Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2115 - Programación como herramienta para la ingeniería

Manejo y análisis de datos tabulares

Profesor: Hans Löbel

A través de este análisis, buscamos responder preguntas relevantes, y/o descubrir aspectos desconocidos, en base a la evidencia dada por los datos

- Los datos tabulares son los más tradicionales, abundantes e intuitivos, lo que los convierte en un excelente punto de inicio.
- Una vez vista las bases en este capítulo, pasaremos a otros tipos de datos en el resto del curso.
- Desde un punto de vista práctico, realizaremos las siguientes tareas para realizar el análisis:
  - Carga y combinación de datos de distintas fuentes
  - Exploración y descripción de distintas dimensiones de los datos
  - Limpieza, corrección y transformación
  - Análisis estadístico

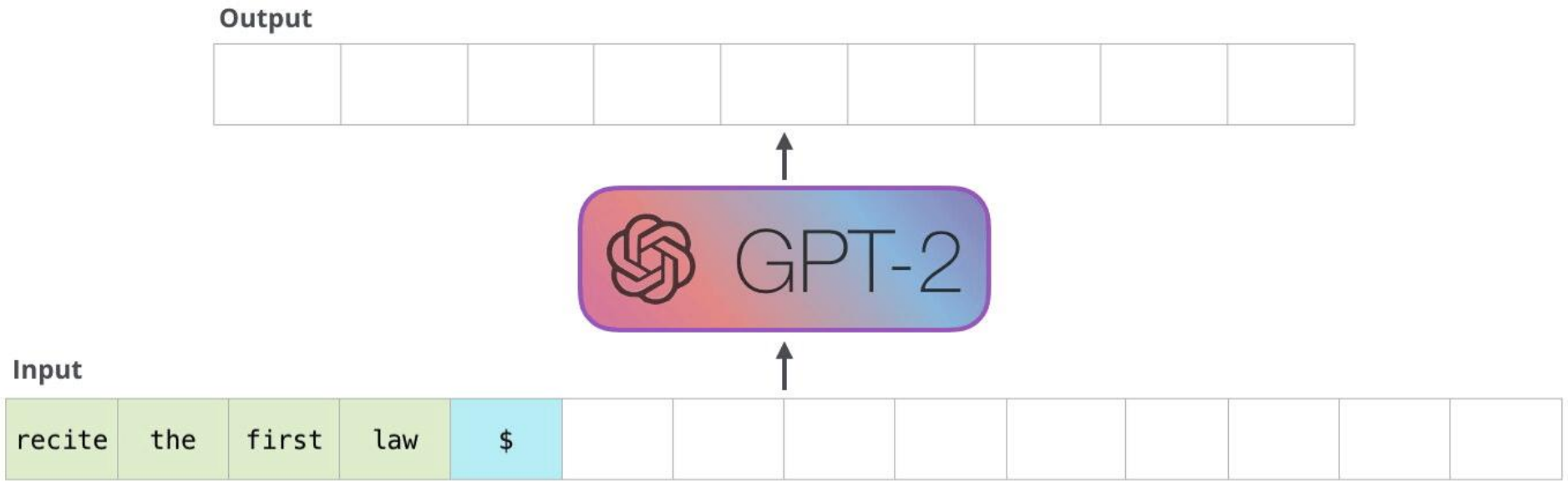
Como problema guía para la clase, estudiaremos la siguiente pregunta:

¿Cómo se distribuye y se comporta la demanda de buses en una ciudad, y qué factores podrían explicar sus variaciones?

Como insumo para el estudio, tenemos un archivo `.csv` con registros diarios de uso de buses en distintas comunas de Santiago

Antes del análisis, lo que nadie quiere que se sepa (en la industria tecnológica al menos) sobre los asistentes basados en IA

- Actuales asistentes basados en IA son, tras bambalinas, modelos autorregresivos que generan las palabras una a una.
- Esto significa que utilizan su propio texto generado como *input* para generar la siguiente palabra.



Antes del análisis, lo que nadie quiere que se sepa (en la industria tecnológica al menos) sobre los asistentes basados en IA

- Sea e la probabilidad de que una palabra generada nos saque del "camino" o "árbol" de la(s) respuesta(s) correcta(s).
- Dado esto, la probabilidad de que una respuesta de largo n sea correcta, asumiendo errores independientes en la generación de cada palabra, es la siguiente:

$$\mathcal{P}(correcta) = (1 - e)^n$$

- Esta probabilidad diverge de manera exponencial, por lo que la única solución es minimizar lo más posible el valor de e, lo que es prácticamente imposible de hacer en todos los casos.
- Esto significa que el riesgo de corrupción de las respuestas de los asistentes crece rápidamente con la extensión/complejidad de estas.

Moraleja: si le pido a un asistente que resuelva por si solo un problema extenso y/o que requiera múltiples pasos por su dificultad, es probable que la respuesta sea incorrecta.

Para el análisis de este archivo utilizaremos Pandas

- Biblioteca de Python que permite manipular, analizar y visualizar datos tabulares.
- Puede ser visto como una herramienta para trabajar datos almacenados en una estructura de tabla o de serie de tiempo.
- Se basa en 2 estructuras de datos (clases) principales:
  - DataFrame
  - Series

En un DataFrame de Pandas, cada columna es una Series

```
1 import pandas as pd
```

```
1 df = pd.read_csv("buses_santiago.csv")
2 df.shape
```

```
(5760, 8)
```

```
1 df.head()
```

	fecha	linea_bus	comuna_origen	comuna_destino	validaciones	clima	dia_semana	eventos_especiales
0	2024-05-16	L06	Las Condes	Pudahuel	18	Lluvia	Jueves	0
1	2024-05-23	L06	Independencia	Pudahuel	19	Soleado	Jueves	0
2	2024-04-23	L07	Ñuñoa	Las Condes	17	Soleado	Martes	0
3	2024-05-21	L02	Providencia	Independencia	19	Nublado	Martes	0
4	2024-05-26	L02	Maipú	Estación Central	10	Nublado	Domingo	0

En un DataFrame de Pandas, cada columna es una Series

```
1 df["validaciones"]
```

	validaciones
0	18
1	19
2	17
3	19
4	10
...	...
5755	26
5756	14
5757	9
5758	18
5759	10

5760 rows × 1 columns

dtype: int64

```
1 df.validaciones
```

	validaciones
0	18
1	19
2	17
3	19
4	10
...	...
5755	26
5756	14
5757	9
5758	18
5759	10

5760 rows × 1 columns

dtype: int64

Ahora que conocemos los aspectos generales sobre el formato y contenido de los datos, ¿cómo podemos avanzar en el problema?

- Necesitamos pensar y evaluar posibles caminos para responder a la pregunta
- Específicamente, necesitamos hacernos una visión general de la demanda y de las variables relacionadas.
- Para esto, el siguiente paso será calcular estadísticos básicos que permitan realizar un análisis exploratorio descriptivo.
- Lo fundamental es poder decir algo más de los datos y el problema, que sea útil para guiar los siguientes pasos de la resolución.

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

	validaciones	eventos_especiales
count	5444.000000	5760.000000
mean	17.311352	0.050000
std	15.350443	0.217964
min	4.000000	0.000000
25%	12.000000	0.000000
50%	15.500000	0.000000
75%	19.000000	0.000000
max	199.000000	1.000000

```
1 df["clima"].value_counts()
```

clima	count
Nublado	2147
Soleado	1862
Lluvia	1587

```
1 df["comuna_origen"].value_counts()
```

comuna_origen	count
Las Condes	480
Independencia	480
Ñuñoa	480
Providencia	480
Maipú	480
Recoleta	480
Santiago	480
Estación Central	480
La Florida	480
Pudahuel	480
Puente Alto	480
San Miguel	480

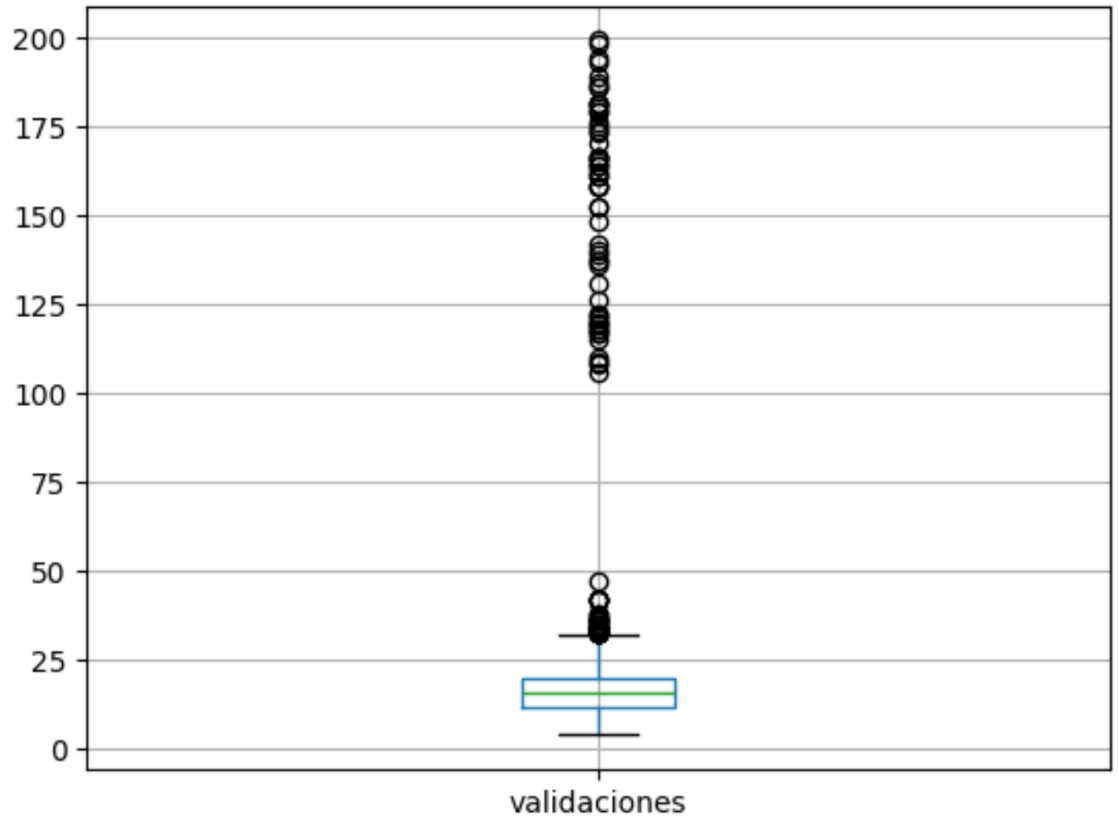
¿Qué cosas nuevas e interesantes podemos decir de los datos?

Si el análisis nos genera ciertas dudas sobre la validez de los datos, es necesario revisarlos en más detalle y corregirlos

```
1 df.isna().sum()
```

	0
fecha	0
linea_bus	0
comuna_origen	0
comuna_destino	0
validaciones	316
clima	164
dia_semana	0
eventos_especiales	0

```
1 df.boxplot(column="validaciones");
```



Estudiemos primero la imputación de valores faltantes

```
1 df["validaciones"].fillna(df["validaciones"].mean())
2 df.describe()
```

```
1 df["validaciones"] = df["validaciones"].fillna(df["validaciones"].mean())
2 df.describe()
```

Estudiemos primero la imputación de valores faltantes

```
1 df["validaciones"].fillna(df["validaciones"].mean())
2 df.describe()
```

	validaciones	eventos_especiales
count	5444.000000	5760.000000
mean	17.311352	0.050000
std	15.350443	0.217964
min	4.000000	0.000000
25%	12.000000	0.000000
50%	15.500000	0.000000
75%	19.000000	0.000000
max	199.000000	1.000000

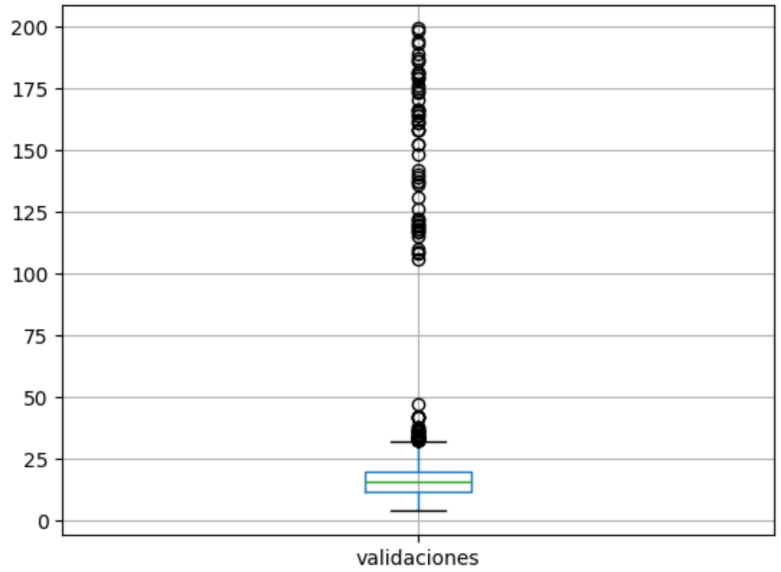
```
1 df["validaciones"] = df["validaciones"].fillna(df["validaciones"].mean())
2 df.describe()
```

	validaciones	eventos_especiales
count	5760.000000	5760.000000
mean	17.311352	0.050000
std	14.923357	0.217964
min	4.000000	0.000000
25%	12.000000	0.000000
50%	16.000000	0.000000
75%	19.000000	0.000000
max	199.000000	1.000000

Aún restan situaciones que corregir

- ¿Cómo podemos corregir los *outliers* de las validaciones? ¿Basta con eliminarlos? ¿Son realmente *outliers*?

```
1 df.boxplot(column="validaciones");
```



Aún restan situaciones que corregir

- ¿Cómo podemos corregir los *outliers* de las validaciones? ¿Basta con eliminarlos? ¿Son realmente *outliers*?
- ¿Qué hacemos con la variable `clima`? ¿Por qué es distinta `clima` de `validaciones`?
    - Imputación simple con la moda (valor más frecuente)
    - Imputación condicional por mes o estación
    - Inferencia a partir de variables relacionadas

```
1 df["clima"].value_counts()
```

clima	count
Nublado	2147
Soleado	1862
Lluvia	1587

Revisitemos parte del análisis exploratorio inicial

```
1 df["comuna_origen"].value_counts()
```

comuna_origen	count
Las Condes	480
Independencia	480
Ñuñoa	480
Providencia	480
Maipú	480
Recoleta	480
Santiago	480
Estación Central	480
La Florida	480
Pudahuel	480
Puente Alto	480
San Miguel	480

```
1 df["comuna_destino"].value_counts()
```

comuna_destino	count
Estación Central	534
Maipú	492
Recoleta	488
Las Condes	488
La Florida	482
Puente Alto	481
Providencia	480
San Miguel	479
Independencia	476
Pudahuel	466
Ñuñoa	452
Santiago	442

Claramente la información tiene sesgo
¿qué información externa nos podría ayudar a corregirla?

Pandas permite incorporar información de otras fuentes en un mismo `DataFrame`

```
poblacion = pd.read_csv("poblacion_comunas.csv")
poblacion.head()
```

	comuna_origen	poblacion
0	Santiago	404495
1	Providencia	151000
2	Las Condes	300000
3	Ñuñoa	240000
4	La Florida	366000

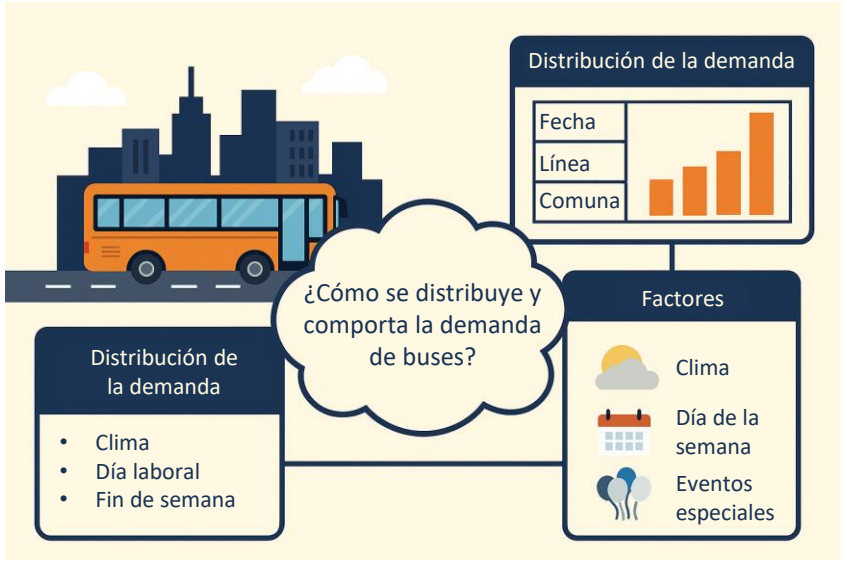
```
df = pd.merge(df, poblacion, on="comuna_origen", how="left")
df.head()
```

	fecha	linea_bus	comuna_origen	comuna_destino	validaciones	clima	dia_semana	eventos_especiales	poblacion
0	2024-05-16	L06	Las Condes	Pudahuel	18.0	Lluvia	Jueves	0	300000
1	2024-05-23	L06	Independencia	Pudahuel	19.0	Soleado	Jueves	0	150000
2	2024-04-23	L07	Ñuñoa	Las Condes	17.0	Soleado	Martes	0	240000
3	2024-05-21	L02	Providencia	Independencia	19.0	Nublado	Martes	0	151000
4	2024-05-26	L02	Maipú	Estación Central	10.0	Nublado	Domingo	0	520000

Centrémonos ahora en el estudio de la demanda

Dado que ya tenemos los datos “preparados”, es posible comenzar a evaluar hipótesis que expliquen el comportamiento de la demanda.

- ¿Qué factores pueden afectar a la demanda?
- ¿Existen subgrupos con patrones distintos de otros?
- ¿Hay nuevas variables relevantes que se pueden crear a partir de las actuales?
- Y muchos más…



Centrémonos ahora en el estudio de la demanda

Enfoquemos el análisis inicial en ciertos subconjuntos de datos y además en derivar nuevas variables que capturen información relevante. Algunas preguntas relevantes son las siguientes:

- ¿Qué criterios son útiles para filtrar datos? (ej. analizar solo horas punta, o excluir líneas con pocos registros).

- ¿Qué nuevas variables pueden enriquecer el análisis sin tener que recurrir a datos externos? (ej. variable binaria de fin de semana, discretización de población en tramos).

```python
df[["fecha", "linea_bus", "validaciones"]]
```

	fecha	linea_bus	validaciones
0	2024-05-16	L06	18.0
1	2024-05-23	L06	19.0
2	2024-04-23	L07	17.0
3	2024-05-21	L02	19.0
4	2024-05-26	L02	10.0
...	...	...	...
5755	2024-05-10	L03	26.0
5756	2024-05-25	L01	14.0
5757	2024-05-25	L04	9.0
5758	2024-05-27	L02	18.0
5759	2024-04-09	L08	10.0

5760 rows × 3 columns

```python
df[df["dia_semana"].isin(["Lunes","Martes","Miércoles","Jueves","Viernes"])]
```

	fecha	linea_bus	comuna_origen	comuna_destino	validaciones	clima	dia_semana	eventos_especiales	poblacion
0	2024-05-16	L06	Las Condes	Pudahuel	18.0	Lluvia	Jueves	0	300000
1	2024-05-23	L06	Independencia	Pudahuel	19.0	Soleado	Jueves	0	150000
2	2024-04-23	L07	Ñuñoa	Las Condes	17.0	Soleado	Martes	0	240000
3	2024-05-21	L02	Providencia	Independencia	19.0	Nublado	Martes	0	151000
5	2024-04-12	L04	Recoleta	Independencia	11.0	Lluvia	Viernes	0	180000
...	...	...	...	...	...	...	...	...	...
5753	2024-05-30	L06	San Miguel	Providencia	15.0	Soleado	Jueves	0	120000
5754	2024-05-03	L02	Independencia	Puente Alto	11.0	Lluvia	Viernes	0	150000
5755	2024-05-10	L03	La Florida	Las Condes	26.0	Soleado	Viernes	0	366000
5758	2024-05-27	L02	Las Condes	Puente Alto	18.0	Soleado	Lunes	0	300000
5759	2024-04-09	L08	Independencia	Ñuñoa	10.0	Lluvia	Martes	0	150000

4224 rows × 9 columns

```python
df["fin_de_semana"] = df["dia_semana"].isin(["Sábado","Domingo"])
df["densidad_validaciones"] = df["validaciones"] / df["poblacion"]
```

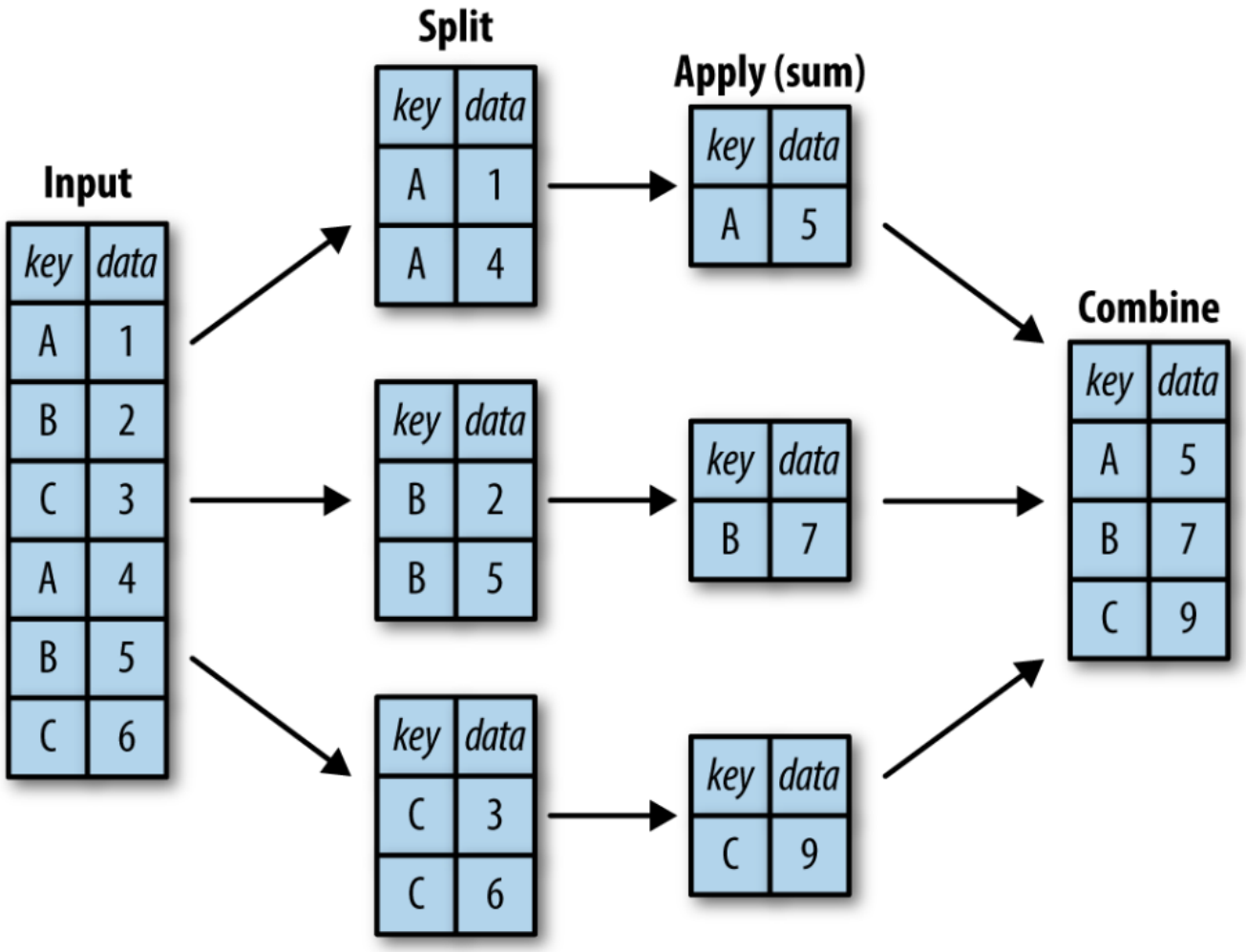
Agregación es otro tipo de análisis útil que podemos hacer

Busquemos también entender tendencias globales. Por ejemplo, para entender cómo cambia la demanda de buses según el día de la semana y la línea de bus, podemos preguntarnos lo siguiente:

- ¿Qué métricas son más informativas? (ej. media vs. mediana para evitar sesgo por *outliers*).

- ¿Qué nivel de granularidad es el adecuado? (por línea, por comuna, por día).

Función groupby de pandas permite combinar todo el procesamiento agregado



Pandas provee múltiples funciones de agregación

Aggregation	Description
`count()`	Total number of items
`first()`, `last()`	First and last item
`mean()`, `median()`	Mean and median
`min()`, `max()`	Minimum and maximum
`std()`, `var()`	Standard deviation and variance
`mad()`	Mean absolute deviation
`prod()`	Product of all items
`sum()`	Sum of all items

```
1 df.groupby("dia_semana")["validaciones"].mean()
```

dia_semana	validaciones
Domingo	10.056175
Jueves	19.468097
Lunes	19.430263
Martes	19.509064
Miércoles	19.949010
Sábado	14.038438
Viernes	17.589538

dtype: float64

```
df.groupby(["linea_bus","dia_semana"])["validaciones"].agg(["mean","median","std"])
```

linea_bus	dia_semana	mean	median	std
L01	Domingo	10.168536	10.000000	3.078729
	Jueves	19.167718	17.000000	19.981879
	Lunes	16.958859	16.000000	4.062942
	Martes	19.150420	16.000000	17.407154
	Miércoles	19.807828	17.000000	20.221883
	Sábado	11.724550	11.000000	2.957183
	Viernes	14.887973	14.000000	3.575008
L02	Domingo	10.876869	10.000000	2.922466
	Jueves	19.683353	17.655676	10.500659
	Lunes	20.062983	18.000000	15.166870
	Martes	21.141161	19.000000	17.254086
	Miércoles	21.542192	20.000000	17.048387
	Sábado	14.353480	12.500000	13.339652
	Viernes	18.794910	17.000000	14.406436
L03	Domingo	12.207646	11.000000	9.958066
	Jueves	23.297958	21.000000	17.842755

Tablas dinámicas son otra forma de agrupar

Queremos cruzar múltiples dimensiones (ej. clima × día de semana × comuna) y además categorizar variables continuas como población o número de validaciones.

Para esto, algunas preguntas interesantes son las siguientes:

- ¿Cuándo una tabla dinámica facilita la lectura frente a un groupby?
- ¿Por qué discretizar variables continuas? (ej. población en comunas pequeñas vs. grandes; validaciones bajas, medias y altas).
- ¿Qué criterios usar para elegir puntos de corte? (cuantiles, umbrales de política pública, etc.).

```
1 pd.pivot_table(df, values="validaciones", index="clima", columns="dia_semana", aggfunc="mean")
```

dia_semana clima	Domingo	Jueves	Lunes	Martes	Miércoles	Sábado	Viernes
Lluvia	9.752944	15.517679	20.268108	18.161643	NaN	12.641847	16.437221
Nublado	9.854344	18.720331	19.127955	19.915195	18.971142	13.458199	NaN
Soleado	10.770664	20.207633	20.326312	21.214691	21.507261	17.178143	21.236303

```python
df["tramo_poblacion"] = pd.cut(df["poblacion"], bins=[0,200000,400000,700000], labels=["Baja","Media","Alta"])
pd.pivot_table(df, values="validaciones",index="tramo_poblacion", columns="clima", aggfunc="mean", observed=True)
```

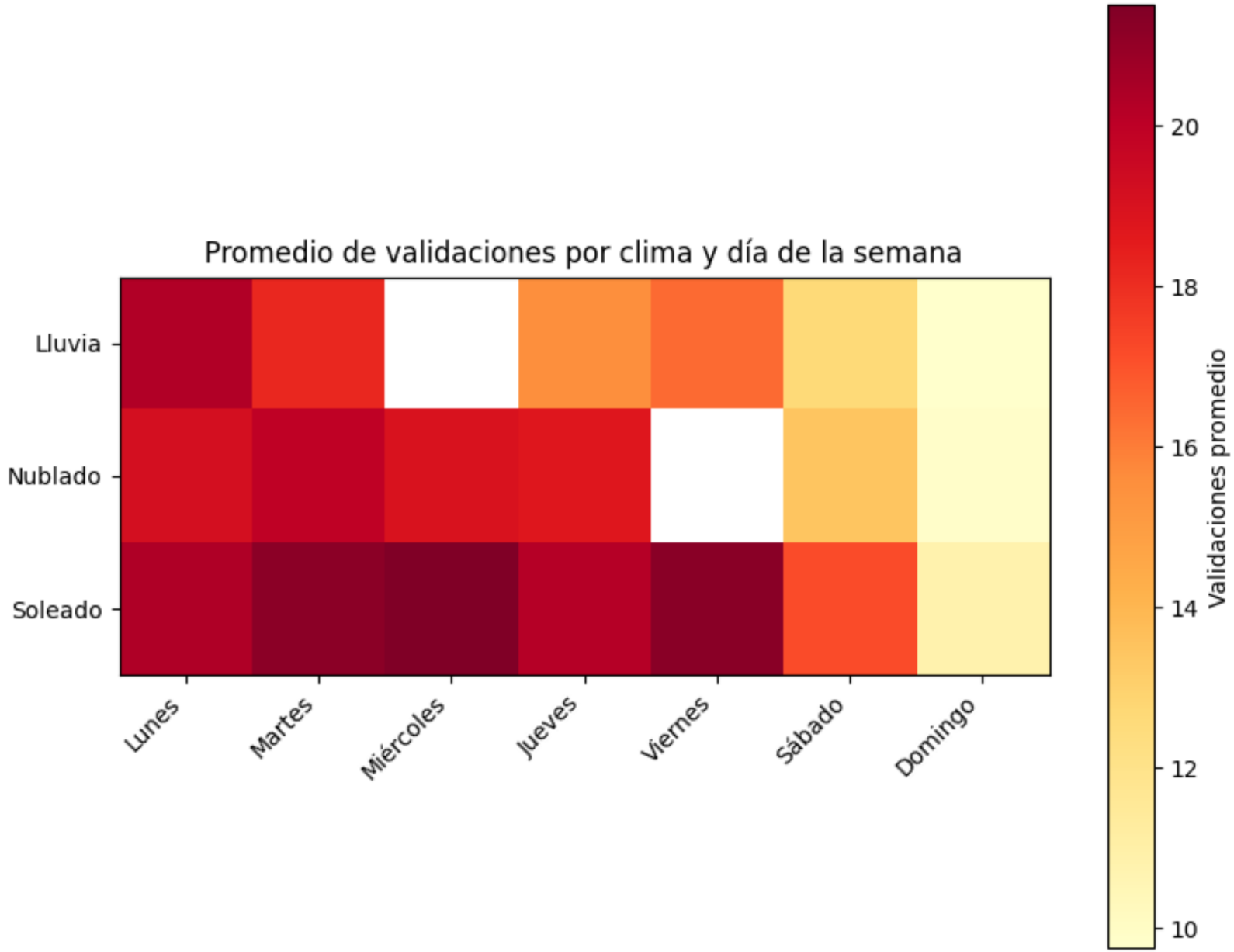
clima	Lluvia	Nublado	Soleado
tramo_poblacion			
Baja	13.960490	15.179606	17.795746
Media	15.632862	16.971555	19.459775
Alta	18.143833	19.122837	22.082659

Siempre podemos complementar el análisis a través de la visualización

Necesitamos mostrar los patrones hallados de forma clara. Ejemplo: la caída de la demanda en días lluviosos, o las diferencias entre líneas. Acá debemos siempre preguntarnos:

- ¿Qué tipo de gráfico es más adecuado según la variable? (barras, histograma, boxplot, mapa de calor, etc.)
- ¿Cuáles son los riesgos de leer demasiado en un gráfico? (interpretar ruido como tendencia, escalas incorrectas, etc.)

```python
import matplotlib.pyplot as plt

pivot = pd.pivot_table(df, values="validaciones", index="clima", columns="dia_semana", aggfunc="mean")

dias = ["Lunes","Martes","Miércoles","Jueves","Viernes","Sábado","Domingo"]
pivot = pivot[dias]

fig, ax = plt.subplots(figsize=(8,6))
cax = ax.imshow(pivot, cmap="YlOrRd")


ax.set_xticks(range(len(pivot.columns)))
ax.set_xticklabels(pivot.columns, rotation=45, ha="right")
ax.set_yticks(range(len(pivot.index)))
ax.set_yticklabels(pivot.index)

ax.set_title("Promedio de validaciones por clima y día de la semana")
fig.colorbar(cax, ax=ax, label="Validaciones promedio")

plt.tight_layout()
plt.show()
```


Promedio de validaciones por clima y día de la semana
Lluvia
Nublado
Soleado
Lunes
Martes
Miércoles
Jueves
Viernes
Sábado
Domingo
Validaciones promedio
20
18
16
14
12
10

Cómo sigue la sesión

- Para este capítulo también hay 3 ejercicios formativos disponibles, teniendo 2 de ellos una versión más guiada.
- El primer ejercicio tiene formato tutorial, con el objetivo de cubrir lo más básico de la materia.
- El segundo es más avanzado les permitirá practicar múltiples comandos y técnicas.
- El tercero es más realista y servirá para preparar el laboratorio.
- NO UTILICEN LOS ASISTENTES PARA ANÁLISIS EXTENSOS, SUBDIVIDAN EL PROBLEMA PARA ASEGURAR CORRECTITUD.
- No olviden responder el ticket de salida, siempre considerando que lo indicado en él debe verse reflejado en el repositorio privado.
- A las 17:30 se cerrará el ticket de manera definitiva.

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2115 - Programación como herramienta para la ingeniería

Manejo y análisis de datos tabulares

Profesor: Hans Löbel