

Resumen comandos para la Ciencia de Datos

Gabriel Pinilla

9 de enero de 2024

Índice general

1. SQL en Postgres	1
1.1. Notas previas...	1
1.2. Comandos básicos	1
1.2.1. Estructura básica de una consulta	1
1.2.2. Where vs Having	2
1.2.3. Crear una base de datos	2
1.2.4. Crear una tabla	2
1.2.5. Insertar datos a una tabla	2
1.2.6. Ver todos los datos	2
1.3. Limpieza, orden y transformación de datos	3
1.3.1. Manipulación de filas y columnas	3
1.3.2. Orden	4
1.3.3. Conversión de tipos	4
1.4. Análisis y visualización	5
1.4.1. Cálculos matemáticos básicos	5
1.4.2. Número de caracteres de cada fila	5
1.4.3. Extraer un substring de un string	6
1.4.4. Devolver valores nulos	6
1.4.5. Subconsulta	6
1.4.6. Consultas en múltiples tablas	6
2. Terminal Windows	9
2.1. Comandos comunes	9
3. Excel y Google Sheets	11
3.1. Notas previas...	11
3.2. Comandos básicos	11
3.3. Limpieza, orden y transformación de datos	11
3.3.1. Filtrado	11
3.3.2. Orden	12
3.3.3. Quitar duplicados	12

3.3.4.	Funciones lógicas	12
3.3.5.	Manipulación de strings	12
3.3.6.	Formatos de fechas	13
3.3.7.	BURSCARV y BURSCARH	13
3.3.8.	Formato de datos	14
3.3.9.	Formato condicional	14
3.4.	Análisis y visualización	14
3.4.1.	Cálculos matemáticos básicos	14
3.4.2.	Cálculos con múltiples criterios	15
3.4.3.	Tablas dinámicas	15
3.4.4.	Creación de gráficos	15
3.5.	Importación de datos	15
3.6.	Errores	15
4.	Python	17
4.1.	Notas previas...	17
4.2.	Librerías	17
4.2.1.	Importación de datos	17
4.3.	Conceptos previos	18
4.3.1.	Manipulación de strings	18
4.3.2.	Estructuras de datos	20
4.4.	Limpieza, orden y transformación de datos	21
4.4.1.	Bibliotecas	21
4.4.2.	NumPy Arrays	22
4.4.3.	Tipos de datos de Pandas	23
4.4.4.	Importación datos	25
4.4.5.	Resúmenes de datos	25
4.4.6.	Manipulación y limpieza de datos	25
4.4.7.	Cruce de tablas	30
4.5.	Probabilidad y estadística	31
4.5.1.	Estadística	31
4.5.2.	Probabilidad	33
4.6.	Estadística inferencial	37
4.6.1.	Prueba de hipótesis	38
4.7.	Análisis y visualización	38
4.7.1.	Gráficas con matplotlib y seaborn	38
5.	R	49
5.1.	Notas previas...	49
5.2.	Paquetes	49
5.3.	Limpieza	50

5.3.1.	Resúmenes de datos	50
5.4.	Orden	50
5.5.	Transformación de datos	51
5.6.	Cálculos matemáticos	51
5.7.	Visualización	51
5.7.1.	Suavizado	53
5.7.2.	Estética y facetas	53
5.7.3.	Guardar gráficas	53
6.	Markdown	55
7.	Git y GitHub	57
7.1.	Git	57
7.1.1.	Repositorios	57
7.1.2.	.gitignore	58
7.1.3.	Ramas	58
8.	Tableau Public	61
8.1.	Creación de visualizaciones	61
8.1.1.	Importación de datos y conexiones	61
8.1.2.	Campos calculados	62
8.1.3.	Conceptos y elementos de Tableau	64
8.1.4.	Filtros	65
8.2.	Creación de visualizaciones básicas	67
8.3.	Diseño y personalización de paneles	73
8.3.1.	Funciones de agregación	74
8.3.2.	Funciones LOD	74
8.4.	Dashboards	75
8.4.1.	Creación de un dashboard	75
9.	Glosario de términos de programación	77

Capítulo 1

SQL en Postgres

1.1. Notas previas...

1. Al finalizar una consulta se debe terminar con ;.
2. El texto se debe escribir entre comillas simples o dobles, dependiendo del programa a utilizar.
3. En SQL, el primer caracter de un string inicia con ' y se escriben entre comillas simples.
4. Para insertar comentarios se debe usar --

1.2. Comandos básicos

1.2.1. Estructura básica de una consulta

La siguiente consulta muestra la estructura básica y el orden que deben seguir los comandos:

```
1 SELECT col-sep-por-coma
2 FROM conj-de-datos.tabla
3 WHERE condicion
4 GROUP BY columnas
5 HAVING condicion
6 ORDER BY columnas DESC
7 LIMIT numero;
```

Si se requieren todas las columnas de la tabla, se debe agregar un * después del comando **SELECT**.

1.2.2. Where vs Having

En esta consulta, la cláusula **WHERE** filtra y después el **GROUP BY** agrupa:

```
1 SELECT col-sep-por-coma
2 FROM conj-de-datos.tabla
3 WHERE condicion
4 GROUP BY columnas;
```

Sin embargo, en la siguiente consulta el **GROUP BY** agrupa y luego el **HAVING** filtra:

```
1 SELECT col-sep-por-coma
2 FROM conj-de-datos.tabla
3 GROUP BY columnas
4 HAVING condicion;
```

1.2.3. Crear una base de datos

Para crear una base de datos:

```
1 CREATE DATABASE nom-base-de-datos;
```

1.2.4. Crear una tabla

Para crear una tabla:

```
1 CREATE TABLE nom-tabla (
2     col1 tipo-de-variable,
3     col2 tipo-de-variable,
4 );
```

El comando **CREATE TABLE IF NOT EXISTS** crea una tabla si es que dicha tabla no existe en la base de datos.

1.2.5. Insertar datos a una tabla

Insertar datos a una tabla ya creada:

```
1 INSERT INTO tabla (col-sep-por-coma)
2 VALUES (valores-sep-por-coma);
```

1.2.6. Ver todos los datos

Para ver la tabla completa:

```
1 SELECT *
2 FROM conj-de-datos.tabla;
```


1.3. Limpieza, orden y transformación de datos

1.3.1. Manipulación de filas y columnas

Para eliminar registros:

```
1 DELETE FROM tabla
2 WHERE nom-col = condicion;
```

Agregar una columna:

```
1 ALTER TABLE tabla
2 ADD columna tipo-de-dato;
```

Eliminar una datos de una columna:

```
1 DELETE conj-de-datos.tabla
2 WHERE condicion;
```

Eliminar una columna:

```
1 ALTER TABLE tabla
2 DROP COLUMN columna;
```

Actualizar datos:

```
1 UPDATE conj-de-datos.tabla
2 SET nom-col = valor-a-cambiar
3 WHERE condicion;
```

Eliminar espacios en blanco:

```
1 SELECT columna
2 FROM conj-de-datos.tabla
3 WHERE
4 TRIM (columna)=criterio;
```

Borrar una base de datos:

```
1 DROP DATABASE base-de-datos;
```

Borrar una tabla:

```
1 DROP TABLE tabla;
```

Evitar selecciones duplicadas:

```
1 SELECT
2 DISTINCT (columna)
3 FROM conj-de-datos.tabla
4 WHERE condicion;
```

Para contar valores distintos:

```
1 SELECT COUNT(DISTINCT(col))
2 FROM conj-de-datos.tabla ;
```

¿Cuántas veces aparece cada elemento?

```
1 SELECT columna, COUNT (*)
2 FROM conj-de-datos.tabla
3 GROUP BY columnas;
```

1.3.2. Orden

Ordenar datos:

```
1 SELECT columnas
2 FROM conj-de-datos.tabla
3 ORDER BY columna DESC;
```

En las consultas SQL, la cláusula **WHERE** filtra los datos según la condición que se le otorgue. Se pueden ordenar los datos en orden ascendente (**ASC**) o descendente (**DESC**)

1.3.3. Conversión de tipos

Para convertir los datos de un tipo a otro se usa la función **CAST**:

```
1 SELECT
2 CAST(columna AS INT) \\
3 FROM conj-de-datos.tabla; \\
```

Los tipos de datos más comunes son:

1. **SMALLINT**
2. **INTEGER** o **INT**
3. **BIGINT**
4. **SERIAL**: Número entero que se autoincrementa.
5. **NUMERIC** o **DECIMAL**
6. **VARCHAR()**: Cadena de caracteres de longitud variable.
7. **TEXT**: Cadena de caracteres de longitud variable que no tiene límite.
8. **DATETIME**
9. **DATE**: Año, mes, día (formato standar).
10. **TIME**: Hora, minuto, segundo.
11. **TIMESTAMP**: Fecha y hora.

12. **BOOLEAN**: TRUE o FALSE.

Si la consulta anterior falla, se puede usar la siguiente consulta:

```
1 SELECT
2 SAFE_CAST(columna AS TIPO-DE-DATO)
3 FROM conj-de-datos.tabla;
```

1.4. Análisis y visualización

1.4.1. Cálculos matemáticos básicos

Suma (**SUM**) y promedio (**AVG**):

```
1 SELECT col, SUM(col2), AVG(col2)
2 FROM conj-de-datos.tabla
3 GROUP BY col;
```

Mínimo y máximo: Devuelve el valor mínimo de una columna con el nombre de min-columna. Lo mismo para el máximo.

```
1 SELECT MIN(columna) AS min-columna,
2 MAX(col2) AS max-columna
3 FROM conj-de-datos.tabla;
```

Conteo de registros:

```
1 SELECT COUNT (*)
2 FROM conj-de-datos.tabla;
```

La consulta anterior contará todos los registros de la tabla. Si se le agrega la cláusula **WHERE** al final, cuenta los registros dependiendo de la condición que tenga.

1.4.2. Número de caracteres de cada fila

Para que devuelva el número de caracteres:

```
1 SELECT
2 LENGTH(col-para-comprobar-su-largo)
3 FROM conj-de-datos.tabla;
```

También se puede escribir como:

```
1 SELECT columna
2 FROM conj-de-datos.tabla
3 WHERE
4 LENGTH(columna)=criterio;
```

En algunos programas usan **LEN**.

Ejemplo:

```
1 SELECT columna1
2 FROM conj-de-datos.tabla
3 WHERE
4 LENGTH(columna1)=5;
```

Esta consulta devolverá todos los registros de la columna 1 que tengan 5 caracteres.

1.4.3. Extraer un substring de un string

```
1 SELECT columna
2 FROM conj-de-datos.tabla
3 WHERE
4 SUBSTRS (col, inicio, num-de-letras-a-extraer) + criterio;
```

1.4.4. Devolver valores nulos

```
1 SELECT columna
2 FROM conj-de-datos.tabla
3 WHERE columna IS NULL;
```

1.4.5. Subconsulta

Es una consulta dentro de otra. Se ejecutan desde la más interna hacia la más externa y van entre paréntesis:

```
1 SELECT col-sep-por-coma
2 FROM conj-de-datos.tabla
3 WHERE col > (SELECT col-sep-por-coma
4              FROM conj-de-datos.tabla);
```

1.4.6. Consultas en múltiples tablas

Para hacer consultas en múltiples tablas, se puede hacer de dos maneras. La primera es usar **SELECT** y **FROM**:

```
1 SELECT tabla1.*, tabla2.*
2 FROM tabla1, tabla 2
3 WHERE tabla1.columna = tabla2.columna;
```

La segunda es usando **JOIN**, los cuales se usan para unir dos tablas bajo la siguiente sintaxis:

```
1 SELECT *  
2 FROM conj-de-datos.tabla1  
3 INNER JOIN tabla2  
4 ON tabla1.col= tabla2.col;
```

La primera tabla o tabla izquierda siempre irá junto al **FROM**, luego irá la segunda tabla o tabla derecha. Se lee *Tabla 1 hará un cruce con la tabla 2*.

Existen 4 formas de cruzar datos:

1. **INNER JOIN**: Devuelve los registros con valores coincidentes de ambas tablas. No muestra valores nulos.



Figura 1.1: Diagrama de Venn de INNER JOIN

2. **LEFT JOIN**: Devuelve todos los registros de la tabla izquierda y solo los registros coincidentes de la tabla derecha. Tanto en el **LEFT JOIN** como el **RIGHT JOIN** van a mostrar valores nulos. Si hace el cruce con columnas que tienen valores nulos, no cruza esos valores.



Figura 1.2: Diagrama de Venn de LEFT JOIN

3. **RIGHT JOIN**: Devuelve todos los registros de la segunda tabla y solo los coincidentes de la primera tabla.



Figura 1.3: Diagrama de Venn de RIGHT JOIN

4. **FULL JOIN**: Une todos los registros de las dos tablas, sean coincidentes o no.



Figura 1.4: Diagrama de Venn de FULL JOIN

5. **CROSS JOIN**: Combina cada uno de los registros de una tabla con los registros de la otra. No se hace sobre una clave, es decir, no va el **ON**. La tabla resultante tendrá un número de filas igual al producto entre los números de filas de cada tabla, repitiéndose los datos de estas. Debido a esto, ocupa más recursos.

Capítulo 2

Terminal Windows

2.1. Comandos comunes

1. \d tabla: muestra información de una tabla y los tipos de datos de cada columna.
2. \c base-de-datos: conectarse a una base de datos.
3. \dt: da una lista de todas las bases de datos.
4. \q: salir de la terminal.
5. nom-programa version: verifica la versión instalada del programa en el sistema.
6. clear: limpia la pantalla.

Capítulo 3

Excel y Google Sheets

3.1. Notas previas...

- Algunas funciones de Excel son parecidas a las de Google Sheets.
- En Google Sheets se usan ; y en Excel ,.
- Para los rangos se usa C:C, por ejemplo: =MAX(A3:A9).

3.2. Comandos básicos

La sintaxis de una función en Excel y Google Sheets es =NOM-FUNCION(argumento1; argumento2;...), donde los argumentos son los valores que se usan como entrada para la función.

Se puede definir la hoja de cálculo con 'nom-hoja'!rango, por ejemplo: =CONTAR.SI('hoja 1'!G:G). En este ejemplo toma toda la columna G de la hoja 1.

3.3. Limpieza, orden y transformación de datos

3.3.1. Filtrado

=FILTRAR(rango; fila-o-col = filtro;""): Devolverá todos los registros del rango, y si no hay devuelve una cadena vacía (""). Se pueden crear filtros en Excel: Datos > Filtro.

3.3.2. Orden

Para ordenar datos en hojas de cálculo, se debe seleccionar Datos > Ordenar. También se puede hacer mediante la función =ORDENAR().

En el siguiente comando, FALSO (o -1 en Excel) indica que el orden es descendiente. Por otro lado, VERDADERO (o 1 en Excel) señala orden ascendente.
=ORDENAR(rango-para-ordenar;segun-columna;FALSO)

Usando =ORDENARPOR():

=ORDENARPOR(rango;col-1; FALSO; col-2;VERDADERO)

3.3.3. Quitar duplicados

Para quitar duplicados en Excel: Datos > Quitar espacios duplicados

3.3.4. Funciones lógicas

Las funciones lógicas utilizadas en Excel y Google Sheets son:

- Y(expresion1;expresion2;...)
- O(expresion1;expresion2;...)
- NO(valor-logico)
- =SI(expresion;valor-si-verdadero;valor-si-falso)
- =SI.ERROR(valor;valor-si-error)

3.3.5. Manipulación de strings

1. =ESPACIOS(valores): Elimina espacios del texto.
2. =MAYUSC(celda): Cambia de minúscula a mayúscula.
3. =MINUSC(celda): Cambia de mayúscula a minúscula.
4. =NOMPROPIO(celda): Cambia a nombre propio.
5. =CONCATENAR(): Une dos o más cadenas de texto en una celda.
6. =ENCONTRAR(): Busca una cadena de texto y devuelve su posición.
7. =DERECHA(): Devuelve el número de caracteres iniciando desde la derecha.

8. =IZQUIERDA(): Devuelve el número de caracteres iniciando desde la izquierda.
9. =EXTRAE(): Devuelve un número específico de caracteres de una posición.
10. =LARGO(): Devuelve la longitud de una cadena de texto.
11. =REEMPLAZAR(): Reemplaza una cadena de texto por otra en una celda.

3.3.6. Formatos de fechas

1. =HOY(): Devuelve la fecha actual.
2. =AHORA(): Devuelve fecha y hora actual.
3. =FECHA(): Crea una fecha. Ej: =FECHA(2023,6,10).
4. =DIAS360(): Calcula el número de días entre dos fechas en un año de 365 días.
5. =DIAS.LAB(): Calcula el número de días laborales entre dos fechas.
6. =DIAS(): Calcula el número de días entre dos fechas.
7. =MES(): Devuelve el número de mes de una fecha.
8. =AÑO(): Devuelve el año de una fecha.
9. =DIASEM(): Devuelve el número de día de la semana de 1 a 7 para una fecha.
10. =DIAS.LAB.INTL(): Calcula el número de días laborales entre dos fechas usando una definición personalizada de días laborales.
11. =FIN.MES(): Devuelve la fecha del último día del mes antes o después de un número determinado de meses.

3.3.7. BURSCARV y BURSCARH

Para relacionar tablas y buscar datos de una tabla a partir de una clave de búsqueda, se usan los comandos BUSCARV y BUSCARH. La principal diferencia entre ellos es que BUSCARV solo realiza la búsqueda en una columna, mientras que BUSCARH lo hace con las filas.

Si se escribe FALSO, la coincidencia será exacta. Si se escribe VERDADERO, la coincidencia será cercana. En Excel 1 es verdadero y 0 es falso:

=BUSCARV(valor-buscado; rango-de-busqueda; num-para-indicar-col-de-busqueda; FALSO)

3.3.8. Formato de datos

Se usa la función =CONVERTIR() se usa para transformar un número de un sistema de medición a otro.

=CONVERTIR(num-a-convertir;unidad-del-num;unidad-resultado)

3.3.9. Formato condicional

El formato condicional es una herramienta que se utiliza para identificar tendencias resaltándolas con colores. Se le añade una condición para que cambie el aspecto de la celda.

Se puede aplicar el formato condicional de la siguiente forma:

Excel: Se debe seleccionar el rango de celdas donde se quiera aplicar el formato > Inicio > Formato condicional > Reglas para resaltar celdas.

Google Sheets: Se debe seleccionar el rango de celdas donde se quiera aplicar el formato > Formato > Formato condicional.

3.4. Análisis y visualización

3.4.1. Cálculos matemáticos básicos

1. =SUMA(rango)
2. =PROMEDIO(rango)
3. =MIN(rango)
4. =MAX(rango)
5. =RESIDUO(): Da como resultado el resto cuando al dividir dos números.
6. =CONTAR.SI(rango; "criterio"): Cuenta el número de celdas que cumplen con un criterio.
7. =SUMAR.SI(rango; "criterio"): Suma los valores de un rango si cumplen con el criterio.
8. =SUMAPRODUCTO(matriz1;matriz2;...): Multiplica las matrices y muestra el resultado de la suma de esos productos.

3.4.2. Cálculos con múltiples criterios

1. =SUMAR.SI.CONJUNTO(rango-suma;rango-criterio1;criterio1;rango-criterio2;criterio2;...)
2. =CONTAR.SI.CONJUNTO(rango-criterio1;criterio1;rango-criterio2;criterio2;...)
3. =MAX.SI.CONJUNTO(rango-max;rango1;criterio1;rango2;criterio2;...)

3.4.3. Tablas dinámicas

Una tabla dinámica o Pivot Table es una tabla que resume, calcula y analiza datos para observar tendencias entre ellos.

Para crear una tabla dinámica tanto en Excel como en Google Sheets, se debe seleccionar Insertar > Tabla dinámica, luego seleccionar los datos preferentemente limpios y con columnas.

3.4.4. Creación de gráficos

Para crear un gráfico tanto en Excel como en Google Sheets, se debe seleccionar Insertar > Gráfico. Se puede personalizar cambiando los colores, dándole nombre a los ejes y variar el tipo de gráfico.

3.5. Importación de datos

Si se desea importar datos de otras hojas de cálculo:

Excel: Datos>Obtener datos> Desde archivo > Desde libro> seleccionar archivo > Importar> seleccionar en el navegador la hoja de trabajo que se quiere importar > Cargar o Transformar datos.

Google Sheets: =IMPORTRANGE(), el cual permite especificar un rango de celdas en la otra hoja de cálculo para duplicarlo en la hoja que se esté trabajando.

3.6. Errores

- #DIV/0!: Fórmula que intenta dividir por cero un valor en una celda o por una celda vacía. Se soluciona con: =SI.ERROR(valor; valor-si-hay-error).
- #ERROR!: Error que solo devuelve Google Sheets. Señala que la fórmula no se puede interpretar tal como se ingresa, es decir, hay un error en la fórmula.

- #N/A: Indica que la hoja de cálculo no puede encontrar los datos de la fórmula. Ocurren generalmente cuando se usa la función =BUSCARV().
- #NAME? o #NOMBRE?: ocurre cuando el nombre de una fórmula no se reconoce.
- #NUM!: Señala que el cálculo de una fórmula no se puede realizar según lo especificado por los datos, como por ejemplo una fecha negativa.
- #VALUE!: Señala un problema con la fórmula o con las celdas con las que hace referencia.
- #REF!: Aparece cuando las celdas de una fórmula se han eliminado.

Capítulo 4

Python

4.1. Notas previas...

1. Python es un lenguaje de programación orientado a objetos de alto nivel.
2. Los tipos de datos en Python son int (números enteros), float (decimales), string (cadenas de texto) y booleanos.
3. Los strings se escriben entre comillas simples o dobles.
4. Los comentarios se hacen con el numeral (#). Los comentarios que contengan más de una línea se hacen en un bloque de tres comillas simples:

```
1 # Esto es un comentario
2
3 '''
4 Esto
5 es un
6 comentario
7 '''
```

4.2. Librerías

En ciencia de datos, las librerías más usadas son NumPy, Matplotlib y Pandas.

4.2.1. Importación de datos

Para importar las librerías, se utiliza el comando **import** y, usualmente, se le asigna un alias:

```
1 import libreria as alias
```

Si se quiere importar solo una función de una librería:

```
1 from libreria import funcion as alias
```

Ejemplo:

```
1 # Funcion que genera un numero aleatorio entre 0 y 10
2 import random
3 num = random.randint(0, 10)
4 print('El numero es ', num)
```

En Google Colaboratory, se importan los datos de drive con Pandas:

```
1 from google.colab import drive
2 drive.mount('/content/drive')
```

Luego, ir a Archivos > Drive > Copiar ruta de acceso. Se debe importar la librería Pandas con un alias:

```
1 import pandas as pd
```

Finalmente, para leer un archivo en formato csv:

```
1 nom_ds = pd.read_csv('ruta-del-archivo.csv')
```

4.3. Conceptos previos

4.3.1. Manipulación de strings

Los strings son cadenas de textos que contienen una secuencia de caracteres, como palabras. Se definen usando comillas simples o dobles y se pueden realizar operaciones con ellas.

Para interactuar con el usuario, se usan los comandos `print` y `input`. El primero imprime un mensaje en pantalla, mientras que el segundo permite que el usuario ingrese datos.

```
1 # Imprimir texto en pantalla
2 print('Hola mundo')
3
4 # Toma informacion y la puede guardar en una variable
5 nombre=input('Cual es tu nombre?')
6
7 # Sustituir variables en un string
8 print(f'Tu nombre es: {nombre}')
```



```
9
10 # Concatenacion
11 print('hola' + ' ' + 'mundo')
12 # salida: hola mundo
13
14 # Repeticion
15 print('Ja' * 4)
16 # salida: JaJaJaJa
```

Para transformar un número a string:

```
1 str(15)
2 # salida: 15
3
4 print('Quiero ' + str(15) + ' panes')
```

Y de un string a número:

```
1 edad = int(input('Cual es tu edad?'))
```

Operaciones con strings:

```
1 texto = 'Hola a todos'
2 # 1. lower(): convierte los caracteres a minusculas
3 texto_1 = texto.lower()
4 print(texto_1)
5 # salida: hola a todos
6
7 # 2. upper(): convierte los caracteres a mayusculas
8 texto_2 = texto.upper()
9 print(texto_2)
10 # salida: HOLA A TODOS
11
12 # 3. replace(sub, new_sub): reemplaza texto por uno
    nuevo
13 texto_3 = texto.replace('todos', 'Marta')
14 print(texto_3)
15 # salida: Hola a Marta
16
17 # 4. split(sep): divide una cadena en una lista con un
    separador
18 textonuevo = 'Hola, Josea'
19 lista = textonuevo.split(',')
20 print(lista)
```

```

21 # salida: ['Hola', 'Josefa']
22
23 # 5. strip(): elimina los espacios en blanco al inicio
    y final de una cadena
24 otrotexto = '    Hola    '
25 texto_4 = otrotexto.strip()
26 print(texto_4)
27 # salida: Hola

```

4.3.2. Estructuras de datos

Listas o arreglos: son secuencias de elementos ordenados por un índice, empezando por el 0.

```

1 # Crear una lista
2 lista_1 = ['elemento0', 'elemento1', 'elemento2']
3 print(lista_1) # Imprime la lista
4
5 # Para acceder a los elementos de la lista:
6 print(lista_1[0])
7 # salida: elemento0
8
9 print(lista_1[2]) # o print(lista_1[-1])
10 # salida: elemento2
11
12 # Para agregar un elemento al final de la lista
13 lista_1.append('elemento3')
14 print(lista_1)
15 # salida: ['elemento0', 'elemento1', 'elemento2', '
    elemento3']
16
17 # Para obtener la longitud
18 longitud = len(lista_1)
19 print(longitud)
20 # salida: 4

```

Diccionarios: es una colección de par clave-valor. Su sintaxis es *{clave : valor}*:

```

1 diccionario_1={'nombre':'Marcelo',
2               'apellido':'Soto',
3               'ciudad':'Santiago',

```

```

4         'edad':38,
5         'profesion':'Presidente'}
6
7 # Imprimir un valor
8 print({diccionario_1['profesion']})
9 # salida: Presidente
10
11 # Agregar un nuevo par clave-valor
12 diccionario_1['estado civil'] = 'Soltero'

```

Tuplas: las tupla no se pueden modificar después de haber sido creada.

```

1 tupla_1=('elemento0', 'elemento1', 'elemento3')
2
3 # Para que devuelva 'elemento0'
4 tupla_1[0]

```

Set: no permite tener elementos repetidos. Guardará solo los datos que no se repiten.

```

1 set_1 = {'banana', 'manzana', 'pera'}

```

4.4. Limpieza, orden y transformación de datos

4.4.1. Bibliotecas

Las bibliotecas que más se usan en Python para la ciencia de datos son Pandas, NumPy, Scikit-learn y Matplotlib.

NumPy se especializa en el cálculo y análisis de datos, la cual permite manejar datos de forma rápida por medio de los NumPy Arrays o ndarray, una estructura de datos que puede ser de una dimensión (vector), dos dimensiones (matriz), tres (cubo) o más.

Por lo general, se importa con el alias np:

```

1 import numpy as np

```

Por otro lado, la biblioteca Pandas permite manipular y analizar estructuras de datos. Se basa en NumPy y proporciona los siguientes tipos de datos:

1. Series: Estructuras de una dimensión.
2. DataFrame: Estructura de dos dimensiones.
3. Panel: Estructura de tres dimensiones.

Se importa bajo el alias de pd:

```
1 import pandas as pd
```

Scikit-learn es una biblioteca de aprendizaje automático que proporciona acceso a algoritmos comunes.

Matplotlib se especializa en la generación de gráficos en dos dimensiones y personalización de estos. Se pueden crear diagramas de barras, mapas de calor, histogramas y más.

Por lo general, se importa con el alias plt:

```
1 import matplotlib.pyplot as plt
```

4.4.2. NumPy Arrays

Un NumPy Array se puede crear a partir de una lista usando el comando `np.array`:

```
1 # Arreglo 1d
2 a = np.array([1,2,3,4,5])
3
4 # Arreglo 2d
5 b = np.array([[3,4,5],
6               [4,3,2]])
```

O por medio de `no.arrange`, el cual crea un arreglo de una dimensión con valores del 0 a n:

```
1 # Sintaxis
2 # a = np.arange(inicio, fin-1, paso)
3 d = np.arange(3,8,1)
```

Propiedades:

```
1 # Dimension del arreglo
2 arreglo.ndim
3
4 # Forma del arreglo (filas, columnas)
5 arreglo.shape
6
7 # Cantidad de datos en el arreglo
8 arreglo.size
```

Para buscar valores en un arreglo, Python toma cada fila como un arreglo, donde el primero tendrá índice cero. Lo mismo ocurre con sus columnas, como se muestra a continuación:

```

1 #      0  1  2
2 array([[1, 0, 3], # indice 0
3        [2, 6, 9], # indice 1
4        [4, 5, 8]]) # indice 2

```

De esta forma, para seleccionar el número 6, se debe insertar arreglo[1,1].

Generación de números aleatorios:

```

1 # Crea un arreglo de numeros aleatorios entre 0 y 1
2 a_1 = np.random.default_rng(n)
3
4 # Genera n valores aleatorios que siguen una
   distribucion normal.
5 normal(media, desv_estandar,n)
6
7 # Genera n numeros aleatorios entre el 0 y fin-1.
8 integers(fin-1, size = n)
9
10 # Genera n numeros aleatorios entre el 0 y fin-1 sin
    repetirse.
11 choice(fin-1, size = n, replace = False)

```

4.4.3. Tipos de datos de Pandas

Series

Las Series se definen bajo la siguiente sintaxis:

```

1 Series(data=datos, index= indices, dtype='int') # tipo
   de dato

```

Por ejemplo:

```

1 # Serie creada con una lista de numeros
2 s = Series([1,2,3,4,5,6], dtype='int')
3
4 # Serie creada a partir de un diccionario
5 sdiccionario = pd.Series({'Bajo':1.40, 'Medio':1.60, '
   Alto:170'})

```

DataFrame

Un DataFrame es una tabla similar a las hojas de cálculo de Excel. Posee filas y columnas. Las columnas solo puede tener datos de un solo tipo y poseen un

encabezado. Cada fila se identifica por un índice único.

Es posible crear un DataFrame mediante listas de diccionarios, listas de listas, un array, etc. Se utiliza el siguiente código:

```
1 df = pd.DataFrame(data = datos, index = 'filas',
2   columns = 'columnas', dtype = 'int')
3 # index=False: retorna indices numericos
```

Ejemplo de un DataFrame:

```
1 indice = ['mes 1', 'mes 2', 'mes 3', 'mes 4', 'mes 5']
2 df1 = pd.DataFrame({'mes': ['marzo', 'abril', 'mayo',
3   'junio', 'julio'], 'ventas': np.random.randint
4   (1,100, 5)}, index = indice)
5 df1
```

Esto mostrará un resultado similar a:

	mes	ventas
mes 1	marzo	18
mes 2	abril	79
mes 3	mayo	18
mes 4	junio	21
mes 5	julio	59

Figura 4.1: DataFrame df1 en Colab

De la misma forma que en un NumPy Array, se pueden seleccionar datos específicos mediante un localizador de etiquetas con loc o índices con iloc:

```
1 # Busca el numero ubicado en la fila con indice 3 y
2   columna con indice 1
3 df1.iloc[3,1]
4 # 21
5 # Busca los datos entre marzo y mayo de df1
6 df1.loc[:,['mes 1': 'mes 3']] # puede ser el nombre de
7   una fila o columna
8 #
9 #      mes    ventas
10 # mes 1  marzo    18
11 # mes 2  abril    79
12 # mes 3  mayo     18
```

4.4.4. Importación datos

La importación de datos se hace mediante el comando `pd.read_csv` en el caso de un archivo `.csv`.

```
1 df = pd.read_csv('archivo.csv', index_col = 'nom_indice')
```

4.4.5. Resúmenes de datos

Por defecto, la función `head()` muestra las primeras cinco filas del `DataFrame`. De lo contrario, el comando `tail()` muestra las últimas cinco:

```
1 # Muestra los primeros 5 datos
2 df.head()
3
4 # Muestra los ultimos 5
5 df.tail()
```

Devuelve una tabla con la media, mediana, desviación estándar, máximo, mínimo y los percentiles del `DataFrame`:

```
1 df.describe()
```

Tipo de datos de las columnas del `DataFrame`:

```
1 df.info()
```

4.4.6. Manipulación y limpieza de datos

Transformación de datos

Cuando se trabaja con fechas, es posible que estos valores aparezcan de tipo `string`. Para transformarlos a un formato de fecha, se ocupa `pd.to_datetime()`:

```
1 df['fecha'] = pd.to_datetime(df['fecha'])
```

Luego, es posible acceder a los métodos de las fechas usando `.dt`:

```
1 # Accediendo a los annos de las fechas
2 df['fecha'].dt.year
3
4 # Accediendo a los meses de las fechas
5 df['fecha'].dt.month
6
7 # Accediendo a los nombres de los meses de las fechas
```

```
8 df['fecha'].dt.month_name()
9
10 # Accediendo a los días de las fechas
11 df['fecha'].dt.day
12
13 # Convirtiendo a formato de fecha y hora especificando
    el formato
14 df['fecha'] = pd.to_datetime(df['fecha'], format='%d-%
    m-%Y')
```

Búsqueda de valores

Además de `iloc` y `loc`, se puede buscar valores en un `DataFrame` con `isin()`, en el cual se le debe indicar entre paréntesis los valores a buscar:

```
1 # Devuelve el df donde encuentre los valores 23 y 45
    en la columna1
2 df[df.columna1.isin([23,45])]
```

Tratamiento de datos nulos

Elimina datos `np.nan`:

```
1 # Devuelve la cantidad de nulos en cada columna
2 df.isna().sum()
3
4 # Borra las columnas y filas que tengan datos
    faltantes
5 df.dropna()
6
7 # Borrando definitivamente los valores nulos
8 df.dropna(inplace=True)
9
10 # Creando un df nuevo sin nulos
11 df_nuevo = df.dropna()
12
13 # Reemplazando los valores nulos con 0
14 df.dropna(0)
15
16 # Reemplazando los valores nulos con 0
17 df.fillna(0)
```


Otra forma de buscar valores nulos en un DataFrame:

```
1 missing_values = df.apply(lambda x: sum(x.isnull()),  
    axis=0)
```

Desglosando el código:

- `df.apply()`: aplica una función a lo largo de un eje del DataFrame. Como `axis = 0`, lo aplica a lo largo de las columnas.
- `lambda x: sum(x.isnull())`: la función lambda toma una columna `x` y calcula la suma de sus valores nulos, es decir, devuelve la cantidad de valores nulos de dicha columna.

Tratamiento de datos duplicados

Quitar filas duplicadas:

```
1 # Devuelve los valores duplicados de columna1  
2 df[df.duplicated(subset=['columna1'])]  
3  
4 # Borrando duplicados de las columnas columna1 y  
   columna2 de df  
5 df_sin_duplicados = df.drop_duplicates(subset=['  
   columna1', 'columna2'])
```

Es posible tratar los datos faltantes rellenando con la media o la mediana:

```
1 # Rellenando con la media  
2 df['columna'].fillna(df['columna'].mean(), inplace=  
    True)
```

Tratamiento de datos atípicos

Una forma de limpiar datos atípicos de una muestra es mediante el método del rango intercuartílico (IQR), el cual consiste en eliminar los datos que estén fuera de $Q3 + 1.5 \cdot IQR$ o de $Q1 - 1.5 \cdot IQR$, donde IQR se define como el cuartil 3 (Q3) menos el cuartil 1 (Q1). Los valores fuera de este rango se consideran atípicos.

En Python, esto se puede hacer de la siguiente forma:

```
1 # Filtrando valores atipicos  
2 data = df['columna']  
3 Q1 = np.percentile(data, 25)  
4 Q3 = np.percentile(data, 75)  
5 IQR = Q3 - Q1
```

```
6 lower_bound = Q1 - 1.5 * IQR
7 upper_bound = Q3 + 1.5 * IQR
8 var_num=var_num.iloc[np.where((data>= lower_bound) * (
    data <= upper_bound))]
9
10 df.reset_index(drop=True, inplace=True)
```

Es posible visualizar los datos atípicos mediante un boxplot.

Filtros

Con Pandas se pueden filtrar datos por condiciones, columnas, filas.

Filtrando por columnas:

```
1 # Devuelve los valores de la columna ventas de df
2 df['ventas']
3
4 # Devuelve las columnas mes y ventas
5 df[['mes', 'ventas']]
```

Filtrando por filas:

```
1 # POR INDICE
2 # Devuelve la fila con indice 0
3 df.iloc[0]
4
5 # Devuelve desde la primera fila hasta la 4ta
6 df.iloc[0:5]
7
8 # Devuelve la fila con indice 0 y la 2
9 df.iloc[[0,2]]
10
11 # POR ETIQUETAS
12 # Seleccionando las etiquetas ventas y annos
13 df.loc[['ventas', 'anno']]
```

Filtrando por condiciones:

```
1 # Devuelve los datos del df donde las ventas son
    mayores a 1000
2 df[df['ventas']>1000]
3
4 # Devuelve los datos del df donde las ventas son
    mayores a 1000 y el mes es igual a abril
5 df[(df['ventas']>1000) & (df['mes'] == 'abril')]
```

```
6
7 # Devuelve los meses que contienen la subcadena a,
   como marzo, abril, mayo, agosto
8 df[df['mes'].str.contains('a')]
```

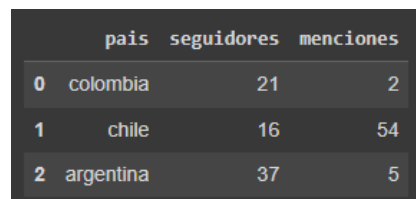
Orden

Para ordenar los datos de en DataFrame, se usa la función `sort_values`, seguida de la columna la cual se quiere ordenar y luego si se quiere ordenar de forma creciente o decreciente.

```
1 df.sort_values(by = 'columna_a_ordenar', ascending=
   True, inplace=False)
```

Agrupación de datos

En Pandas se utiliza `groupby` para agrupar datos de una columna para aplicarle funciones como la media, suma, etc.



	pais	seguidores	menciones
0	colombia	21	2
1	chile	16	54
2	argentina	37	5

Figura 4.2: DataFrame

El DataFrame anterior muestra cada país con su número de seguidores y menciones. Se que quiere agrupar el promedio de ambas columnas según su país, se debe aplicar el siguiente comando:

```
1 # Devuelve el promedio de los datos agrupados por sus
   paises
2 df.groupby('pais').mean()
```

Y devolverá la siguiente tabla:

	seguidores	menciones
pais		
argentina	37.0	5.0
chile	16.0	54.0
colombia	21.0	2.0

Figura 4.3: DataFrame agrupado según su país

Por otro lado, si se necesitan aplicar más funciones de agregación, se debe añadir el comando `agg` con un diccionario de las columnas y las funciones:

```
1 # Devuelve la suma de seguidores y el promedio de
  # menciones de df agrupados por pais
2 df.groupby('pais').agg({'seguidores': 'sum', '
  menciones': 'mean'})
```

	seguidores	menciones
pais		
argentina	37	5.0
chile	16	54.0
colombia	21	2.0

Figura 4.4: DataFrame agrupado según su país

4.4.7. Cruce de tablas

Con la librería Pandas, se pueden cruzar dos tablas mediante `merge()`, donde la primera tabla será la tabla derecha (`tabla1`) y la segunda la izquierda (`tabla2`):

```
1 # Cruce entre tabla1 y tabla2
2 df_merged = df_tabla1.merge(df_tabla2, left_on = '
  col1_t1', right_on = 'col1_t2', how='inner',
  validate = 'many_to_one')
```

Sus parámetros son:

- `left_on`: Columnas a cruzar por la izquierda.
- `right_on`: Columnas a cruzar por la derecha.

- on: En el caso de que ambas columnas tengan el mismo nombre, se usa on y se escribe el nombre de la columna.
- how: Es el tipo de cruce. Puede ser left, right, inner, outer o cross.
- validate: Comprueba si el cruce es del tipo indicado. Puede ser one_to_one, one_to_many, many_to_one, many_to_many.

4.5. Probabilidad y estadística

4.5.1. Estadística

Tipos de datos en estadística

En estadística, existen diferentes tipos de datos que se clasifican en:

1. Variables cualitativas ordinales: son variables que pueden ser ordenadas o poseen un tipo de jerarquía.
2. Variables cualitativas nominales: son variables que no pueden ser ordenadas. Se utilizan para clasificar elementos.
3. Variables cuantitativas discretas: son variables que representan un conteo donde los valores forman un conjunto finito.
4. Variables cuantitativas continuas: son las variables que representan una medición en escalas continuas.

Medidas de tendencia central

Las medidas de tendencia central son medidas estadísticas que se utilizan para resumir un conjunto de valores. Las medidas más comunes son:

- Media aritmética
- Moda
- Mediana

En Python, es posible calcular las medidas de tendencia central mediante las siguientes fórmulas:

```
1 # Media
2 media = df['columna'].mean()
3
4 # Moda
5 moda = df['columna'].mode()
6
7 # Mediana
8 mediana = df['columna'].median()
```

Indicadores de posición

Mientras que los indicadores de posición son indicadores estadísticos que se utilizan para resumir los valores en uno solo o dividirlos en intervalos del mismo tamaño. Dentro de ellos se encuentran:

- Percentiles
- Cuartiles
- Quintiles
- Deciles

En Python, se pueden calcular usando `quantile` y escribiendo el percentil. Por ejemplo, si se quiere obtener un percentil en específico:

```
1 # Percentiles
2 percentil = df['columna'].quantile(0.30)
```

O los quintiles de una columna:

```
1 # Quintiles
2 quintil = df['columna'].quantile([0, 0.2, 0.4, 0.6,
    0.8, 1])
```

Lo mismo con los cuartiles:

```
1 # Cuartiles
2 cuartil = df['columna'].quantile([0, 0.25, 0.50, 0.75,
    1])
```

Medidas de dispersión

A diferencia de las medidas de tendencia central que entregan una idea sobre la ubicación central de los datos, las medidas de dispersión se utilizan para describir el cambio o dispersión de un conjunto. Algunas son:

- Rango
- Rango intercuartil
- Desviación media
- Desviación estándar
- Varianza
- Coeficiente de variación

En Python, se pueden obtener de la siguiente forma:

```
1 # Rango
2 rango = df['columna'].max() - df['columna'].min()
3
4 # Rango intercuartil
5 rango_inter = df['columna'].quantile(0.75) - df['columna'].quantile(0.25)
6
7 # Desviacion estandar
8 des_estandar = df['columna'].std()
9
10 # Devuelve indicadores estadisticos
11 df.describe()
```

4.5.2. Probabilidad

La probabilidad es una medida que determina qué tan posible es que ocurra un evento. Para calcularla, se utilizan las siguientes definiciones básicas:

1. Experimento aleatorio: Es un experimento cuyo resultado está determinado por el azar.
2. Espacio muestral (Ω): Es el conjunto de todos los resultados posibles de un experimento.

3. Evento (A, B, \dots): Es un conjunto de resultados posibles, es decir, un subconjunto del espacio muestral.
4. Probabilidad ($P(A), P(B), \dots$): Es un valor que se le asigna a un evento entre 0 y 1, donde 0 es imposible y 1 es seguro de que ocurra. Se obtiene dividiendo el número de casos favorables por el número total de resultados posibles.
5. Complemento (A^c): Es la probabilidad de que un evento no ocurra. Si $P(A)$ es la probabilidad de que ocurra un evento A , entonces $P(A^c) = 1 - P(A)$ es la probabilidad de que no ocurra A .
6. Intersección ($A \cap B$): Son todos los elementos que son de un evento A y un evento B .
7. Unión ($A \cup B$): Son todos los elementos de A y todos los elementos de B .
8. Probabilidad clásica: Se basa en que todos los resultados del espacio muestral tienen la misma probabilidad.
9. Probabilidad frecuencial: Se calcula dividiendo el número de veces que ocurre un evento y el total de repeticiones del experimento.

Probabilidad condicional

La probabilidad condicional se refiere a que suceda un evento (A) sabiendo que sucede otro (B). Se expresa como:

$$P(A|B) \quad (4.1)$$

Y se lee la probabilidad de A dado B , la cual se puede calcular por medio del Teorema de Bayes:

$$P(A \cap B) = P(B) * P(A/B) \quad (4.2)$$

$$\Rightarrow P(A|B) = \frac{P(A \cap B)}{P(B)} \quad (4.3)$$

En Python, se pueden calcular probabilidades por medio de los siguientes comandos:

```

1  \\\ conteo
2  conteo = df['columna1'].value_counts()['valor1']
3
4  \\\ Conteo de cada registro
5  conteo_grupal = df['columna1'].value_counts()
```



```
6
7 \\ Conteo con condiciones
8 condicion1 = df['columna1'] == 'valor1'
9 condicion2 = df['columna2'] == 'valor2'
10
11 interseccion = (condicion1 & condicion2).sum()
12 union = (condicion1 | condicion2).sum()
```

Variable aleatoria

Se define variable aleatoria a una función que asigna un valor numérico a los resultados de un experimento aleatorio. Se clasifican en dos tipos:

1. Variables aleatorias continuas: pueden tomar cualquier valor dentro de un rango continuo de números reales.
2. Variables aleatorias discretas: son aquellas que pueden tomar un número finito o infinito numerable de valores posibles.

Distribución de probabilidad

La distribución de probabilidad de una variable aleatoria es una función que asigna a cada evento la probabilidad de que este ocurra. Describe cómo se distribuyen las probabilidades de los valores que puede tomar la variable.

Para una variable aleatoria continua, puede tomar cualquier valor dentro de un cierto rango. Se definen por medio de una función de probabilidad:

$$P(a \leq X \leq b) = \int_a^b f(x)dx \quad (4.4)$$

La cual representa el área debajo de la curva de $f(x)$ en el intervalo (a,b) .

Distribución binomial

La distribución binomial es una distribución de probabilidad que describe el número de veces que un evento aparece en una cantidad de experimentos. Los experimentos deben ser independientes entre sí, solo deben tener dos posibles resultados (éxito y fracaso) y la probabilidad de éxito debe ser constante para todos los experimentos.

Si se repite el experimento n veces, la probabilidad de tener k éxitos es de:

$$P(k) = \frac{n!}{k!(n-k)!} p^k (1-p)^{n-k} \quad (4.5)$$

donde:

- $P(k)$: es la probabilidad de que ocurra k éxitos.
- n : número total de experimentos.
- k : número de éxitos deseados.
- p : probabilidad de éxito en cada experimento.
- $1-p$: probabilidad de fracaso.

Por ejemplo, si se lanza una moneda al aire 10 veces y la probabilidad de tener sello es de 0,5, la probabilidad de tener 5 caras es:

$$P(5) = \frac{10!}{5!(10-5)!} 0,5^5 (1-0,5)^{10-5} \quad (4.6)$$

$$P(5) = \frac{10!}{5!(10-5)!} 0,5^5 (0,5)^5 \quad (4.7)$$

$$0,2373 \quad (4.8)$$

Distribución normal

La distribución normal es una curva que tiene forma de campana, donde la mayoría de los valores de la variable aleatoria se encuentran agrupados cerca de la media, y pocos valores en los extremos de ésta.

La distribución normal se puede utilizar para calcular la probabilidad de que una variable aleatoria continua tome un valor dentro de un cierto rango. Se caracteriza por su función de densidad de probabilidad, la cual describe la curva y la probabilidad de que un valor ocurra:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-0,5\left(\frac{x-\mu}{\sigma}\right)^2} \quad (4.9)$$

Una distribución normal posee simetría, donde la media, la mediana y la moda son iguales. Además, tiene una forma unimodal, los extremos son infinitos y posee parámetros de media y desviación estándar.

Ley de los grandes números

La ley de los grandes números establece que a medida que aumenta el número de repeticiones de un experimento (aumenta el tamaño de la muestra), la media tiende a acercarse a la media teórica de la distribución de probabilidad.

Ley débil: indica que la media muestral se acerca a la media poblacional cuando el tamaño de la muestra aumenta.

Ley fuerte: la muestra tiende a la media poblacional en la medida que el tamaño de la muestra tiende a infinito.

Teorema del límite central

El teorema del límite central establece que la distribución de la suma o media de una gran cantidad de variables aleatorias tiende a aproximarse a una distribución normal. Para esto, las variables aleatorias deben ser independientes entre sí, tener la misma distribución (media y varianza) y el tamaño de la muestra debe ser lo suficientemente grande.

4.6. Estadística inferencial

La estadística inferencial busca describir por medio de una muestra cómo puede ser la población. Posee parámetros, los cuales son indicadores que corresponden a la población, y estimadores, los que están asociados a la muestra:

Nombre	Parámetro	Estimador
Media	μ	\bar{x}
Varianza	σ^2	S^2
Desviación	σ	S
Proporción	p	\hat{p}

A través de los estadísticos de prueba, se puede relacionar un indicador estadístico con su respectivo estimador, lo cual permite hacer inferencia estadística de una muestra de datos.

Si una variable aleatoria X es tal que $X \sim N(\mu, \sigma)$ y x_1, x_2, \dots, x_n es una muestra aleatoria de X , entonces:

Si se conoce el valor de σ :

$$(\bar{x} - \mu) \frac{\sqrt{n}}{\sigma} \sim N(0, 1) \quad (4.10)$$

donde $\bar{x} - \mu$ es el error muestral definido como la diferencia entre los resultados de una muestra y los resultados que se habrían obtenido si se hubiera analizado toda la población.

De lo contrario, se puede usar el estimador S :

$$(\bar{x} - \mu) \frac{\sqrt{n}}{S} \sim t_{n-1} \quad (4.11)$$

Si X es una variable aleatoria y x_1, x_2, \dots, x_n es una muestra aleatoria de X ($n > 30$):

Si X tiene una distribución desconocida y se conoce S :

$$(\bar{x} - \mu) \frac{\sqrt{n}}{S} \approx N(0, 1) \quad (4.12)$$

Si X tiene distribución Bernouilli con probabilidad p :

$$\frac{\hat{p} - p}{\sqrt{p(1-p)}} \sim N(0, 1) \quad (4.13)$$

Cuando la muestra es menor que 30 o la desviación estándar de la población es desconocida, se utiliza la distribución t , o distribución de Student.

$$t = (\bar{X} - \mu) \frac{\sqrt{n}}{S} \quad (4.14)$$

donde t es el estadístico de prueba t que se utiliza para evaluar la diferencia entre la media muestral y la media poblacional.

4.6.1. Prueba de hipótesis

Una prueba de hipótesis se utiliza para evaluar si una afirmación sobre una población es compatible con la evidencia observada. Para realizar una prueba de hipótesis se siguen los siguientes pasos:

1. Formular la hipótesis nula (H_0), afirmación inicial que se somete a prueba, y alternativa (H_1), afirmación alternativa a la nula.
2. Establecer el nivel de significancia (α), el cual se refiere a la probabilidad de rechazar la hipótesis nula cuando es verdadera. Se usan valores como 0,05 o 0,01.
3. Elegir y calcular el estadístico de prueba.
4. Determinación de la región crítica, la cual se refiere al área de la distribución de probabilidad donde se encuentran los valores que llevarían a rechazar la hipótesis nula.
5. Tomar una decisión comparando el valor obtenido del estadístico de prueba y los valores de la región crítica.

4.7. Análisis y visualización

4.7.1. Gráficas con matplotlib y seaborn

Para crear visualizaciones en Python, por lo general se usan las librerías matplotlib y seaborn. Se deben seguir los siguientes pasos para poder hacer un gráfico:

1. Importar las bibliotecas.

2. Crear la figura.
3. Agregar los datos.
4. Agregar ejes.

Un ejemplo de un gráfico de líneas en matplotlib:

```
1 # Importando la libreria
2 import matplotlib.pyplot as plt
3
4 # Datos
5 x = [1, 2, 3, 4, 5]
6 y = [2, 4, 6, 12, 20]
7
8 # Creando el grafico de lineas
9 plt.plot(x, y, marker='o', label = 'Linea')
10
11 # Agregando los nombres de los ejes
12 plt.xlabel('Eje X') # Eje x
13 plt.ylabel('Eje Y') # Eje y
14 plt.title('Grafico de Ejemplo') # Titulo
15
16 # Para agregar una leyenda
17 plt.legend()
18
19 # Mostrando el grafico
20 plt.show()
```

Se verá de la siguiente forma:

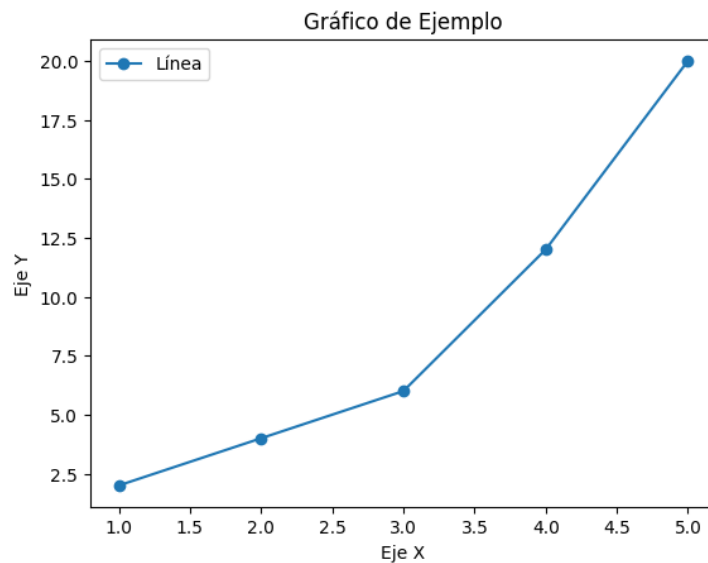


Figura 4.5: Gráfica de ejemplo

Parámetros:

```
1 # Agrega cuadrícula de fondo
2 plt.grid(True)
3
4 # marker: da opciones de la forma de los puntos
5 plt.plot(x, y, marker='o', label = 'Linea')
6 plt.plot(x, y, marker='-', label = 'Linea')
7
8 # markersize: grosor de los puntos
9 plt.plot(x, y, marker='o', markersize=6, label = '
   Linea')
10
11 # linestyle: configura el tipo de linea
12 plt.plot(x, y, marker='o', linestyle='-' label = '
   Linea')
13
14 # color: agrega color a la linea
15 plt.plot(x, y, color='b', label = 'Linea') # linea
   azul
16 plt.plot(x, y, color='r', label = 'Linea') # linea
   roja
```

marker también posee:

- `.`: punto.
- `,`: píxel.
- `^`: triángulo hacia arriba.
- `v`: triángulo hacia abajo.
- `s`: cuadrado.
- `+`: cruz.
- `*`: estrella.
- `x`: cruz diagonal.
- `D`: diamante.
- `p`: pentágono.

Es posible configurar el tamaño de la figura con:

```
1 # Configurando el tamaño a 8 x 6 pulgadas
2 plt.figure(figsize=(8,6))
```

Gráfico de líneas

En seaborn, un gráfico de líneas sería de la siguiente forma:

```
1 import seaborn as sns
2 import matplotlib.pyplot as plt
3
4 # Datos
5 x = [1, 2, 3, 4, 5]
6 y = [2, 4, 6, 12, 20]
7
8 # Creando el grafico de linea
9 sns.lineplot(x=x, y=y, label='Linea')
10
11 # Agregando etiquetas y titulo
12 plt.xlabel('Eje X')
13 plt.ylabel('Eje Y')
14 plt.title('Grafico de linea')
15
16 # Mostrando el grafico
17 plt.show()
```

Mostrará el siguiente gráfico:

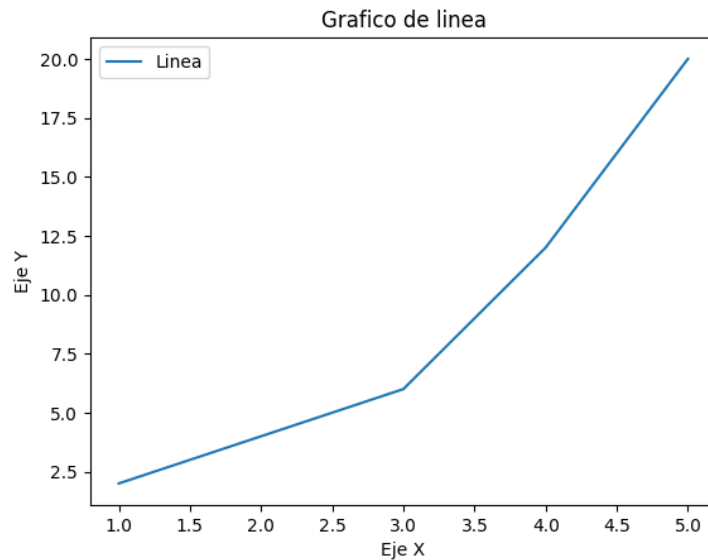


Figura 4.6: Gráfico de líneas con sns

Histograma

Con matplotlib, se puede construir un histograma de la siguiente forma:

```
1 import matplotlib.pyplot as plt
2
3 # Creando un histograma con 10 bins de color azul
  transparente
4 plt.hist(x, color='blue', alpha=.4, bins=10, label='x'
5 )
6 # Agregando titulo
7 plt.title('Histograma de la variable x')
8 # Leyenda
9 plt.legend()
```

Para construir un histograma en seaborn:

```
1 import seaborn as sns
2 import matplotlib.pyplot as plt
3
4 # Construyendo el histograma
5 sns.histplot(data=df, bins=10, color='skyblue')
6 # Agregando etiquetas y titulo
```



```
7 plt.title('Histograma con sns.histplot()')
8 plt.xlabel('Valores')
9 plt.ylabel('Frecuencia')
10 # Mostrando el grafico
11 plt.show()
```

Donde:

- data: datos.
- bins: cantidad de barras.

Con displot:

```
1 import seaborn as sns
2 import matplotlib.pyplot as plt
3
4 sns.displot(data, color='skyblue', bins=10)
5 # Agregando etiquetas y titulo
6 plt.title('Histograma con sns.displot()')
7 plt.xlabel('Valores')
8 plt.ylabel('Frecuencia')
9 # Mostrando el grafico
10 plt.show()
```

displot permite utilizar otros parámetros, como por ejemplo permite construir dos histogramas en uno:

```
1 import seaborn as sns
2 import matplotlib.pyplot as plt
3
4 # Creando un objeto en el cual grafique los datos de
   data correspondientes a mean_perimeter
   distinguiendo por el valor de diagnosis
5 ax = sns.displot(data=df, x='mean_perimeter', hue='
   diagnosis', fill=True)
6
7 # Agregando etiquetas y titulo
8 ax.set(xlabel='Media del Perimetro por Diagnostico',
   ylabel='Densidad', title='Perimetro v/s densidad')
9 # Mostrando el grafico
10 plt.show()
```

Distribución normal

Para crear una gráfica que represente una distribución normal, primero se debe crear un arreglo de valores equidistantes entre el valor mínimo y el máximo de los datos con `linspace`. Luego, se crea una función llamada `pdf` que calcula el valor de la función de distribución de probabilidad normal para finalmente construir el histograma.

```
1 from scipy.stats import norm
2 import seaborn as sns
3 import matplotlib.pyplot as plt
4
5 # Calculando el maximo y el minimo de los datos
6 minimo = df['columna'].min()
7 maximo = df['columna'].max()
8
9 # Arreglo de valores equidistantes
10 x = np.linspace(minimo, maximo)
11
12 # Crando la funcion pdf
13 # 'sigma' funciona como 'escala'
14 pdf = norm.pdf(x, loc=mu, scale=sigma)
15
16
17 # Construyendo un histograma
18 # 'density=True' construye el histograma haciendo que
19 # la suma de todas las areas de las barras sea igual
20 # a 1, para representar las probabilidades
21 plt.hist(maths, bins=100, density=True, alpha=0.5,
22         label='Muestra') #alpha=0.5 da un 50% de
23         transparencia
24
25 # Graficando la distribucion normal
26 plt.plot(x, pdf, color='red', label='PDF')
27 # Agregando etiquetas y titulo
28 plt.xlabel('x')
29 plt.ylabel('Densidad de probabilidad normalizada')
30 plt.title('Distribuciones comparadas')
31 # Leyenda
32 plt.legend()
33 # Mostrando el grafico
34 plt.show()
```

Boxplot

Anteriormente, se mencionó que se pueden visualizar de mejor manera los datos atípicos de una muestra con un boxplot.

```
1 import pandas as pd
2 import numpy as np
3 import seaborn as sns
4 import matplotlib.pyplot as plt
5
6 # df
7 data = {
8     'Grupo': ['A', 'A', 'B', 'B', 'B', 'C', 'C', 'C', 'C'],
9     'Valor': [5, 8, 12, 15, 20, 7, 10, 14, 18]
10 }
11
12 df = pd.DataFrame(data)
13
14 # Creaando un boxplot
15 sns.boxplot(x='Grupo', y='Valor', data=df)
16
17 # Agregando etiquetas y titulo
18 plt.xlabel('Grupo')
19 plt.ylabel('Valor')
20 plt.title('Boxplot por Grupo')
21
22 # Mostrando el grafico
23 plt.show()
```

Se verá de la siguiente forma:

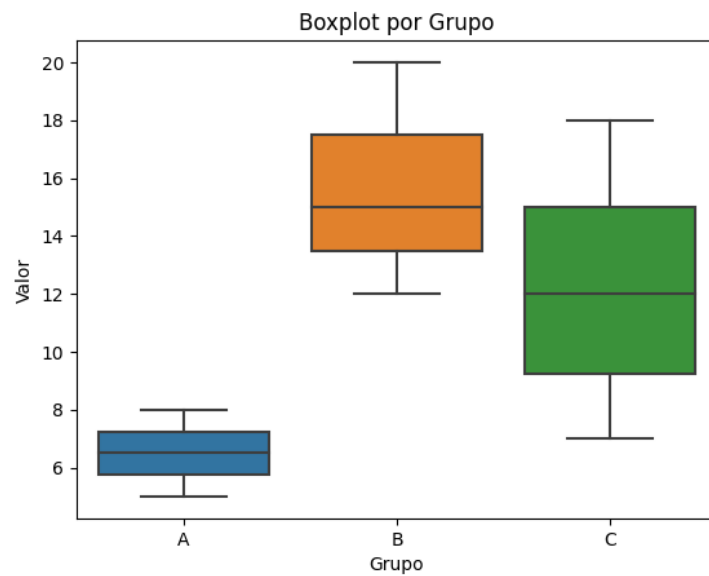


Figura 4.7: Boxplot

Donde un boxplot con valores atípicos se vería de la siguiente forma:

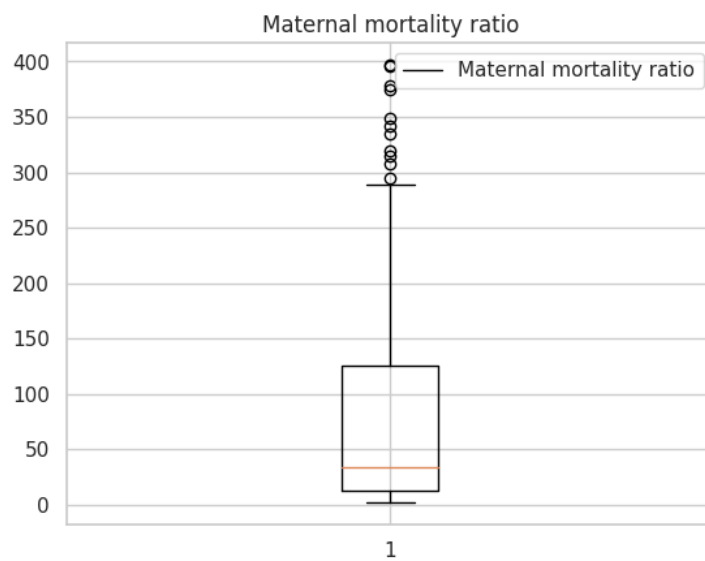


Figura 4.8: Boxplot con valores atípicos

Gráfico de barras

```
1 import seaborn as sns
```

```
2 import matplotlib.pyplot as plt
3
4 # Creando el grafico de barras
5 sns.barplot(x=grouped_data.index, y=grouped_data.
6             values) # Agregando etiquetas y titulo
7 plt.xlabel('x')
8 plt.ylabel('y')
9 plt.title('Grafica de barras')
10 # Etiquetas a las barras
11 custom_labels = ['Barra 1', 'Barra 2']
12 plt.xticks(range(len(custom_labels)), custom_labels)
13 # Mostrando el grafico
14 plt.show()
```

Scatterplot

```
1 import seaborn as sns
2 import matplotlib.pyplot as plt
3
4 sns.set(style='whitegrid')
5 # Creando el grafico
6 sns.scatterplot(x='x', y='y', hue='columna', data=df,
7                palette='Set1')
8 plt.xlabel('x')
9 plt.ylabel('y')
10 plt.title('Grafica scatterplot')
11 # Mostrando el grafico
12 plt.show()
```

El parámetro hue especifica que se utilizará una columna del DataFrame para colorear los puntos de la gráfica. palette='Set1' especifica la paleta de colores que se utilizará para colorearla.

Capítulo 5

R

5.1. Notas previas...

- 1.

5.2. Paquetes

Los paquetes en R incluyen funciones, documentación sobre estas mismas, muestras de conjuntos de datos y pruebas para verificar el código. R incluye un conjunto de paquetes denominados Base R.

Existe una colección de paquetes llamado Tidyverse que contiene los siguientes paquetes:

1. ggplot2: se usa para visualizar datos y crear visualizaciones.
2. tidyr: se usa para la limpieza de datos.
3. readr: se usa para importar datos.
4. dplyr: ofrece funciones que ayudan a la manipulación de datos.

Otros paquetes:

1. here
2. skimr: facilita el resumen de los datos.
3. janitor

Para ver los paquetes instalados:

`installed.packages()`

Para cargar paquetes:

```
library(paquete)
```

5.3. Limpieza

5.3.1. Resúmenes de datos

Funciones para obtener resúmenes de los marcos de datos:

```
#Muestra el tipo de datos  
str(df)  
#Devuelve el min,max,media,mediana,1er y 3r cuartil. skimr  
summary(df)  
# skimr  
skim_without_charts(df)  
# Skimr  
skim(df)  
# Devuelve el numero de filas y columnas.  
glimpse(df)  
# Muestra solo las primeras seis filas.  
head(df)
```

5.4. Orden

```
# Se usa para elegir qué variable se quiere ordenar.  
arrange()  
# Para ordenar de forma descendente.  
arrange(-)  
# Agrupa.  
group_by()  
# Filtra los datos.  
filter()  
# Excluye valores NA.  
drop_na()
```


5.5. Transformación de datos

```
# Divide datos en columnas separadas.
separate(df, columna-a-separar, into=c('nom-col-nueva',
  ↪ 'nom-col-nueva-2'), sep= 'tipo-de-separador')
# Permite fusionar columnas entre sí.
unite(df, 'nom-col', col-a-combinar, col-a-combinar-2,
  ↪ sep='separador')
# Se puede usar para añadir columnas con cálculos.
mutate()
```

5.6. Cálculos matemáticos

```
# Media aritmética.
mean()
# Valor máximo.
max()
# Desviación estándar.
sd()
# Correlación.
cor(x=var-x, y=var-y, method='pearson o kendall o spearman')
```

5.7. Visualización

Se utiliza la función ggplot para crear gráficas:

```
ggplot((data=df) + geom_función(mapping=aes(x=var-x,
  ↪ y=var-y, color=var1, shape=var2, size=var3,
  ↪ alpha=var4))), (color= color-para-todos-los-puntos) +
  ↪ labs() + annotate(text)
```

Conceptos de la función ggplot:

```
# Se inicia un diagrama y se le pueden agregar capas con +.
ggplot()
# Dataset.
data
# Usa puntos para crear diagramas (figura geométrica).
geom_función()
# Define cómo se aplican las variables a las propiedades
  ↪ visuales.
```

```

mapping # Va junto a la función aes()
# Especifica qué variables aplicar a los ejes x e y, como el
  ↪ color (color),
# color de relleno (fill), forma de puntos (shape), tipo de
  ↪ línea (linetype)
# y tamaño (size).
aes()
# Grado de transparencia del color.
alpha()
# Se pueden introducir etiquetas como
  ↪ title="", subtitle="", caption="".
labs
# Se utiliza para agregar texto dentro de la cuadrícula.
annotate(text)
# Más información en
  ↪ https://ggplot2.tidyverse.org/reference/annotate.html

```

Funciones geom:

```

# Crea diagramas de dispersion.
geom_point()
# Crea graficos de barra.
geom_bar()
# Gráfico de líneas.
geom_line()
# Suavizado. Crea una línea de tendencia.
geom_smooth()
# Crea un histograma.
geom_histogram()
# Crea un diagrama de dispersión y agrega una pequeña cantidad de ruido
# aleatorio para lidiar con la superposición de puntos.
geom_jitter()

```

Cuando se usa la función `geom_bar`, R cuenta automáticamente cuantas veces aparece cada valor `x` en los datos y muestra los recuentos en el eje `y`. Por esta razón, se puede poner en el código solo el eje `x`. Se puede usar `fill` para llenar de color las barras.

En el caso de `geom_histogram`, los argumentos de esta función son `bins=numero`, el cual hace referencia al número de intervalos y `binwidth=numero` es la amplitud de los intervalos.

5.7.1. Suavizado

El suavizado permite detectar una tendencia de datos aun cuando no se pueda notar con facilidad una tendencia en los puntos de datos graficados. En ggplot2, suma una línea de suavizado como otra capa en un diagrama. Ayuda a que los diagramas sean más legibles.

1. El suavizado LOESS es óptimo para suavizar diagramas con menos de 1000 puntos.

```
ggplot(data, aes(x=, y=))+ geom_point() +  
  ↪ geom_smooth(method="loess")
```

2. El suavizado GAM es útil para suavizar diagramas con un gran numero de puntos.

```
ggplot(data, aes(x=, y=))+ geom_point() +  
  ↪ geom_smooth(method="gam", formula = y ~ s(x))
```

5.7.2. Estética y facetas

Una faceta es una cara o sección de un objeto. Sirve para comparar datos. Permiten mostrar grupos más pequeños, o subconjuntos, de datos.

```
# Facetar el diagrama con una variable.  
facet_wrap(~variable)  
# Facetar el diagrama con dos variables.  
facet_grid(var1~var2)
```

5.7.3. Guardar gráficas

Para guardar una gráfica se puede exportar desde RStudio o usar la función:

```
ggsave(\nombre-gráfico.tipo-archivo", width=ancho,  
  ↪ height=largo) # del paquete ggplot2. Ejemplo:  
ggsave(\Pingüinos.png")
```


Capítulo 6

Markdown

Markdown es un lenguaje usado en documentos donde existen dos tipos de celdas: una de texto y otra de código. Las celdas de texto están formateadas con este lenguaje, lo que lo hace fácil y simple de escribir.

Tanto Colab como R permiten insertar ecuaciones usando la notación de LaTeX y personalizar la escritura con código HTML.

Se le puede dar formato a la escritura con los siguientes comandos:

1. *Cursiva*: `*texto*`
2. **Negrita**: `**texto**`
3. Tachado: `texto`
4. [Enlaces](#): `[Clic aquí](aquiVaElEnlace)`
5. Monoespaciado: `'texto'`
6. Imagen: `![Una imagen](linkDeLaImagen)`

Los encabezados en Markdown se inician con un numeral. Mientras más numerales tenga, más pequeño es el encabezado.

Capítulo 1: **Capítulo 1:**

Sección 1: **Sección 1:**

Subsección: **Subsección:**

Se pueden insertar bloques de código en las celdas de texto usando las comillas

“.

“python

import pandas as pd ““

De esta forma, el código en la celda tendrá un formato especial:

```
1 import pandas as pd
```

Se pueden crear listas ordenadas:

1. Propiedad 1
2. Propiedad 2
3. Propiedad 3

Y listas desordenadas con guiones (-) o asteriscos (*):

- Propiedad 1
- Propiedad 2
- Propiedad 3

Es posible insertar líneas horizontales con tres asteriscos (***) o tres guiones (—).

Capítulo 7

Git y GitHub

7.1. Git

Se puede hacer correr desde la cmd con el comando `git`.

Lista de comandos:

- `git config --global user.name "nom-usuario"`: se configura el nombre de usuario que va entre las comillas. Para configurar el email se debe poner `git config --global user.email` seguido del email entre comillas.
- `git config user.name`: para ver el nombre de usuario. Para ver el email es `git config user.email`

7.1.1. Repositorios

Para abrir Git en el proyecto, se debe hacer clic derecho Git Bash Here en la carpeta donde se ubica dicho proyecto. Una vez abierto, se puede crear un nuevo repositorio con `git init` en la rama Master o main.

Con el comando `git add nom-archivo.extension` se puede enviar archivos al repositorio. Si se quieren subir todos los archivos, se debe usar el comando `git add ..`

Una vez que se hayan realizado todos los cambios en los archivos, se le puede añadir un comentario con `git commit -m "Mensaje"` y se suben con `git push origin nom-rama`.

Si los archivos fueron modificados en GitHub, se pueden bajar al computador con `git pull`.

Lista de comandos:

1. `git status`: ver el estado del proyecto.

2. `git commit -m "mensaje"`: se utiliza para tomar una instantánea del proyecto y dejar un mensaje con los cambios realizados en este.
3. `git log`: para ver todos los mensajes y cambios que han realizado los autores.
4. `git log --stat`: sirve para visualizar los commits y muestra un pequeño resumen de lo modificado, como la cantidad de líneas e inserciones.
5. `git log --oneline`: muestra un código que identifica al commit.
6. `git checkout codigo-commit`: actualiza el archivo para que coincida con el commit señalado en el código-commit. Una vez que se vuelve a un commit anterior, los otros que están en medio se borran. Por esta razón se recomienda trabajar con ramas.
7. `git rm --cached nom-archivo.extension`: deja de rastrear un archivo sin eliminarlo del directorio de trabajo. Si se quiere hacer con todos los archivos: `git rm --cached ..`
8. `git clone direccion-del-repositorio`: clonar un repositorio en el computador.
9. `touch nom_archivo.extension`: crea un archivo.
10. `clear`: limpiar la pantalla.

7.1.2. .gitignore

El archivo `.gitignore` se crea con fines de que Git ignore ciertos archivos. Para crearlo, se puede hacer mediante la consola de Git con `touch .gitignore`. Luego de crearlo, se puede abrir con cualquier aplicación y escribir los archivos que se desean ignorar. Se deben escribir los nombres de los proyectos, como `mi_proyecto.txt`, o se pueden ignorar todas las extensiones, como `*.txt`.

7.1.3. Ramas

Una rama en Git es una versión del código de un proyecto, las cuales ayudan a mantener el orden en el control de versiones. El control de versiones es la práctica de gestionar los cambios que se realizan sobre un código. Cuando se habla del estado en el que se encuentra, se refiere a la versión, revisión o edición de este. Cada repositorio en Git comienza con una rama principal predeterminada llamada `master`.

Git almacena una serie de instantáneas al ejecutar el comando commit junto con los metadatos de quién lo haya modificado.

Comandos:

- `git branch`: devuelve la cantidad de ramas que tiene un proyecto.
- `git checkout rama`: cambia de rama.
- `git checkout -b nombre-rama`: crear una rama.
- `git merge nombre-rama`: unir ramas. El código de nombre-rama se une al de la rama en la que se está trabajando.
- `git branch -d nombre-rama`: eliminar una rama solo si la rama se ha fusionado. De lo contrario, usar `-D` que elimina la rama independientemente de su estado de fusión.
- `git branch -d nombre-remoto/nombre-rama`: eliminar una rama remota de VSCode. Generalmente, `nombre-remoto` es `origin`.

Capítulo 8

Tableau Public

Tableau Public es una plataforma para crear y compartir visualizaciones de datos en línea.

Una visualización de datos es una representación gráfica de los mismos para comunicar información de manera efectiva. Se usa principalmente para identificar patrones que pueden ser difíciles de detectar en una tabla de datos, explorar datos, comunicar información, ayudar a tomar decisiones informadas, hacer seguimiento a procesos, identificar problemas de procesos y más.

8.1. Creación de visualizaciones

8.1.1. Importación de datos y conexiones

Para importar datos, se pueden cargar archivos desde el equipo o conectarse con Google Drive. Si se desean agregar más de un archivo, se debe seleccionar el signo + al lado de Conexiones.

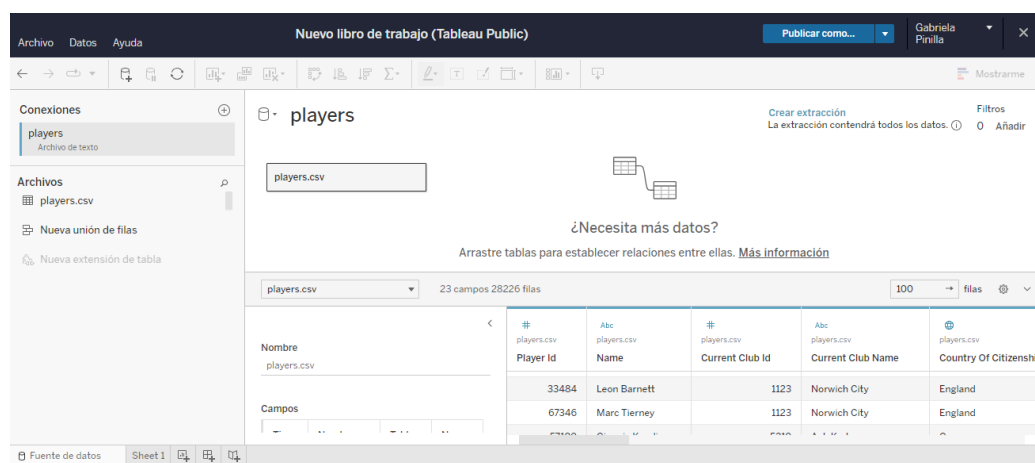


Figura 8.1: Fuente de datos

En la esquina inferior derecha se muestran los datos en una tabla. Hacia la izquierda en Conexiones, se muestran las fuentes de los datos, los cuales se pueden conectar unos con otros arrastrándolos hacia el panel que dice ¿Necesita más datos?, quedando de la siguiente forma:

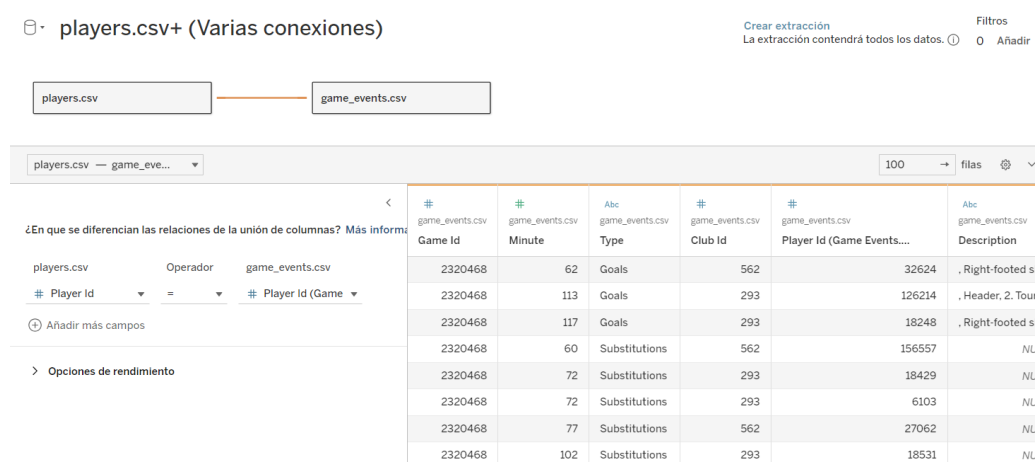


Figura 8.2: Campos en común

Tableau detecta si hay un campo en común entre ambos archivos csv la cual muestra bajo la conexión de las fuentes.

8.1.2. Campos calculados

Desde la misma hoja de Fuentes de datos se pueden realizar cálculos en las tablas. Para hacerlo, se debe seleccionar la tabla, luego la flecha de la columna en

la que se desea crear un campo calculado, > Crear > Campo calculado...

Se le puede asignar un nombre al campo calculado y escribir la fórmula para calcular los valores del campo seleccionado con los nombres de los campos entre paréntesis de corchetes. Ejemplo: [Campo 1]*5.

En Tableau existen funciones básicas que se pueden utilizar para las fórmulas de los campos calculados:

1. CONTAINS: Retorna verdadero si una cadena contiene una subcadena. Ejemplo: CONTAINS("Hola a todos", "todos"), devolverá verdadero.
2. LTRIM: Elimina los espacios en blanco del lado izquierdo de una cadena. Ejemplo: LTRIM("Hola"), devuelve "Hola".
3. TRIM: Elimina los espacios en blanco de ambos lados de una cadena. Ejemplo: TRIM("Hola "), retorna "Hola"
4. SPLIT(): Divide un campo según un separador especificado y un token. Ejemplo: SPLIT([Nombre completo], ' ', 2), en el caso de que el nombre sea Juan Soto, devolverá Soto. Si el token es 1, devolverá Juan.
5. LEN: Devuelve el largo de una cadena de texto. Ejemplo: LEN("Hola a todos"), devuelve 12.
6. LEFT: Devuelve los primeros caracteres de una cadena según una cantidad especificada. Ejemplo: LEFT("Hola a todos", 4), devuelve "Hola".
7. RIGHT: Devuelve los caracteres iniciando desde la derecha hacia la izquierda. Ejemplo: RIGHT("Hola a todos", 5), devuelve "todos".
8. LOWER: Convierte el texto en minúsculas. Ejemplo: LOWER("HOLA"), devuelve hola.
9. UPPER: Convierte el texto en mayúsculas. Ejemplo: UPPER("hola"), devuelve "HOLA".
10. MAX: Retorna el valor máximo de una expresión. Ejemplo: MAX([Ventas]), retornará el valor máximo de la columna Ventas.
11. REPLACE: Reemplaza una subcadena de texto por otra. Ejemplo: REPLACE("Hola a todos", "todos", "todas las mujeres presentes aquí"), devuelve "Hola a todas las mujeres presentes aquí".
12. COUNT: Muestra la cantidad de filas de un campo. Ejemplo: COUNT([Libros]), devolverá la cantidad de libros.

13. **CEILING:** Devuelve el número entero más pequeño mayor o igual que un número especificado. Ejemplo: `CEILING(5,13)`, devuelve 5.
14. **FLOOR:** Devuelve el número entero más grande menor o igual que un número especificado. Ejemplo: `FLOOR(5,16)`, devuelve 3.
15. **ROUND:** Redondea un número con decimales especificados. Ejemplo: `ROUND("5,1634546", 2)`, devuelve 5,16.
16. **LOG:** Devuelve el logaritmo en base e de un número.
17. **DATE:** Devuelve la fecha. Ejemplo: `DATE(01/05/2023 00:00:00)`, devuelve 01/05/2023.
18. **DATEDIFF:** Devuelve la diferencia entre dos fechas. Ejemplo: `DATE-DIFF(01/09/2023, 10/09/2023, DAY)`, devolverá la cantidad de días entre ambas fechas.
19. **DATENAME:** Devuelve el nombre de un mes, día de la semana o del año.
20. **DATEPART:** Devuelve una parte específica de una fecha. Ejemplo: `DATEPART(01/09/2023, YEAR)`, devolverá 2023.

8.1.3. Conceptos y elementos de Tableau

Al momento de crear un dashboard, se debe tener en consideración ciertos términos y elementos de la plataforma.

Una hoja de trabajo es un espacio donde se puede construir la visualización de datos. Sus elementos principales son:

- **Filas y columnas:** aquí es donde el usuario puede arrastrar los campos de datos para definir las dimensiones y medidas de la visualización.
- **Dimensiones y medidas:** en el panel de datos se pueden encontrar los datos divididos por una línea donde los de arriba representan las dimensiones y los de abajo las medidas.
 - Las dimensiones son los datos categóricos. Su fin es clasificar la información. Los tipos de dimensiones son: fechas, cadenas de texto, booleanos (verdadero o falso) y rol geográfico. Ejemplo: género, estado civil.
 - Las medidas son datos de origen cuantitativos que asignan valores a las dimensiones. Los tipos de medidas son: números y rol geográfico (latitud y longitud). Ejemplo: cantidad de habitantes.

En este mismo panel, se encuentran datos de color azules, los cuales corresponden a datos discretos. Se tratan como finitos. Por el contrario, los verdes son datos continuos. Se tratan como un intervalo infinito.

- **Marcas:** en la tarjeta de marcas se configura la forma en que las dimensiones y medidas serán representadas. Sus elementos son: Tipo de marca: el cual indica la forma en la que se van a graficar los datos; Color: permite cambiar los colores de la visualización; Tamaño: varía el tamaño de la visualización; Texto: permite definir el texto de la etiqueta a mostrar en la gráfica; Detalle: permite agregar información a las marcas; Descripción emergente: permite que al pasar el mouse por una marca se muestre una descripción emergente con información.

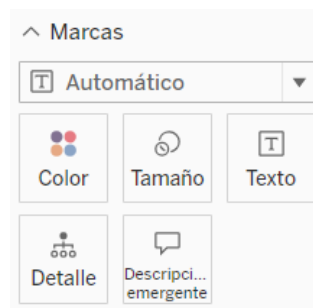


Figura 8.3: Tarjeta de marcas

- **Filtros:** aplica filtros al arrastrar los campos a este estante.
- **Páginas:** se pueden arrastrar campos para crear páginas en la visualización.

8.1.4. Filtros

1. **Filtros de extracción:** se usan para limitar el conjunto de datos desde la Fuente de datos. Al aplicarlo, se abrirá una ventana para añadir los filtros que se deseen.

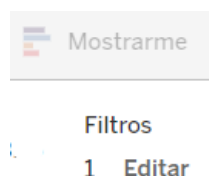


Figura 8.4: Filtro de extracción

2. Filtros de contexto: se usan para definir un contexto específico en los cálculos. En lugar de aplicar un filtro para el conjunto de datos, crea un subconjunto de datos para hacer cálculos. Para crear un filtro de contexto, se debe arrastrar un campo al estante de filtros en la hoja de trabajo, luego seleccionar la flecha de la cápsula del campo y seleccionar Añadir a contexto.

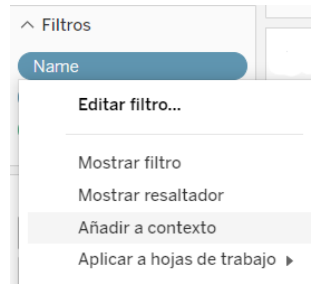


Figura 8.5: Filtro de contexto

3. Filtro de dimensión: se usan para filtrar datos de dimensiones discretas.

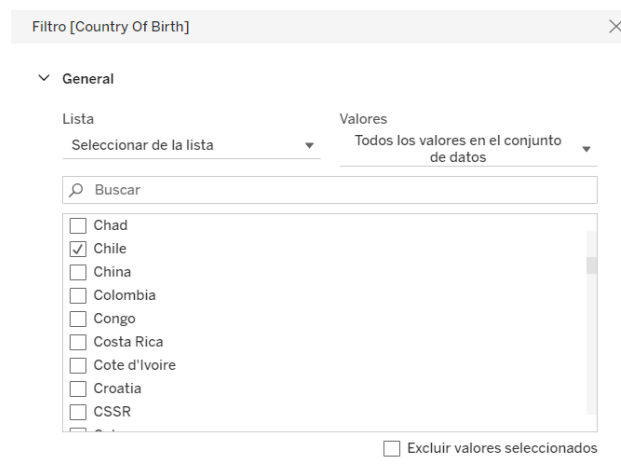


Figura 8.6: Filtro de dimensión

4. Filtro de medida: se usan para filtrar datos en función de una medida continua.

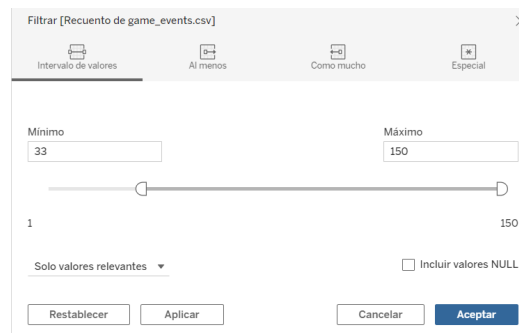


Figura 8.7: Filtro de medida

5. Filtro en gráfico: se pueden filtrar datos en un gráfico.

8.2. Creación de visualizaciones básicas

La creación de visualizaciones en Tableau se hace arrastrando las tablas de datos hacia los estantes de filas, columnas, marcas y filtros.

En el siguiente ejemplo se utilizó la base de datos del siguiente enlace de Kaggle [Uhttps://www.kaggle.com/datasets/ishikajohari/shazam-global-top-200-per-week?resource=download](https://www.kaggle.com/datasets/ishikajohari/shazam-global-top-200-per-week?resource=download), el cual muestra datos del top 200 por semana de Shazam, una aplicación para identificar música y otros desde los dispositivos móviles.

Con la base de datos de la primera semana (Week1 -22-Jul-to-28-Jul-2023.csv), se construyó la siguiente tabla:

The screenshot shows the Tableau Public interface. On the left, the 'Datos' (Data) pane lists several data sources: 'Week1 - 22-Jul-to-28-Ju...', 'Artist', 'Title', 'Nombres de medidas', 'Rank', 'Week1 - 22-Jul-to-28-Ju...', and 'Valores de medidas'. The 'Marcas' (Marks) shelf contains 'SUMA(Rank)'. The 'Columnas' (Columns) shelf contains 'Artist' and 'Title'. The 'Filas' (Rows) shelf is empty. The main view shows a table with the following data:

Artist	Title	Rank
A.V.G	Я плачу	56
Aaron Smith	Dancin (feat. Luvli) [Krono..	118
Adele	Set Fire to the Rain	45
Aerosmith	Dream On	98
Akon	Lonely	194
	Right Now (Na Na Na)	84
Alec Benjamin	Let Me Down Slowly	117
Aqua	Barbie Girl	120
Asake	Lonely At The Top	33
AYLIVA	Aber sie	170
Ayra Starr	Rush	42
Bad Bunny	WHERE SHE GOES	60
Bad Gyal, Tokíscha & YOUN...	Chulo pt. 2 (Pt. 2)	167
Bakar	Hell N Back	198
Beastie Boys	No Sleep Till Brooklyn	105
Becky G.	Arranca (feat. Omega)	173
Ben E. King	Stand By Me	146
Benny Benassi & The Biz	Satisfaction (Radio Edit)	100
Bibi Babydoll & Dj Brunin ..	Automotivo Bibi Fogosa	5
Billie Eilish	headline (edit)	104

Figura 8.8: Tabla de datos creada en Tableau

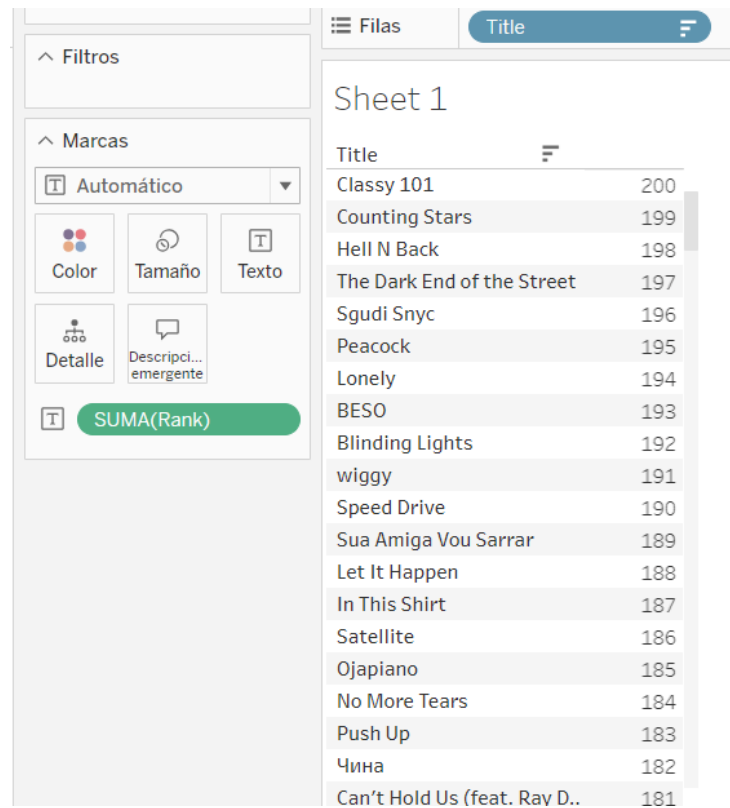
En el estante de filas se añadieron Artist (dimensión) y Title (dimensión), mientras que Rank (medida) se añadió a Texto en Marcas, mostrando en la tabla el rankin de la canción y al artista a quien le pertenece. Por defecto, Tableau añade una función de agregación, que en el caso del ejemplo es SUMA, sin embargo, como se trata de un rankin, solo corresponde a un número y no a una suma.

En la misma tabla, se pueden ordenar los datos por el rankin, como de mayor a menor, sin embargo, al tener dos dimensiones en las filas no mostrará cuál canción es menos escuchada en el rankin, sino que mostrará un conjunto de canciones con su rankin perteneciente al mismo artista.

Artist	Title	
Travis Scott	FE!N (feat. Playboi Carti)	131
	FRANCHISE (feat. Young T..	133
	I KNOW ?	88
	MELTDOWN (feat. Drake)	39
	MY EYES	83
	TELEKINESIS (feat. SZA & ..	29
OneRepublic	Counting Stars	199
	I Ain't Worried	172
	RUNAWAY	123
Lana Del Rey	Radio	138
	Say Yes To Heaven	180
	Summertime Sadness	153
The Weeknd	After Hours	99
	Blinding Lights	192
	Save Your Tears	127
Taylor Swift	cardigan	169
	Cruel Summer	64
	seven	161

Figura 8.9: Tabla de datos creada en Tableau

Para descubrir cuál es la canción menos escuchada, se debe dejar solo una dimensión en filas, como Title, y luego ordenarlas. Si se quiere saber cuál es el artista menos escuchado, se debe hacer lo mismo con la dimensión Artist.

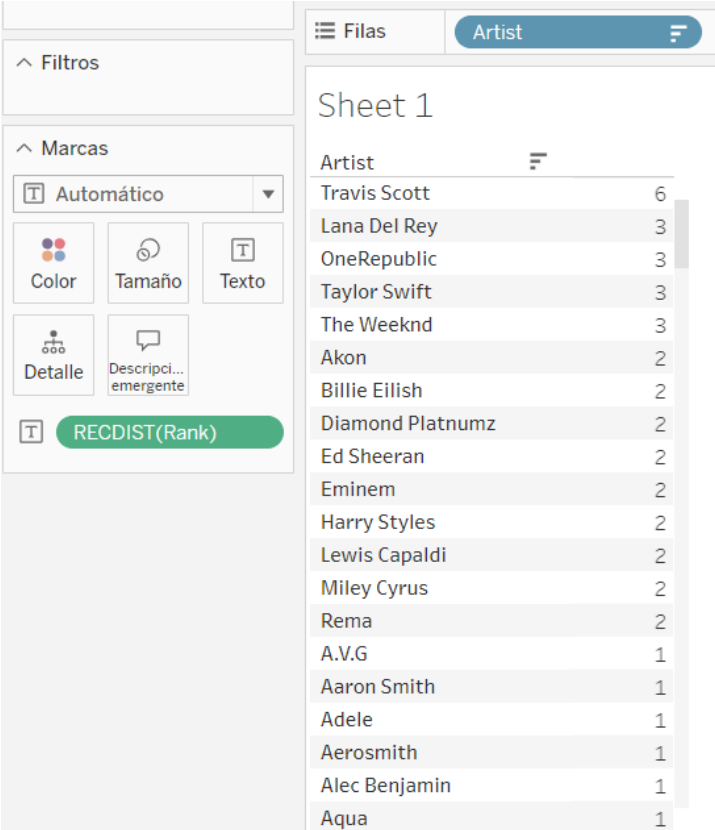


Sheet 1

Title	
Classy 101	200
Counting Stars	199
Hell N Back	198
The Dark End of the Street	197
Sgudi Snyc	196
Peacock	195
Lonely	194
BESO	193
Blinding Lights	192
wiggy	191
Speed Drive	190
Sua Amiga Vou Sarrar	189
Let It Happen	188
In This Shirt	187
Satellite	186
Ojapiano	185
No More Tears	184
Push Up	183
Чина	182
Can't Hold Us (feat. Ray D..	181

Figura 8.10: Tabla de datos creada en Tableau

También, se puede modificar la medida Rank para que cuente la cantidad de canciones con la que los artistas aparecen en el rankin.



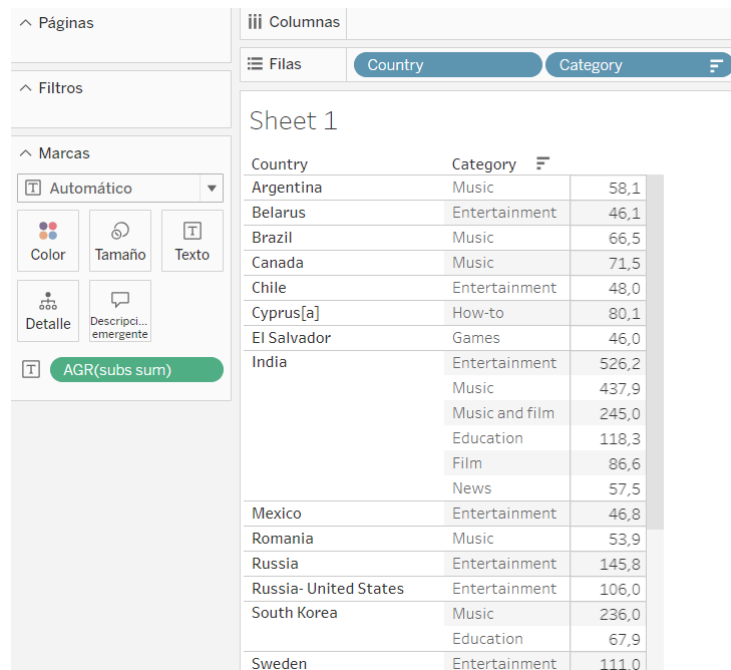
The screenshot shows the Tableau interface. On the left, the 'Marcas' (Marks) shelf contains 'Automático' (Automatic) and 'RECDIST(Rank)'. The 'Filtros' (Filters) shelf is empty. The 'Filas' (Rows) shelf contains 'Artist'. The main view displays a table titled 'Sheet 1' with the following data:

Artist	RECDIST(Rank)
Travis Scott	6
Lana Del Rey	3
OneRepublic	3
Taylor Swift	3
The Weeknd	3
Akon	2
Billie Eilish	2
Diamond Platnumz	2
Ed Sheeran	2
Eminem	2
Harry Styles	2
Lewis Capaldi	2
Miley Cyrus	2
Rema	2
A.V.G	1
Aaron Smith	1
Adele	1
Aerosmith	1
Alec Benjamin	1
Aqua	1

Figura 8.11: Tabla de datos creada en Tableau

Para el siguiente ejemplo, se tomó la base de datos mensual de las 50 principales cuentas de redes sociales de <https://www.kaggle.com/datasets/amyrmahdy/monthly-top-50-social-media-accounts-dataset>.

Con la base de datos del top 50 de Youtube (youtube_top_50_2023-07-03), se construyó la siguiente visualización:



Country	Category	AGR(subs sum)
Argentina	Music	58,1
Belarus	Entertainment	46,1
Brazil	Music	66,5
Canada	Music	71,5
Chile	Entertainment	48,0
Cyprus[a]	How-to	80,1
El Salvador	Games	46,0
India	Entertainment	526,2
	Music	437,9
	Music and film	245,0
	Education	118,3
	Film	86,6
	News	57,5
Mexico	Entertainment	46,8
Romania	Music	53,9
Russia	Entertainment	145,8
Russia- United States	Entertainment	106,0
South Korea	Music	236,0
	Education	67,9
Sweden	Entertainment	111,0

Figura 8.12: Tabla de datos creada en Tableau

Esta tabla muestra la categoría (Category) de cada país (Country) ordenado de mayor a menor por la suma de los subscriptores (subs sum) de cada categoría. Esta suma se realizó en el campo calculado de la Fuente de datos con el código `SUM([Subscribers (millions)])`.

Sin embargo, estos datos también se pueden representar de forma geoespacial por medio de un mapa.

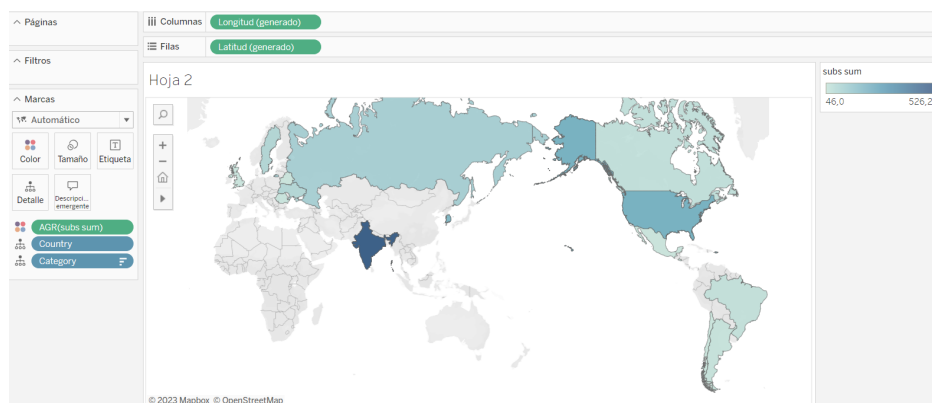


Figura 8.13:

Ordenando los subscriptores de forma descendente, se obtiene un mapa que

muestra la categoría con más subscriptores en los países, donde el color azul indica un número más alto que el color celeste claro.

Para hacer esto, se agregó la tabla de subscriptores a la marca de Color, de esta forma Tableau puede hacer un mapa de calor indicando qué países tienen una mayor cantidad de subscriptores. La categoría y el país se arrastraron hacia la marca de Detalles, por lo que al pasar el mouse por encima de alguno de estos países mostrará el nombre del país, la categoría y la cantidad de subscriptores. La latitud y longitud en filas y columnas se generó a partir de la tabla Country.

8.3. Diseño y personalización de paneles

Al graficar, se debe tener en consideración el tipo de datos que presenta la base de datos dado que no todos los gráficos sirven para representar cualquier tipo de datos.

Para elegir un gráfico que va a representar un conjunto de datos, es importante saber qué es lo que se quiere mostrar con la visualización, conocer los tipos de gráficos y saber cuál es la audiencia a quienes se les va a presentar.

A continuación, se presentan algunos tipos de gráficos:

1. Gráfico de columnas: Compara categorías usando barras verticales.
2. Gráfico de barras: Compara categorías usando barras horizontales.
3. Gráfico circular: Proporciona las categorías en relación al total.
4. Gráfico de líneas: Muestra las tendencias a lo largo del tiempo usando líneas.
5. Gráfico de área: Muestra las tendencias a lo largo del tiempo usando un área sombreada bajo la línea.
6. Gráfico de dispersión: Relaciona dos conjuntos de datos usando puntos.
7. Gráfico de burbujas: Relaciona dos conjuntos de datos usando burbujas de diferentes tamaños para mostrar la magnitud de un tercer conjunto de datos.
8. Histograma: Distribuye los datos en un rango continuo.
9. Gráfico de radar: Expone los datos en un formato circular con categorías que se extienden radialmente.

8.3.1. Funciones de agregación

Las funciones de agregación son una funciones que operan sobre un conjunto de datos que devuelven un único valor. Se pueden obtener el promedio de un conjunto, máximo, mínimo, recuento o suma.

Las funciones de agregación más comunes en Tableau son:

- Promedio
- Suma
- Promedio o media
- Mediana
- Recuento: devuelve la cantidad de filas. Ejemplo: $\text{Rec}(a,a,b,b)=4$.
- Recuento (distintos): devuelve la cantidad de valores distintos de una columna. Ejemplo: $\text{RecDis}(a,a,b,b)=2$.
- Máximo y mínimo
- Desviación estándar y desviación estándar poblacional
- Varianza y varianza poblacional

Para agregar una función de agregación, se debe seleccionar Medida > Suma, Promedio, Mediana...

Por defecto, Tableau agrega la suma para medidas numéricas y el recuento para las no numéricas.

8.3.2. Funciones LOD

Las funciones LOD (Level of Detail) en Tableau permite realizar cálculos en un nivel de detalle específico en los datos. Son útiles cuando se requiere realizar cálculos que no se pueden lograr con las funciones de agregación.

Tableau tiene tres tipos de funciones LOD:

1. **FIXED**: permite definir un nivel de detalle específico para un cálculo. Ejemplo: Para calcular la suma de ventas por categoría: `FIXED [Categoría] : SUM([Ventas])`.
2. **INCLUDE**: permite incluir un nivel de detalle específico a un cálculo sin afectar el nivel de detalle general de la visualización. Ejemplo: Para calcular la suma de ventas por categorías e incluir la ciudad: `INCLUDE [Categoría],[Ciudad] : SUM([Ventas])`.

3. EXCLUDE: permite excluir un nivel de detalle específico a un cálculo sin afectar el nivel de detalle general de la visualización.

8.4. Dashboards

Un dashboard consiste en una pantalla que presenta una colección de visualizaciones que resumen información importante.

Uno de los propósitos de un dashboard es monitorear el rendimiento de una empresa por medio de KPI, presentando la información de forma clara para que el público pueda entenderla.

Un KPI (Key Performance Indicator o Indicador Clave de Rendimiento) es una medida que evalúa el rendimiento de una empresa o proceso con respecto al objetivo empresarial. Los KPI siguen la metodología S.M.A.R.T., un acrónimo que explica las características básicas de un objetivo SMART. Estos deben ser específicos (Specific), medibles (Measurable), alcanzables (Attainable), realistas (Realistic) y con un tiempo límite (Time).

Ejemplos de métricas que se pueden usar para los KPI:

- ROI
- Número de clientes nuevos
- Tasa de satisfacción
- Tasa de rotación del personal
- Porcentaje de clics en una página web
- Cantidad de carritos de compras abandonados
- IPC

8.4.1. Creación de un dashboard

Para crear un dashboard, se debe seleccionar el icono de Nuevo dashboard. Luego, arrastrar las vistas desde la lista de Hojas hacia el dashboard.

Capítulo 9

Glosario de términos de programación

La siguiente lista muestra las traducciones al español de los términos de programación en inglés más usados.

1. Array: Arreglo, listas.
2. Dataset: Conjunto de datos.
3. Debugging: Depuración.
4. Default: Predeterminado.
5. Delete: Borrar.
6. Drop: Borrar.
7. Loop: Bucle.
8. Return: Retornar.
9. String: Cadena.
10. Database: Base de datos.
11. Null: Nulo.
12. Fork: Bifurcación, clonación de un programa.

Bibliografía

- [1] <https://www.javatpoint.com/postgresql-tutorial>
- [2] <https://git-scm.com/doc>
- [3] <https://git-scm.com/book/es/v2>
- [4] https://help.tableau.com/current/pro/desktop/es-es/dashboards_create.htm
- [5] <https://docs.python.org/es/3/index.html>