

# Tarea 1

Paola Mejía

Otoño 2019

## 1. Problema 1

Para crear un archivo llamado problema-1.sh, se creó una carpeta nueva llamada “tarea1” y se utilizaron las siguientes instrucciones en la línea de comandos:

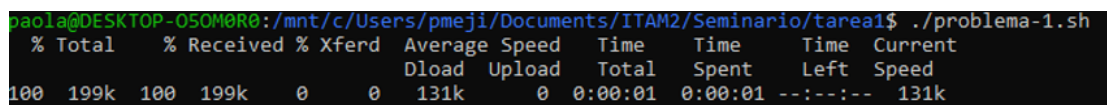
```
mkdir tarea1
cd tarea1
touch problema-1.sh
```

A continuación, el resto del ejercicio se realizó de acuerdo a las siguientes instrucciones:

1. Descarga el libro de The Time Machine de H. G. Wells,

```
#!/bin/bash
curl https://www.gutenberg.org/files/35/35-0.txt \
> time_machine.txt
```

El comando **curl** se usa para hacer peticiones de HTTP desde la línea de comandos. La figura 1 muestra los detalles de la descarga, incluyendo el tiempo de descarga, progreso y tamaño del archivo. El archivo descargado se guarda en un archivo de texto “time\_machine.txt”



```
paola@DESKTOP-OSOM0R0:/mnt/c/Users/pmeji/Documents/ITAM2/Seminario/tarea1$ ./problema-1.sh
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total   Spent    Left  Speed
100 199k 100 199k    0     0 131k      0  0:00:01  0:00:01 --:--:-- 131k
```

Figura 1: Información de descarga

2. Remover el encabezado (desde la primera línea hasta la línea que contiene \*\*\*START OF THIS PROJECT GUTENBERG EBOOK THE TIME MACHINE \*\*\*)

Una opción es utilizar el comando **tail** que muestra el final de un archivo. La “-n” indica que el siguiente argumento es el número de líneas que se deseen

obtener y el signo de “+” invierte el argumento de forma que en vez de devolver las últimas líneas, devuelve todo menos las primeras n líneas. Esta solución funciona cuando sabemos el número de línea en la cuál se encuentra la cadena que estamos buscando.

```
tail -n +23 time_machine.txt > time_machine_new.txt
```

Una alternativa es utilizar el comando **awk**. Para no guardar archivos temporales, podemos redirigir las entrada de la descarga del paso uno utilizando un pipe (—). El siguiente comando descarga el libro y redirige la salida a la entrada del comando **awk**. El comando **awk** busca la cadena “\*\*\* START OF THIS PROJECT GUTENBERG EBOOK THE TIME MACHINE \*\*\*” (también acepta otras expresiones regulares) y regresa todas las líneas después de la expresión. La salida es redirigida al archivo de texto “time\_machine.txt”.

```
#!/bin/bash
curl https://www.gutenberg.org/files/35/35-0.txt | \
awk 'f;/*** START OF THIS PROJECT GUTENBERG EBOOK \
THE TIME MACHINE   ***/{f=1}' \
>> time_machine.txt
```

Las instrucciones siguientes seguirán la misma estructura, es decir, se concatenaran las instrucciones a través de los pipes. Es decir, las instrucciones asumen que la salida de la instrucción anterior está redirigida para ser la entrada estándar.

3. Remover la licencia (desde \*\*\*END OF THIS PROJECT GUTENBERG EBOOK THE TIME MACHINE \*\*\* hasta el final del archivo)

Asumimos que la salida del inciso anterior se redirecciona a esta segunda instrucción. De nuevo usamos **awk**. **NR** indica el número de línea actual y después se indica en cuál patrón parar. El primer comando regresa el archivo hasta \*\*\*END OF THIS PROJECT GUTENBERG EBOOK THE TIME MACHINE \*\*\*, incluyendo esta última línea.

De forma similar a la primera estrategia en el inciso 2, se utiliza el comando **head** para regresar desde la fila 1 hasta la n-1. En otras palabras, elimina la última línea.

```
awk 'NR==1,/*** END OF THIS PROJECT GUTENBERG EBOOK \
THE TIME MACHINE   ***/' | \
head -n-1
```

4. Convertirlo a minúsculas.

Se utilizó el comando **awk** para imprimir en minúsculas el texto.

```
awk '{print tolower($0)}'
```

5. Extraer las palabras

El comando **gawk** es una implementación de **awk** más potente que soporta archivos gigantes. En este caso lo estamos usando por especificar el separador. Para extraer las palabras pudimos haberlas separado por espacios. Sin embargo, las comas o signos de puntuación hubieran causado problemas. Utilizando la expresión regular '[[:alpha:]]+' forzamos a que el separador sea cualquier secuencia de caracteres no alfabéticos.

```
gawk -v RS='[[:alpha:]]+' '{print}'
```

6. Ordenarlas

Se utilizó el comando **sort** que sirve para ordenar archivos. Puede ordenar por columnas, numérico, mes y aleatorio. Sin parámetros hace un ordenamiento ascendente.

```
sort
```

7. Eliminar duplicados y contarlos

El comando **uniq** identifica aquellos renglones consecutivos que son iguales. Con el parámetro “-c” cuenta el número de duplicados y los elimina.

```
uniq -c
```

8. Ordenar de mayor a menor

De nuevo se utilizó el comando **sort** con el parámetro “-r” que ordena de forma descendente.

```
sort -r
```

9. Mostrar el top 10 de palabras con su frecuencia

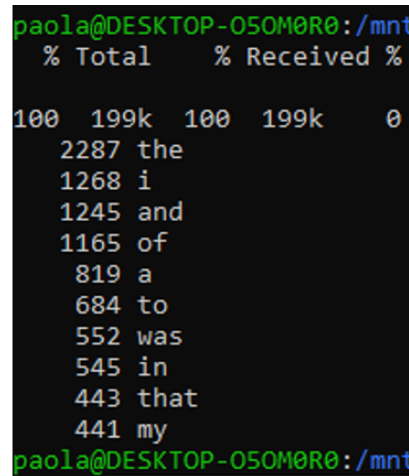
De nuevo se utilizó el comando **head** que muestra el principio del archivo.

```
head -n 10
```

Finalmente, el script queda así:

```
#!/bin/bash
curl https://www.gutenberg.org/files/35/35-0.txt | \
awk 'f;/** START OF THIS PROJECT GUTENBERG EBOOK THE \
TIME MACHINE ***/{f=1}' | \
awk 'NR==1,/** END OF THIS PROJECT GUTENBERG EBOOK \
THE TIME MACHINE ***/' | \
head -n1 | \
awk '{print tolower($0)}' | \
gawk -v RS='[[:alpha:]]+' '{print}' | \
sort | \
uniq -c | \
sort -r | \
head -n 10
```

La figura 2 muestra el despliegue del programa. Desde que descarga el archivo hasta que muestra las 10 palabras con mayor frecuencia.



```

paola@DESKTOP-050M0R0:/mnt
% Total    % Received %
100 199k 100 199k 0
2287 the
1268 i
1245 and
1165 of
819 a
684 to
552 was
545 in
443 that
441 my
paola@DESKTOP-050M0R0:/mnt

```

Figura 2: Despliegue de resultados

## 2. Problema 2

Usando el archivo UFO-Nov-Dic-2014.psv

1. ¿Cuántos avistamientos por estado hay? (Guárdalo en el archivo problema-2a.sh)

En primer lugar, el programa utilizando el comando **cat** manda el archivo UFO-Nov-Dic-2014.tsv a la salida estándar del comando **cut**. **cat** se utiliza para ver el contenido de un archivo mientras que **cut** sirve para dividir el archivo por columnas. El parámetro “-d” sirve para especificar el delimitador de las columnas. En este caso escogimos el delimitador “**^**” (tab); el siguiente parámetro -f indica los campos que deseamos seleccionar, en este ejemplo seleccionamos la columna 3.

En segundo lugar, se utilizó el comando **tail** para eliminar la primera fila que indicaba el nombre de columna.

En tercer lugar se utilizó el comando **awk** para eliminar las columnas vacías.

En cuarto lugar, se ordenaron los elementos de la columna con **sort** para después contar las ocurrencias con **sort**.

Finalmente, se ordenaron de forma descendiente y se guardaron en el archivo de texto “avistamientos\_edo.txt”.

El siguiente código muestra el funcionamiento del programa y la figura 3 muestra el despliegue de resultados.

```

#!/bin/bash
cat UFO-Nov-Dic-2014.tsv | \
cut -d$'\t' -f3 | \

```

```
tail -n +2 | \
awk NF | \
sort | \
uniq -c | \
sort -r | \
>avistamientos_edo.txt
```



```
paola@DESKTOP-050M0R0:/mnt/c/Users/pmej1
cat avistamientos_edo.txt | head -n 10
123 CA
87 FL
43 WA
42 AZ
38 PA
34 TX
34 NY
32 CO
31 NC
27 GA
paola@DESKTOP-050M0R0:/mnt/c/Users/pmej1
```

Figura 3: Despliegue de resultados

2. ¿Cuántos avistamientos no tienen forma de esferoide? (Guárdalo en el archivo problema-3a.sh)

**Respuesta: 924**

De forma similar al inciso anterior, se selecciono una columna del archivo UFO-Nov-Dic-2014.tsv. En este caso, se seleccionó la columna cuatro, se volvió a eliminar el nombre de la columna y las columnas vacías.

El comando **grep** nos permite buscar líneas que contengan un patrón específico y el parámetro **-v** invierte la selección. Es decir, selecciona todas las líneas que no hagan match. De esta forma seleccionamos las columnas que o tuvieran forma de esferoide (Sphere).

Finalmente se utilizó el comando **wc** para contar el número de palabras (con el parámetro **-w**). El siguiente fragmento de código muestra esta funcionalidad.

```
#!/bin/bash
cat UFO-Nov-Dic-2014.tsv | \
cut -d$'\t' -f4 | \
tail -n +2 | \
awk NF | \
grep -v 'Sphere' | \
wc -w
```

### 3. Problema 3

Guárdalo en el archivo problema-3.sh Usando el archivo UFO-Nov-Dic-2014.psv. Describe estadísticamente (max, min, mean) los tiempos de duración de la observación Bash programming La mayor parte del tiempo usaremos el shell, para hacer

pequeños scripts, pero existen ocasiones en las cuales es necesario tratar al shell como un lenguaje de programación

La estrategia fue solo utilizar las observaciones cuya respuesta tenía unidades de tiempo como segundos, minutos o horas. Los valores en horas y minutos se convirtieron en segundos para calcular estadísticas utilizando el siguiente fragmento de código:

```
#!/bin/bash
mean=`awk '{ total += $0; count++ } END { print total/count }' all_seconds`
min=`awk 'BEGIN{a=1000}{if ($0<0+a) a=$0} END{print a}' all_seconds`
max=`awk 'BEGIN{a= 0}{if ($0>0+a) a=$0} END{print a}' all_seconds`
```

## 4. Problema 4

Crearemos un script que analice todos los datos (de 1990 a la fecha) de la página web de UFOs. El cascarón de este archivo se encuentra en scripts/ufo-analysis.sh. Úsalo como punto de partida. Agrega las funciones:

1. clean\_data Convierte a minúsculas, separador a —

- Convierte a minúsculas con la siguiente instrucción:

```
awk '{print tolower($0)}'
```

- Para agregar el separador —, se utilizó la siguiente instrucción:

```
tr -d '\r'
```

2. concat\_data Concatena todos los meses en un solo archivo.

- La función elimina los espacios en blanco al principio de los archivos y concatena los meses en un solo archivo llamado extracted. El siguiente fragmento de código muestra la función:

```
function concat_data(){
    tail -n +16 |\
    head -n -2 \
    >> extracted
}
```

3. calculate\_stats Calcula los conteos por estado, color, forma, año, mes y hora.

- Se creo la sguiente función auxiliar format\_stats para realizar los conteos.

```
function format_stats(){
    awk NF |\
    sort | \
    uniq -c |\
    sort
}
```

4. Crea un script de python o R que genere las gráficas de estos conteos, guarda estos archivos.

- Se creo un script de R con una función que crea recibe el nombre del archivo y guarda la gráfica de conteos.

A continuación, se muestran las gráficas de los conteos.

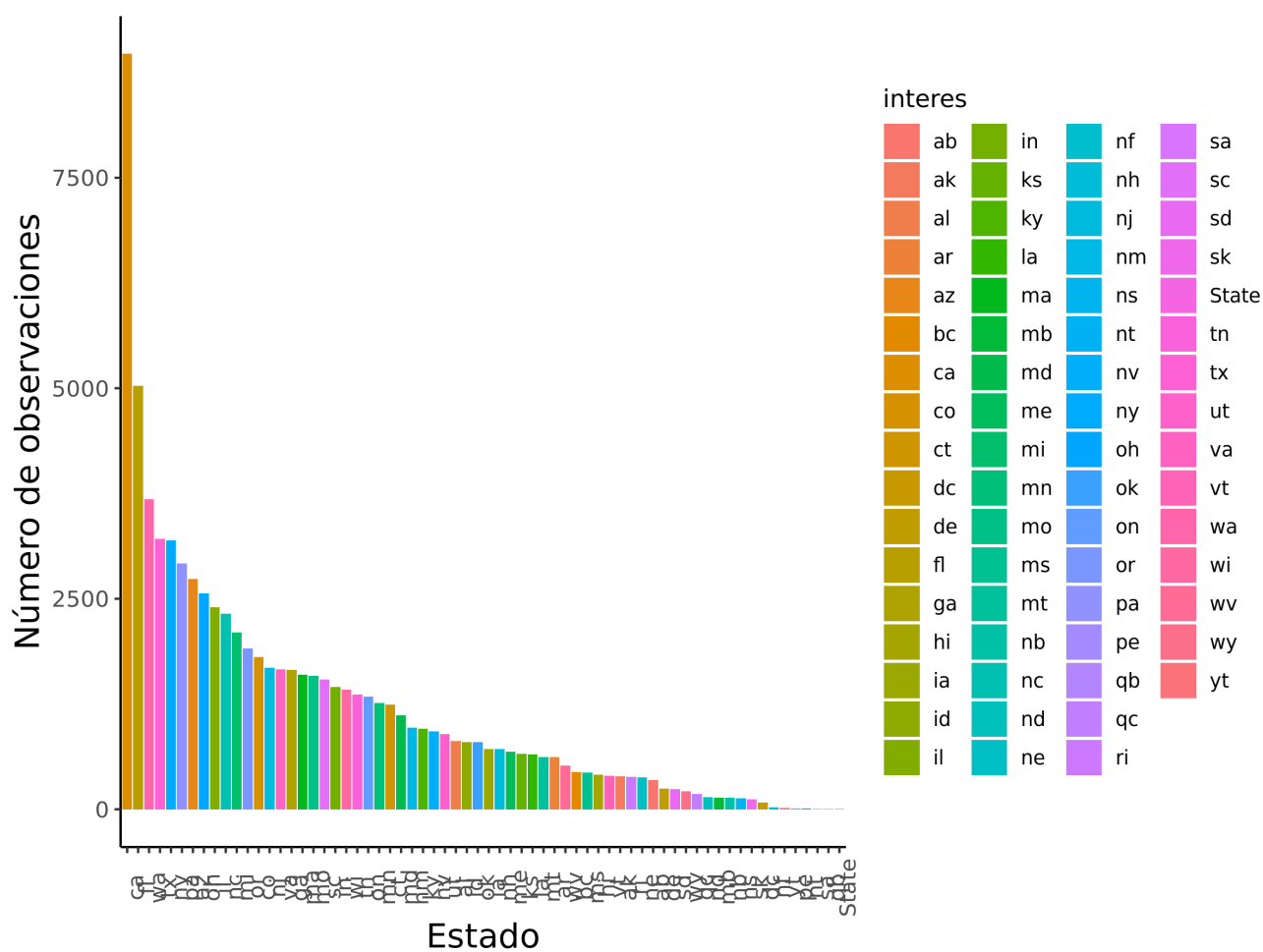


Figura 4: Conteo por estados

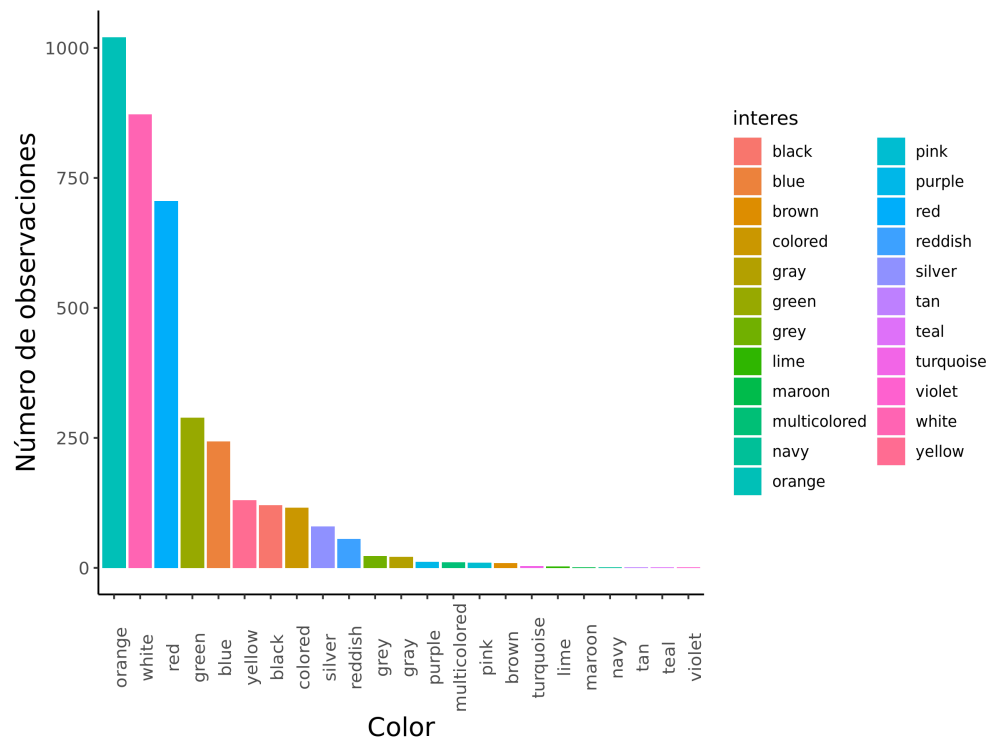


Figura 5: Conteo por color



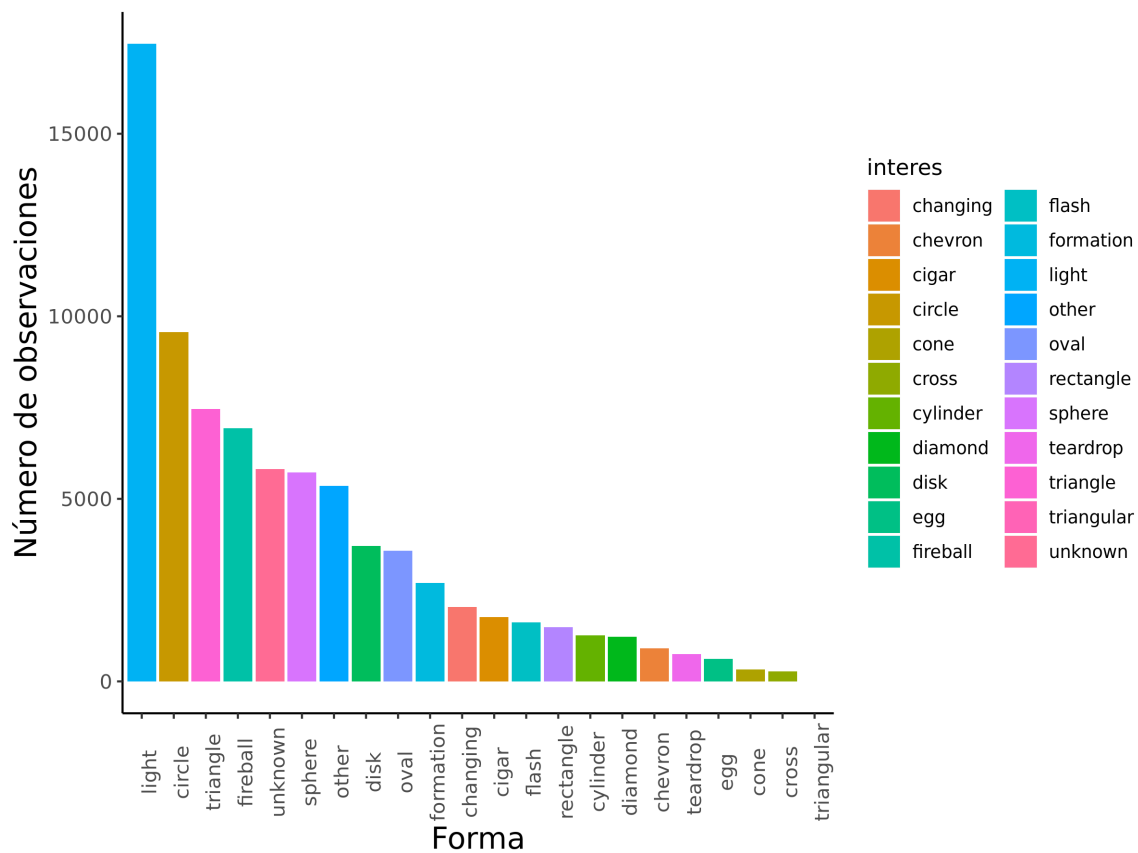


Figura 6: Conteo por forma

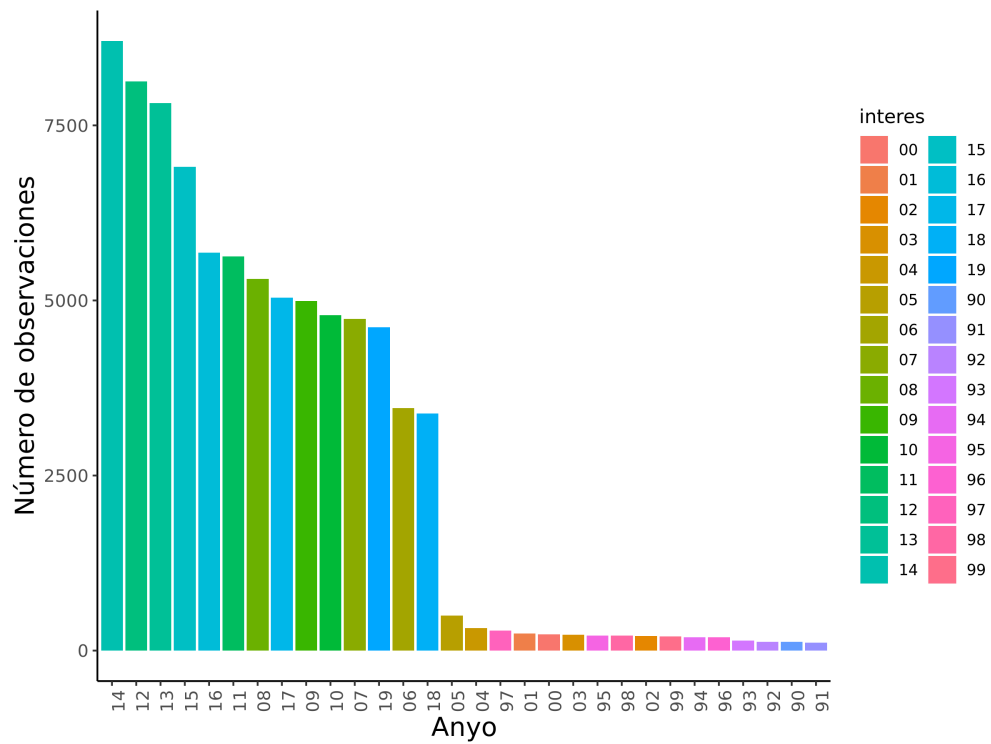


Figura 7: Conteo por año

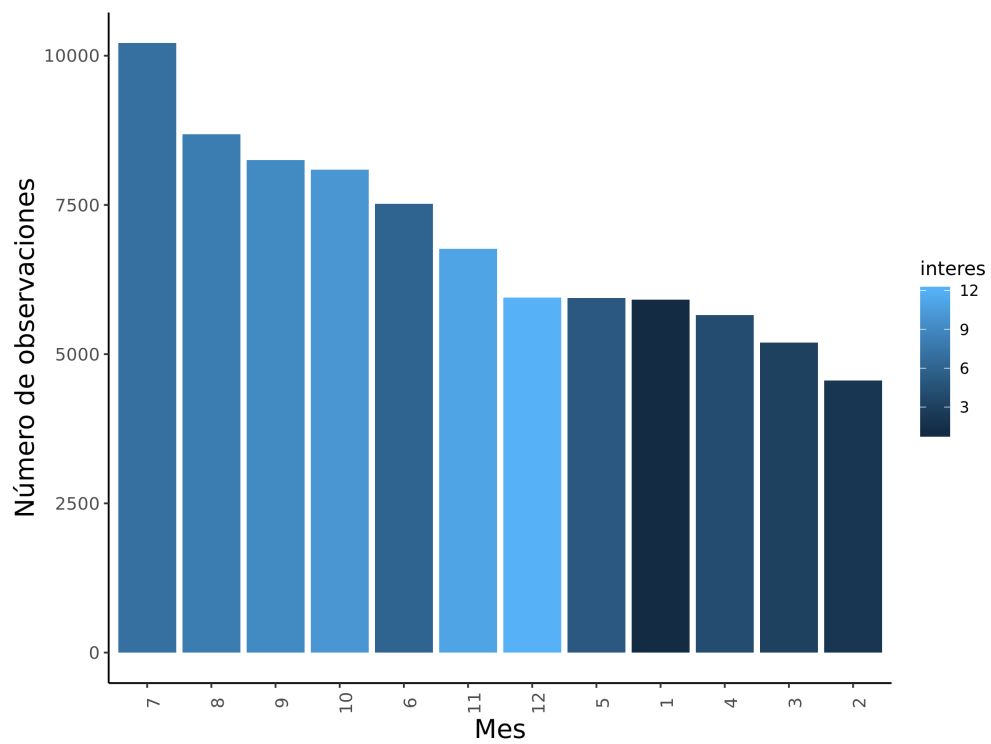


Figura 8: Conteo por mes

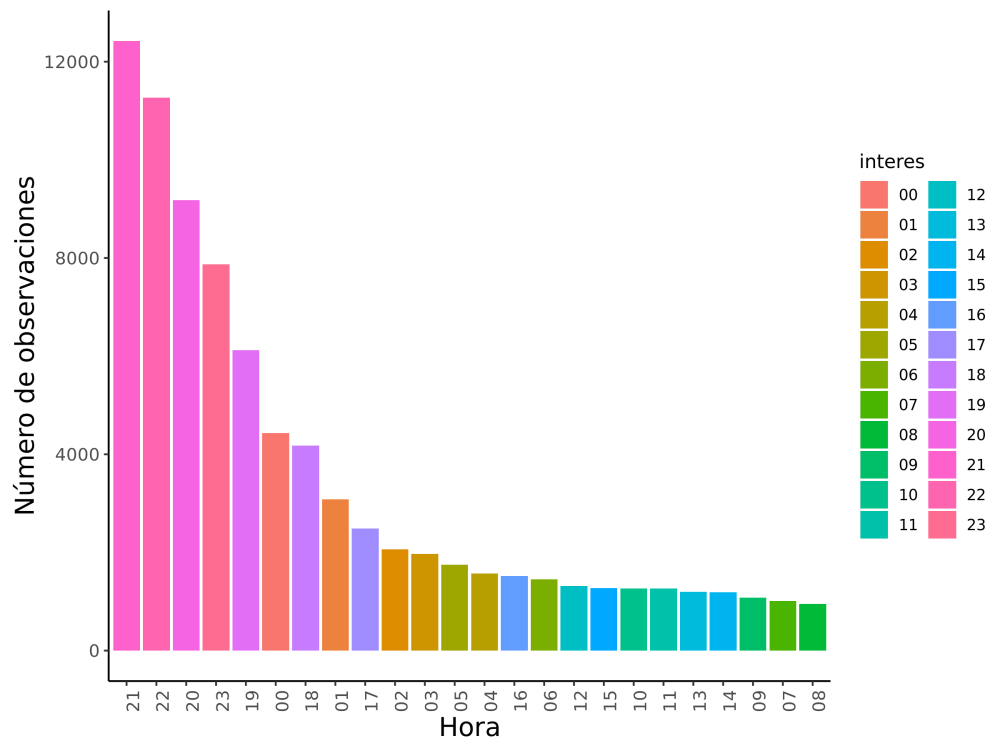


Figura 9: Conteo por hora