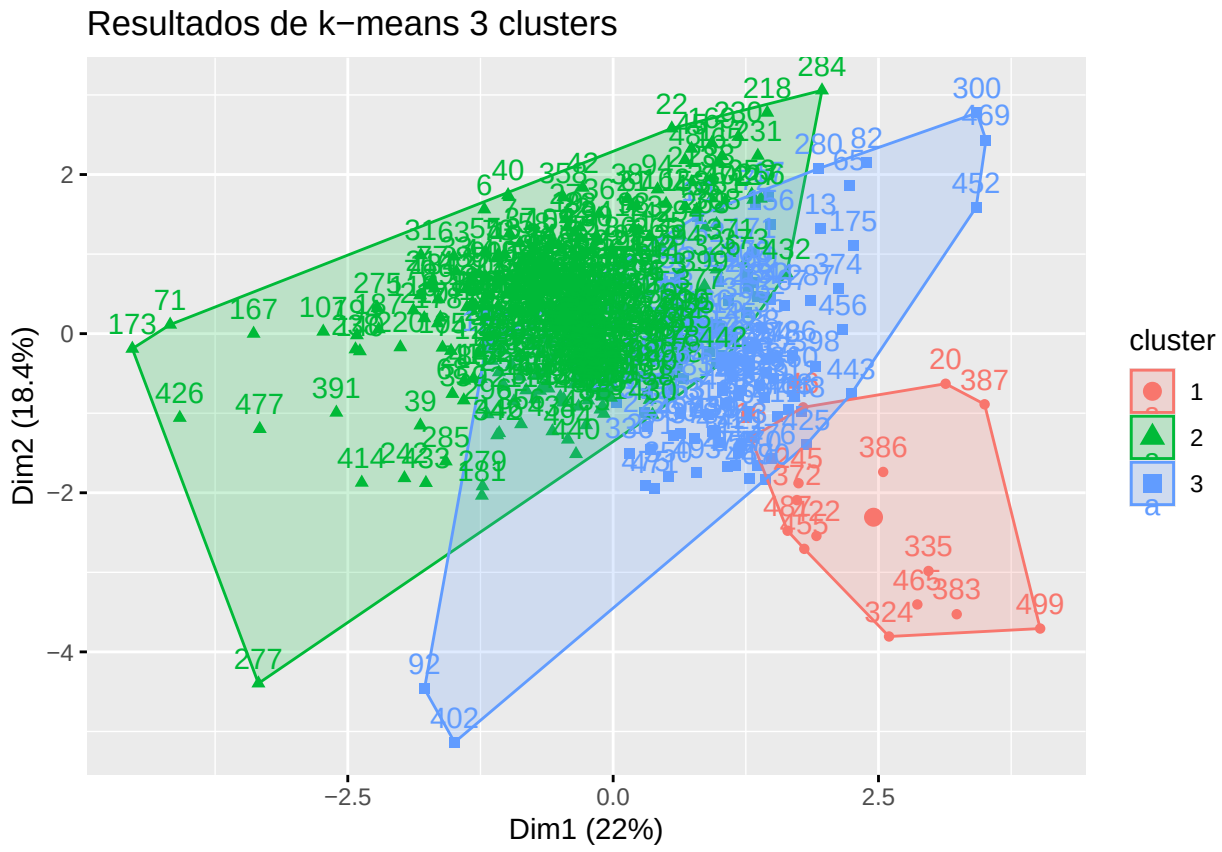

```
if (!requireNamespace("cluster", quietly = TRUE)) {
  install.packages("cluster")
}
library(cluster)

#datos_escalados <- scale(df[sapply(df, is.numeric)])

dist_matrix <- dist(datos_escalados)

hclust_result <- hclust(dist_matrix, method = "ward.D2")

fviz_dend(hclust_result,
          k = 10,
          rect = TRUE,
          rect_fill = TRUE,
          rect_border = "jco",
          show_labels = FALSE)
```

```
## Warning: The ‘<scale>‘ argument of ‘guides()‘ cannot be ‘FALSE‘. Use "none" instead as
## of ggplot2 3.3.4.
## i The deprecated feature was likely used in the factoextra package.
##   Please report the issue at <https://github.com/kassambara/factoextra/issues>.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

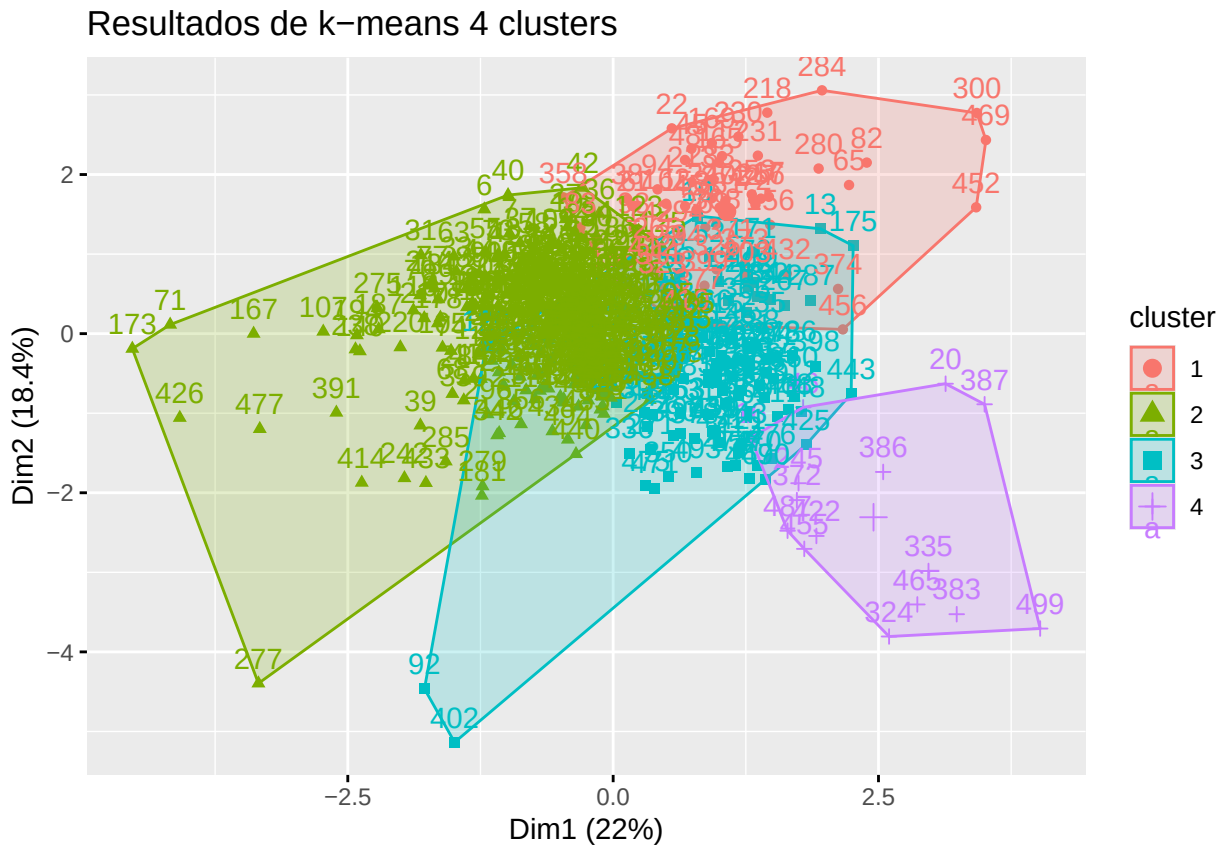


Si vien la clasificacion jerarquica ascendente (yo lo aprendi en frances y creo que se llama igual, CAH). No esta relacionada con Kmeans, puede servir tambien para hacerse una idea visual de cuantos clusters podemos utilizar viendo la distancia entre los clusters.

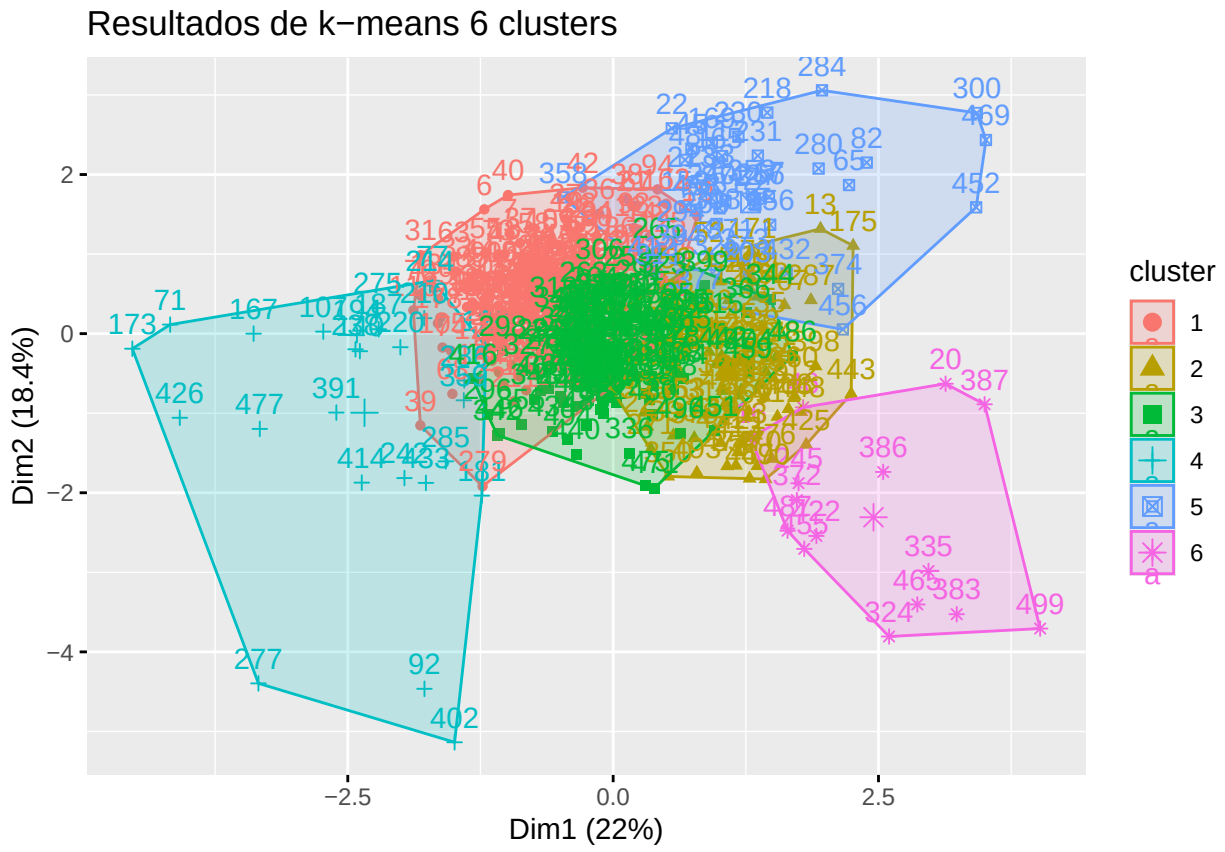
```
set.seed(123)
resultado_kmeans <- kmeans(datos_escalados, centers = 3, nstart = 25)
fviz_cluster(resultado_kmeans, data = datos_escalados) +
  ggtitle("Resultados de k-means 3 clusters")
```



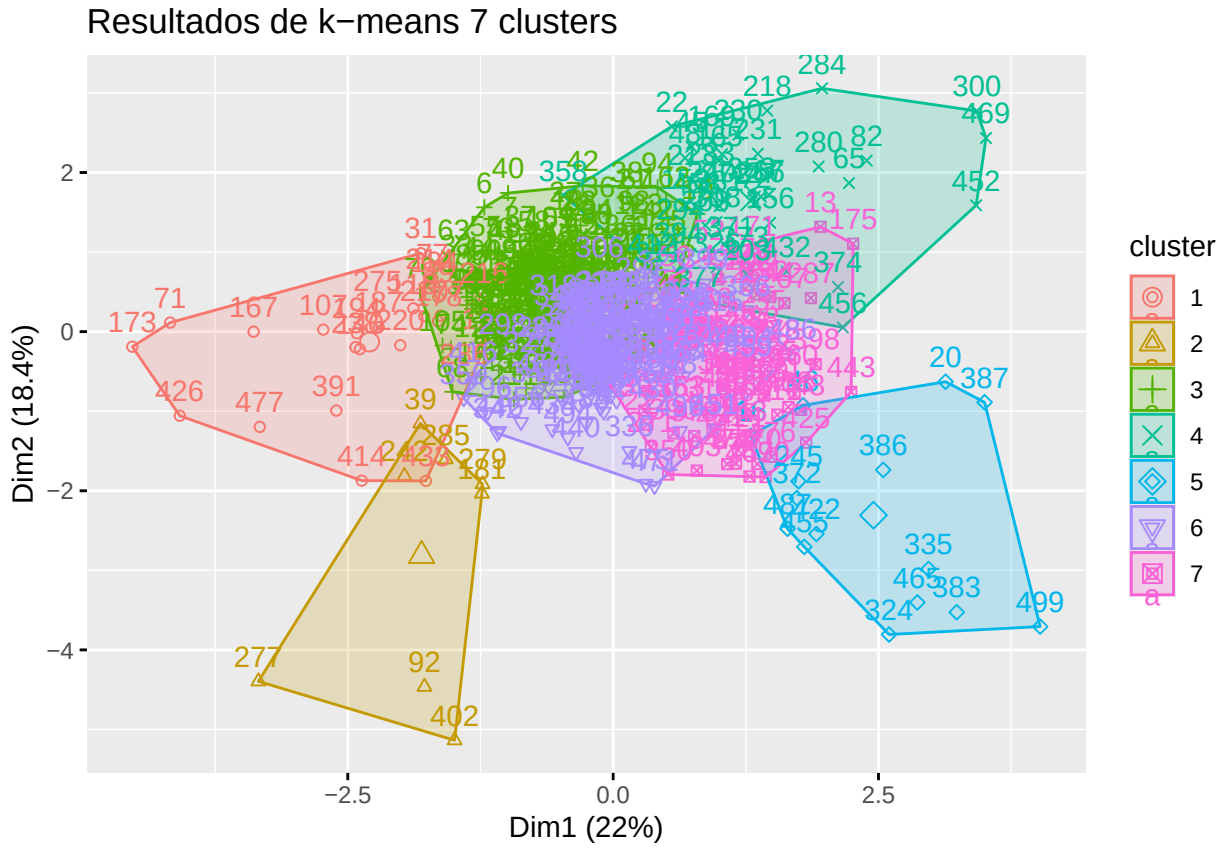

```
resultado_kmeans4 <- kmeans(datos_escalados, centers = 4, nstart = 25)
fviz_cluster(resultado_kmeans4, data = datos_escalados) +
  ggtitle("Resultados de k-means 4 clusters")
```




```
resultado_kmeans6 <- kmeans(datos_escalados, centers = 6, nstart = 25)
fviz_cluster(resultado_kmeans6, data = datos_escalados) +
  ggtitle("Resultados de k-means 6 clusters")
```




```
resultado_kmeans7 <- kmeans(datos_escalados, centers = 7, nstart = 25)
fviz_cluster(resultado_kmeans7, data = datos_escalados) +
  ggtitle("Resultados de k-means 7 clusters")
```




```
summary(df)
```

```
##        X            title              author              price          
##  Min.   :  0.0   Length:502         Length:502         Min.   :  0.99  
##  1st Qu.:152.2   Class :character   Class :character   1st Qu.: 21.12  
##  Median :297.5   Mode  :character   Mode  :character   Median : 36.99  
##  Mean   :307.7                                         Mean   : 42.32  
##  3rd Qu.:468.5                                         3rd Qu.: 49.99  
##  Max.   :636.0                                         Max.   :287.14  
##  price..including.used.books.    pages            avg_reviews    
##  Length:502                   Length:502         Min.   :1.000  
##  Class :character             Class :character   1st Qu.:4.400  
##  Mode  :character             Mode  :character   Median :4.500  
##                                                  Mean   :4.486  
##                                                  3rd Qu.:4.700  
##                                                  Max.   :5.000  
##    n_reviews          star5             star4           star3        
##  Min.   :  1.0   Min.   : 22.00   Min.   : 0.00   Min.   : 0.000  
##  1st Qu.: 12.0   1st Qu.: 64.25   1st Qu.:10.00   1st Qu.: 2.000  
##  Median : 47.7   Median : 74.00   Median :15.00   Median : 6.000  
##  Mean   :109.8   Mean   : 73.62   Mean   :15.28   Mean   : 6.353  
##  3rd Qu.:142.0   3rd Qu.: 81.00   3rd Qu.:19.00   3rd Qu.: 9.000  
##  Max.   :988.0   Max.   :100.00   Max.   :64.00   Max.   :34.000  
##      star2             star1            ingles        publisher        
##  Min.   : 0.000   Min.   : 0.00   Min.   :0.0000   Length:502  
```

```
##  1st Qu.: 0.000   1st Qu.: 0.00   1st Qu.:1.0000   Class :character
##  Median : 2.000   Median : 0.00   Median :1.0000   Mode  :character
##  Mean   : 2.813   Mean   : 1.93   Mean   :0.9701
##  3rd Qu.: 4.000   3rd Qu.: 3.00   3rd Qu.:1.0000
##  Max.   :22.000   Max.   :19.00   Max.   :1.0000
##     Peso..g.
##  Min.   :   15.88
##  1st Qu.:  680.39
##  Median : 1018.31
##  Mean   : 2047.91
##  3rd Qu.: 2782.56
##  Max.   :13698.48
```

```
summary(datos_escalados)
```

```
##        X                 price          avg_reviews         n_reviews
##  Min.   :-1.68579   Min.   :-1.2429   Min.   :-8.25398   Min.   :-0.6940
##  1st Qu.:-0.85155   1st Qu.:-0.6375   1st Qu.:-0.20467   1st Qu.:-0.6238
##  Median :-0.05568   Median :-0.1603   Median : 0.03207   Median :-0.3961
##  Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.0000
##  3rd Qu.: 0.88129   3rd Qu.: 0.2305   3rd Qu.: 0.50556   3rd Qu.: 0.2055
##  Max.   : 1.79908   Max.   : 7.3612   Max.   : 1.21579   Max.   : 5.6025
##      ingles            Peso..g.
##  Min.   :-5.6923   Min.   :-0.9736
##  1st Qu.: 0.1753   1st Qu.:-0.6552
##  Median : 0.1753   Median :-0.4933
##  Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.1753   3rd Qu.: 0.3520
##  Max.   : 0.1753   Max.   : 5.5822
```

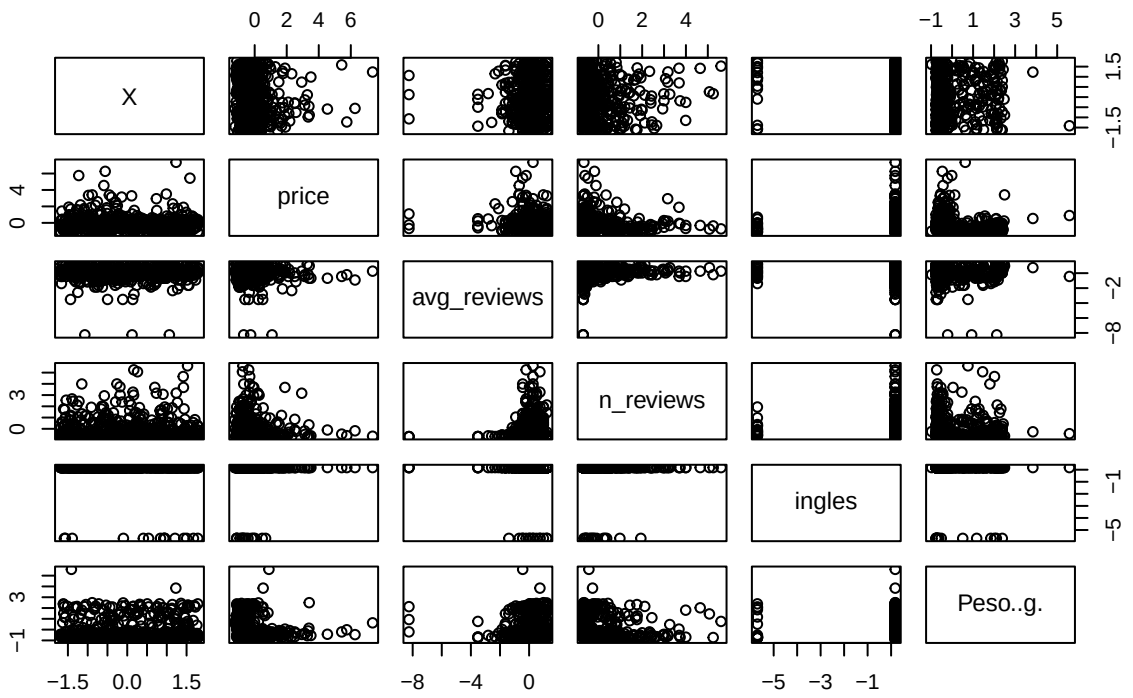
```
library(corrplot)
```

```
## corrplot 0.94 loaded
```

```
res <- cor(datos_escalados, use = "complete.obs")
corrplot(res, method = "color", tl.col = "black", tl.srt = 30, order = "AOE",
         number.cex = 0.75, addCoef.col = "black")
```


```
pairs(datos_escalados, main = "Datos Escalados")
```

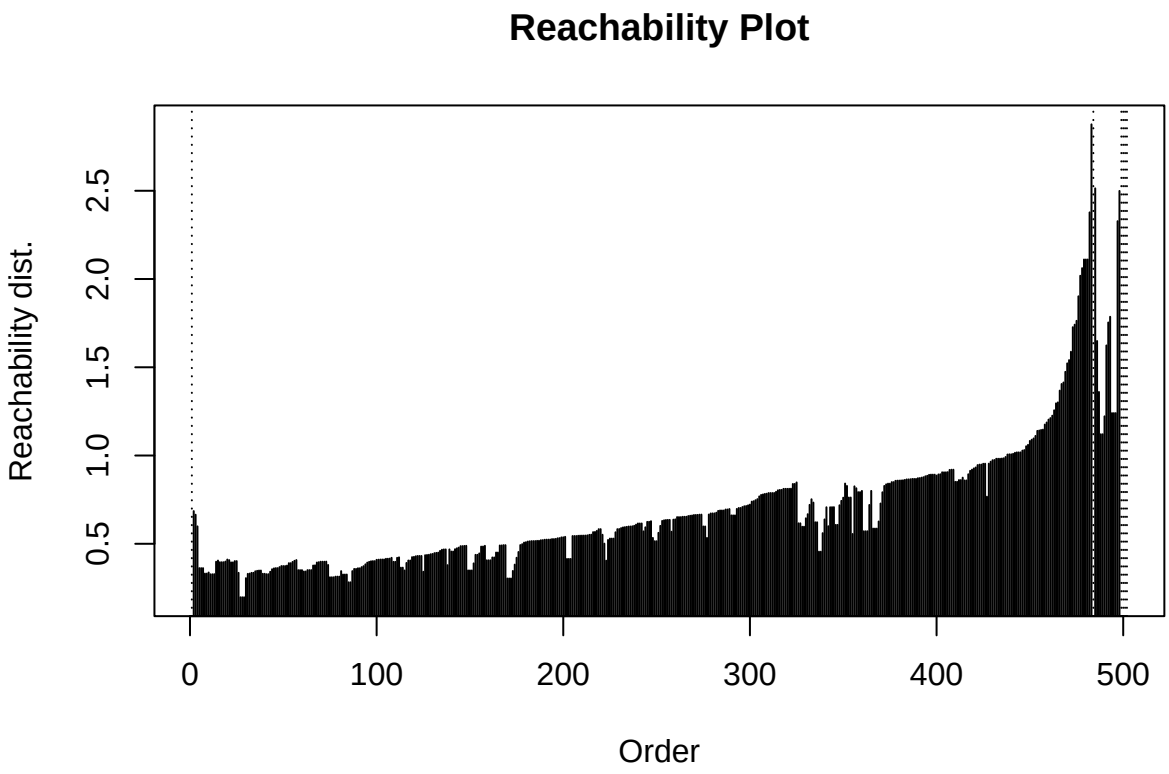
Datos Escalados



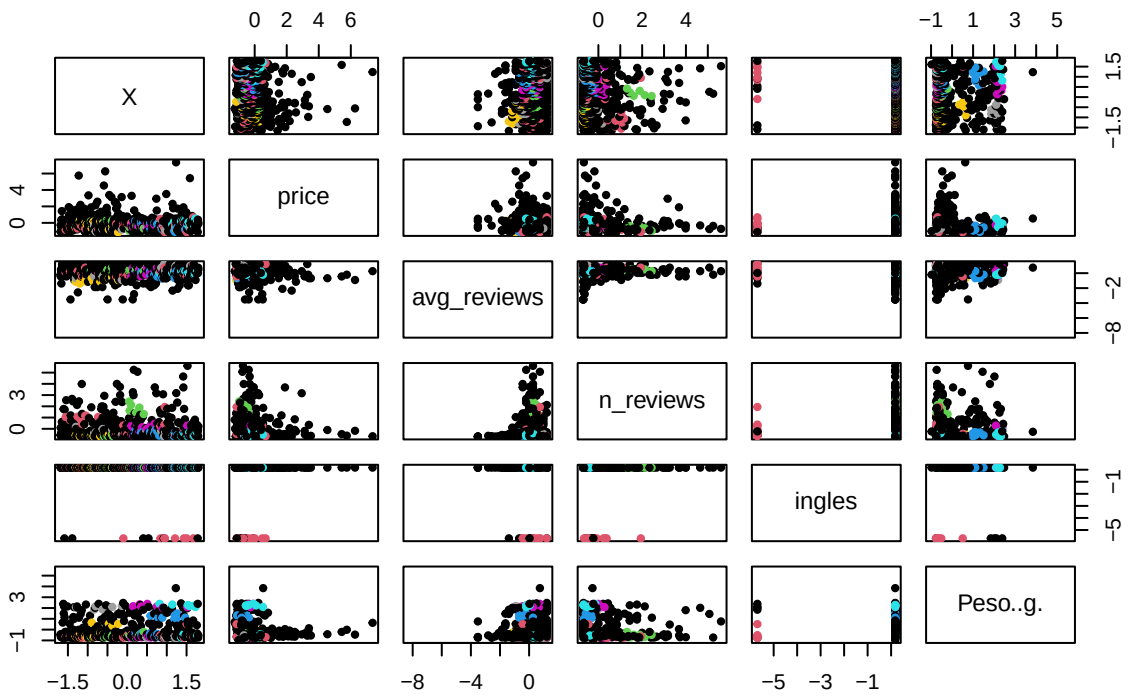

```
if (!requireNamespace("dbscan", quietly = TRUE)) {
  install.packages("dbscan")
}
library(dbscan)
```

```
##
## Attaching package: 'dbscan'

## The following object is masked from 'package:stats':
##
##     as.dendrogram
```

```
#datos_escalados <- scale(df[sapply(df, is.numeric)])

optics_result <- optics(datos_escalados, eps = 3, minPts = 4)

plot(optics_result)
```




```r
clusters_optics <- extractXi(optics_result, xi = 0.03)
plot(datos_escalados, col = clusters_optics$cluster, pch = 20, main = "Clustering OPTICS")
```



No me detecta bien los clusters, pero ha sido util para entender mejor los datos y ver que no hay clusters claros, por lo que kmeans es mejor en este caso, aunque kmeans tampoco tiene un desarrollo tan claro.

```
head(datos_escalados)
```

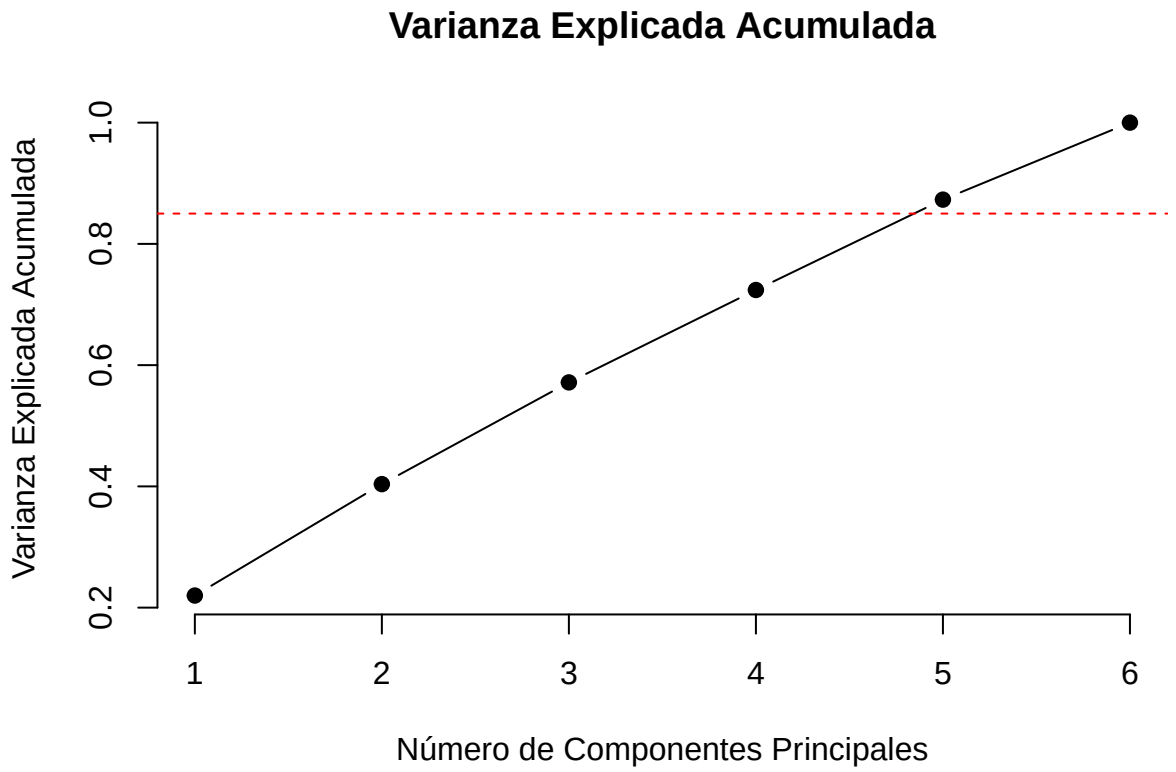
```
##           X      price avg_reviews   n_reviews    ingles   Peso..g.
## 1 -1.685786 -1.0696769  -0.2046753 -0.55367240 0.1753267 -0.4313758
## 2 -1.680307 -0.2587316  -0.4414196  0.09065168 0.1753267 -0.5552543
## 3 -1.674827 -0.3083445   0.5055575 -0.63660520 0.1753267 -0.6769595
## 4 -1.669348  0.3507553   0.2688132 -0.35335779 0.1753267 -0.6617464
## 5 -1.663869 -0.5362631  -1.3883968 -0.62384631 0.1753267 -0.6986926
## 6 -1.652910  1.4335192   0.5055575  0.60738683 0.1753267 -0.5139615
```

Ahora voy a probar a reducir la dimensionalidad, y quizas asi tengamos mejores conclusiones

```
pca_result <- prcomp(datos_escalados, center = TRUE, scale. = TRUE)

varianza_explicada <- pca_result$sdev^2 / sum(pca_result$sdev^2)

varianza_acumulada <- cumsum(varianza_explicada)

print(varianza_acumulada)
```

```
## [1] 0.2199564 0.4037325 0.5715168 0.7239691 0.8730124 1.0000000
```

```r
plot(varianza_acumulada, type = "b", pch = 19, frame = FALSE,
     xlab = "Número de Componentes Principales",
     ylab = "Varianza Explicada Acumulada",
     main = "Varianza Explicada Acumulada")
abline(h = 0.85, col = "red", lty = 2)
```



Voy a hacerlo con 4 componentes, ya que el objetivo del ejercicio es entender la dimensionalidad y no tendria sentido hacerlo con 5 componentes y poder explicar el 100%, aunque podria ser interesante si hubiera sido el 90%, voy a usar el 87%

```r
num_componentes <- 4
df_reducido <- pca_result$x[, 1:num_componentes]
head(df_reducido)
```

```
##             PC1        PC2        PC3        PC4
## [1,] -0.3440605 0.59055293 -1.0154728 -1.3573357
## [2,] -0.6909876 0.85748136 -0.8030565 -1.1766259
## [3,] -0.7520643 0.94414263 -0.2970098 -1.2191406
## [4,] -1.0911491 0.97049241 -0.2039189 -1.1005470
## [5,] -1.1395535 0.02611503 -1.2941592 -1.3848796
## [6,] -1.2136508 1.56259356  0.2085945 -0.7753396
```

Guardo para mas adelante las primeras 4 componentes

```r
pca_result <- prcomp(datos_escalados, center = TRUE, scale. = TRUE)

summary(pca_result)
```

```
## Importance of components:
##                           PC1    PC2    PC3    PC4    PC5    PC6
## Standard deviation     1.149 1.0501 1.0033 0.9564 0.9457 0.8729
## Proportion of Variance 0.220 0.1838 0.1678 0.1525 0.1490 0.1270
## Cumulative Proportion  0.220 0.4037 0.5715 0.7240 0.8730 1.0000
```

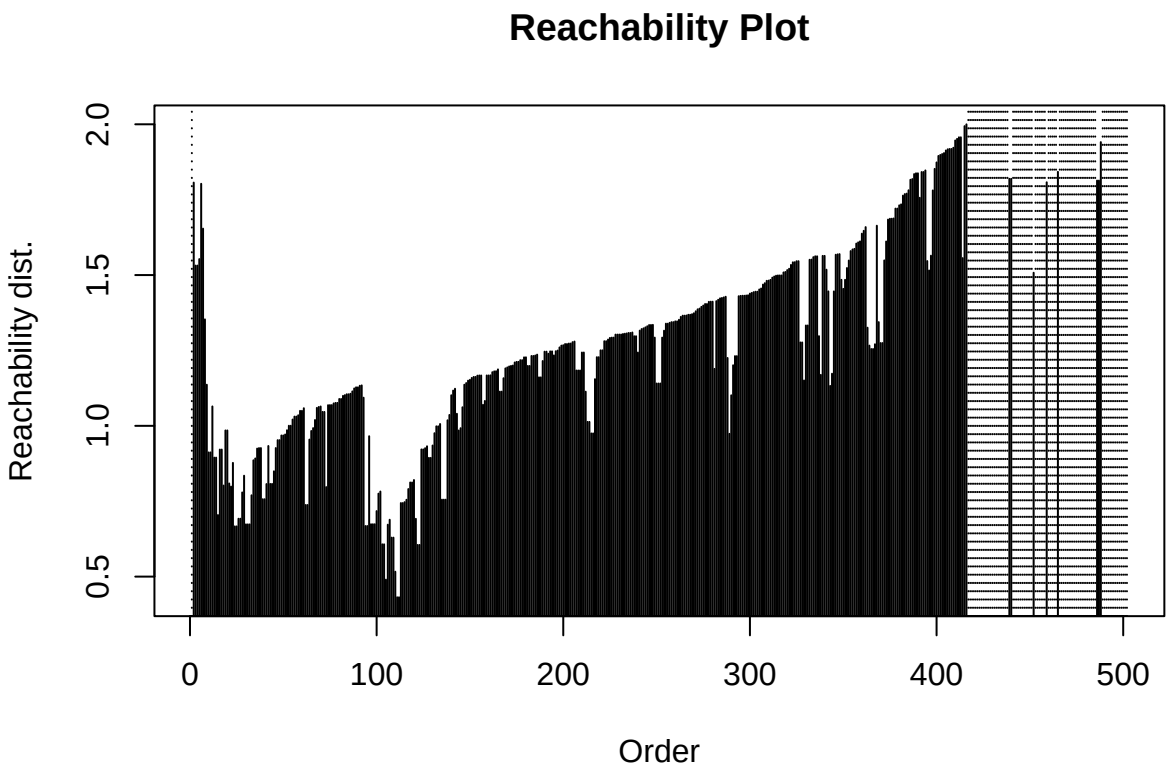
```r
datos_reducidos <- pca_result$x[, 1:4]
```

```r
pca_result <- prcomp(as.data.frame(scale(columnas_numericas)), center = TRUE, scale. = TRUE)

summary(pca_result)
```

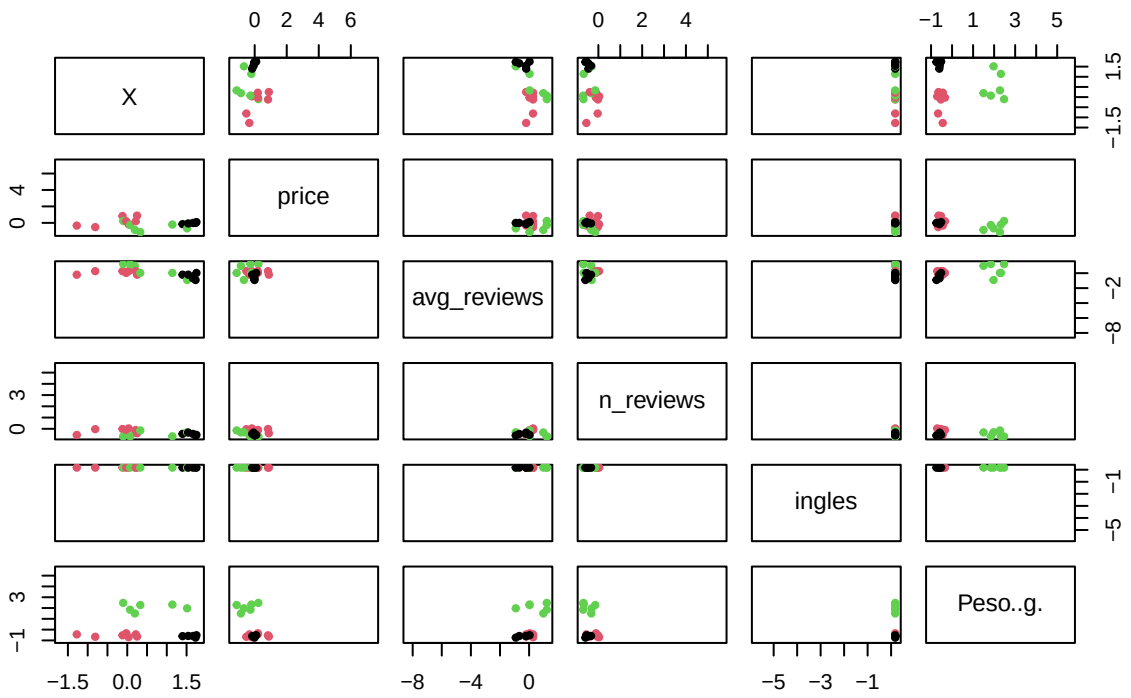
```
## Importance of components:
##                            PC1    PC2    PC3    PC4     PC5     PC6     PC7
## Standard deviation     1.6289 1.1520 1.0856 1.0533 0.98308 0.94981 0.94196
## Proportion of Variance 0.2412 0.1206 0.1071 0.1009 0.08786 0.08201 0.08066
## Cumulative Proportion  0.2412 0.3619 0.4690 0.5698 0.65770 0.73972 0.82038
##                            PC8     PC9    PC10    PC11
## Standard deviation     0.87343 0.82478 0.72930 0.02871
## Proportion of Variance 0.06935 0.06184 0.04835 0.00007
## Cumulative Proportion  0.88973 0.95157 0.99993 1.00000
```

Si incluyo las estrellas, con 8 componentes tendria la misma informacion que sin incluir las estrellas con 5, lo que quiere decir que 2 variables de estrellas no dan mucha informacion

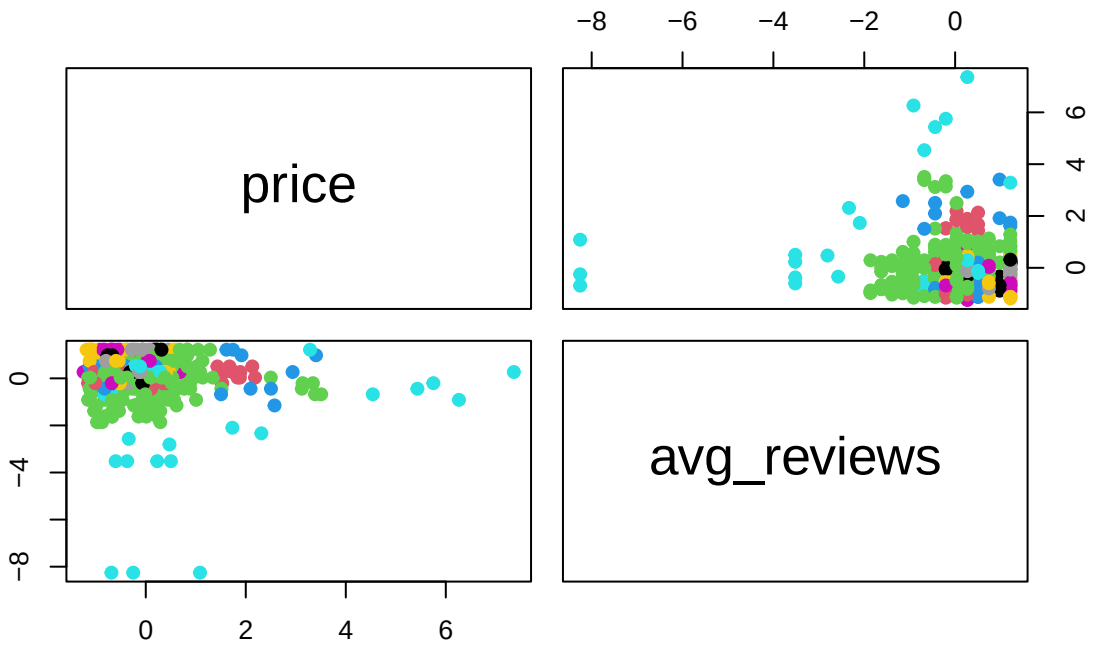
```r
optics_result <- optics(as.data.frame(scale(columnas_numericas)), eps = 2, minPts = 3)

plot(optics_result)
```




```
clusters_optics <- extractXi(optics_result, xi = 0.09)
plot(datos_escalados, col = clusters_optics$cluster, pch = 20, main = "Clustering OPTICS")
```




```
columnas_seleccionadas <- columnas_numericas[, c("price", "avg_reviews")]
datos_escalados_min <- as.data.frame(scale(columnas_seleccionadas))
optics_result <- optics(datos_escalados_min, eps = 20, minPts = 5)

clusters_optics <- extractXi(optics_result, xi = 0.10)
pairs(datos_escalados_min,
      col = clusters_optics$cluster + 1,
      pch = 19,
      main = "Clustering OPTICS - price, avg_reviews y n_reviews")
```



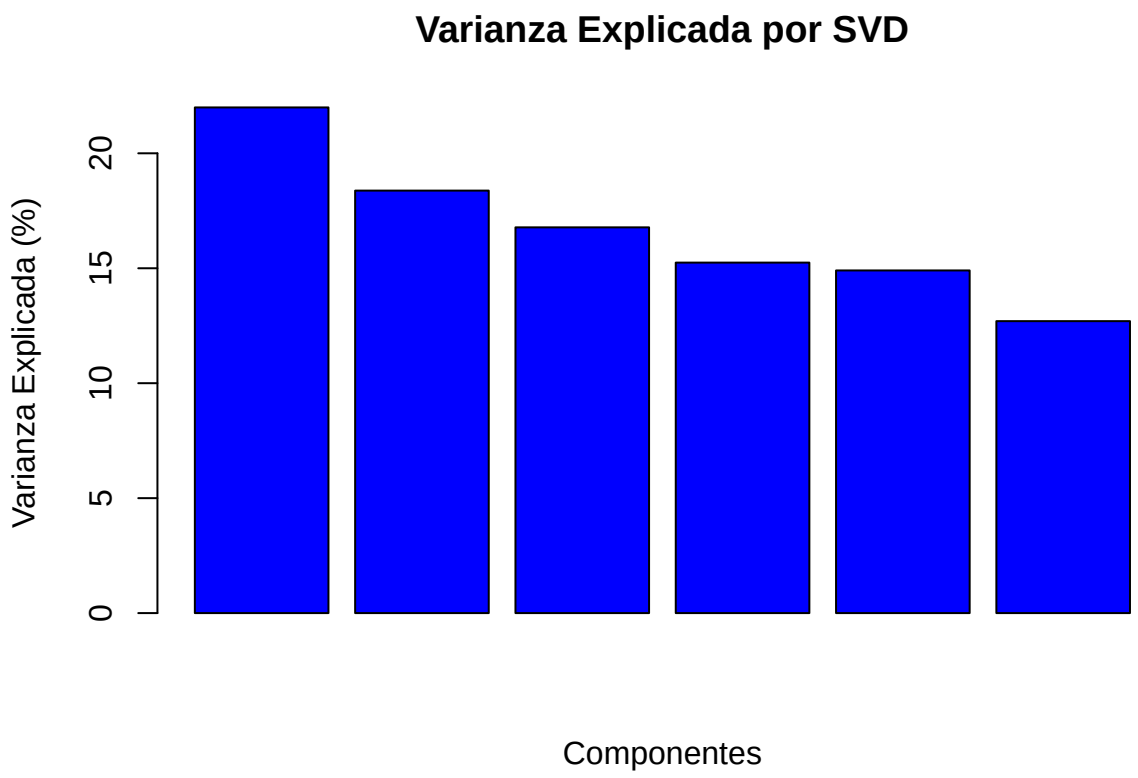
Y si pruebo con SVD?

```
#datos_escalados
svd_result <- svd(scale(datos_escalados))

U <- svd_result$u
D <- svd_result$d
V <- svd_result$v

varianza_explicada <- (D^2) / sum(D^2)
print(varianza_explicada)
```

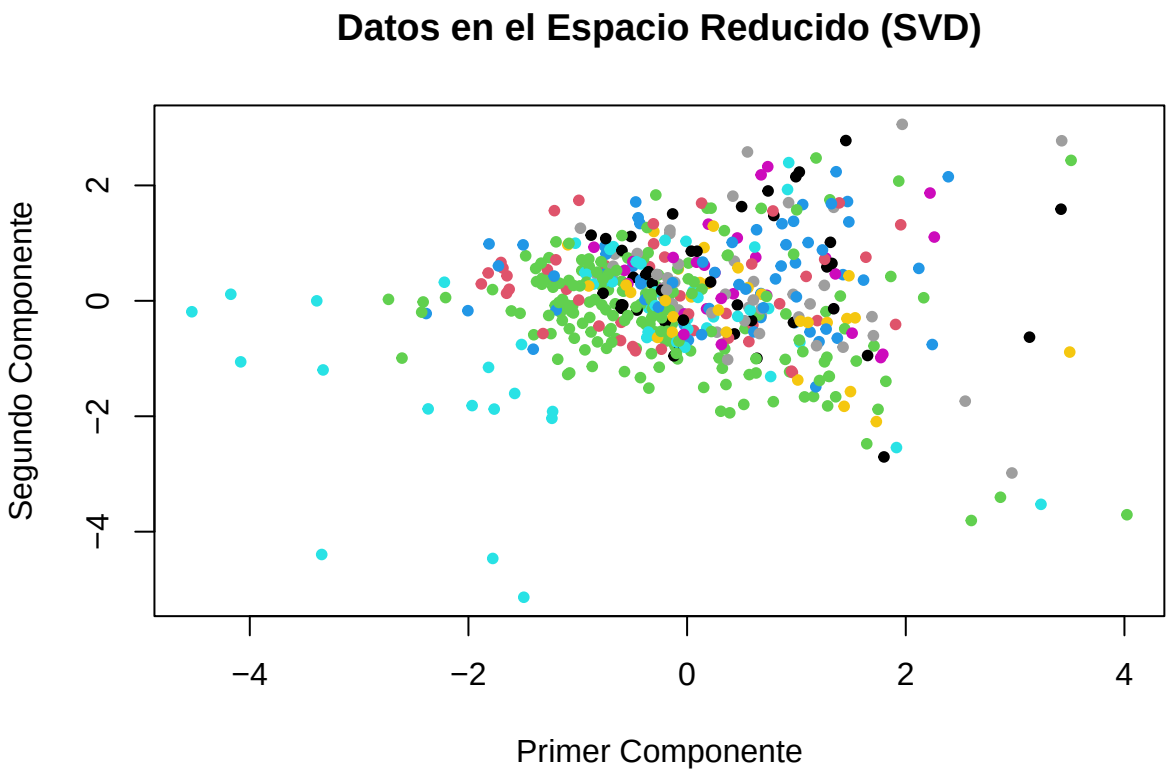
```
## [1] 0.2199564 0.1837761 0.1677843 0.1524523 0.1490433 0.1269876
```

```
datos_reducidos_svd <- U[, 1:2] %*% diag(D[1:2])
```

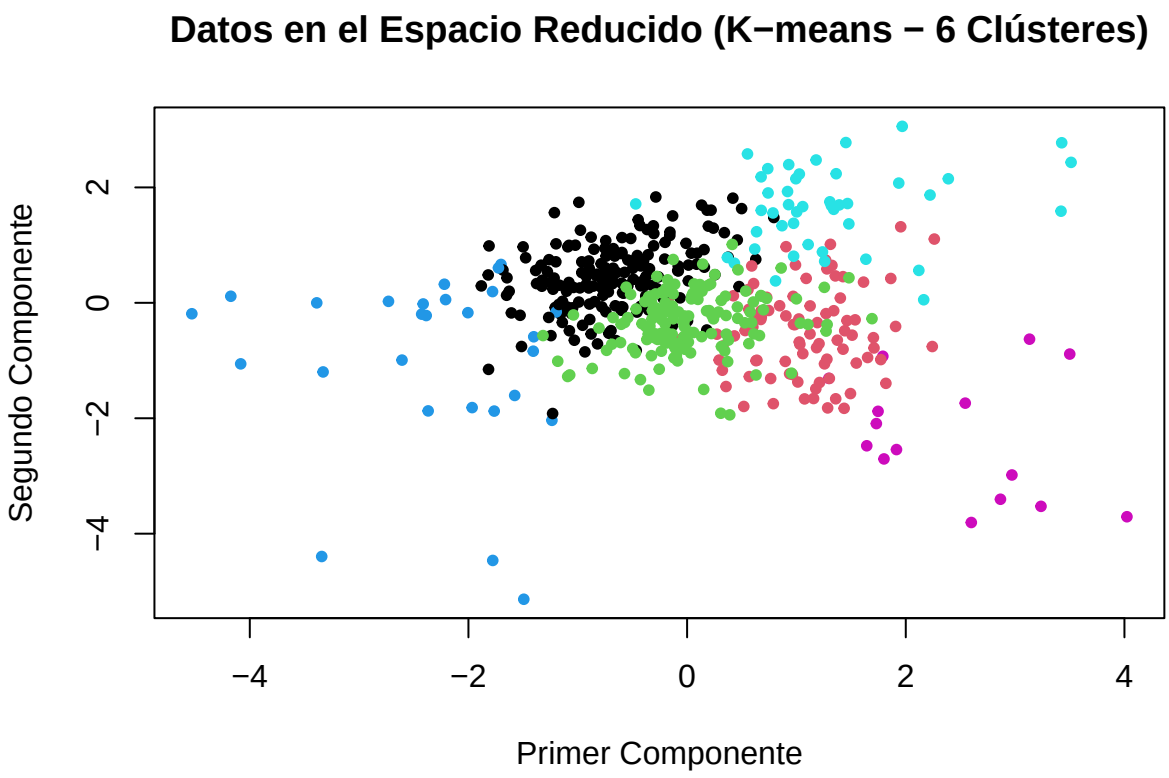
```
barplot(varianza_explicada * 100, main = "Varianza Explicada por SVD",
        xlab = "Componentes", ylab = "Varianza Explicada (%)", col = "blue")
```



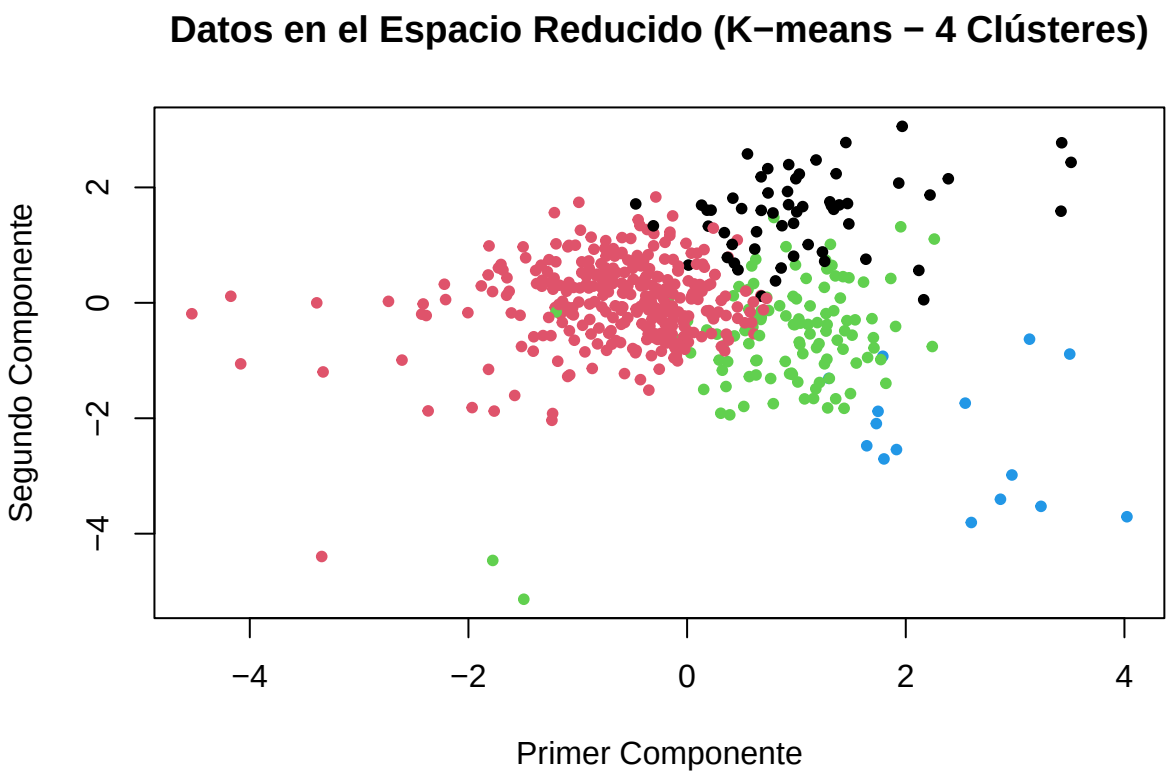

```
plot(datos_reducidos_svd,
     col = clusters_optics$cluster + 1,
     pch = 20,
     main = "Datos en el Espacio Reducido (SVD)",
     xlab = "Primer Componente",
     ylab = "Segundo Componente")
```



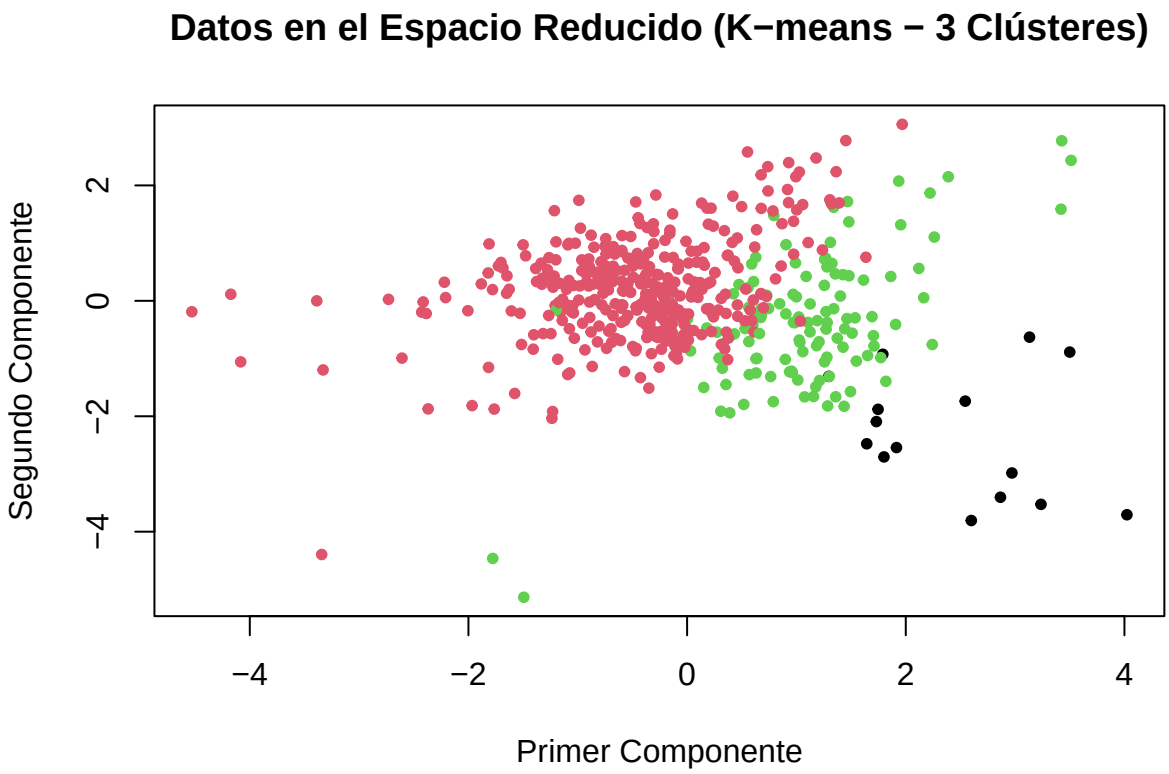

```r
plot(datos_reducidos,
     col = resultado_kmeans6$cluster,
     pch = 20,
     main = "Datos en el Espacio Reducido (K-means - 6 Clústeres)",
     xlab = "Primer Componente",
     ylab = "Segundo Componente")
```



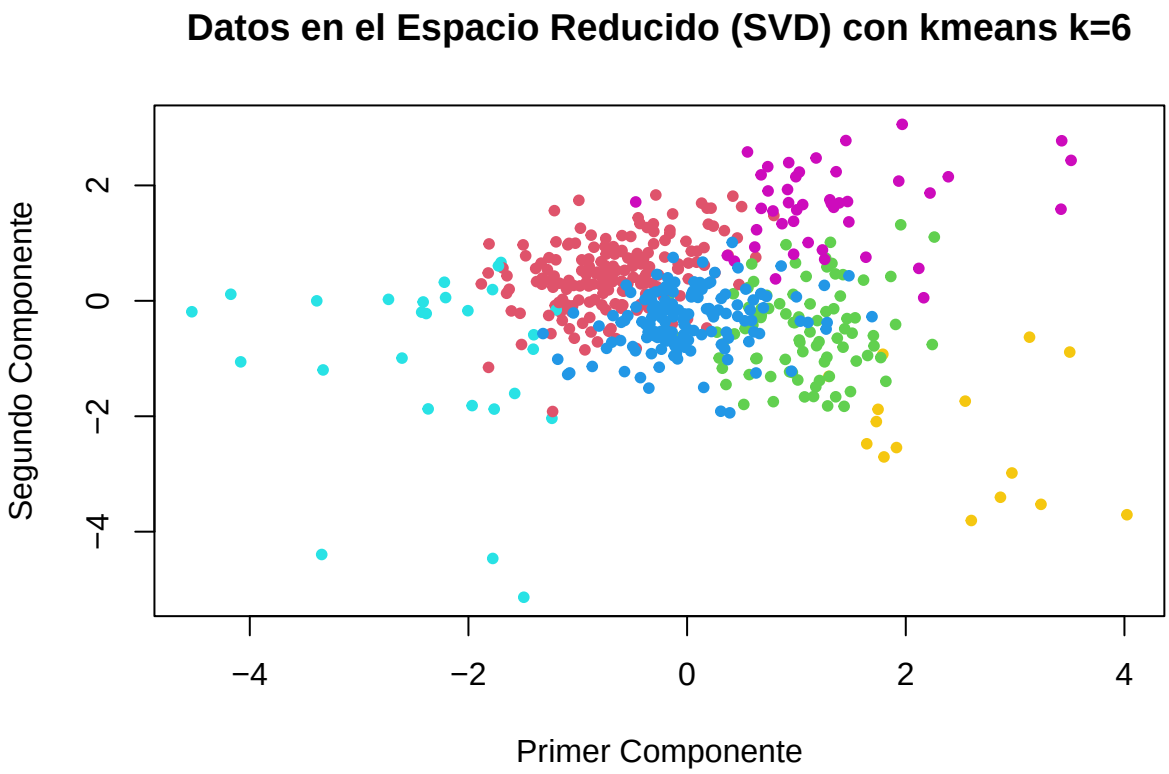

```r
plot(datos_reducidos,
     col = resultado_kmeans4$cluster,
     pch = 20,
     main = "Datos en el Espacio Reducido (K-means - 4 Clústeres)",
     xlab = "Primer Componente",
     ylab = "Segundo Componente")
```



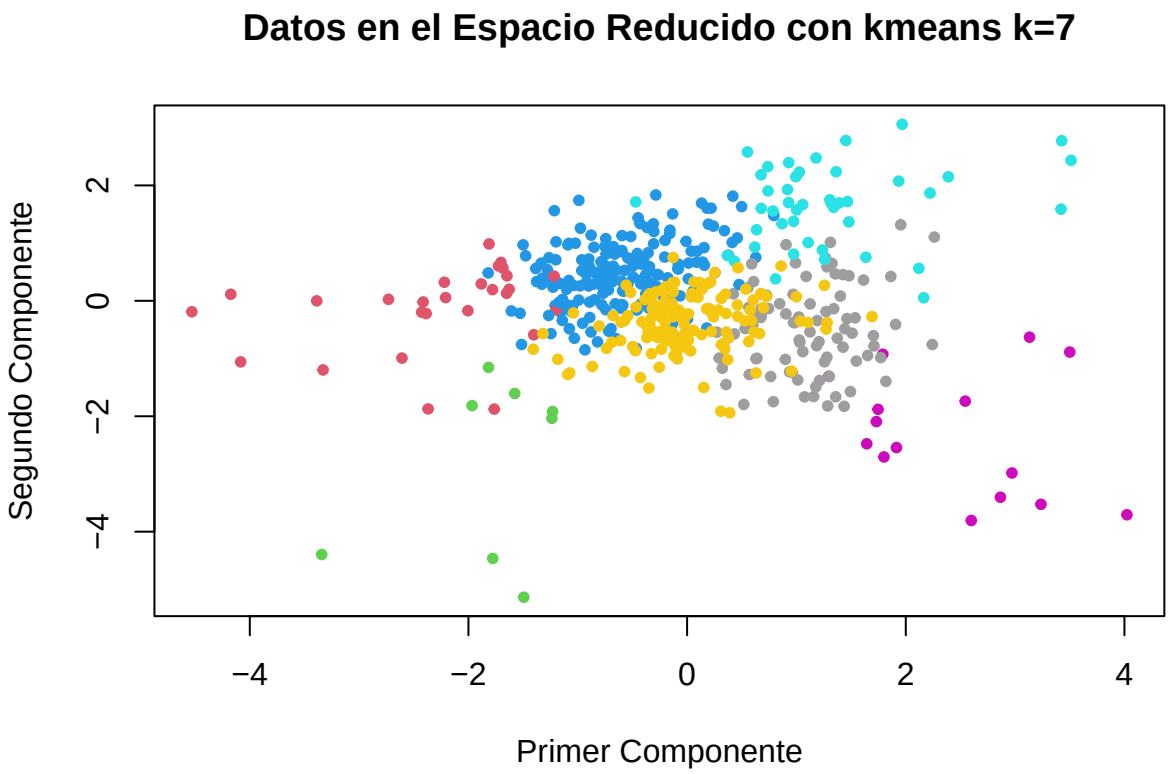

```
plot(datos_reducidos,
     col = resultado_kmeans$cluster,
     pch = 20,
     main = "Datos en el Espacio Reducido (K-means - 3 Clústeres)",
     xlab = "Primer Componente",
     ylab = "Segundo Componente")
```




```r
plot(datos_reducidos_svd,
     col = resultado_kmeans6$cluster + 1,
     pch = 20,
     main = "Datos en el Espacio Reducido (SVD) con kmeans k=6",
     xlab = "Primer Componente",
     ylab = "Segundo Componente")
```




```r
plot(datos_reducidos,
     col = resultado_kmeans7$cluster + 1,
     pch = 20,
     main = "Datos en el Espacio Reducido con kmeans k=7",
     xlab = "Primer Componente",
     ylab = "Segundo Componente")
```



Después de todas las pruebas, visualizaciones y análisis, he llegado a las siguientes conclusiones: Relación precio-peso: Existe una relación evidente entre el precio y el peso, por lo que estos son factores importantes a considerar. Estrellas: Las estrellas no aportan información relevante en el análisis, por lo que podrían eliminarse en futuros estudios. Número de reseñas vs. promedio de reseñas: El número total de reseñas no resulta muy útil, pero la media de las reseñas sí lo es. Por lo tanto, sería recomendable conservar únicamente la media y eliminar el número total de reseñas en futuros analisis. Número de páginas vs. peso: El número de páginas no aporta información significativa, pero el peso sí. Por lo tanto, sería conveniente eliminar el número de páginas y trabajar únicamente con el peso como variable.

Respecto a la PCA y SVD, ambos métodos han sido útiles para reducir la dimensionalidad de los datos y facilitar su análisis. Sin embargo, la PCA ha resultado más efectiva en este caso, ya que ha permitido explicar el 87% de la varianza con solo 4 componentes principales. Por otro lado, el SVD ha sido menos efectivo, ya que ha explicado solo el 72% de la varianza con 4 componentes.

Respecto a DBSCAN no me ha sido de mucha utilidad, ya que no ha detectado clusters claros en los datos. He probado decenas de combinaciones de xi, eps y minPts, pero no he encontrado clusters claros, por lo que he decidido quedarme con kmeans, que ha sido mas efectivo en este caso. Sin embargo kmeans ha actuado mejor, aunque no ha sido perfecto, ya que no ha detectado clusters totalment definidos, pero si ha sido de utilidad para entender mejor los datos. Por tanto me queda mostrar los datos mas llamativos de cada cluster.

Lo que mas me ha llamado la atencion, o lo que mas me ha sorprendido, es que el precio no esta tan relacionado con las estrellas, lo que quiere decir que un producto puede tener 5 estrellas y ser barato, o tener 1 estrella y ser caro, lo que me hace pensar que el precio no esta relacionado con la calidad del producto, sino con otros factores, como el peso, por ejemplo.

Lo que me ha gustado menos es que no he podido hacer un estudio de los clusters con DBSCAN, ya que no ha detectado clusters claros, pero me ha servido para entender mejor los datos y ver que no hay clusters claros, por lo que kmeans es mejor en este caso, aunque kmeans tampoco tiene un desarrollo tan claro.

Me hubiera gustado encontrar un dataset mejor, ya que si bien me ha parecido interesante, me hubiera gustado encontrar insights mas claros y utiles.

```
df_con_clusters <- cbind(df, cluster = resultado_kmeans6$cluster)
head(df_con_clusters)
```

```
##   X
## 1 0
## 2 1
## 3 2
## 4 3
## 5 4
## 6 6
##                                                                                                  title
## 1                     Data Analysis Using R (Low Priced Edition): A Primer for Data Scientist
## 2 Head First Data Analysis: A learner's guide to big numbers, statistics, and good decisions
## 3   Guerrilla Data Analysis Using Microsoft Excel: Overcoming Crap Data and Excel Skirmishes
## 4                   Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython
## 5                              Excel Data Analysis For Dummies (For Dummies (Computer/Tech))
## 6             SQL for Data Analysis: Advanced Techniques for Transforming Data into Insights
##                              author price price..including.used.books.
## 1                [ Dr Dhaval Maheta]  6.75                        6.75
## 2                               N/A 33.72               21.49 - 33.72
## 3 [ Oz du Soleil,  and , Bill Jelen] 32.07                       32.07
## 4                [ William McKinney] 53.99                       53.99
## 5                  [ Paul McFedries] 24.49                       24.49
## 6                  [ Cathy Tanimura] 90.00                       40.49
##          pages avg_reviews n_reviews star5 star4 star3 star2 star1 ingles
## 1 Menos de 750         4.4      23.0    55    39     6     0     0      1
## 2 Menos de 500         4.3     124.0    61    20     9     4     6      1
## 3 Menos de 500         4.7      10.0    87    13     0     0     0      1
## 4 Menos de 750         4.6      54.4    75    16     5     2     2      1
## 5 Menos de 500         3.9      12.0    52    17    10    10    10      1
## 6 Menos de 500         4.7     205.0    75    18     2     2     2      1
##                                      publisher  Peso..g. cluster
## 1    Notion Press Media Pvt Ltd (November 22, 2021) 1147.5878       1
## 2     O'Reilly Media; 1st edition (August 18, 2009)  889.0403       1
## 3 Holy Macro! Books; Third edition (August 1, 2022)  635.0288       1
## 4   O'Reilly Media; 2nd edition (November 14, 2017)  666.7802       1
## 5       For Dummies; 5th edition (February 3, 2022)  589.6696       1
## 6     O'Reilly Media; 1st edition (October 5, 2021)  975.2228       1
```

```
library(tidyr)
df_con_clusters <- df_con_clusters %>% select(-starts_with("X"))

df_numerico <- df_con_clusters %>%
  select(where(is.numeric), cluster)

resumen_clusters <- df_numerico %>%
  group_by(cluster) %>%
  summarise(across(where(is.numeric), list(
    media = mean,
    mediana = median,
```

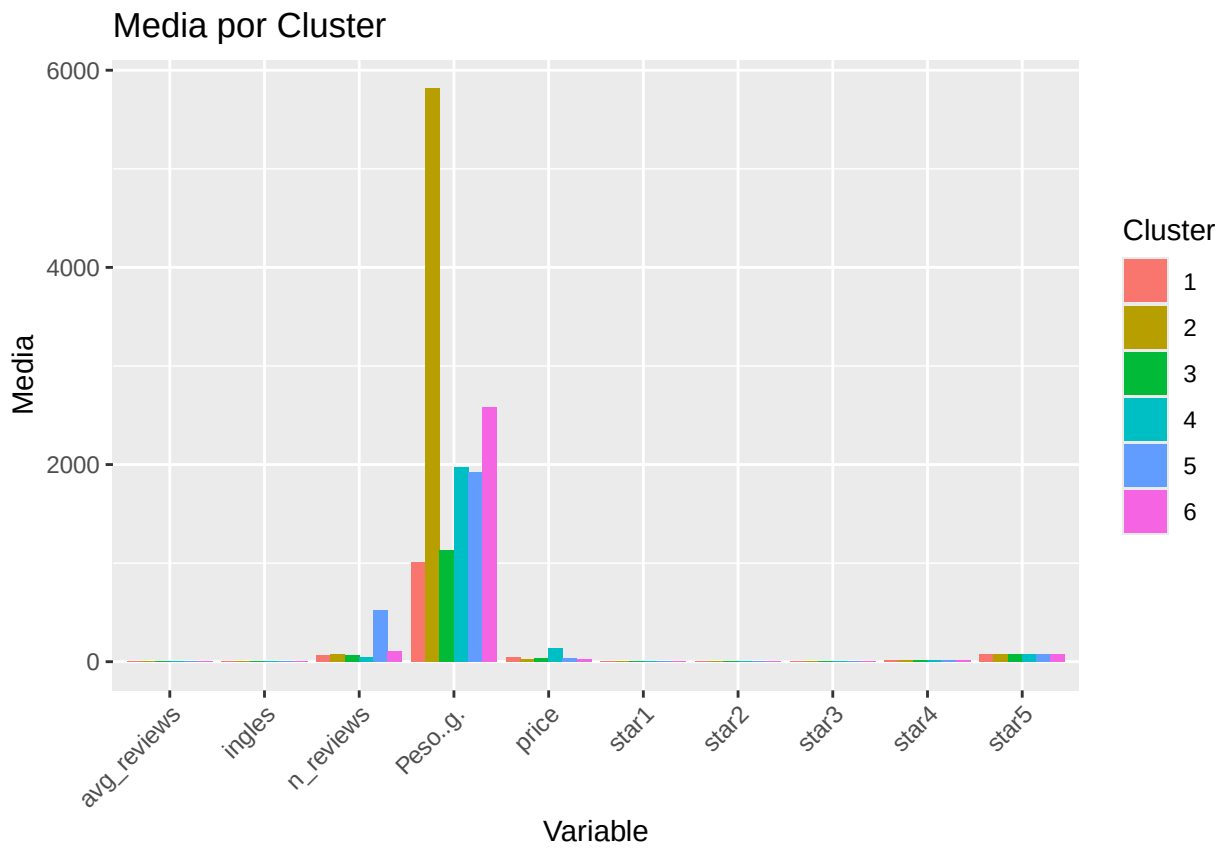
```
    sd = sd
  ), na.rm = TRUE))
```

```
## Warning: There was 1 warning in `summarise()`.
## i In argument: `across(...)`.
## i In group 1: `cluster = 1`.
## Caused by warning:
## ! The `...` argument of `across()` is deprecated as of dplyr 1.1.0.
## Supply arguments directly to `.fns` through an anonymous function instead.
##
##   # Previously
##   across(a:b, mean, na.rm = TRUE)
##
##   # Now
##   across(a:b, \(x) mean(x, na.rm = TRUE))
```

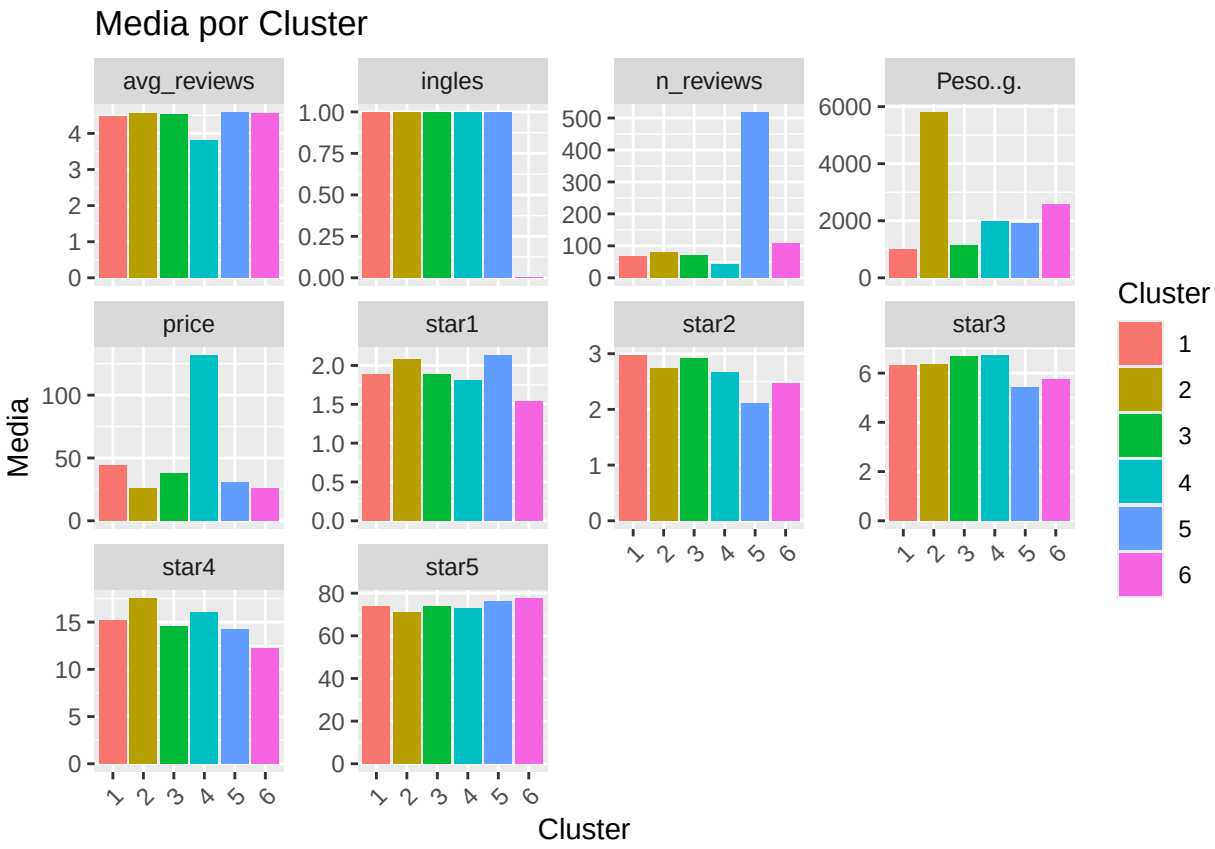
```
resumen_largo <- resumen_clusters %>%
  pivot_longer(-cluster, names_to = "variable", values_to = "valor")

resumen_largo
```

```
## # A tibble: 180 x 3
##    cluster variable             valor
##      <int> <chr>                <dbl>
##  1       1 price_media          44.2
##  2       1 price_mediana        42.0
##  3       1 price_sd             22.8
##  4       1 avg_reviews_media     4.48
##  5       1 avg_reviews_mediana   4.5
##  6       1 avg_reviews_sd        0.337
##  7       1 n_reviews_media      67.2
##  8       1 n_reviews_mediana    32
##  9       1 n_reviews_sd         79.8
## 10       1 star5_media          73.7
## # i 170 more rows
```

```
resumen_medias <- df_con_clusters %>%
  select(where(is.numeric), cluster) %>%
  group_by(cluster) %>%
  summarise(across(where(is.numeric), mean, na.rm = TRUE)) %>%
  pivot_longer(-cluster, names_to = "variable", values_to = "media")

ggplot(resumen_medias, aes(x = variable, y = media, fill = as.factor(cluster))) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Media por Cluster", x = "Variable", y = "Media", fill = "Cluster") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```
ggplot(resumen_medias, aes(x = as.factor(cluster), y = media, fill = as.factor(cluster))) +
  geom_bar(stat = "identity", position = "dodge") +
  facet_wrap(~ variable, scales = "free_y") +
  labs(title = "Media por Cluster", x = "Cluster", y = "Media", fill = "Cluster") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Respecto a los clusters, el numero 2 destaca por tener un peso mas alto que el resto el 4 por un precio mas alto, el 5 por tener un gran numero de reviews, y seguramente el que parece mas genuino respecto a las valoraciones de los clientes es el 6, ya que tiene mas valoraciones de 5 estrellas, y no tiene tantas como el resto de 1 estrella. Tambien destaca en el cluster 5 el numero de valoraciones, que indicaria seguramente que son libros ya maduros en el mercado y que llevan mas tiempo vendiendose, o que son superventas. Por ultimo la variable ingles no ha dado mucha informacion util.