

Ejercicio 1:

Apartado a:

Creamos el script, le damos permisos de ejecución y probamos los resultados:

```
./prepare.awk Jan_2020_ontime.csv > flights_jan_20.csv
```

En la captura podemos ver la comprobación con el archivo aportado en la práctica, mediante el comando **diff** y filtrando los resultados:

```

1 # 28,Moday,CVG,DFW,0710,0.00,0.00,0.00,1503.82
2 # 28,Moday,DFW,CVG,0940,0.00,0.00,0.00,1503.82
3 # 28,Moday,TYS,DTW,0910,0.00,0.00,0.00,820.436
4 # 28,Moday,MSP,PIT,1353,0.00,0.00,0.00,1344.55
5 # 28,Moday,JFK,PIT,0743,0.00,0.00,0.00,629.68
6 B# 28,Moday,PIT,JFK,1021,0.00,0.00,0.00,629.68
7 { 28,Moday,STL,MSP,1940,0.00,0.00,0.00,829.696
8 { 28,Moday,LGA,MEN,1443,0.00,0.00,0.00,1783.48
9 ) 28,Moday,MEM,LGA,1726,0.00,0.00,0.00,1783.48
10 ) 28,Moday,LGA,STL,1467,0.00,0.00,0.00,1644.58
11 L# 28,Moday,STL,LGA,1648,0.00,0.00,0.00,1644.58
12 28,Moday,LGA,SFO,1704,0.00,0.00,0.00,1042.68
13 C# 28,Moday,LGA,CGO,1848,0.00,0.00,0.00,1042.68
14 28,Moday,CLT,LGA,1848,0.00,0.00,0.00,1007.49
15 28,Moday,ATL,SFO,1227,0.00,0.00,0.00,1042.68
16 28,Moday,ATL,SFO,1349,0.00,0.00,0.00,1042.68
17 28,Moday,BOS,MKE,0795,0.00,0.00,0.00,1592.72
18 28,Moday,BOS,RDU,0826,0.00,0.00,0.00,1133.42
19 28,Moday,GSO,LGA,1135,0.00,0.00,0.00,853.772
20 28,Moday,LGA,GSO,0849,0.00,0.00,0.00,853.772
21 28,Moday,ATL,TR1,2220,0.00,0.00,0.00,420.404
22 28,Moday,ATL,XNA,1136,0.00,0.00,0.00,1096.83
23 28,Moday,XNA,ATL,1314,0.00,0.00,0.00,1096.83
24 28,Moday,LGA,MKE,1726,0.00,0.00,0.00,1366.78
25 28,Moday,LGA,RIC,1631,0.00,0.00,0.00,540.784
26 28,Moday,RIC,LGA,1850,0.00,0.00,0.00,540.784
27 28,Moday,CVG,DCA,1645,0.00,0.00,0.00,761.172
LibreOffice Calc DCA,CVG,1849,0.00,0.00,0.00,761.172
28,Moday,DCA,CVG,1599,0.00,0.00,0.00,761.172
30 ] 28,Moday,JFK,BIV,1515,0.00,0.00,0.00,492.632
31 ] 28,Moday,ORD,JFK,1035,0.00,0.00,0.00,1370.48
28,Moday,LGA,CLT,2000,0.00,0.00,0.00,1007.49
pscrimaseda@pscrimaseda-VirtualBox:~$ diff -u flights_jan_20.csv flights_jan_20_v2.csv | grep -E '^+|-|-' | wc -l
76645
pscrimaseda@pscrimaseda-VirtualBox:~$ ./prepare.awk Jan_2020_ontime.csv > flights_jan_20.csv
pscrimaseda@pscrimaseda-VirtualBox:~$ diff -u flights_jan_20.csv flights_jan_20_v2.csv | grep -E '^+|-|-' | wc -l
0
pscrimaseda@pscrimaseda-VirtualBox:~$ diff -u flights_jan_20.csv flights_jan_20_v2.csv | grep -E '^+|-|-' | awk '{print $1}'
pscrimaseda@pscrimaseda-VirtualBox:~$ 
```

Al principio hay muchos errores debido a Moday, en vez de Monday, una vez cambiado en el script podemos ver que los ficheros son exactamente iguales.

Apartado b:

He empleado el archivo generado para obtener los datos deseados, como hemos podido comprobar no existe diferencia entre el nuevo archivo creado por mi script y el aportado en la asignatura.

Creamos el script con los requerimientos del ejercicio y lo ejecutamos:

```
./delayed_by_day_jan20.awk flights_jan_20.csv
```

En el ejemplo de salida mostrado en el enunciado aparecen distintos formatos a la hora de imprimir, he decidido delimitar a 5 decimales para crear todas las salidas iguales ("%.⁵f"), los valores no se redondean pero me ha parecido correcto hacerlo así, vemos la salida:

The screenshot shows a Linux desktop environment with a terminal window open. The terminal window title is "delayed_by_day_jan20.awk". The terminal content displays an awk script and its execution results. The script reads a CSV file named "flights_jan20.csv" and processes it to count delayed flights by day of the week. The output shows counts for Monday through Sunday.

```
#!/usr/bin/awk -f
1
2
3 BEGIN {
4     FS=",";
5     OFS=",";
6     CONVFMT="%s, %F"
7 }
8 if (NR==1) {print "Weekday, % delayed, delayed, total flights"}
9 {
10     if (NR==1) {print "Monday, 9,86118,7558,76644"}
11     else {
12         if ($2==
13             Wednesday,13,43898,13933,103676
14             Thursday,15,28725,15870,103812
15             Friday,19,02015,12462,65528
16             Saturday,14,42692,11326,78506
17             Sunday,13,56492,11176,82389
18         )
19         if ($2==
20             )
21         if ($2==
22             )
23         if ($2==
24             )
25         if ($2==
26             )
27         if ($2==
28             )
29         if ($2==
30             )
31         if ($2==
32             )
33         if ($2==
34             )
35         if ($2==
36             )
37         if ($2==
38             )
39         if ($2==
40             )
41         if ($2==
42             )
43             delay_saturday++;
44         }
45         if ($2=="Sunday") {
46             total_sunday++;
47         }
48     }
}
pscrimaseda@pscrimaseda-VirtualBox:~
```

pscrimaseda@pscrimaseda-VirtualBox:~ \$./delayed_by_day_jan20.awk flights_jan20.csv

Weekday	% delayed	Delayed	Total
Monday	9.86118	7558	76644
Tuesday	9.87510	9559	96799
Wednesday	13.43898	13933	103676
Thursday	15.28725	15870	103812
Friday	19.02015	12462	65528
Saturday	14.42692	11326	78506
Sunday	13.56492	11176	82389

Apartado c:

Creamos el script con los requerimientos del enunciado y lo ejecutamos, en este caso debemos introducir un valor en la ejecución, mostraremos el mismo aportado en el enunciado (airport="ORD"), vemos los resultados (mismo criterio con los decimales que en el apartado anterior):

```
./airport_delays_by_day_jan20.awk -v airport="ORD" flights_jan_20.csv
```

Ejercicio 2:

El objetivo del proyecto elegido es el tratamiento de un conjunto de datos numéricos, quizás no es el conjunto que más se adapta a todo lo visto a lo largo de la asignatura pero me pareció interesante abordar este tema, normalmente cuando nos enfrentamos a conjuntos de datos así, empleamos funciones sencillas en distintos lenguajes de programación que nos aportan información sobre dichos datos, como `summary()` en R, `describe()` de pandas... También GNU nos ofrece herramientas para esto:

<https://www.gnu.org/software/datamash/>

Pero me pareció interesante para profundizar en el aprendizaje de programación en scripting tratar de hacer mis propios scripts que realicen estas funciones, además de esto también he creado una serie de funciones para tratar los valores nulos, concretamente sustituyéndolos por la media, moda, mediana o un valor introducido por nosotros mismos, la normalización de los datos, por máximo y mínimo, normalización estándar, y una función para realizar lo que se llama one hot encoder y dividir una variable categórica en distintas columnas con valores binarios (0-1), además de esto me pareció interesante aprender sobre gnuplot, de forma que se mostrarán algunas estadísticas y datos filtrados en los distintos scripts. Por tanto, podríamos decir que el objetivo del proyecto es lidiar con un conjunto de datos numéricos sin emplear funciones predefinidas por los distintos lenguajes de programación, calcular todos los valores a través de un script y transformar los datos (dependiendo de las necesidades de los modelos a aplicar). Hoy en día quizás suponga una perdida de tiempo hacer este tipo de cosas y no suponga una mejora a nivel de eficiencia que justifique el tiempo que he tardado en generar todos los scripts, pero como he dicho, a nivel de aprendizaje me parece una buena forma de profundizar en muchos aspectos de lo estudiado en la asignatura, también podemos ver la velocidad que ofrecen estas herramientas para lidiar con conjuntos de datos grandes, en el ejercicio 1 hemos visto como los tiempos de ejecución para distintas operaciones con un dataset con más de 600.000 registros son realmente buenos, en mi caso, el conjunto de datos es bastante más reducido, pero para las numerosas pruebas que he tenido que ir haciendo creo que es mejor lidiar con menos datos (3.277 registros y 10 atributos en mi caso), en cualquier caso, creo que es un conjunto de datos lo suficientemente grande para justificar el uso de herramientas informáticas, estos cálculos sin dichas herramientas supondrían un trabajo tedioso.

Haré un pequeño índice de los puntos del ejercicio:

1. **Apartado a:** Informe de los datos y enlace de descarga (ir a pág 18 para aclaraciones en la descarga) – pág 4
2. **Apartado b:** Preparación y visualización de los datos – pág 8
3. **Apartado c:** script informe estadístico (en awk) – pág 13
script de la transformación de los datos (en bash) – pág 14
4. **Apartado d:** Informe en htm – pág 15
5. **Apartado e:** script global “run.sh” – pág 17
6. **(IMPORTANTE) Aclaraciones y requisitos** – pág 18

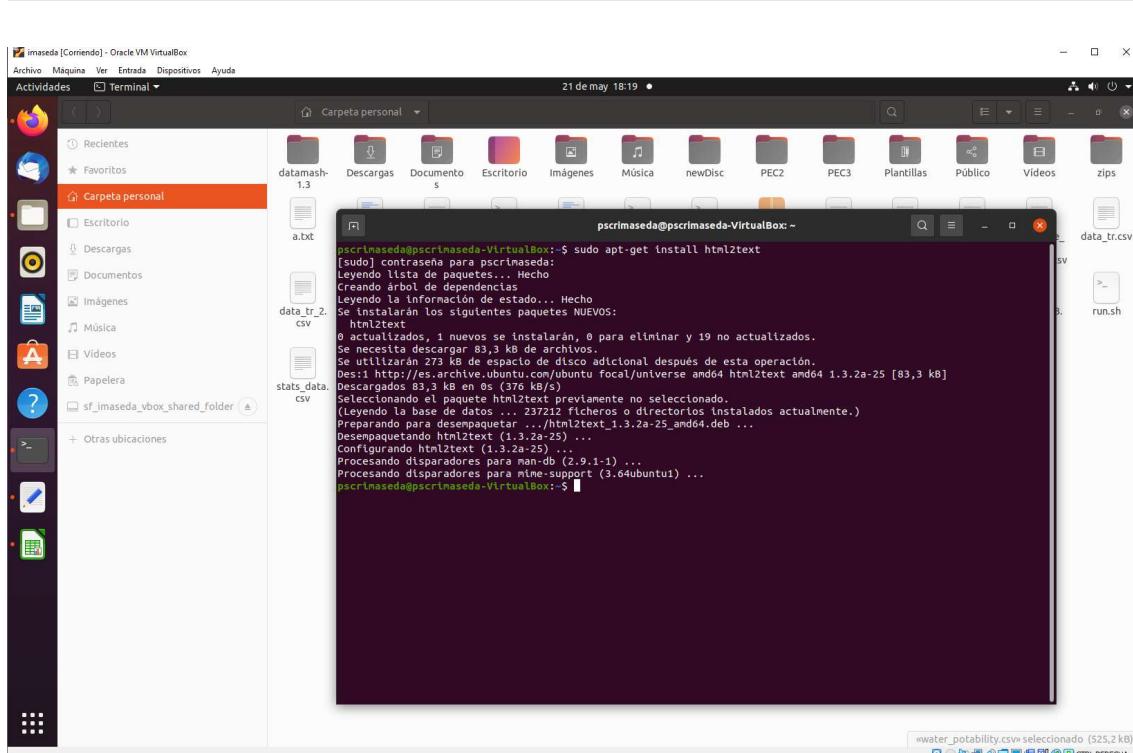
Apartado a:

El conjunto de datos lo he obtenido de Kaggle, concretamente:

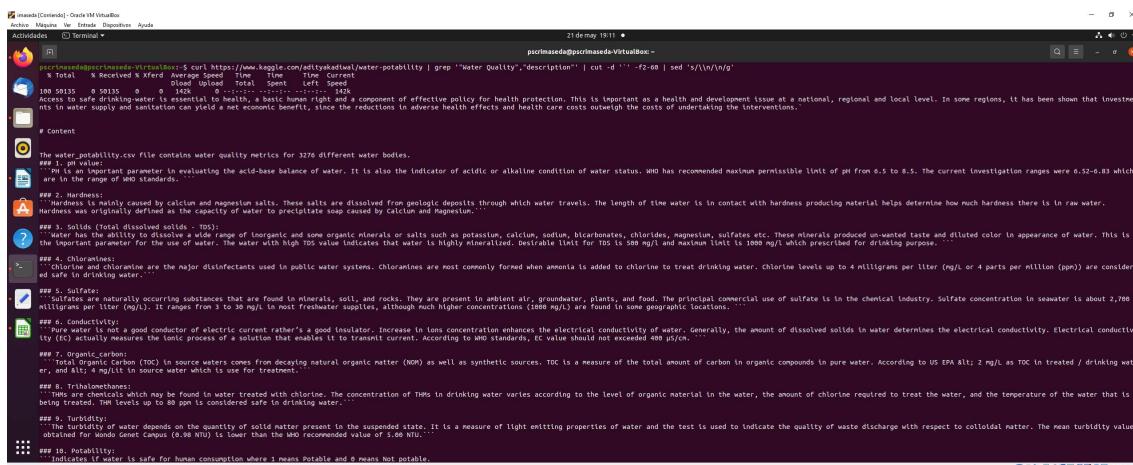
<https://www.kaggle.com/adityakadiwal/water-potability>

He tratado de descargar algunas herramientas para obtener información del código fuente de la página web, pero no he obtenido los resultados deseados, lo he intentado con html2text, en la captura vemos la instalación, pero el resultado no ha sido el adecuado, por tanto he decidido una opción un poco rudimentaria pero vemos que muestra el resultado deseado en pantalla, observando el código de la página he podido filtrar la descripción de los atributos en un formato bastante legible, y también pasarlo a un archivo .txt:

```
curl https://www.kaggle.com/adityakadiwal/water-potability | \
grep '"Water Quality","description"' | cut -d '"' -f2-60 | sed 's/\n/\n/g'
```



```
pscrinaseda@pscrinaseda-VirtualBox:~$ curl https://www.kaggle.com/adityakadiwal/water-potability | grep '"Water Quality","description"' | cut -d '"' -f2-60 | sed 's/\n/\n/g'
[sudo] contraseña: pscrinaseda
Leiendo lista de paquetes... Hecho
Creando árbol de dependencias
Leiendo la información de estado... Hecho
Se han añadido los siguientes paquetes NUEVOS:
  html2text
  0 actualizados, 1 nuevos se instalarán, 0 para eliminar y 19 no actualizados.
Se necesita descargar 83,3 kB de archivos.
Se utilizarán 273 kB de espacio de disco adicional después de esta operación.
Des:1 http://es.archive.ubuntu.com/ubuntu focal/universe amd64 html2text amd64 1.3.2a-25 [83,3 kB]
Descargados 83,3 kB en 0s (376 kB/s)
Seleccionando el paquete html2text previamente no seleccionado.
(leyendo la base de datos... 237212 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar.../html2text_1.3.2a-25_amd64.deb ...
Desempaquetando html2text (1.3.2a-25) ...
Configurando html2text (1.3.2a-25) ...
Procesando disparadores para man-db (2.9.1-1) ...
Procesando disparadores para mine-support (3.64ubuntui) ...
pscrinaseda@pscrinaseda-VirtualBox:~$
```

```
# Content
The water_potability.csv file contains water quality metrics for 3276 different water bodies.
## 1. pH value:
  `` pH is an important parameter in evaluating the acid-base balance of water. It is also the indicator of acidic or alkaline condition of water status. WHO has recommended maximum permissible limit of pH from 6.5 to 8.5. The current investigation ranges were 6.52-8.83 which is in range of WHO standards.``

## 2. Hardness:
  `` Hardness is mainly caused by calcium and magnesium salts. These salts are dissolved from geologic deposits through which water travels. The length of time water is in contact with hardness producing material helps determine how much hardness there is in raw water. Hardness is originally defined as the capacity of water to precipitate soap caused by calcium and magnesium.``

## 3. Solids (total dissolved solids):
  `` TDS is the total amount of inorganic and some organic minerals or salts such as potassium, calcium, sodium, bicarbonates, chlorides, magnesium, sulfates etc. These minerals produce unwanted taste and diluted color in appearance of water. This is the important parameter for the use of water. The water with high TDS value indicates that water is highly mineralized. Desirable limit for TDS is 500 mg/l and maximum limit is 1000 mg/l which is prescribed for drinking purpose.``

## 4. Chlorate:
  `` Chlorate is a strong oxidizing agent and a good disinfectant. The water with high chlorine content is used for drinking water.``

## 5. Sulfate:
  `` Sulfates are naturally occurring substances that are found in minerals, soil, and rocks. They are present in ambient air, groundwater, plants, and food. The principal commercial use of sulfate is in the chemical industry. Sulfate concentration in seawater is about 2,700 mg/L. It ranges from 3 to 30 mg/L in most freshwater supplies, although much higher concentrations (1000 mg/L) are found in some geographic locations.``

## 6. Conductivity:
  `` Pure water is not a good conductor of electric current rather it's a good insulator. Increase in ionic concentration enhances the electrical conductivity of water. Generally, the amount of dissolved solids in water determines the electrical conductivity. Electrical conductivity (EC) actually measures the ionic process of a solution that enables it to transmit current. According to WHO standards, EC value should not exceed 400 µS/cm.``

## 7. Organic_carbon:
  `` Total organic carbon (TOC) in source waters come from decaying natural organic matter (NOM) as well as synthetic sources. TOC is a measure of the total amount of carbon in organic compounds in pure water. According to US EPA 811; 2 mg/L as TOC in treated / drinking water, and 4 mg/L in source water which is use for treatment.``

## 8. Trihalomethanes:
  `` THMs are chemicals which may be found in water treated with chlorine. The concentration of THMs in drinking water varies according to the level of organic material in the water, the amount of chlorine required to treat the water, and the temperature of the water that is being treated. Levels up to 80 ppb can be considered safe to drinking water.``

## 9. Turbidity:
  `` The turbidity of water depends on the quantity of solid matter present in the suspended state. It is a measure of light scattering properties of water and the test is used to indicate the quality of waste discharge with respect to colloidal matter. The mean turbidity value obtained for Wondo Genet Campus (8.98 NTU) is lower than the WHO recommended value of 5.00 NTU.``

## 10. Potability:
  `` Water is said to be potable if water is safe for human consumption where 1 means Potable and 0 means Not potable.``
```

```

#!/bin/bash
# Access to safe drinking-water is essential to health, a basic human right and a component of effective policy for health protection. This is important as a health and development issue at a national, regional and local level. In some regions, it has been shown that investments in water supply and sanitation can yield a net economic benefit, since the reductions in adverse health effects and health care costs outweigh the costs of undertaking the interventions.

# 1. pH value:
# pH is an important parameter in maintaining the acid-base balance of water. It is also the indicator of acidic or alkaline condition of water status. WHO has recommended maximum permissible limit of pH from 6.5 to 8.5. The current investigation ranges were 6.52-6.83

# 2. Hardness:
# Hardness is mainly caused by calcium and magnesium salts. These salts are dissolved from geological deposits through which water travels. The length of time water is in contact with hardness producing material helps determine how much hardness there is in raw water.

# 3. Dissolved Solids (Total dissolved solids - TDS):
# Water has the ability to dissolve a wide range of inorganic and some organic minerals or salts such as potassium, calcium, sodium, bicarbonates, chlorides, magnesium, sulfates etc. These minerals produce unwanted taste and diluted color in appearance of water. This is the important parameter for the use of water. The water with high TDS value indicates that water is highly mineralized. Desirable limit for TDS is 300 mg/L and maximum limit is 1000 mg/L which is prescribed for drinking purpose.

# 4. Chlorine and Chlorite are the major disinfectants used in public water systems. Chlorates are most commonly formed when chlorine is added to chlorine to treat drinking water. Chlorite levels up to 4 milligrams per liter (mg/L) or 4 parts per million (ppm) are considered safe for drinking water.

# 5. Sulfate:
# Sulfates are naturally occurring substances that are found in minerals, soil, and rocks. They are present in ambient air, groundwater, plants, and food. The principal commercial use of sulfate is in the chemical industry. Sulfate concentration in seawater is about 2,700 milligrams per liter (mg/L). It ranges from 1 to 30 mg/L in most freshwater supplies, although much higher concentrations (1000 mg/L) are found in some geographic locations.

# 6. Conductivity:
# Conductivity is a good conductor of electric current rather than a good insulator. Increase in ionic concentration enhances the electrical conductivity of water. Generally, the amount of dissolved solids in water determines the electrical conductivity. Electrical conductivity (EC) actually measures the ionic process of a solution that enables it to transmit current. According to WHO standards, EC value should not exceed 400 µS/cm.

# 7. Organic Carbon:
# Total Organic Carbon (TOC) in source waters comes from decaying natural organic matter (NOM) as well as synthetic sources. TOC is a measure of the total amount of carbon in organic compounds in pure water. According to US EPA 611; 2 mg/L as TOC in treated / drinking water, and also 4 mg/L in source water which is used for treatment.

# 8. Trihalomethanes:
# Trihalomethanes are chemicals which may be found in water treated with chlorine. The concentration of THMs in drinking water varies according to the level of organic material in the water, the amount of chlorine required to treat the water, and the temperature of the water that is treated. THM levels up to 88 ppm is considered safe in drinking water.

# 9. Turbidity:
# Turbidity of water depends on the quantity of solid matter present in the suspended state. It is a measure of light scattering properties of water and the test is used to indicate the quality of waste discharge with respect to colloidal matter. The mean turbidity value obtained for Wondo Genet Campus (9.98 NTU) is lower than the WHO recommended value of 5.00 NTU.

# 10. Potability:
# Indicates if water is safe for human consumption where 1 means Potable and 0 means Not potable.

```

La licencia de los datos es pública, como podemos ver en la página de descarga:

Dataset

Water Quality
Drinking water potability

Aditya Kadiwal • updated a month ago (Version 3)

Data Tasks (1) Code (3) Discussion (1) Activity Metadata Download (513 KB) New Notebook

Usability 10.0 License CC0-Public Domain Tags earth and nature, beginner, energy, public health, environment and 2 more

Description

Context

Access to safe drinking-water is essential to health, a basic human right and a component of effective policy for health protection. This is important as a health and development issue at a national, regional and local level. In some regions, it has been shown that investments in water supply and sanitation can yield a net economic benefit, since the reductions in adverse health effects and health care costs outweigh the costs of undertaking the interventions.

Content

Dataset

Water Quality
Drinking water potability

Aditya Kadiwal • updated a month ago (Version 3)

Data Tasks (1) Code (3) Discussion (1) Activity Metadata Download (513 KB) New Notebook

Metadata

Usage Information License CC0: Public Domain
Visibility Public

Provenance Sources Synthetically generated

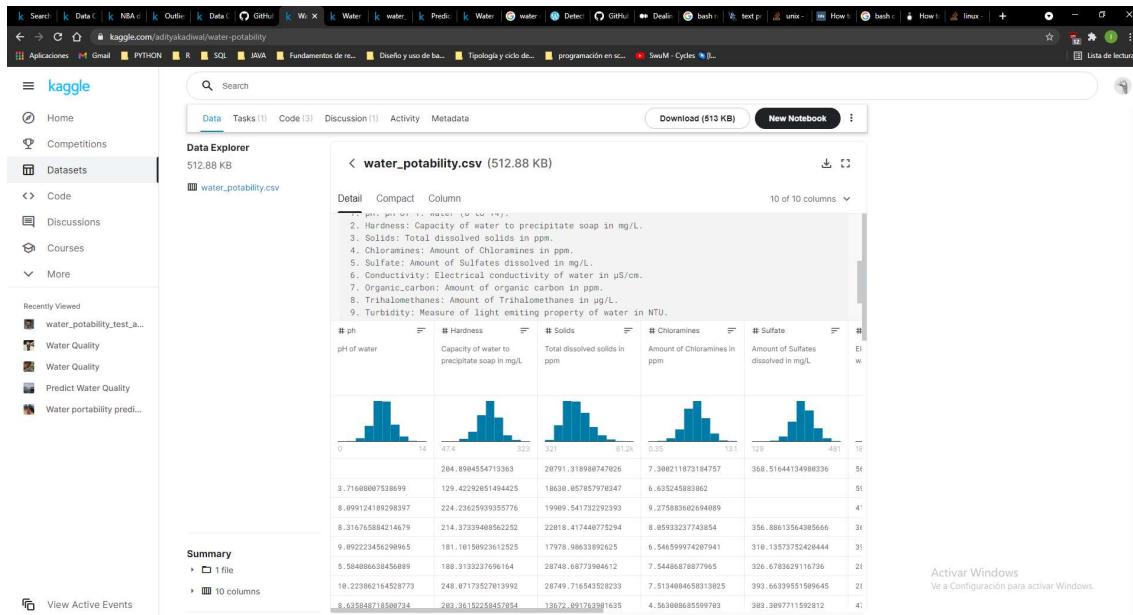
Maintainers Dataset owner Aditya Kadiwal

Updates Expected update frequency Annually
Last updated 2021-04-25
Date created 2021-04-24
Current version Version 3

Para obtener el número de registros y columnas bastará con un simple comando en awk:

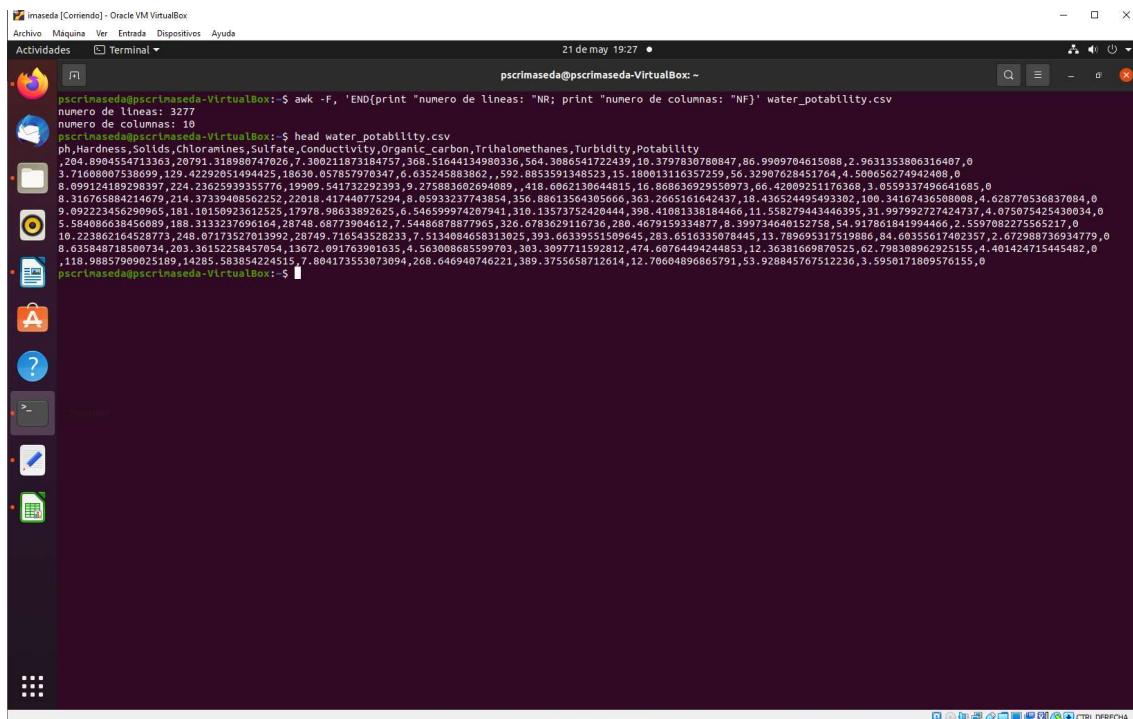
```
awk -F, 'END{print "numero de lineas: "NR; print "numero de columnas: "NF}' \
water_potability.csv
```

En cuanto al formato del archivo, Kaggle nos facilita mucho las cosas ofreciendo mucha información al respecto de todas las variables, el tipo de archivo, incluso podemos ver distribuciones:



En cualquier caso, podemos imprimir la cabecera en nuestra consola y comprobar que todos los datos son numéricos:

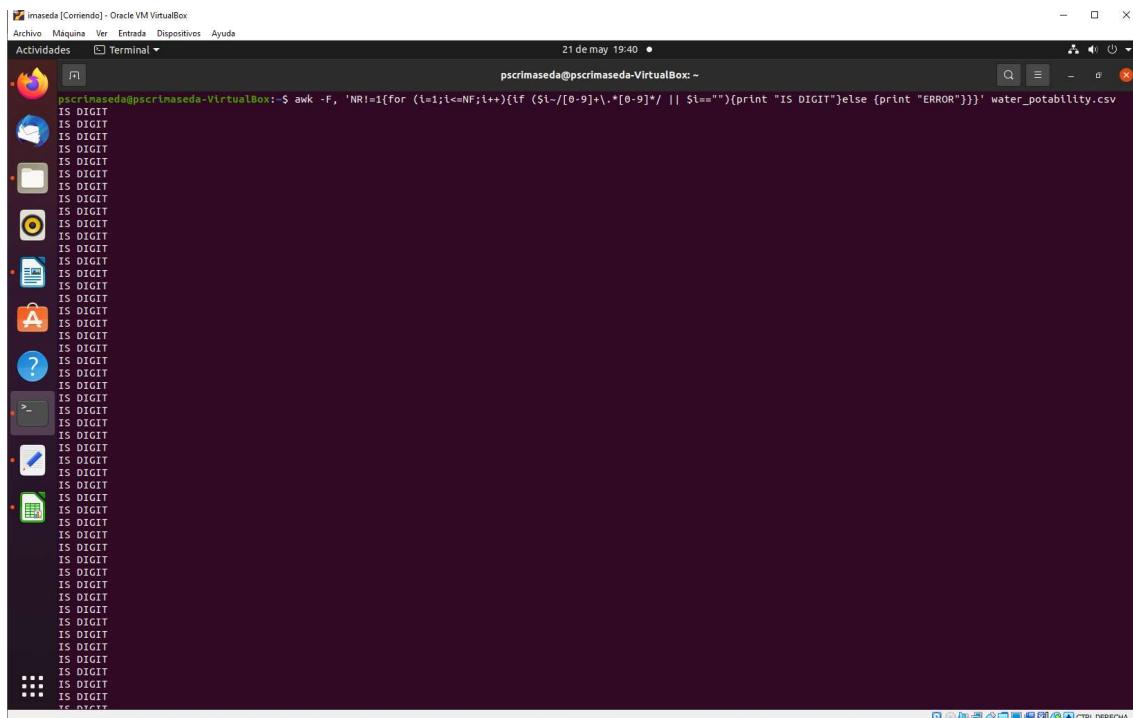
```
head water_potability.csv
```



También podemos hacer una pequeña comprobación con awk:

```
awk -F, 'NR!=1{for (i=1;i<NF;i++) {if ($i~/[0-9]+\.*[0-9]*/ || $i=="") \
{print "IS DIGIT"}else {print "ERROR"}}}' water_potability.csv
```

En caso de obtener algún error podríamos filtrar y buscar el número de fila, pero no es necesario



Apartado b (b_part.sh):

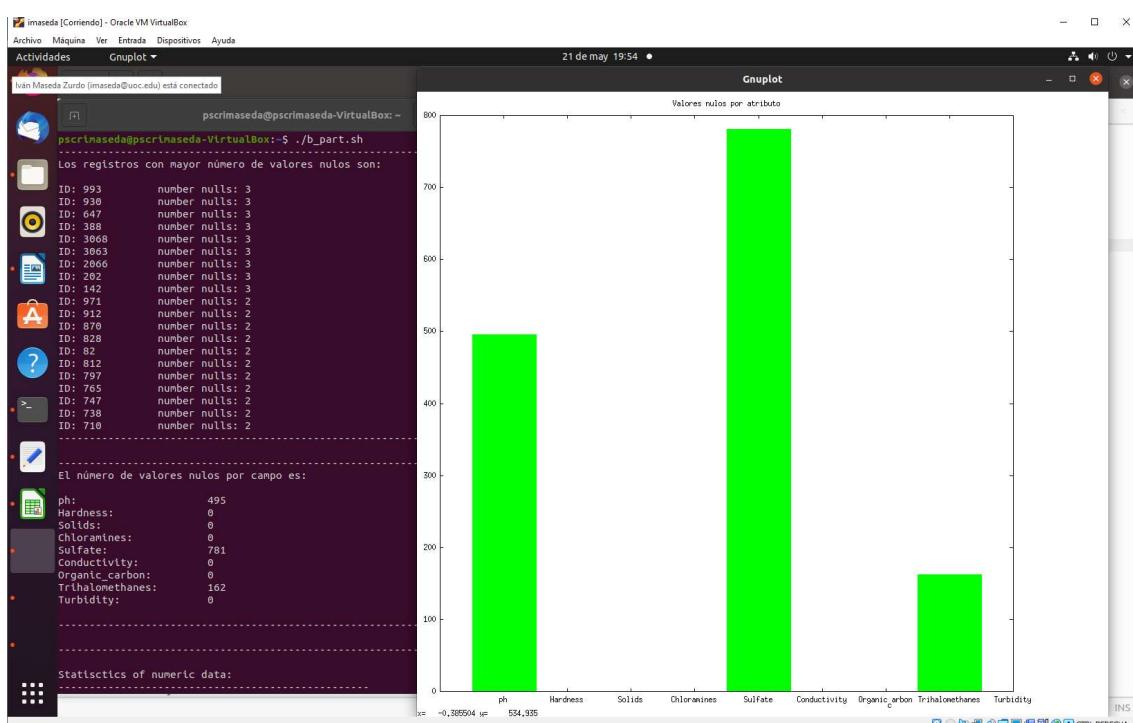
En este script se toma un primer contacto con los datos y se corrigen ciertos errores que podrían ser molestos en un futuro, lo primero es eliminar el retorno de carro “\r” que puede ser muy incómodo en algunos casos y aprovechamos para copiar los datos en un nuevo archivo que llamaremos data_tr.csv, para conservar los datos originales sin modificar.

Lo siguiente que haremos será un primer tratamiento de datos nulos, vemos que están representados por una cadena de texto vacía, o así lo interpreta al no existir nada entre las comas, vamos a sustituirlo por la cadena de texto “null”. En este paso también corregimos el valor de pH, sabemos que puede tomar valores entre 1 y 14, cualquier valor fuera de ese rango no será válido, por tanto, lo convertimos en “null” (luego trataremos todos estos valores juntos).

He decidido crear un nuevo atributo en cada registro con el número de campos nulos que contiene, quizás podamos descartar algunos registros si presentan un número de registros nulos elevado, y también añadimos un campo “id” para identificar cada uno de los registros, esto en caso de awk no tiene mucho sentido, ya que NR nos indica en todo momento la línea en la que estamos y podemos imprimirla siempre, pero me parece que puede ser de utilidad un identificador.

Ya que hemos hecho estas transformaciones, podemos aprovechar y mostrar los datos con mayor número de registros nulos, se mostrará un top 20 en la consola, con su identificador y el número de atributos nulos.

Otro dato interesante puede ser el número de registros nulos por atributo, para imprimir esto por pantalla en forma de gráfico con gnuplot me he visto obligado a crear un archivo con estos datos (plot_nulls.csv, al final lo eliminaremos), introducir comandos awk dentro de gnuplot me ha producido distintos errores y como el nuevo archivo va a ser muy pequeño, no creo que suponga un problema. Por tanto, veremos por pantalla lo siguiente:



NOTA: Finalmente todas las gráficas se guardan en archivos .png sin mostrarse por pantalla, por si se lanzará desde una terminal sin modo gráfico, como me dijeron por correo.

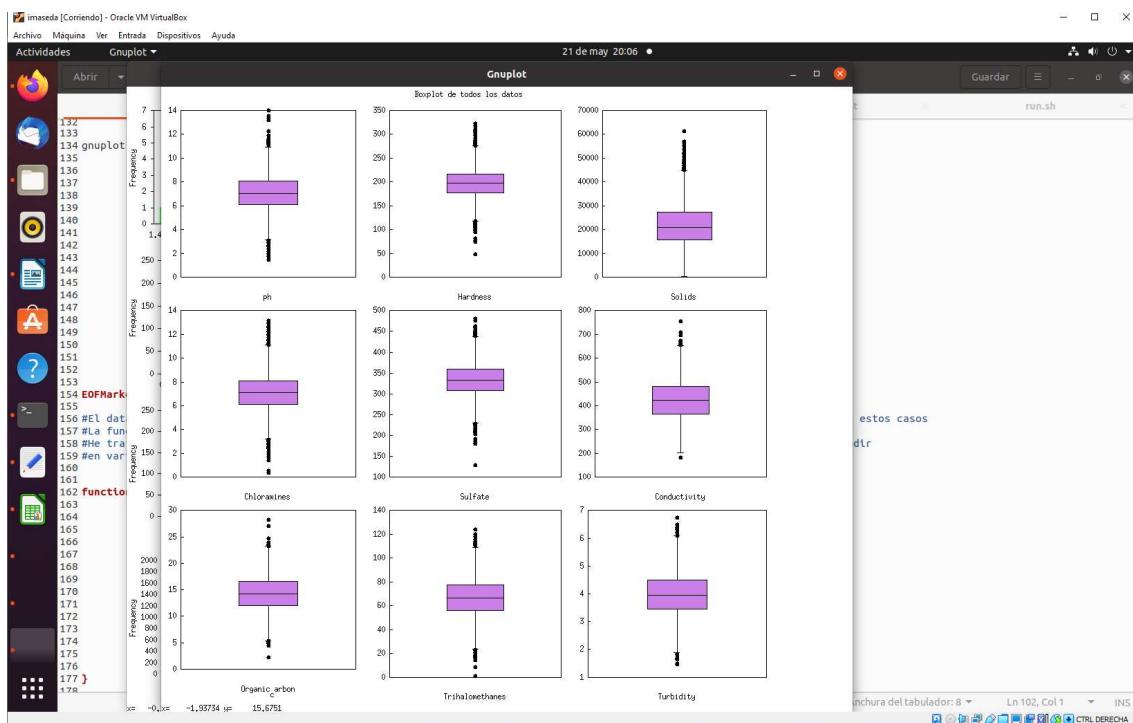
Antes de esto he decidido crear un archivo sin la etiqueta “potable” (0-1), es decir, nos quedamos con los 9 atributos descriptores, sin clase, el nuevo archivo se llamará “data_numeric.csv”.

Lo siguiente, y dado que estamos presentando los datos, he creído interesante mostrar con gnuplot una gráfica con la distribución de todos los atributos, no es una transformación, pero es el comienzo de cualquier proceso de tratamiento de datos y me ha parecido interesante investigar sobre gnuplot. He consultado diferentes guías y documentación, existen distintas maneras y seguro que mejores de hacerlo, pero he decidido iterar sobre las columnas creando un plot para cada una, ajustando los valores según su máximo o mínimo (obtenido con “stats”) y generar una pequeña función para darle el nombre de la variable a cada gráfico, juntamos todos los gráficos con “multiplot” y se obtiene un resultado bastante limpio (está dividido en 50 barras cada gráfico).

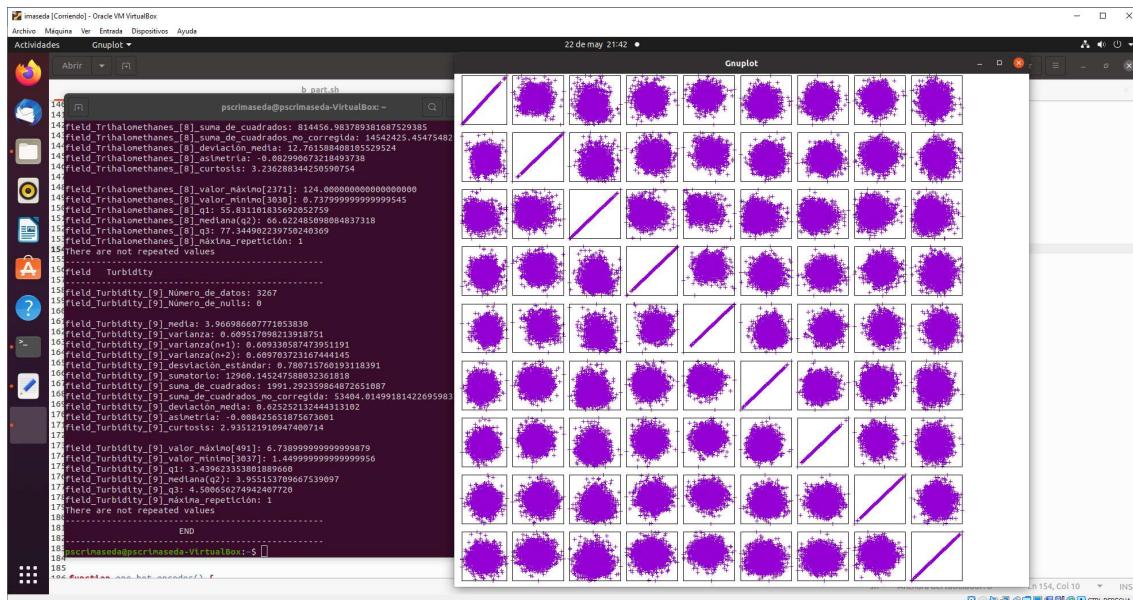
También probé a generar un boxplot para visualizar outliers, seguimos una lógica similar, pero esta vez con “style boxplot” y con los datos numéricos, en este caso no tiene sentido visualizar la clase, aplicamos la misma función para los nombres de los atributos y lo imprimimos con multiplot, se podría imprimir en un solo gráfico, pero debido a la diferencia entre los rangos de los valores en los distritos atributos, he creído que sería mejor así, y vemos cada uno escalado en su rango de valores.

Los resultados de ambos gráficos son los siguientes:





El último gráfico será una matriz de dispersión un poco rudimentaria pero que puede cumplir su función, hay un cambio en el estilo comentado, pero creo que las cruces por defecto se visualizan mejor:



Siguiendo con las transformaciones, aun siendo un dataset con datos numéricos exclusivamente, me ha parecido interesante abordar el caso de atributos categóricos, una transformación muy común y necesaria para algunos modelos es la sustitución de estos por valores numéricos y separarlos en distintas columnas con valores binarios (one hot encoder), he creado una función para llevar a cabo dicha transformación (`one_hot_encoder()`), y he probado su funcionamiento con el atributo de valores nulos creados, tendría más sentido hacerlo con la etiqueta, pero así podemos comprobar que lo hace para cualquier número de categorías, he tratado de hacerlo lo más genérico posible, para poder reutilizar dicha función en otros conjuntos de datos. También se aporta comentado una parte de código en la que se

crean variables en formato texto y hace lo mismo, siempre buscando como he dicho, la reutilización del código en un futuro. Aplicando dicha función a la columna null, que puede tomar 4 valores [0-3] obtenemos el siguiente resultado:

A screenshot of the LibreOffice Calc application window. The title bar reads "data_one_hot_encoder.csv - LibreOffice Calc". The menu bar includes Archivo, Editar, Ver, Insertar, Formato, Estilos, Hoja, Datos, Herramientas, Ventana, Ayuda. The toolbar has various icons for file operations, cell selection, and data processing. The spreadsheet contains 38 rows of data across 20 columns, labeled A1 to V38. Column A contains IDs from 1 to 38. Columns B through V contain numerical values representing different features. The data includes several null entries and some very large numbers. The bottom status bar shows "Hoja 1 de 1", "Predeterminado", "Español (España)", and "Promedio: Suma: 0".

Crearás tantas columnas como valores pueda tomar la columna que introducimos como argumento.

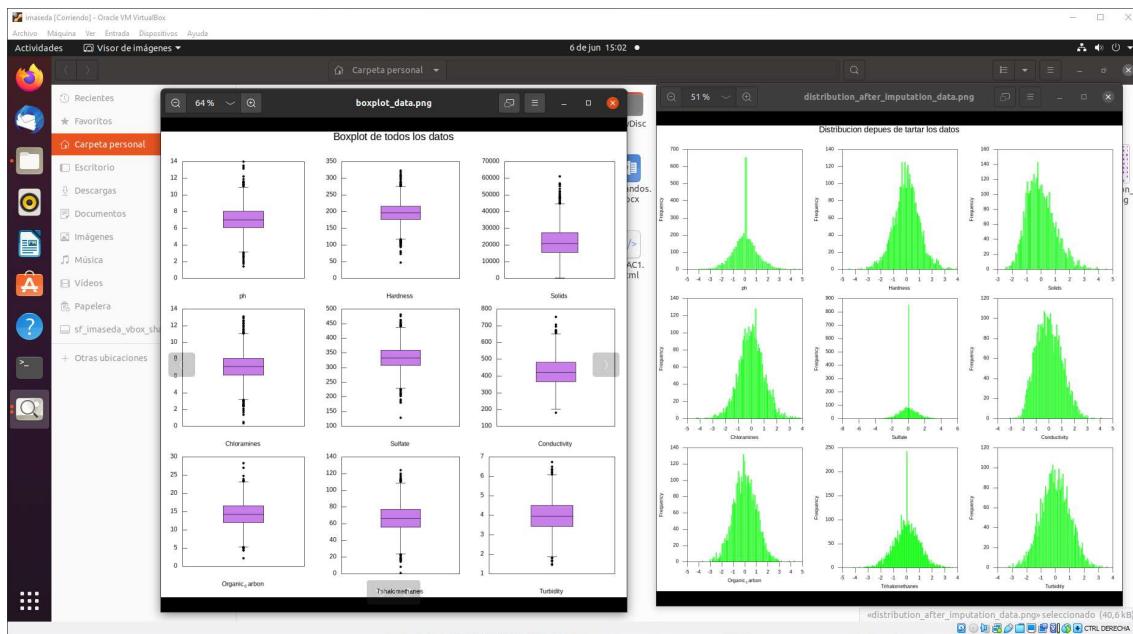
Finalmente, ya que hemos analizado el número de valores nulos para cada registro, me ha parecido correcto eliminar las filas con 3 valores nulos, representan un porcentaje mínimo de los datos y pueden ser contraproducentes aplicando distintos modelos.

Se aporta una parte de código comentada, para visualizar los plots directamente, si se dispone de modo gráfico:

```
3djun 2000 •
b_part.sh

230 cat -o, -t1-9 data_tr.csv > data_numeric.csv
231
232 #se apunta comentando la generación de dos de los plots anteriores en pantalla
233
234 #gnuplot -geometry 1000x1000 -persist <<-EOFMarker
235 #    set terminal x11
236 #    set key autotitle columnhead
237 #    unset key
238 #    set datafile separator ","
239 #    set datafile numeric.csv"
240 #    set multiplot layout 9,9 rowsfirst
241 #    set ytics auto
242 #    set size 0.8,1
243 #    plot data using 1:2 pt 7
244 #    #no me gustó por puntos, pero creo que se ve mejor por defecto:
245 #    #set style line 1 lc rgb "#0000ad" pt 7
246 #    do for [i=1:9] [
247 #        do for [j=1:9] [
248 #            set size 0.12,0.12
249 #            unset xtics
250 #            unset ytics
251 #            plot data using i:j # para imprimir con puntos-> w p ls 1 con set style sin comentar
252 #        ]
253 #    ]
254 #    unset multiplot
255 #    unset output
256 #EOFMarker
257 #gnuplot -geometry 1000x1000 -persist <<-EOFMarker
258 #    set terminal x11
259 #    data="data_tr.csv"
260 #    set key autotittle columnhead
261 #    unset key
262 #    set datafile separator ","
263 #    set title
264 #    set multiplot layout 4,3 rowsfirst title "Distribución de todos los datos originales\n" font ",20"
265 #    do for [i=1:10] [
266 #        getTitle=(columnNum)=system(sprintf("head -n1 '%s' | cut -f%d -d ','", data, colNum+1))
267 #        stats data using i nooutput
268 #        n=$0
269 #        max=STATS_max
270 #        min=STATS_min
271 #        width=(max-min)/n
272 #        hist(x,width)width*floor(x/width)+width/2.0
273 #        set xrange [min,max]
274 #        set yrange [0,1]
275 #        set offset graph 0.05,0.05,0.05,0.0
276 #        set xtics min,(max-min)/5,max
277 #        set boxwidth width*0.9
```

Como he mencionado anteriormente, los gráficos se guardan en archivos .png:



También he decidido incluir en este script la ejecución del segundo script “c_awk.awk” que nos ofrecerá mucha información sobre los datos y explicamos a continuación.

Apartado c (c_awk.awk y c_bash.sh):

Empezaré por “c_awk.awk”, este script no es una transformación de los datos originales como tal, lo que hace es obtener información de los datos haciendo operaciones matemáticas comunes en el tratamiento de datos. Como he comentado al principio, lo normal es utilizar las funciones sencillas que nos ofrecen todos los lenguajes de programación, pero ha sido todo un reto crear el script, me ha supuesto muchos quebraderos de cabeza y he tratado de hacerlo de forma óptima para no tener que iterar muchas veces sobre el total de los datos. Mediante este script obtenemos el número de registros total, el número de registros nulos, la media, la varianza (N , $N+1$ y $N+2$), desviación estándar, sumatorio de valores, suma de cuadrados ($x_i - \bar{x}$), suma de cuadrados no corregida, desviación media, coeficiente de asimetría, coeficiente de curtosis, valor máximo [y el número de fila en que se encuentra], valor mínimo [y le número de fila], los cuartiles q_1 , q_2 (mediana) y q_3 y la moda, en caso de no existir valores repetidos nos lo indica por pantalla, en caso de existir, nos dice el valor y las veces que se repite.

En principio este script iba a guardar los valores en un nuevo archivo para poder acceder a ellos cuando quisiéramos, pero al final creo que sería de bastante utilidad contar con un script que nos imprimiera esta información por pantalla, luego en el script “c_bash.sh” se crea una versión modificada de este script para guardar estos valores y poder acceder a ellos para las transformaciones que haremos. Como he dicho, ha sido un auténtico reto hacerlo en un solo comando awk y tratando de no iterar muchas veces sobre los datos, para esto hemos tenido que generar una matriz con todos los datos (mat[i,NR]), ya que para ciertas operaciones debemos haber iterado al menos una vez sobre los datos (debemos saber la media para calcular la varianza...), guardamos esta matriz en el bloque BEGIN, y en el bloque END iteramos sobre la misma para hacer los cálculos necesarios, también debemos crear otra matriz con los datos ordenados para obtener la moda y los cuartiles, para ello empleamos assort (assorti para claves) para ordenar los valores de un array de cada columna y lo vamos eliminando en cada iteración de columna. Ha sido un proceso complejo y como he dicho, no es una transformación como tal, pero si nos sirve para objetivo que es tartar los valores nulos y normalizar el conjunto de datos, también es el comienzo básico para abordar un dataset de esta tipología.

Este es el script que ejecutamos en “b_part.sh” para presentar los datos. Y la salida es un formato así para cada atributo:

```

Statistics of numeric data:
Field pb
-----
field_ph [1] Número_de_datos: 2781
field_ph [1] Número_de_nulos: 486

field_ph [1] media: 7.099190472908337149
field_ph [1] varianza: 2.482847974977679346
field_ph [1] varianza(n+1): 2.481955500000000219472
field_ph [1] varianza(n+2): 2.481955500000000219472
field_ph [1] desviación estándar: 1.575705458330993674
field_ph [1] sumatorio: 19717.81970515808643404
field_ph [1] suma_de_cuadrados: 6994.888218413482070900
field_ph [1] desviación media: 1.23817402177381550
field_ph [1] desviación media: 1.23817402177381550
field_ph [1] asimetría: 0.1256993105379491
field_ph [1] curtosis: 3.458910915200625880

field_ph [1] valor_máximo[2070]: 13.99999999999999824
field_ph [1] valor_mínimo[2894]: 1.431781554742741491
field_ph [1] q1: 6.09615418692717283
field_ph [1] mediana(q2): 7.038347516788062919
field_ph [1] q3: 10.03510517144
field_ph [1] Maxima_repetición: 1
There are not repeated values

Field Hardness
-----
field_Hardness [2] Número_de_datos: 3267
field_Hardness [2] Número_de_nulos: 0

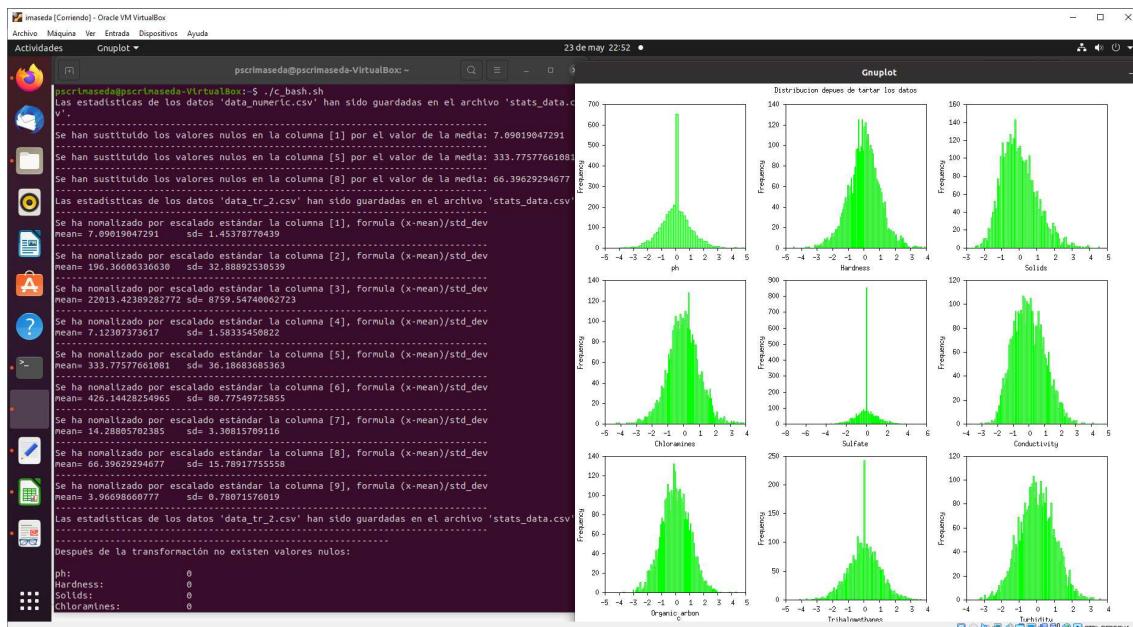
field_Hardness [2] media: 196.366063366303592375
field_Hardness [2] varianza: 1081.081407743716818004
field_Hardness [2] varianza(n+1): 1081.350415880882110287
field_Hardness [2] varianza(n+2): 1082.01260229595975510
field_Hardness [2] desviación_estándar: 32.8889253053952225
field_Hardness [2] sumatorio: 641527.929377138729799

```

El siguiente script, “c_bash.sh” si es una transformación de los datos, en este caso, como he dicho, he tenido que introducir una versión modificada del script anterior, en este caso en una función (`save_statistics()`), para poder acceder a estos datos en cualquier momento. Una vez definida la función, primero, para el tratamiento de datos nulos, he decidido crear 4 funciones (`mean_replace()`, `median_replace()`, `mode_replace()` y `value_replace()`) para sustituir los registros que contengan la cadena de texto “null” por su media, mediana, moda o un valor introducido como argumento, respectivamente, obviamente no podemos aplicar todas las transformaciones, por eso hay partes de código comentadas, en caso de optar por una u otra estrategia simplemente tendríamos que aplicar la función deseada. En todas las funciones introducimos el archivo que contiene los datos que queremos transformar y la columna objetivo (no trata el dataset en conjunto porque no me parece lo apropiado, me parece correcto seleccionar las columnas concretas), accede a las estadísticas de los datos previamente calculadas y realiza dicha transformación. En todos los casos sobrescribe el archivo que le pasemos como argumento, por eso es importante conservar los datos originales, y nos indica el archivo de origen y el de destino por pantalla (consola).

En cuanto a la normalización, se ofrecen dos funciones(`norm_max_min()` y `norm_standard()`), una por máximo-mínimo y otra normalización estándar $((x_i - \bar{x})/sd)$, en ambos casos se accede al archivo con los cálculos realizados previamente y se realiza la operación seleccionada para cada valor. He intentado optimizar todas las funciones y hacerlas genéricas para cualquier conjunto de datos, obviamente los valores nulos deben estar representados por la cadena de texto “null”, y en todos los casos nos ofrece información de los valores de media, sd, max o min con los que se han realizado los cálculos.

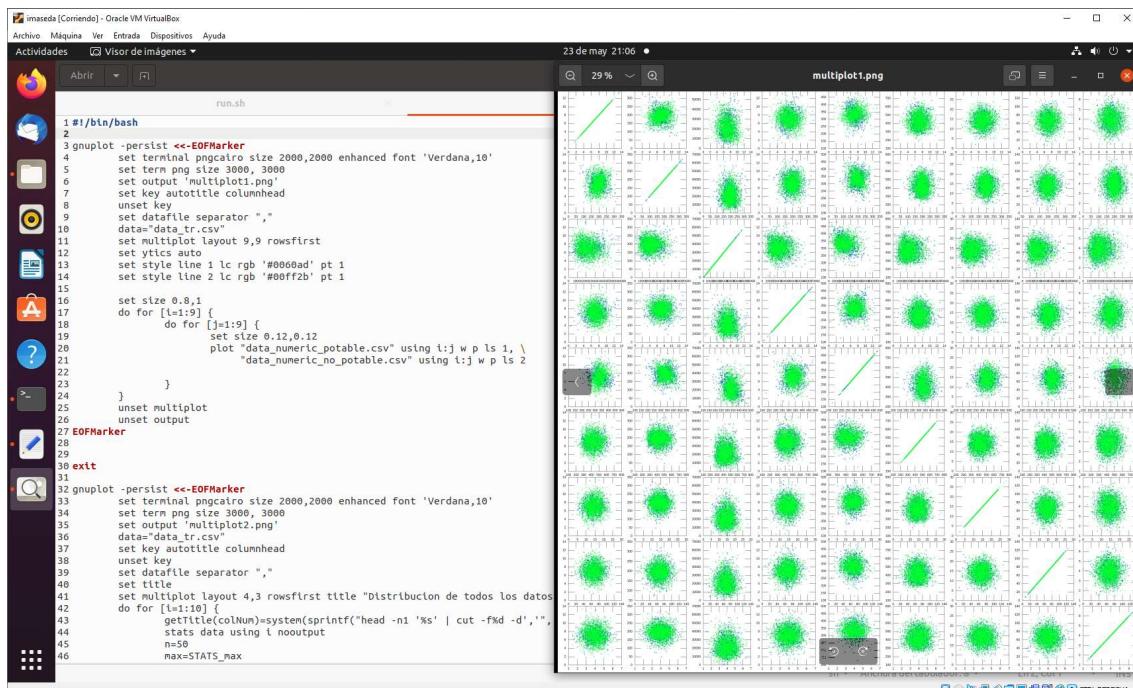
Finalmente comprobamos que ya no hay valores nulos en nuestro conjunto de datos, mostrándolo por pantalla (consola), y vemos la distribución de los valores una vez normalizados. En el script he decidido aplicar las funciones `mean_replace()` y `norm_standard()`, como he dicho, hay partes de código comentadas, si se quieren probar otras funciones bastaría con hacer pequeñas variaciones en los comentarios del script. Vemos una captura de pantalla con los resultados:



Apartado d (d.sh):

En este apartado, dado la tipología de los datos y el método de obtención, he decidido aprovechar la propia salida de los datos generada por mis scripts. En teoría se trata de resultados agregados, filtrados por atributos y además he decidido diferenciar por etiquetas, es decir, potable o no potable, esto nos ofrece una visión global de los datos según su etiqueta y poder visualizar a simple vista si existieran diferencias significativas.

Pensé en filtrar por rangos de valores o hacer algún tipo de distinción, pero los datos son bastante similares independientemente de la clase, no hay una correlación marcada entre algunos valores ni un atributo determinante a primera vista, habría que estudiar a fondo los datos. Podemos ver un script sencillo para mostrar con gnuplot:



```

#!/bin/bash
gnuplot -persist <<-EOFMarker
set terminal pngcairo size 2000,2000 enhanced font 'Verdana,10'
set term png size 3000, 3000
set output "multiplot1.png"
set key autotitle columnhead
unset key
set datafile separator ","
data="data_tr.csv"
set multiplot layout 9,9 rowsfirst
set ytics auto
set style lline 1 lc rgb '#0000ad' pt 1
set style lline 2 lc rgb '#00ff2b' pt 1
set size 0.8,1
do for [i=1:9] {
    do for [j=1:9]
        set size 0.12,0.12
        plot "data_numeric_potable.csv" using i:j w p ls 1,
             "data_numeric_no_potable.csv" using i:j w p ls 2
}
unset multiplot
unset output
EOFMarker
exit
gnuplot -persist <<-EOFMarker
set terminal pngcairo size 2000,2000 enhanced font 'Verdana,10'
set term png size 3000, 3000
set output 'multiplot2.png'
data="data_tr.csv"
set key autotitle columnhead
unset key
set datafile separator ","
set title
set multiplot layout 4,3 rowsfirst title "Distribucion de todos los datos"
do for [i=1:10] {
    getTitle(column)=system(sprintf("head -n1 '%s' | cut -f%d -d ','",
                                     noutput
                                     n=50
                                     max=STATS_max
)

```

Por eso he decidido finalmente filtrar los datos obtenidos mediante mi script “c_awk.awk”, podemos ver en la siguiente captura el formato de la salida de texto:

```

pruebas3.sh
#!/bin/bash
#<!/bin/bash>
#<body>
<h1>WATHER QUALITY DATASET</h1>
#<h2>Este conjunto de datos seleccionado nos ofrece una serie de métricas de distintas muestras de agua y si son potables o no, al ser un conjunto de datos heterogéneo ofrece muchas posibilidades de presentar los filtrados, tenemos el único atributo categórico es el campo Potability, agua es potable o no, por tanto, he decidido aprovechar los datos nulos para poder aplicarlos a la muestra total con las diferencias que existen entre los datos potables y no potables, de esta forma obtener un formato más rápido y eficiente.
<h3>WATHER QUALITY DATASET</h3>
<table border="1">
<tr><td style="text-align:center;color:white;background-color:black;" colspan="3">AQUA NO POTABLE</td></tr>
<tr><td>input1=a.txt</td>
<td>input2=b.txt</td>
<td>i=1</td>
<td>name_old1=</td>
<td>name_old2=</td>
<td>while IFS= read -r linea1 && IFS= read -r linea2 < &3
    do
        name1=<echo $linea1 | grep "field_</td> | cut -d " -f1 | tr -d "\:>; sed 's/field_</td> /<td> /g'>;
        data1=<echo $linea1 | grep "field_</td> | cut -d " -f1 | tr -d "\:>; sed 's/field_</td> /><td> /g'>;
        value1=<echo $linea1 | grep "field_</td> | cut -d " -f2>;
        name2=<echo $linea2 | grep "field_</td> | cut -d " -f1 | tr -d "\:>; sed 's/field_</td> /><td> /g'>;
        data2=<echo $linea2 | grep "field_</td> | cut -d " -f1 | tr -d "\:>; sed 's/field_</td> /><td> /g'>;
        value2=<echo $linea2 | grep "field_</td> | cut -d " -f2>;
        tf [ -z '$name1' ]
        then
            continue
        elif [ -z '$name2' ]
        then
            echo
            <tr><td style="background-color:LightGrey">$value1</td></tr>
        else
            echo
            <tr><td style="text-align:center;color:white;background-color:black;" rowspan="19"><span style="background-color:#cccccc;float:left;white-space:pre;position: absolute; top: 0; left: 0; width: 100%; height: 100%; z-index: 1;></span><span style="float: right; position: absolute; top: 0; right: 0; width: 100%; height: 100%; z-index: 2;></span></td><td>$name1</td>
            <td>$name2</td>
            <td>$value1</td>
        fi
    done
</td></tr>
</table>
</body>
</html>

```

Para cada atributo sigue la misma estructura con líneas en blanco y guiones.

```

Statistics of numeric data:
field_ph
-----
field_ph[1] Número_de_datos: 2781
field_ph[1] Número_de_nulos: 486

field_ph[1] media: 7.096190472908337149
field_ph[1] varianza: 2.482847974977879346
field_ph[1] varianza(n+1): 2.481955506259339472
field_ph[1] varianza(n+2): 2.483741085760245326
field_ph[1] sumatorio: 19717.819705158088435404
field_ph[1] suma_de_cuadrados: 6904.800218413482070900
field_ph[1] suma_de_cuadrados_mo: corregida: 146707.897638450289377943
field_ph[1] desviación_media: 1.230176402177381556
field_ph[1] q1: 6.096154118692717283
field_ph[1] mediana(q2): 7.038347516788062919
field_ph[1] q3: 8.663487890130517144
field_ph[1] máxima_repetición: 1
There are not repeated values

field Hardness
-----
field_Hardness[2] Número_de_datos: 3267
field_Hardness[2] Número_de_nulos: 0

field_Hardness[2] media: 196.366063366303592375
field_Hardness[2] varianza: 1081.681407743716818004
field_Hardness[2] varianza(n+1): 1081.350415880882110287
field_Hardness[2] varianza(n+2): 1082.01260229595975510
field_Hardness[2] desviación_estándar: 32.888925380539295225
field_Hardness[2] sumatorio: 641527.929017713875509799
field_Hardness[2] suma_de_cuadrados: 553853.37558972506315456
field_Hardness[2] suma_de_cuadrados_mo: corregida: 1295988167.119844317436218262
field_Hardness[2] desviación_media: 25.40205234377844883
field_Hardness[2] asimetría: 0.038281503574845651
field_Hardness[2] curtosis: 3.458910915200625880
field_Hardness[2] valor_máximo[2070]: 13.999999999999998224
field_Hardness[2] valor_minimo[2894]: 1.431781554742741491
field_Hardness[2] q1: 6.096154118692717283
field_Hardness[2] mediana(q2): 7.038347516788062919
field_Hardness[2] q3: 8.663487890130517144
field_Hardness[2] máxima_repetición: 1
There are not repeated values

```

Tenemos que lidiar con líneas en blanco y dividir las líneas de diferentes formas para obtener los resultados deseados, además de leer los dos archivos de forma simultánea para generar la tabla en html, creo que habría sido incluso más sencillo hacerlo desde un csv, ya que filtrar valores, columnas o filas con awk en un tipo de archivo ordenado es relativamente sencillo, obviamente aquí también existen patrones que permiten hacerlo, si no habría que especificar una condición prácticamente para cada línea, pero al final creo que el resultado es satisfactorio, aunque quizás debí elegir otro conjunto de datos, obtenerlo desde una API...Pero una vez hechos todos los scripts ya no disponía de tiempo.

Como he dicho, dividimos los datos del archivo “data_tr.csv” según su etiqueta y seleccionamos las columnas deseadas para después aplicar el script mencionado (“c_awk.awk”) y generar una archivo de texto para cada uno (“a.txt” y “b.txt”), luego generamos los marcadores en html que se imprimirán después por pantalla y darán lugar a nuestro archivo html, iteramos sobre los archivos de texto mencionados, lo hacemos simultáneamente para no hacer dos tablas, hacer una que contenga ambos datos y poder visualizarlos mejor, filtramos por cada línea los valores deseados, guardándolos en las variables name, data y value (1 y 2 en todos los casos), y ponemos la condición de las líneas en blanco para no generar celdas vacías, luego he decidido añadir otra condición para unificar celdas, obviamente sabemos que cada atributo consta de 19 datos, de ahí el valor “rowspan=19”, que genera una columna que une 19 celdas, hasta que el atributo name cambia de nombre y las vuelve a generar, es sencillo porque ambas tienen el mismo número de datos, por eso comprobamos únicamente name1, de ser distintas habríamos tenido que introducir otras condiciones y probablemente habría sido mucho más complejo. Por último, añadimos una serie de detalles visuales para hacerlo más atractivo, rellenamos algunas celdas, centramos algunos textos y ponemos en negrita algunas celdas, he decidido cambiar el contraste con letras blancas y fondo negro en los atributos y cabecera porque creo que visualmente ayudan a comprender mejor la tabla en un primer contacto. Vemos una captura del formato final:

WATHER QUALITY DATASET																																															
	AGUA POTABLE																																														
ph	<table border="1"> <thead> <tr> <th>Número de datos</th> <td>1099</td> </tr> <tr> <th>Número de nulls</th> <td>178</td> </tr> <tr> <th>media</th> <td>7.085548677844964338</td> </tr> <tr> <th>varianza</th> <td>2.022284405303553356</td> </tr> <tr> <th>varianza(n+1)</th> <td>2.020445964935095429</td> </tr> <tr> <th>varianza(n+2)</th> <td>2.024126194379421761</td> </tr> <tr> <th>desviación estándar</th> <td>1.422070464253988087</td> </tr> <tr> <th>sumatorio</th> <td>7787.017996951615714352</td> </tr> <tr> <th>suma de cuadrados</th> <td>2222.490561428604905814</td> </tr> <tr> <th>suma de cuadrados mo corregida</th> <td>57397.785634084189950954</td> </tr> <tr> <th>desviación media</th> <td>1.102541780936219240</td> </tr> <tr> <th>asimetría</th> <td>0.128817960589128283</td> </tr> <tr> <th>curtosis</th> <td>3.525569224707134186</td> </tr> <tr> <th>valor máximo[15]</th> <td>13.175401724233024581</td> </tr> <tr> <th>valor mínimo[193]</th> <td>1.757037115490782675</td> </tr> <tr> <td>q1</td> <td>6.185924892052045543</td> </tr> <tr> <td>mediana(q2)</td> <td>7.039094321089830686</td> </tr> <tr> <td>q3</td> <td>7.935607325113078758</td> </tr> <tr> <td>máxima repetición</td> <td>1</td> </tr> <tr> <td>Número de datos</td> <td>1277</td> </tr> <tr> <td>Número de nulls</td> <td>0</td> </tr> <tr> <td>media</td> <td>195.841856669781634537</td> </tr> <tr> <td>varianza</td> <td>1261.431994538311982978</td> </tr> </thead></table>	Número de datos	1099	Número de nulls	178	media	7.085548677844964338	varianza	2.022284405303553356	varianza(n+1)	2.020445964935095429	varianza(n+2)	2.024126194379421761	desviación estándar	1.422070464253988087	sumatorio	7787.017996951615714352	suma de cuadrados	2222.490561428604905814	suma de cuadrados mo corregida	57397.785634084189950954	desviación media	1.102541780936219240	asimetría	0.128817960589128283	curtosis	3.525569224707134186	valor máximo[15]	13.175401724233024581	valor mínimo[193]	1.757037115490782675	q1	6.185924892052045543	mediana(q2)	7.039094321089830686	q3	7.935607325113078758	máxima repetición	1	Número de datos	1277	Número de nulls	0	media	195.841856669781634537	varianza	1261.431994538311982978
Número de datos	1099																																														
Número de nulls	178																																														
media	7.085548677844964338																																														
varianza	2.022284405303553356																																														
varianza(n+1)	2.020445964935095429																																														
varianza(n+2)	2.024126194379421761																																														
desviación estándar	1.422070464253988087																																														
sumatorio	7787.017996951615714352																																														
suma de cuadrados	2222.490561428604905814																																														
suma de cuadrados mo corregida	57397.785634084189950954																																														
desviación media	1.102541780936219240																																														
asimetría	0.128817960589128283																																														
curtosis	3.525569224707134186																																														
valor máximo[15]	13.175401724233024581																																														
valor mínimo[193]	1.757037115490782675																																														
q1	6.185924892052045543																																														
mediana(q2)	7.039094321089830686																																														
q3	7.935607325113078758																																														
máxima repetición	1																																														
Número de datos	1277																																														
Número de nulls	0																																														
media	195.841856669781634537																																														
varianza	1261.431994538311982978																																														
ph	<table border="1"> <thead> <tr> <th>Número de datos</th> <td>1682</td> </tr> <tr> <th>Número de nulls</th> <td>308</td> </tr> <tr> <th>media</th> <td>7.093223369920629295</td> </tr> <tr> <th>varianza</th> <td>2.783751787055090965</td> </tr> <tr> <th>varianza(n+1)</th> <td>2.782097745589223514</td> </tr> <tr> <th>varianza(n+2)</th> <td>2.78540779644657338</td> </tr> <tr> <th>desviación estándar</th> <td>1.668457906887402142</td> </tr> <tr> <th>sumatorio</th> <td>11930.801708206498005893</td> </tr> <tr> <th>suma de cuadrados</th> <td>4682.27050582666283919</td> </tr> <tr> <th>suma de cuadrados mo corregida</th> <td>89310.112004366281325929</td> </tr> <tr> <th>desviación media</th> <td>1.313525937197633775</td> </tr> <tr> <th>asimetría</th> <td>0.103167948489466685</td> </tr> <tr> <th>curtosis</th> <td>3.3334882579353863390</td> </tr> <tr> <th>valor máximo[1271]</th> <td>13.999999999999998224</td> </tr> <tr> <th>valor mínimo[1776]</th> <td>1.431781554742741491</td> </tr> <tr> <td>q1</td> <td>6.039759379118511795</td> </tr> <tr> <td>mediana(q2)</td> <td>7.038219911170481957</td> </tr> <tr> <td>q3</td> <td>8.158055238038359747</td> </tr> <tr> <td>máxima repetición</td> <td>1</td> </tr> <tr> <td>Número de datos</td> <td>1990</td> </tr> <tr> <td>Número de nulls</td> <td>0</td> </tr> <tr> <td>media</td> <td>196.702451281609171474</td> </tr> <tr> <td>varianza</td> <td>966.044427012453866155</td> </tr> </thead></table>	Número de datos	1682	Número de nulls	308	media	7.093223369920629295	varianza	2.783751787055090965	varianza(n+1)	2.782097745589223514	varianza(n+2)	2.78540779644657338	desviación estándar	1.668457906887402142	sumatorio	11930.801708206498005893	suma de cuadrados	4682.27050582666283919	suma de cuadrados mo corregida	89310.112004366281325929	desviación media	1.313525937197633775	asimetría	0.103167948489466685	curtosis	3.3334882579353863390	valor máximo[1271]	13.999999999999998224	valor mínimo[1776]	1.431781554742741491	q1	6.039759379118511795	mediana(q2)	7.038219911170481957	q3	8.158055238038359747	máxima repetición	1	Número de datos	1990	Número de nulls	0	media	196.702451281609171474	varianza	966.044427012453866155
Número de datos	1682																																														
Número de nulls	308																																														
media	7.093223369920629295																																														
varianza	2.783751787055090965																																														
varianza(n+1)	2.782097745589223514																																														
varianza(n+2)	2.78540779644657338																																														
desviación estándar	1.668457906887402142																																														
sumatorio	11930.801708206498005893																																														
suma de cuadrados	4682.27050582666283919																																														
suma de cuadrados mo corregida	89310.112004366281325929																																														
desviación media	1.313525937197633775																																														
asimetría	0.103167948489466685																																														
curtosis	3.3334882579353863390																																														
valor máximo[1271]	13.999999999999998224																																														
valor mínimo[1776]	1.431781554742741491																																														
q1	6.039759379118511795																																														
mediana(q2)	7.038219911170481957																																														
q3	8.158055238038359747																																														
máxima repetición	1																																														
Número de datos	1990																																														
Número de nulls	0																																														
media	196.702451281609171474																																														
varianza	966.044427012453866155																																														

NOTA: Tras probar distintas opciones en la obtención de los datos por clase para mejorar el rendimiento:

```

3 #awk 'BEGIN {FS=OFST=","} NR==1{print $0} NR>1{if ($10==1) {print $0}}' data_tr.csv | cut -d, -f1-9 > data_numeric_potable.csv
4 #awk 'BEGIN {FS=OFST=","} NR==1{print $0} NR>1{if ($10==0) {print $0}}' data_tr.csv | cut -d, -f1-9 > data_numeric_no_potable.csv
5
6 ./c_awk.awk data_numeric_potable.csv > a.txt
7 ./c_awk.awk data_numeric_no_potable.csv > b.txt
8
9 awk 'BEGIN {FS=OFST=","} NR==1{print $0} NR>1{if ($10==1) {print $0}}' data_tr.csv | cut -d, -f1-9 | ./c_awk.awk - > a.txt
10 awk 'BEGIN {FS=OFST=","} NR==1{print $0} NR>1{if ($10==0) {print $0}}' data_tr.csv | cut -d, -f1-9 | ./c_awk.awk - > b.txt

58
59 done < $input1 3>$input2
58 #done < $(awk 'BEGIN {FS=OFST=","} NR==1{print $0} NR>1{if ($10==1) {print $0}}' data_tr.csv | cut -d, -f1-9 | ./c_awk.awk -) 3< $(awk 'BEGIN {FS=OFST=","} NR==1{print $0} NR>1{if ($10==0) {print $0}}' data_tr.csv | cut -d, -f1-9 | ./c_awk.awk -)
51

```

Sorprendentemente obtenía tiempos de ejecución menores si creaba los archivos a.txt y b.txt antes de leerlos que introduciendo los comandos awk en el propio bucle “while”, por eso es la opción elegida, al final de run.sh se borrará los archivos a.txt y b.txt por considerarlos innecesarios.

Apartado e (run.sh):

En este script simplemente se ejecutan los scripts anteriores, he añadido comentarios explicando el proceso, también se emplea el comando sleep en 2 ocasiones, para no mostrar de golpe todas las gráficas y datos. Comenté anteriormente que decidí no generar archivos .png para los plots porque ya se crean suficientes .csv a lo largo de la ejecución, además no requiere un tiempo de ejecución elevado, y tienen un título que los identifica, aún así, decidí añadir “sleep” para introducir 2 pequeñas pausas.

NOTA: Finalmente los comandos sleep han sido eliminados y las gráficas se generan en archivos .png siguiendo las recomendaciones que me han dado por correo y para no exceder el tiempo de ejecución permitido.

El proceso es como ya hemos explicado, primero se hace un tratamiento mínimo de los datos y se muestra información, tanto en texto como en gráficas, luego se pasa a tratamiento de valores nulos y normalización, y por último se genera el informe en html por clases.

```
./run.sh
```

IMPORTANTE (Aclaraciones): Es necesario tener el archivo descargado, se ha añadido el enlace, pero se necesita ser usuario en Kaggle, por eso no he añadido un comando directo de descarga. El nombre que se da al archivo original en los scripts es el que viene por defecto en la descarga “wáter_potability.csv”, sería raro que algo cambiara en tan poco tiempo.

También es necesario la instalación de gnuplot en la máquina virtual, se adjunta captura y comando:

```
sudo apt-get install gnuplot

[imaseda [Corriendo] - Oracle VM VirtualBox]
Archivo Maquina Ver Entrada Dispositivos Ayuda
Actividades Terminal 14 de may 13:06 • *c_bash.sh Guardar
Abrir
*pscrimaseda@pscrimaseda-VirtualBox: ~
1 #!/bin/bash
2 #!/usr/bin/python
3 #!/usr/bin/python3
4 #!/usr/bin/python2
5 #!/usr/bin/python
6 tr Leyendo lista de paquetes... Hecho
7 Creando árbol de dependencias
8 #S Leyendo la información de estado... Hecho
9 awk se instalarán los siguientes paquetes adicionales:
10 aglfn gnuplot data gnuplot-gt libdouble-conversion3 libqt5core5a libqt5dbus5 libqt5gui5 libqt5networks
11 libqt5printsupport5 libqt5svg5 libqt5widgets5 libwxbase3.0-0v5 libwxgtk3.0-gtk3-0v5 libxcb-xinerama0
12 libxcb-xinerama0 libxcb-xinput0 qt5-gtk-platformtheme qtxtranslations5-l10n
13 awk Paquetes superiores
14 awk gnuplot-doc qt5-image-formats_qtwayland5
15 sose Instalarán los siguientes paquetes NUEVOS:
16 aglfn gnuplot gnuplot-data gnuplot-gt libdouble-conversion3 libqt5core5a libqt5dbus5 libqt5gui5
17 set libqt5networks libqt5printsupport5 libqt5svg5 libqt5widgets5 libwxbase3.0-0v5 libwxgtk3.0-gtk3-0v5
18 set libxcb-xinerama0 libxcb-xinput0 qt5-gtk-platformtheme qtxtranslations5-l10n
19 set actualizados, 18 nuevos se instalarán, 0 para eliminar y 10 no actualizados.
20 se necesita descargar 16,6 MB de archivos.
21 plase utilizarán 69,0 MB de espacio de disco adicional después de esta operación.
22 ¿Desea continuar? [S/n] s
23 Des:1 http://es.archive.ubuntu.com/ubuntu focal/universe amd64 libdouble-conversion3 amd64 3.1.5-4ubuntu1 [3
24 7,9 kB]
Des:2 http://es.archive.ubuntu.com/ubuntu focal/universe amd64 libqt5core5a amd64 5.12.8+dfsg-0ubuntu1 [2.00
25 5 kB]
Des:3 http://es.archive.ubuntu.com/ubuntu focal/universe amd64 libqt5dbus5 amd64 5.12.8+dfsg-0ubuntu1 [208 k
B]
Des:4 http://es.archive.ubuntu.com/ubuntu focal/universe amd64 libqt5networks5 amd64 5.12.8+dfsg-0ubuntu1 [67
26 4 kB]
Des:5 http://es.archive.ubuntu.com/ubuntu focal/main amd64 libxcb-xinerama0 amd64 1.14-2 [5.266 B]
Des:6 http://es.archive.ubuntu.com/ubuntu focal/main amd64 libxcb-xinput0 amd64 1.14-2 [29.3 kB]
Des:7 http://es.archive.ubuntu.com/ubuntu focal/universe amd64 libqt5gui5 amd64 5.12.8+dfsg-0ubuntu1 [2.971
kB]
Des:8 http://es.archive.ubuntu.com/ubuntu focal/universe amd64 libqt5widgets5 amd64 5.12.8+dfsg-0ubuntu1 [2.
293 kB]
Des:9 http://es.archive.ubuntu.com/ubuntu focal/universe amd64 libqt5svg5 amd64 5.12.8-0ubuntu1 [131 kB]
Des:10 http://es.archive.ubuntu.com/ubuntu focal/universe amd64 aglfn all 1.7-gt20191031.4036a9c2 [30.6 kB]
Des:11 http://es.archive.ubuntu.com/ubuntu focal/universe amd64 gnuplot-data all 5.2.8+dfsg1-2 [56.5 kB]
Des:12 http://es.archive.ubuntu.com/ubuntu focal/universe amd64 libqt5printsupport5 amd64 5.12.8+dfsg-0ubuntu
1 [193 kB]
```

Captura de la ejecución de “run.sh”:

The screenshot shows a Linux desktop environment with several windows open:

- Terminal:** The terminal window is titled "pscrimaldesa@pscrimaldesa-VirtualBox: ~" and contains the command "RIPF + ./run.sh". The output of the script is displayed, showing statistics for various water quality parameters.
- Browser:** A browser window displays a data frame with columns "mediana(q2)", "q3", "máxima repetición", "Número de datos", "Número de nulls", "media", "varianza", "varianza(n+1)", "varianza(n+2)", "desviación estándar", "sumatorio", "suma de cuadrados", "suma de cuadrados mo corregida", "desviación media", "asimetría", "curtosis", "valor máximo[371]", "valor mínimo[1912]", "q1", "mediana(q2)", "q3", "máxima repetición", "Número de datos", "Número de nulls", "media", "varianza", "varianza(n+1)", "varianza(n+2)", "desviación estándar", "sumatorio", "suma de cuadrados", "suma de cuadrados mo corregida", "desviación media", "asimetría", and "curtosis".
- File Manager:** A file manager window titled "ethanes" is visible on the left side of the screen.

Terminal Output (run.sh):

```
los registros con mayor numero de valores nulos son:  
ID: 903 number nulls: 3  
ID: 559 number nulls: 3  
ID: 647 number nulls: 3  
ID: 388 number nulls: 3  
ID: 3068 number nulls: 3  
ID: 3063 number nulls: 3  
ID: 2066 number nulls: 3  
ID: 1023 number nulls: 3  
ID: 142 number nulls: 3  
ID: 971 number nulls: 2  
ID: 912 number nulls: 2  
ID: 870 number nulls: 2  
ID: 828 number nulls: 2  
ID: 802 number nulls: 2  
ID: 812 number nulls: 2  
ID: 797 number nulls: 2  
ID: 765 number nulls: 2  
ID: 747 number nulls: 2  
ID: 738 number nulls: 2  
ID: 710 number nulls: 2  
  
el numero de valores nulos por campo es:  
ph: 495  
Hardness: 6  
Solids: 0  
Chloramines: 0  
Sulfate: 781  
Conductivity: 0  
Organic_Carbon: 0  
Trihalomethanes: 162  
Turbidity: 0  
  
Statistics of numeric data:  
field ph  
field_ph [1] Número de datos: 2781  
field_ph [1] Número de nulls: 486
```