

Amgad Abdallah

Introduction to python programming



Developer Student Clubs
Al-Azhar University

AGENDA

01

Type of programming language

02

Why python?

03

Python application

04

Python advantages and disadvantage

05

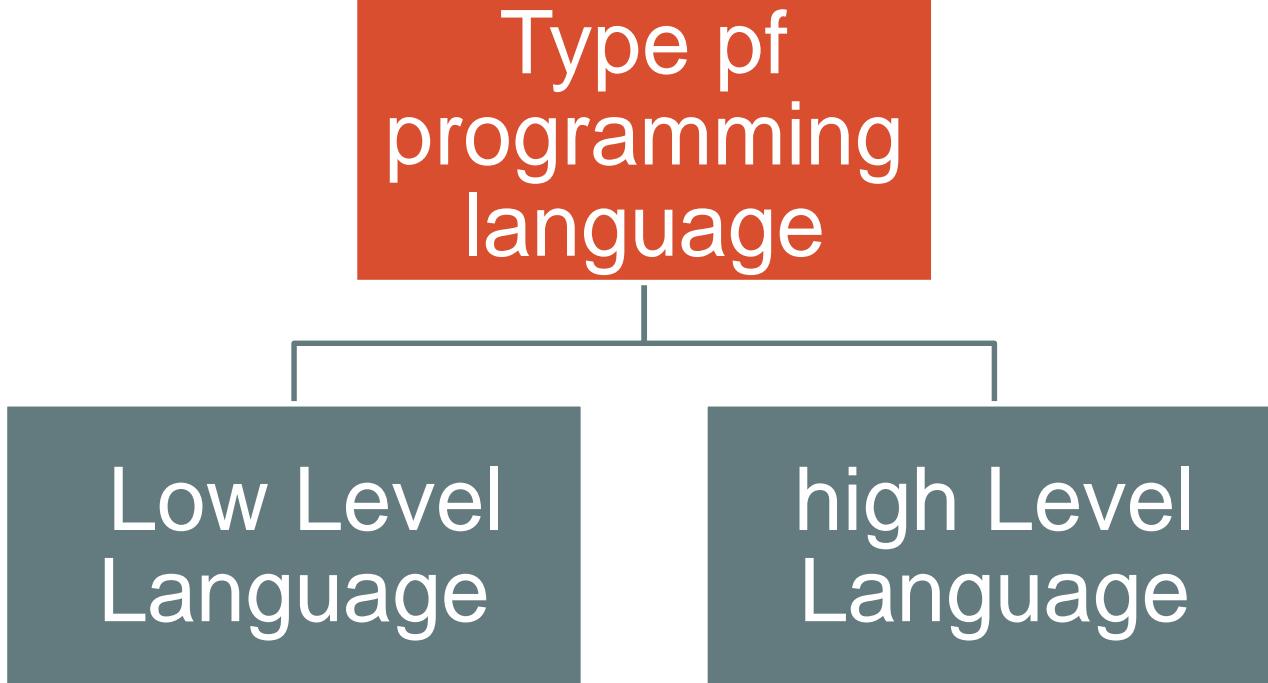
Python basics syntax,
Data type and variable

01

Type of programming language



Type pf programming language



```
graph TD; A[Type pf programming language] --> B[Low Level Language]; A --> C[high Level Language]
```

Low Level
Language

high Level
Language

Low level language



The Language of
Computer



It's easy to be executed

01

It's consists of (0,1)



Can run only one kind
of computer

High level language



Easy to program



Easy to be corrected



Takes less to be Coded



Portable(can run on
any pc)

02

Why python

Data science

Machine
learning

Ease to use

salary

Libraries and
framework

Web
development

speedy

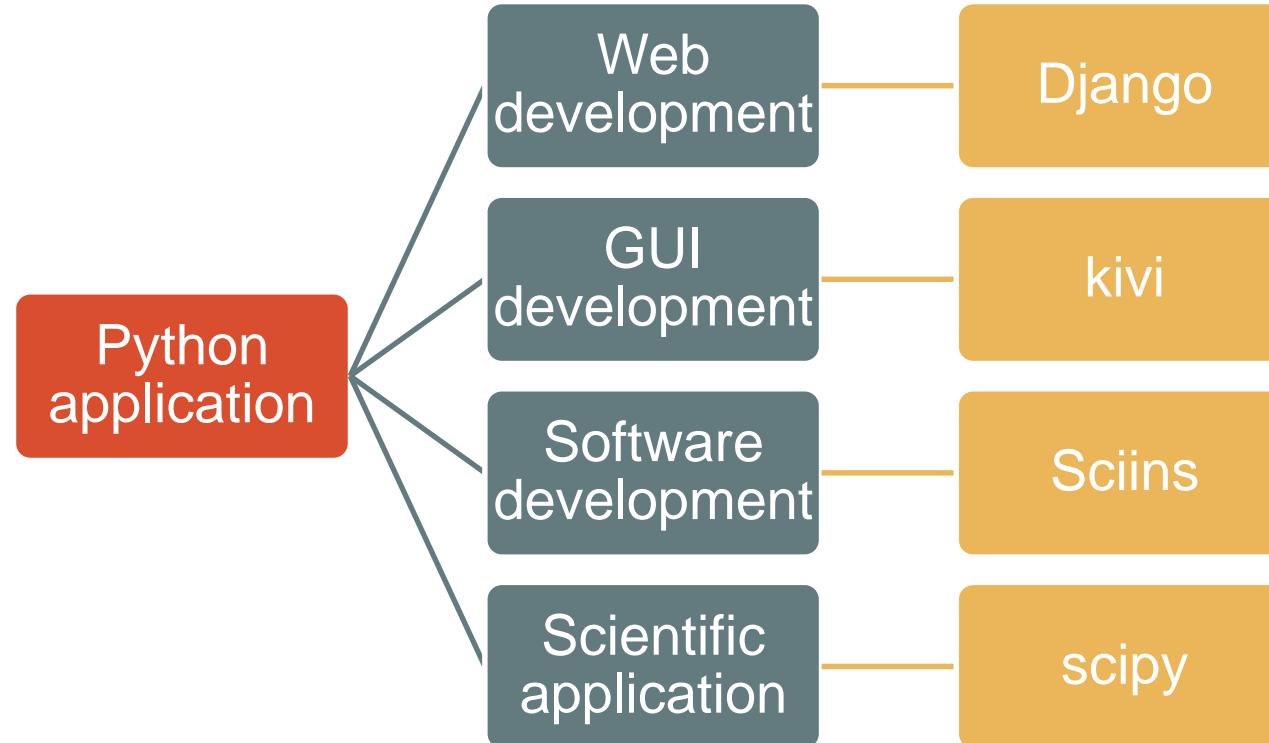


Rank	Language	Type	Score
1	Python	🌐💻⚙️	100.0
2	Java	🌐📱💻	96.3
3	C	📱💻⚙️	94.4
4	C++	📱💻⚙️	87.5
5	R	💻	81.5
6	JavaScript	🌐	79.4
7	C#	🌐📱💻⚙️	74.5
8	Matlab	💻	70.6
9	Swift	📱💻	69.1
10	Go	🌐💻	68.0

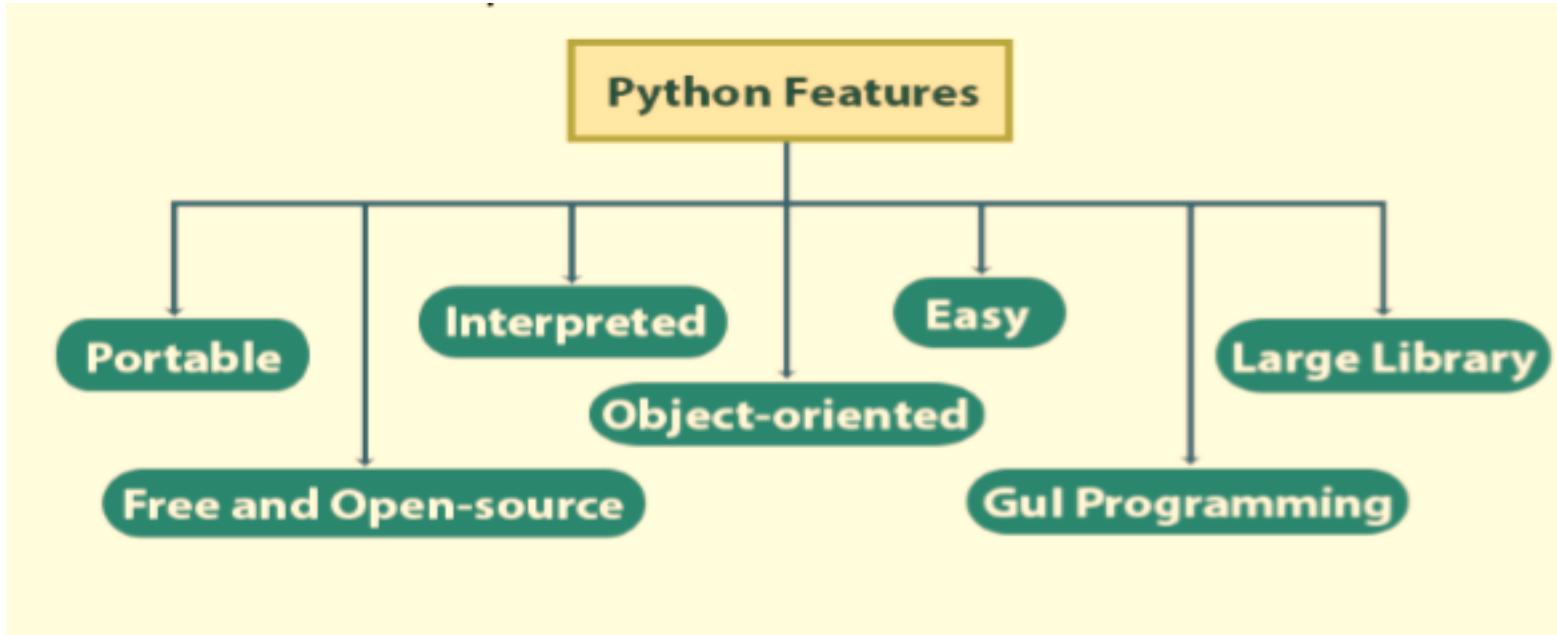
IEEE trends spectrum 2019

03

Python application



More Features:



Python Applications

- Build a website
- Develop a game
- Perform Computer Vision (Facilities like face-detection and color-detection)
- Implement Machine Learning (Give a computer the ability to learn)
- Enable Robotics
- Perform Web Scraping (Harvest data from websites)
- Perform Data Analysis
- Automate a web browser
- Perform Scripting
- Perform Scientific
- Build Artificial Intelligence

04

Python advantages and disadvantage



Python Advantages & Disadvantages



Data Science
Library



Embeddable



IOT Opportunities



Object-Oriented



Extensible



Improved
Productivity



Readable



Portable, Free
and Open-Source

ADVANTAGES



Speed Limitations



Weak in Mobile
Computing



Design Restrictions



Underdeveloped
DB Layers

DISADVANTAGES

Python Editors

- 1) PyCharm



- 2) Spyder



- 3) IDLE



- 4) Sublime Text 3



- 5) Visual Studio Code



- 6) Jupyter



Anaconda (jupyter notebook).
Colab.



05

Python basics syntax, Data type and variable

Some Basics..

1. Variables: is a name that refers to a value. Legal Variable Names Variable names can consist of any number of letters, underscores and digits, Variable should not start with a number.

Examples:

- x = True # valid
- _y = True # valid
- 9x = False # starts with numeral
- => SyntaxError: invalid syntax
- \$y = False # starts with symbol
- => SyntaxError: invalid syntax

Python Keywords are not allowed as variable names.

Variable names are case-sensitive. For example, computer and Computer are different variables

List of Keywords in Python

and	as	not
assert	finally	or
break	for	pass
class	from	nonlocal
continue	global	raise
def	if	return
del	import	try
elif	in	while
else	is	with
except	lambda	yield
False	True	None

Assigning Values to Variables:

- The general format for assigning values to variables is as follows: variable_name = expression

Single assignment:

- X=100
- X="hundred"
- **Multi assignment:**
- X,Y=3,4
- X=Y=Z=2

Variable types examples:

```
# Integer  
a = 2  
print(a)  
# Output: 2  
# Integer  
b = 9223372036854775807  
print(b)  
# Output: 9223372036854775807  
# Floating point  
pi = 3.14  
print(pi)  
# Output: 3.14
```

```
# String  
c = 'A'  
print(c)  
# Output: A  
# String  
name = 'John Doe'  
print(name)  
# Output: John Doe
```

```
# Boolean  
q = True  
print(q)  
# Output: True  
# Empty value or null data  
type  
x = None  
print(x)  
# Output: None
```

How to know the type of the variable?

- Pretty easy...

Just use `type()` function.

EX:

```
>>> x = 5  
>>> y = "Hello"  
>>> type(x); type(y)  
• <class 'int'>  
• <class 'str'>
```

Operators:

1. Arithmetic Operators

2. Assignment Operators

3. Comparison Operators

4. Logical Operators

5. Bitwise Operators

List of Arithmetic Operators

Operator	Operator Name	Description	Example
+	Addition operator	Adds two operands, producing their sum.	$p + q = 5$
-	Subtraction operator	Subtracts the two operands, producing their difference.	$p - q = -1$
*	Multiplication operator	Produces the product of the operands.	$p * q = 6$
/	Division operator	Produces the quotient of its operands where the left operand is the dividend and the right operand is the divisor.	$q / p = 1.5$
%	Modulus operator	Divides left hand operand by right hand operand and returns a remainder.	$q \% p = 1$
**	Exponent operator	Performs exponential (power) calculation on operators.	$p^{**}q = 8$
//	Floor division operator	Returns the integral part of the quotient.	$9 // 2 = 4$ and $9.0 // 2.0 = 4.0$

Note: The value of p is 2 and q is 3.

List of Assignment Operators

Operator	Operator Name	Description	Example
=	Assignment	Assigns values from right side operands to left side operand.	$z = p + q$ assigns value of $p + q$ to z
+=	Addition Assignment	Adds the value of right operand to the left operand and assigns the result to left operand.	$z += p$ is equivalent to $z = z + p$
-=	Subtraction Assignment	Subtracts the value of right operand from the left operand and assigns the result to left operand.	$z -= p$ is equivalent to $z = z - p$
*=	Multiplication Assignment	Multiplies the value of right operand with the left operand and assigns the result to left operand.	$z *= p$ is equivalent to $z = z * p$
/=	Division Assignment	Divides the value of right operand with the left operand and assigns the result to left operand.	$z /= p$ is equivalent to $z = z / p$
**=	Exponentiation Assignment	Evaluates to the result of raising the first operand to the power of the second operand.	$z **= p$ is equivalent to $z = z ** p$
//=	Floor Division Assignment	Produces the integral part of the quotient of its operands where the left operand is the dividend and the right operand is the divisor.	$z //= p$ is equivalent to $z = z // p$
%=	Remainder Assignment	Computes the remainder after division and assigns the value to the left operand.	$z %= p$ is equivalent to $z = z \% p$

List of Comparison Operators

Operator	Operator Name	Description	Example
<code>==</code>	Equal to	If the values of two operands are equal, then the condition becomes True.	$(p == q)$ is not True.
<code>!=</code>	Not Equal to	If values of two operands are not equal, then the condition becomes True.	$(p != q)$ is True
<code>></code>	Greater than	If the value of left operand is greater than the value of right operand, then condition becomes True.	$(p > q)$ is not True.
<code><</code>	Lesser than	If the value of left operand is less than the value of right operand, then condition becomes True.	$(p < q)$ is True.
<code>>=</code>	Greater than or equal to	If the value of left operand is greater than or equal to the value of right operand, then condition becomes True.	$(p >= q)$ is not True.
<code><=</code>	Lesser than or equal to	If the value of left operand is less than or equal to the value of right operand, then condition becomes True.	$(p <= q)$ is True.

Note: The value of p is 10 and q is 20.

Example

- 1. `>>>10 == 12`
False
- 2. `>>>10 != 12`
• True
- 3. `>>>10 < 12`
• True
- 4. `>>>10 > 12`
• False
- 5. `>>>10 <= 12` • True
- 6. `>>>10 >= 12`
• False
- 7. `>>> "P" < "Q"`
• True
- 8. `>>> "Aston" > "Asher"`
• True
- 9. `>>> True == True`
• True

List of Logical Operators

Operator	Operator Name	Description	Example
and	Logical AND	Performs AND operation and the result is True when both operands are True	p and q results in False
or	Logical OR	Performs OR operation and the result is True when any one of both operand is True	p or q results in True
not	Logical NOT	Reverses the operand state	not p results in False

Note: The Boolean value of p is True and q is False.

Boolean Logic Truth Table

P	Q	P and Q	P or Q	Not P
True	True	True	True	False
True	False	False	True	False
False	True	False	True	True
False	False	False	False	True

Example

- 1. `>>> True and False`
• False
- 2. `>>> True or False`
• True
- 3. `>>> not(True) and False`
• False
- 4. `>>> not(True and False)`
• True
- 5. `>>> (10 < 0) and (10 > 2)`
• False
- 6. `>>> (10 < 0) or (10 > 2)`
True
- 7. `>>> not(10 < 0) or (10 > 2)`
True
- 8. `>>> not(10 < 0 or 10 > 2)`
False

THANKS

Any questions?



Developer Student Clubs

Al-Azhar University





- Colaboratory, or “**Colab**
- is a product from **Google** Research.
- **Colab** allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education.
- It runs entirely on the cloud
- Sorted in google drive
- Libraries are included

A large, stylized word "colab" in a bold, sans-serif font. The letters are primarily yellow, with some orange and white highlights, giving it a three-dimensional appearance. The "o" has a yellow-to-white gradient fill, while the "l", "a", and "b" are solid yellow with white outlines.



Data type

- Numeric
 - Integer
 - Float
 - Complex number
- Sequence type
 - String
 - List
 - Tuple
- Set
- Boolean
- Dictionary

Statements

Functions

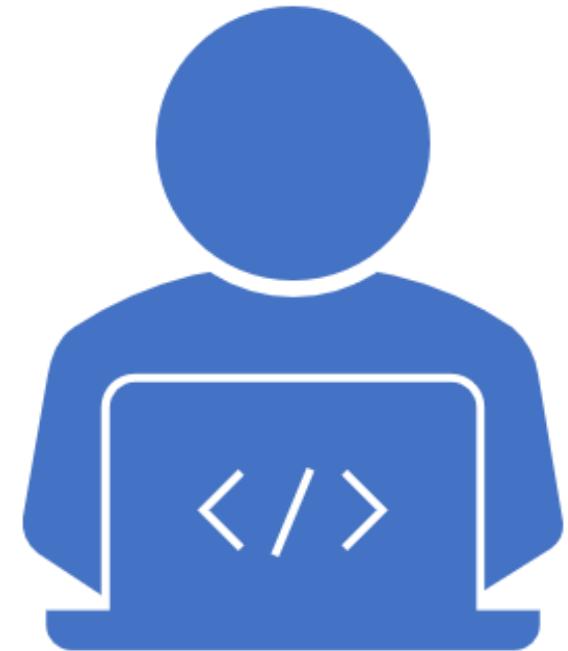
OOP

pandas

Numpy

Data type

- a data type is an attribute of data which tells the compiler or interpreter how the programmer intends to use the data.
- Classifications of data
- A=10
- b = “hello”



PYTHON DATA TYPE

Numeric

Sequence
type

Dictionary

Set

Boolean

Integer

Strings

Float

List

Complex
number

Tuple

Numeric

Integer

Float

Complex
number

{... -3, -2, -1, 0, 1, 2, 3 ...}

Negative integers

Positive integers

0 is neither positive nor negative

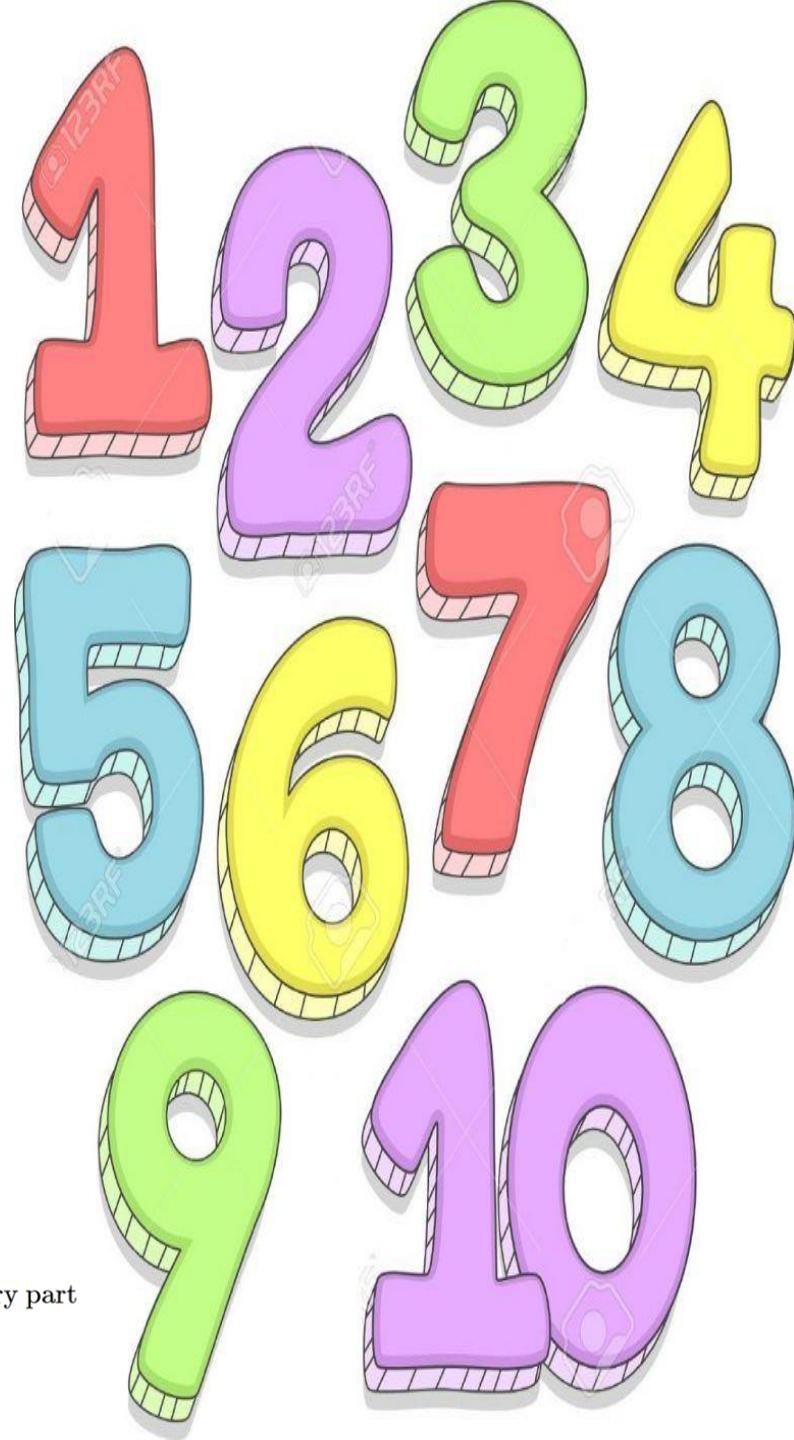
1 3 6 . 5 1

decimal point

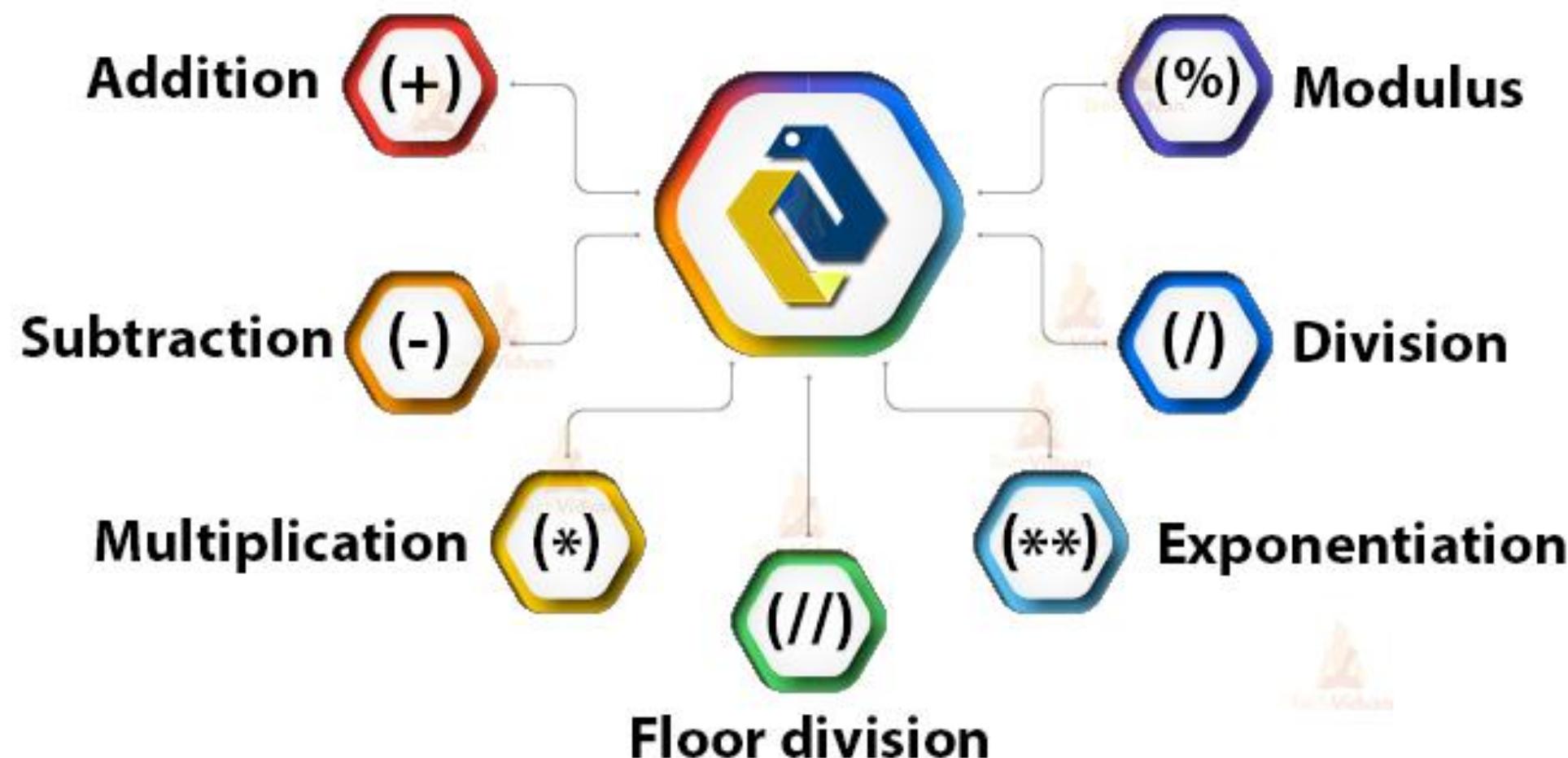
$a + bi$

↑ ↑

Real part Imaginary part



Python Arithmetic Operators



Sequence type

Strings

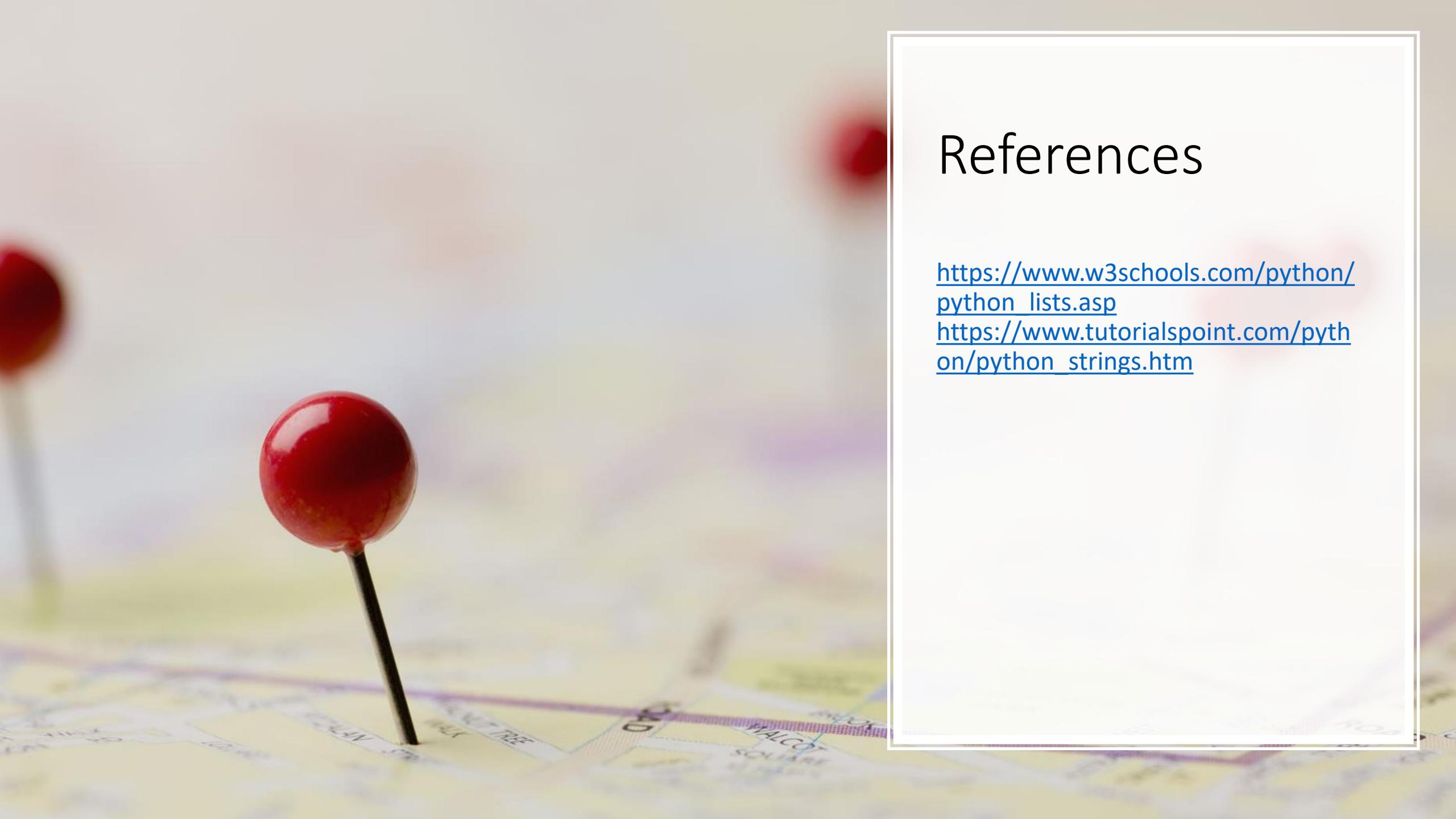
“Hello”
0 1 2 3 4
-5 -4 -3 -2 -1

List

[‘h’ ,‘e’ , ‘l’ , ‘l’ , ‘o’]

Tuple

(‘h’ ,‘e’ , ‘l’ , ‘l’ , ‘o’)

A close-up photograph of a red pushpin with a black stem, pinned into a yellowish-green map. The map shows street names like "WALCOT" and "SQUARE" and numbers like "40".

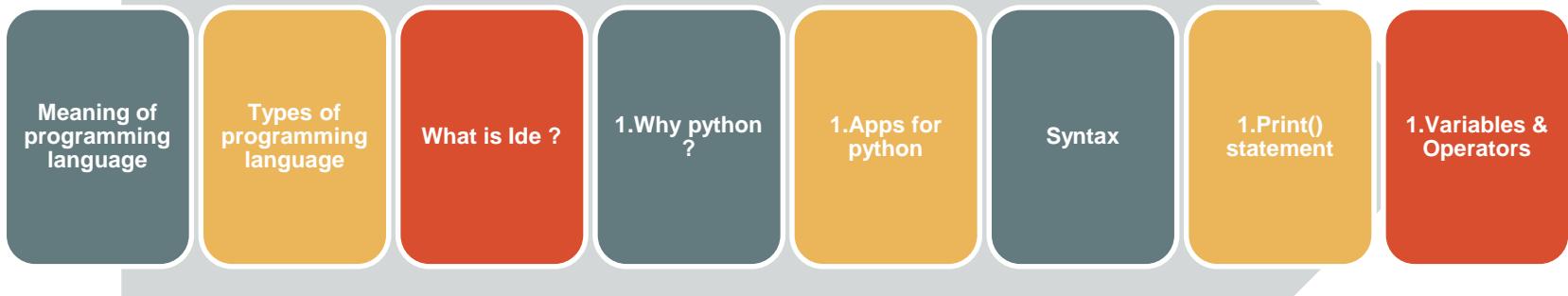
References

https://www.w3schools.com/python/python_lists.asp
https://www.tutorialspoint.com/python/python_strings.htm

Doaa Mostafa
Amgad Abdallah

Python





A close-up photograph of a man with dark hair and a well-groomed mustache. He is wearing a white t-shirt and is captured in the middle of a performance, singing into a black microphone. His eyes are closed, and his mouth is open as if he is singing. The background is dark and out of focus.

جائز

110101010101010010100010
100101110000010001001010
010101010100101110001010
010001010010101110001010
010101010000111101010010

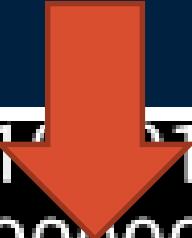


- Computers only understand (0 , 1)

- We need to turn
that code to (0 ,1)

```
>>> print("Hello World!")  
Hello World!
```

```
>>>
```



```
11010101011001001000100010  
1001011100000100010010101  
0101010101001011100011  
010001010010101110001001  
010101010000111110101010  
010101010101010100100011
```

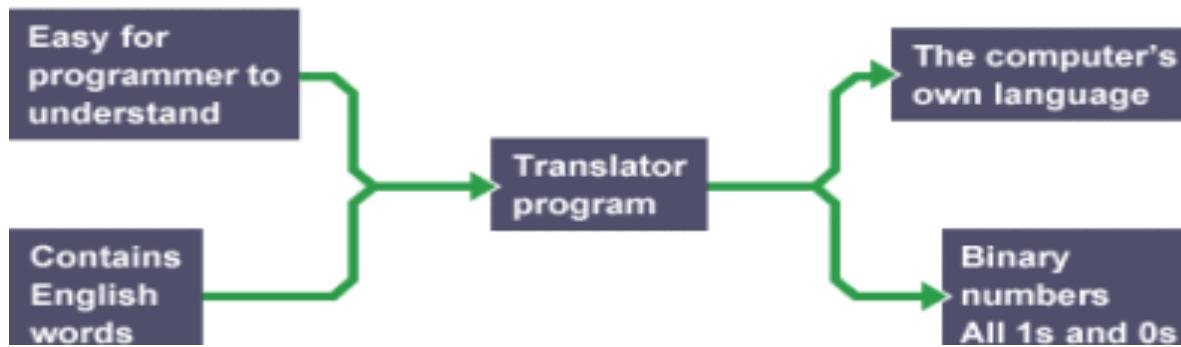
So we use Translator to link between them



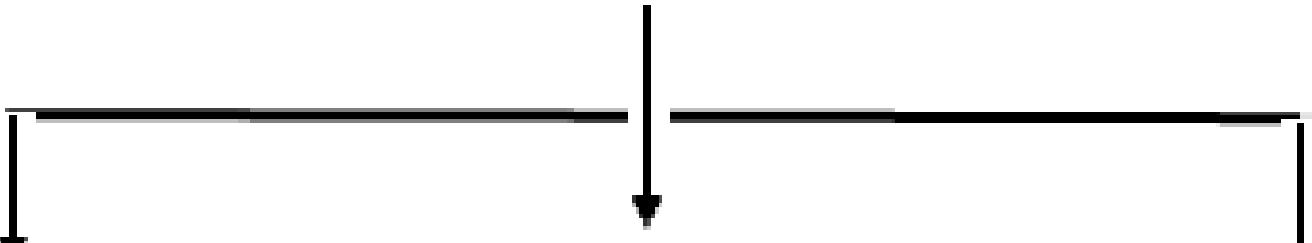
High level language



Machine code



Translators



Assembler

Compiler

Interpreter

Difference between the translators

Assembler	Compiler	Interpreter
Translate low level language into machine code	Translate high level language into machine code	Translate high level language into machine code
Written for particular hardware	Written for particular language	Written for particular language
One instruction translates to one instruction (one to one)	One instruction translates to many instructions (one to many)	One instruction translates to many instructions
Translates entire program before running	Translates entire program before running	Translates program instruction by instruction until an either completed or error detected
They produce object code		It does not produce an object

Computer Languages

**Low Level Language
(Machine Language)**

Use 1' s & 0' s to
create instructions

Ex: Binary Language

**Middle Level Language
(Assembly Language)**

Use mnemonics to
create instructions

Assembly Language

High Level Language

Similar to
human language

COBOL, FORTRAN, BASIC
C, C++, JAVA



Visual
Studio



What is an IDE?

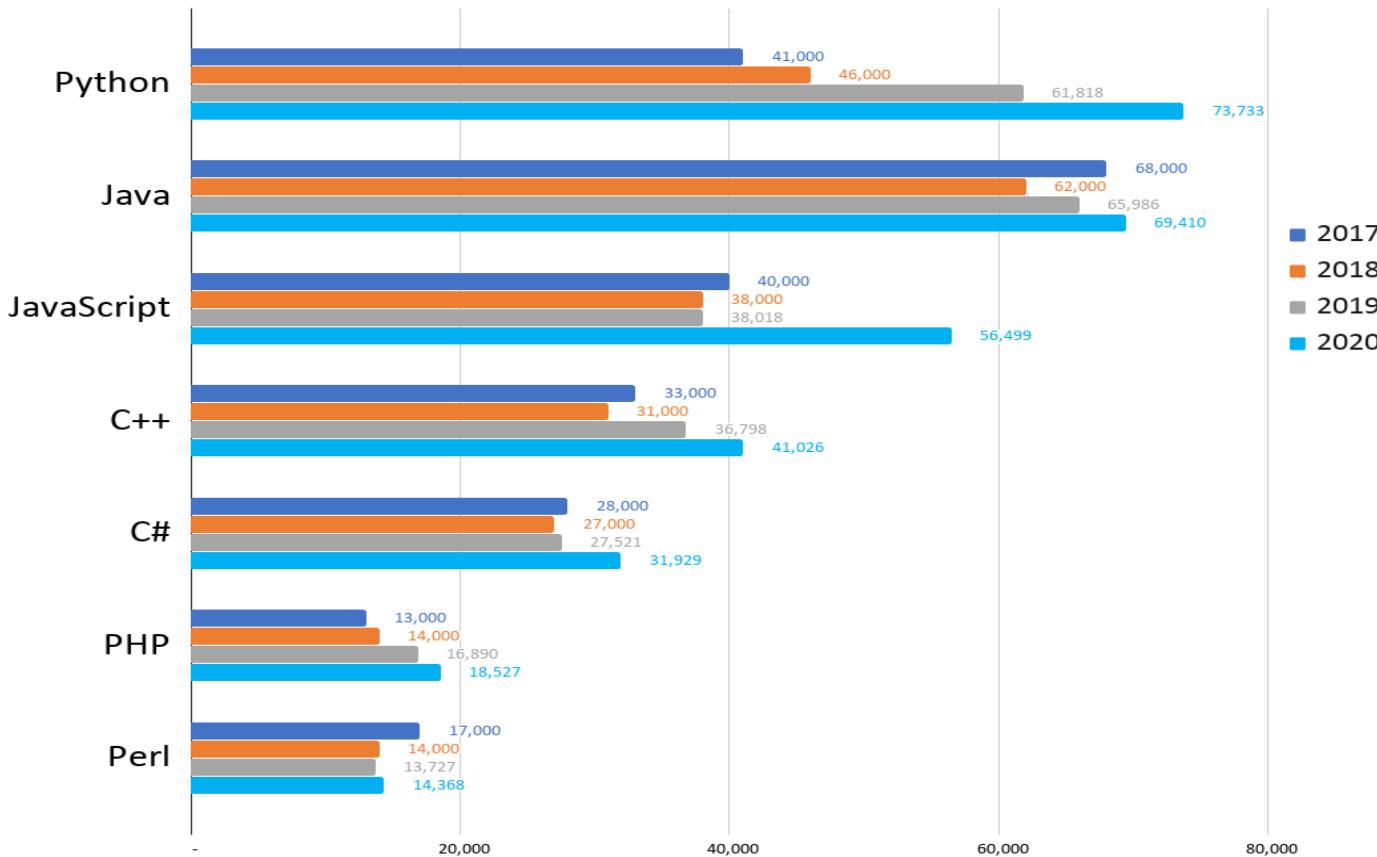
An IDE, or Integrated Development Environment, enables programmers to consolidate the different aspects of writing a computer program.



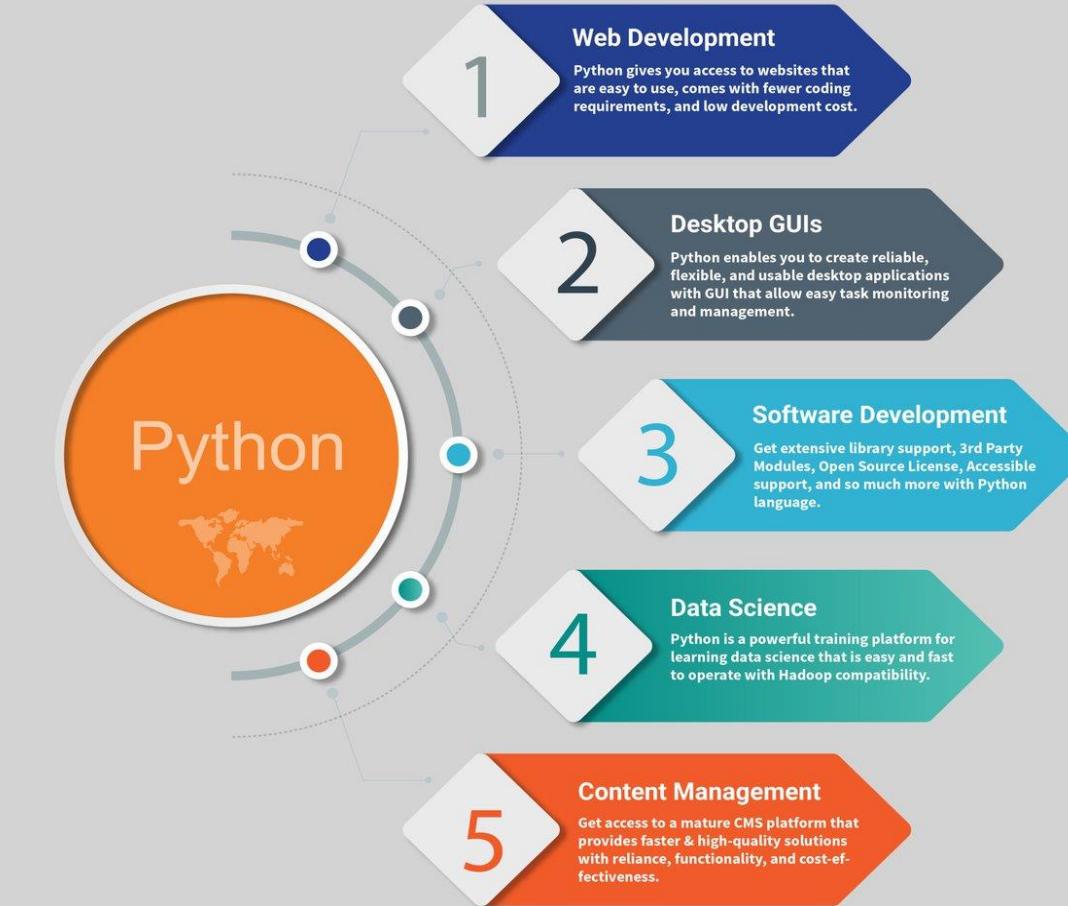
Colab
is a free Jupyter
notebook environment that
runs entirely in the cloud.

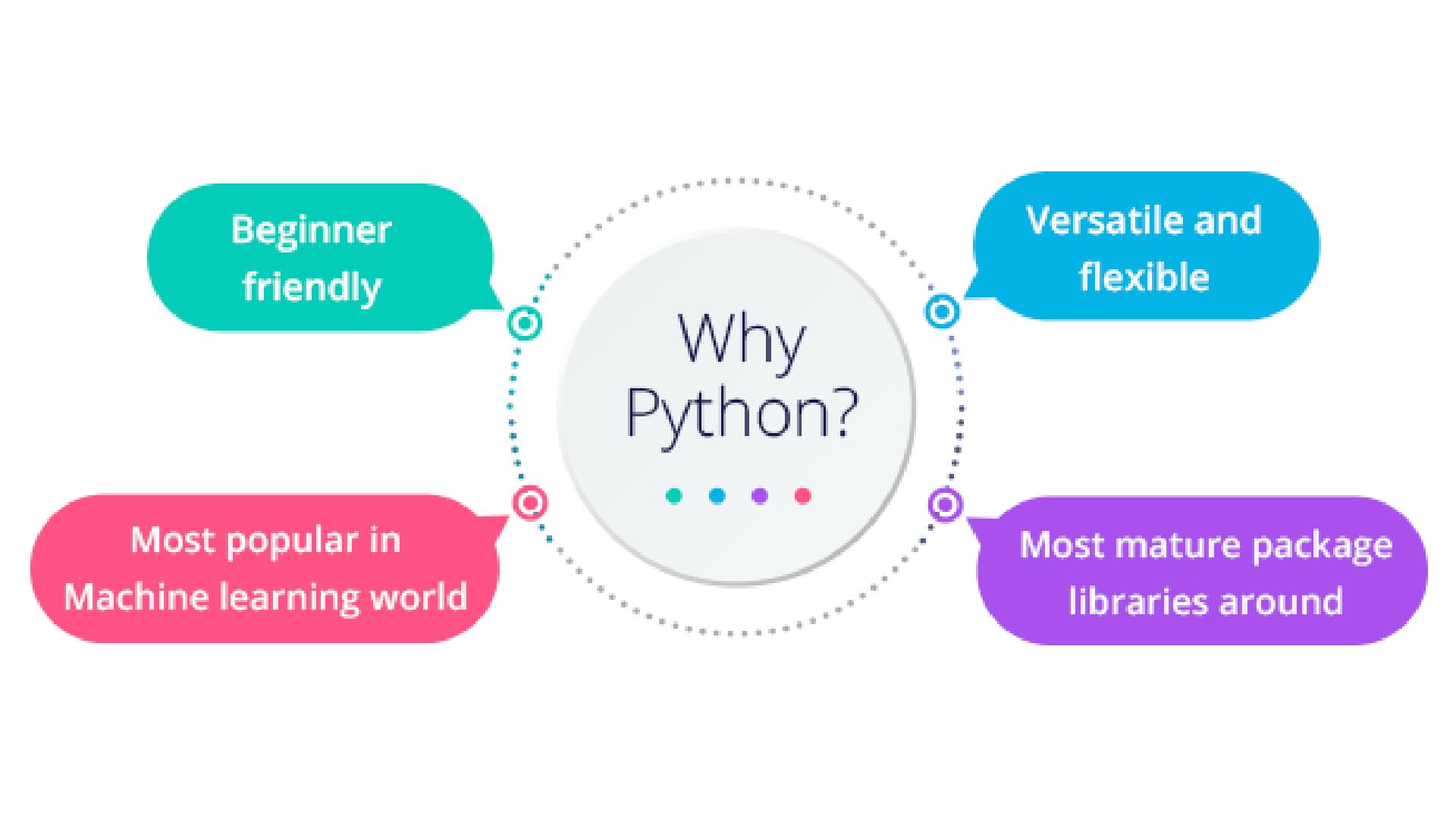


How do our usual languages fare? Worldwide jobs on indeed.com



What Can You Do With Python?





Why Python?

• • • •

Beginner friendly

Versatile and flexible

Most popular in Machine learning world

Most mature package libraries around

Aps created with python



Advantages of Python

Extensive Libraries

Extensible

Embeddable

Improved Productivity

IOT Opportunities

Simple and Easy

Readable

Object-Oriented

Free and Open-Source

Portable

Disadvantages of Python

Speed Limitations

Weak in Mobile Computing and Browsers

Design Restrictions

Underdeveloped Database Access Layers

Simple

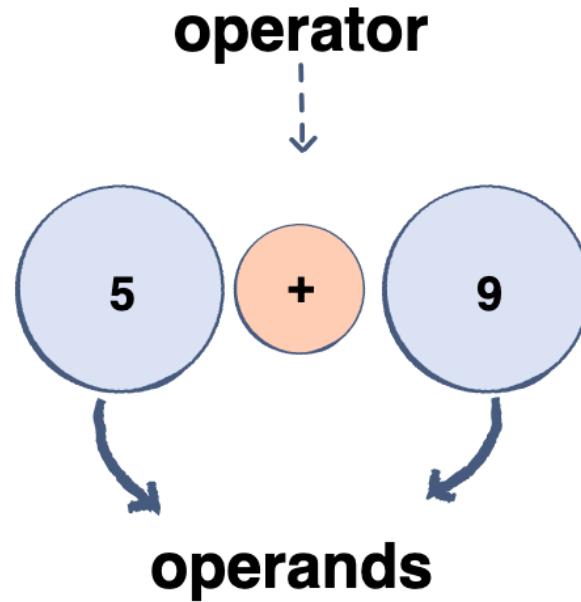
- variable is a reserved memory location to store values.
- there are some rules to write it
- it can only contain letters (a - z, A - Z), numbers or underscores (_).
- the first character cannot be a number.
- there are some reserved words that you cannot use as a variable name



Keywords in Python

False	class	<u>finally</u>	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	<u>elif</u>	if	or	yield
assert	else	import	pass	
break	except	in	raise	

Operators are special symbols you will use for operations between two values, or variables.



1 Arithmetic Operators

2 Assignment Operators

3 Comparison Operators

4 Logical Operators

5 Bitwise Operators

6 Identity Operators

7 Membership Operators

+

Add two operands or unary plus

```
>> 2 + 3  
5  
>> +2
```

-

Subtract two operands or unary subtract

```
>> 3 - 1  
2  
>> -2
```

*

Multiply two operands

```
>> 2 * 3  
6
```

/

Divide left operand with the right and result is in float

```
>> 6 / 3  
2.0
```

1 Arithmetic Operators

2 Assignment Operators

3 Comparison Operators

4 Logical Operators

5 Bitwise Operators

6 Identity Operators

7 Membership Operators

**

Left operand raised to the power of right

```
>> 2 ** 3  
8
```

%

Remainder of the division of left operand by the right

```
>> 5 % 2  
1
```

//

division that results into whole number adjusted to the left in the number line

```
>> 7 // 3  
2
```

1 Arithmetic Operators

2 Assignment Operators

3 Comparison Operators

4 Logical Operators

5 Bitwise Operators

6 Identity Operators

7 Membership Operators

`/=`

`x = x / <right operand>`

`>> x = 5`

```
>> x/=5  
>> print(x)  
1.0
```

`%=`

`x = x % <right operand>`

```
>> x% =5  
>> print(x)  
0
```

`//=`

`x = x // <right operand>`

```
>> x // =2  
>> print(x)  
2
```

`**=`

`x = x ** <right operand>`

```
>> x** =5  
>> print(x)  
3125
```

1 Arithmetic Operators

2 Assignment Operators

3 Comparison Operators

4 Logical Operators

5 Bitwise Operators

6 Identity Operators

7 Membership Operators

>

True if left operand is greater than the right

>> 2 > 3
False

<

True if left operand is less than the right

>> 2 < 3
True

==

True if left operand is equal to right

>> 2 == 2
True

!=

True if left operand is not equal to the right

>> x != 2
True

1 Arithmetic Operators

2 Assignment Operators

3 Comparison Operators

4 Logical Operators

5 Bitwise Operators

6 Identity Operators

7 Membership Operators

X = [1, 2, 3, 4, 5]

in

True if it finds elements in the specified sequence

>> 3 in x
True

Not in

True if it does not find elements in the specified sequence

>> 3 not in x
False

Comments

it used to explain Python code.

it used to make the code more readable.

- """
- This is a multi
- line comment

- """
- print("Hello, World!")

- #This is a comment
- #single line comment
- print("Hello, World!")

Escape Sequence in C

Escape sequence	Meaning	Elucidation
\n	New line	Used to shift the cursor control to the new line.
\t	Horizontal tab	Used to shift the cursor to a couple of spaces to the right in the same line.
\a	Audible bell	A beep is generated indicating the execution of the program to alert the user.
\r	Carriage Return	Used to position the cursor to the beginning of the current line.
\\	Backslash	Used to display the backslash character.



SCATTER PLOT



Two Variables

RELATIONSHIP

SCATTER PLOT BUBBLE SIZE



Three or more Variables

COMPARISON

What would you like to show?

COMPOSITION

DISTRIBUTION

Few Data Points

Single Variable

Many Data Points

Two Variables

BAR HISTOGRAM

LINE HISTOGRAM

SCATTER PLOT

Changing Over Time

Static

Few Periods

Many Periods

Only Relative Differences Matter

Relative and Absolute Differences Matter

Only Relative Differences Matter

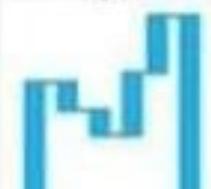
Relative and Absolute Differences Matter

Simple Share of Total

Accumulation or Subtraction to Total

Components of Components

Accumulation to total and absolute difference matters

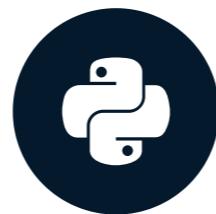


Introduction to Data Visualization with Matplotlib

INTRODUCTION TO DATA VISUALIZATION WITH MATPLOTLIB

Ariel Rokem

Data Scientist



Data visualization

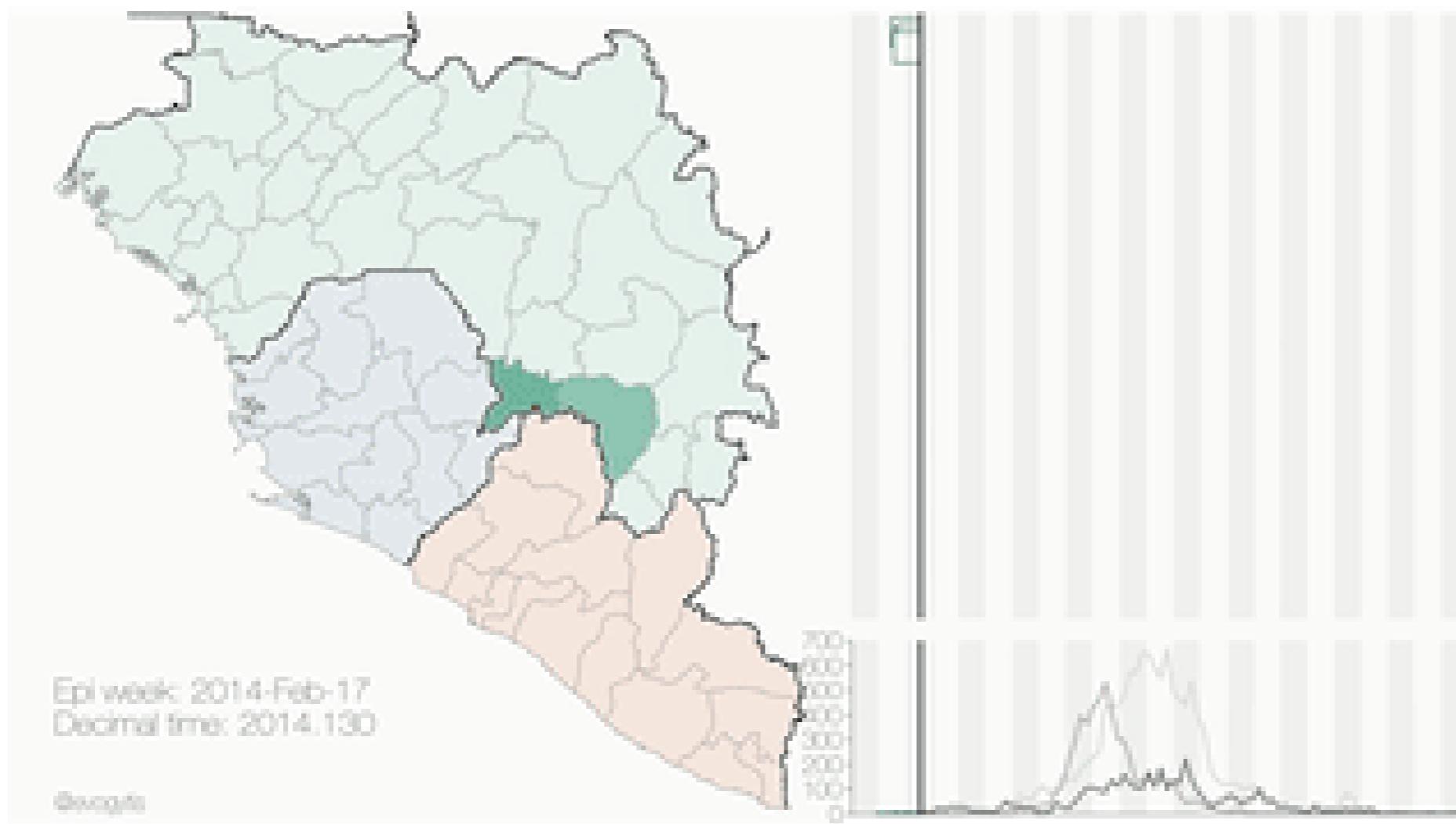
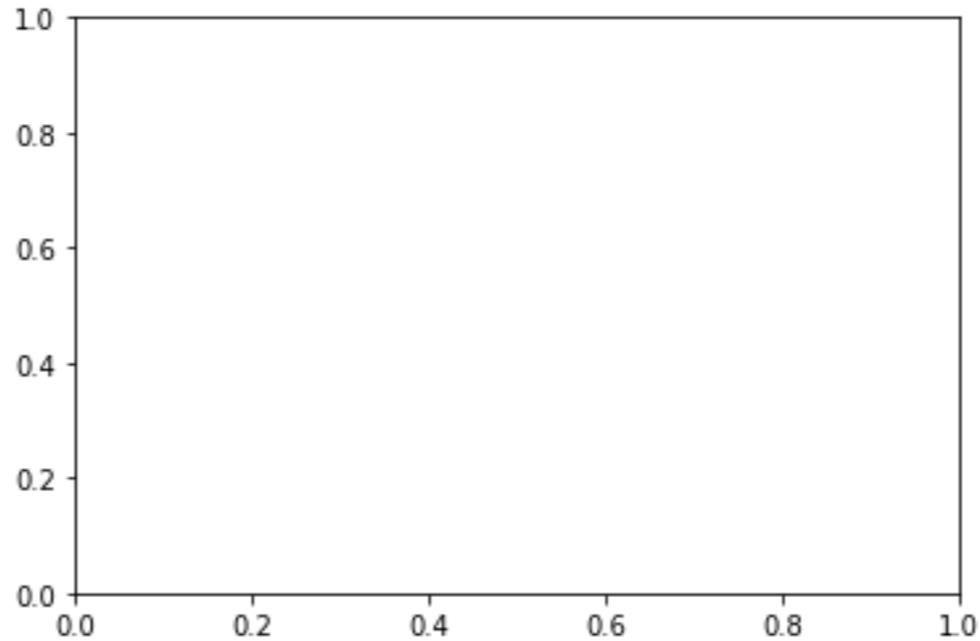


Image credit: [Gytis Dudas](#) and [Andrew Rambaut](#)

Introducing the pyplot interface

```
import matplotlib.pyplot as plt  
fig, ax = plt.subplots()  
plt.show()
```



Adding data to axes

```
seattle_weather["MONTH"]
```

```
DATE
1 Jan
2 Feb
3 Mar
4 Apr
5 May
6 Jun
7 Jul
8 Aug
9 Sep
10 Oct
11 Nov
12 Dec
```

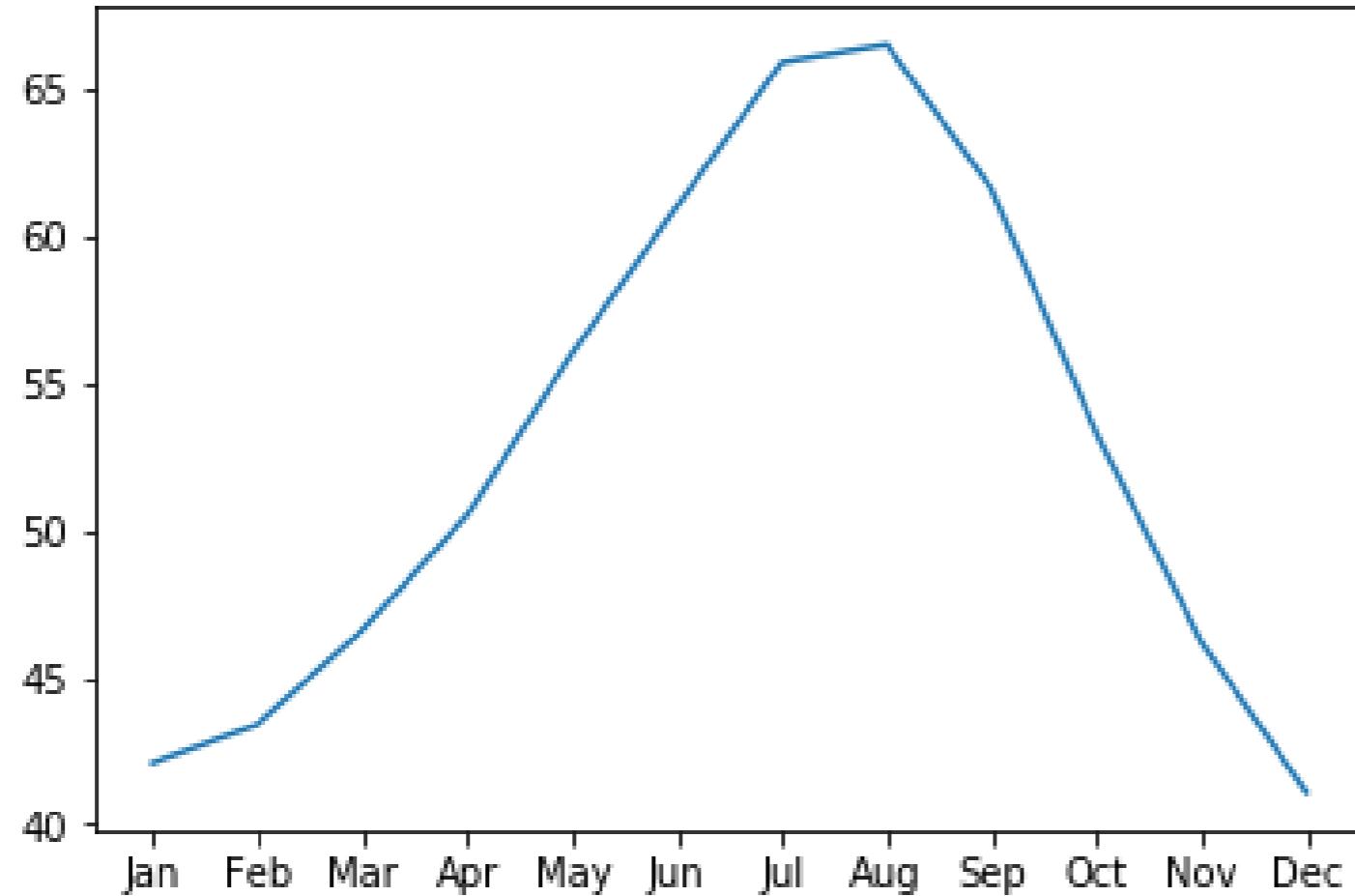
```
Name: MONTH, dtype: object
```

```
seattle_weather["MLY-TAVG-NORMAL"]
```

```
1 42.1
2 43.4
3 46.6
4 50.5
5 56.0
6 61.0
7 65.9
8 66.5
9 61.6
10 53.3
11 46.2
12 41.1
Name: MLY-TAVG-NORMAL, dtype: float64
```

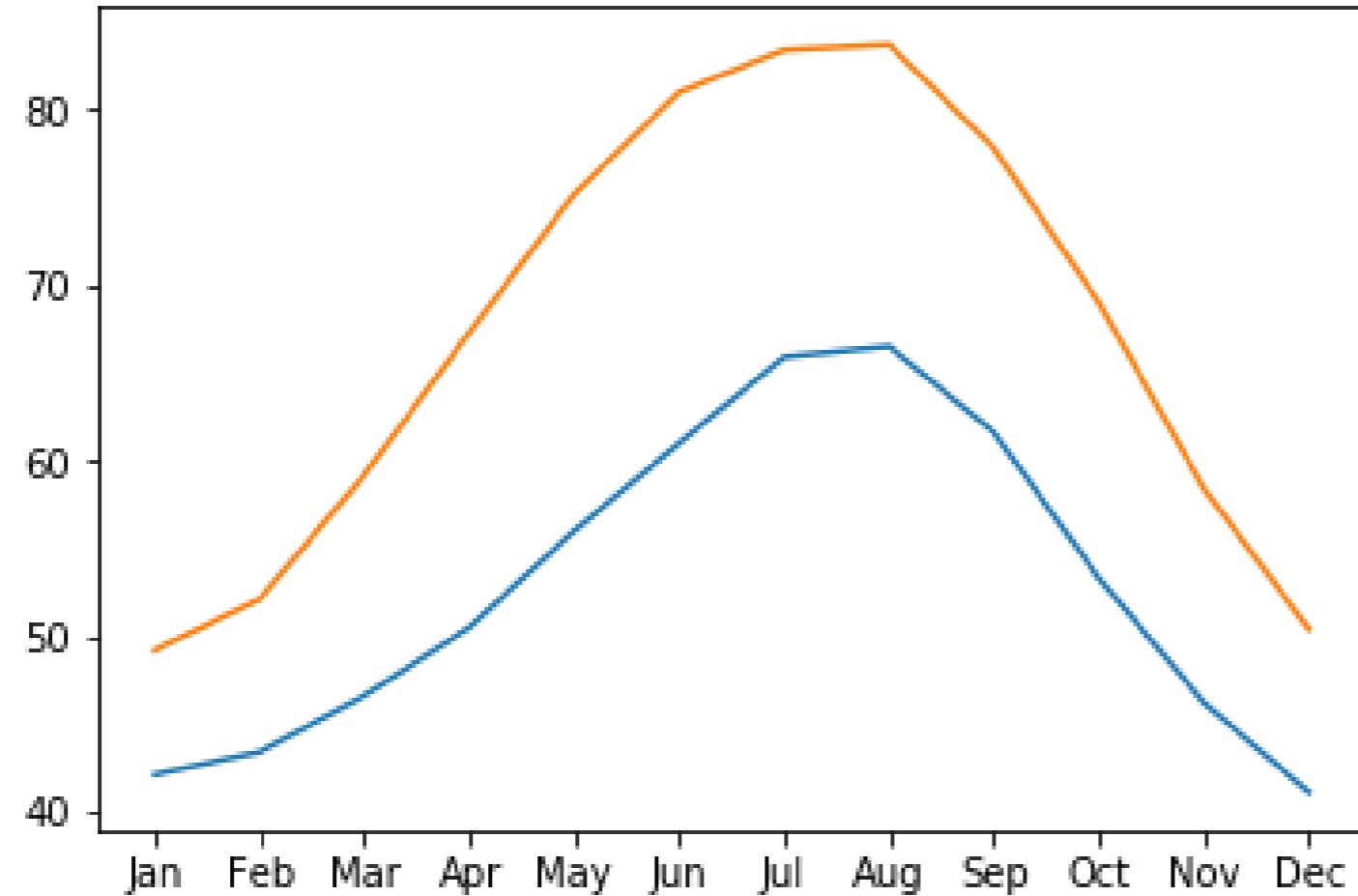
Adding data to axes

```
ax.plot(seattle_weather["MONTH"], seattle_weather["MLY-TAVG-NORMAL"])
plt.show()
```



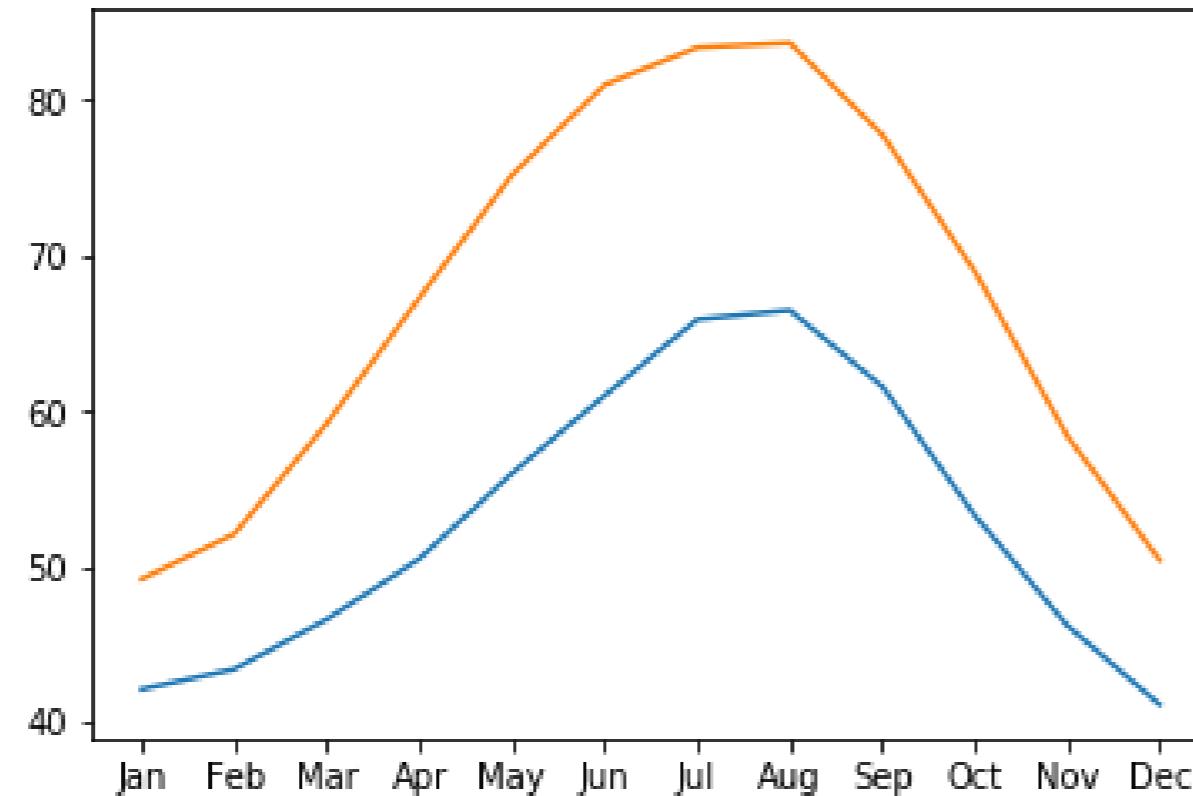
Adding more data

```
ax.plot(austin_weather["MONTH"], austin_weather["MLY-TAVG-NORMAL"])
plt.show()
```



Putting it all together

```
fig, ax = plt.subplots()  
ax.plot(seattle_weather["MONTH"], seattle_weather["MLY-TAVG-NORMAL"]  
ax.plot(austin_weather["MONTH"], austin_weather["MLY-TAVG-NORMAL"])  
plt.show()
```

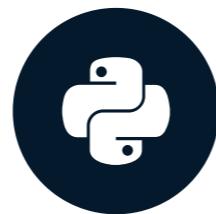


Practice making a figure!

INTRODUCTION TO DATA VISUALIZATION WITH MATPLOTLIB

Customizing your plots

INTRODUCTION TO DATA VISUALIZATION WITH MATPLOTLIB

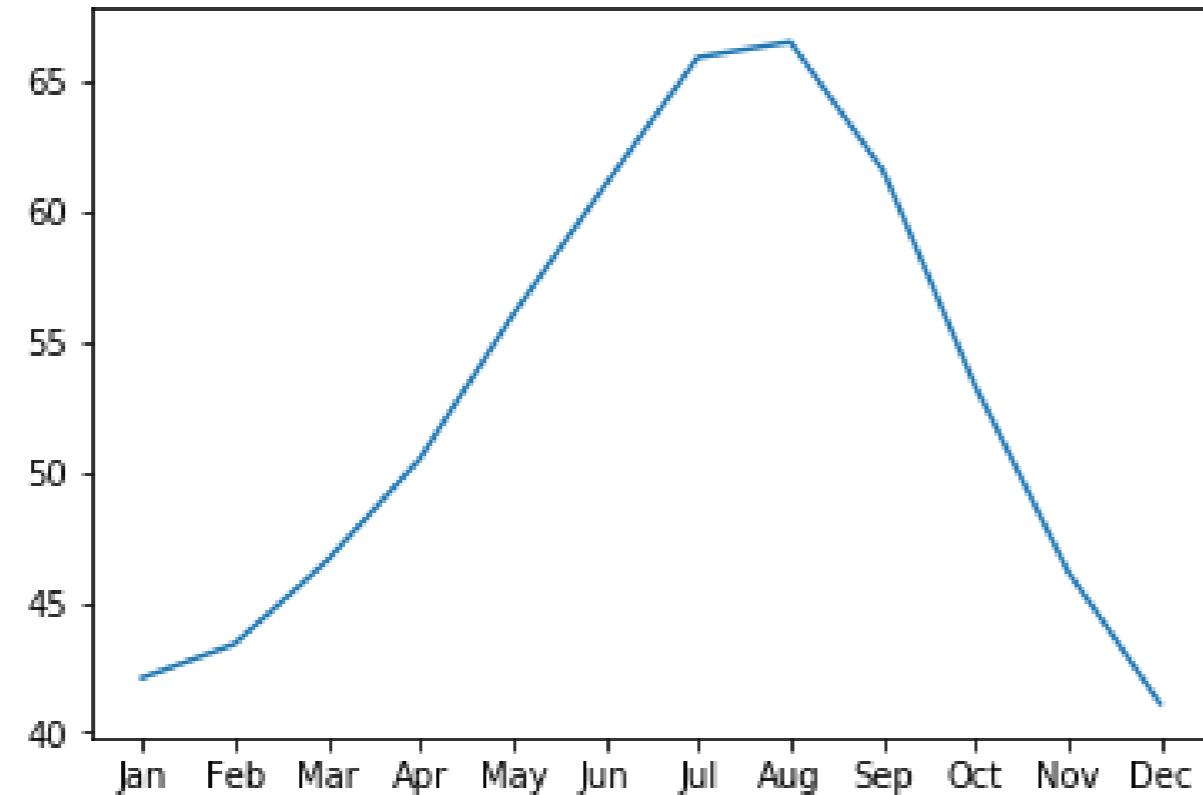


Ariel Rokem

Data Scientist

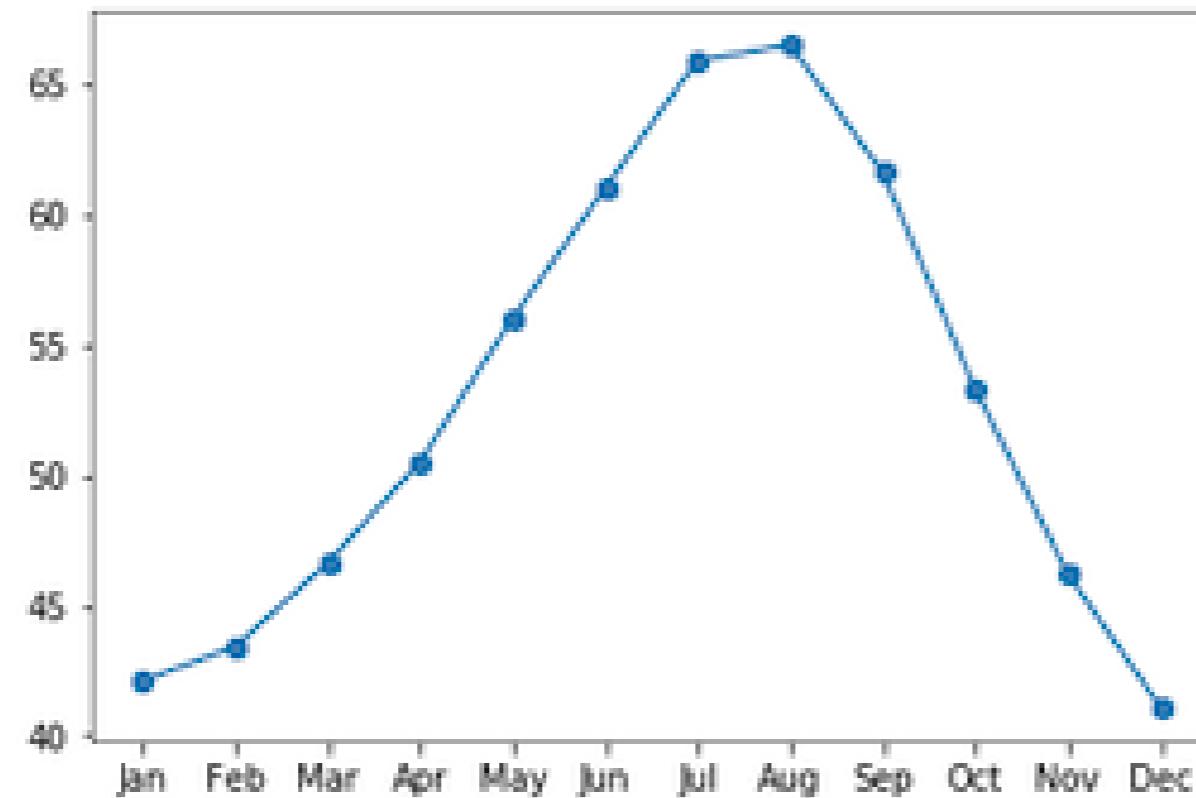
Customizing data appearance

```
ax.plot(seattle_weather["MONTH"],  
        seattle_weather["MLY-PRCP-NORMAL"])  
plt.show()
```



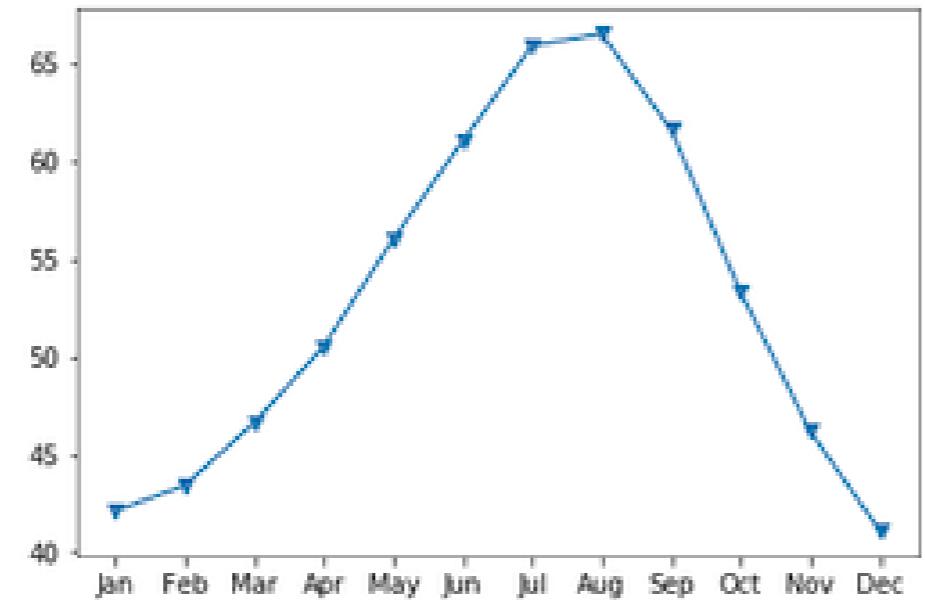
Adding markers

```
ax.plot(seattle_weather["MONTH"],  
        seattle_weather["MLY-PRCP-NORMAL"],  
        marker="o")  
plt.show()
```



Choosing markers

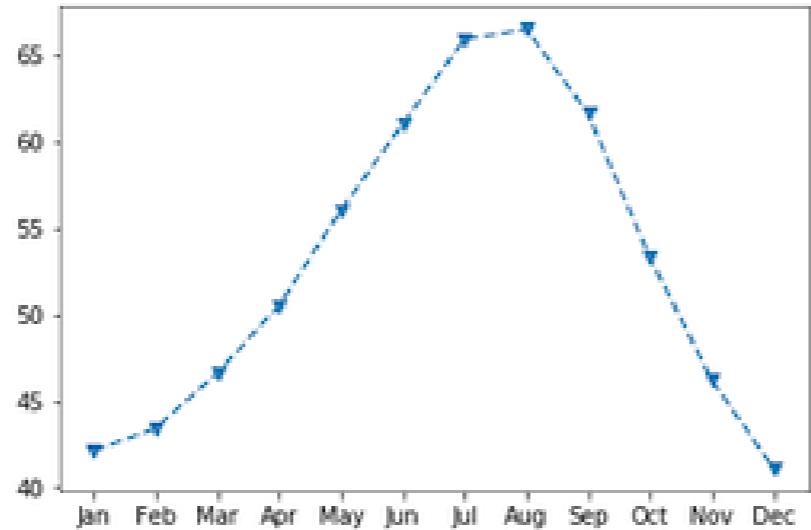
```
ax.plot(seattle_weather["MONTH"],  
        seattle_weather["MLY-PRCP-NORMAL"],  
        marker="v")  
  
plt.show()
```



https://matplotlib.org/api/markers_api.html

Setting the linestyle

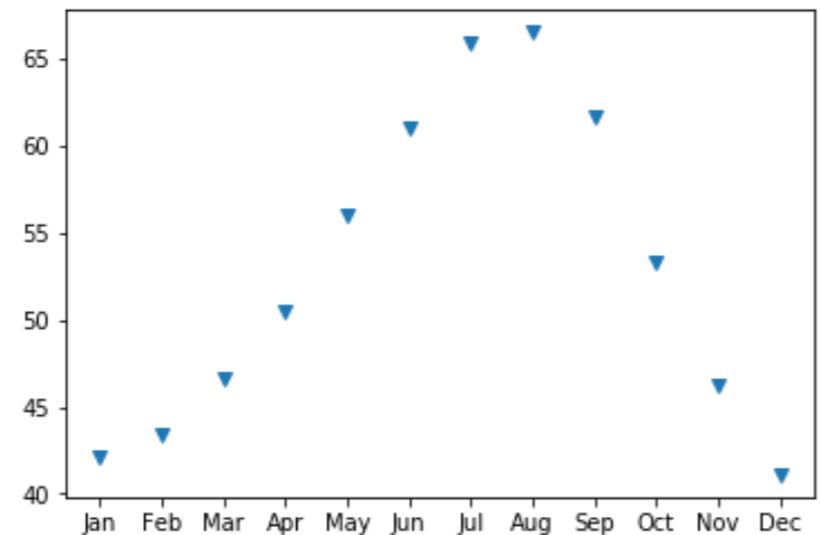
```
fig, ax = plt.subplots()  
ax.plot(seattle_weather["MONTH"],  
        seattle_weather["MLY-TAVG-NORMAL"],  
        marker="v", linestyle="--")  
plt.show()
```



https://matplotlib.org/gallery/lines_bars_and_markers/line_styles_reference.html

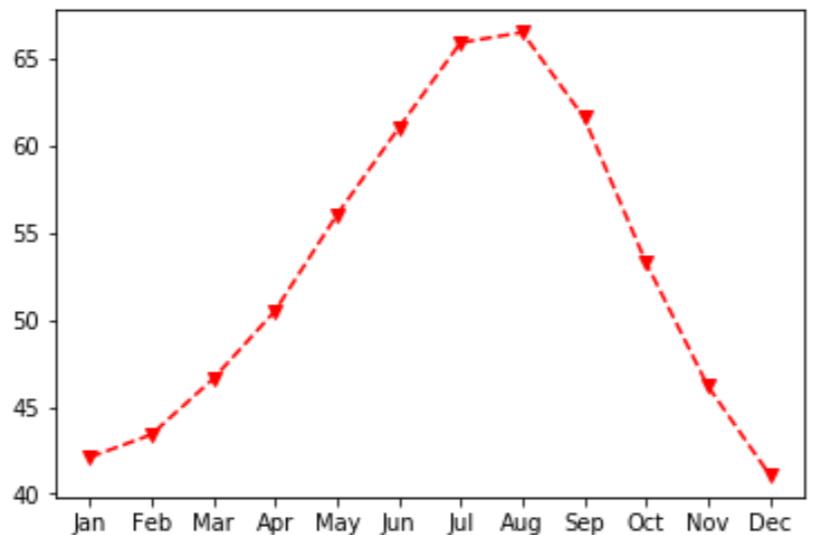
Eliminating lines with linestyle

```
fig, ax = plt.subplots()  
ax.plot(seattle_weather["MONTH"],  
        seattle_weather["MLY-TAVG-NORMAL"],  
        marker="v", linestyle="None")  
plt.show()
```



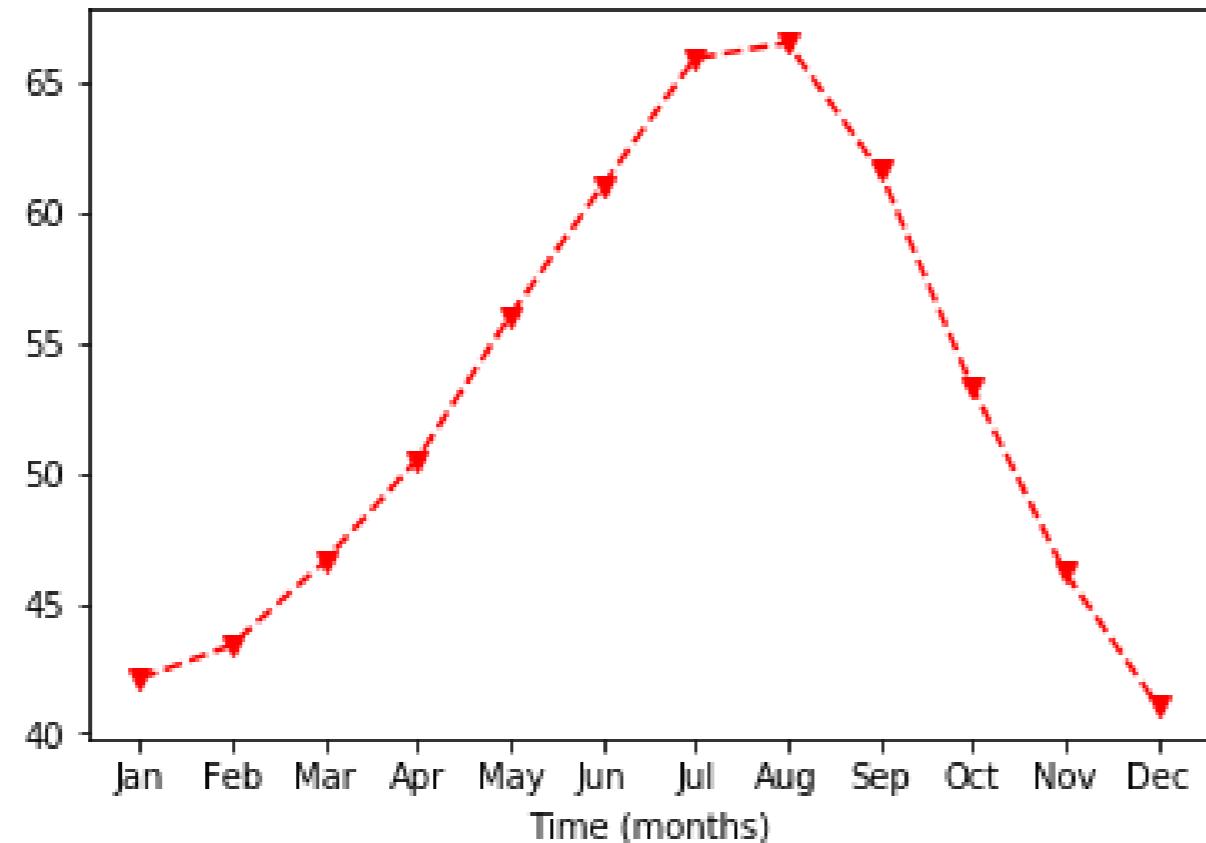
Choosing color

```
fig, ax = plt.subplots()  
ax.plot(seattle_weather["MONTH"],  
        seattle_weather["MLY-TAVG-NORMAL"],  
        marker="v", linestyle="--", color="r")  
plt.show()
```



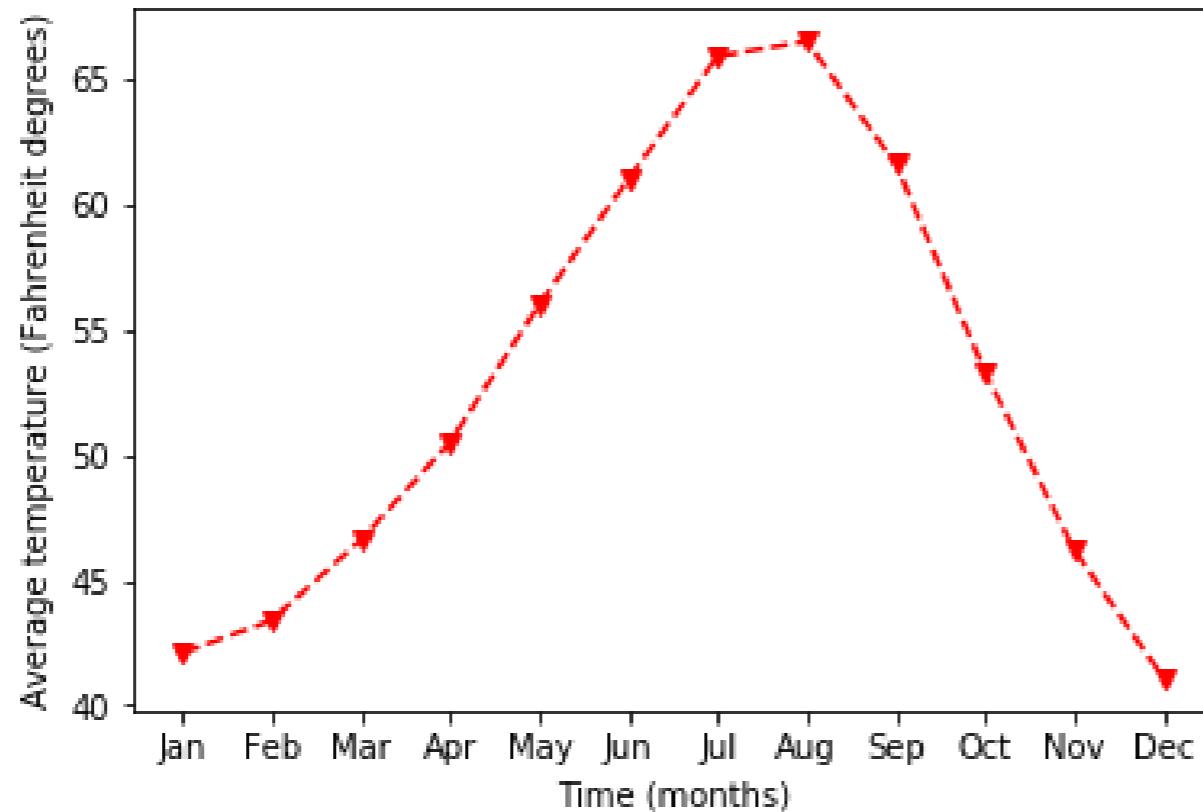
Customizing the axes labels

```
ax.set_xlabel("Time (months)")  
plt.show()
```



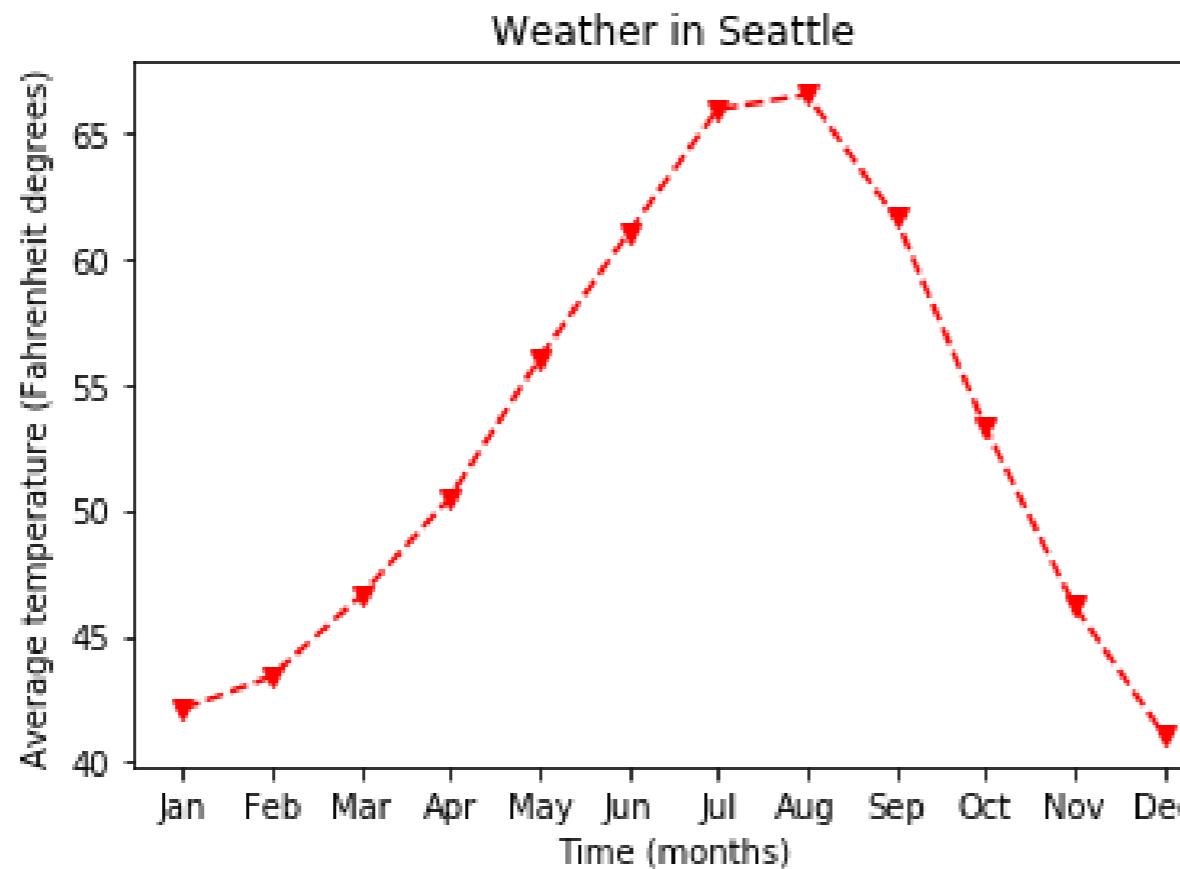
Setting the y axis label

```
ax.set_xlabel("Time (months)")  
ax.set_ylabel("Average temperature (Fahrenheit degrees)")  
plt.show()
```



Adding a title

```
ax.set_title("Weather in Seattle")  
plt.show()
```

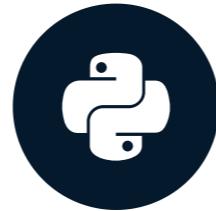


Practice customizing your plots!

INTRODUCTION TO DATA VISUALIZATION WITH MATPLOTLIB

Small multiples

INTRODUCTION TO DATA VISUALIZATION WITH MATPLOTLIB

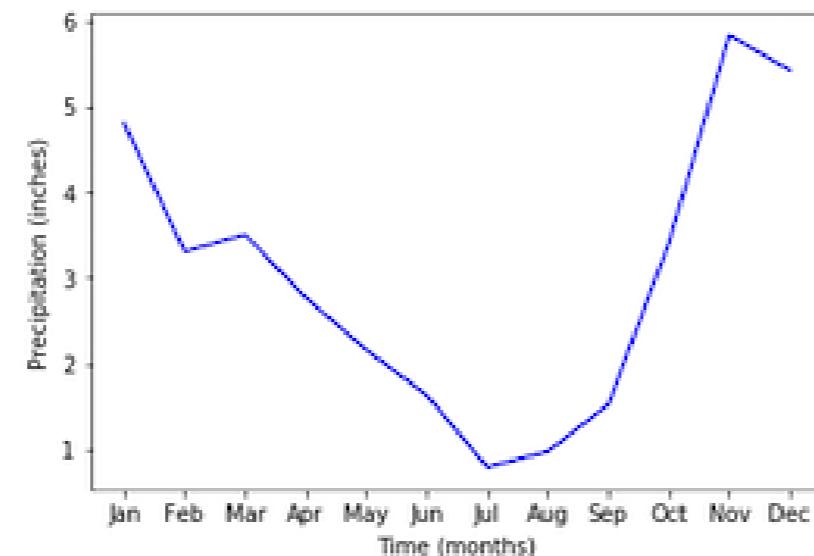


Ariel Rokem

Data Scientist

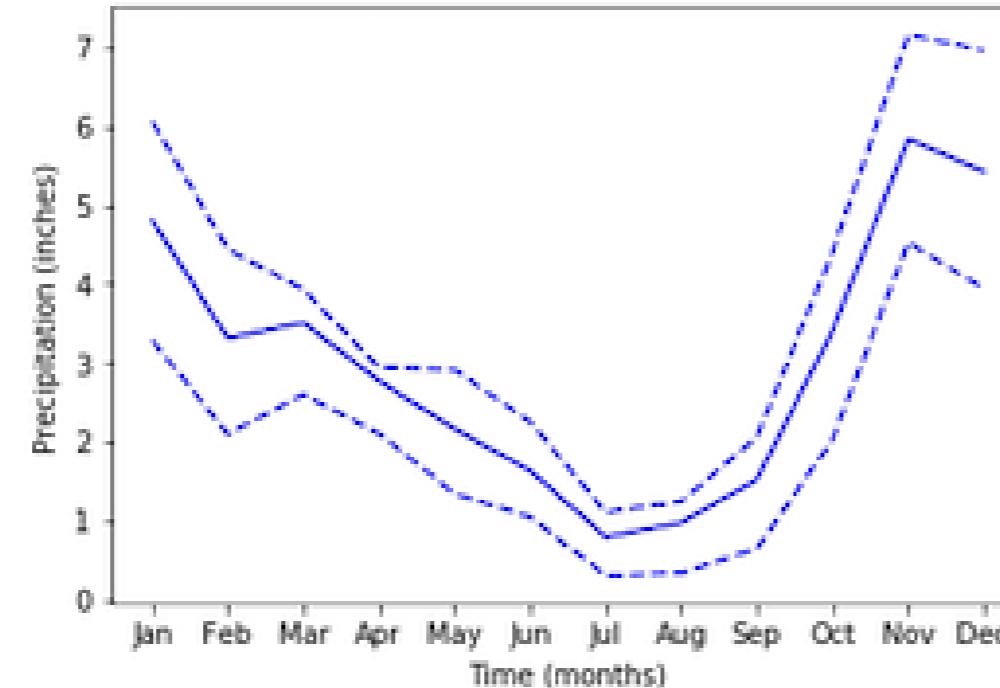
Adding data

```
ax.plot(seattle_weather["MONTH"],  
        seattle_weather["MLY-PRCP-NORMAL"],  
        color='b')  
  
ax.set_xlabel("Time (months)")  
ax.set_ylabel("Precipitation (inches)")  
plt.show()
```



Adding more data

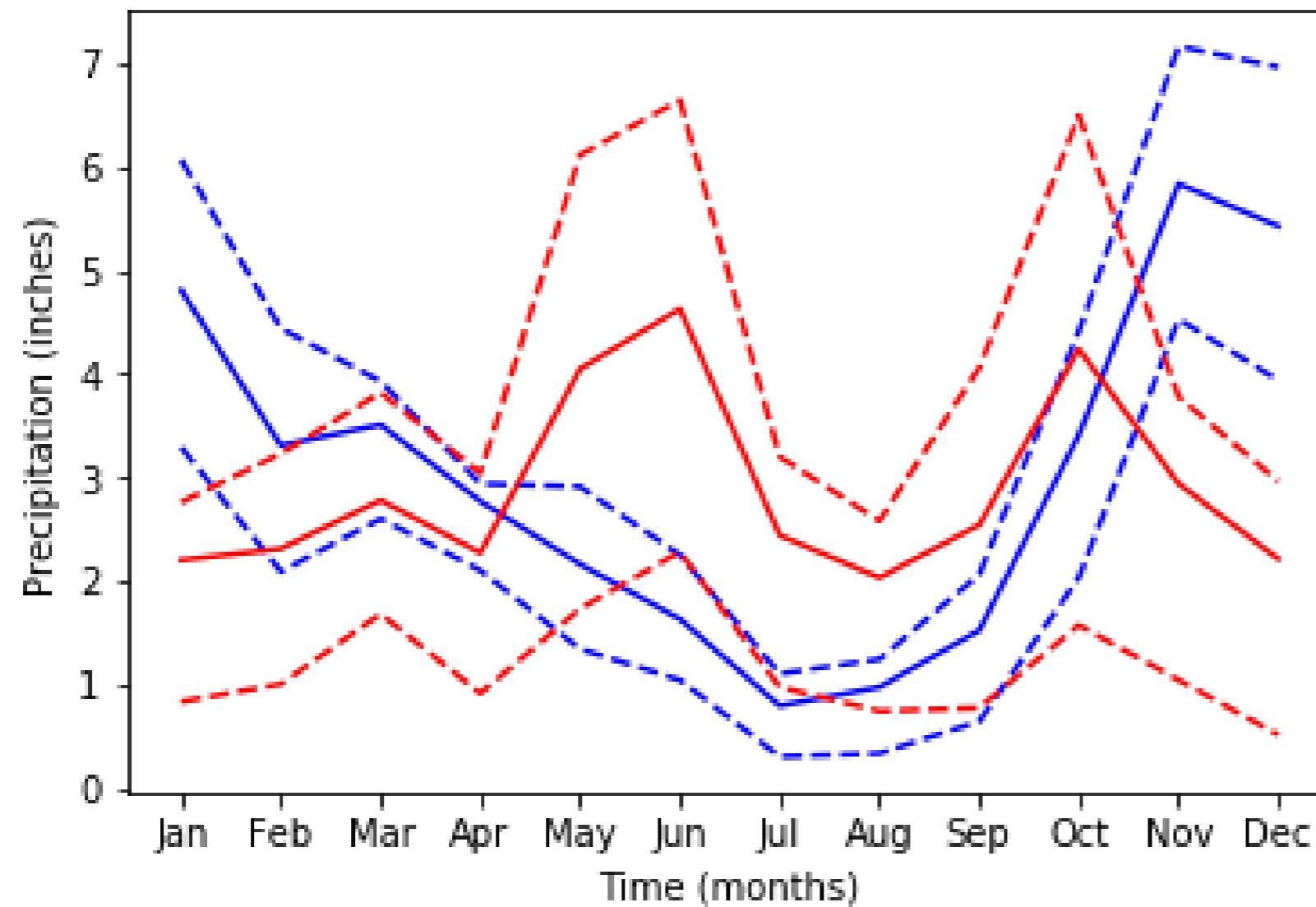
```
ax.plot(seattle_weather["MONTH"], seattle_weather["MLY-PRCP-25PCTL"],
        linestyle='--', color='b')
ax.plot(seattle_weather["MONTH"], seattle_weather["MLY-PRCP-75PCTL"],
        linestyle='--', color=color)
plt.show()
```



And more data

```
ax.plot(austin_weather["MONTH"], austin_weather["MLY-PRCP-NORMAL"],
        color='r')
ax.plot(austin_weather["MONTH"], austin_weather["MLY-PRCP-25PCTL"],
        linestyle='--', color='r')
ax.plot(austin_weather["MONTH"], austin_weather["MLY-PRCP-75PCTL"],
        linestyle='--', color='r')
plt.show()
```

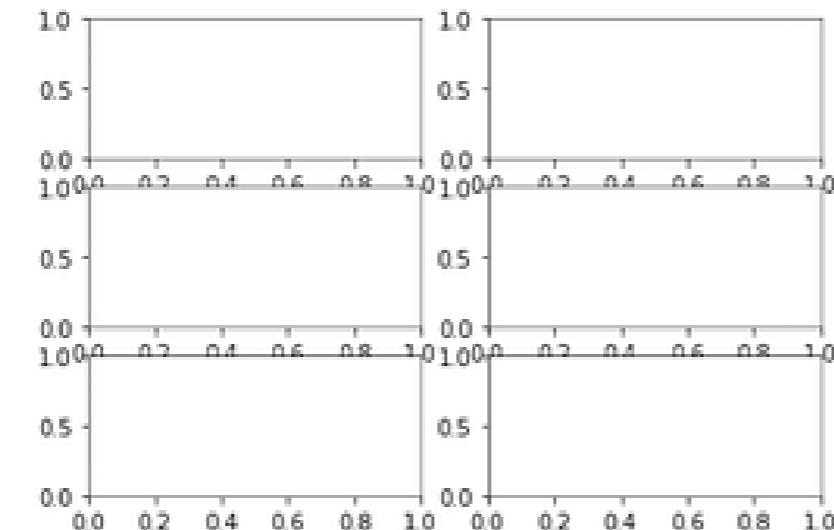
Too much data!



Small multiples with plt.subplots

```
fig, ax = plt.subplots()
```

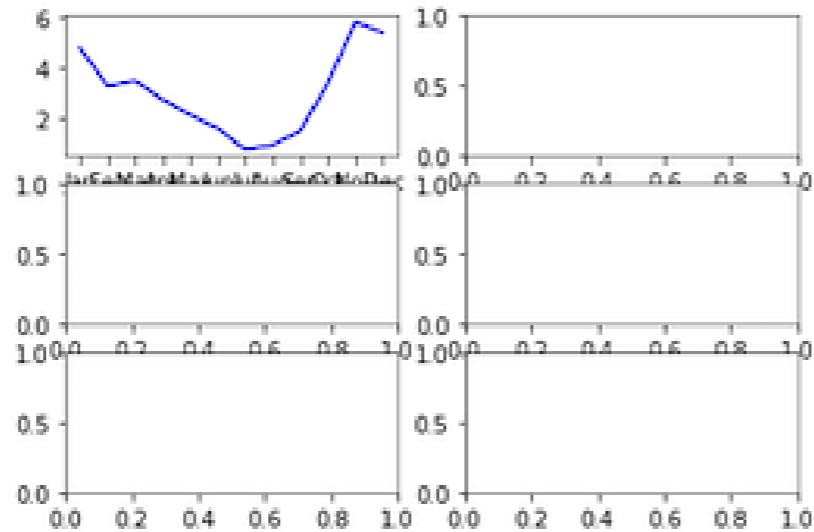
```
fig, ax = plt.subplots(3, 2)  
plt.show()
```



Adding data to subplots

```
ax.shape  
(3, 2)
```

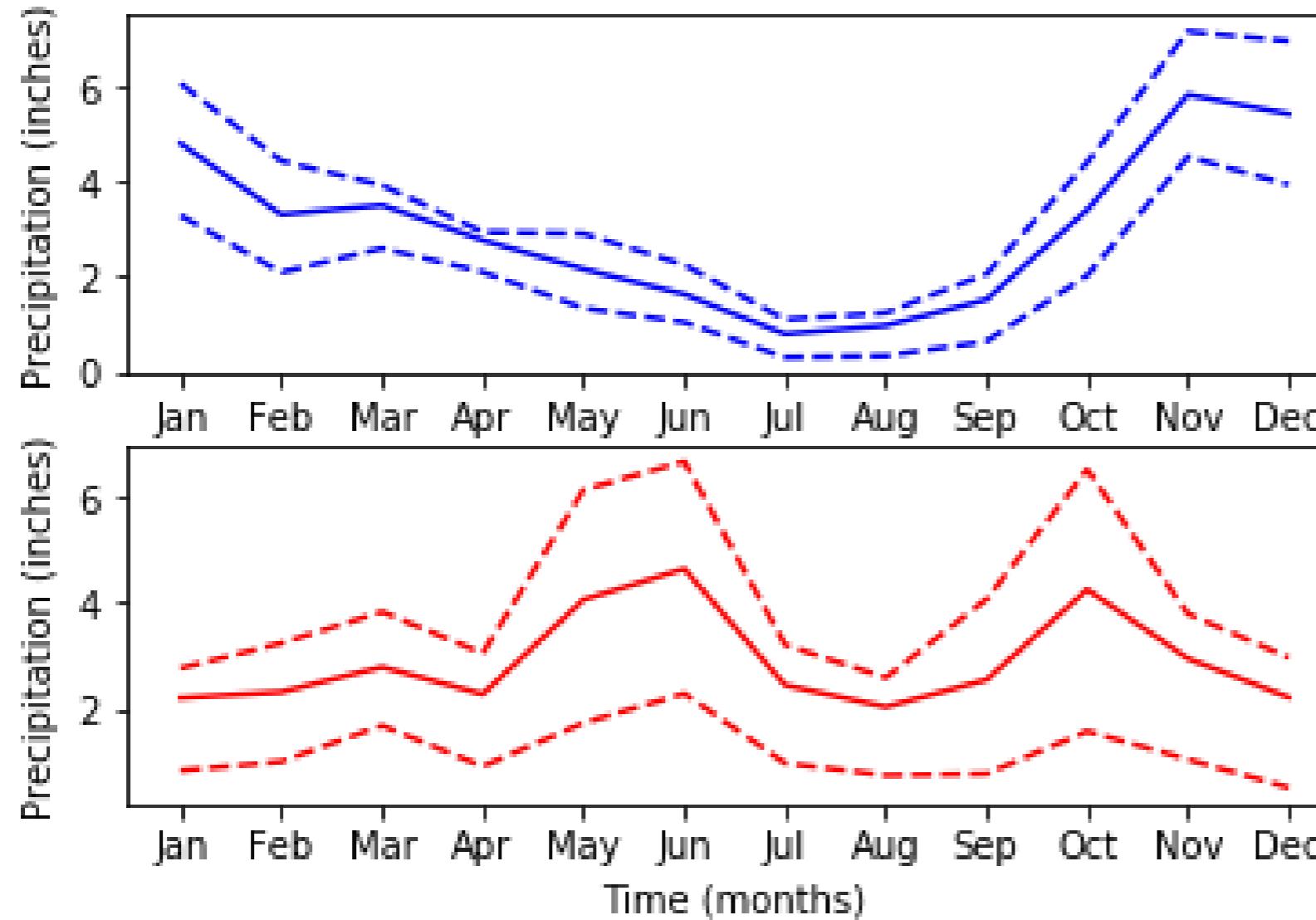
```
ax[0, 0].plot(seattle_weather["MONTH"],  
               seattle_weather["MLY-PRCP-NORMAL"],  
               color='b')  
  
plt.show()
```



Subplots with data

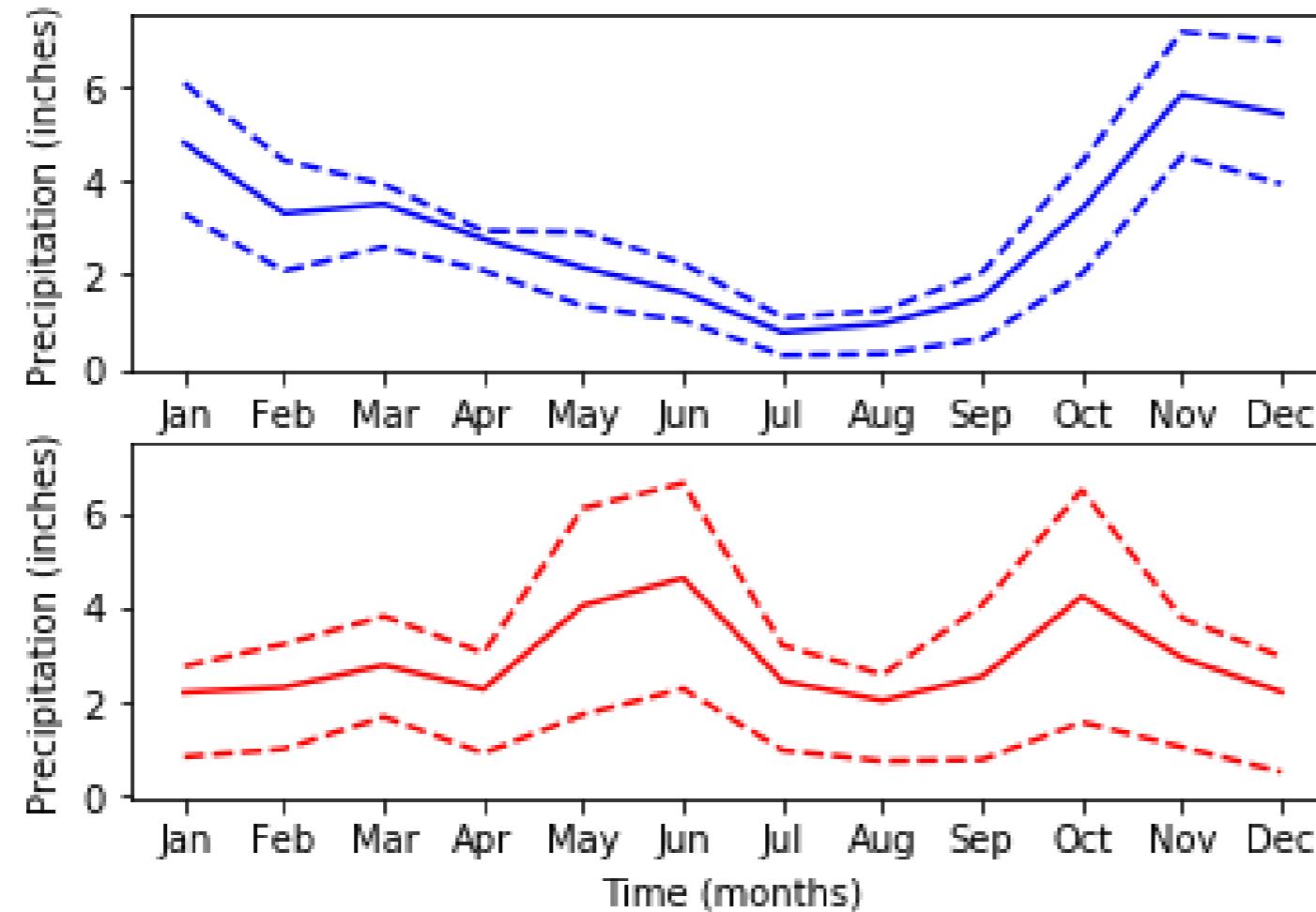
```
fig, ax = plt.subplots(2, 1)
ax[0].plot(seattle_weather["MONTH"], seattle_weather["MLY-PRCP-NORMAL"],
           color='b')
ax[0].plot(seattle_weather["MONTH"], seattle_weather["MLY-PRCP-25PCTL"],
           linestyle='--', color='b')
ax[0].plot(seattle_weather["MONTH"], seattle_weather["MLY-PRCP-75PCTL"],
           linestyle='--', color='b')
ax[1].plot(austin_weather["MONTH"], austin_weather["MLY-PRCP-NORMAL"],
           color='r')
ax[1].plot(austin_weather["MONTH"], austin_weather["MLY-PRCP-25PCTL"],
           linestyle='--', color='r')
ax[1].plot(austin_weather["MONTH"], austin_weather["MLY-PRCP-75PCTL"],
           linestyle='--', color='r')
ax[0].set_ylabel("Precipitation (inches)")
ax[1].set_ylabel("Precipitation (inches)")
ax[1].set_xlabel("Time (months)")
plt.show()
```

Subplots with data



Sharing the y-axis range

```
fig, ax = plt.subplots(2, 1, sharey=True)
```

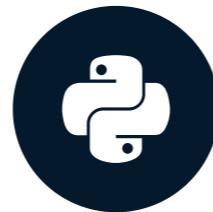


Practice making subplots!

INTRODUCTION TO DATA VISUALIZATION WITH MATPLOTLIB

Plotting time-series data

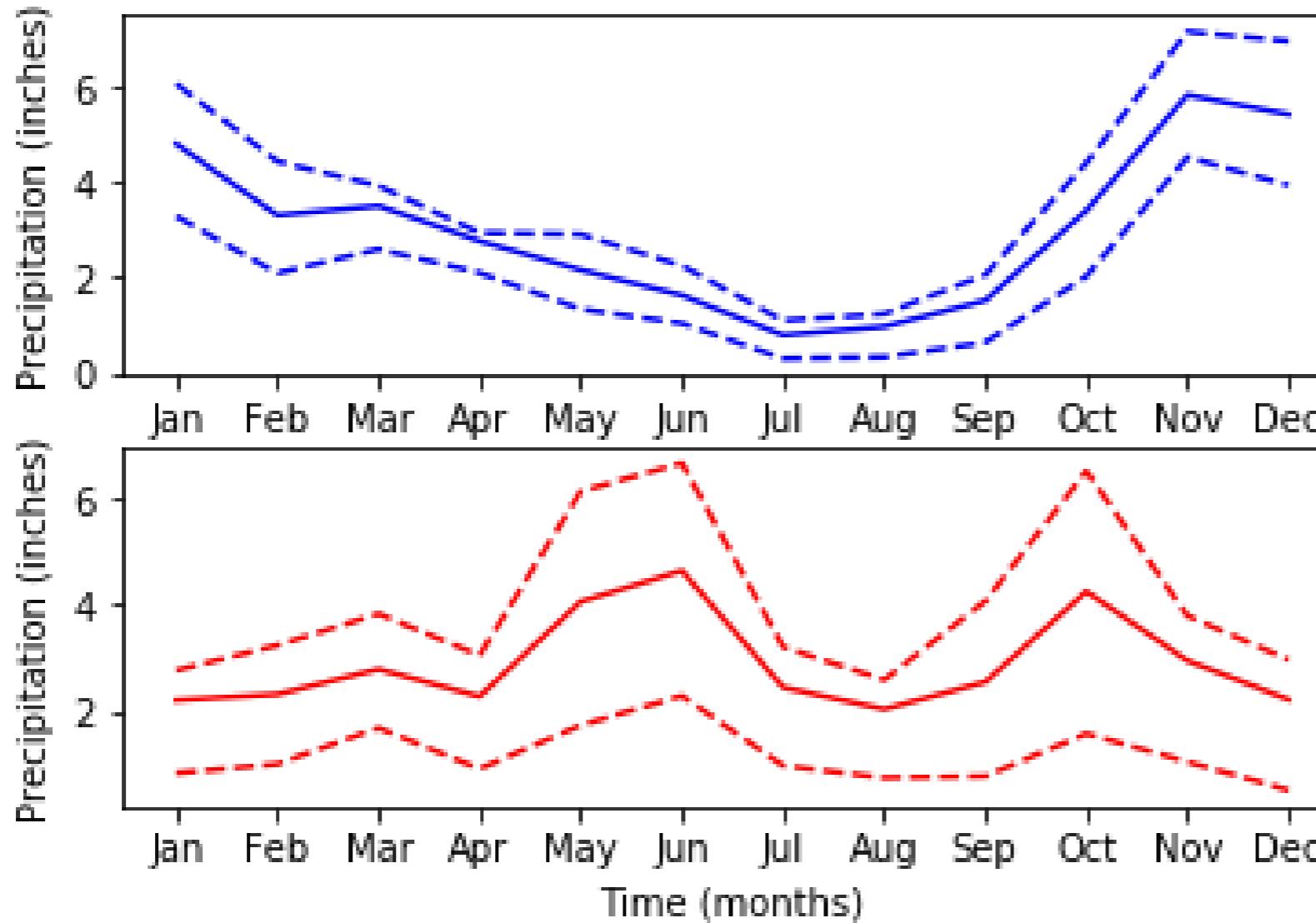
INTRODUCTION TO DATA VISUALIZATION WITH MATPLOTLIB



Ariel Rokem

Data Scientist

Time-series data



Climate change time-series

```
date,co2,relative_temp  
1958-03-06,315.71,0.1  
1958-04-06,317.45,0.01  
1958-05-06,317.5,0.08  
1958-06-06,-99.99,-0.05  
1958-07-06,315.86,0.06  
1958-08-06,314.93,-0.06  
...  
2016-08-06,402.27,0.98  
2016-09-06,401.05,0.87  
2016-10-06,401.59,0.89  
2016-11-06,403.55,0.93  
2016-12-06,404.45,0.81
```

DateTimelIndex

climate_change.index

```
DatetimeIndex(['1958-03-06', '1958-04-06', '1958-05-06', '1958-06-06',
                 '1958-07-06', '1958-08-06', '1958-09-06', '1958-10-06',
                 '1958-11-06', '1958-12-06',
                 ...
                 '2016-03-06', '2016-04-06', '2016-05-06', '2016-06-06',
                 '2016-07-06', '2016-08-06', '2016-09-06', '2016-10-06',
                 '2016-11-06', '2016-12-06'],
                dtype='datetime64[ns]', name='date', length=706, freq=None)
```

Time-series data

```
climate_change['relative_temp']
```

```
0      0.10  
1      0.01  
2      0.08  
3     -0.05  
4      0.06  
5     -0.06  
6     -0.03  
7      0.04  
...  
701    0.98  
702    0.87  
703    0.89  
704    0.93  
705    0.81  
Name:co2, Length: 706, dtype: float64
```

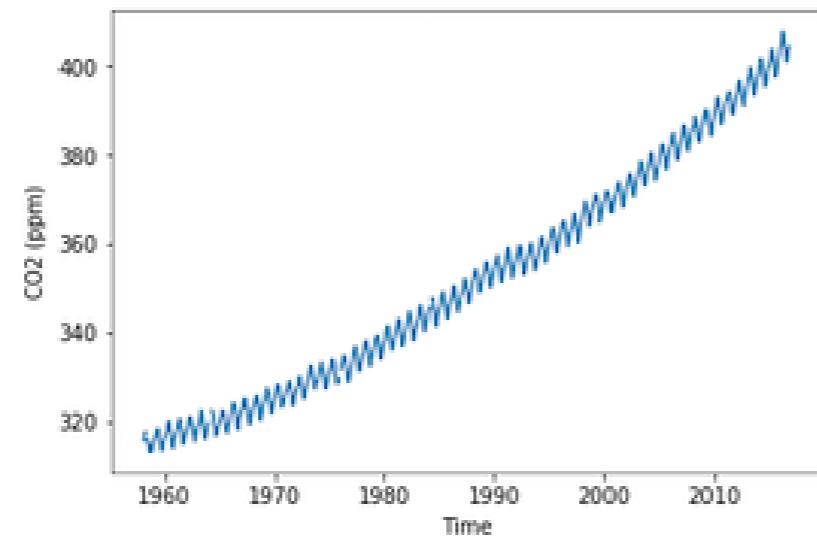
```
climate_change['co2']
```

```
0      315.71  
1      317.45  
2      317.50  
3        NaN  
4      315.86  
5      314.93  
6      313.20  
7        NaN  
...  
701    402.27  
702    401.05  
703    401.59  
704    403.55  
705    404.45  
Name:co2, Length: 706, dtype: float64
```

Plotting time-series data

```
import matplotlib.pyplot as plt  
fig, ax = plt.subplots()
```

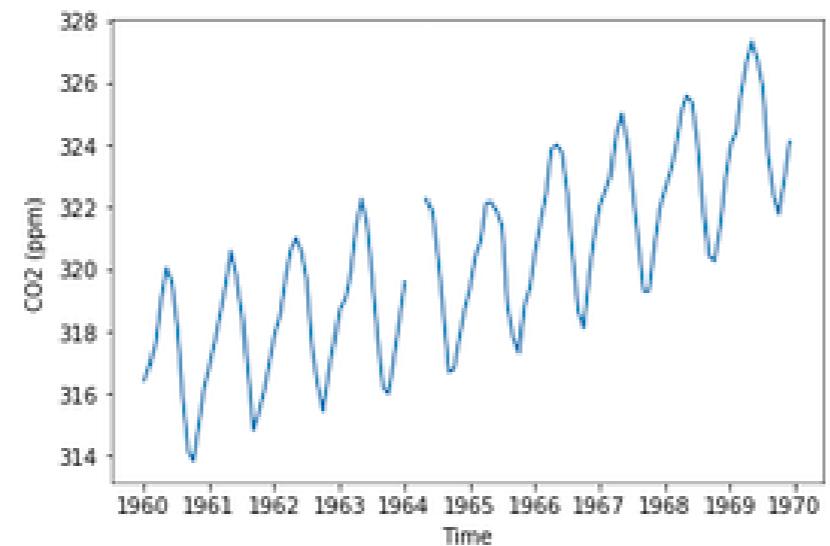
```
ax.plot(climate_change.index, climate_change['co2'])  
ax.set_xlabel('Time')  
ax.set_ylabel('CO2 (ppm)')  
plt.show()
```



Zooming in on a decade

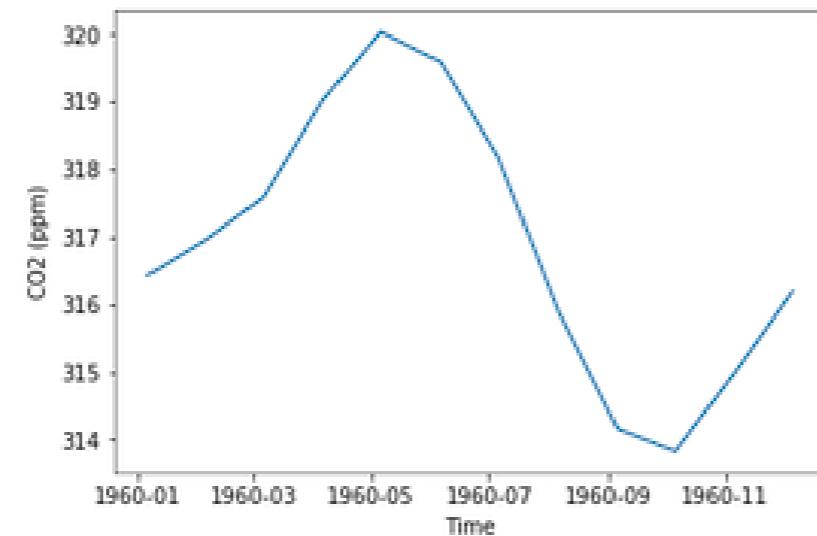
```
sixties = climate_change["1960-01-01":"1969-12-31"]
```

```
fig, ax = plt.subplots()  
ax.plot(sixties.index, sixties['co2'])  
ax.set_xlabel('Time')  
ax.set_ylabel('CO2 (ppm)')  
plt.show()
```



Zooming in on one year

```
sixty_nine = climate_change["1969-01-01":"1969-12-31"]
fig, ax = plt.subplots()
ax.plot(sixty_nine.index, sixty_nine['co2'])
ax.set_xlabel('Time')
ax.set_ylabel('CO2 (ppm)')
plt.show()
```



Let's practice time-series plotting!

INTRODUCTION TO DATA VISUALIZATION WITH MATPLOTLIB

Plotting time-series with different variables

INTRODUCTION TO DATA VISUALIZATION WITH MATPLOTLIB

Ariel Rokem

Data Scientist



Plotting two time-series together

```
import pandas as pd  
climate_change = pd.read_csv('climate_change.csv',  
                             parse_dates=["date"],  
                             index_col="date")
```

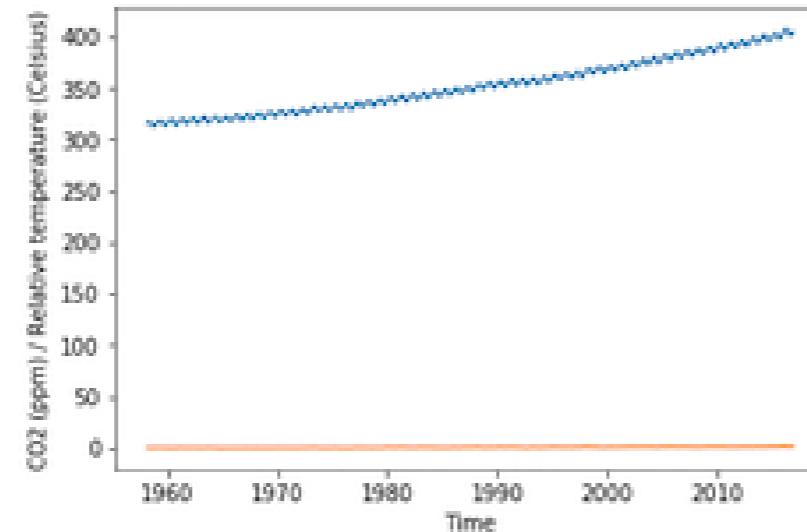
```
climate_change
```

```
      co2  relative_temp  
date  
1958-03-06    315.71        0.10  
1958-04-06    317.45        0.01  
1958-07-06    315.86        0.06  
...            ...          ...  
2016-11-06    403.55        0.93  
2016-12-06    404.45        0.81
```

[706 rows x 2 columns]

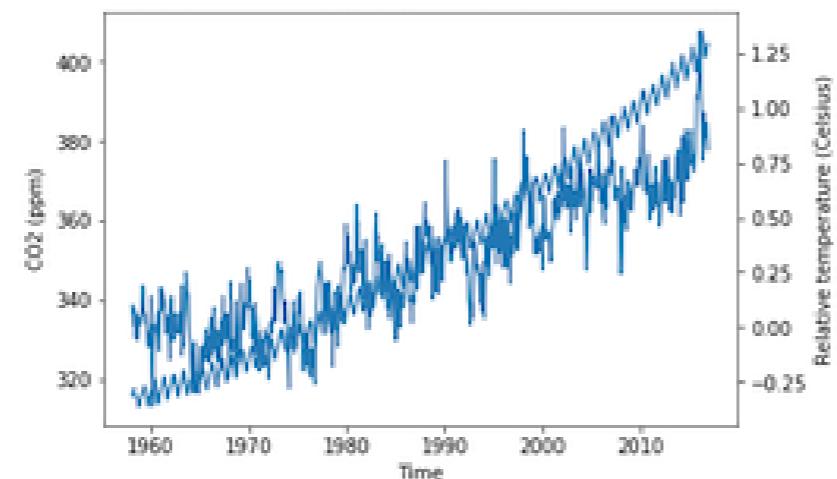
Plotting two time-series together

```
import matplotlib.pyplot as plt  
fig, ax = plt.subplots()  
ax.plot(climate_change.index, climate_change["co2"])  
ax.plot(climate_change.index, climate_change["relative_temp"])  
ax.set_xlabel('Time')  
ax.set_ylabel('CO2 (ppm) / Relative temperature')  
plt.show()
```



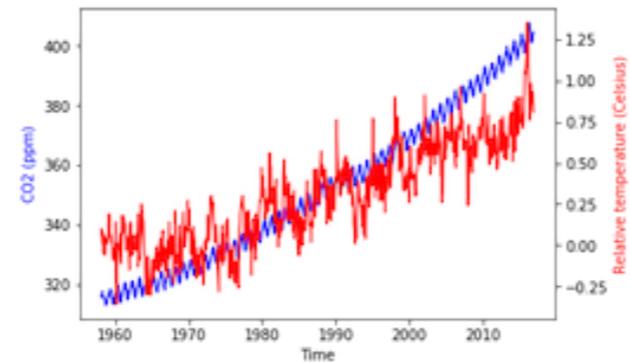
Using twin axes

```
fig, ax = plt.subplots()  
ax.plot(climate_change.index, climate_change["co2"])  
ax.set_xlabel('Time')  
ax.set_ylabel('CO2 (ppm)')  
ax2 = ax.twinx()  
ax2.plot(climate_change.index, climate_change["relative_temp"])  
ax2.set_ylabel('Relative temperature (Celsius)')  
plt.show()
```



Separating variables by color

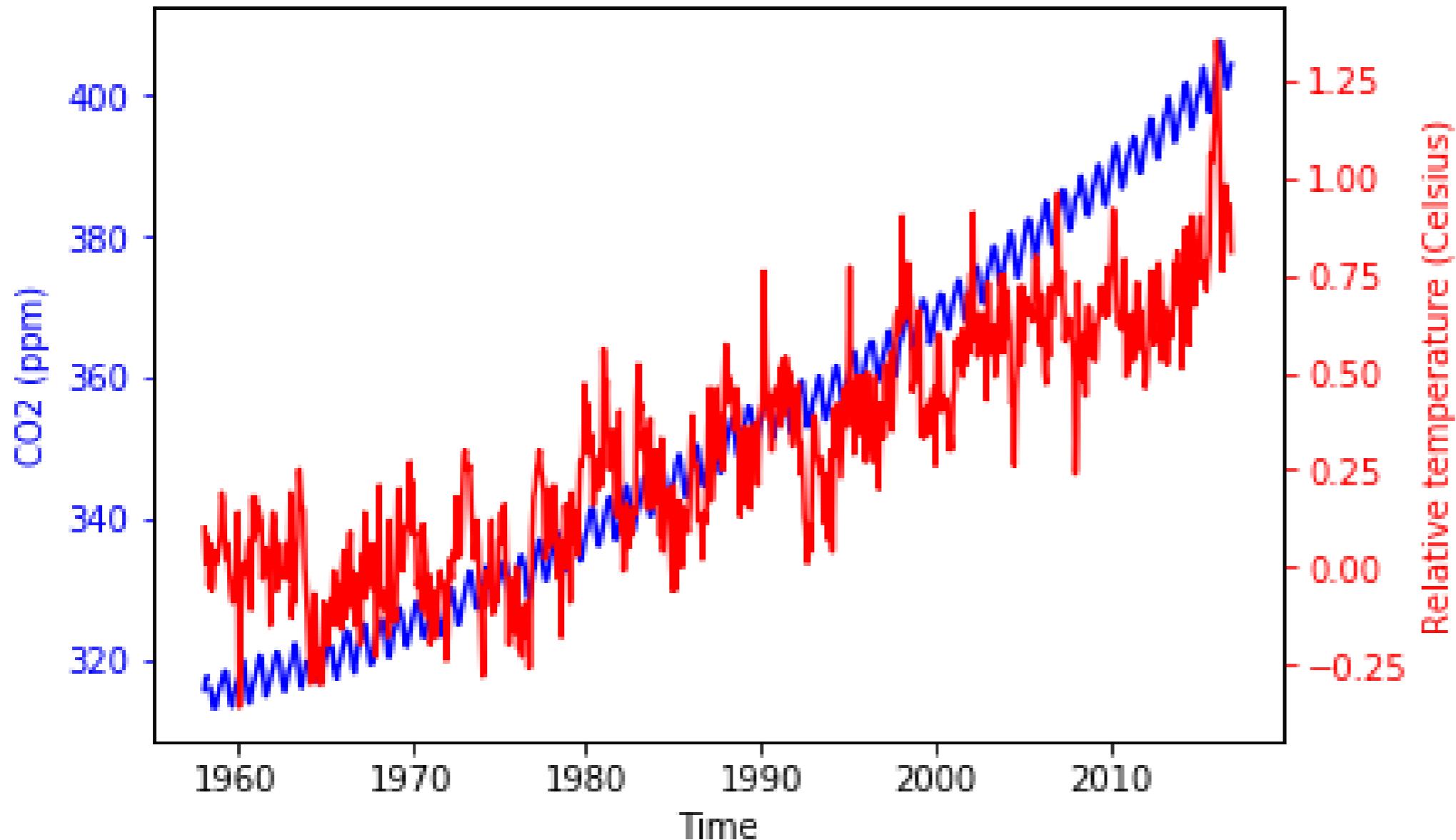
```
fig, ax = plt.subplots()
ax.plot(climate_change.index, climate_change["co2"], color='blue')
ax.set_xlabel('Time')
ax.set_ylabel('CO2 (ppm)', color='blue')
ax2 = ax.twinx()
ax2.plot(climate_change.index, climate_change["relative_temp"],
          color='red')
ax2.set_ylabel('Relative temperature (Celsius)', color='red')
plt.show()
```



Coloring the ticks

```
fig, ax = plt.subplots()
ax.plot(climate_change.index, climate_change["co2"],
         color='blue')
ax.set_xlabel('Time')
ax.set_ylabel('CO2 (ppm)', color='blue')
ax.tick_params('y', colors='blue')
ax2 = ax.twinx()
ax2.plot(climate_change.index,
          climate_change["relative_temp"],
          color='red')
ax2.set_ylabel('Relative temperature (Celsius)',
color='red')
ax2.tick_params('y', colors='red')
plt.show()
```

Coloring the ticks

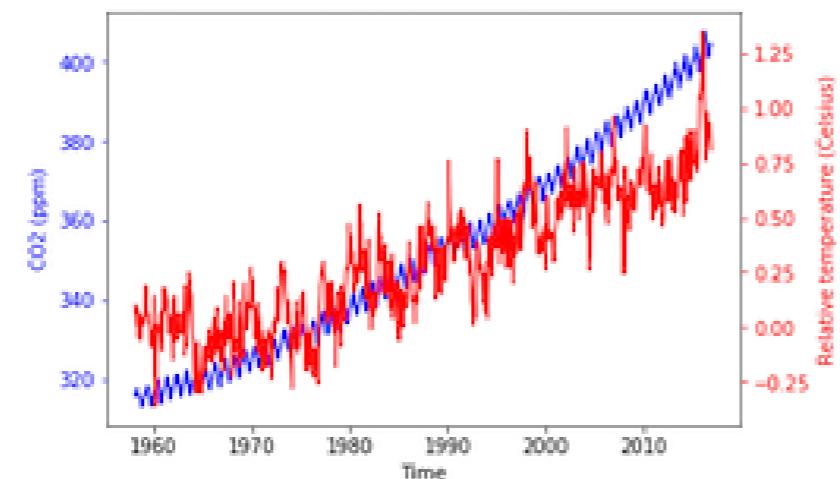


A function that plots time-series

```
def plot_timeseries(axes, x, y, color, xlabel, ylabel):  
    axes.plot(x, y, color=color)  
    axes.set_xlabel(xlabel)  
    axes.set_ylabel(ylabel, color=color)  
    axes.tick_params('y', colors=color)
```

Using our function

```
fig, ax = plt.subplots()  
plot_timeseries(ax, climate_change.index, climate_change['co2'],  
                 'blue', 'Time', 'CO2 (ppm)')  
ax2 = ax.twinx()  
plot_timeseries(ax, climate_change.index,  
                 climate_change['relative_temp'],  
                 'red', 'Time', 'Relative temperature (Celsius)')  
plt.show()
```



Create your own function!

INTRODUCTION TO DATA VISUALIZATION WITH MATPLOTLIB

Annotating time-series data

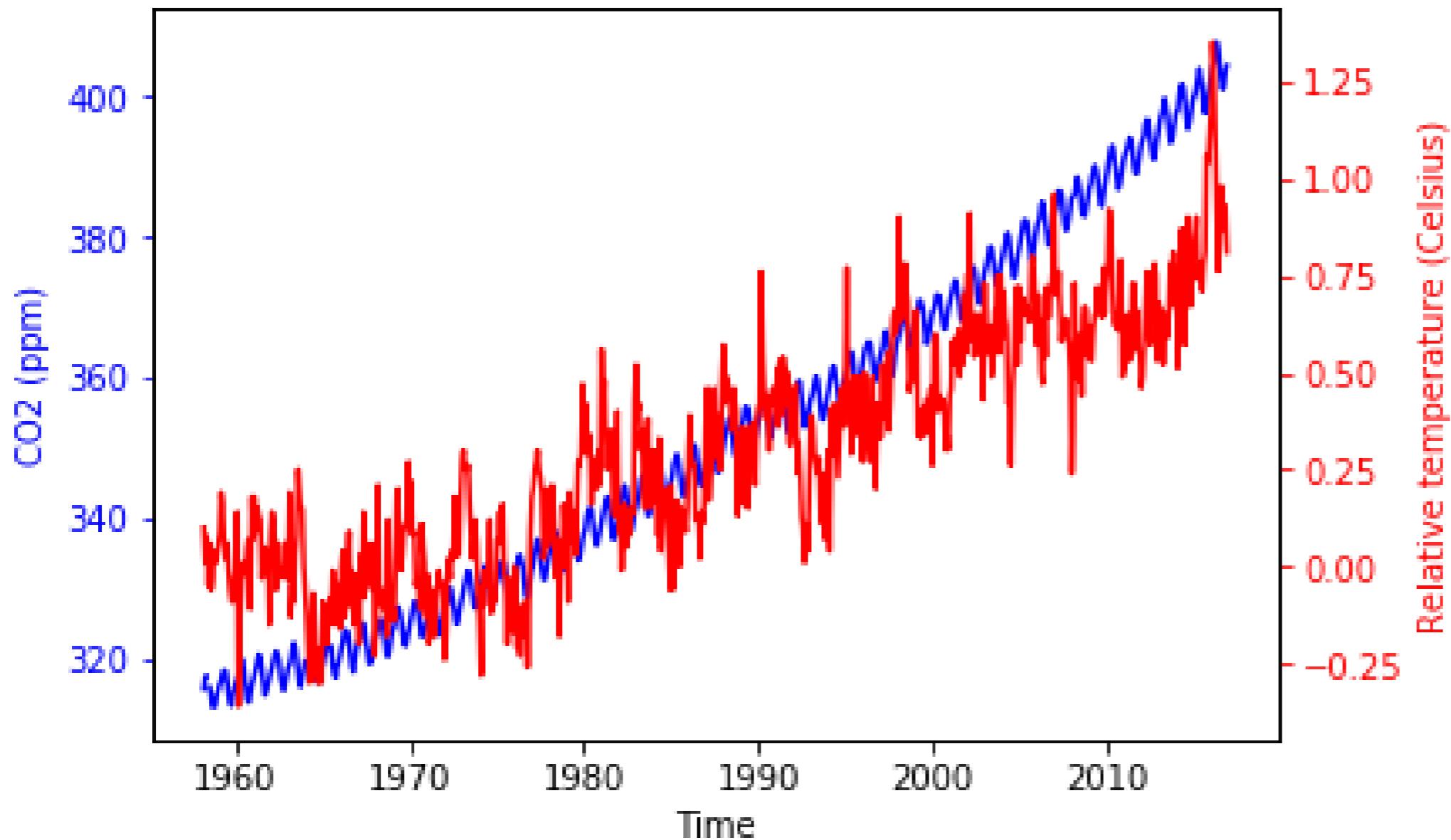
INTRODUCTION TO DATA VISUALIZATION WITH MATPLOTLIB



Ariel Rokem

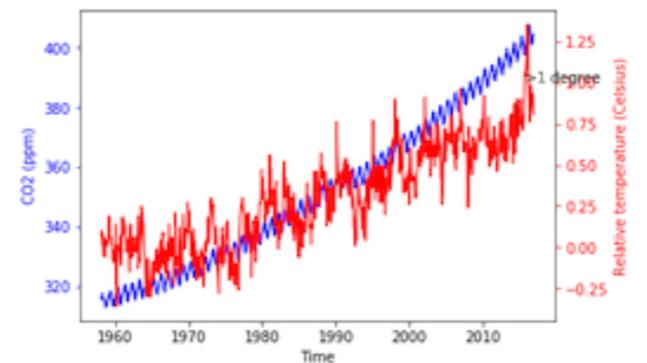
Data Scientist

Time-series data



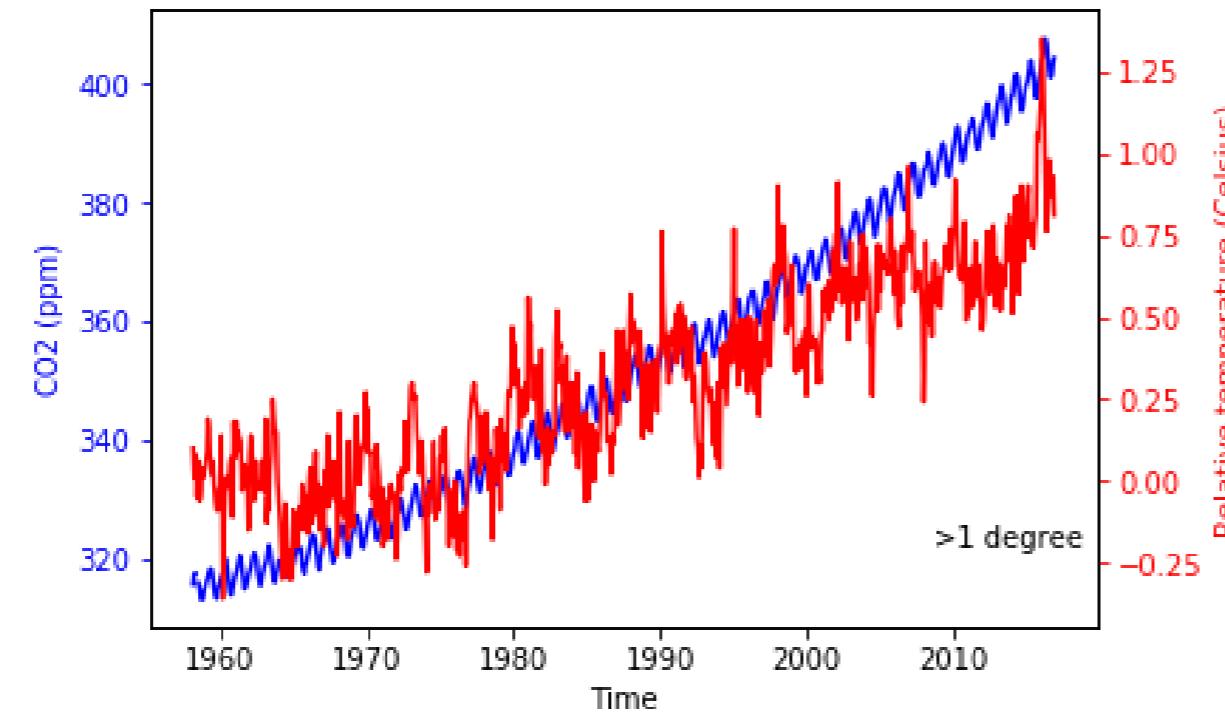
Annotation

```
fig, ax = plt.subplots()
plot_timeseries(ax, climate_change.index, climate_change['co2'],
                 'blue', 'Time', 'CO2 (ppm)')
ax2 = ax.twinx()
plot_timeseries(ax2, climate_change.index,
                 climate_change['relative_temp'],
                 'red', 'Time', 'Relative temperature (Celsius)')
ax2.annotate(">1 degree", xy=[pd.Timestamp("2015-10-06"), 1])
plt.show()
```



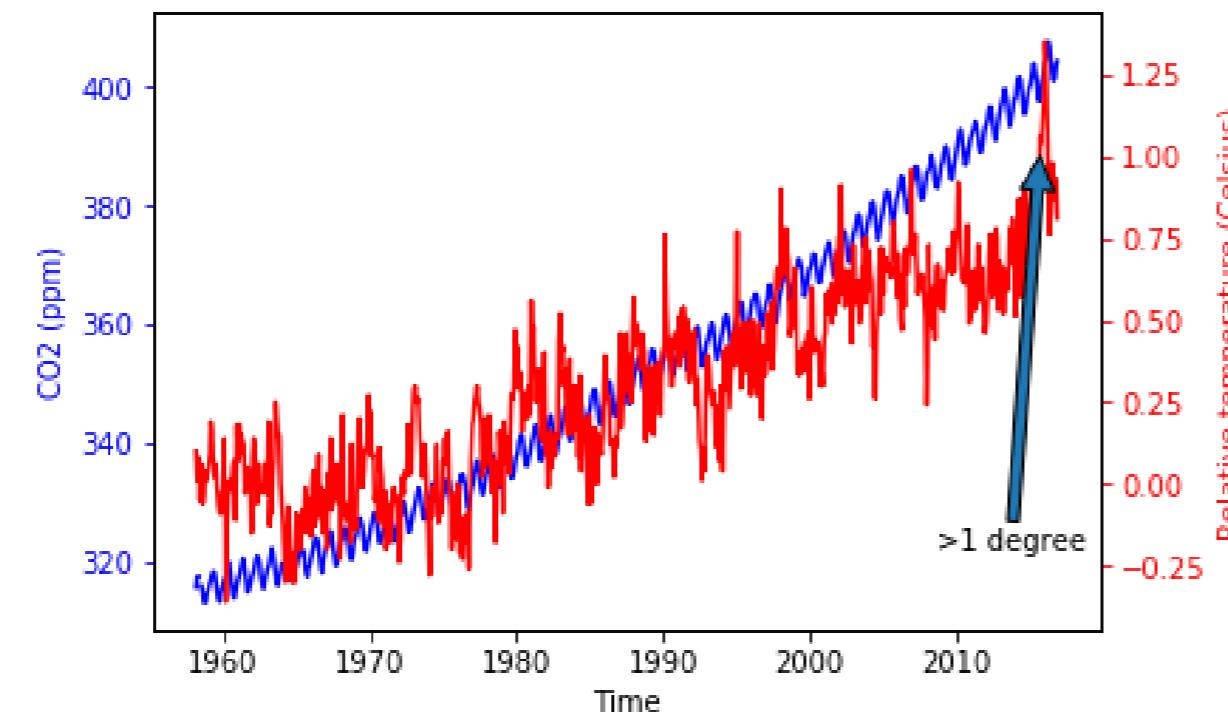
Positioning the text

```
ax2.annotate(">1 degree",  
            xy=(pd.Timestamp('2015-10-06'), 1),  
            xytext=(pd.Timestamp('2008-10-06'), -0.2))
```



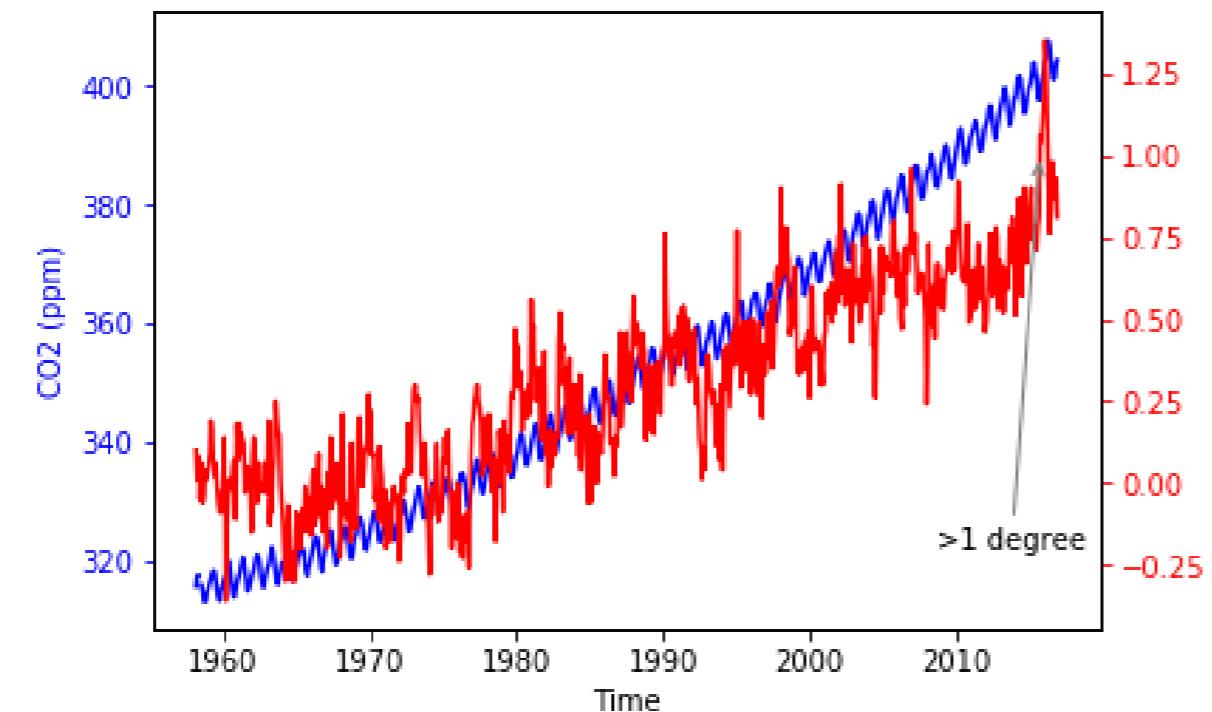
Adding arrows to annotation

```
ax2.annotate(">1 degree",
             xy=(pd.Timestamp('2015-10-06'), 1),
             xytext=(pd.Timestamp('2008-10-06'), -0.2),
             arrowprops={})
```



Customizing arrow properties

```
ax2.annotate(">1 degree",
    xy=(pd.Timestamp('2015-10-06'), 1),
    xytext=(pd.Timestamp('2008-10-06'), -0.2),
    arrowprops={"arrowstyle": "->", "color": "gray"})
```



Customizing annotations

<https://matplotlib.org/users/annotations.html>

Practice annotating plots!

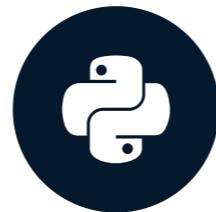
INTRODUCTION TO DATA VISUALIZATION WITH MATPLOTLIB

Quantitative comparisons: bar- charts

INTRODUCTION TO DATA VISUALIZATION WITH MATPLOTLIB

Ariel Rokem

Data Scientist

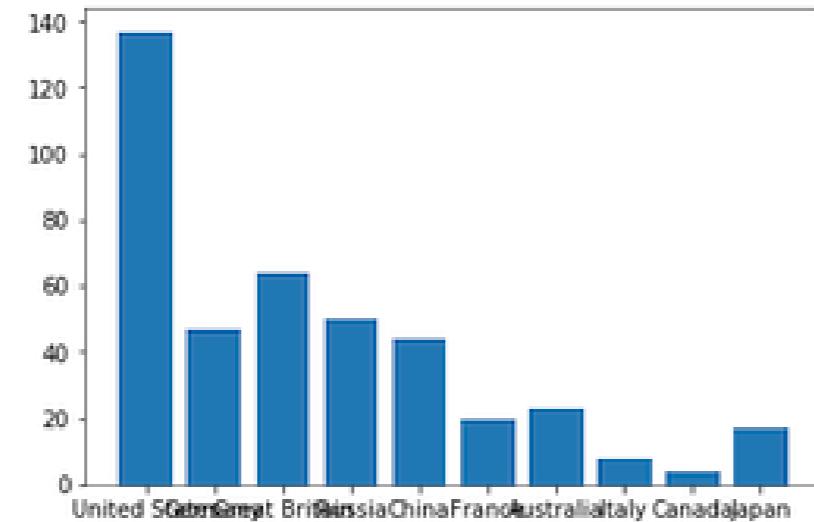


Olympic medals

, Gold, Silver, Bronze
United States, 137, 52, 67
Germany, 47, 43, 67
Great Britain, 64, 55, 26
Russia, 50, 28, 35
China, 44, 30, 35
France, 20, 55, 21
Australia, 23, 34, 25
Italy, 8, 38, 24
Canada, 4, 4, 61
Japan, 17, 13, 34

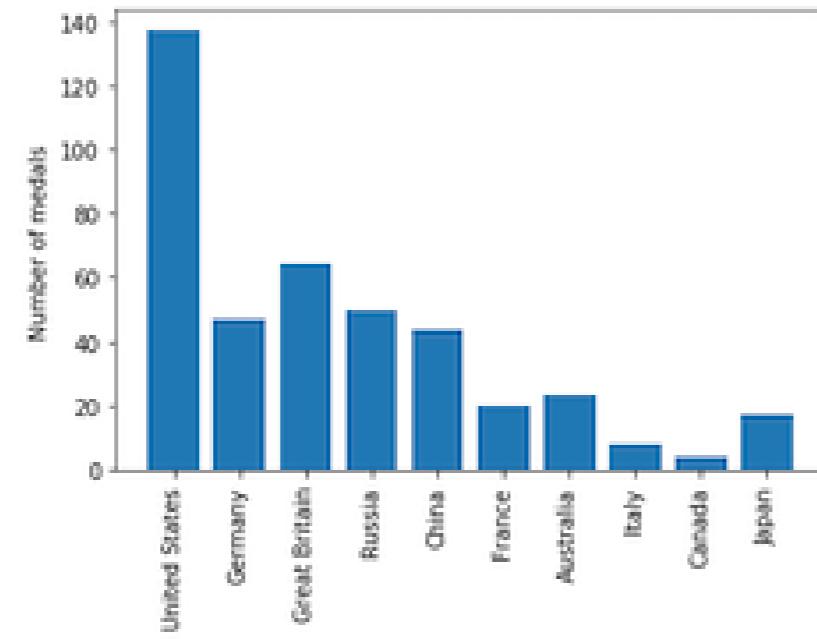
Olympic medals: visualizing the data

```
medals = pd.read_csv('medals_by_country_2016.csv', index_col=0)
fig, ax = plt.subplots()
ax.bar(medals.index, medals["Gold"])
plt.show()
```



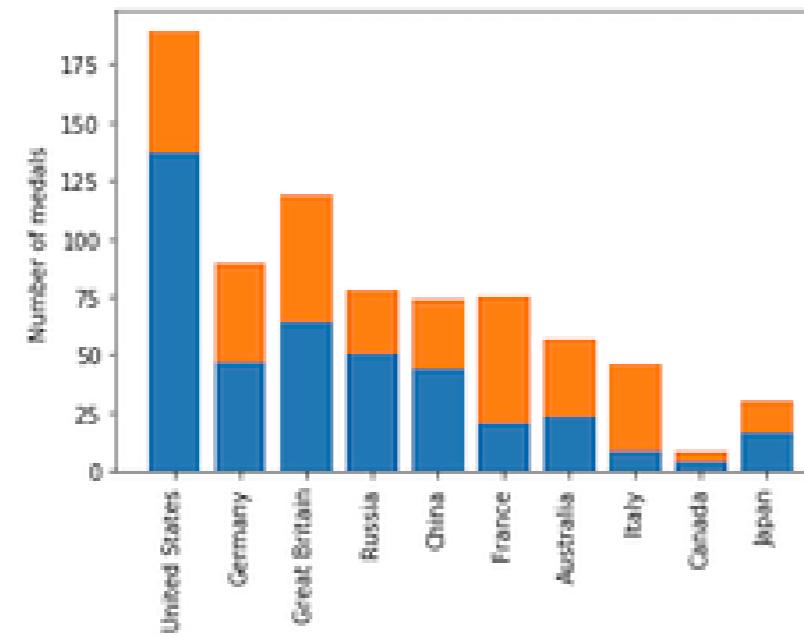
Interlude: rotate the tick labels

```
fig, ax = plt.subplots()  
ax.bar(medals.index, medals["Gold"])  
ax.set_xticklabels(medals.index, rotation=90)  
ax.set_ylabel("Number of medals")  
plt.show()
```



Olympic medals: visualizing the other medals

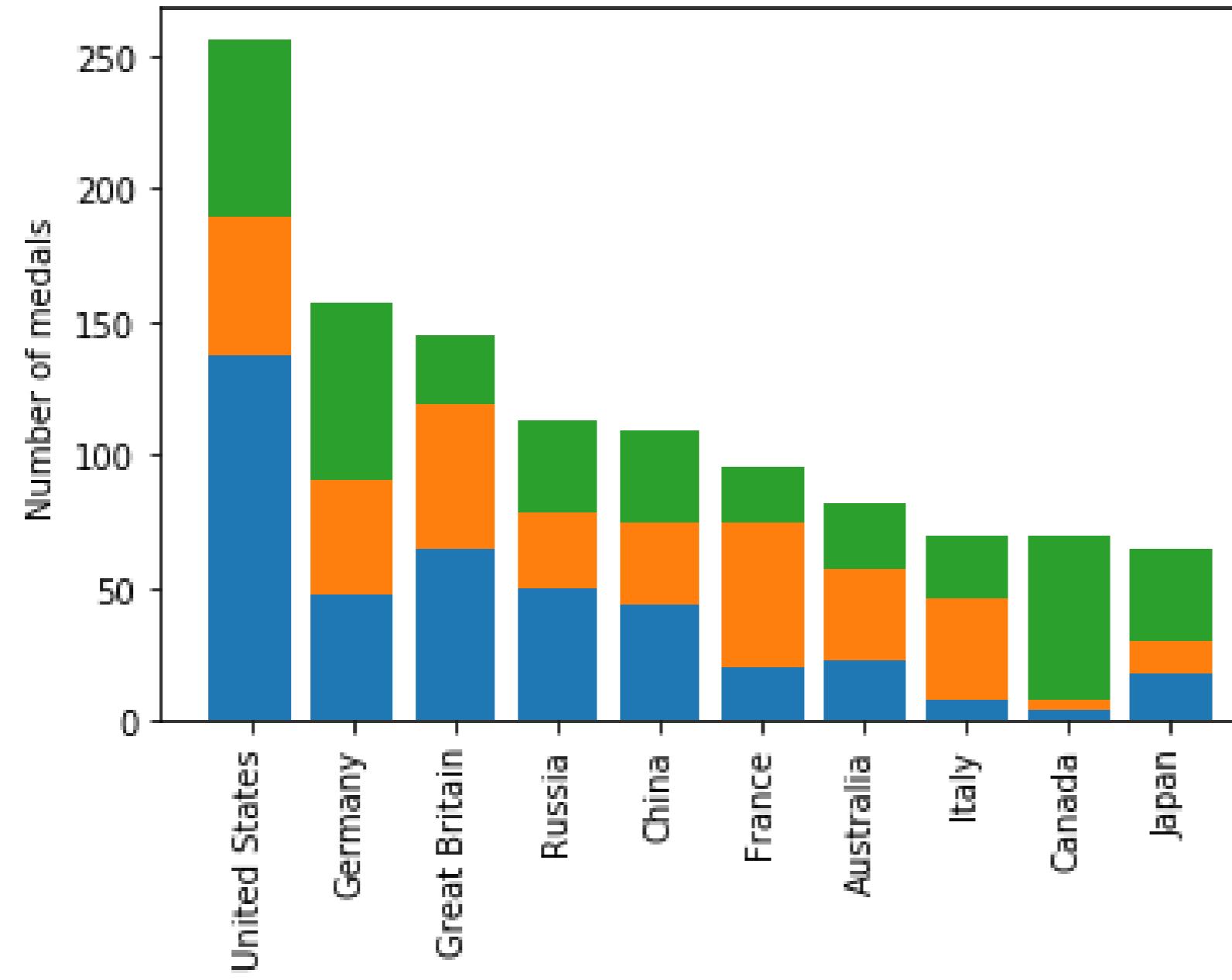
```
fig, ax = plt.subplots  
ax.bar(medals.index, medals["Gold"])  
ax.bar(medals.index, medals["Silver"], bottom=medals["Gold"])  
ax.set_xticklabels(medals.index, rotation=90)  
ax.set_ylabel("Number of medals")  
plt.show()
```



Olympic medals: visualizing all three

```
fig, ax = plt.subplots  
ax.bar(medals.index, medals["Gold"])  
  
ax.bar(medals.index, medals["Silver"], bottom=medals["Gold"])  
ax.bar(medals.index, medals["Bronze"],  
       bottom=medals["Gold"] + medals["Silver"])  
ax.set_xticklabels(medals.index, rotation=90)  
ax.set_ylabel("Number of medals")  
plt.show()
```

Stacked bar chart



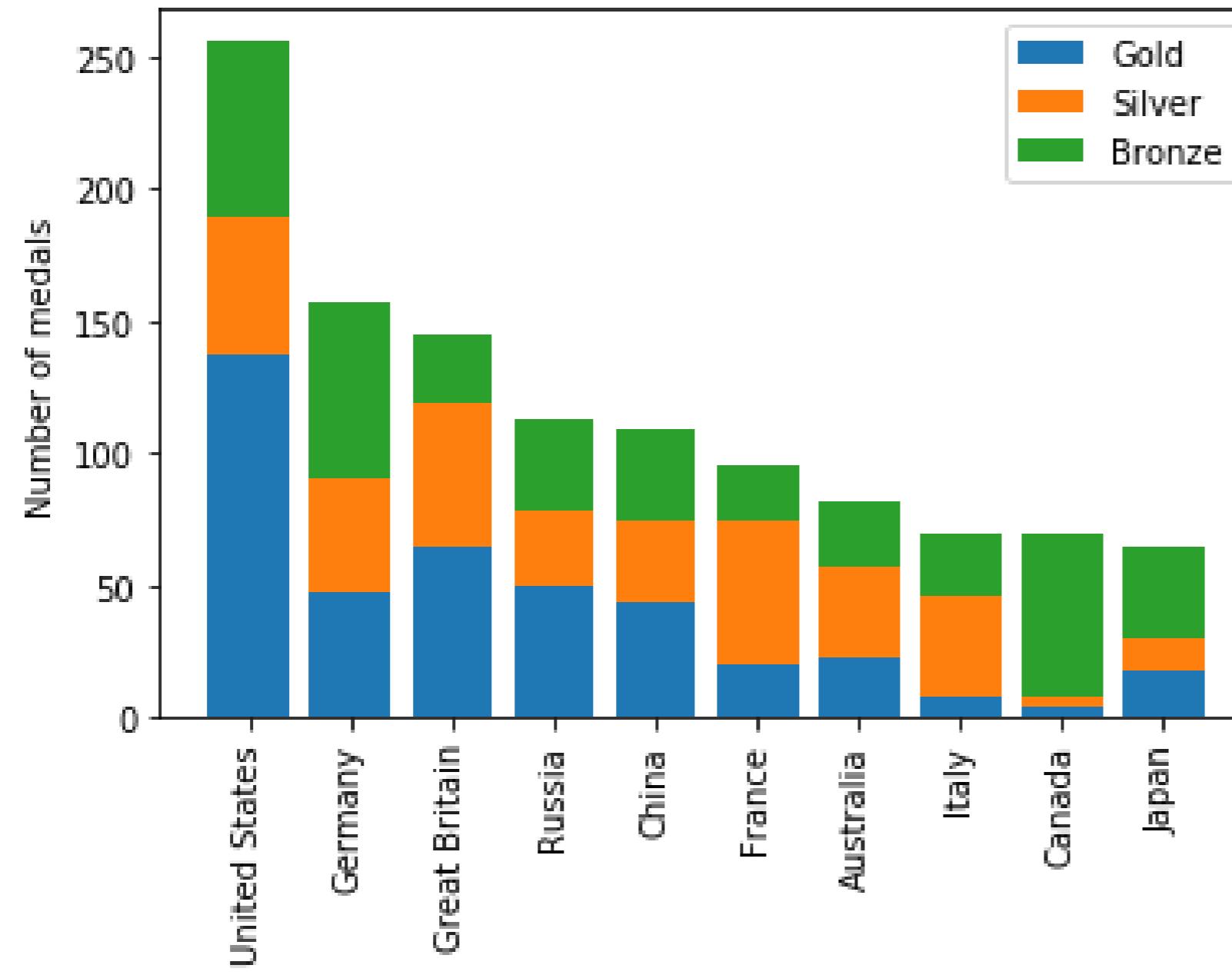
Adding a legend

```
fig, ax = plt.subplots  
ax.bar(medals.index, medals["Gold"])  
ax.bar(medals.index, medals["Silver"], bottom=medals["Gold"])  
ax.bar(medals.index, medals["Bronze"],  
       bottom=medals["Gold"] + medals["Silver"])  
  
ax.set_xticklabels(medals.index, rotation=90)  
ax.set_ylabel("Number of medals")
```

Adding a legend

```
fig, ax = plt.subplots  
ax.bar(medals.index, medals["Gold"], label="Gold")  
ax.bar(medals.index, medals["Silver"], bottom=medals["Gold"],  
       label="Silver")  
ax.bar(medals.index, medals["Bronze"],  
       bottom=medals["Gold"] + medals["Silver"],  
       label="Bronze")  
  
ax.set_xticklabels(medals.index, rotation=90)  
ax.set_ylabel("Number of medals")  
ax.legend()  
plt.show()
```

Stacked bar chart with legend



Create a bar chart!

INTRODUCTION TO DATA VISUALIZATION WITH MATPLOTLIB

Quantitative comparisons: histograms

INTRODUCTION TO DATA VISUALIZATION WITH MATPLOTLIB

Ariel Rokem

Data Scientist

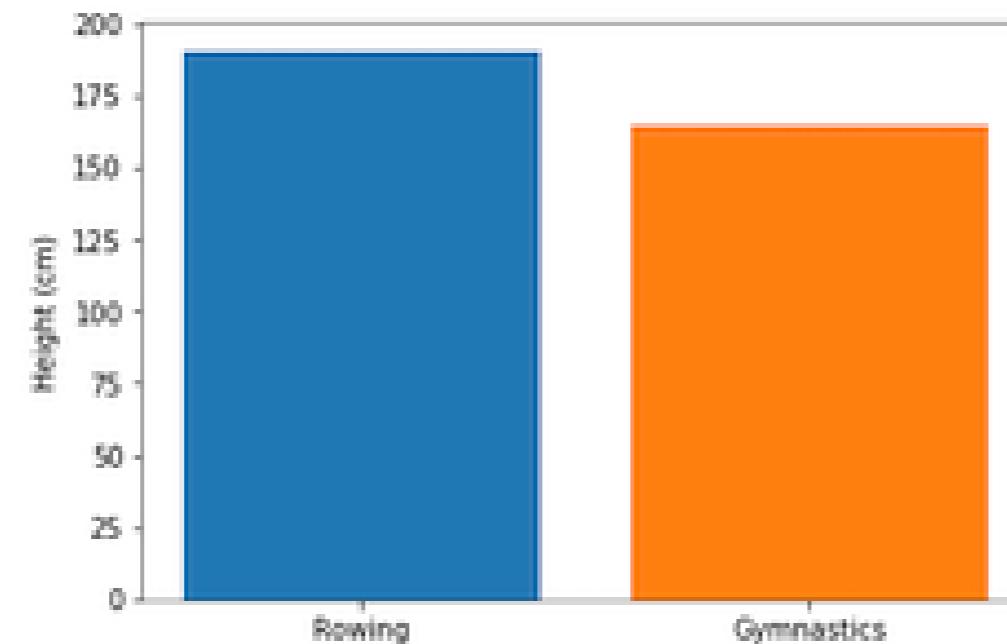


Histograms

ID		Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Season	City	Sport	Event	Medal
158	62	Giovanni Abagnale	M	21.0	198.0	90.0	Italy	ITA	2016 Summer	2016	Summer	Rio de Janeiro	Rowing	Rowing Men's Coxless Pairs	Bronze
11648	6346	Jrmie Azou	M	27.0	178.0	71.0	France	FRA	2016 Summer	2016	Summer	Rio de Janeiro	Rowing	Rowing Men's Lightweight Double Sculls	Gold
14871	8025	Thomas Gabriel Jrmie Baroukh	M	28.0	183.0	70.0	France	FRA	2016 Summer	2016	Summer	Rio de Janeiro	Rowing	Rowing Men's Lightweight Coxless Fours	Bronze
15215	8214	Jacob Jepsen Barse	M	27.0	188.0	73.0	Denmark	DEN	2016 Summer	2016	Summer	Rio de Janeiro	Rowing	Rowing Men's Lightweight Coxless Fours	Silver
18441	9764	Alexander Belonogoff	M	26.0	187.0	90.0	Australia	AUS	2016 Summer	2016	Summer	Rio de Janeiro	Rowing	Rowing Men's Quadruple Sculls	Silver

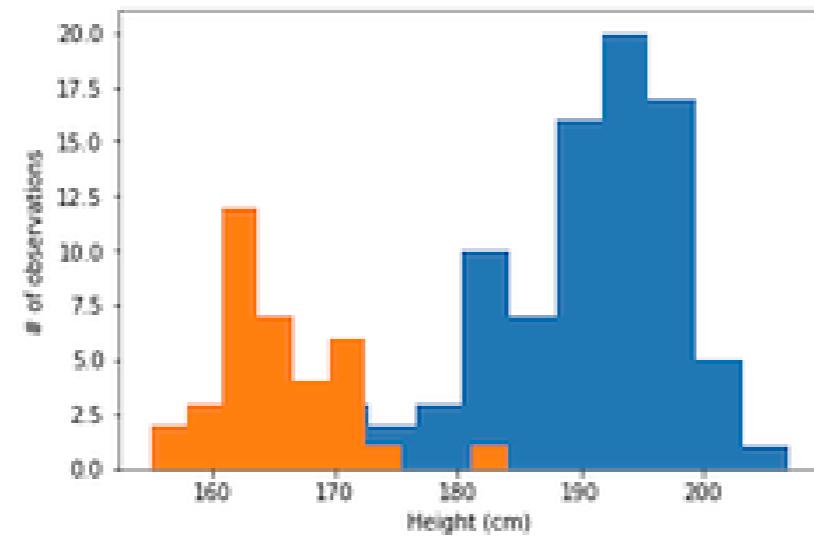
A bar chart again

```
fig, ax = plt.subplots()  
ax.bar("Rowing", mens_rowing["Height"].mean())  
ax.bar("Gymnastics", mens_gymnastics["Height"].mean())  
ax.set_ylabel("Height (cm)")  
plt.show()
```



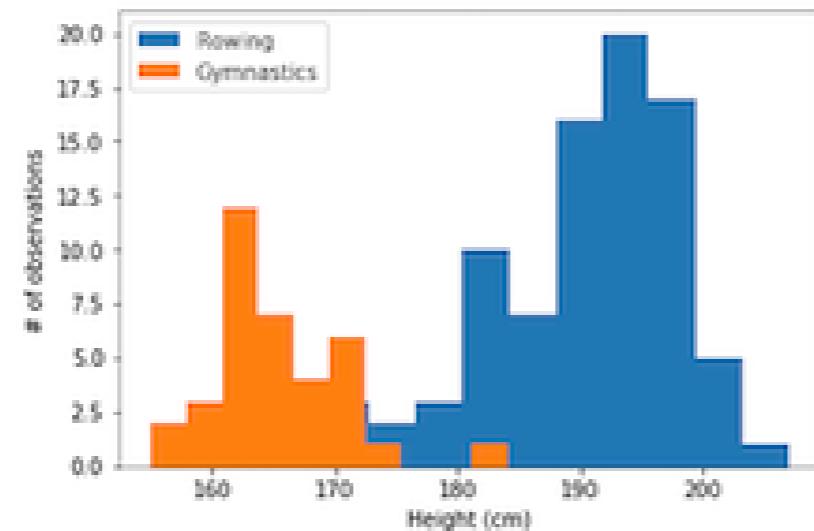
Introducing histograms

```
fig, ax = plt.subplots()  
ax.hist(mens_rowing["Height"])  
ax.hist(mens_gymnastic["Height"])  
ax.set_xlabel("Height (cm)")  
ax.set_ylabel("# of observations")  
plt.show()
```



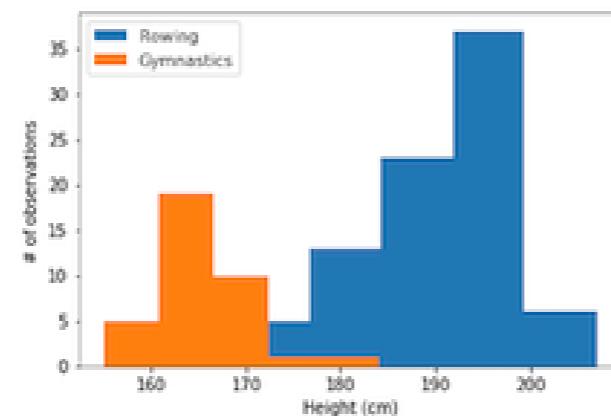
Labels are needed

```
ax.hist(mens_rowing["Height"], label="Rowing")
ax.hist(mens_gymnastic["Height"], label="Gymnastics")
ax.set_xlabel("Height (cm)")
ax.set_ylabel("# of observations")
ax.legend()
plt.show()
```



Customizing histograms: setting the number of bins

```
ax.hist(mens_rowing["Height"], label="Rowing", bins=5)
ax.hist(mens_gymnastic["Height"], label="Gymnastics", bins=5)
ax.set_xlabel("Height (cm)")
ax.set_ylabel("# of observations")
ax.legend()
plt.show()
```

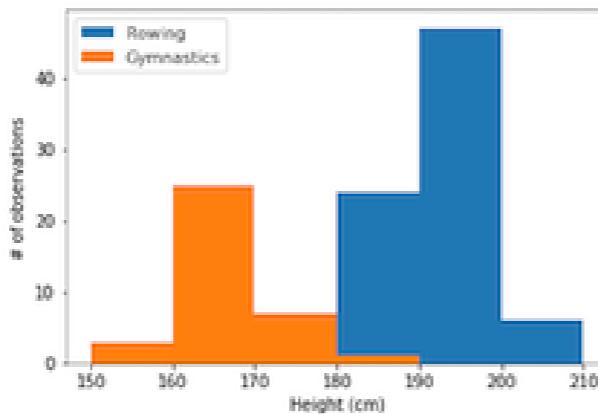


Customizing histograms: setting bin boundaries

```
ax.hist(mens_rowing["Height"], label="Rowing",
        bins=[150, 160, 170, 180, 190, 200, 210])

ax.hist(mens_gymnastic["Height"], label="Gymnastics",
        bins=[150, 160, 170, 180, 190, 200, 210])

ax.set_xlabel("Height (cm)")
ax.set_ylabel("# of observations")
ax.legend()
plt.show()
```



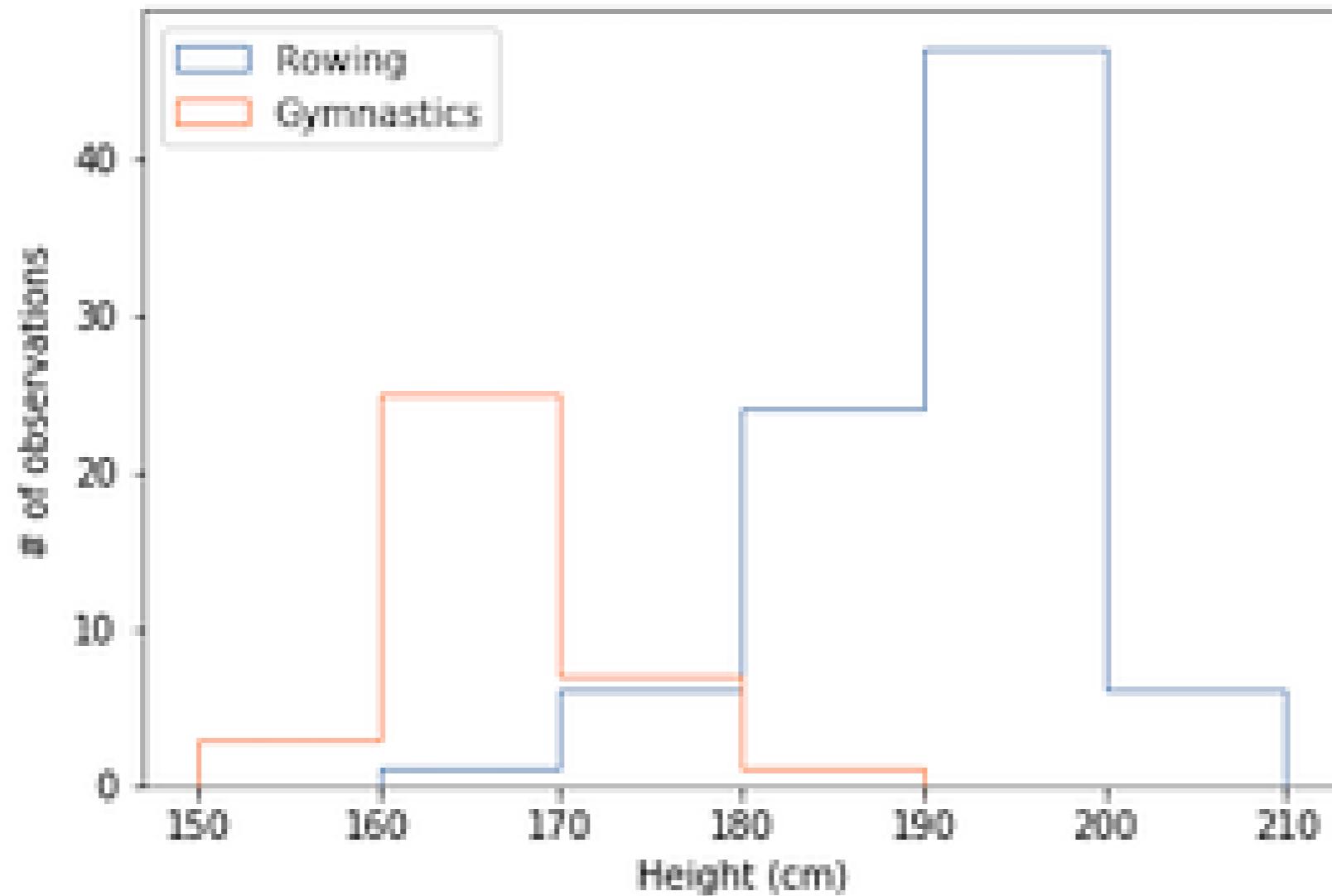
Customizing histograms: transparency

```
ax.hist(mens_rowing["Height"], label="Rowing",
        bins=[150, 160, 170, 180, 190, 200, 210],
        histtype="step")

ax.hist(mens_gymnastic["Height"], label="Gymnastics",
        bins=[150, 160, 170, 180, 190, 200, 210],
        histtype="step")

ax.set_xlabel("Height (cm)")
ax.set_ylabel("# of observations")
ax.legend()
plt.show()
```

Histogram with a histtype of step



Create your own histogram!

INTRODUCTION TO DATA VISUALIZATION WITH MATPLOTLIB

Statistical plotting

INTRODUCTION TO DATA VISUALIZATION WITH MATPLOTLIB



Ariel Rokem

Data Scientist

Adding error bars to bar charts

```
fig, ax = plt.subplots()

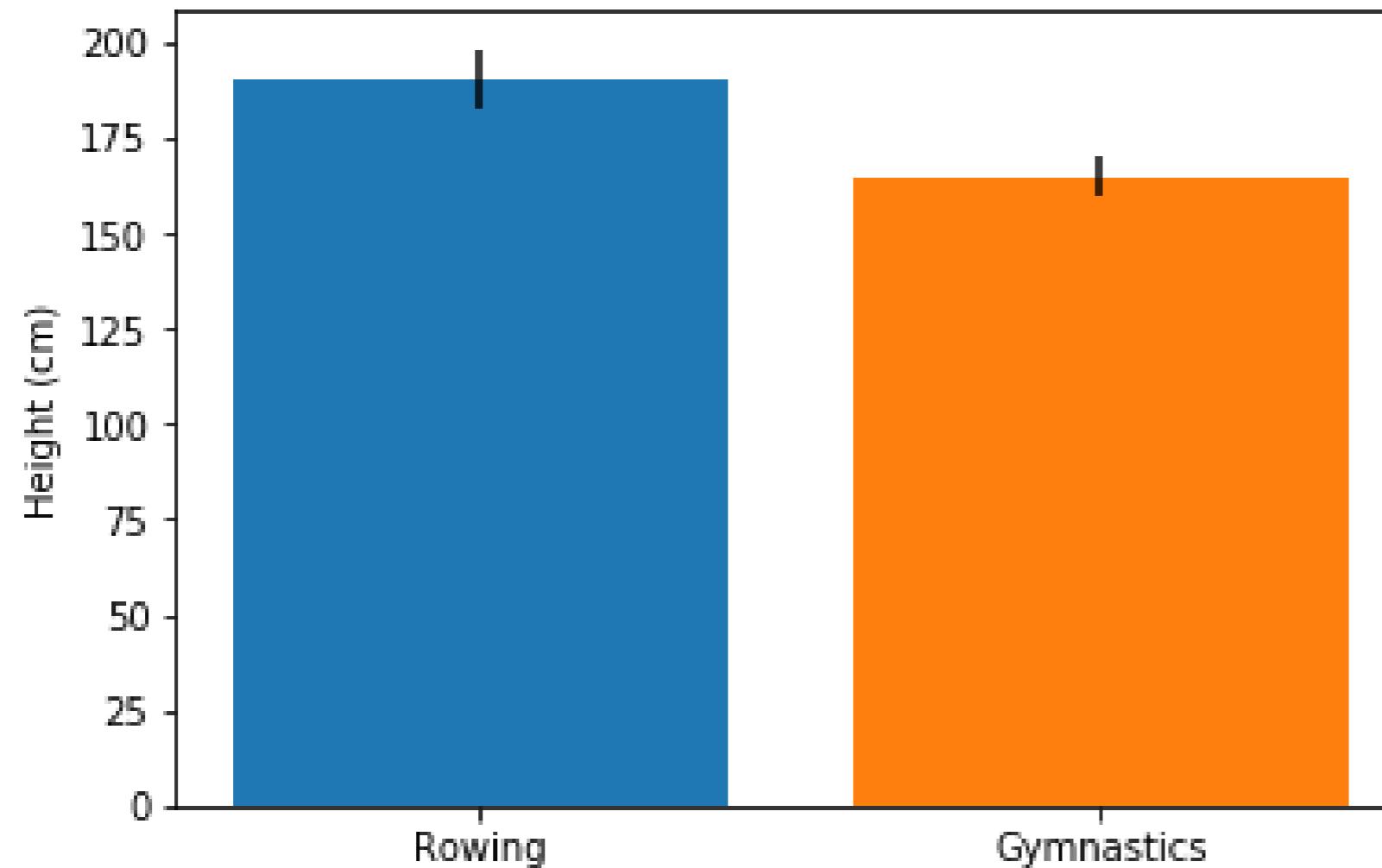
ax.bar("Rowing",
       mens_rowing["Height"].mean(),
       yerr=mens_rowing["Height"].std())

ax.bar("Gymnastics",
       mens_gymnastics["Height"].mean(),
       yerr=mens_gymnastics["Height"].std())

ax.set_ylabel("Height (cm)")

plt.show()
```

Error bars in a bar chart



Adding error bars to plots

```
fig, ax = plt.subplots()

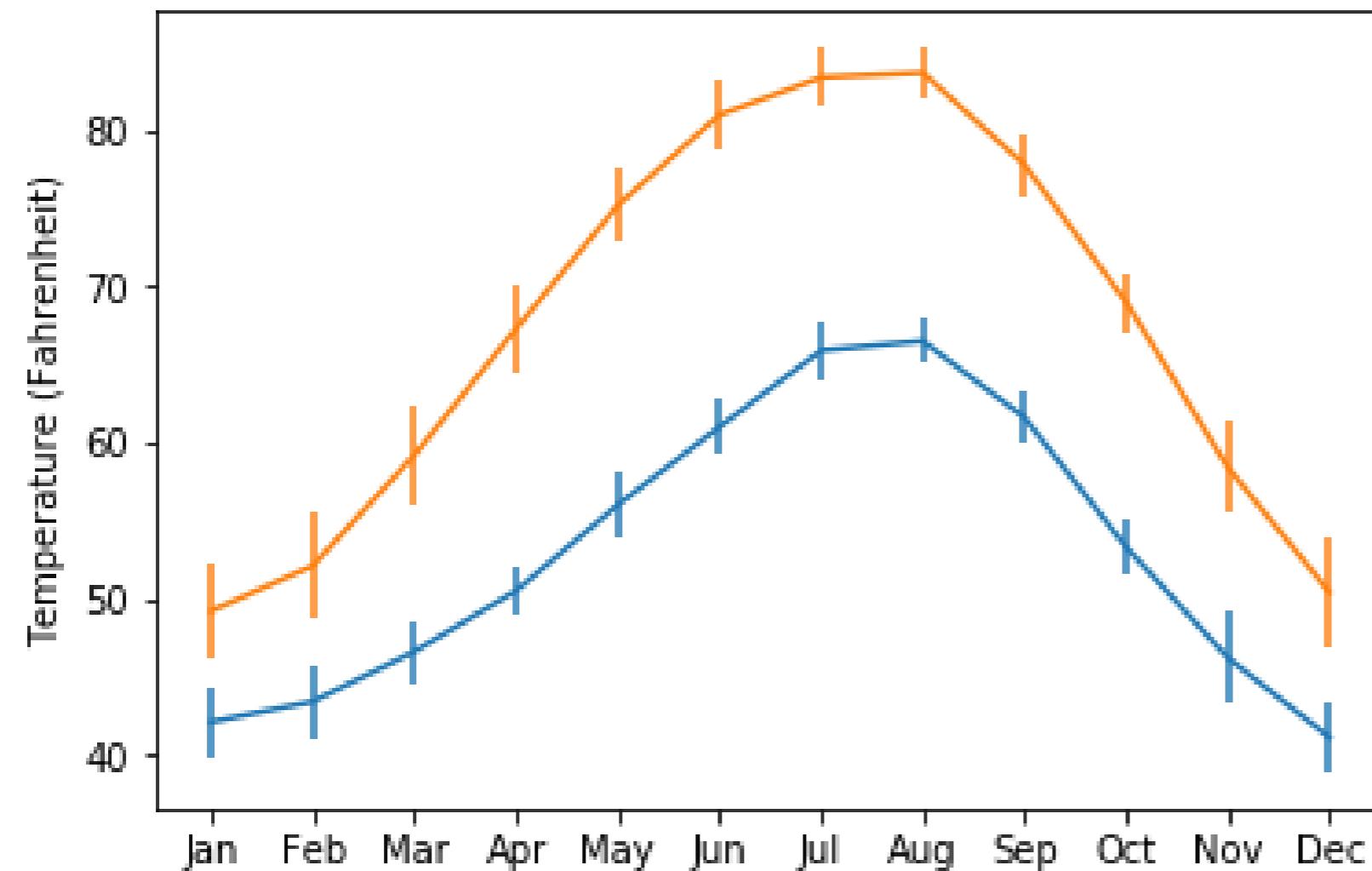
ax.errorbar(seattle_weather["MONTH"],
            seattle_weather["MLY-TAVG-NORMAL"],
            yerr=seattle_weather["MLY-TAVG-STDDEV"])

ax.errorbar(austin_weather["MONTH"],
            austin_weather["MLY-TAVG-NORMAL"],
            yerr=austin_weather["MLY-TAVG-STDDEV"])

ax.set_ylabel("Temperature (Fahrenheit)")

plt.show()
```

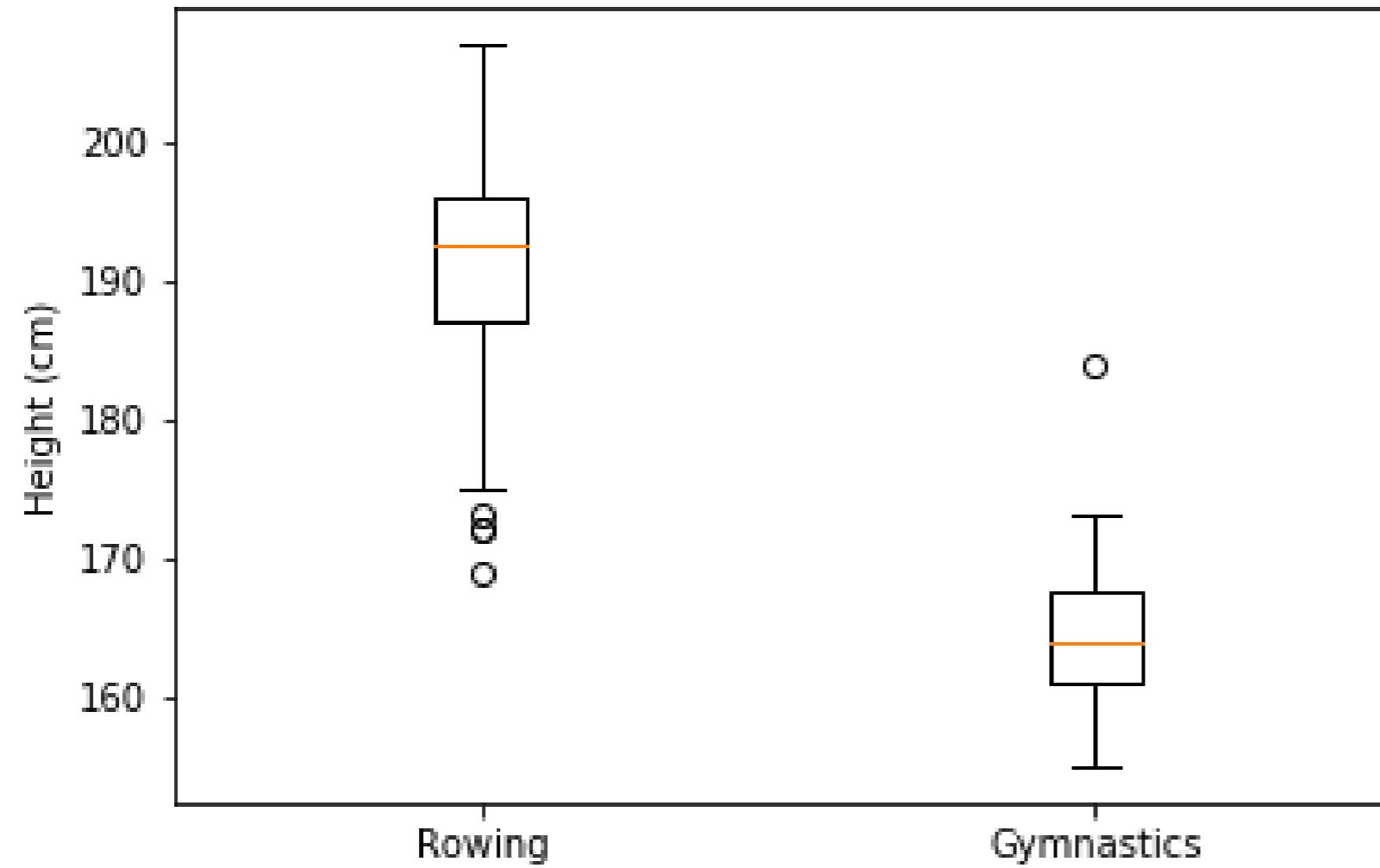
Error bars in plots



Adding boxplots

```
fig, ax = plt.subplots()  
ax.boxplot([mens_rowing["Height"],  
            mens_gymnastics["Height"]])  
ax.set_xticklabels(["Rowing", "Gymnastics"])  
ax.set_ylabel("Height (cm)")  
  
plt.show()
```

Interpreting boxplots



Try it yourself!

INTRODUCTION TO DATA VISUALIZATION WITH MATPLOTLIB

Quantitative comparisons: scatter plots

INTRODUCTION TO DATA VISUALIZATION WITH MATPLOTLIB

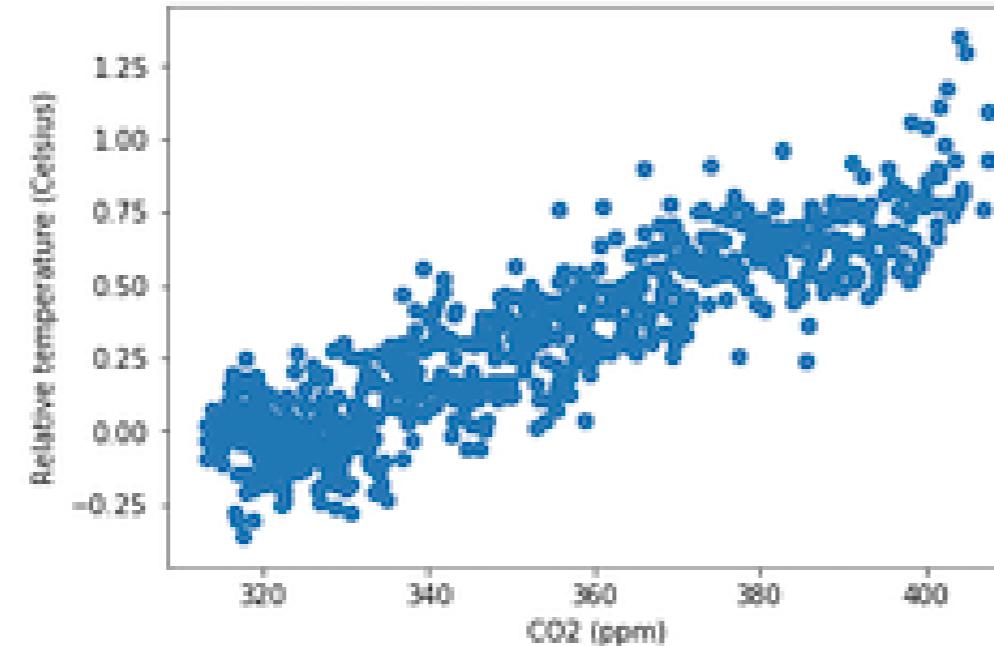
Ariel Rokem

Data Scientist



Introducing scatter plots

```
fig, ax = plt.subplots()  
ax.scatter(climate_change["co2"], climate_change["relative_temp"])  
ax.set_xlabel("CO2 (ppm)")  
ax.set_ylabel("Relative temperature (Celsius)")  
plt.show()
```



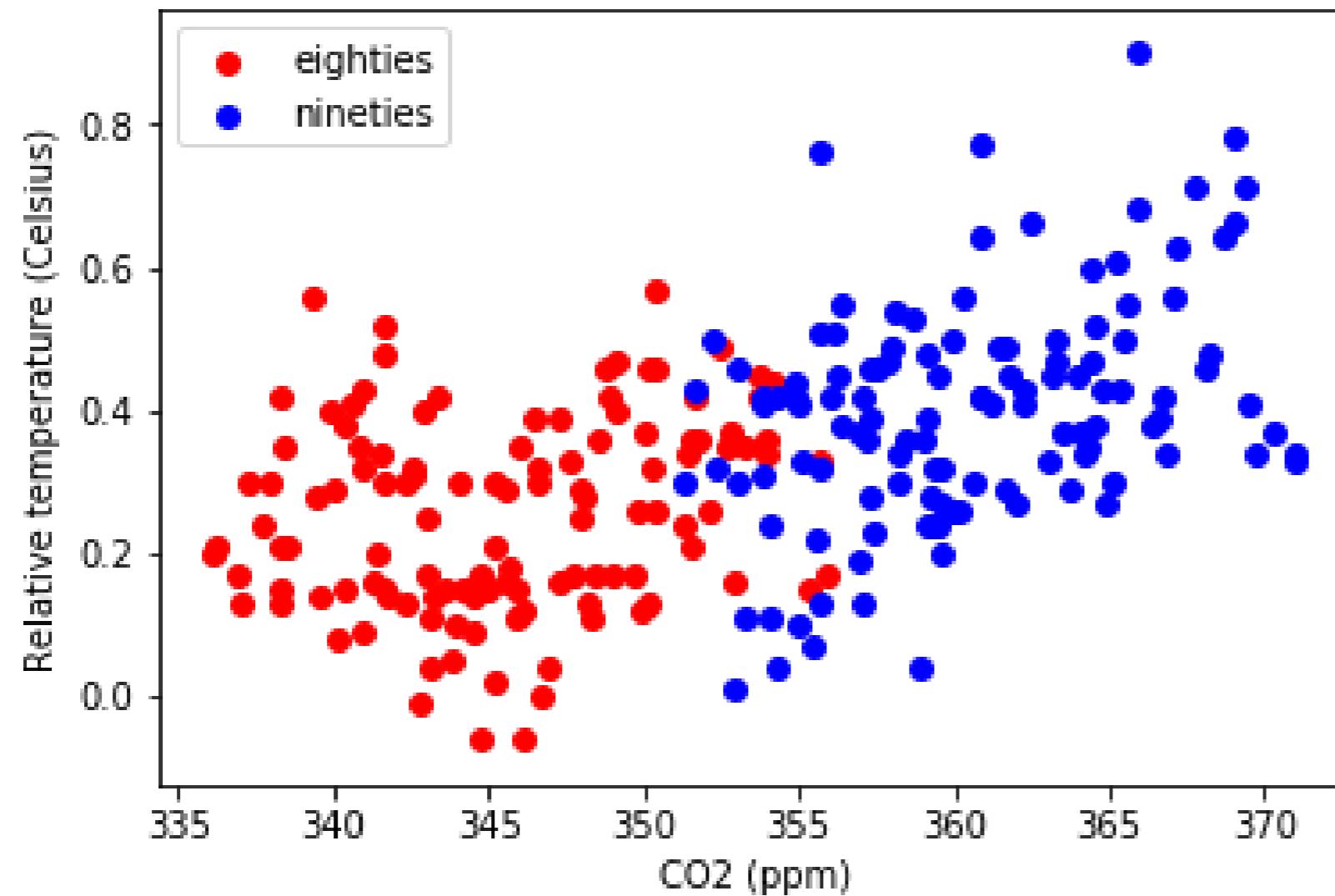
Customizing scatter plots

```
eighties = climate_change["1980-01-01":"1989-12-31"]
nineties = climate_change["1990-01-01":"1999-12-31"]
fig, ax = plt.subplots()
ax.scatter(eighties["co2"], eighties["relative_temp"],
           color="red", label="eighties")
ax.scatter(nineties["co2"], nineties["relative_temp"],
           color="blue", label="nineties")
ax.legend()

ax.set_xlabel("CO2 (ppm)")
ax.set_ylabel("Relative temperature (Celsius)")

plt.show()
```

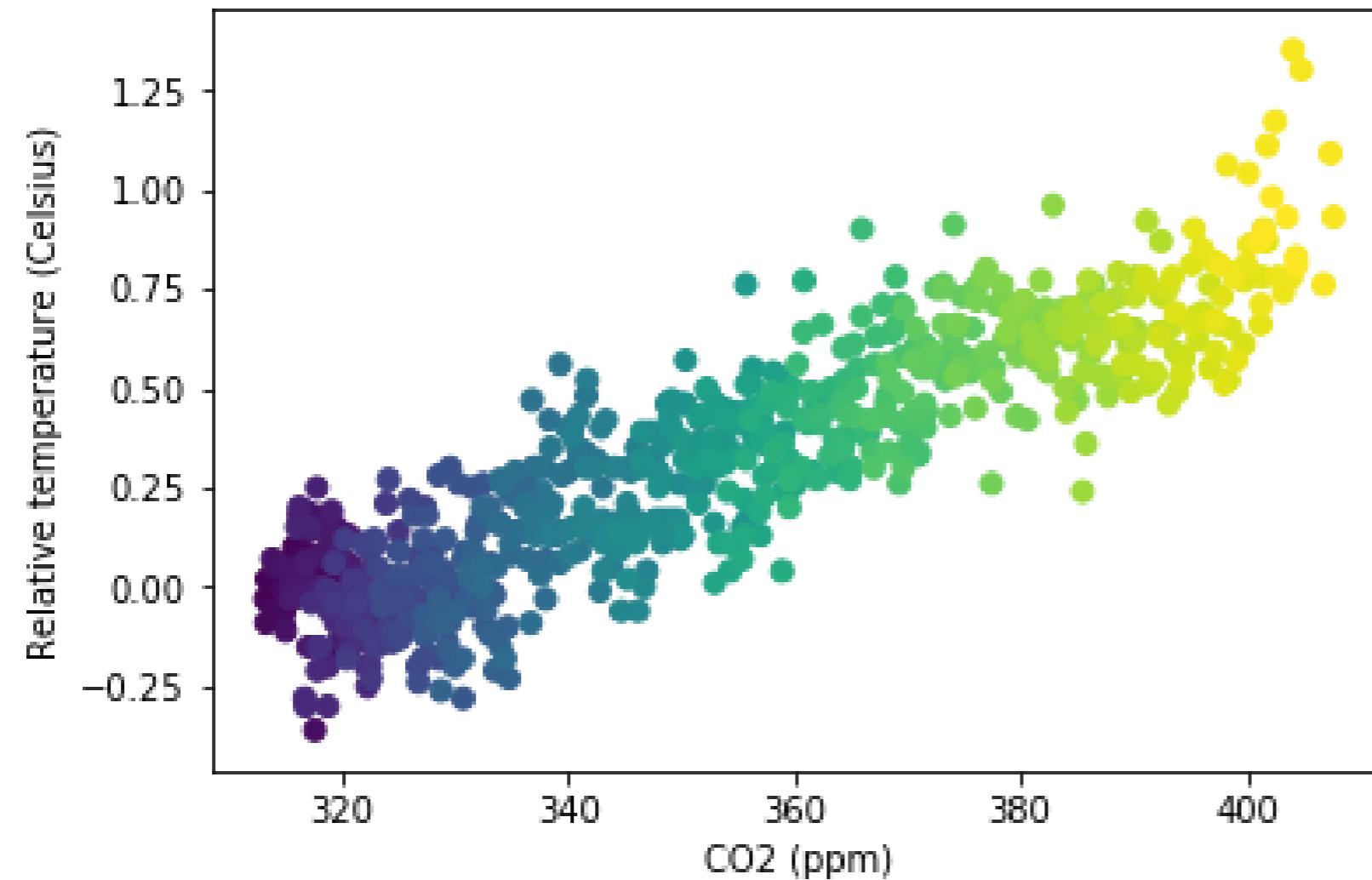
Encoding a comparison by color



Encoding a third variable by color

```
fig, ax = plt.subplots()  
ax.scatter(climate_change["co2"], climate_change["relative_temp"],  
           c=climate_change.index)  
ax.set_xlabel("CO2 (ppm)")  
ax.set_ylabel("Relative temperature (Celsius)")  
plt.show()
```

Encoding time in color



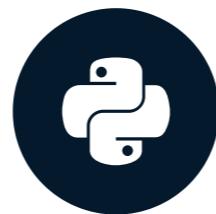
**Practice making
your own scatter
plots!**

INTRODUCTION TO DATA VISUALIZATION WITH MATPLOTLIB

Preparing your figures to share with others

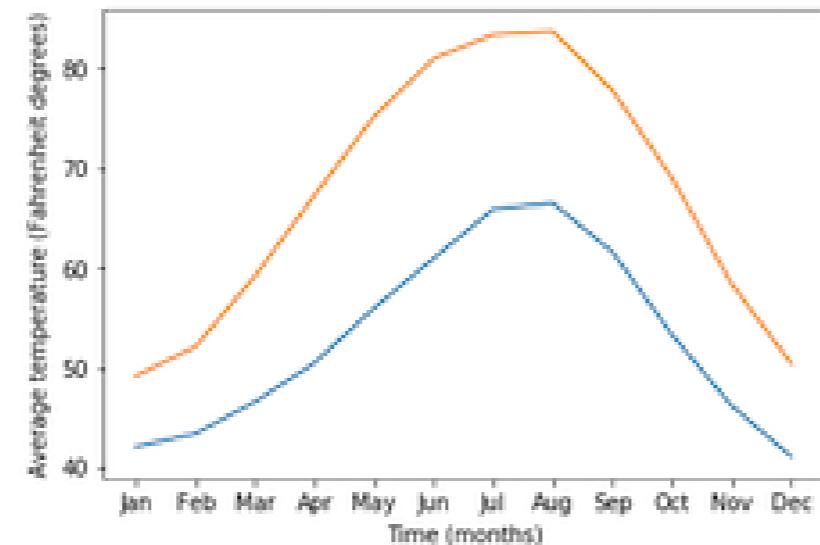
INTRODUCTION TO DATA VISUALIZATION WITH MATPLOTLIB

Ariel Rokem
Data Scientist



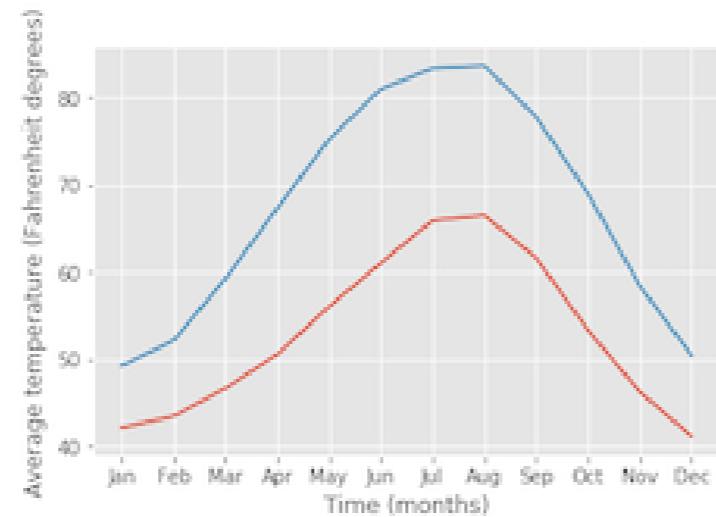
Changing plot style

```
import matplotlib.pyplot as plt  
fig, ax = plt.subplots()  
ax.plot(seattle_weather["MONTH"], seattle_weather["MLY-TAVG-NORMAL"]  
ax.plot(austin_weather["MONTH"], austin_weather["MLY-TAVG-NORMAL"])  
ax.set_xlabel("Time (months)")  
ax.set_ylabel("Average temperature (Fahrenheit degrees)")  
plt.show()
```



Choosing a style

```
plt.style.use("ggplot")
fig, ax = plt.subplots()
ax.plot(seattle_weather["MONTH"], seattle_weather["MLY-TAVG-NORMAL"])
ax.plot(austin_weather["MONTH"], austin_weather["MLY-TAVG-NORMAL"])
ax.set_xlabel("Time (months)")
ax.set_ylabel("Average temperature (Fahrenheit degrees)")
plt.show()
```



Back to the default

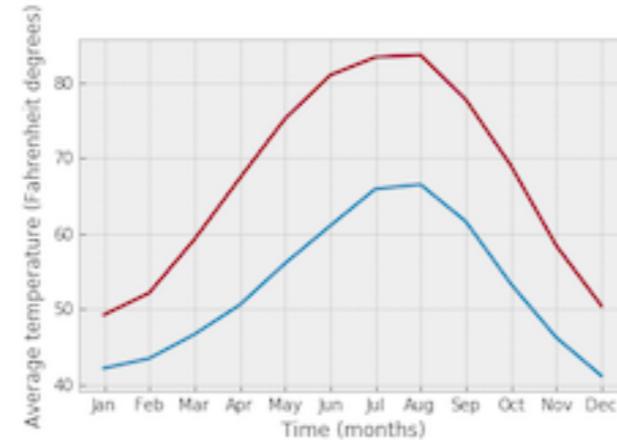
```
plt.style.use("default")
```

The available styles

https://matplotlib.org/gallery/style_sheets/style_sheets_reference.html

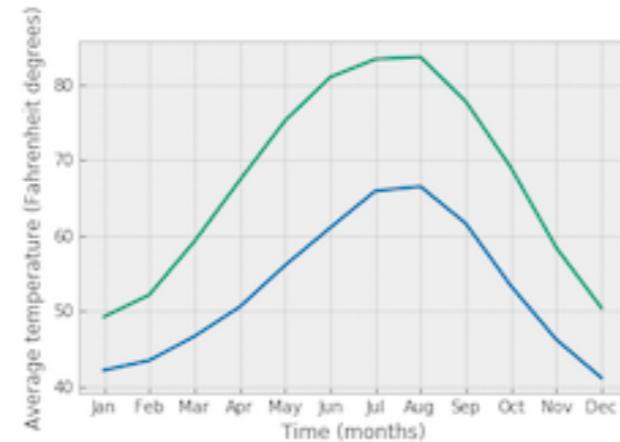
The "bmh" style

```
plt.style.use("bmh")
fig, ax = plt.subplots()
ax.plot(seattle_weather["MONTH"], seattle_weather["MLY-TAVG-NORMAL"])
ax.plot(austin_weather["MONTH"], austin_weather["MLY-TAVG-NORMAL"])
ax.set_xlabel("Time (months)")
ax.set_ylabel("Average temperature (Fahrenheit degrees)")
plt.show()
```



Seaborn styles

```
plt.style.use("seaborn-colorblind")
fig, ax = plt.subplots()
ax.plot(seattle_weather["MONTH"], seattle_weather["MLY-TAVG-NORMAL"])
ax.plot(austin_weather["MONTH"], austin_weather["MLY-TAVG-NORMAL"])
ax.set_xlabel("Time (months)")
ax.set_ylabel("Average temperature (Fahrenheit degrees)")
plt.show()
```



Guidelines for choosing plotting style

- Dark backgrounds are usually less visible
- If color is important, consider choosing colorblind-friendly options
 - "seaborn-colorblind" or "tableau-colorblind10"
- If you think that someone will want to print your figure, use less ink
- If it will be printed in black-and-white, use the "grayscale" style

**Practice choosing
the right style for
you!**

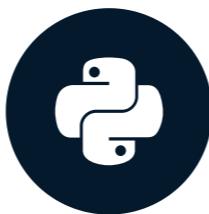
INTRODUCTION TO DATA VISUALIZATION WITH MATPLOTLIB

Sharing your visualizations with others

INTRODUCTION TO DATA VISUALIZATION WITH MATPLOTLIB

Ariel Rokem

Data Scientist

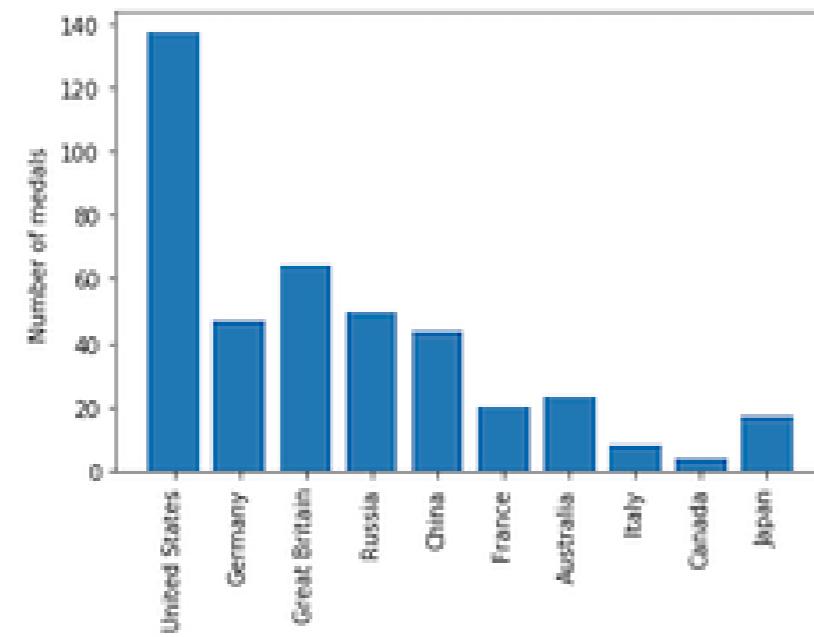


A figure to share

```
fig, ax = plt.subplots()

ax.bar(medals.index, medals["Gold"])
ax.set_xticklabels(medals.index, rotation=90)
ax.set_ylabel("Number of medals")

plt.show()
```



Saving the figure to file

```
fig, ax = plt.subplots()  
  
ax.bar(medals.index, medals["Gold"])  
ax.set_xticklabels(medals.index, rotation=90)  
ax.set_ylabel("Number of medals")  
  
fig.savefig("gold_medals.png")
```

```
ls
```

```
gold_medals.png
```

Different file formats

```
fig.savefig("gold_medals.jpg")
```

```
fig.savefig("gold_medals.jpg", quality=50)
```

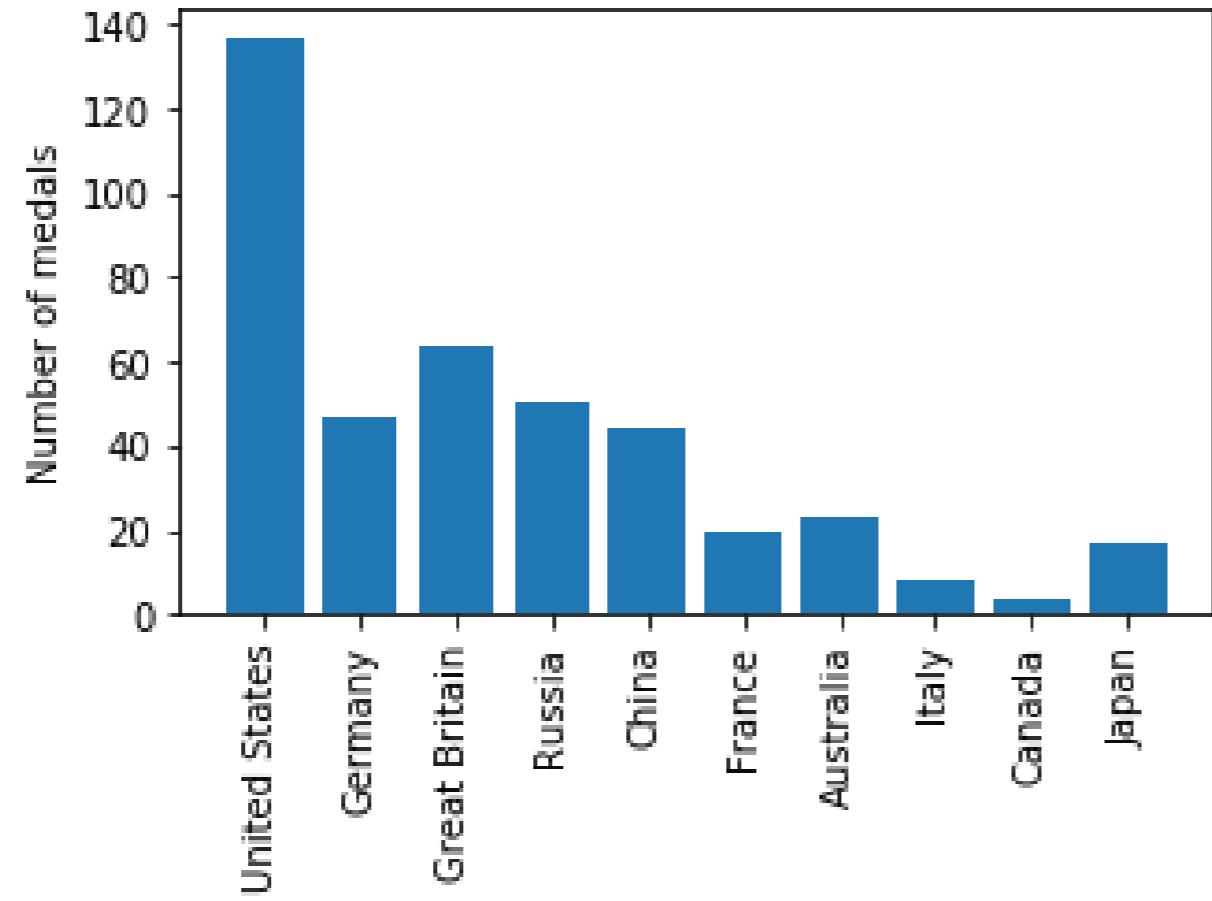
```
fig.savefig("gold_medals.svg")
```

Resolution

```
fig.savefig("gold_medals.png", dpi=300)
```

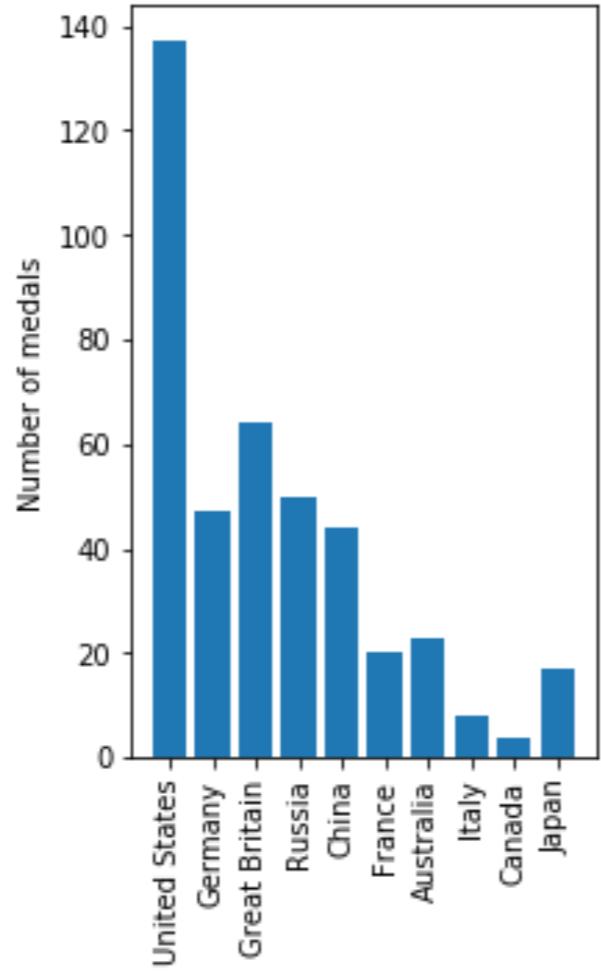
Size

```
fig.set_size_inches([5, 3])
```



Another aspect ratio

```
fig.set_size_inches([3, 5])
```

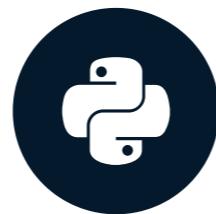


Practice saving your visualizations!

INTRODUCTION TO DATA VISUALIZATION WITH MATPLOTLIB

Automating figures from data

INTRODUCTION TO DATA VISUALIZATION WITH MATPLOTLIB



Ariel Rokem

Data Scientist

Why automate?

- Ease and speed
- Flexibility
- Robustness
- Reproducibility

How many different kinds of data?

```
summer_2016_medals["Sport"]
```

```
ID  
62      Rowing  
65      Taekwondo  
73      Handball  
       ...  
134759   Handball  
135132   Volleyball  
135205   Boxing  
Name: Sport, Length: 976, dtype: object
```

Getting unique values of a column

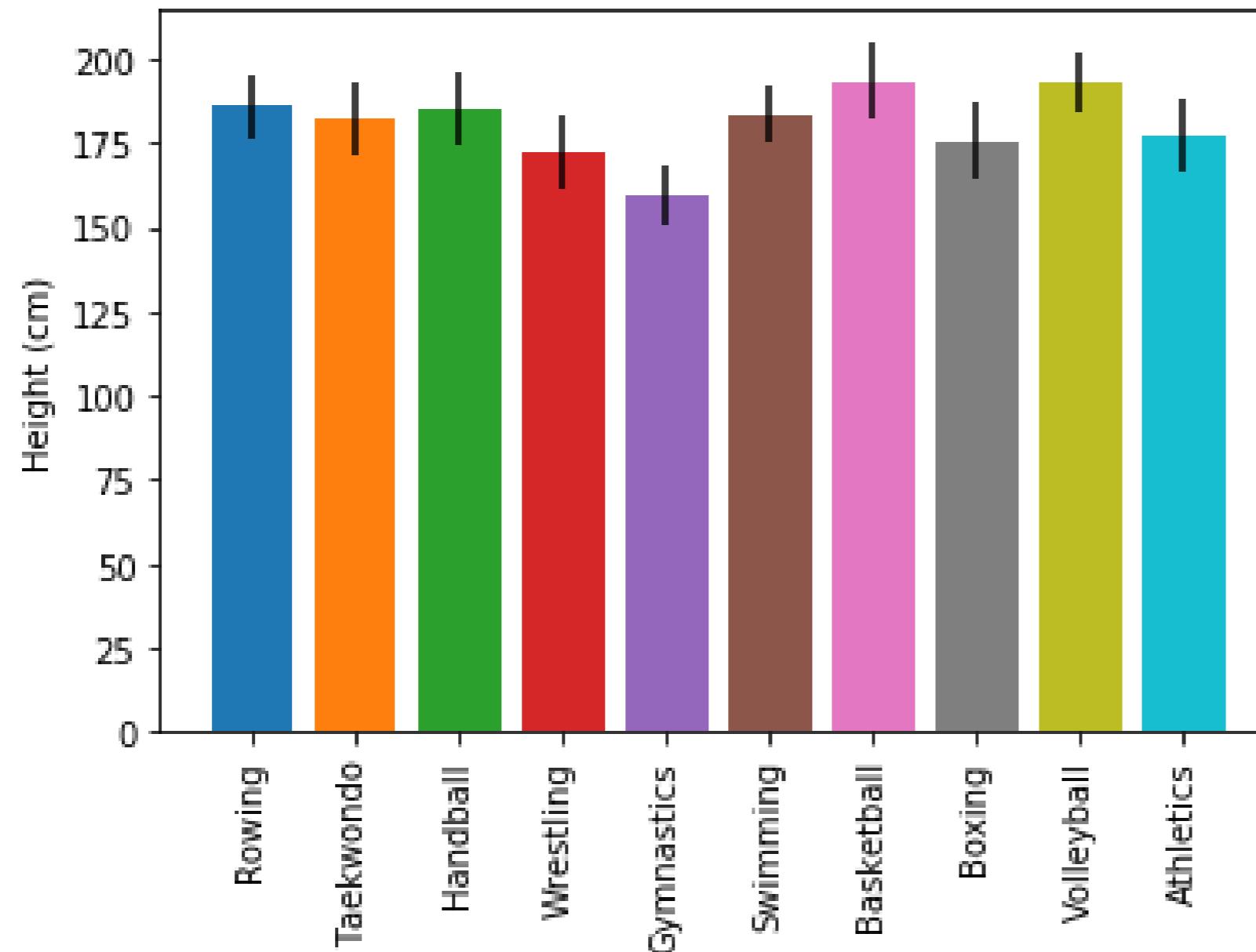
```
sports = summer_2016_medals["Sport"].unique()  
print(sports)  
['Rowing' 'Taekwondo' 'Handball' 'Wrestling'  
'Gymnastics' 'Swimming' 'Basketball' 'Boxing'  
'Volleyball' 'Athletics']
```

Bar-chart of heights for all sports

```
fig, ax = plt.subplots()

for sport in sports:
    sport_df = summer_2016_medals[summer_2016_medals["Sport"] == sport]
    ax.bar(sport, sport_df["Height"].mean(),
           yerr=sport_df["Height"].std())
ax.set_ylabel("Height (cm)")
ax.set_xticklabels(sports, rotation=90)
plt.show()
```

Figure derived automatically from the data

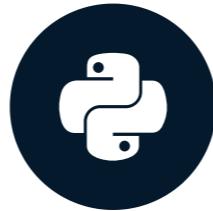


Practice automating visualizations!

INTRODUCTION TO DATA VISUALIZATION WITH MATPLOTLIB

Where to go next

INTRODUCTION TO DATA VISUALIZATION WITH MATPLOTLIB



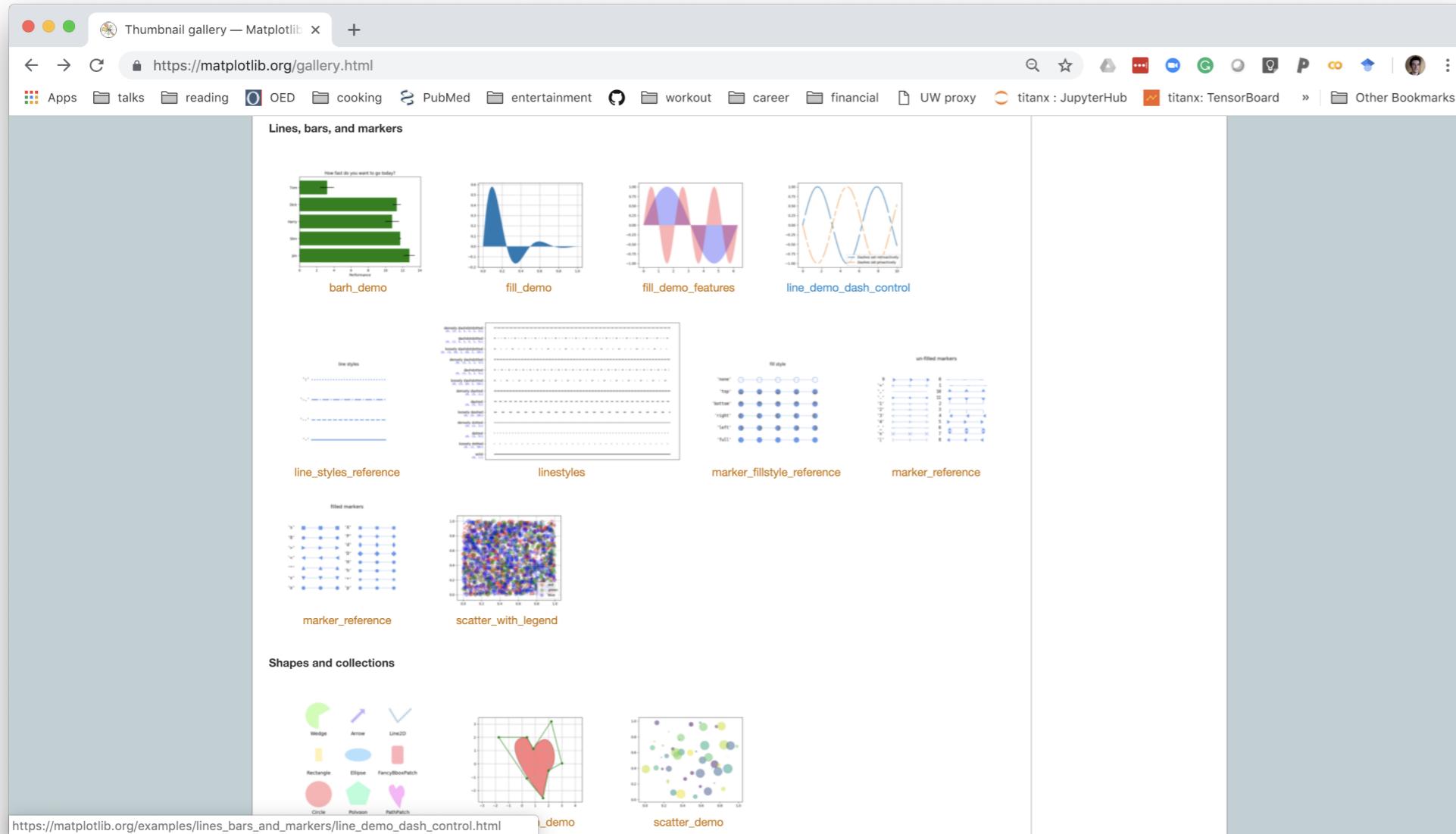
Ariel Rokem

Data Scientist

The Matplotlib gallery

<https://matplotlib.org/gallery.html>

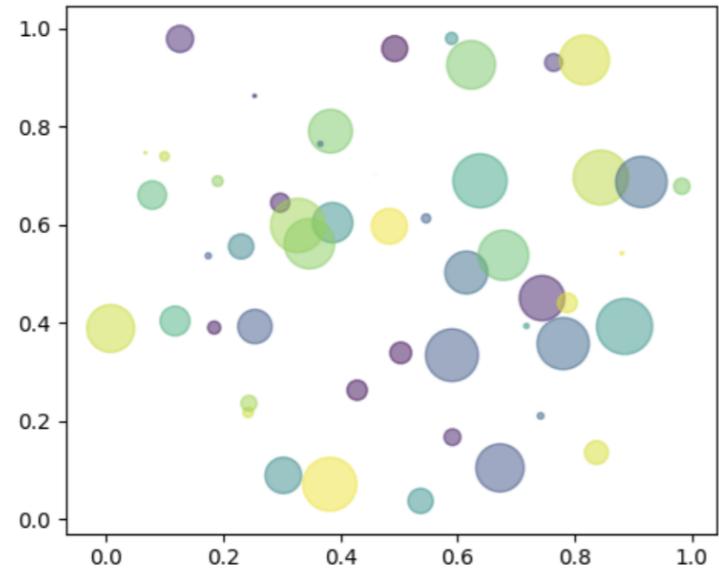
Gallery of examples



Example page with code

shapes_and_collections example code: scatter_demo.py

([Source code](#), [png](#), [pdf](#))



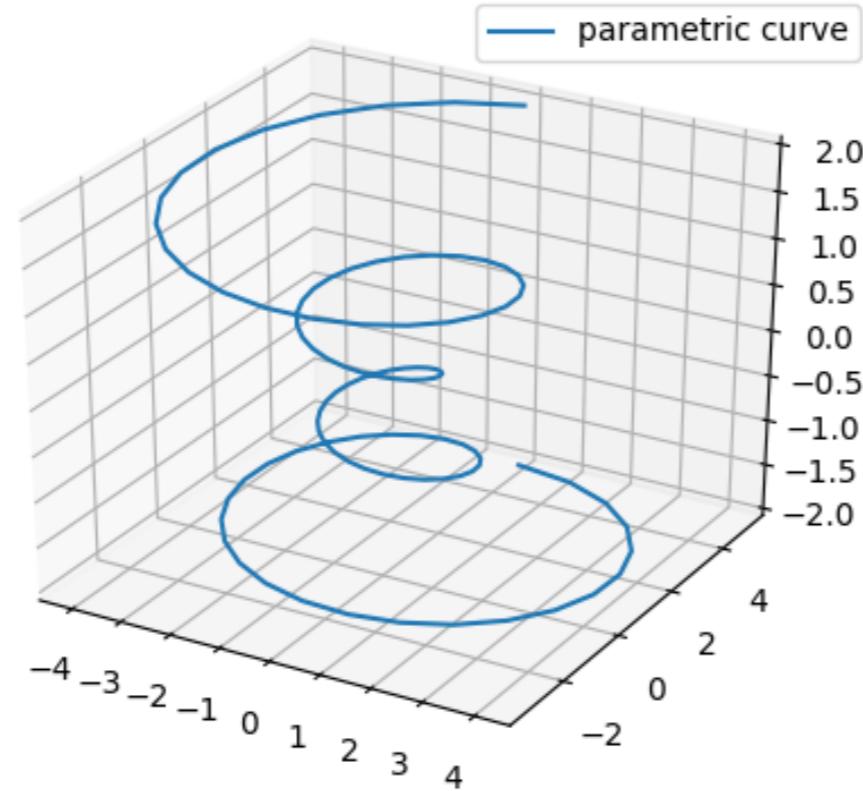
```
"""
Simple demo of a scatter plot.
"""

import numpy as np
import matplotlib.pyplot as plt

N = 50
x = np.random.rand(N)
y = np.random.rand(N)
colors = np.random.rand(N)
area = np.pi * (15 * np.random.rand(N))**2 # 0 to 15 point radii

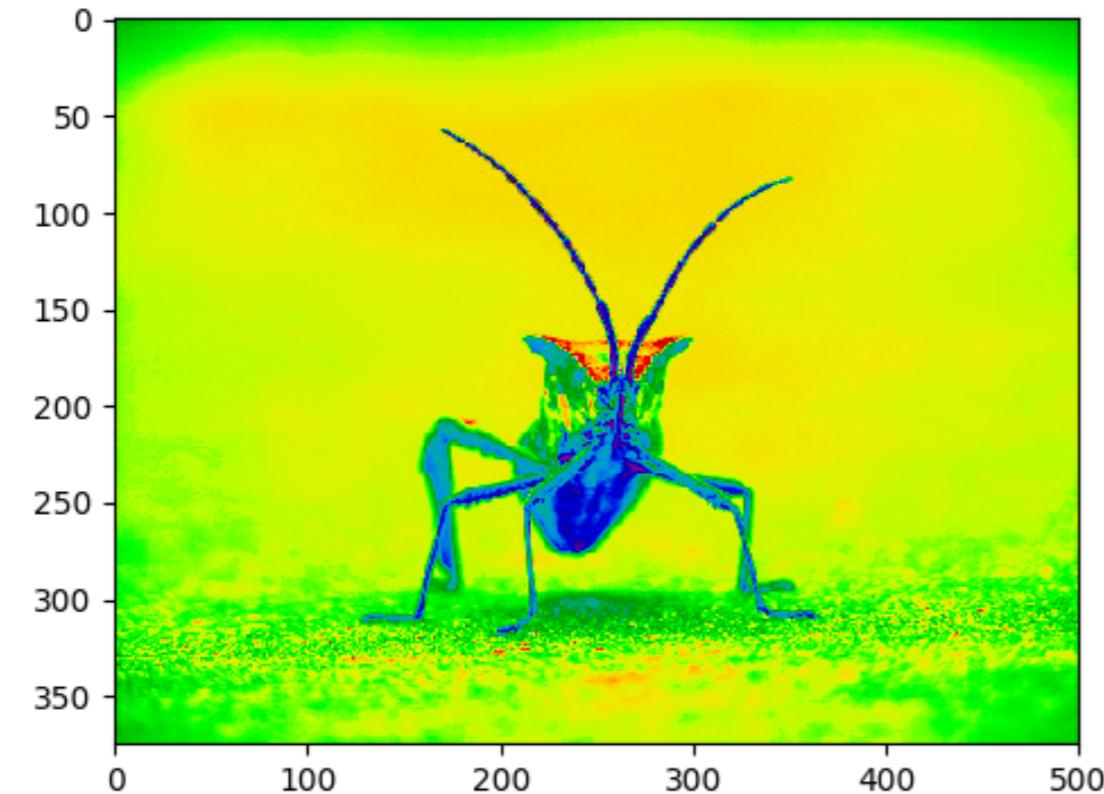
plt.scatter(x, y, s=area, c=colors, alpha=0.5)
plt.show()
```

Plotting data in 3D



https://matplotlib.org/mpl_toolkits/mplot3d/tutorial.html

Visualizing images with pseudo-color



https://matplotlib.org/users/image_tutorial.html

Animations

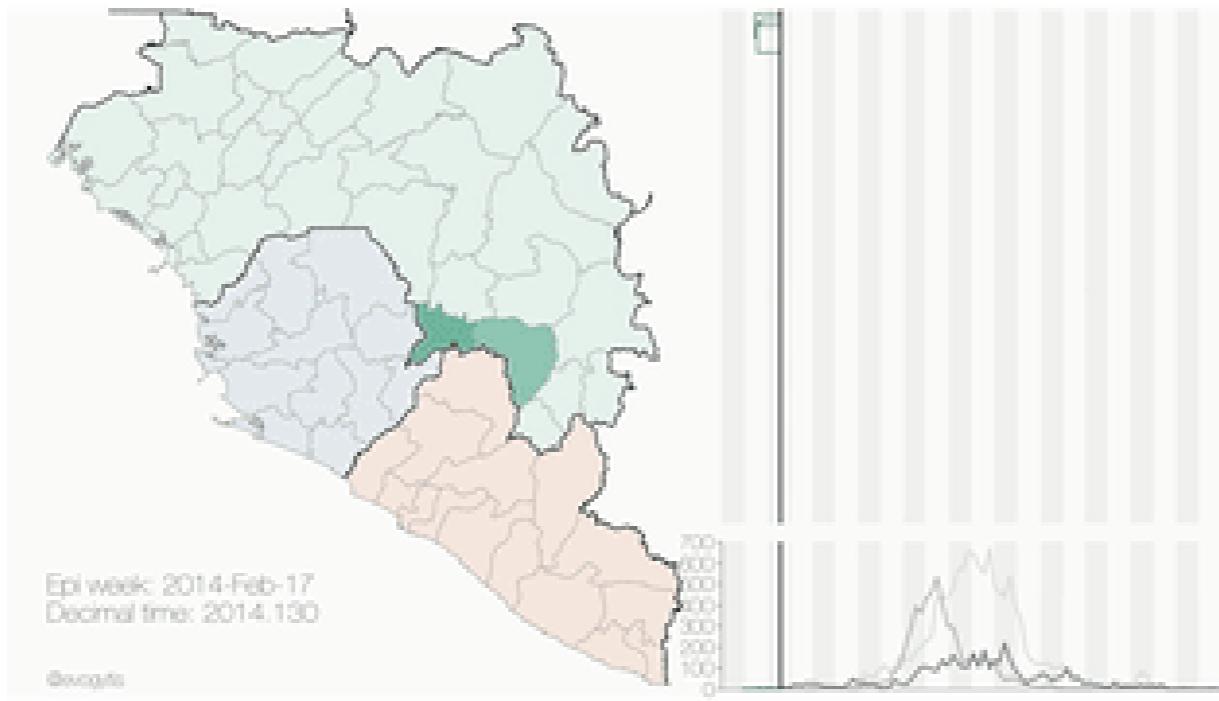
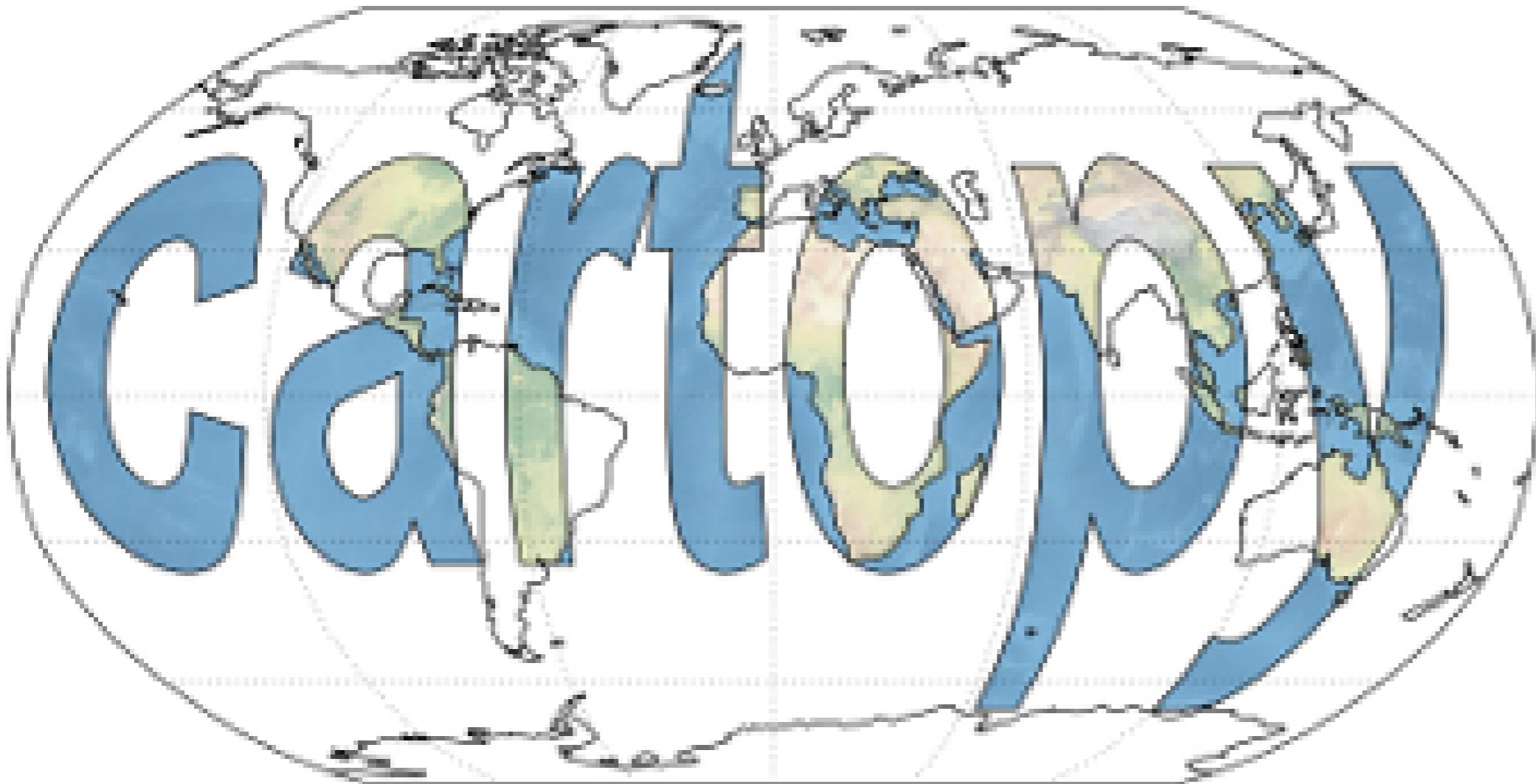


Image credit: [Gytis Dudas](#) and [Andrew Rambaut](#)

https://matplotlib.org/api/animation_api.html

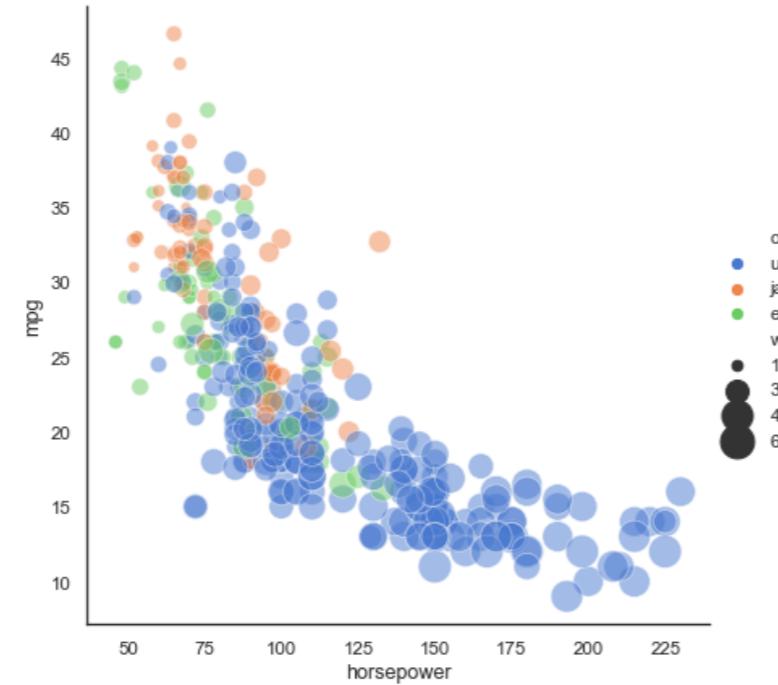
Using Matplotlib for geospatial data



<https://scitools.org.uk/cartopy/docs/latest/>

Pandas + Matplotlib = Seaborn

```
seaborn.relplot(x="horsepower", y="mpg", hue="origin", size="weight  
sizes=(40, 400), alpha=.5, palette="muted",  
height=6, data=mpg)
```



Seaborn example gallery

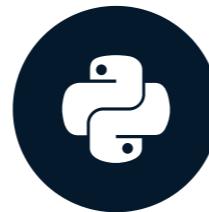
<https://seaborn.pydata.org/examples/index.html>

**Good luck
visualizing your
data!**

INTRODUCTION TO DATA VISUALIZATION WITH MATPLOTLIB

Introduction to Seaborn

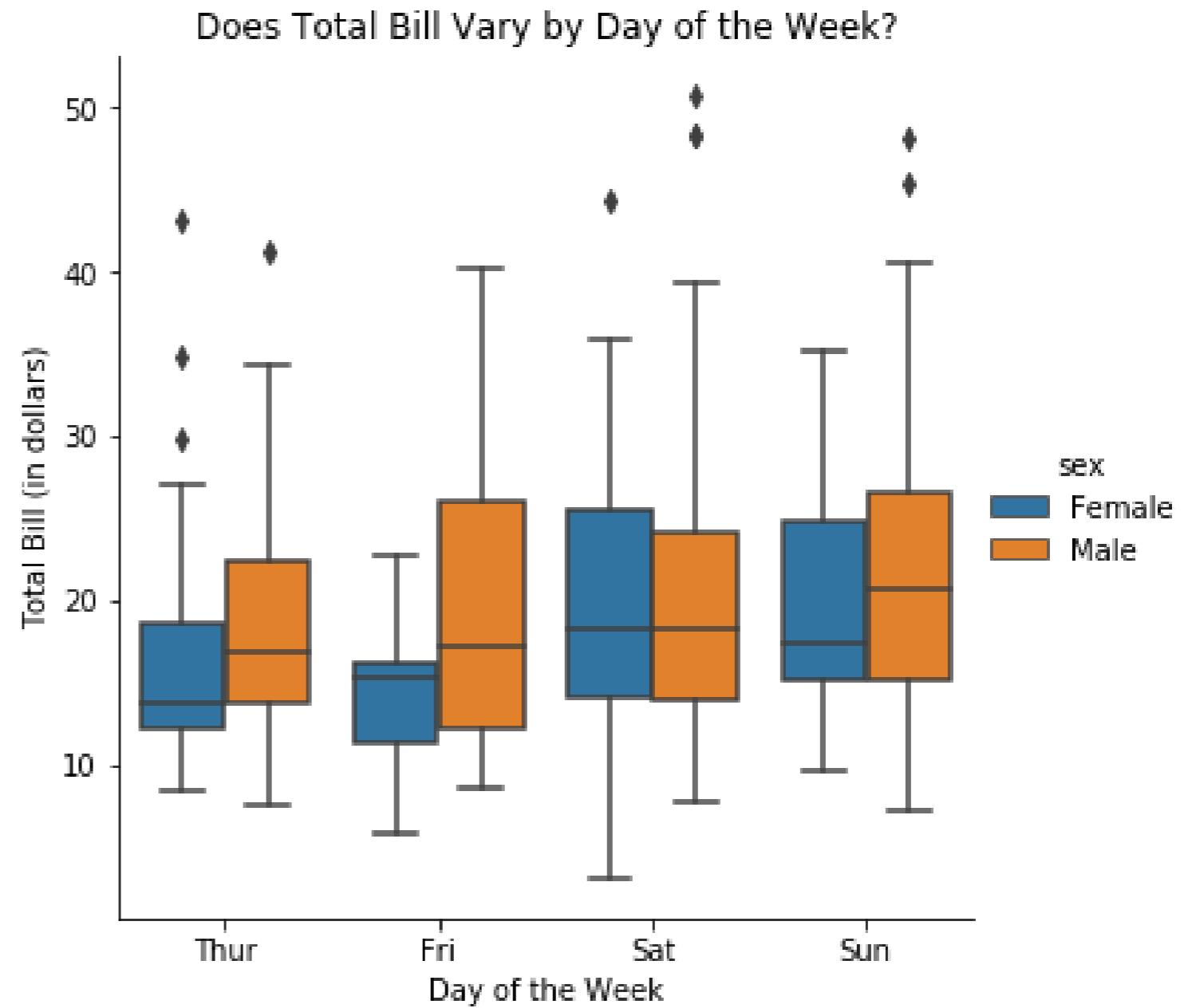
INTRODUCTION TO DATA VISUALIZATION WITH SEABORN



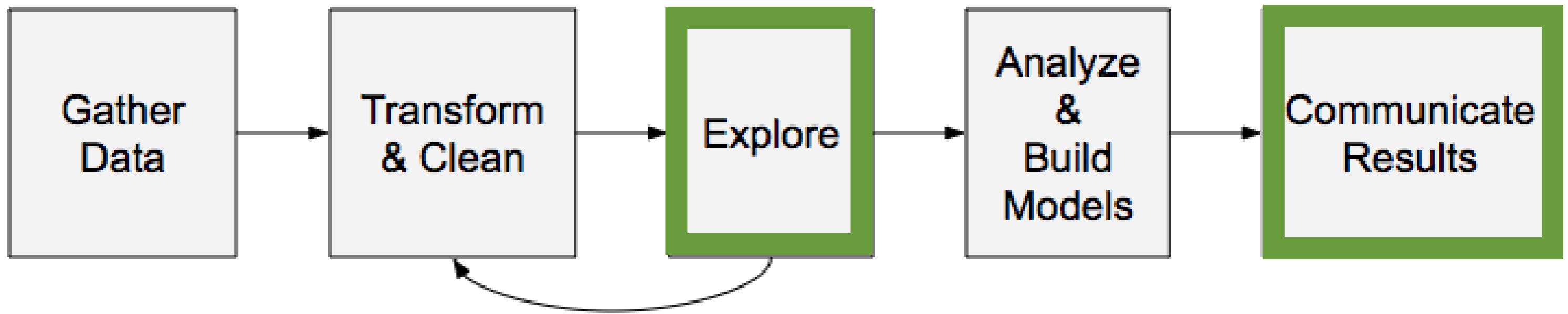
Erin Case
Data Scientist

What is Seaborn?

- Python data visualization library
- Easily create the most common types of plots



Why is Seaborn useful?



Advantages of Seaborn

- Easy to use
- Works well with `pandas` data structures
- Built on top of `matplotlib`

Getting started

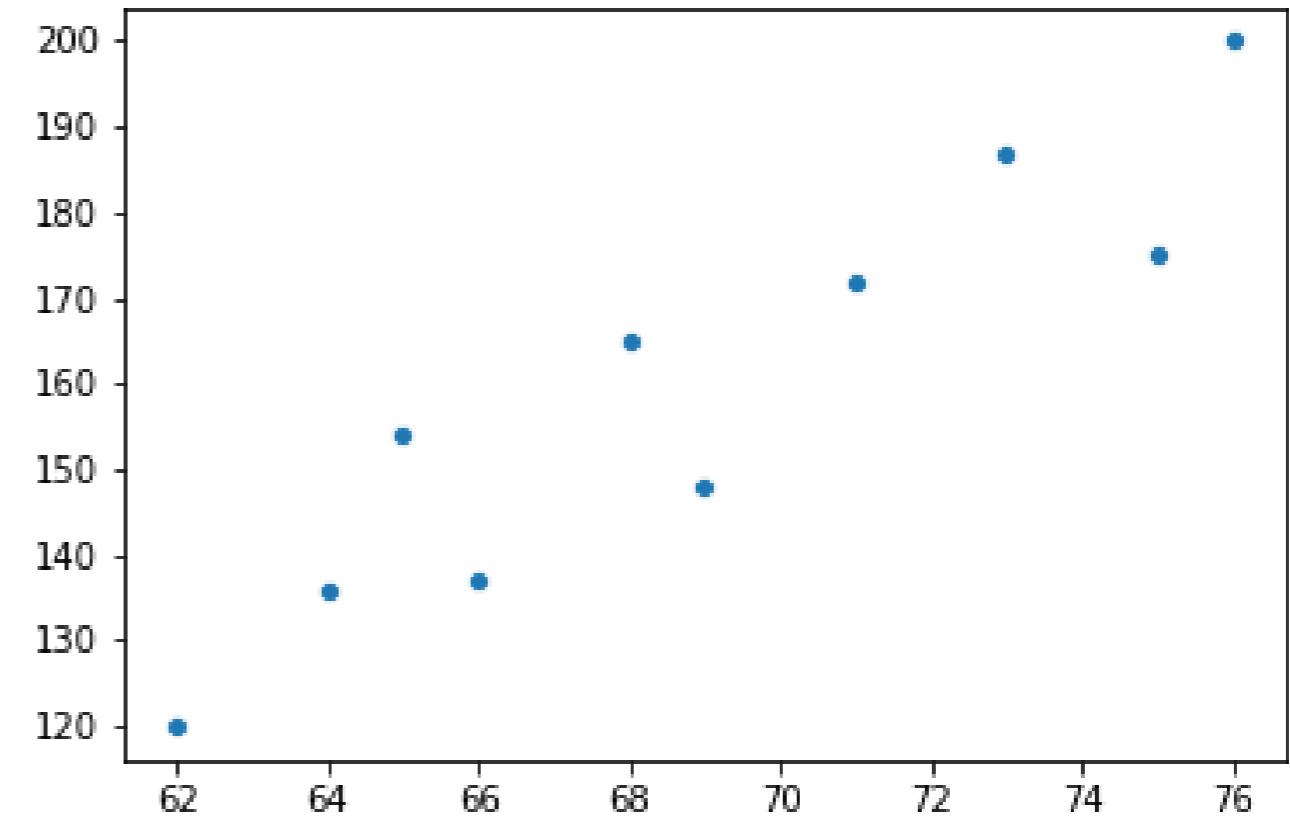
```
import seaborn as sns  
import matplotlib.pyplot as plt
```

Samuel Norman Seaborn (sns)

- "The West Wing" television show

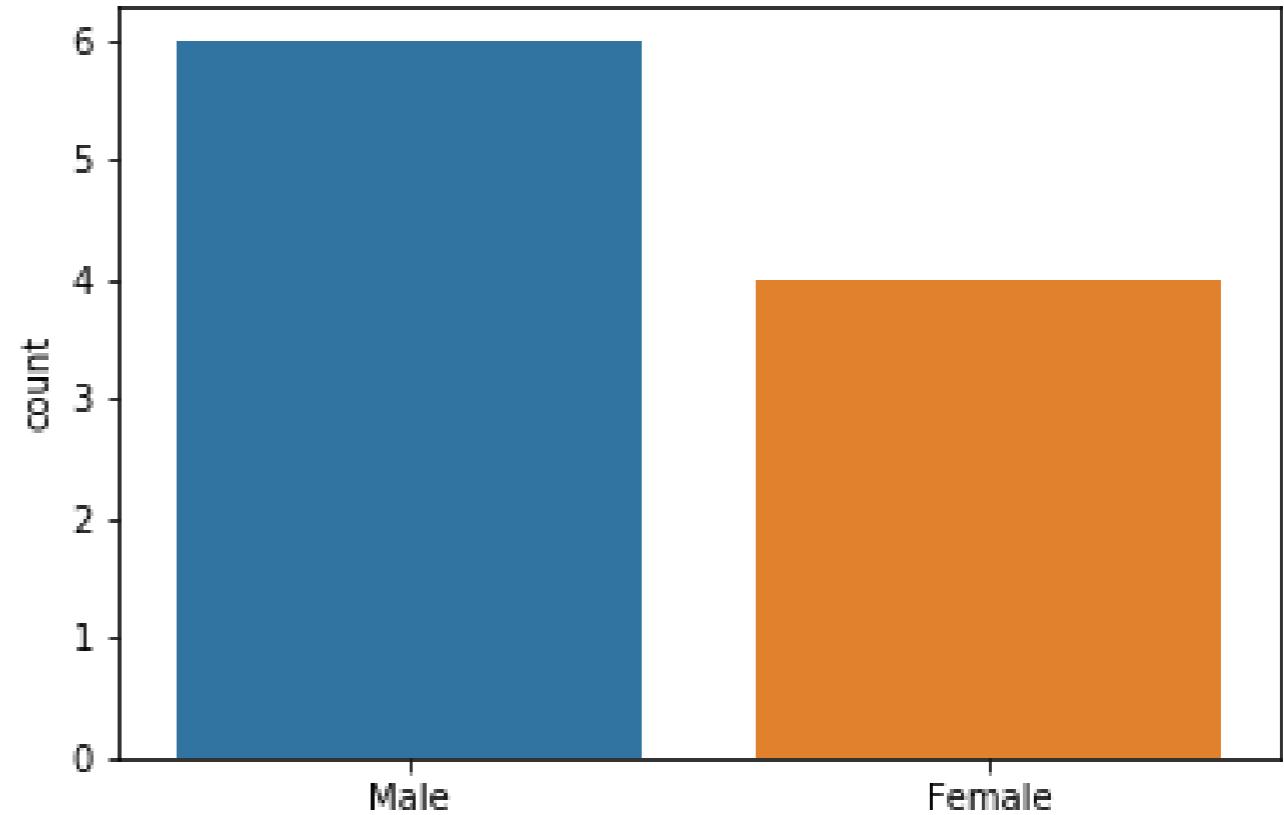
Example 1: Scatter plot

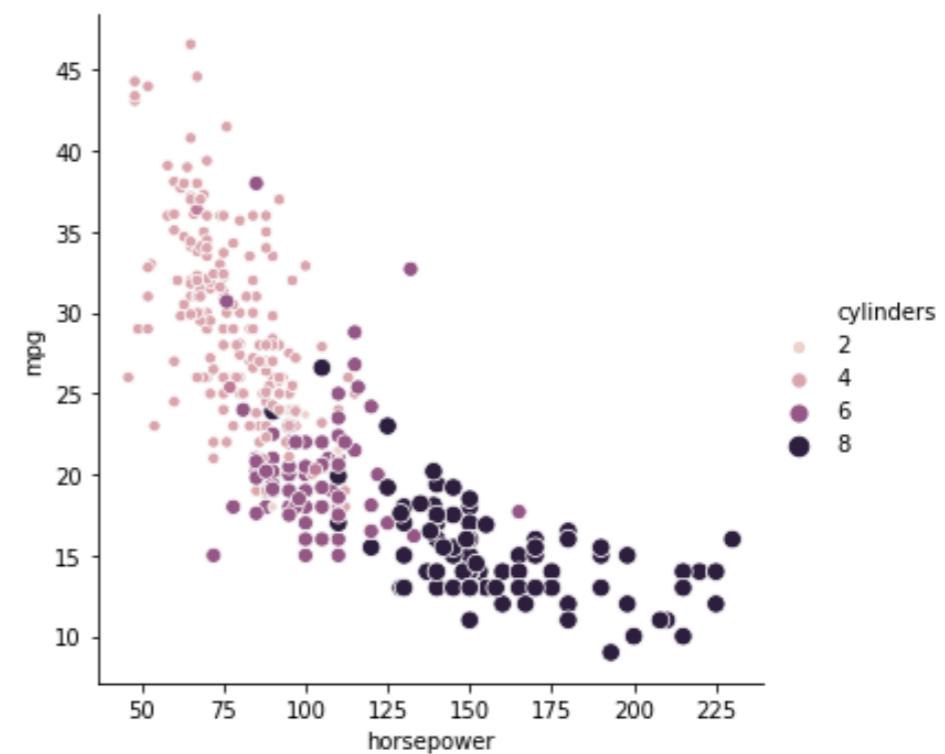
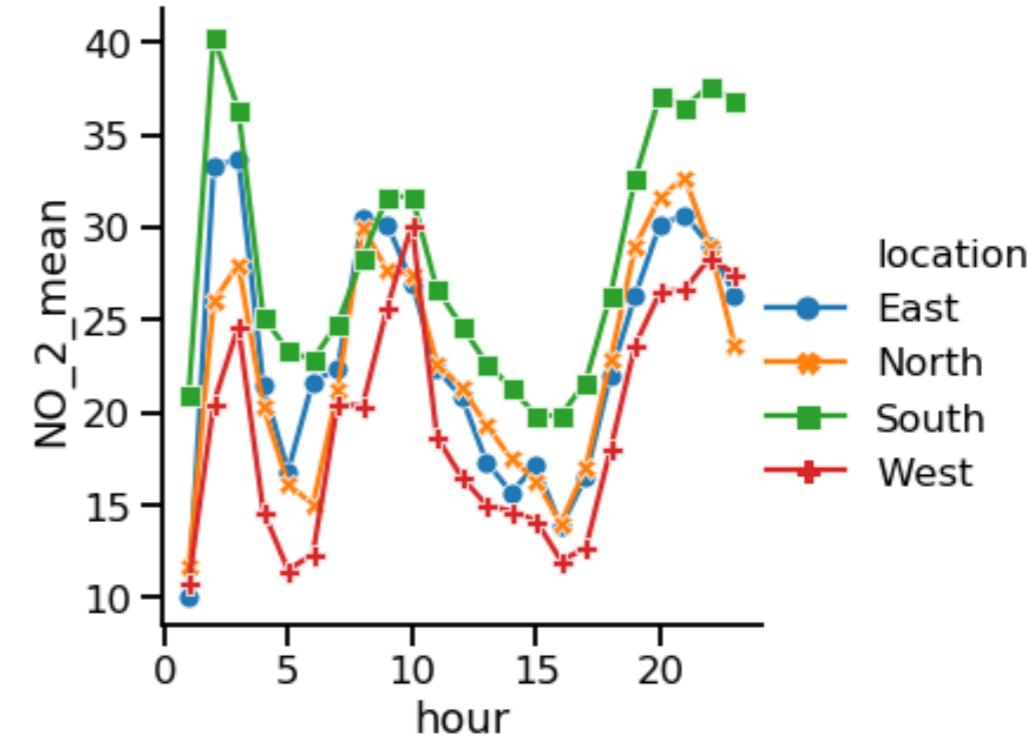
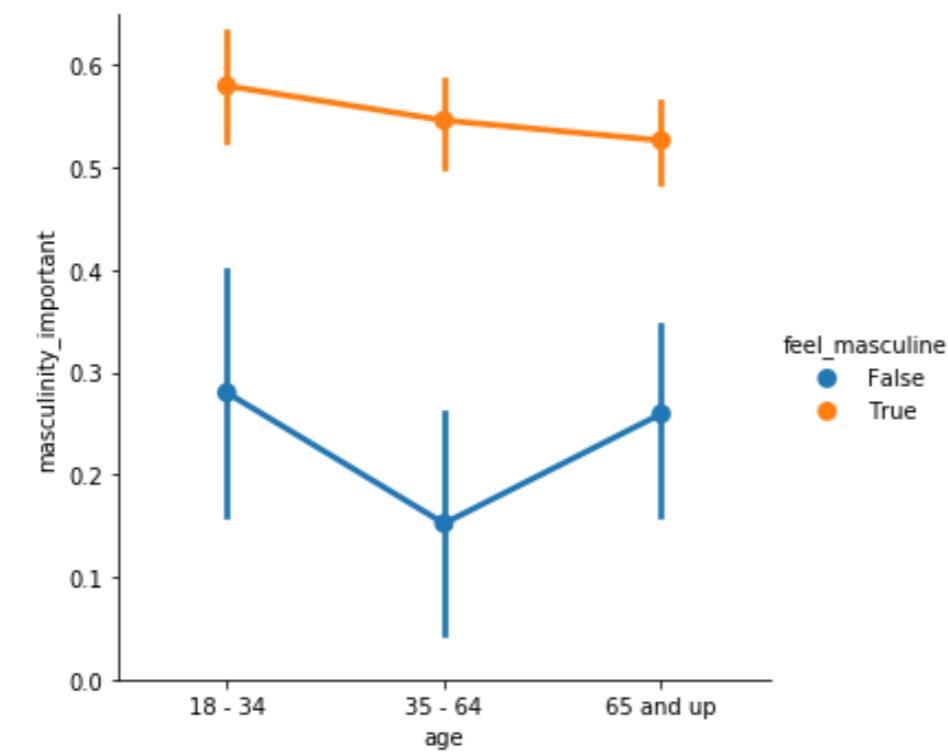
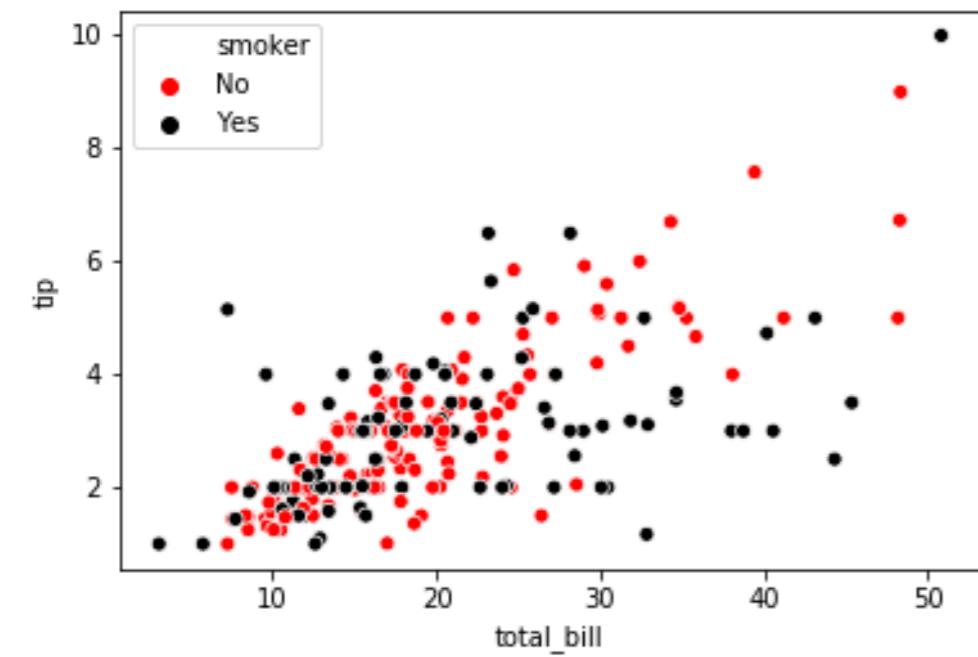
```
import seaborn as sns  
import matplotlib.pyplot as plt  
  
height = [62, 64, 69, 75, 66,  
          68, 65, 71, 76, 73]  
weight = [120, 136, 148, 175, 137,  
          165, 154, 172, 200, 187]  
  
sns.scatterplot(x=height, y=weight)  
plt.show()
```



Example 2: Create a count plot

```
import seaborn as sns  
import matplotlib.pyplot as plt  
  
gender = ["Female", "Female",  
          "Female", "Female",  
          "Male", "Male", "Male",  
          "Male", "Male", "Male"]  
  
sns.countplot(x=gender)  
plt.show()
```



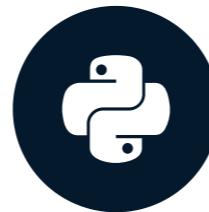


Let's practice!

INTRODUCTION TO DATA VISUALIZATION WITH SEABORN

Using pandas with Seaborn

INTRODUCTION TO DATA VISUALIZATION WITH SEABORN



Erin Case
Data Scientist

What is pandas?

- Python library for data analysis
- Easily read datasets from csv, txt, and other types of files
- Datasets take the form of `DataFrame` objects

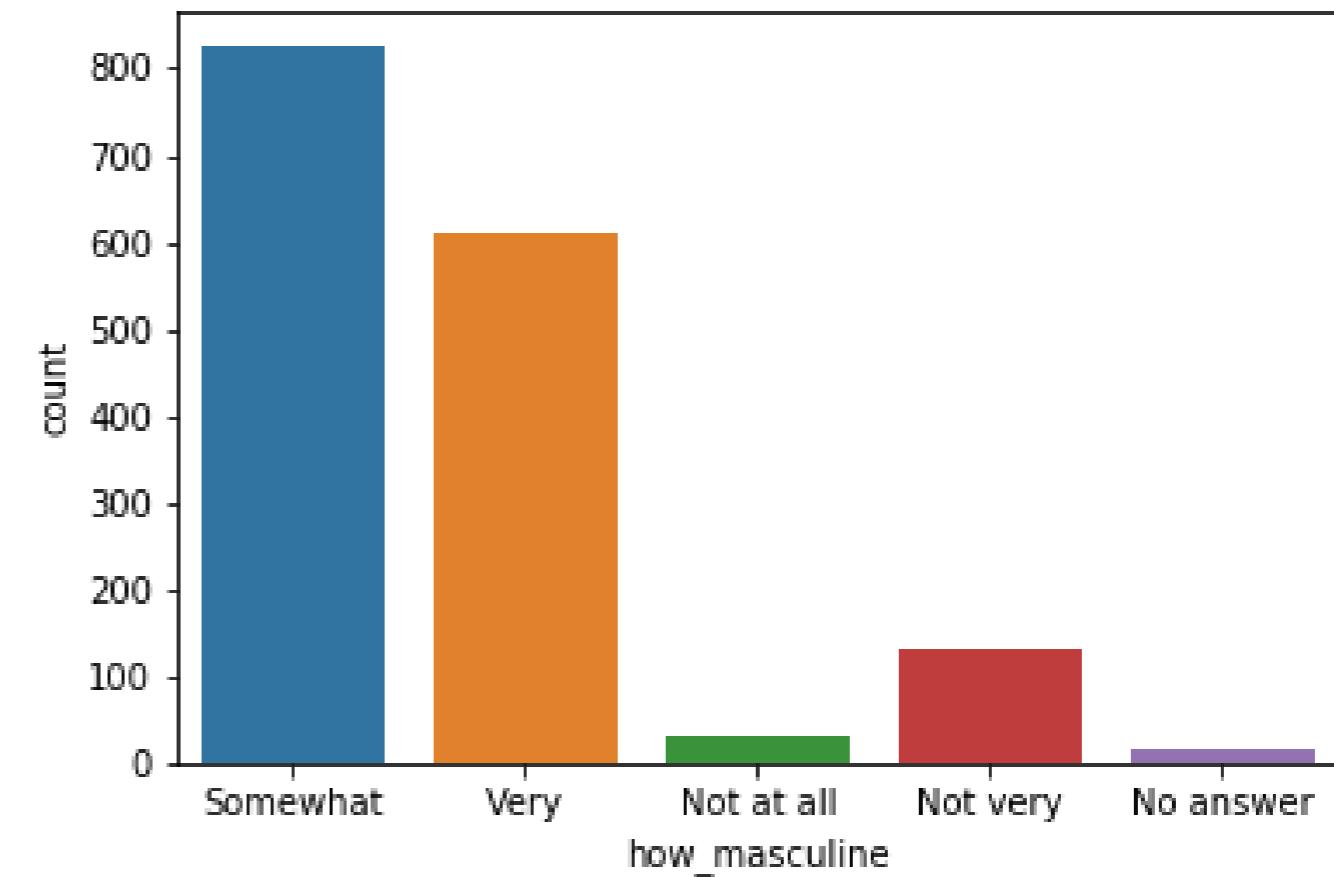
Working with DataFrames

```
import pandas as pd  
df = pd.read_csv("masculinity.csv")  
df.head()
```

	participant_id	age	how_masculine	how_important
0	1	18 – 34	Somewhat	Somewhat
1	2	18 – 34	Somewhat	Somewhat
2	3	18 – 34	Very	Not very
3	4	18 – 34	Very	Not very
4	5	18 – 34	Very	Very

Using DataFrames with countplot()

```
import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns  
  
df = pd.read_csv("mascularity.csv")  
sns.countplot(x="how_masculine",  
               data=df)  
  
plt.show()
```



	participant_id	age	how_masculine	how_important
0	1	18 - 34	Somewhat	Somewhat
1	2	18 - 34	Somewhat	Somewhat
2	3	18 - 34	Very	Not very
3	4	18 - 34	Very	Not very
4	5	18 - 34	Very	Very
5	6	18 - 34	Very	Somewhat
6	7	18 - 34	Somewhat	Not very
7	8	18 - 34	Somewhat	Somewhat
8	9	18 - 34	Very	Not at all
9	10	18 - 34	Somewhat	Somewhat

	AMONG ADULT MEN	Unnamed: 1	Adult Men	Age	Unnamed: 4	Unnamed: 5
0				18 - 34	35 - 64	65 and up
1	In general, how masculine or "manly" do you feel?					
2		Very masculine	37%	29%	42%	37%
3		Somewhat masculine	46%	47%	46%	47%
4		Not very masculine	11%	13%	9%	13%
5		Not at all masculine	5%	10%	2%	3%
6		No answer	1%	0%	1%	1%
7	How important is it to you that others see you as masculine?					
8		Very important	16%	18%	17%	13%
9		Somewhat important	37%	38%	37%	32%
10		Not too important	28%	18%	31%	37%
11		Not at all important	18%	26%	15%	18%
12		No answer	0%	0%	1%	0%

Let's practice!

INTRODUCTION TO DATA VISUALIZATION WITH SEABORN

Adding a third variable with hue

INTRODUCTION TO DATA VISUALIZATION WITH SEABORN



Erin Case
Data Scientist

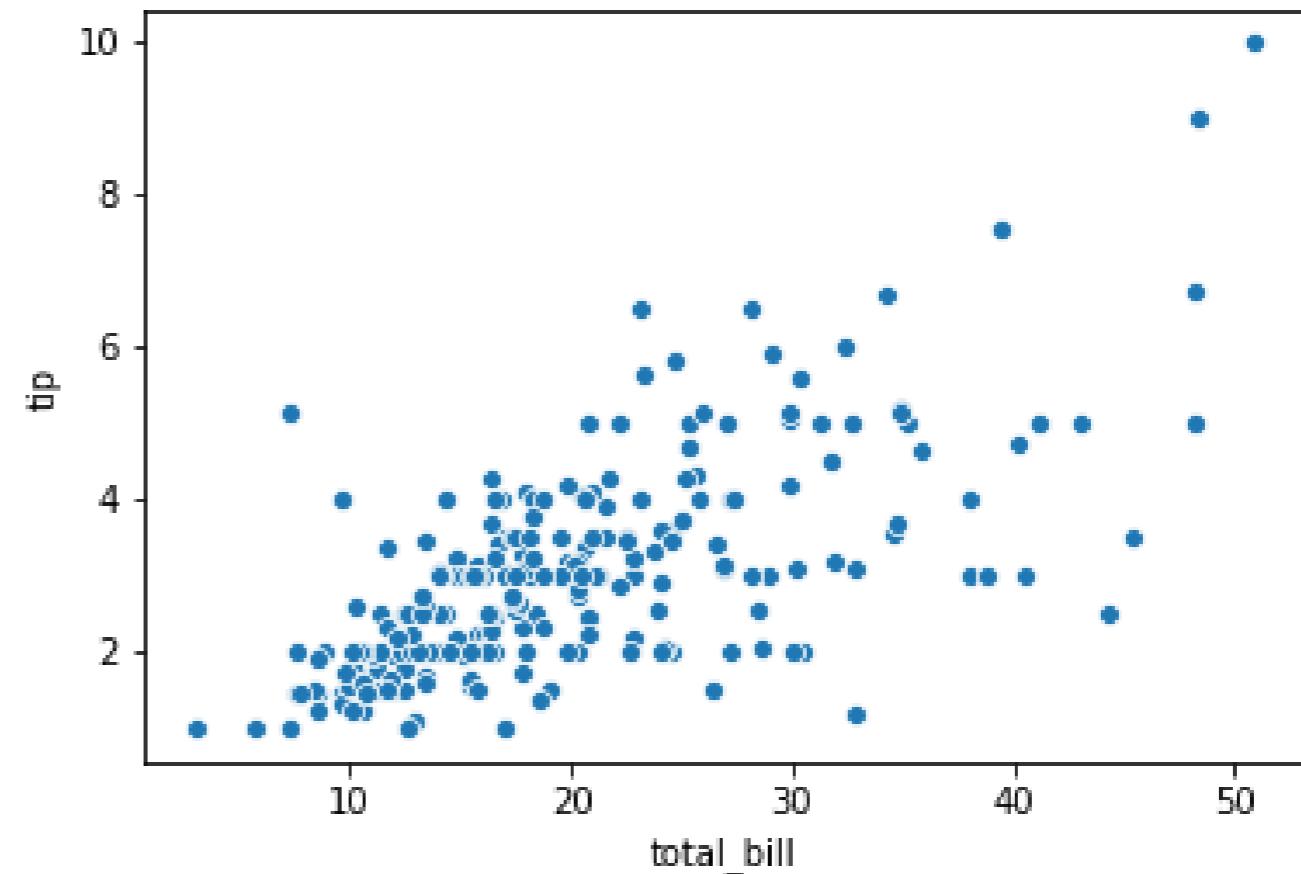
Tips dataset

```
import pandas as pd  
import seaborn as sns  
tips = sns.load_dataset("tips")  
tips.head()
```

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

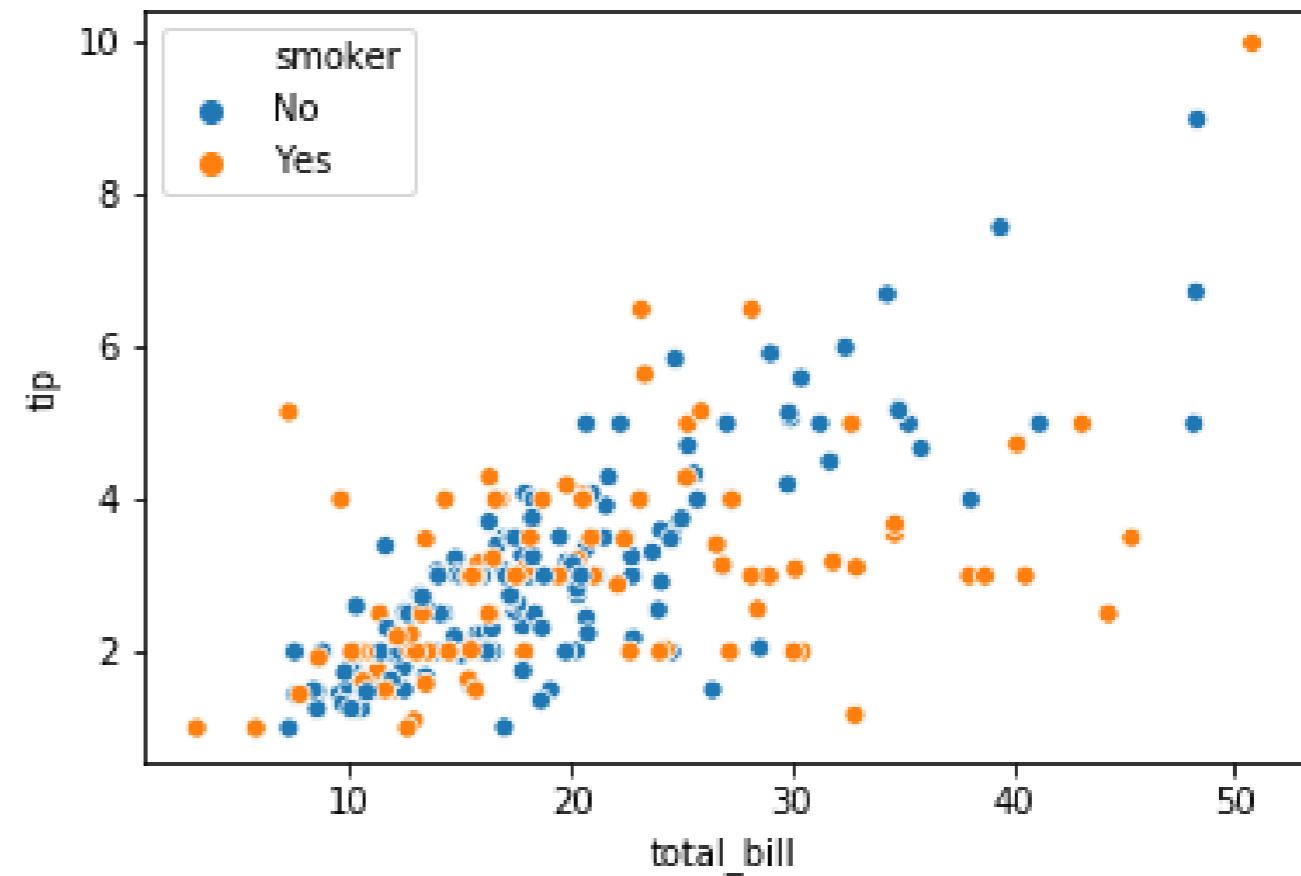
A basic scatter plot

```
import matplotlib.pyplot as plt  
import seaborn as sns  
  
sns.scatterplot(x="total_bill",  
                 y="tip",  
                 data=tips)  
  
plt.show()
```



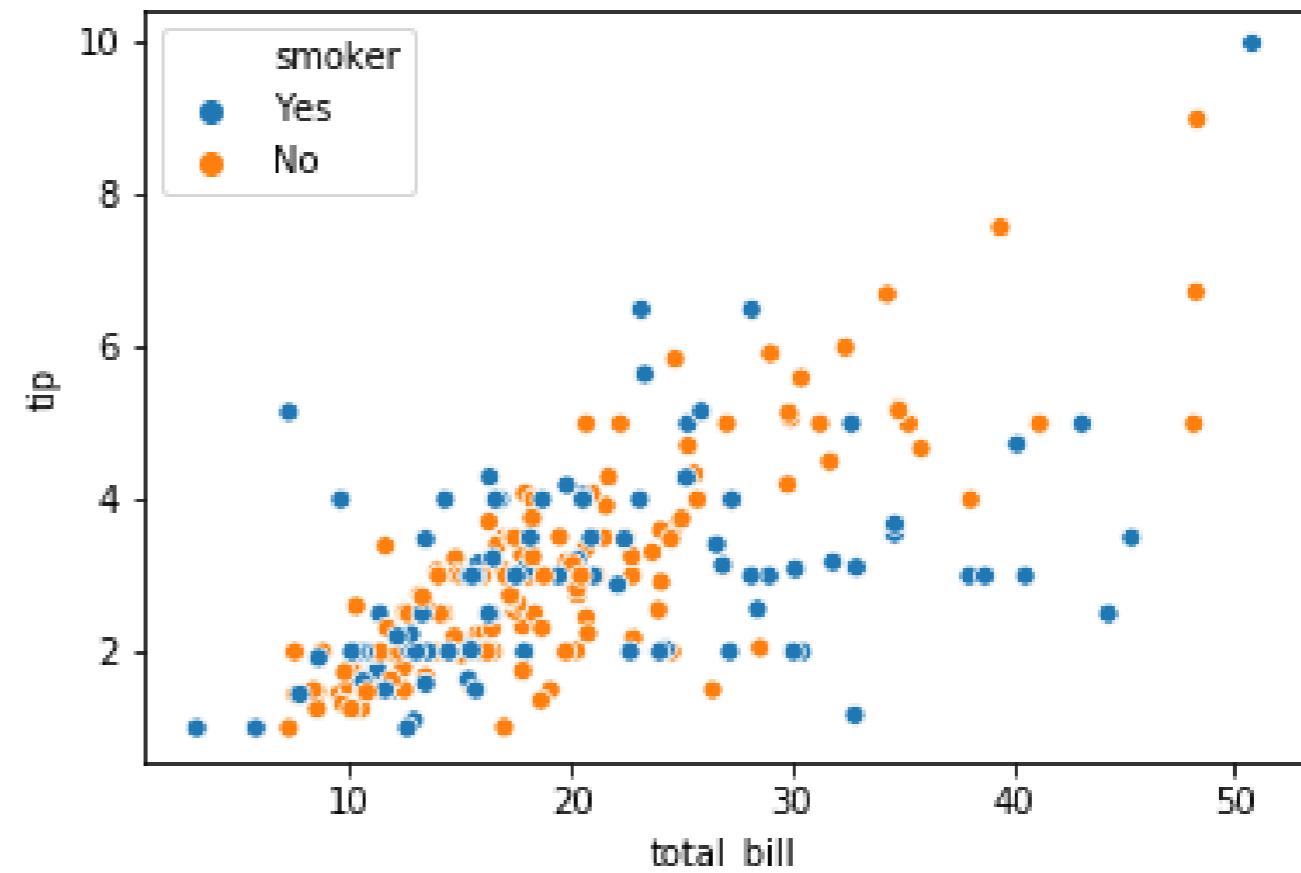
A scatter plot with hue

```
import matplotlib.pyplot as plt  
import seaborn as sns  
  
sns.scatterplot(x="total_bill",  
                 y="tip",  
                 data=tips,  
                 hue="smoker")  
  
plt.show()
```



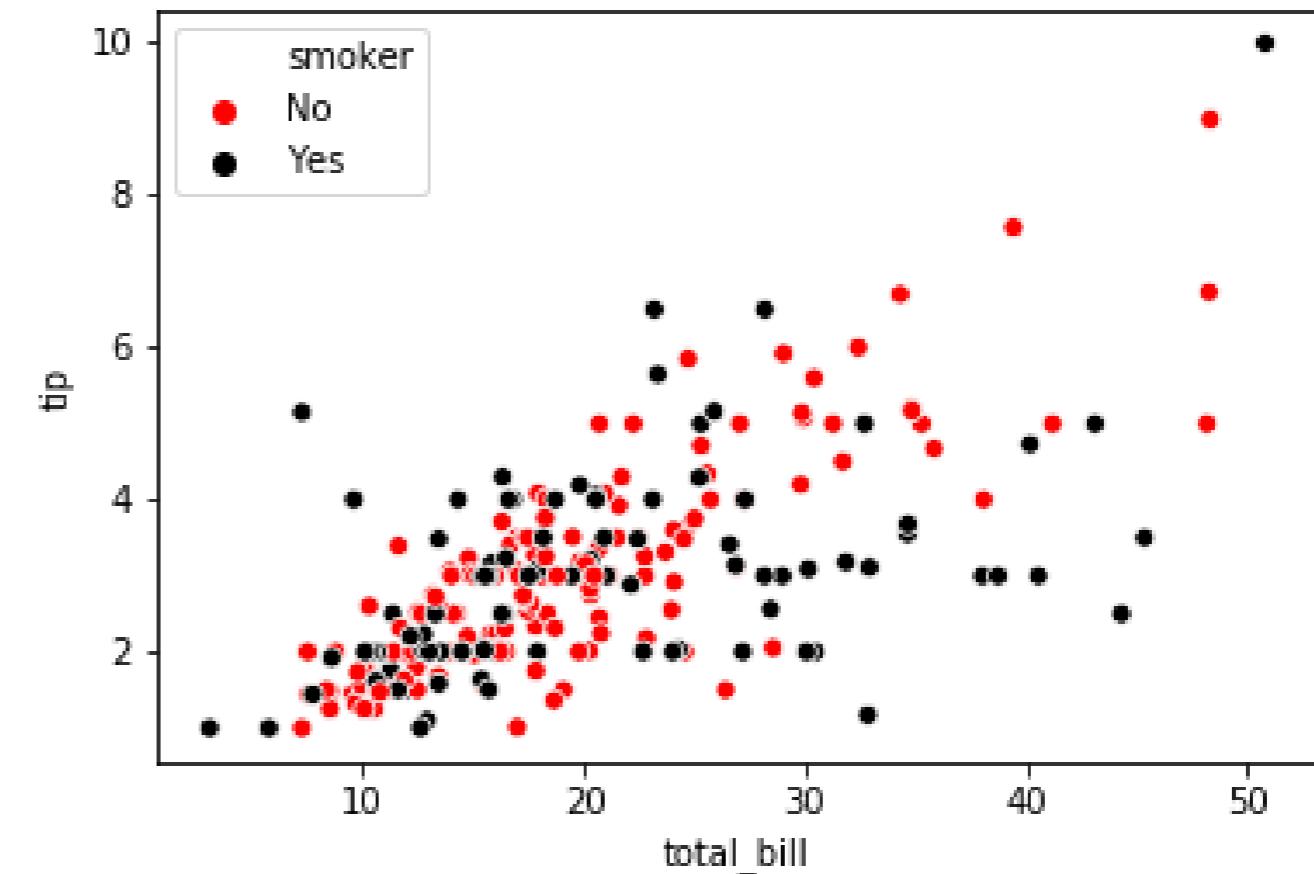
Setting hue order

```
import matplotlib.pyplot as plt  
import seaborn as sns  
  
sns.scatterplot(x="total_bill",  
                 y="tip",  
                 data=tips,  
                 hue="smoker",  
                 hue_order=["Yes",  
                            "No"])  
  
plt.show()
```



Specifying hue colors

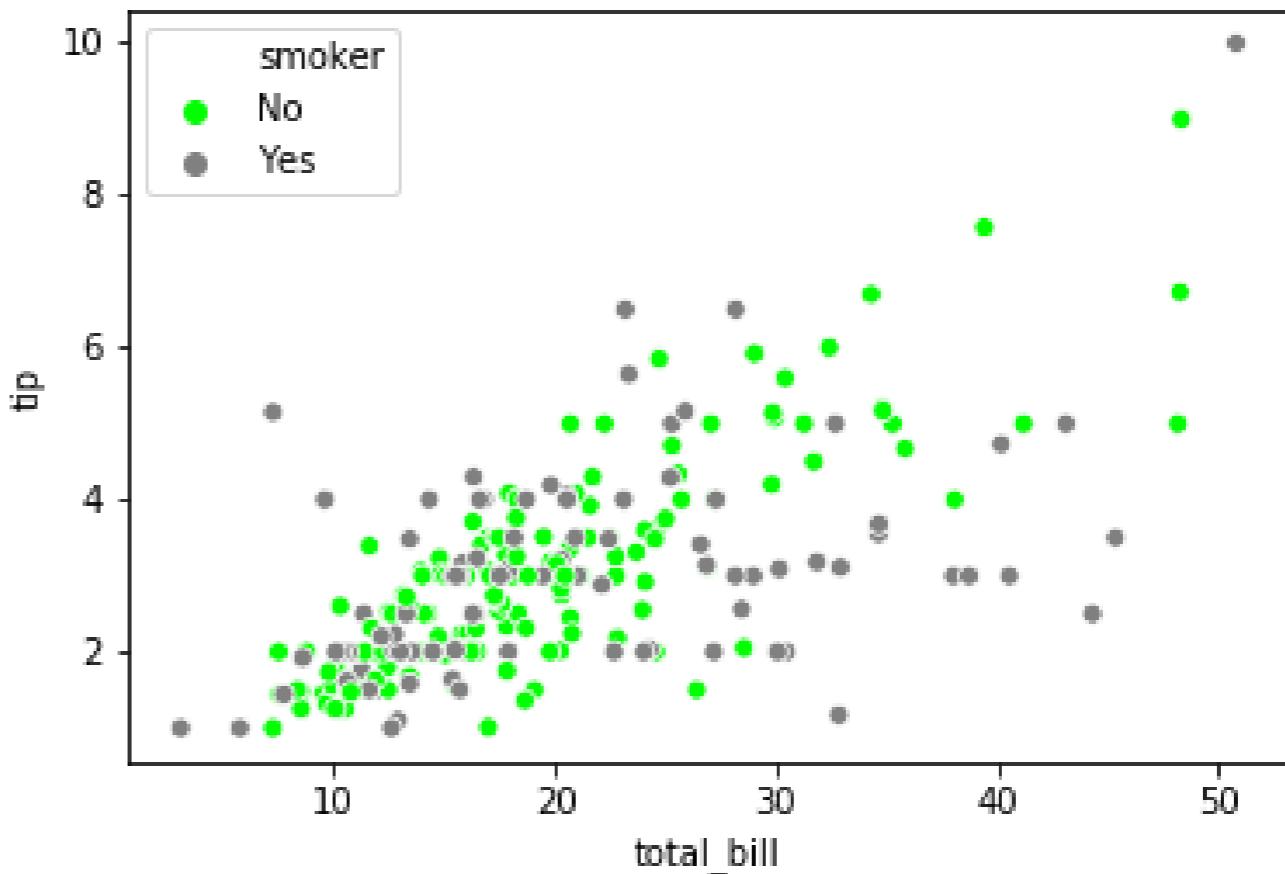
```
import matplotlib.pyplot as plt  
import seaborn as sns  
  
hue_colors = {"Yes": "black",  
              "No": "red"}  
  
sns.scatterplot(x="total_bill",  
                 y="tip",  
                 data=tips,  
                 hue="smoker",  
                 palette=hue_colors)  
  
plt.show()
```



	Color	Matplotlib name	Matplotlib abbreviation	HTML color code (hex)
	blue	"blue"	"b"	#0000ff
	green	"green"	"g"	#008000
	red	"red"	"r"	#ff0000
	green/blue	"cyan"	"c"	#00bfbf
	purple	"magenta"	"m"	#bf00bf
	yellow	"yellow"	"y"	#bfbf00
	black	"black"	"k"	#000000
	white	"white"	"w"	#ffffff

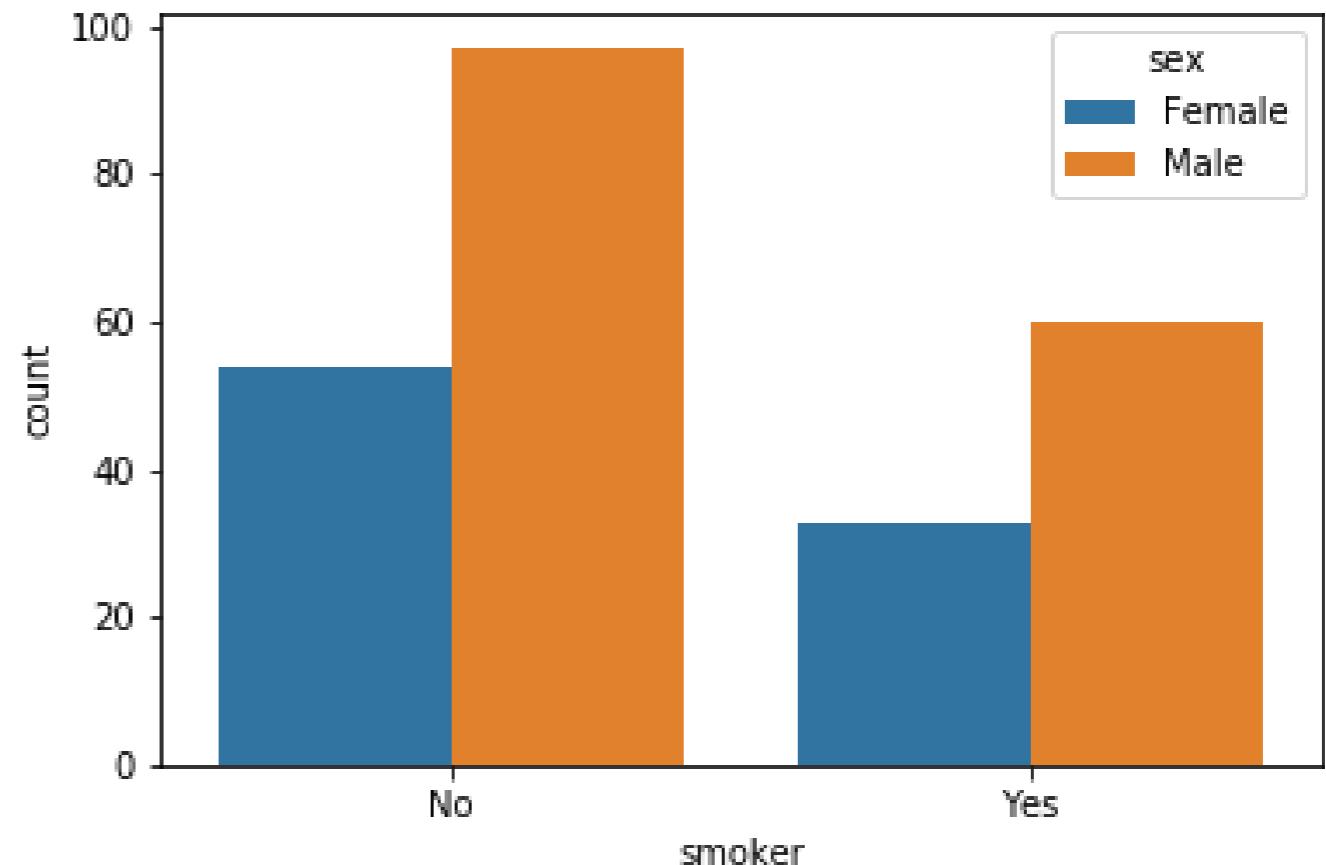
Using HTML hex color codes with hue

```
import matplotlib.pyplot as plt  
import seaborn as sns  
  
hue_colors = {"Yes": "#808080",  
              "No": "#00FF00"}  
  
sns.scatterplot(x="total_bill",  
                 y="tip",  
                 data=tips,  
                 hue="smoker",  
                 palette=hue_colors)  
  
plt.show()
```



Using hue with count plots

```
import matplotlib.pyplot as plt  
import seaborn as sns  
  
sns.countplot(x="smoker",  
               data=tips,  
               hue="sex")  
  
plt.show()
```



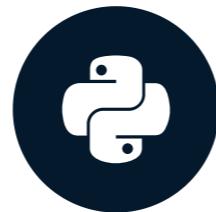
Let's practice!

INTRODUCTION TO DATA VISUALIZATION WITH SEABORN

Introduction to relational plots and subplots

INTRODUCTION TO DATA VISUALIZATION WITH SEABORN

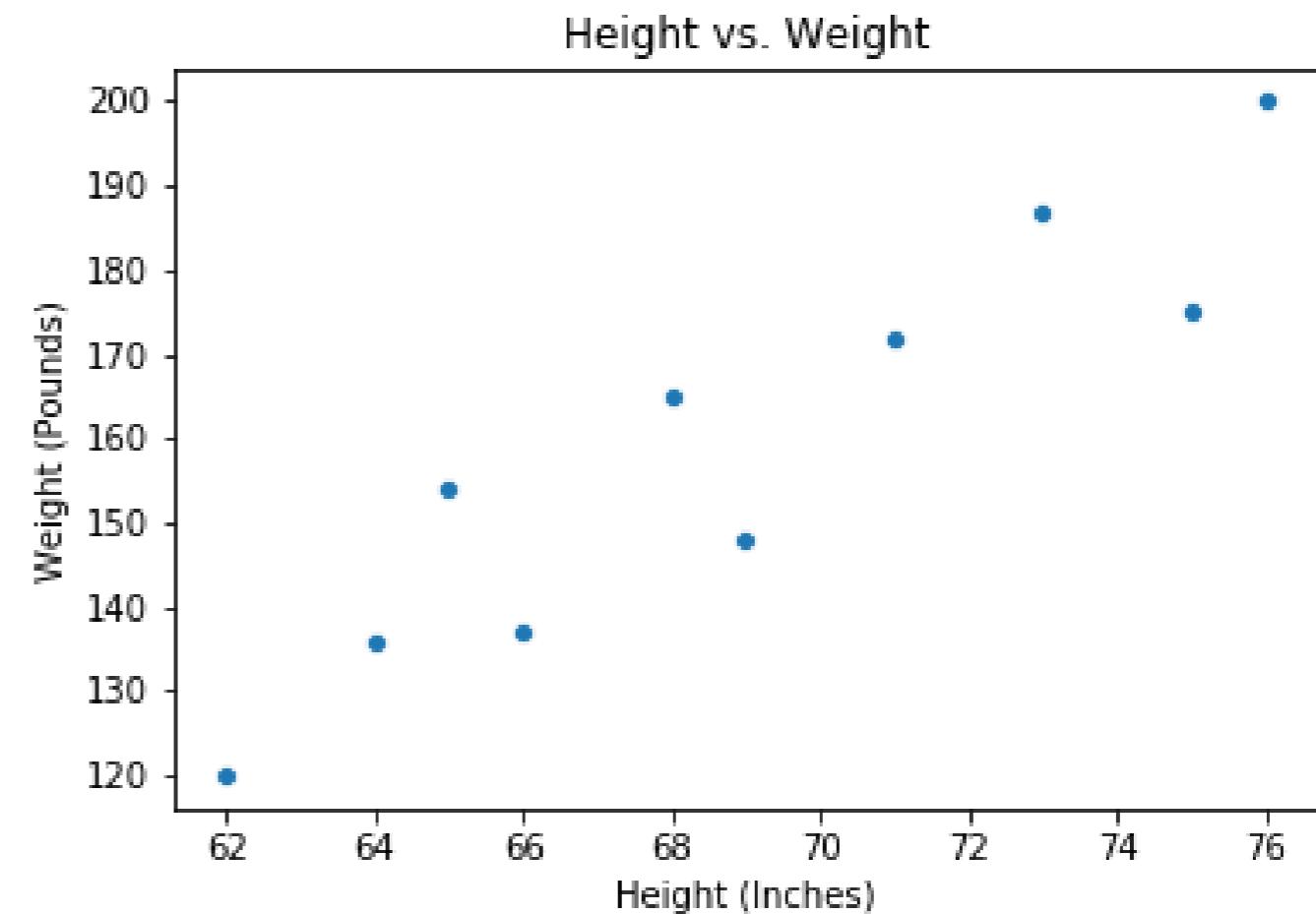
Erin Case
Data Scientist



Questions about quantitative variables

Relational plots

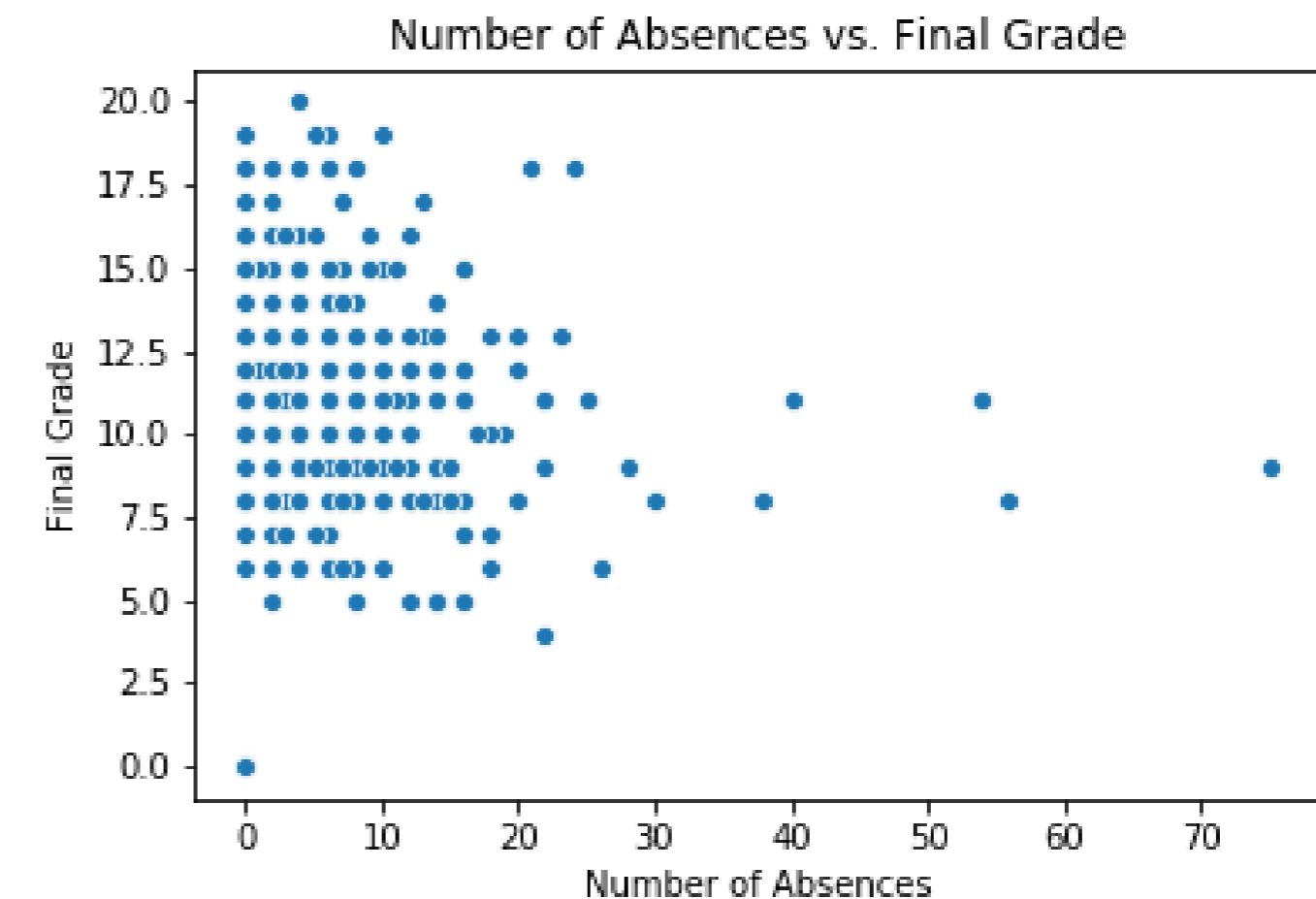
- Height vs. weight



Questions about quantitative variables

Relational plots

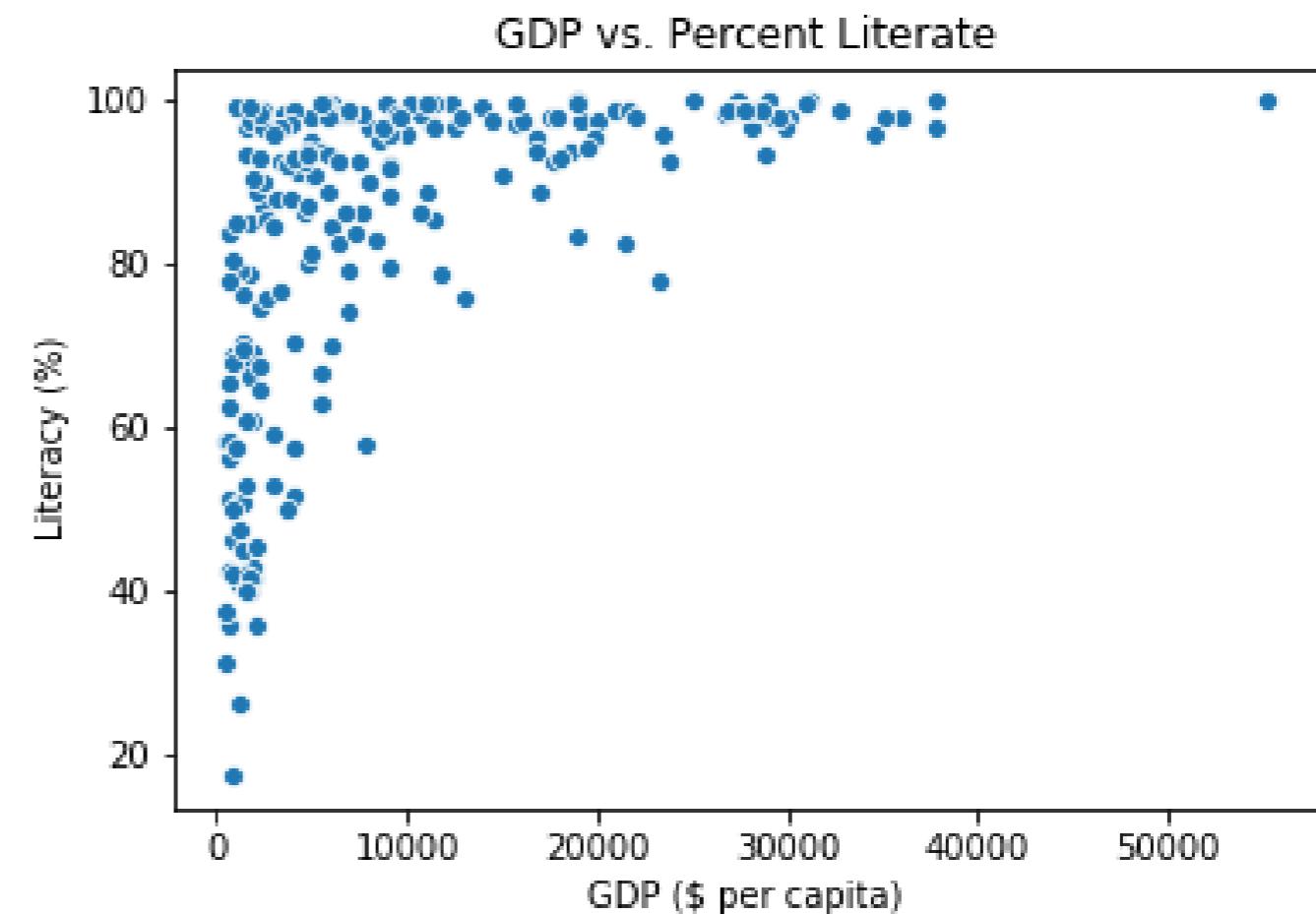
- Height vs. weight
- Number of school absences vs. final grade

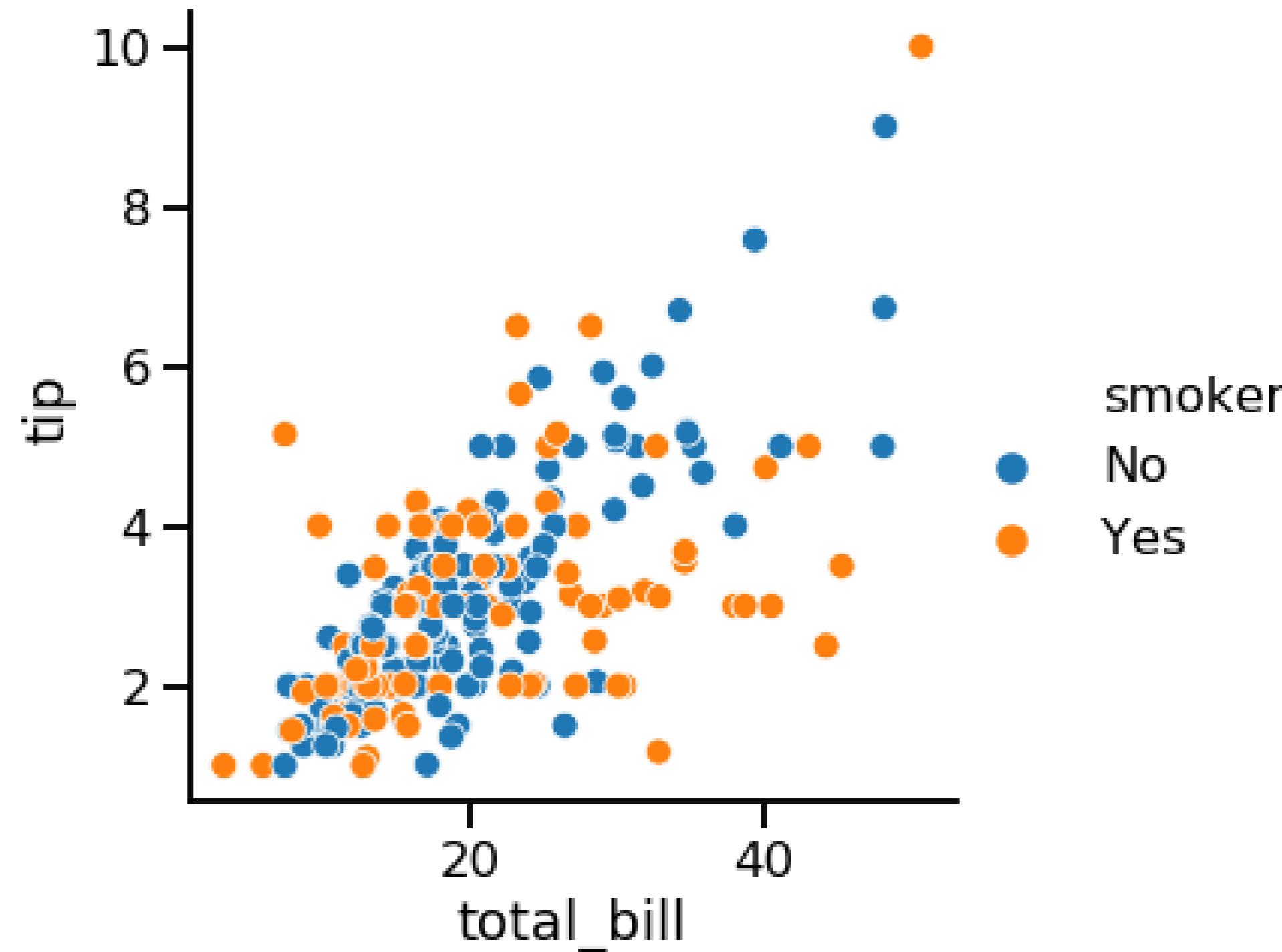


Questions about quantitative variables

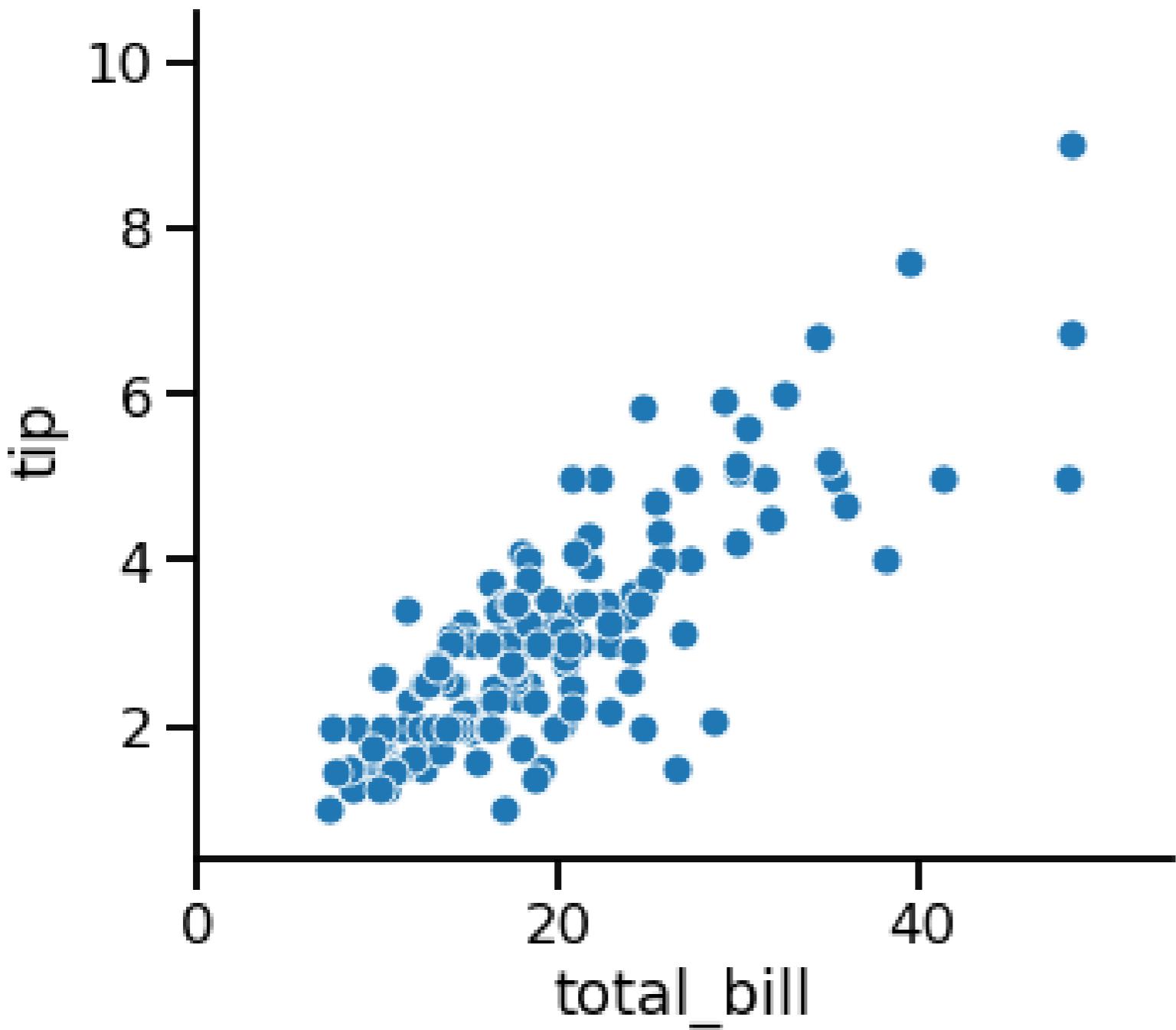
Relational plots

- Height vs. weight
- Number of school absences vs. final grade
- GDP vs. percent literate

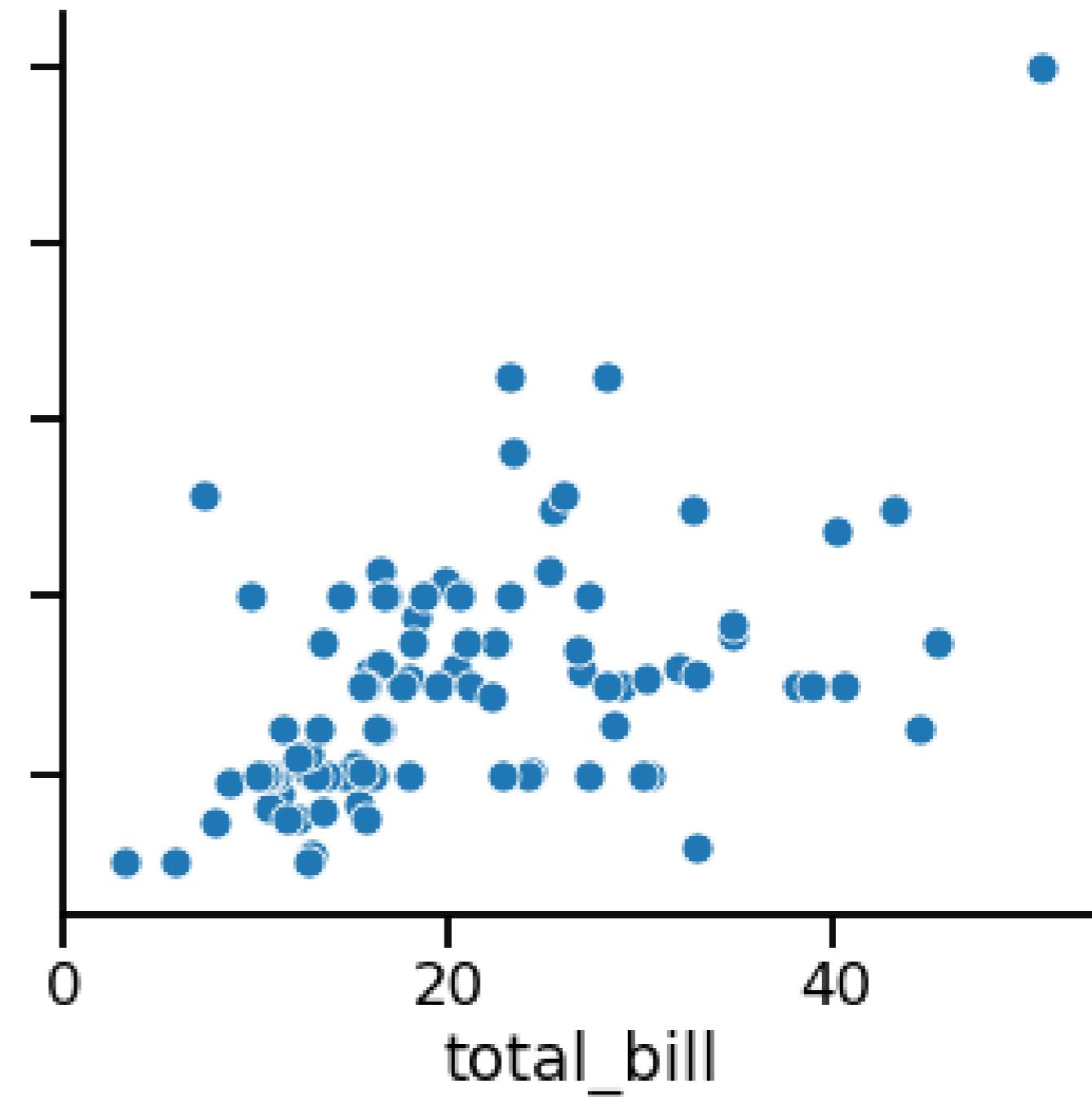




smoker = No



smoker = Yes



Introducing relplot()

- Create "relational plots": scatter plots or line plots

Why use `relplot()` instead of `scatterplot()` ?

- `relplot()` lets you create subplots in a single figure

scatterplot() vs. relplot()

Using `scatterplot()`

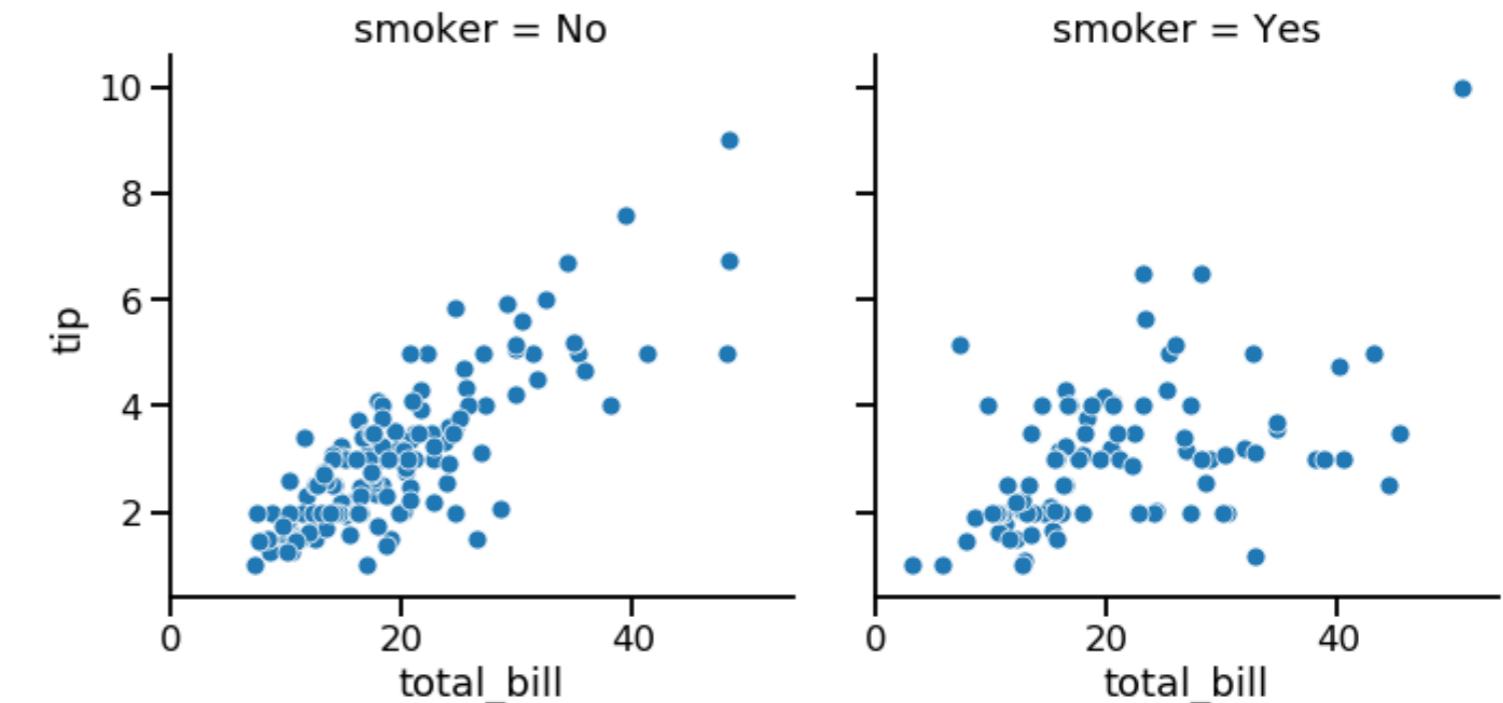
```
import seaborn as sns  
import matplotlib.pyplot as plt  
  
sns.scatterplot(x="total_bill",  
                 y="tip",  
                 data=tips)  
  
plt.show()
```

Using `relplot()`

```
import seaborn as sns  
import matplotlib.pyplot as plt  
  
sns.relplot(x="total_bill",  
            y="tip",  
            data=tips,  
            kind="scatter")  
  
plt.show()
```

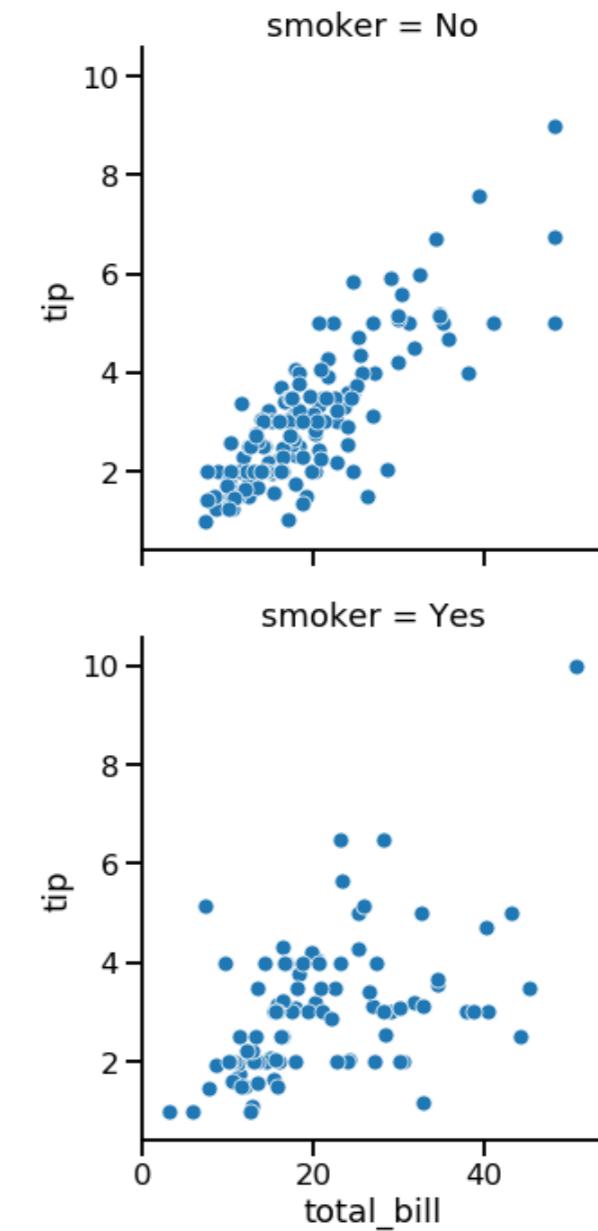
Subplots in columns

```
import seaborn as sns  
  
import matplotlib.pyplot as plt  
  
sns.relplot(x="total_bill",  
             y="tip",  
             data=tips,  
             kind="scatter",  
             col="smoker")  
  
plt.show()
```



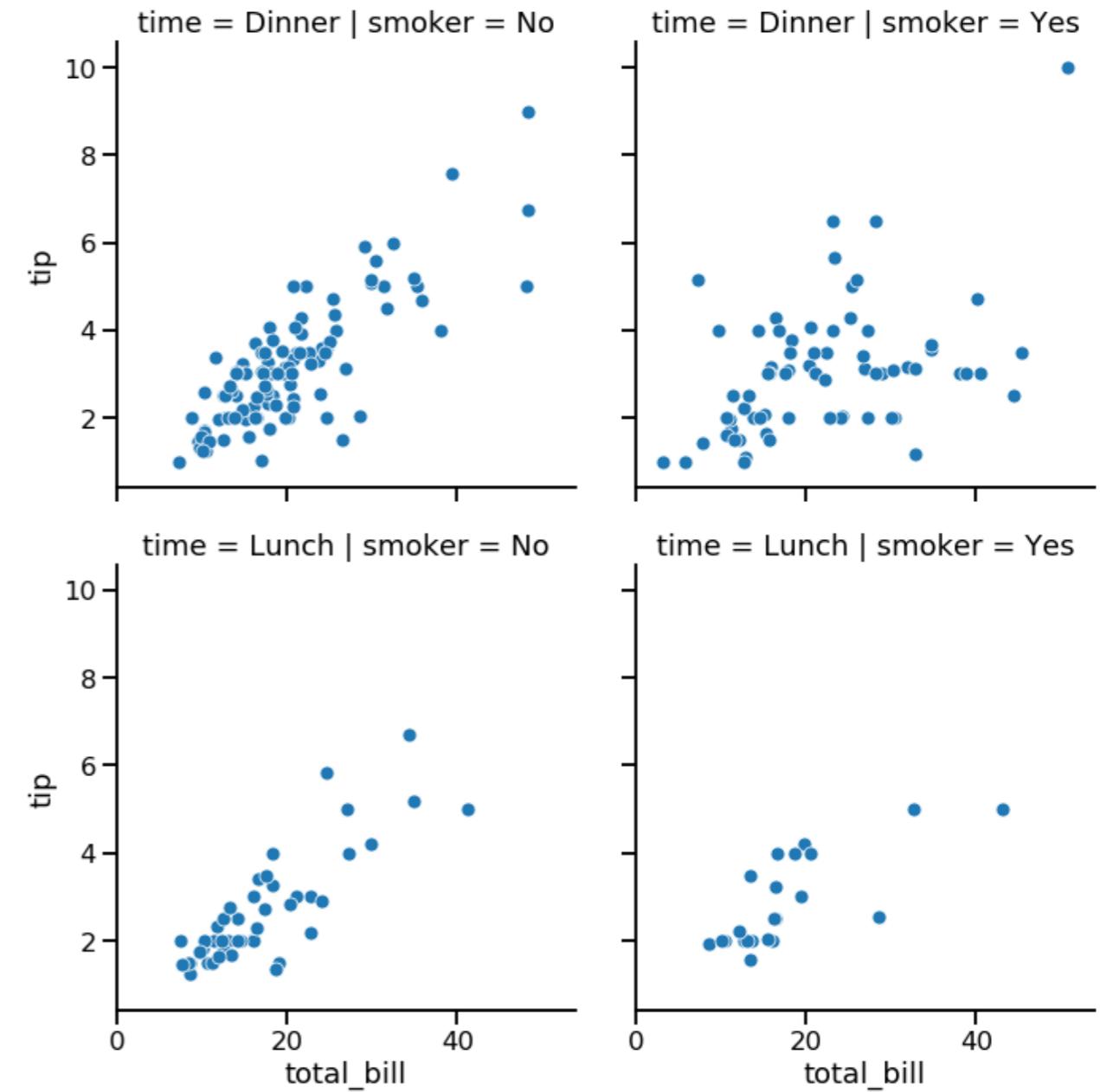
Subplots in rows

```
import seaborn as sns  
import matplotlib.pyplot as plt  
  
sns.relplot(x="total_bill",  
             y="tip",  
             data=tips,  
             kind="scatter",  
             row="smoker")  
  
plt.show()
```

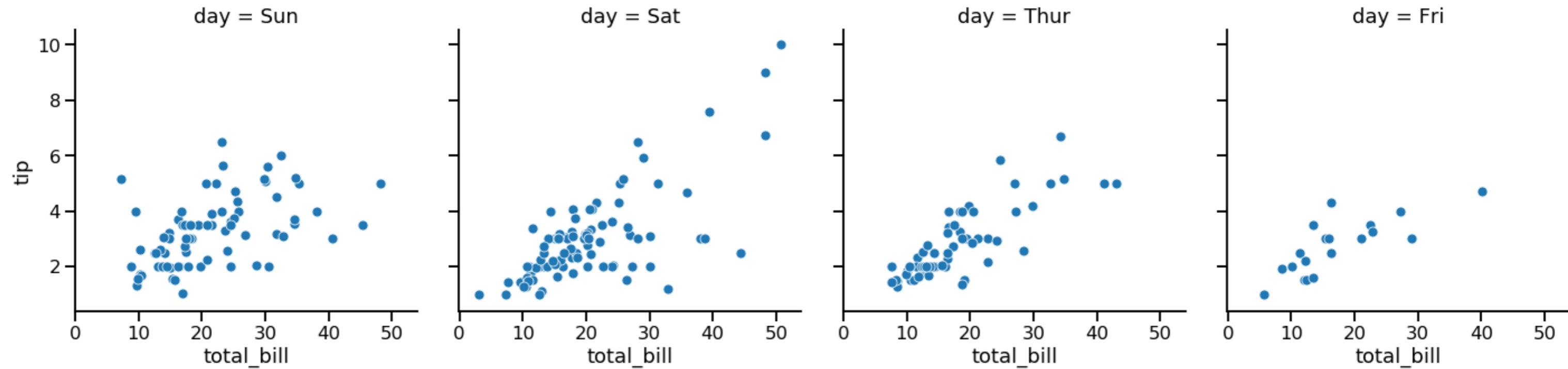


Subplots in rows and columns

```
import seaborn as sns  
  
import matplotlib.pyplot as plt  
  
sns.relplot(x="total_bill",  
             y="tip",  
             data=tips,  
             kind="scatter",  
             col="smoker",  
             row="time")  
  
plt.show()
```

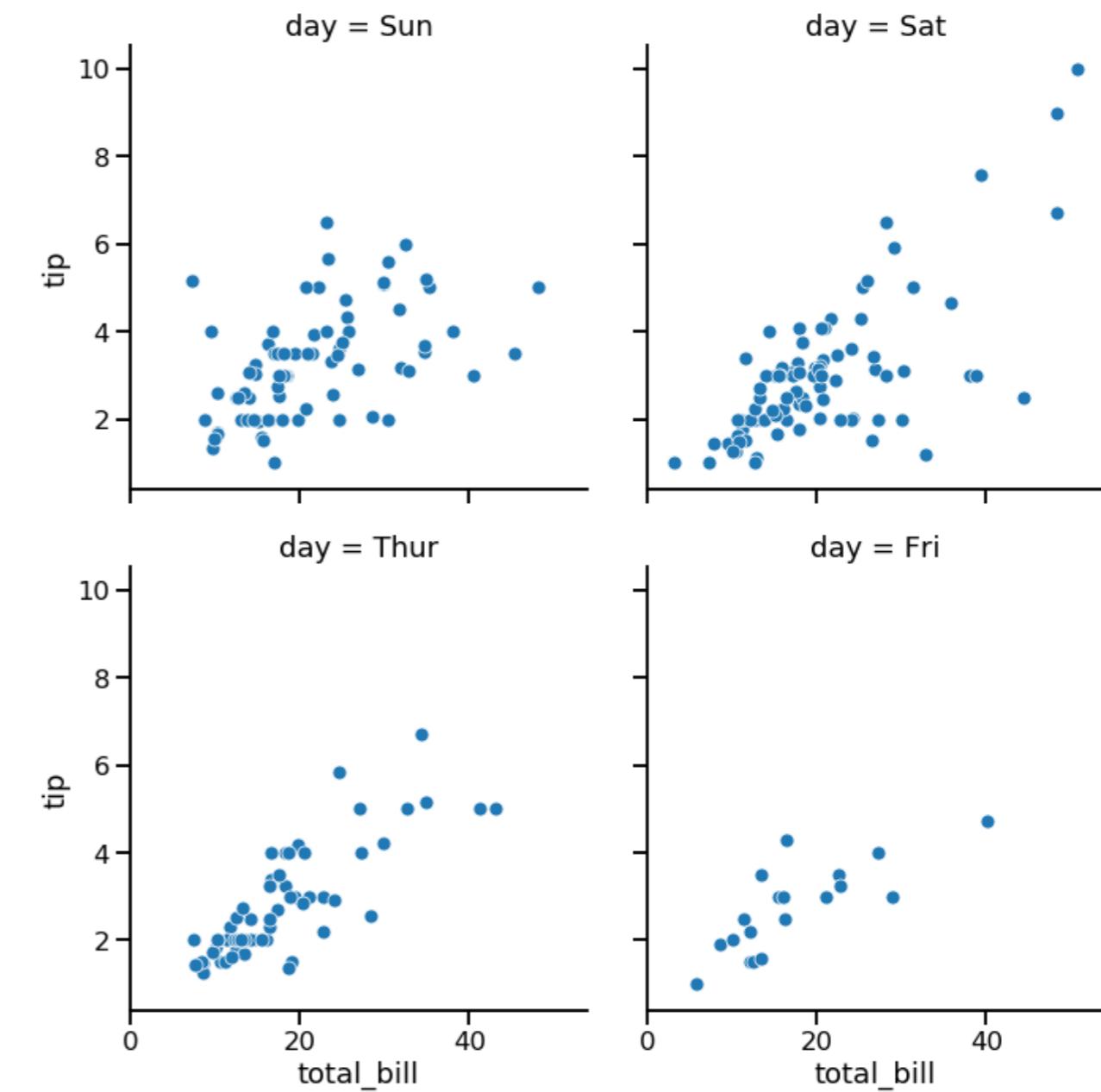


Subgroups for days of the week



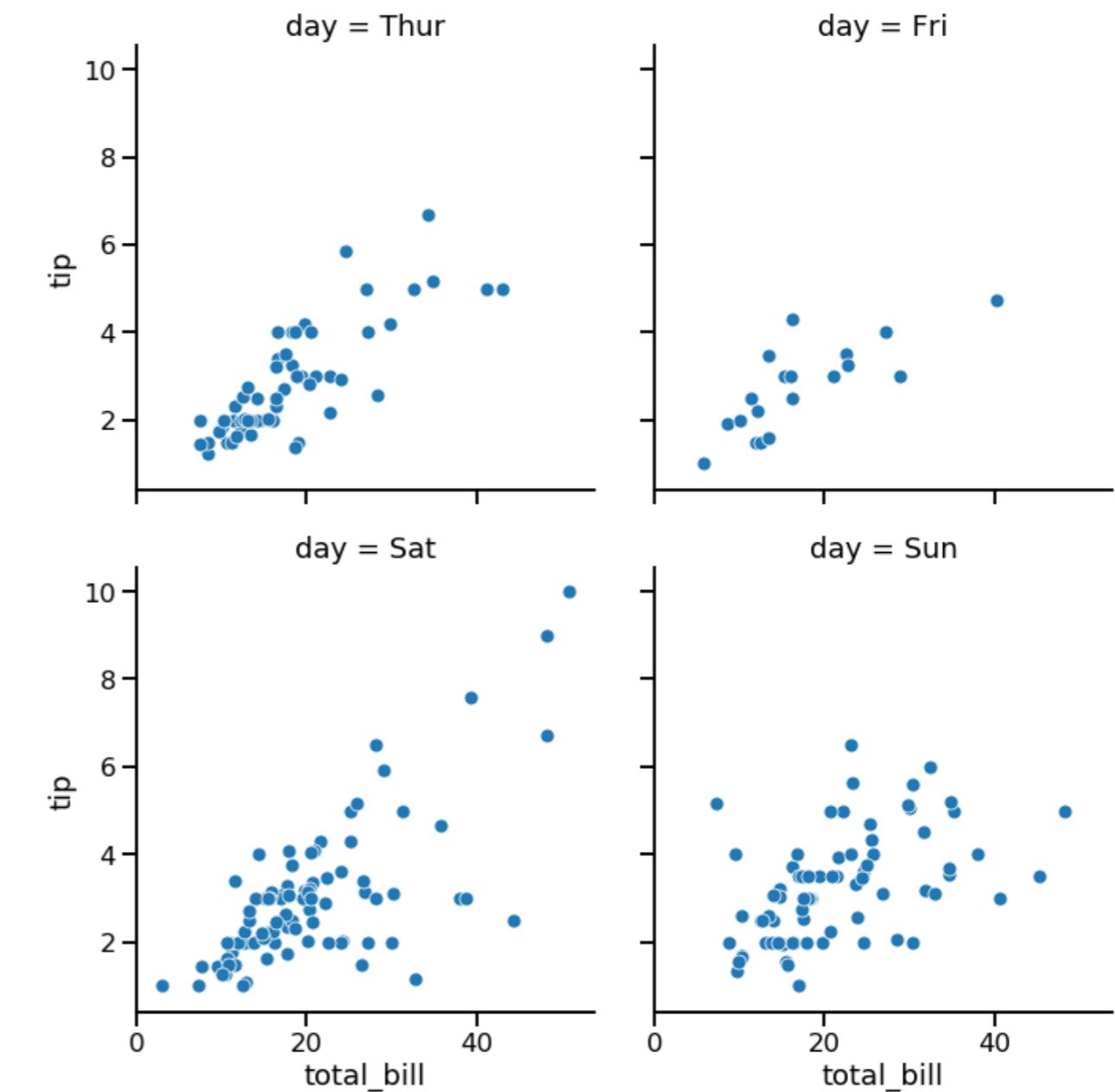
Wrapping columns

```
import seaborn as sns  
  
import matplotlib.pyplot as plt  
  
sns.relplot(x="total_bill",  
             y="tip",  
             data=tips,  
             kind="scatter",  
             col="day",  
             col_wrap=2)  
  
plt.show()
```



Ordering columns

```
import seaborn as sns  
  
import matplotlib.pyplot as plt  
  
sns.relplot(x="total_bill",  
             y="tip",  
             data=tips,  
             kind="scatter",  
             col="day",  
             col_wrap=2,  
             col_order=["Thur",  
                        "Fri",  
                        "Sat",  
                        "Sun"])  
  
plt.show()
```

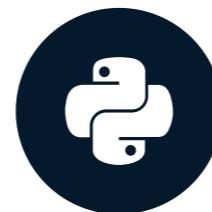


Let's practice!

INTRODUCTION TO DATA VISUALIZATION WITH SEABORN

Customizing scatter plots

INTRODUCTION TO DATA VISUALIZATION WITH SEABORN



Erin Case
Data Scientist

Scatter plot overview

Show relationship between two quantitative variables

We've seen:

- Subplots (`col` and `row`)
- Subgroups with color (`hue`)

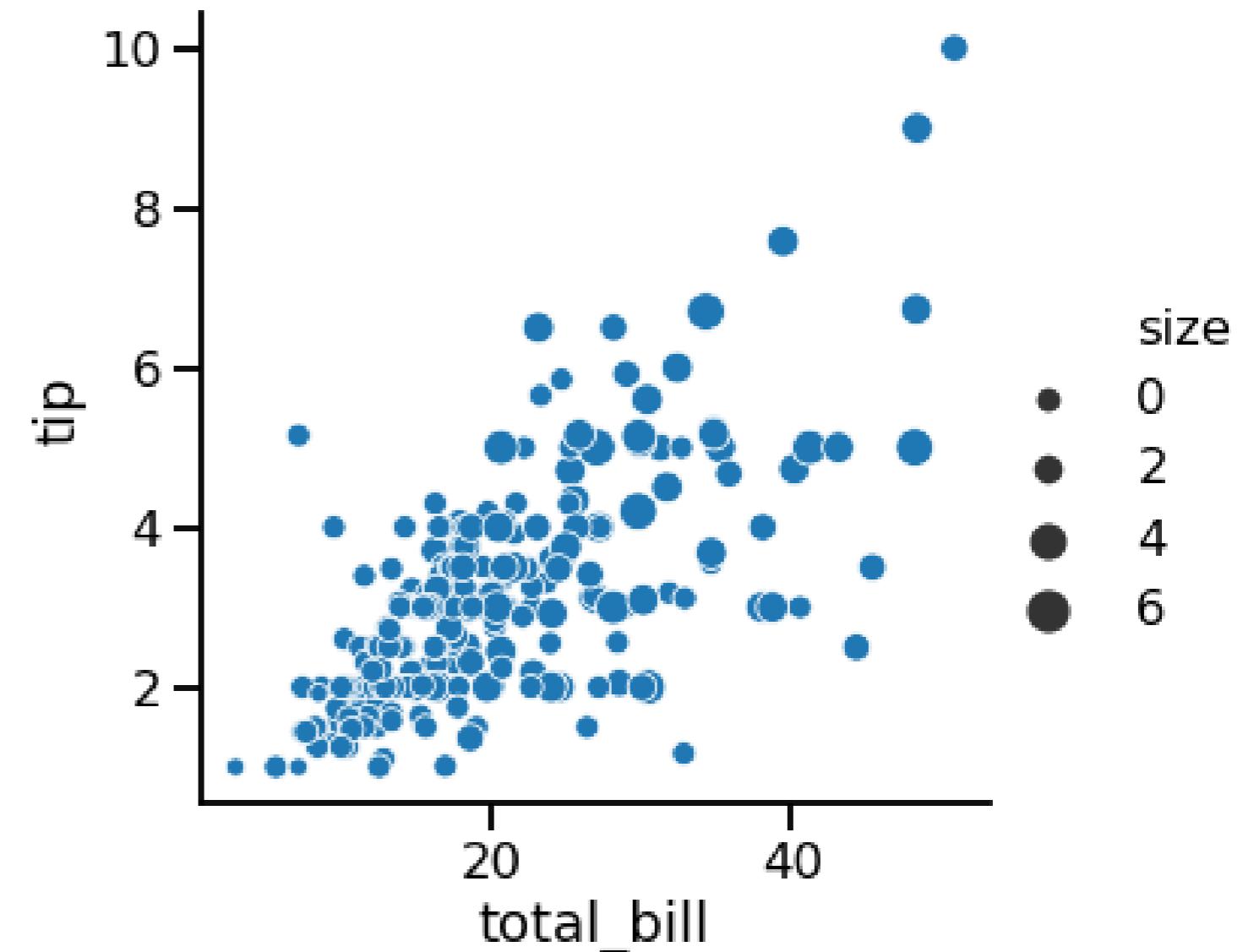
New Customizations:

- Subgroups with point size and style
- Changing point transparency

Use with both `scatterplot()` and `relplot()`

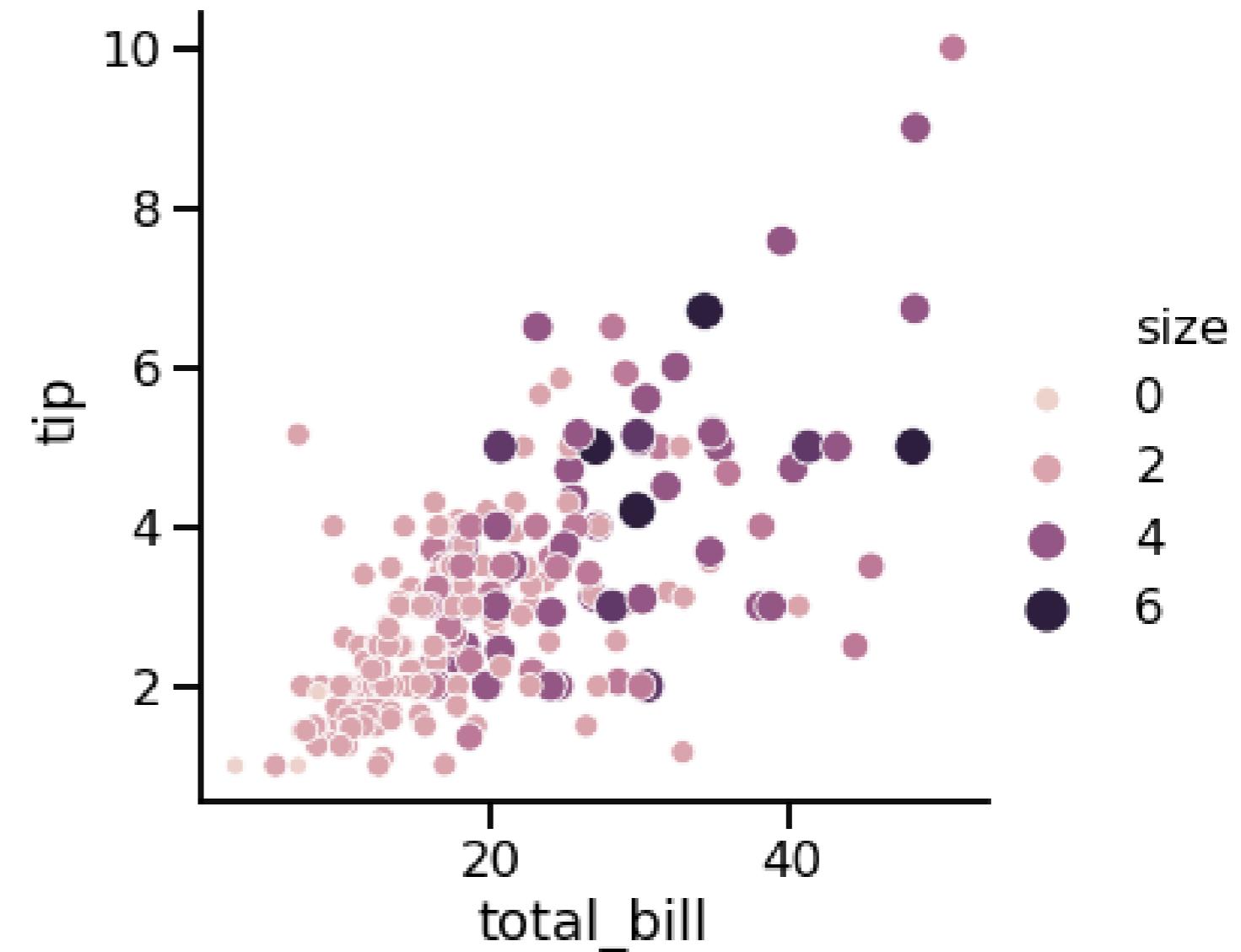
Subgroups with point size

```
import seaborn as sns  
import matplotlib.pyplot as plt  
  
sns.relplot(x="total_bill",  
             y="tip",  
             data=tips,  
             kind="scatter",  
             size="size")  
plt.show()
```



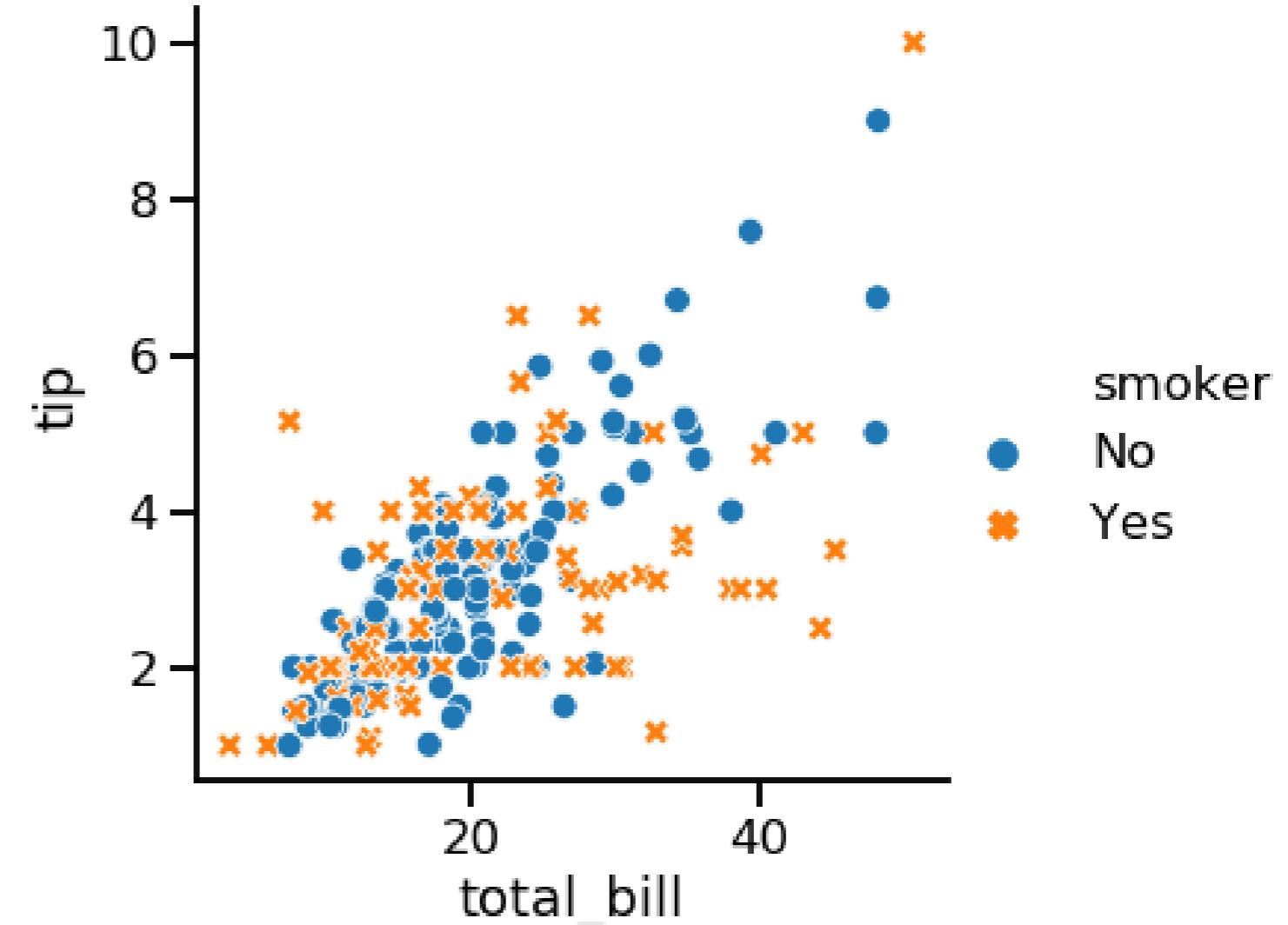
Point size and hue

```
import seaborn as sns  
import matplotlib.pyplot as plt  
  
sns.relplot(x="total_bill",  
             y="tip",  
             data=tips,  
             kind="scatter",  
             size="size",  
             hue="size")  
  
plt.show()
```



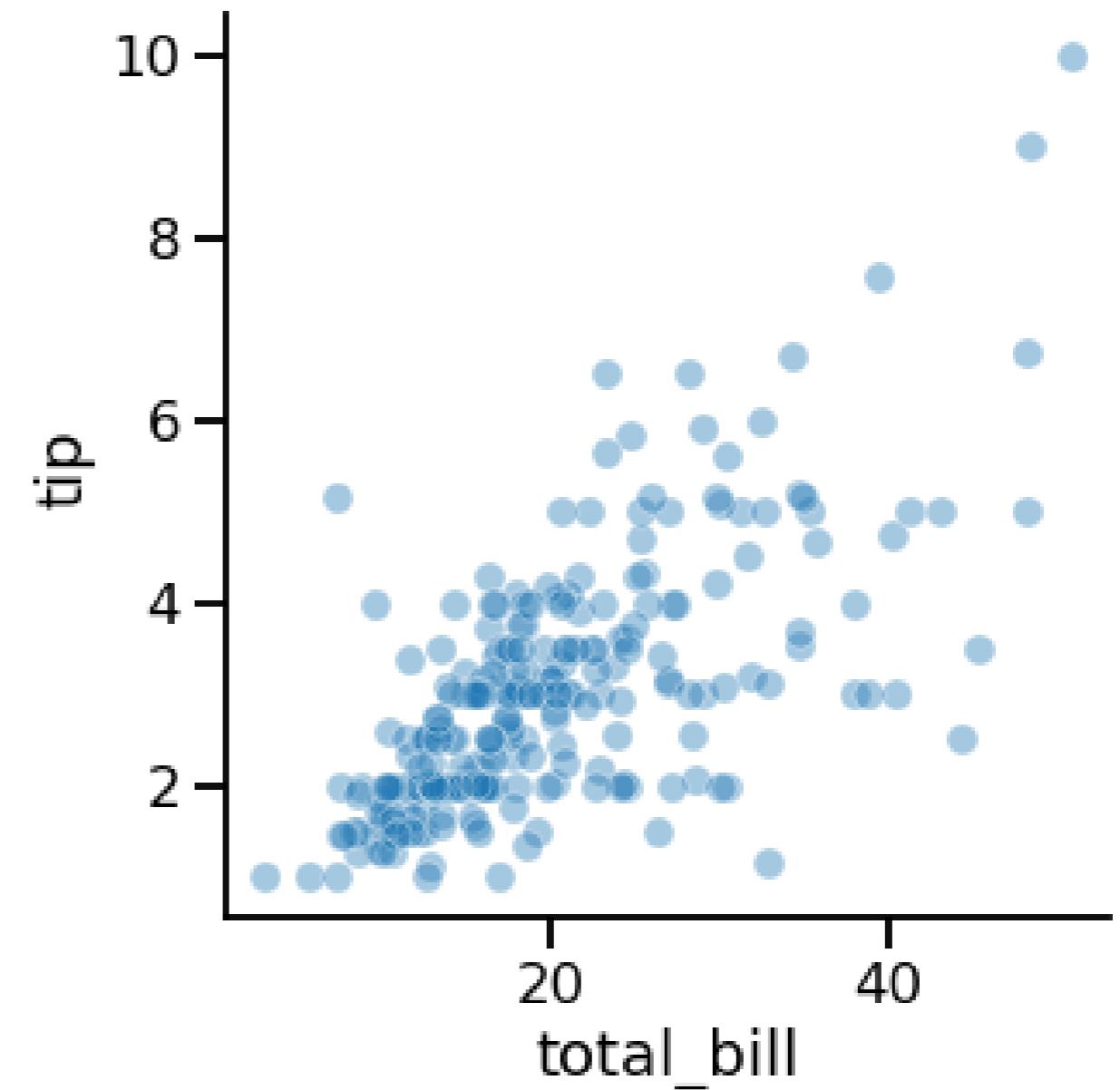
Subgroups with point style

```
import seaborn as sns  
import matplotlib.pyplot as plt  
  
sns.relplot(x="total_bill",  
             y="tip",  
             data=tips,  
             kind="scatter",  
             hue="smoker",  
             style="smoker")  
  
plt.show()
```



Changing point transparency

```
import seaborn as sns  
import matplotlib.pyplot as plt  
  
# Set alpha to be between 0 and 1  
sns.relplot(x="total_bill",  
             y="tip",  
             data=tips,  
             kind="scatter",  
             alpha=0.4)  
plt.show()
```

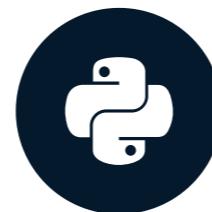


Let's practice!

INTRODUCTION TO DATA VISUALIZATION WITH SEABORN

Introduction to line plots

INTRODUCTION TO DATA VISUALIZATION WITH SEABORN



Erin Case
Data Scientist

What are line plots?

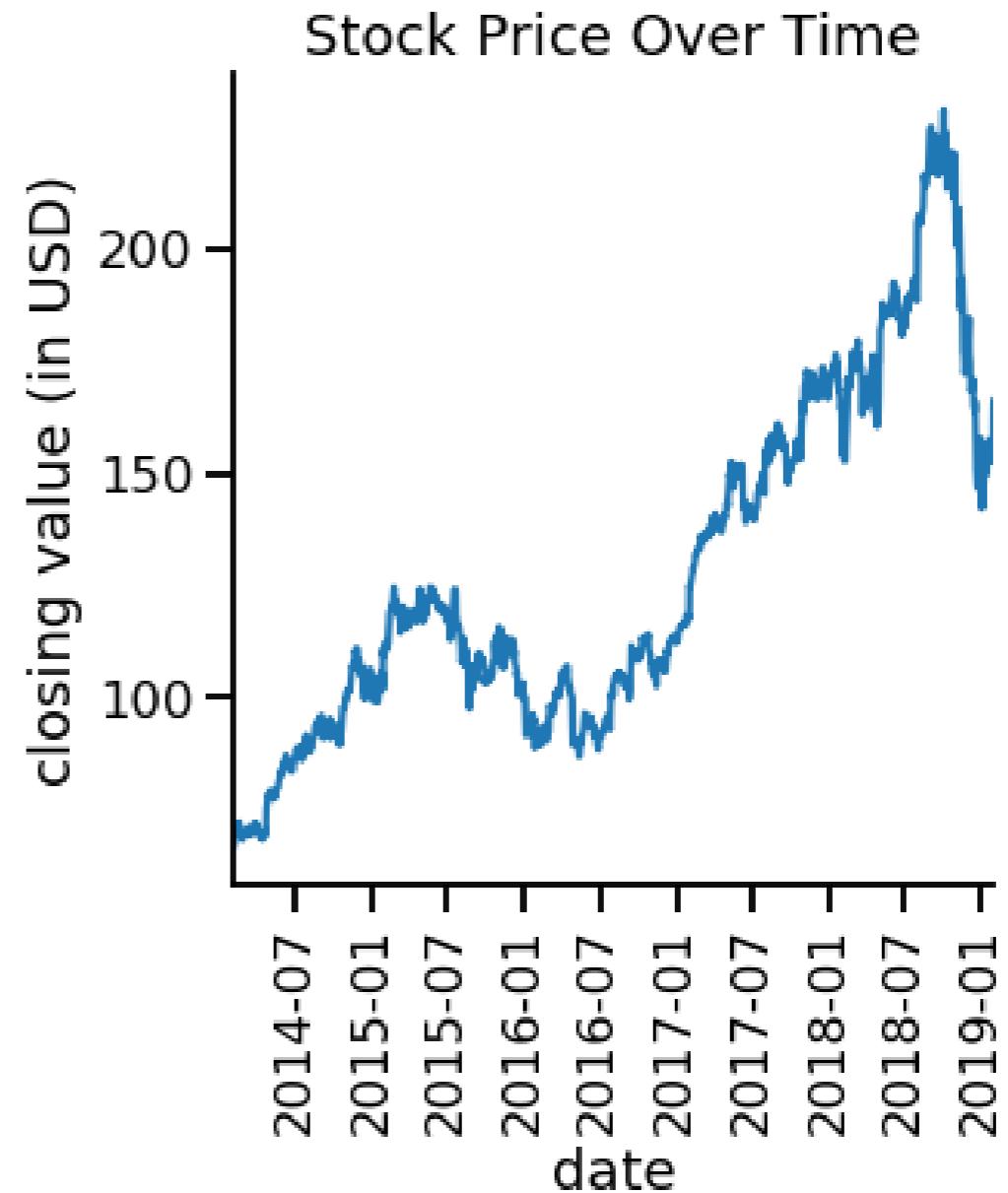
Two types of relational plots: scatter plots and line plots

Scatter plots

- Each plot point is an independent observation

Line plots

- Each plot point represents the same "thing", typically tracked over time



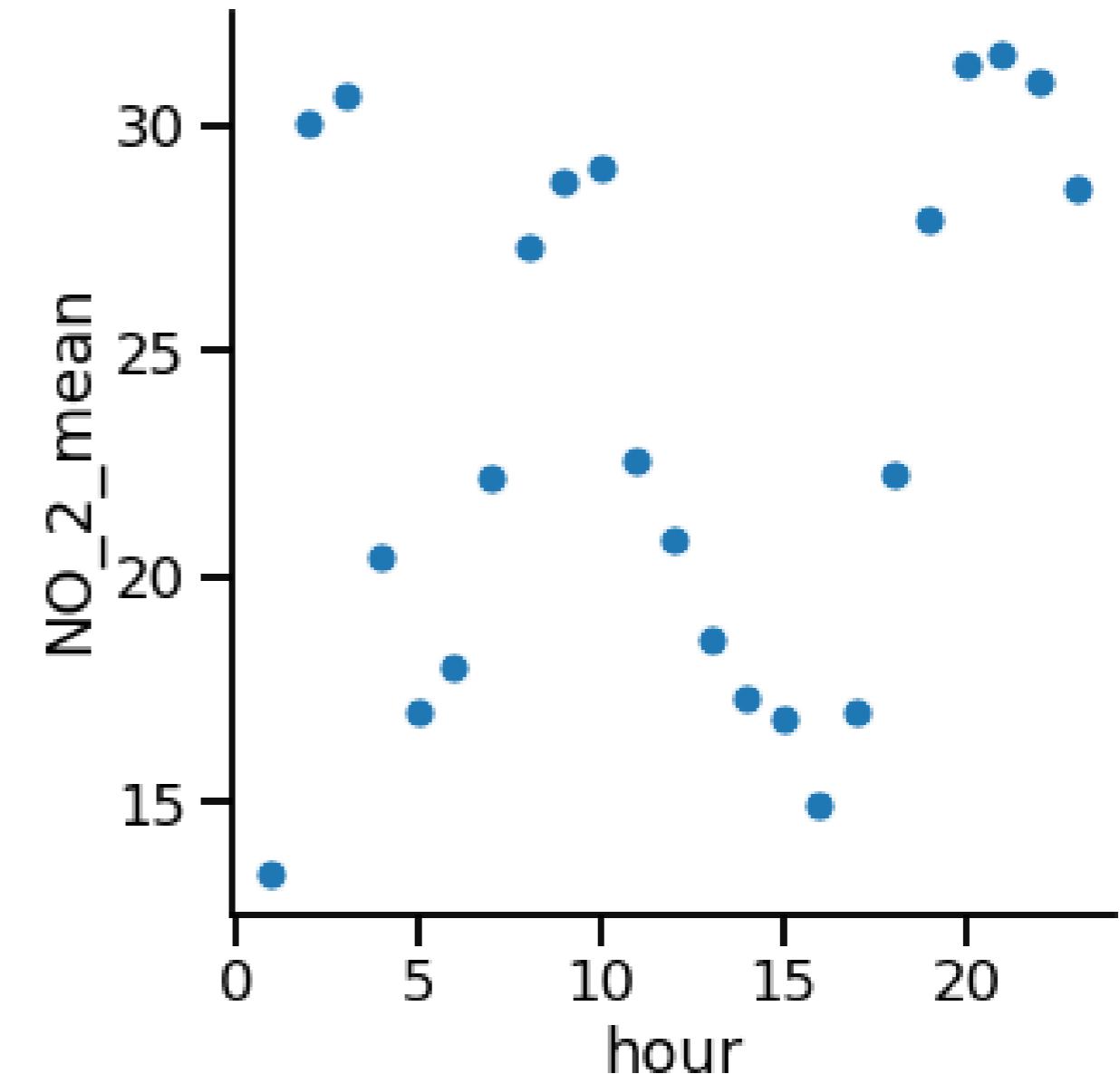
Air pollution data

- Collection stations throughout city
- Air samples of nitrogen dioxide levels

hour	NO_2_mean
0	13.375000
1	30.041667
2	30.666667
3	20.416667
4	16.958333

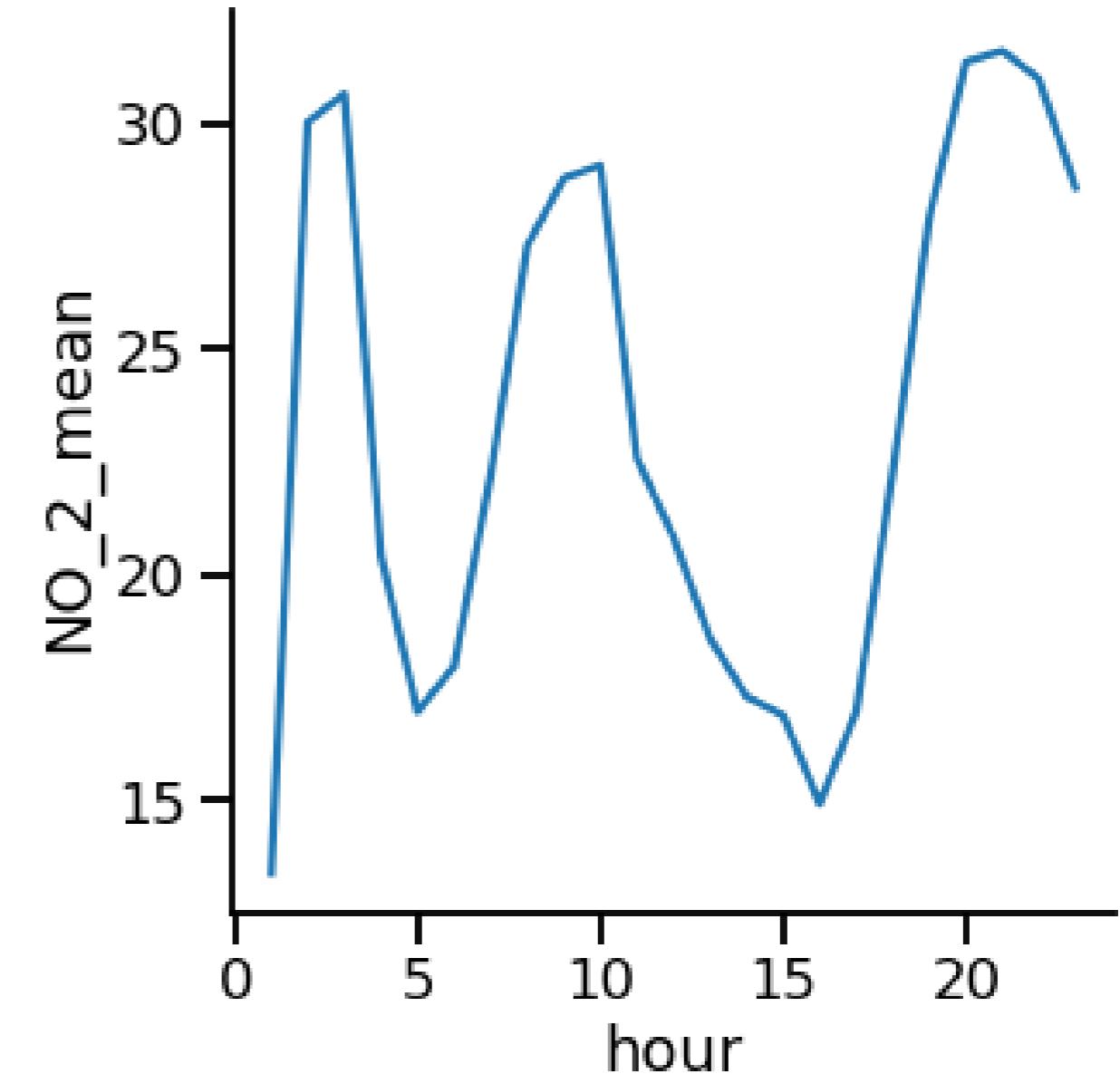
Scatter plot

```
import matplotlib.pyplot as plt  
import seaborn as sns  
  
sns.relplot(x="hour", y="NO_2_mean",  
             data=air_df_mean,  
             kind="scatter")  
  
plt.show()
```



Line plot

```
import matplotlib.pyplot as plt  
import seaborn as sns  
  
sns.relplot(x="hour", y="NO_2_mean",  
             data=air_df_mean,  
             kind="line")  
  
plt.show()
```

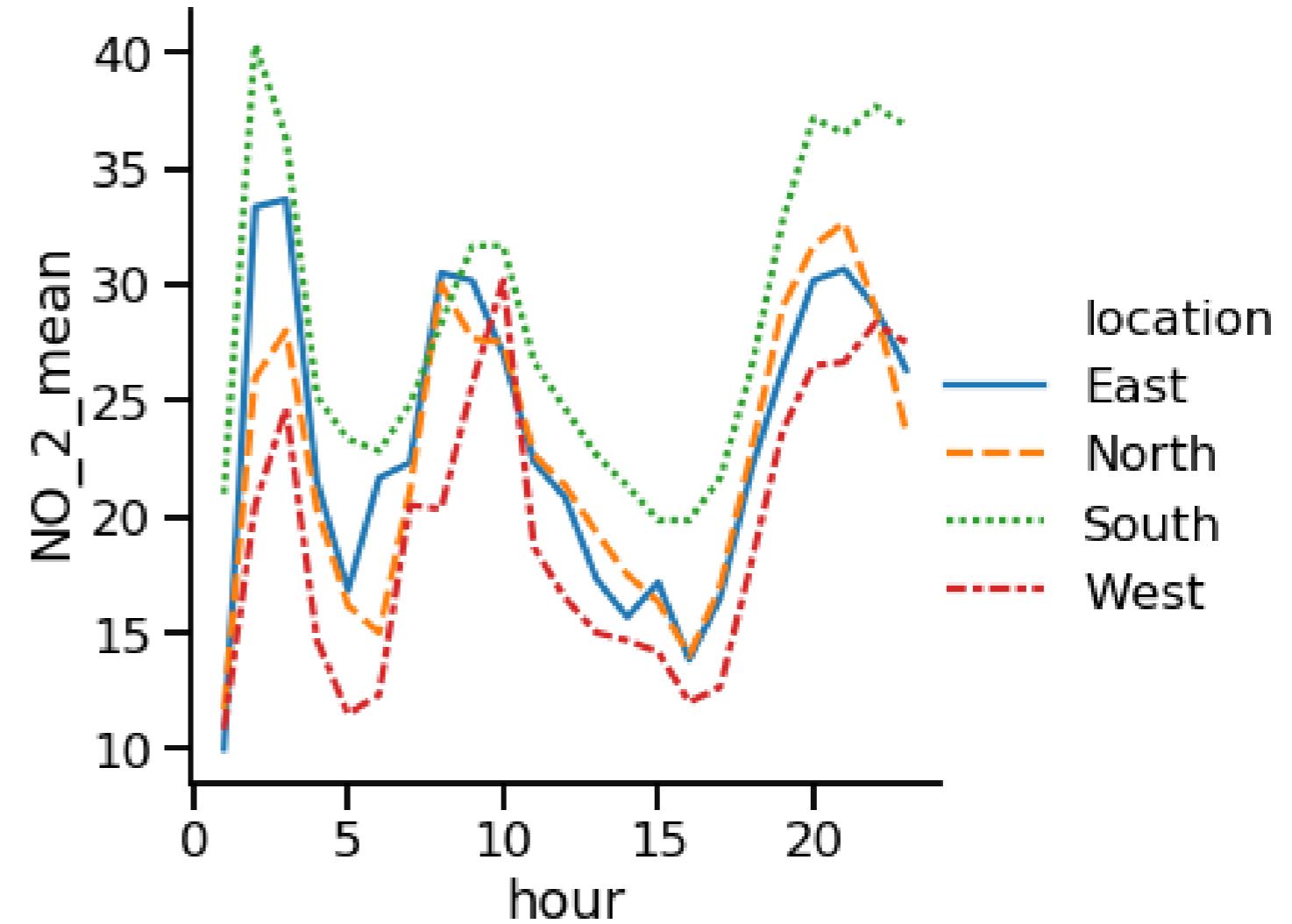


Subgroups by location

	hour	location	NO_2_mean
0	1	East	10.000000
1	1	North	11.666667
2	1	South	21.000000
3	1	West	10.833333
4	2	East	33.333333

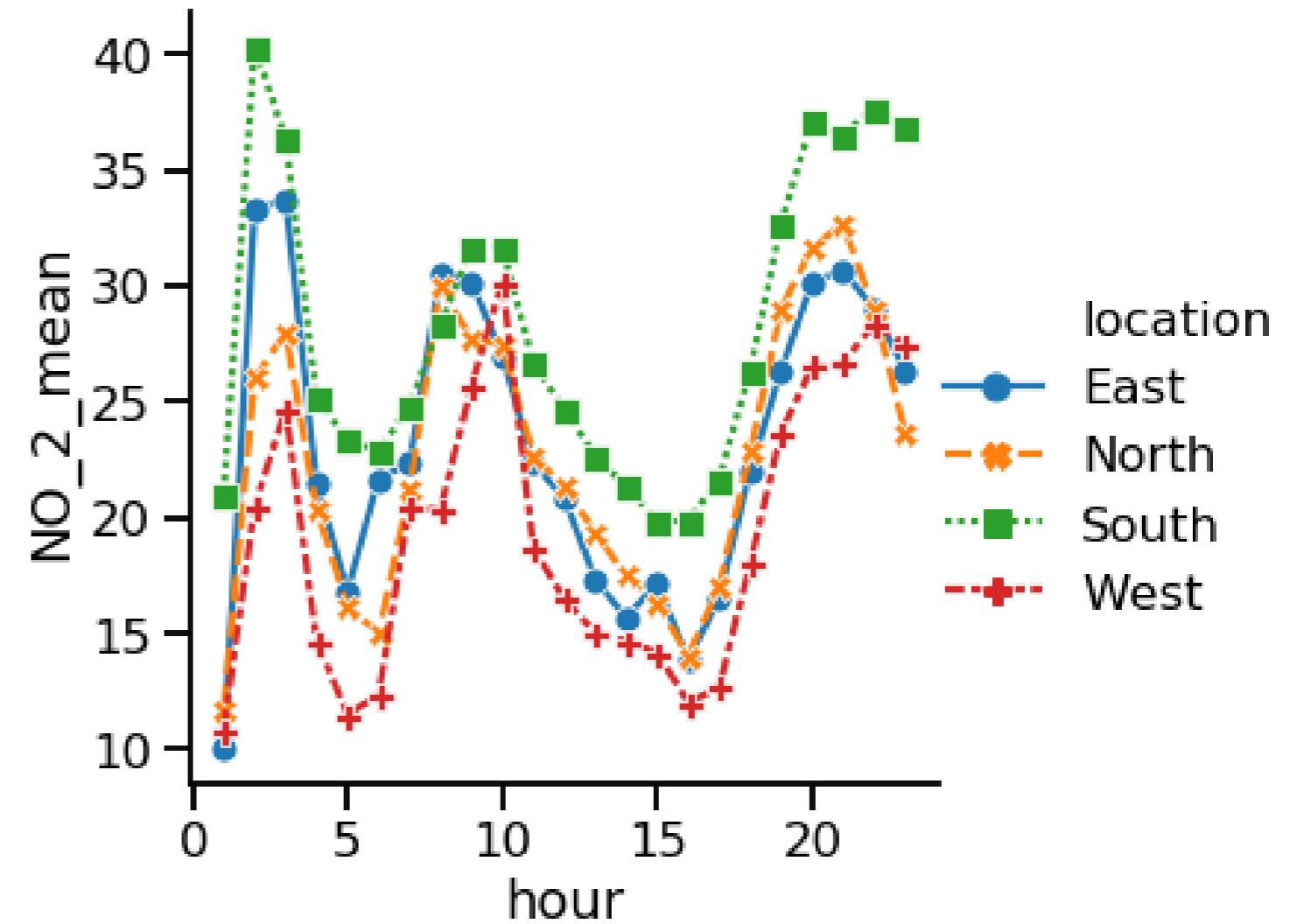
Subgroups by location

```
import matplotlib.pyplot as plt  
import seaborn as sns  
  
sns.relplot(x="hour", y="NO_2_mean",  
             data=air_df_loc_mean,  
             kind="line",  
             style="location",  
             hue="location")  
  
plt.show()
```



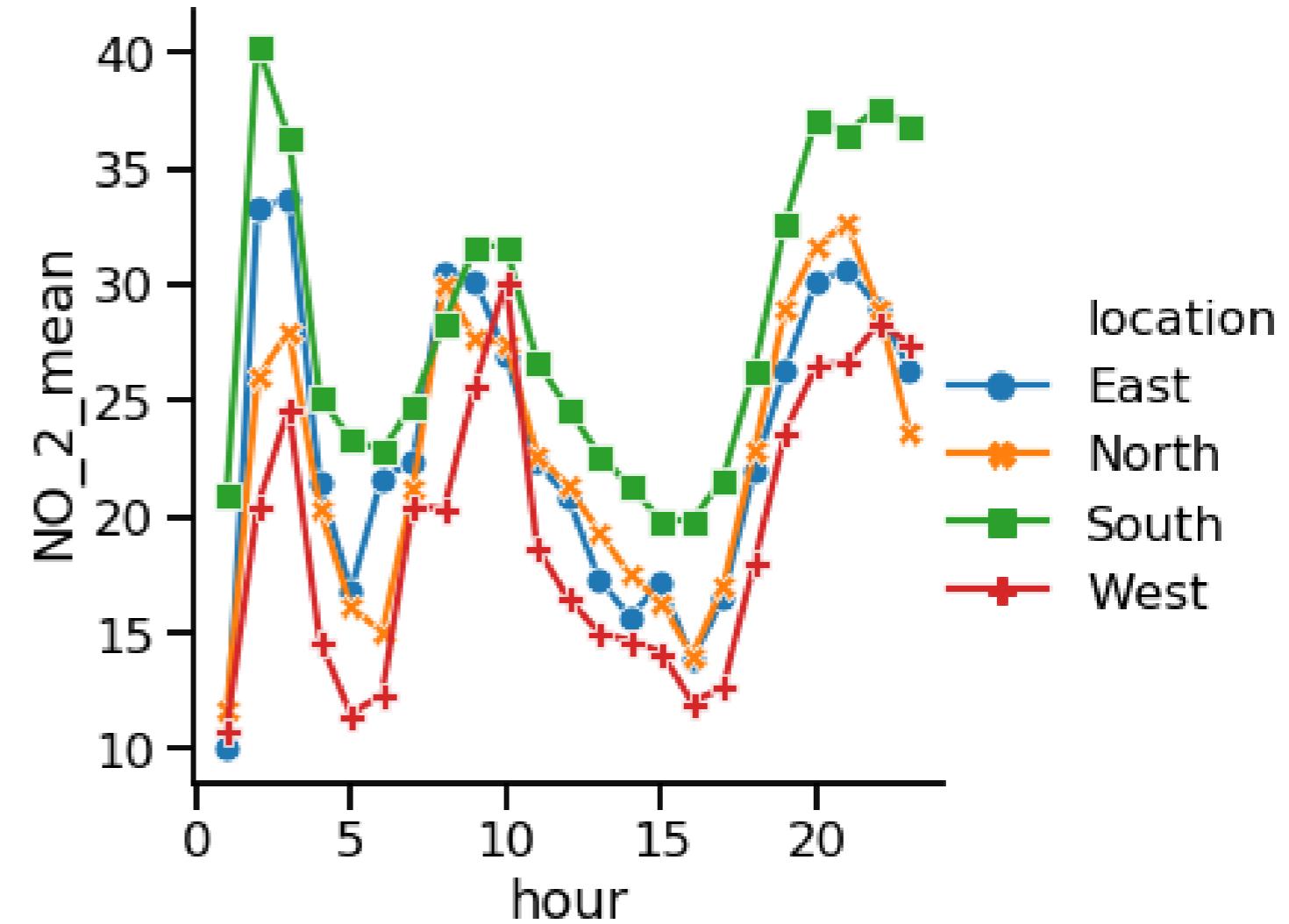
Adding markers

```
import matplotlib.pyplot as plt  
import seaborn as sns  
  
sns.relplot(x="hour", y="NO_2_mean",  
             data=air_df_loc_mean,  
             kind="line",  
             style="location",  
             hue="location",  
             markers=True)  
  
plt.show()
```



Turning off line style

```
import matplotlib.pyplot as plt  
import seaborn as sns  
  
sns.relplot(x="hour", y="NO_2_mean",  
             data=air_df_loc_mean,  
             kind="line",  
             style="location",  
             hue="location",  
             markers=True,  
             dashes=False)  
  
plt.show()
```



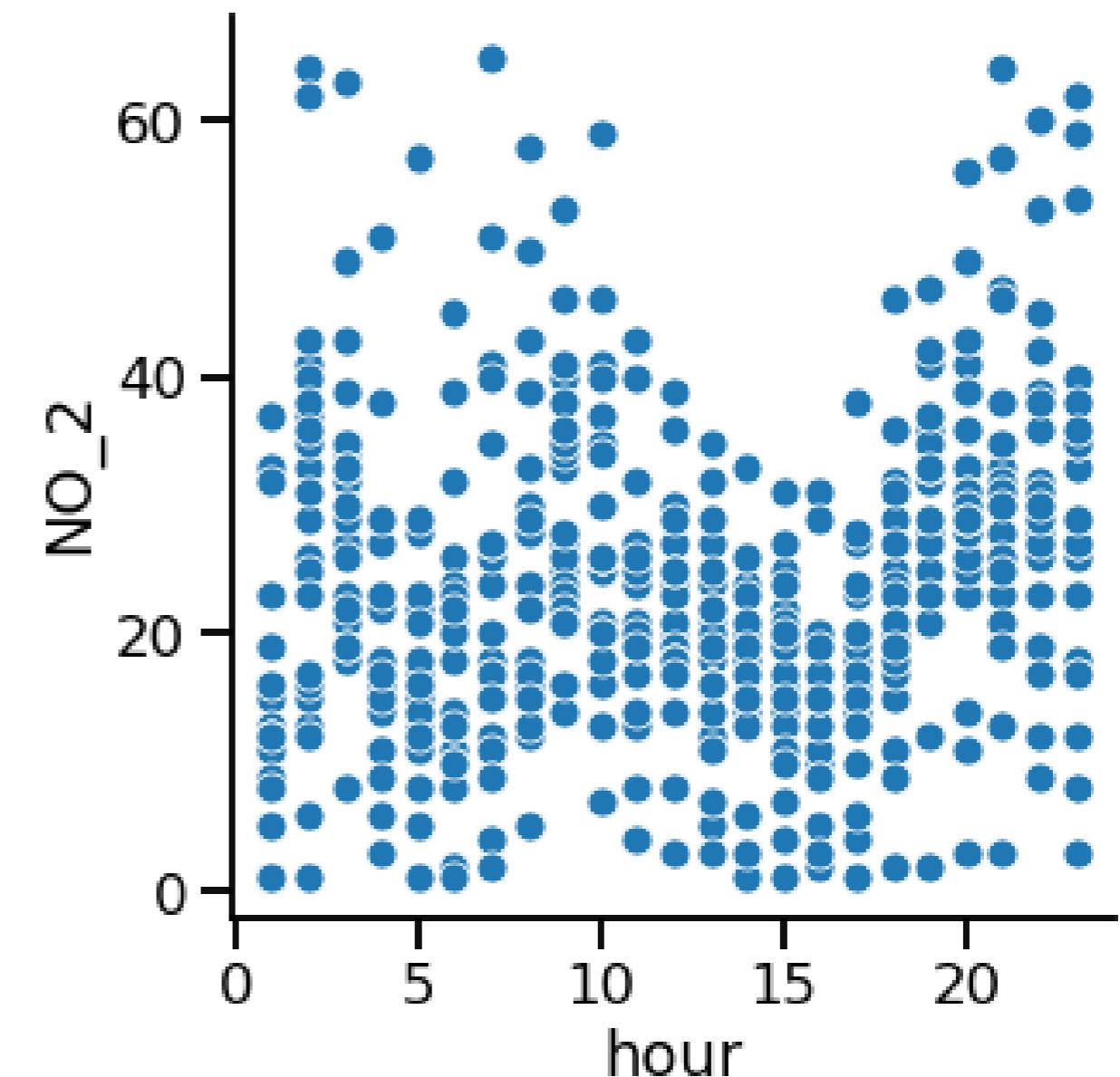
Multiple observations per x-value

	hour	NO_2	station	location
0	1	15.0	28079004	South
1	1	33.0	28079008	South
2	1	11.0	28079011	South
3	1	12.0	28079016	South
4	1	23.0	28079017	South

Multiple observations per x-value

Scatter plot

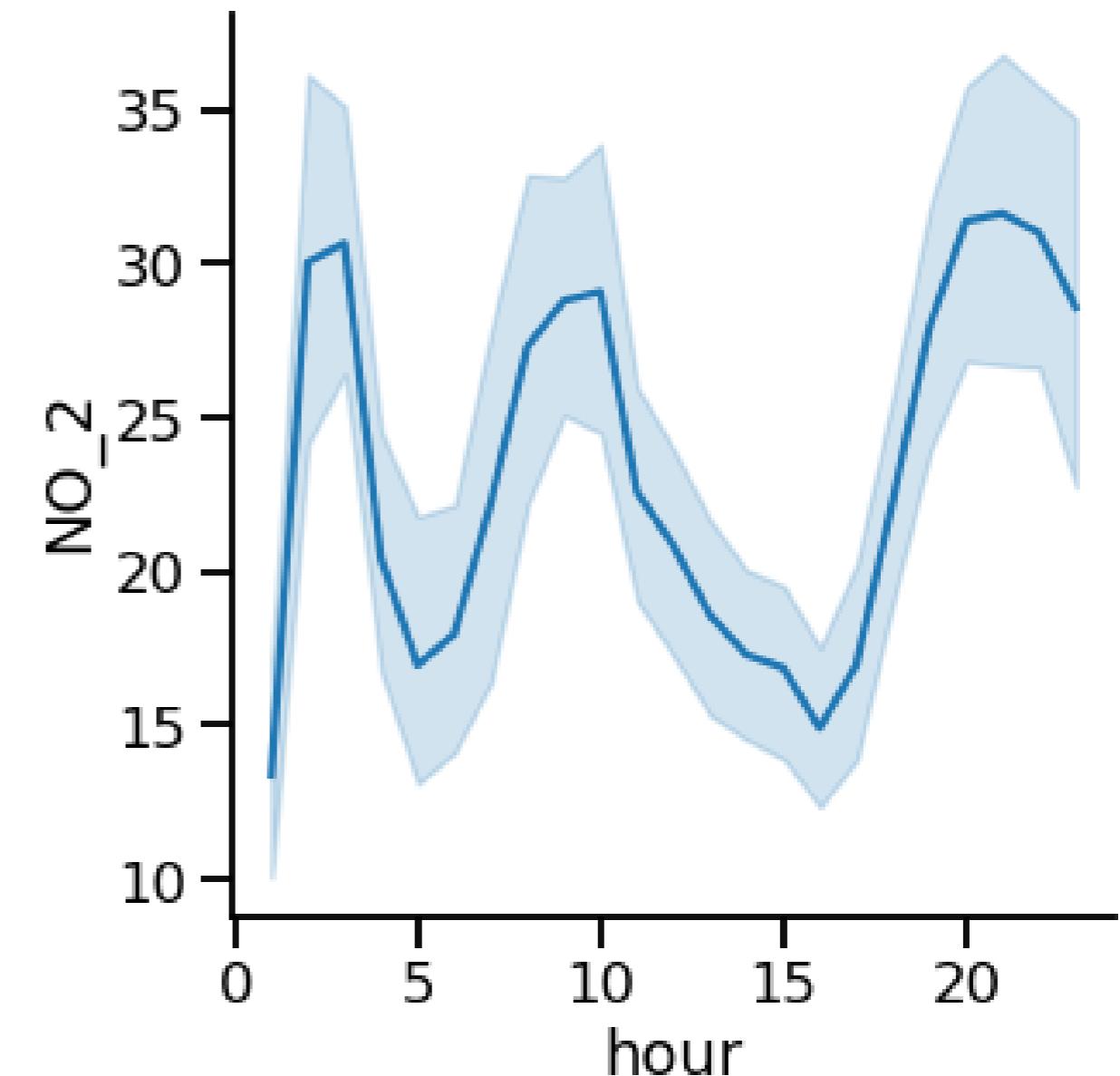
```
import matplotlib.pyplot as plt  
import seaborn as sns  
  
sns.relplot(x="hour", y="NO_2",  
             data=air_df,  
             kind="scatter")  
  
plt.show()
```



Multiple observations per x-value

Line plot

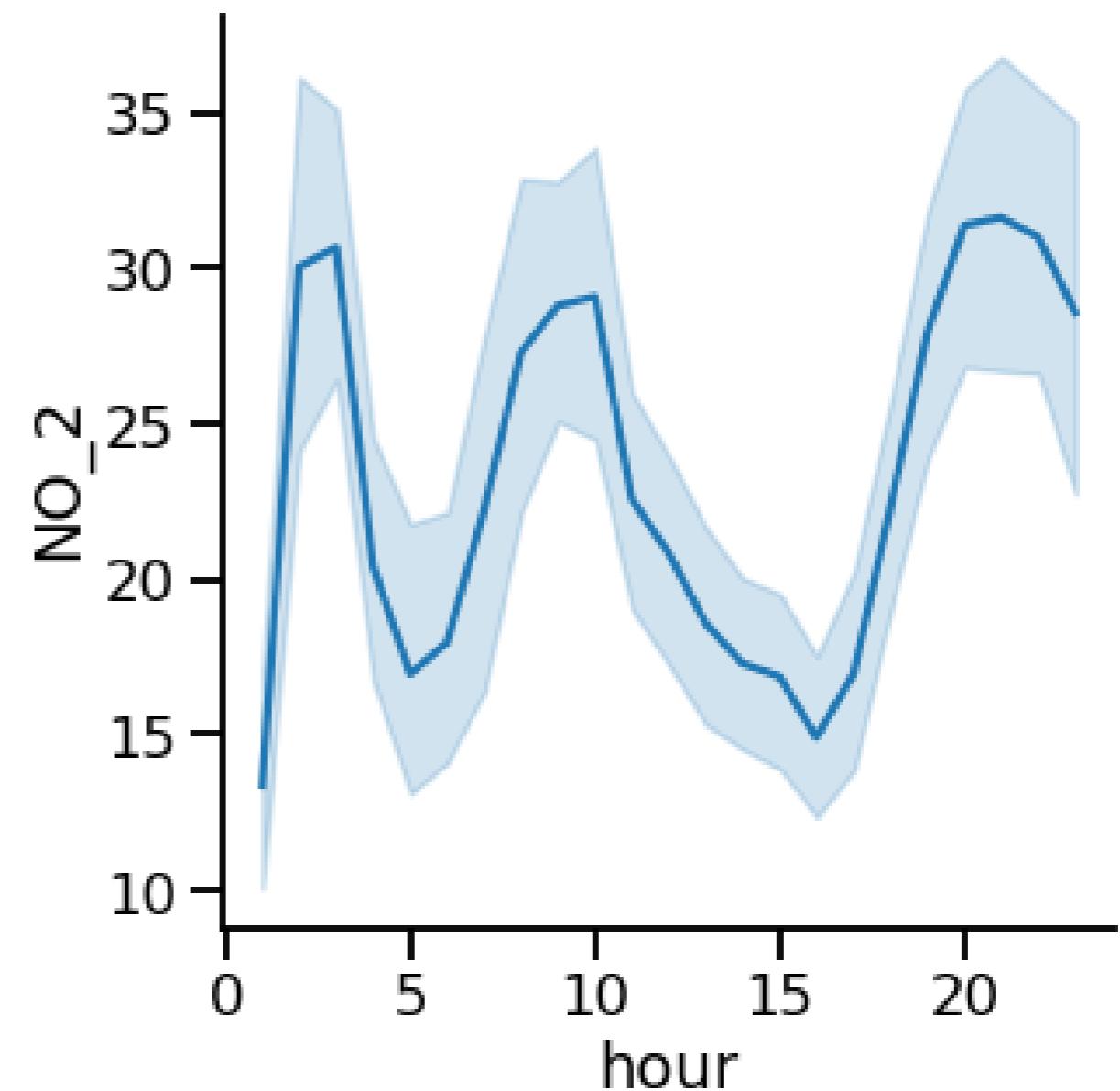
```
import matplotlib.pyplot as plt  
import seaborn as sns  
  
sns.relplot(x="hour", y="NO_2",  
             data=air_df,  
             kind="line")  
  
plt.show()
```



Multiple observations per x-value

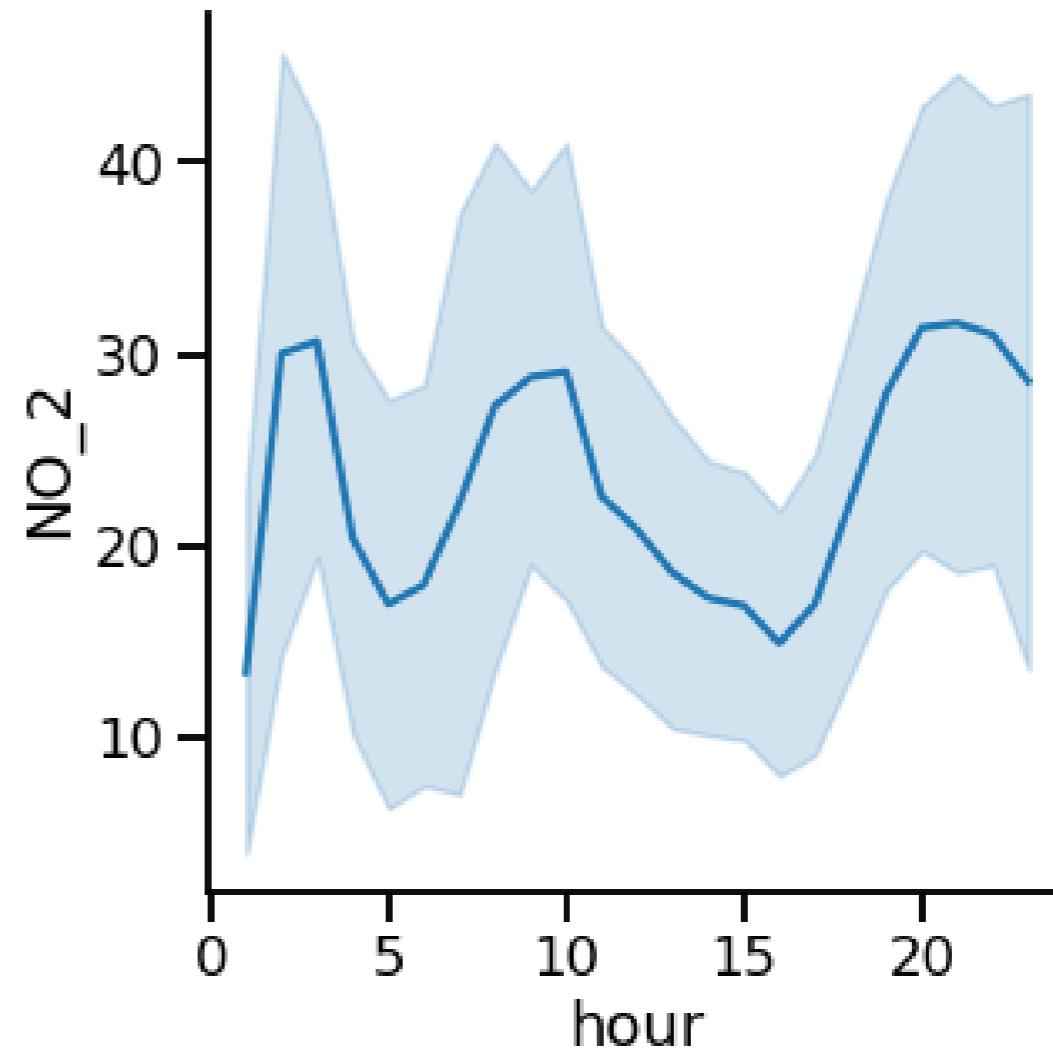
Shaded region is the confidence interval

- Assumes dataset is a random sample
- 95% confident that the mean is within this interval
- Indicates uncertainty in our estimate



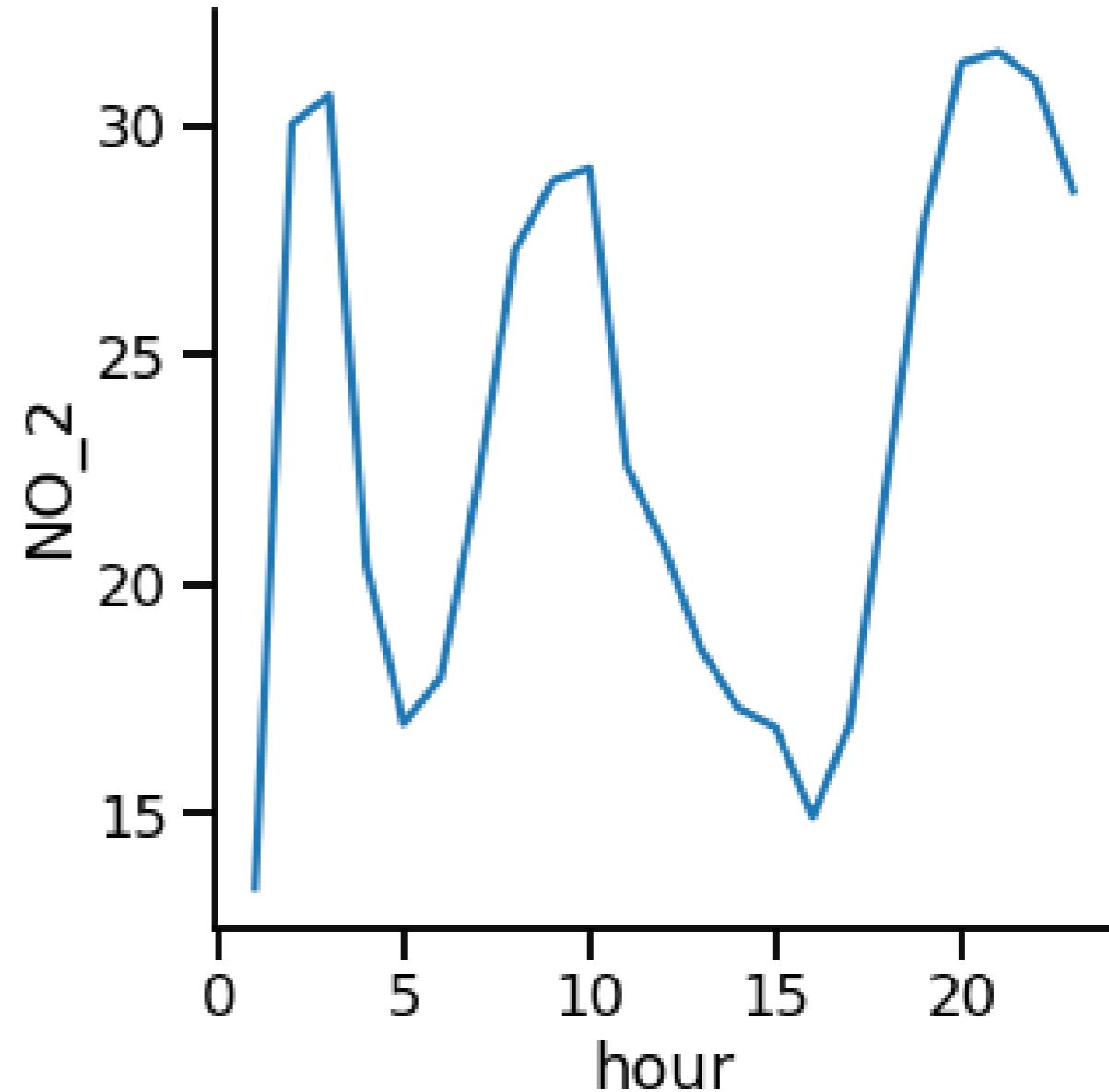
Replacing confidence interval with standard deviation

```
import matplotlib.pyplot as plt  
import seaborn as sns  
  
sns.relplot(x="hour", y="NO_2",  
             data=air_df,  
             kind="line",  
             ci="sd")  
  
plt.show()
```



Turning off confidence interval

```
import matplotlib.pyplot as plt  
import seaborn as sns  
  
sns.relplot(x="hour", y="NO_2",  
             data=air_df,  
             kind="line",  
             ci=None)  
  
plt.show()
```



Let's practice!

INTRODUCTION TO DATA VISUALIZATION WITH SEABORN

Count plots and bar plots

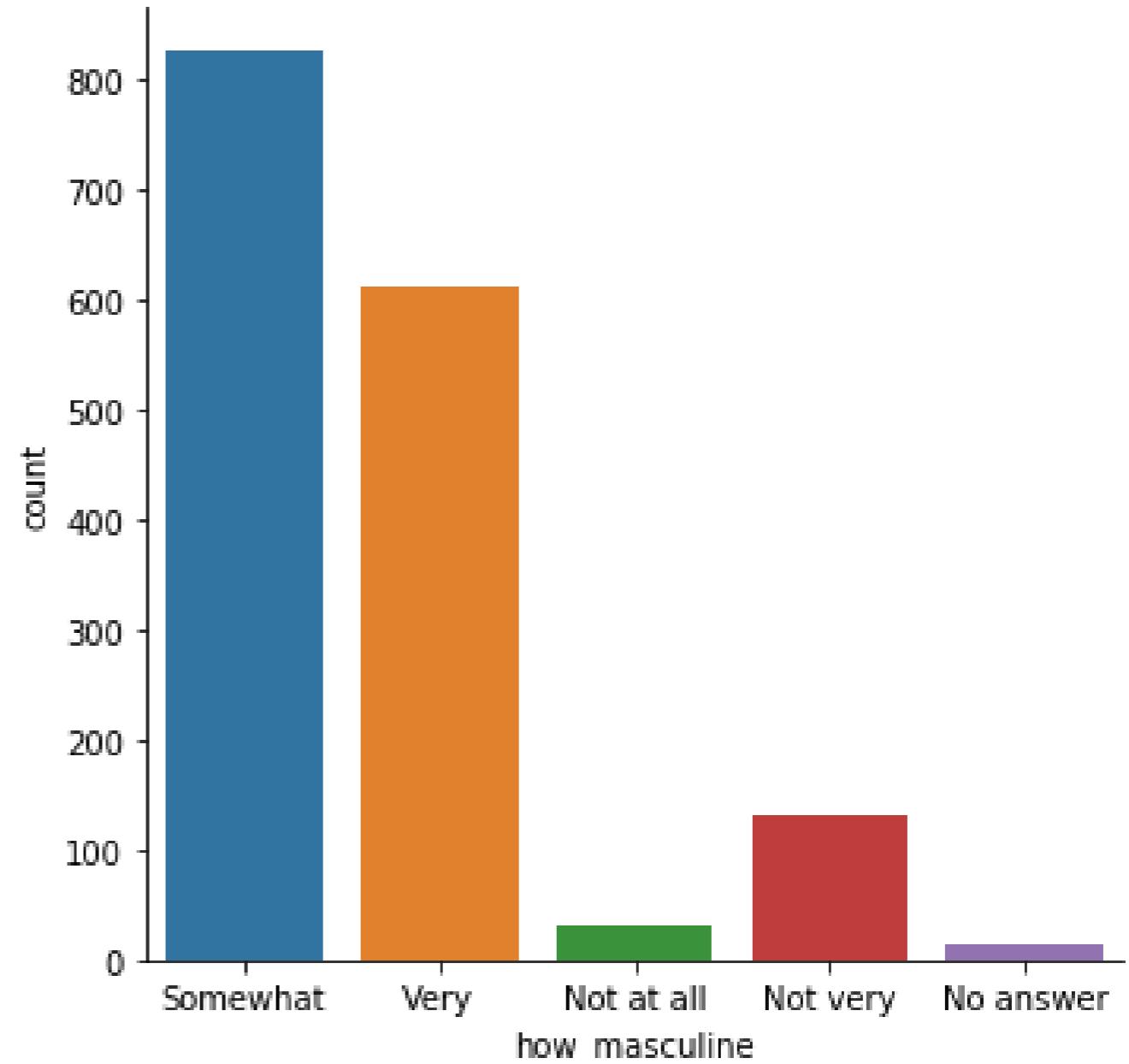
INTRODUCTION TO DATA VISUALIZATION WITH SEABORN



Erin Case
Data Scientist

Categorical plots

- Examples: count plots, bar plots
- Involve a categorical variable
- Comparisons between groups

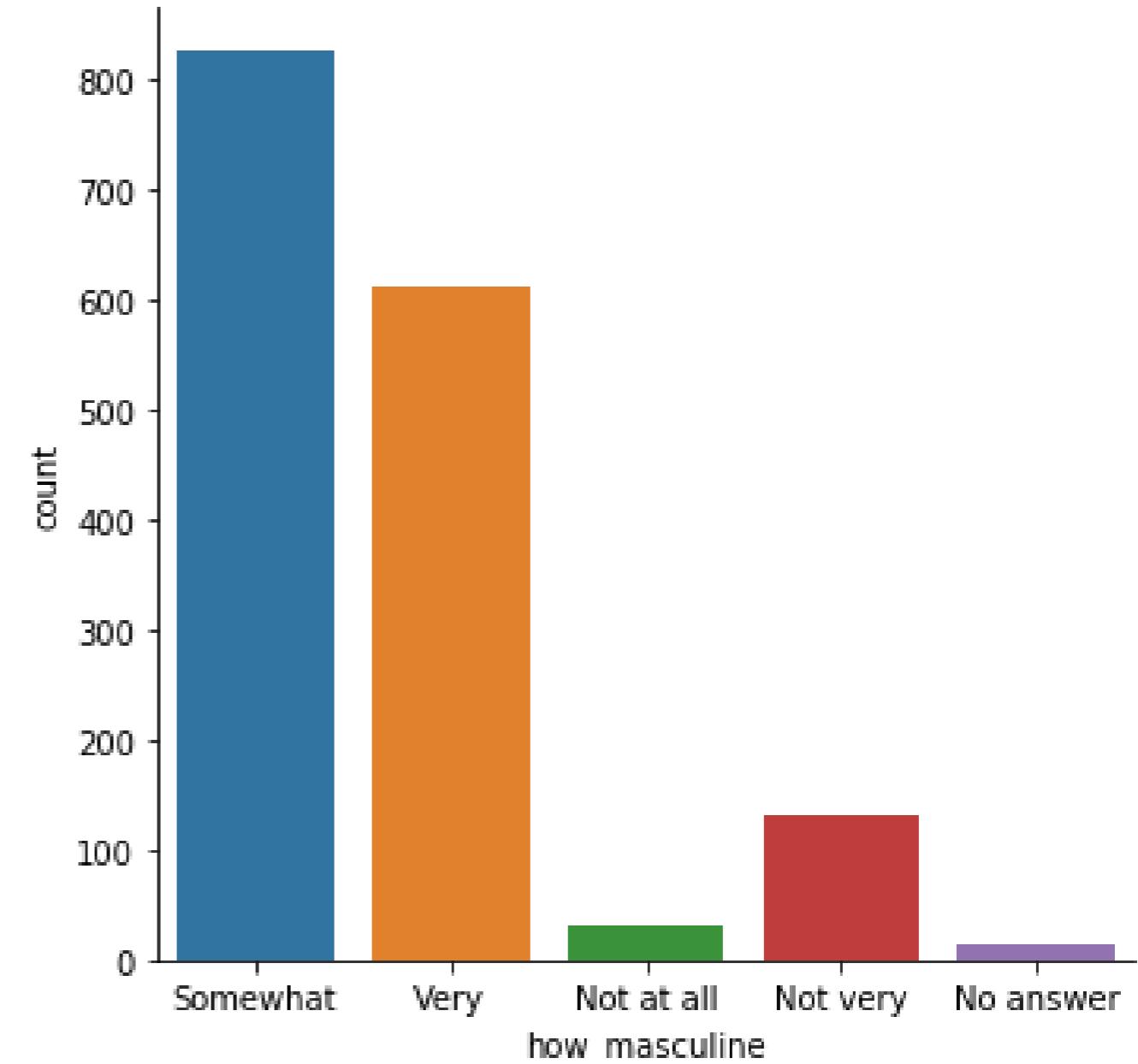


catplot()

- Used to create categorical plots
- Same advantages of `relplot()`
- Easily create subplots with `col=` and `row=`

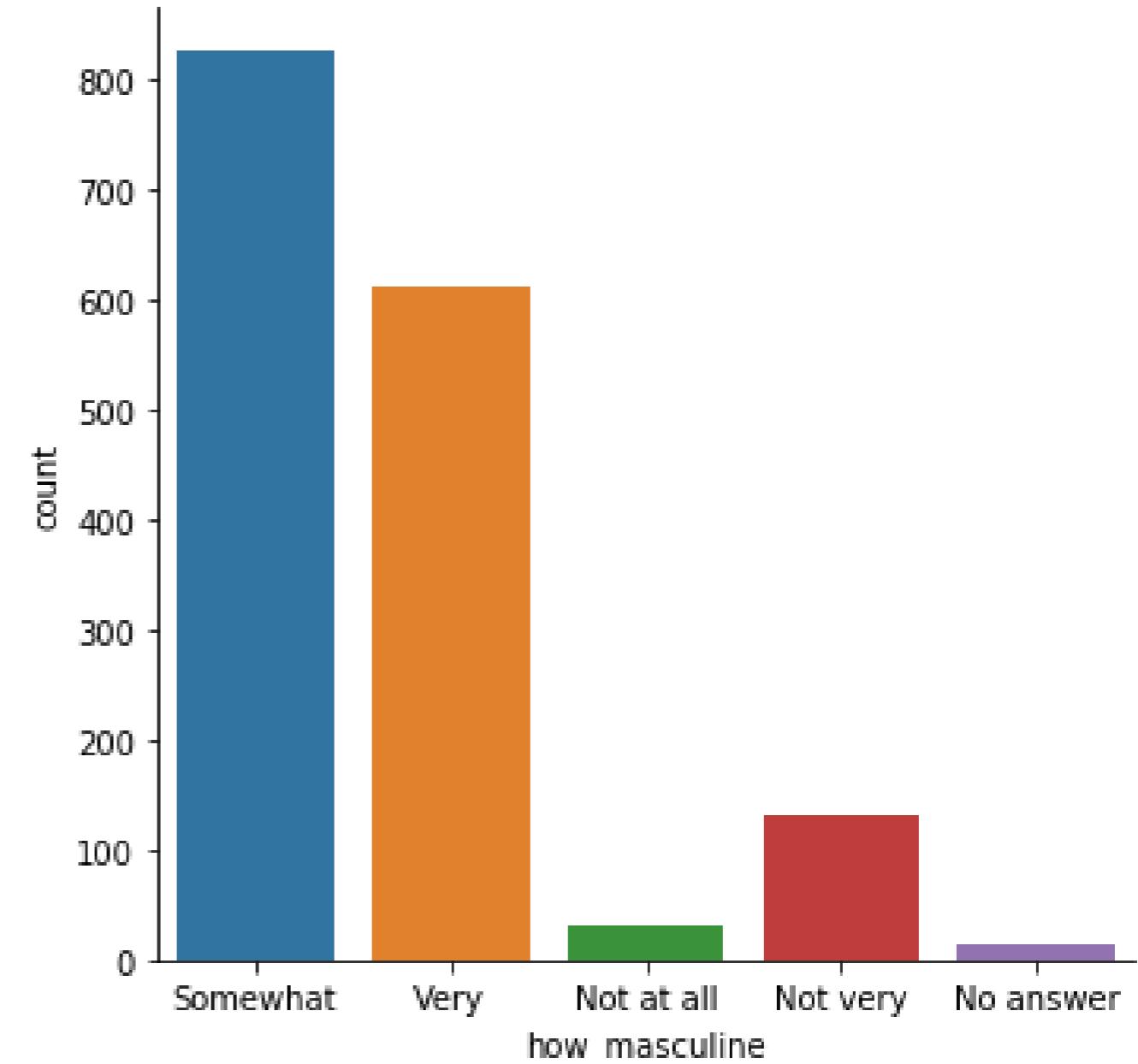
countplot() vs. catplot()

```
import matplotlib.pyplot as plt  
import seaborn as sns  
  
sns.countplot(x="how_masculine",  
               data=mascularity_data)  
  
plt.show()
```



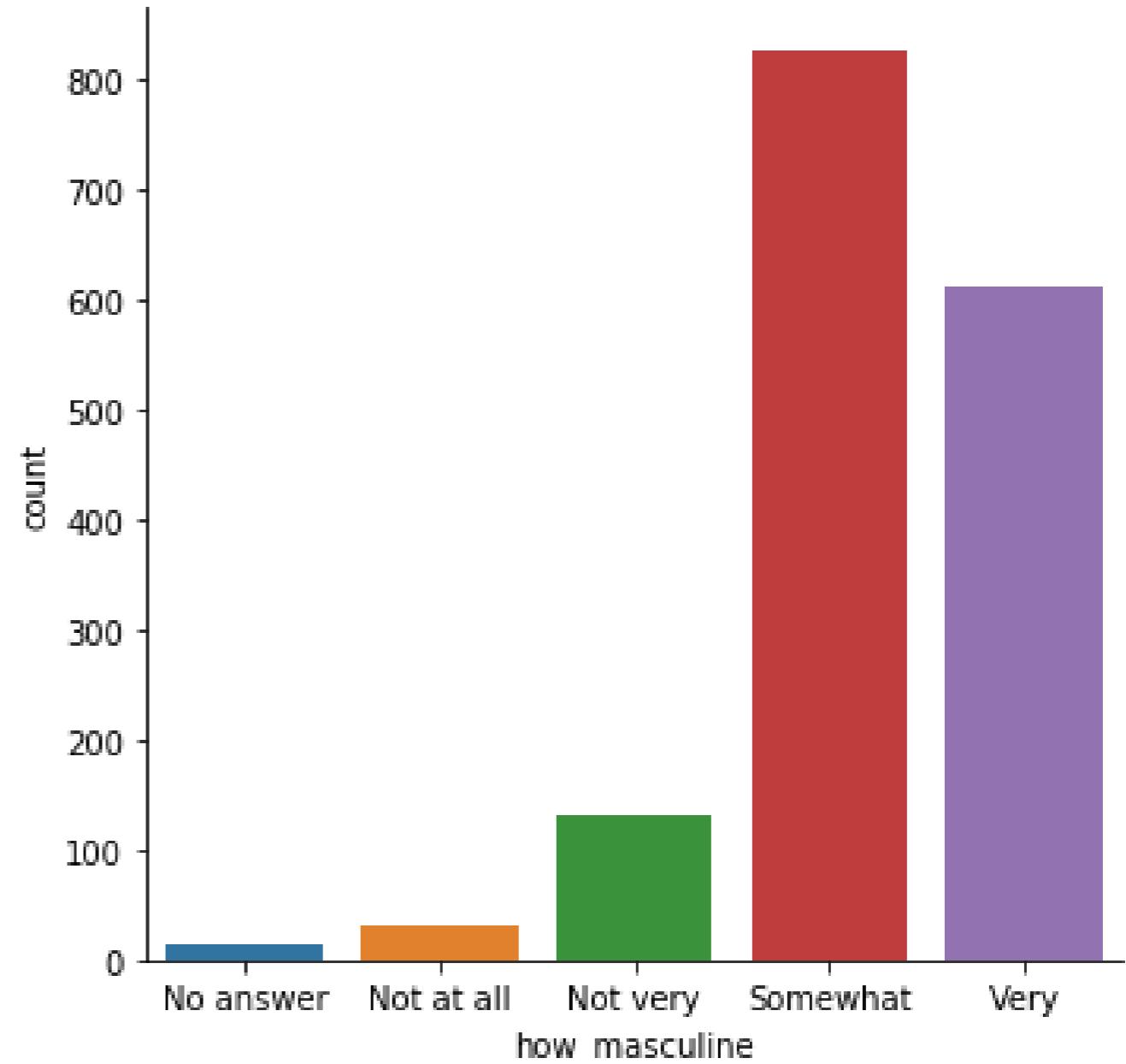
countplot() vs. catplot()

```
import matplotlib.pyplot as plt  
import seaborn as sns  
  
sns.catplot(x="how_masculine",  
             data=mascularity_data,  
             kind="count")  
  
plt.show()
```



Changing the order

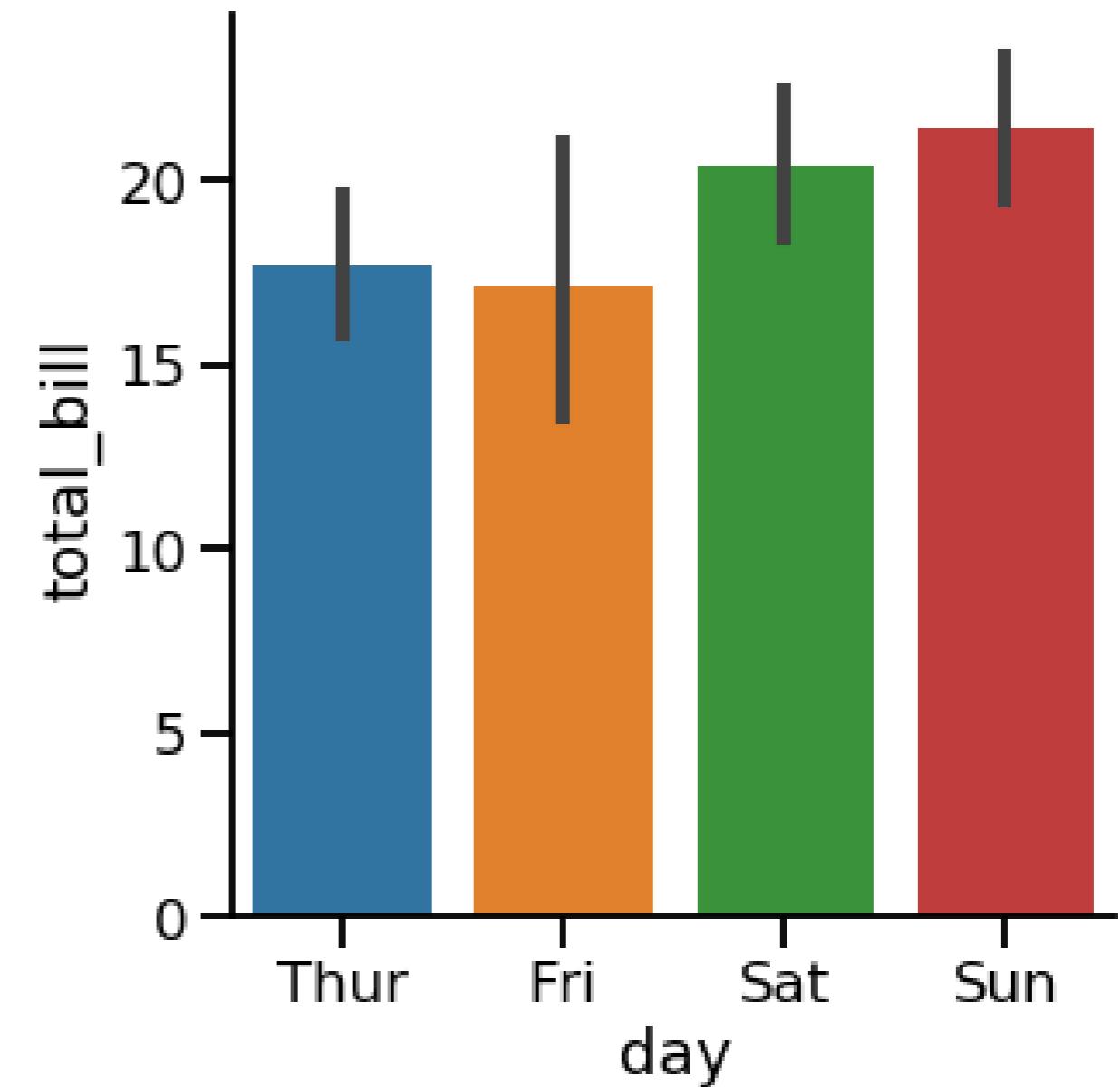
```
import matplotlib.pyplot as plt
import seaborn as sns
category_order = ["No answer",
                  "Not at all",
                  "Not very",
                  "Somewhat",
                  "Very"]
sns.catplot(x="how_masculine",
             data=masculinity_data,
             kind="count",
             order=category_order)
plt.show()
```



Bar plots

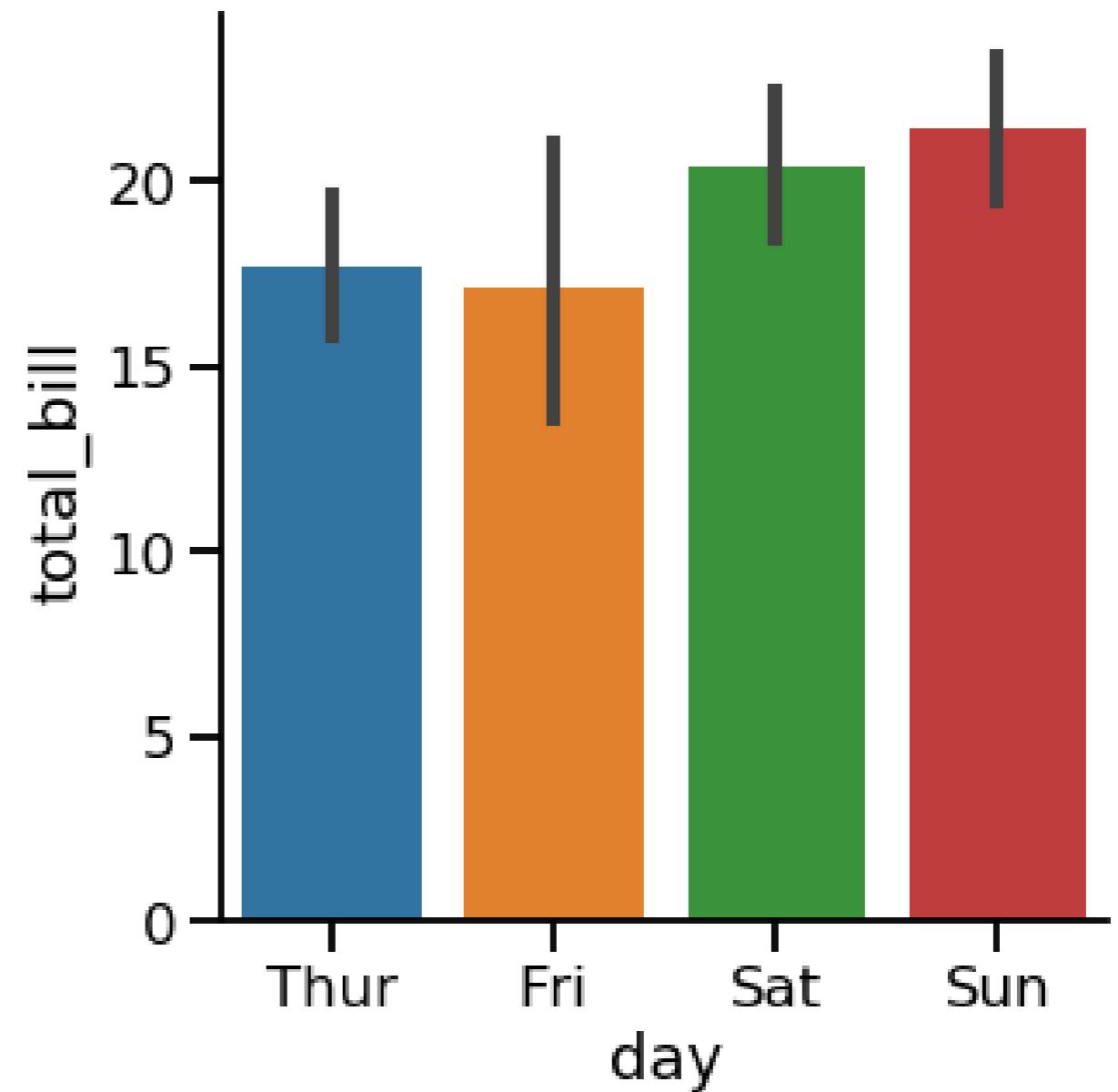
Displays mean of quantitative variable per category

```
import matplotlib.pyplot as plt  
import seaborn as sns  
  
sns.catplot(x="day",  
             y="total_bill",  
             data=tips,  
             kind="bar")  
  
plt.show()
```



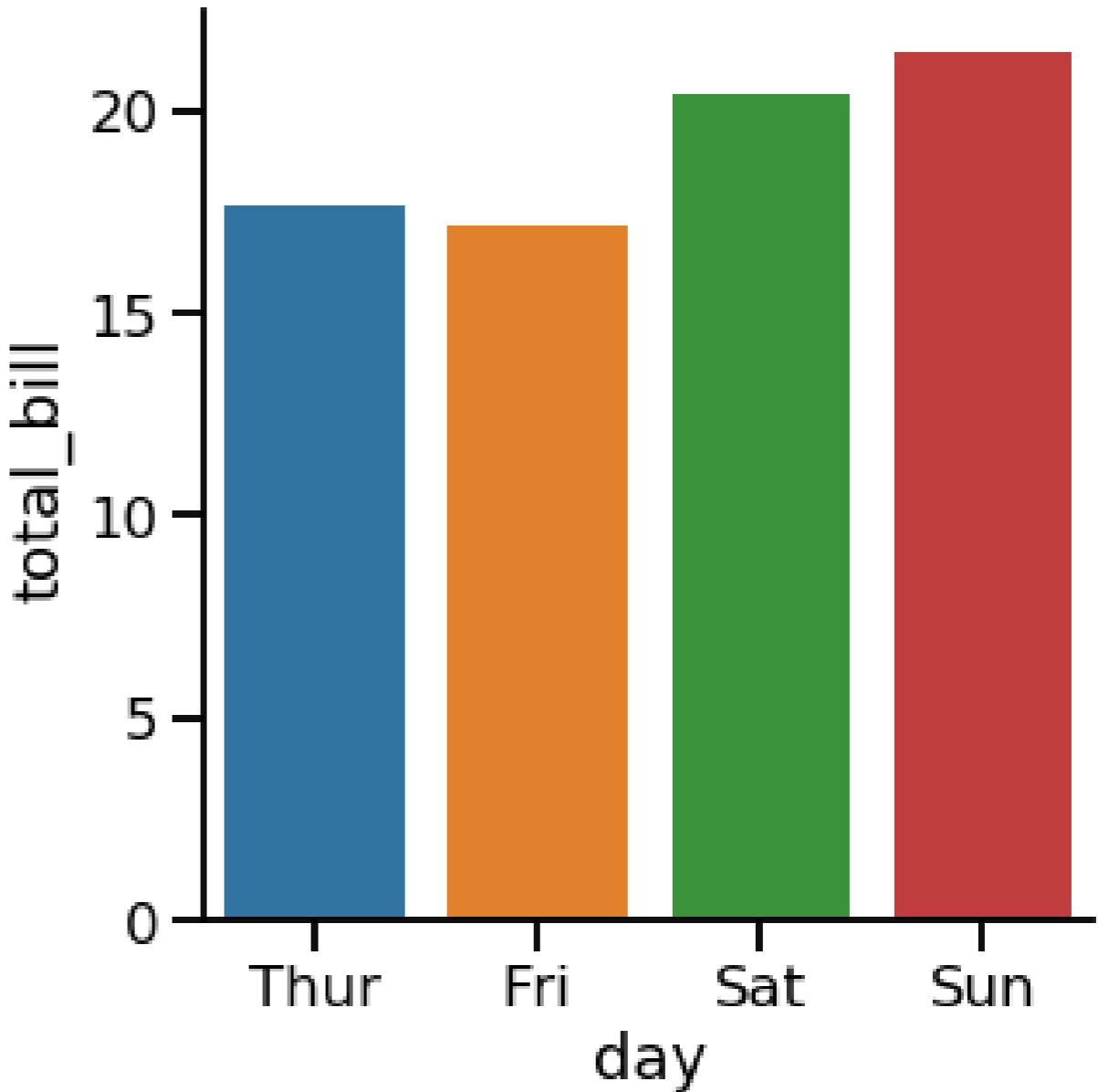
Confidence intervals

- Lines show 95% confidence intervals for the mean
- Shows uncertainty about our estimate
- Assumes our data is a random sample



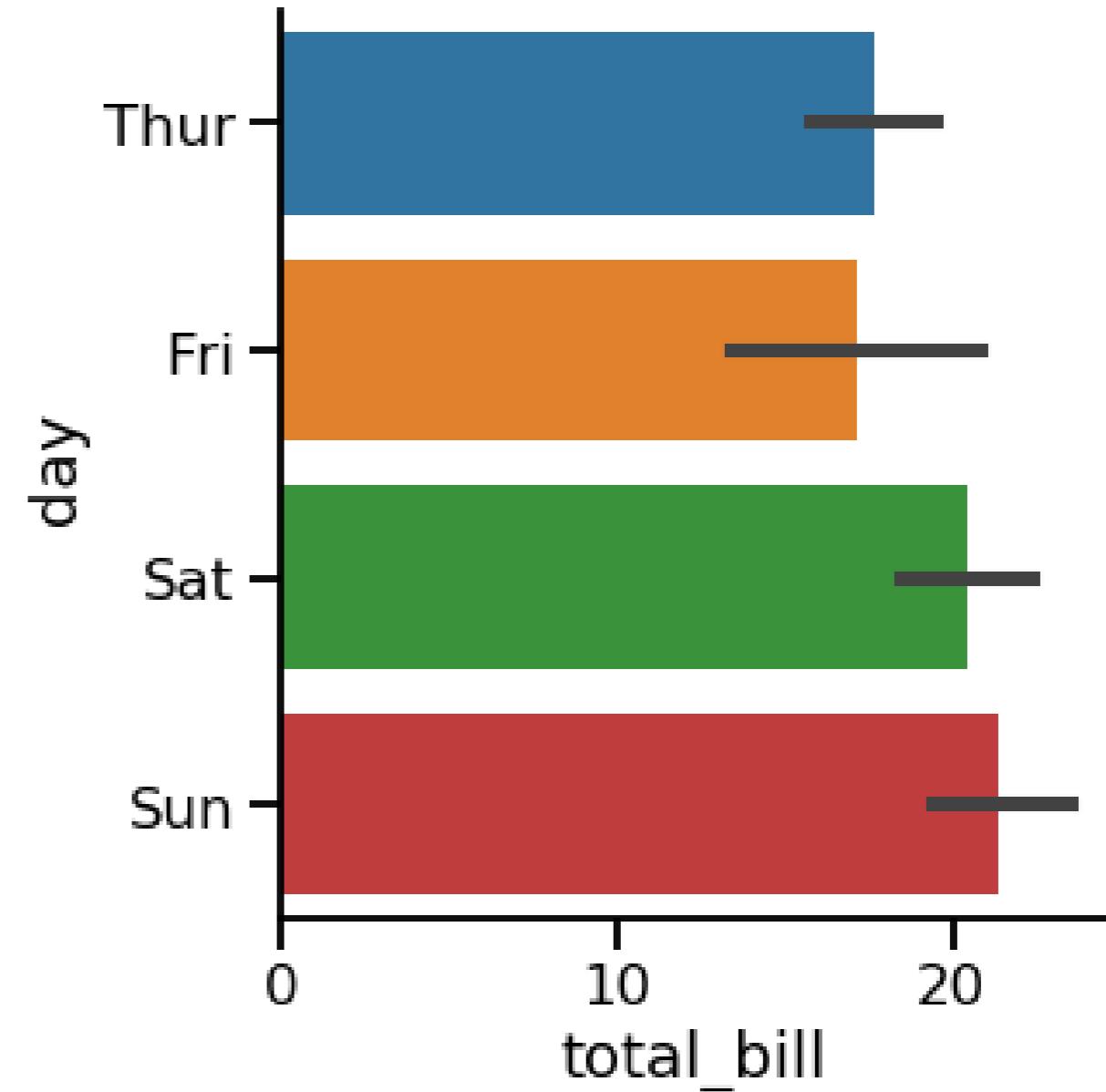
Turning off confidence intervals

```
import matplotlib.pyplot as plt  
import seaborn as sns  
  
sns.catplot(x="day",  
             y="total_bill",  
             data=tips,  
             kind="bar",  
             ci=None)  
  
plt.show()
```



Changing the orientation

```
import matplotlib.pyplot as plt  
import seaborn as sns  
  
sns.catplot(x="total_bill",  
             y="day",  
             data=tips,  
             kind="bar")  
  
plt.show()
```

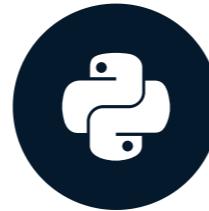


Let's practice!

INTRODUCTION TO DATA VISUALIZATION WITH SEABORN

Creating a box plot

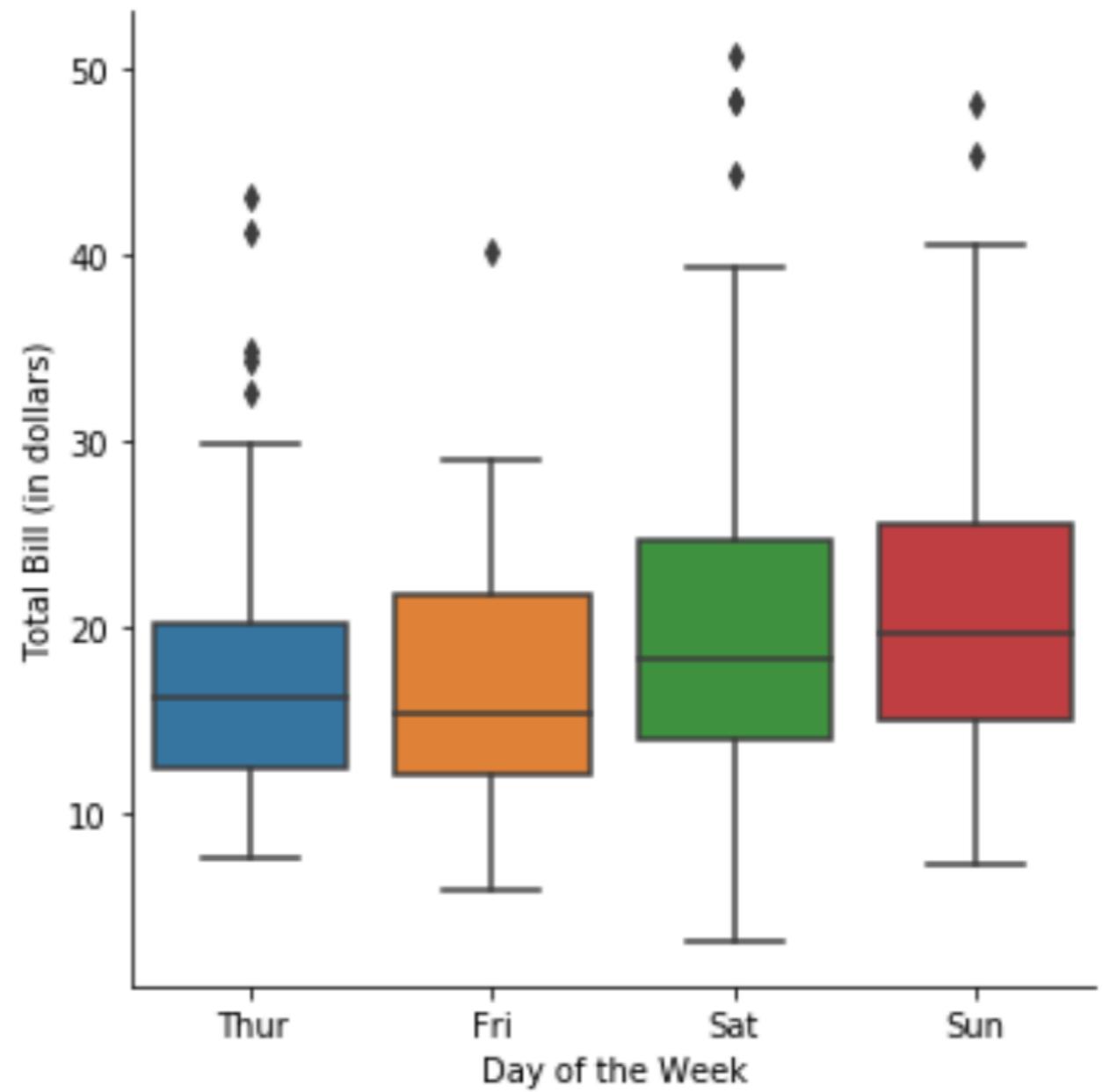
INTRODUCTION TO DATA VISUALIZATION WITH SEABORN



Erin Case
Data Scientist

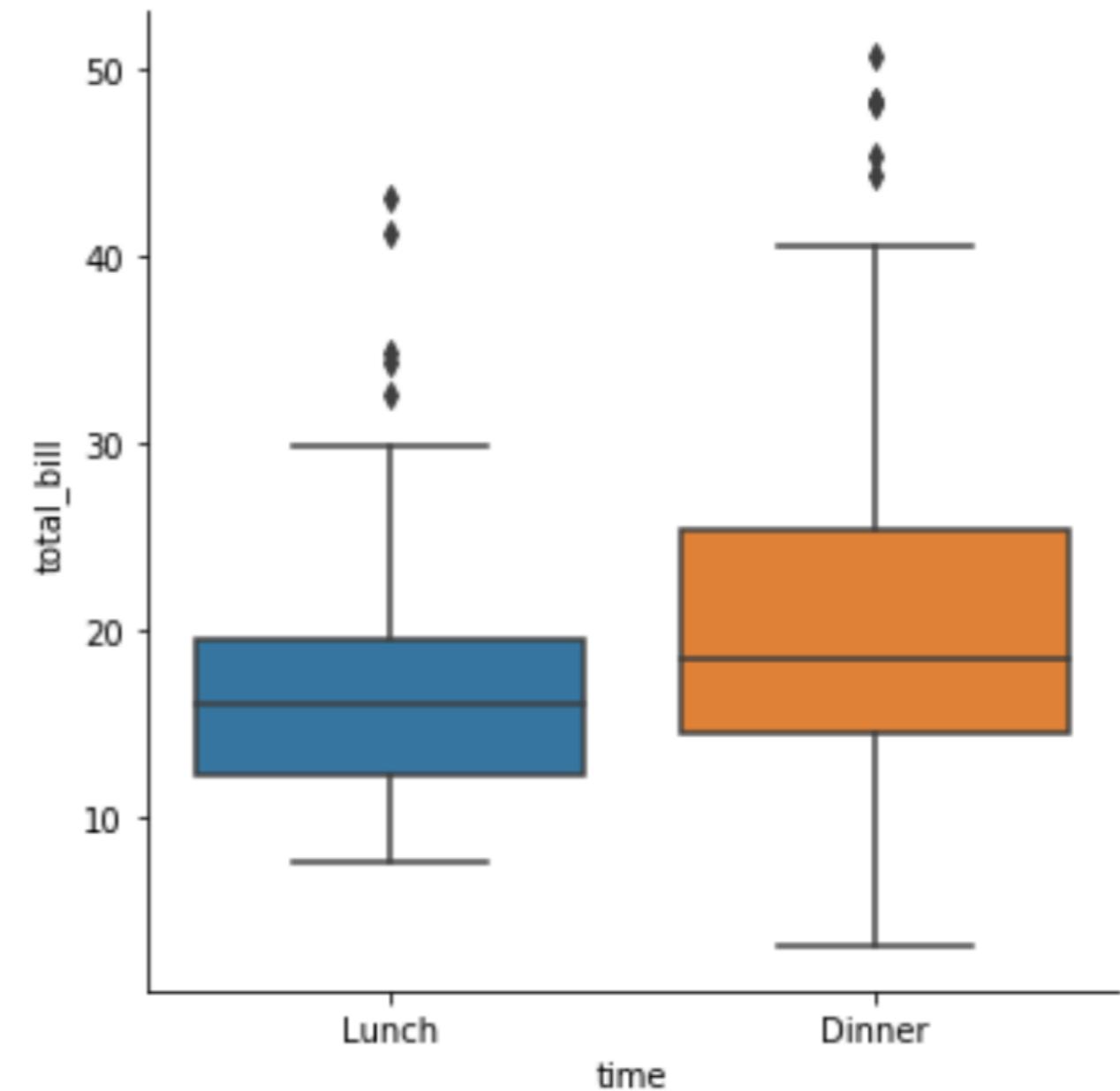
What is a box plot?

- Shows the distribution of quantitative data
- See median, spread, skewness, and outliers
- Facilitates comparisons between groups



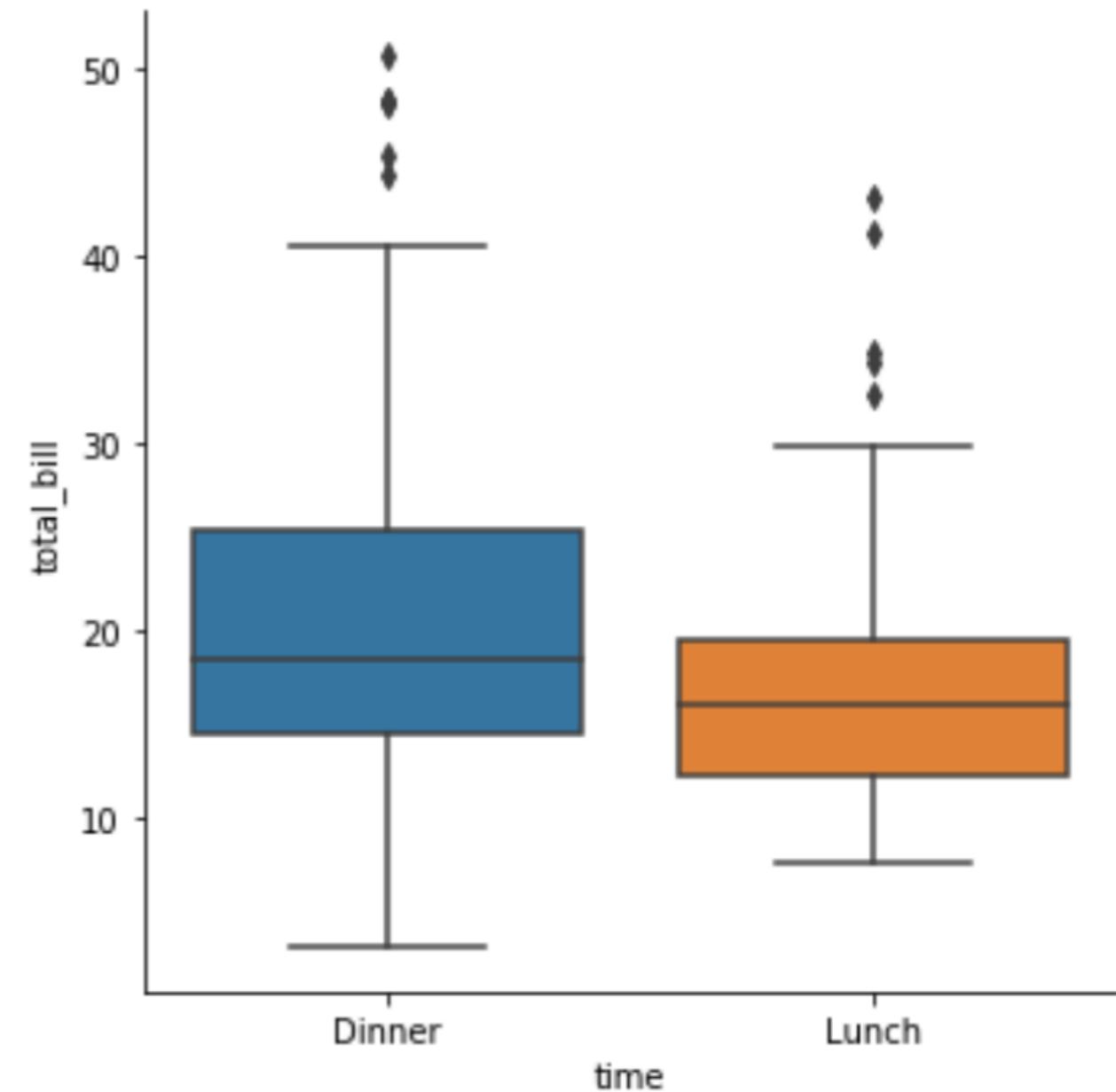
How to create a box plot

```
import matplotlib.pyplot as plt  
import seaborn as sns  
  
g = sns.catplot(x="time",  
                 y="total_bill",  
                 data=tips,  
                 kind="box")  
  
plt.show()
```



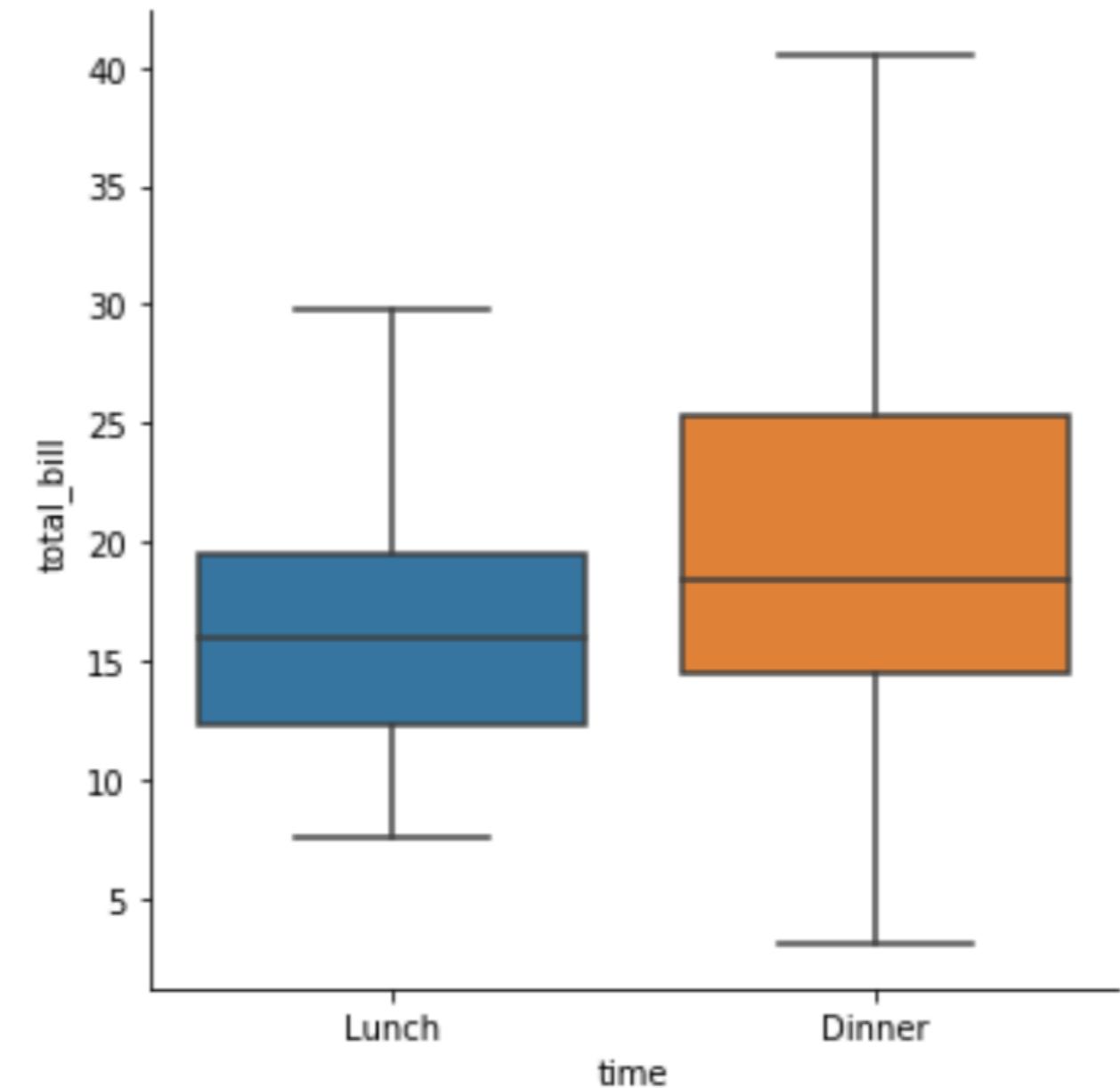
Change the order of categories

```
import matplotlib.pyplot as plt  
import seaborn as sns  
  
g = sns.catplot(x="time",  
                 y="total_bill",  
                 data=tips,  
                 kind="box",  
                 order=["Dinner",  
                        "Lunch"])  
  
plt.show()
```



Omitting the outliers using 'sym'

```
import matplotlib.pyplot as plt  
import seaborn as sns  
  
g = sns.catplot(x="time",  
                 y="total_bill",  
                 data=tips,  
                 kind="box",  
                 sym="")  
  
plt.show()
```

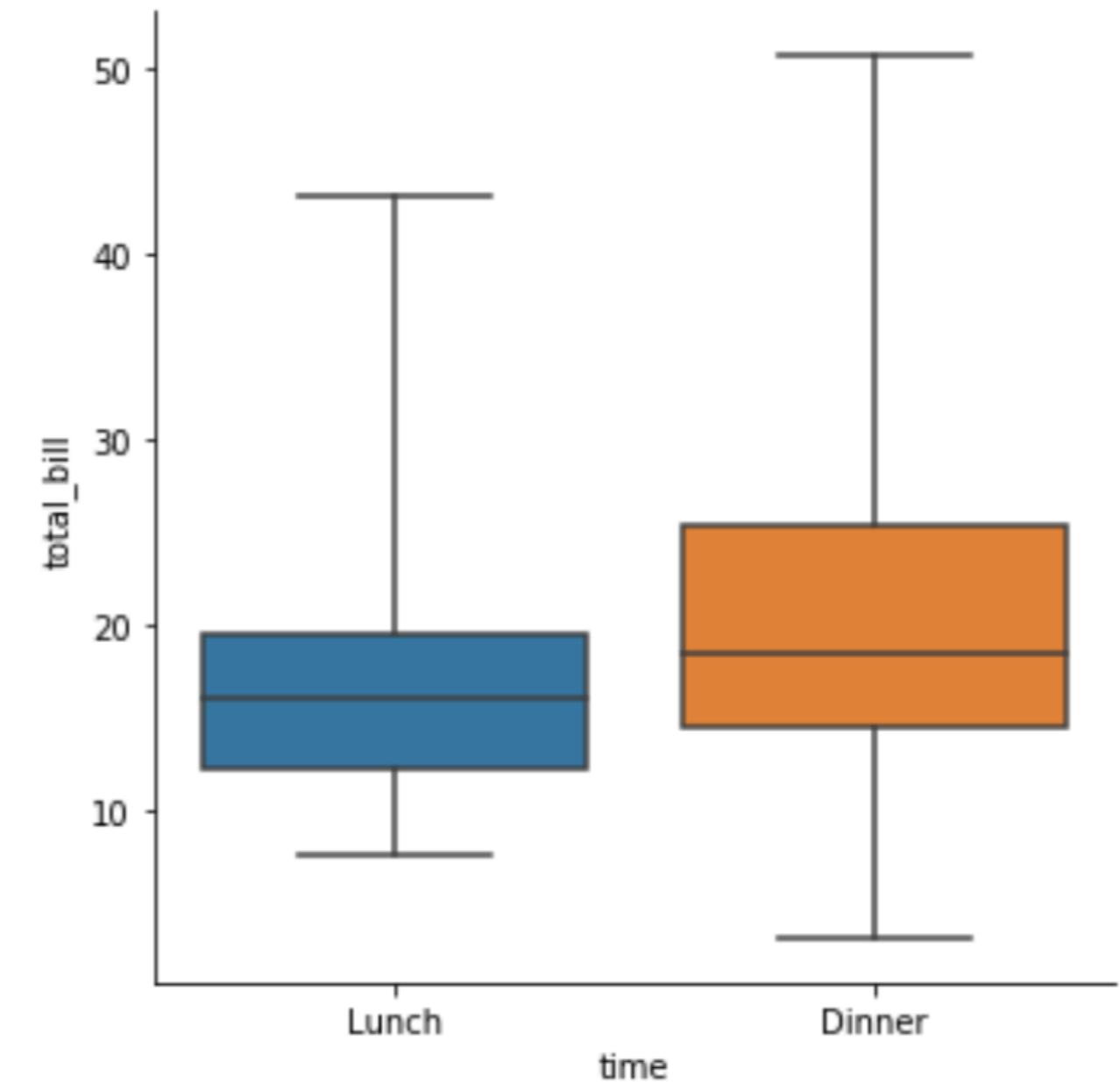


Changing the whiskers using `whis`

- By default, the whiskers extend to $1.5 * \text{the interquartile range}$
- Make them extend to $2.0 * \text{IQR}$: `whis=2.0`
- Show the 5th and 95th percentiles: `whis=[5, 95]`
- Show min and max values: `whis=[0, 100]`

Changing the whiskers using `whis`

```
import matplotlib.pyplot as plt  
import seaborn as sns  
  
g = sns.catplot(x="time",  
                 y="total_bill",  
                 data=tips,  
                 kind="box",  
                 whis=[0, 100])  
  
plt.show()
```

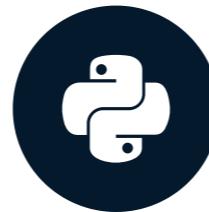


Let's practice!

INTRODUCTION TO DATA VISUALIZATION WITH SEABORN

Point plots

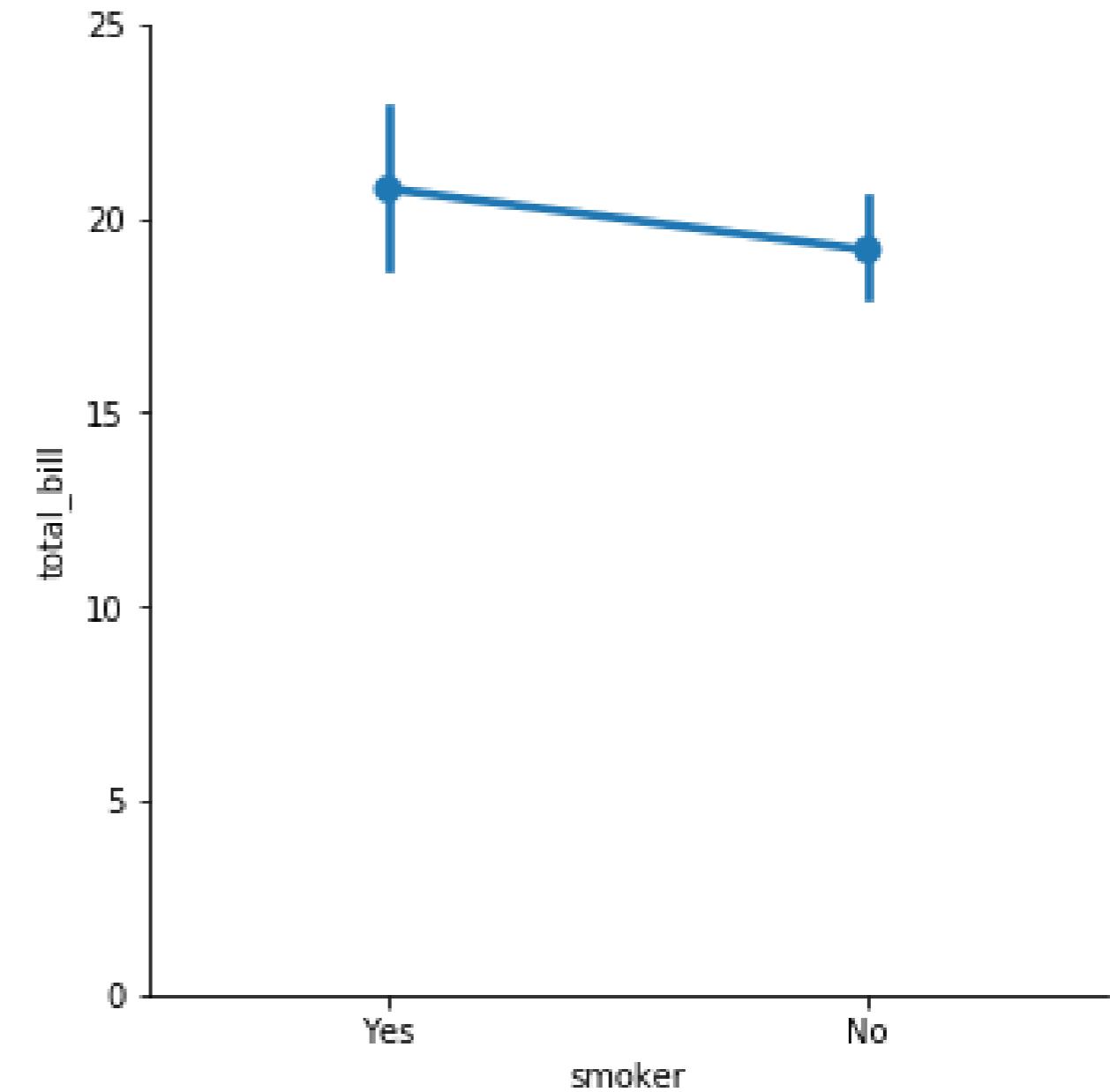
INTRODUCTION TO DATA VISUALIZATION WITH SEABORN



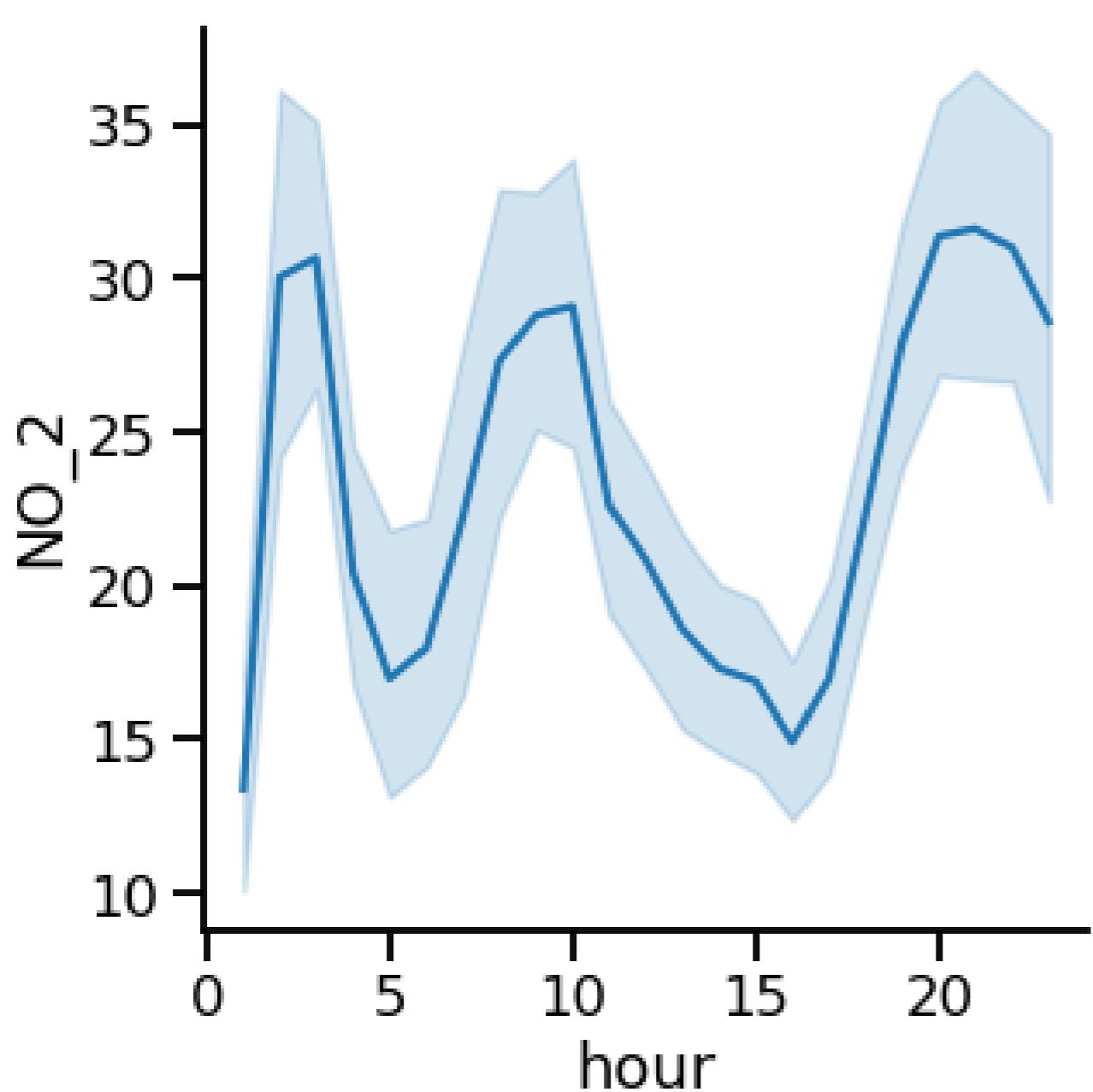
Erin Case
Data Scientist

What are point plots?

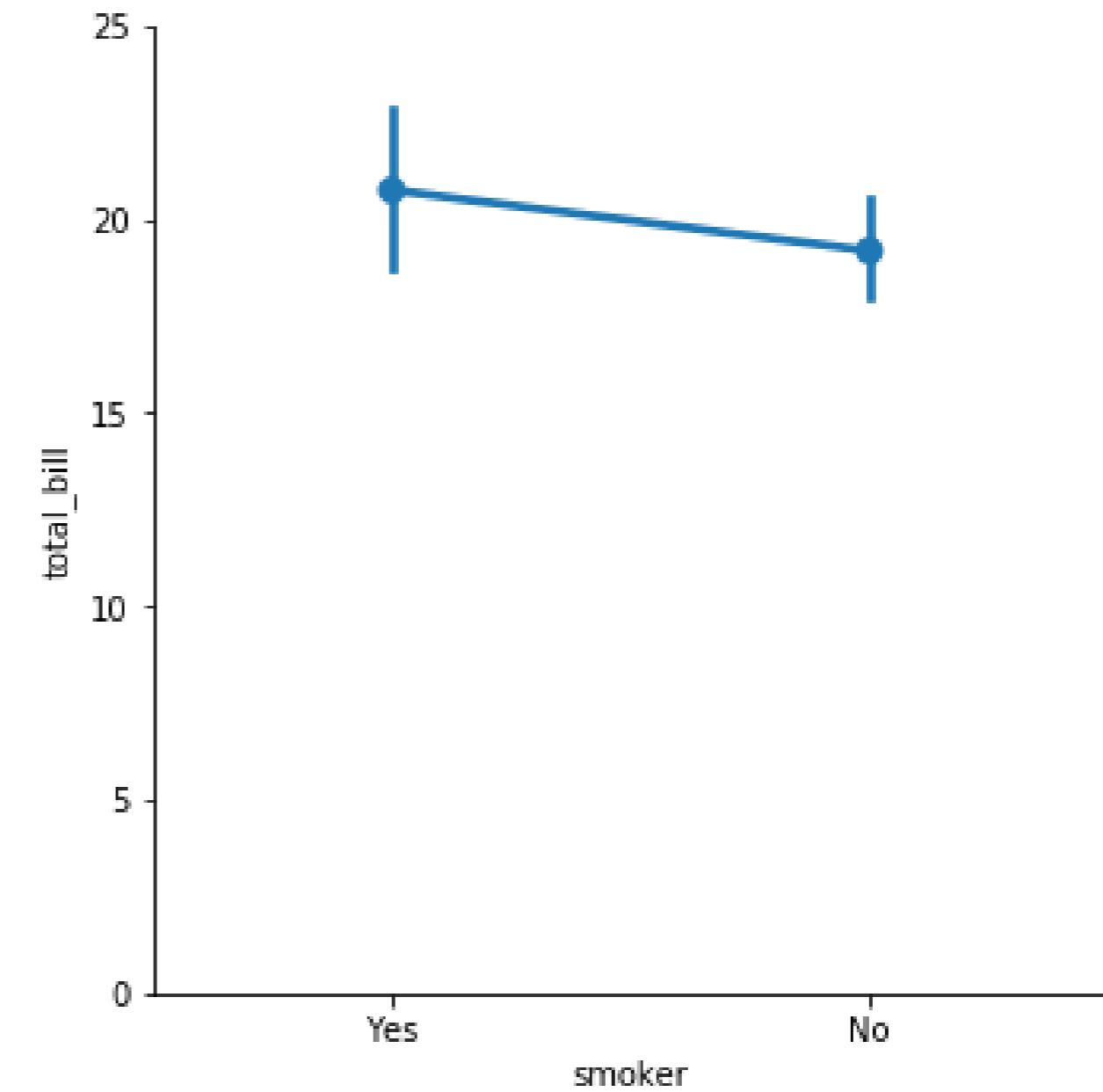
- Points show mean of quantitative variable
- Vertical lines show 95% confidence intervals



Line plot: average level of nitrogen dioxide over time



Point plot: average restaurant bill, smokers vs. non-smokers



Point plots vs. line plots

Both show:

- Mean of quantitative variable
- 95% confidence intervals for the mean

Differences:

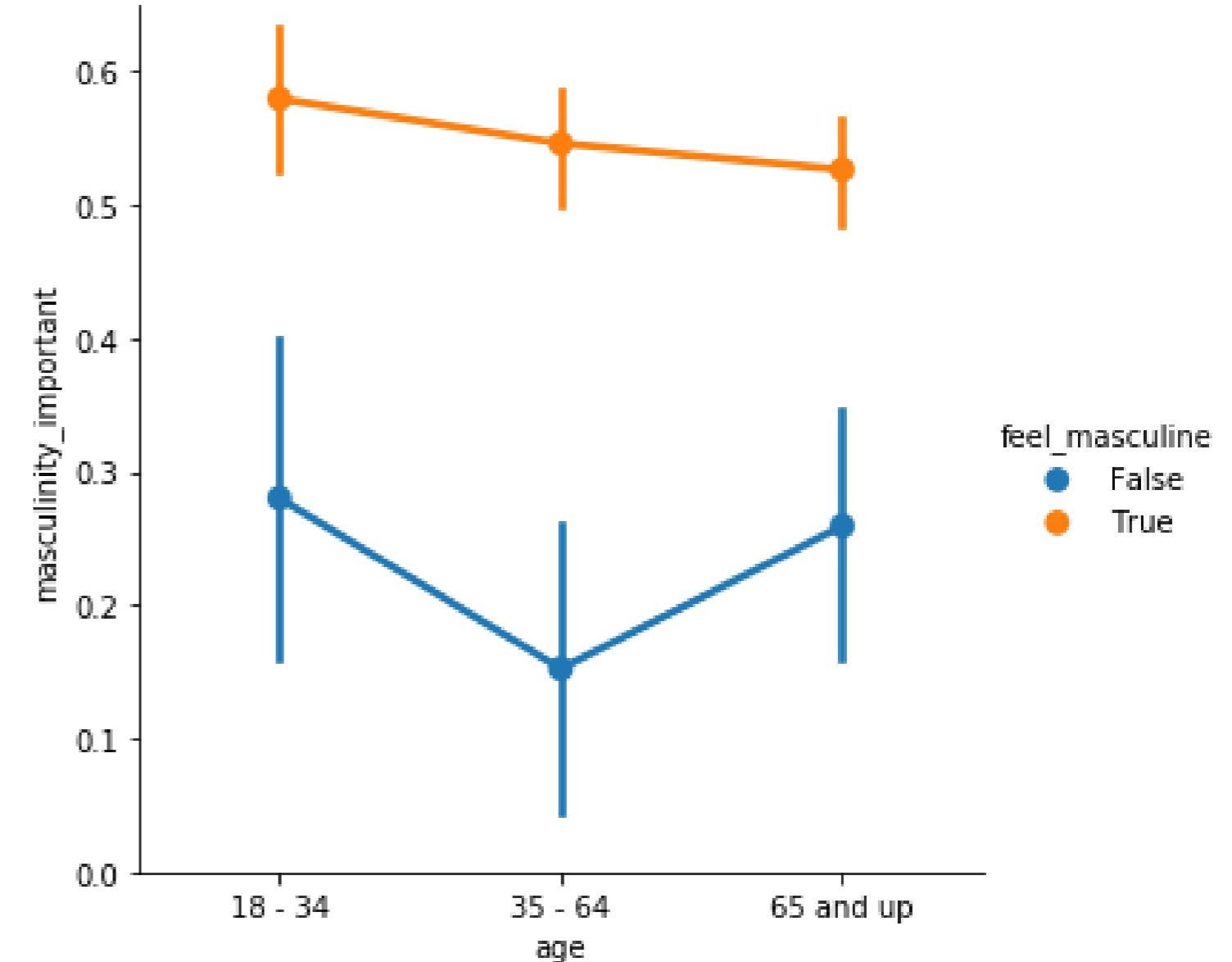
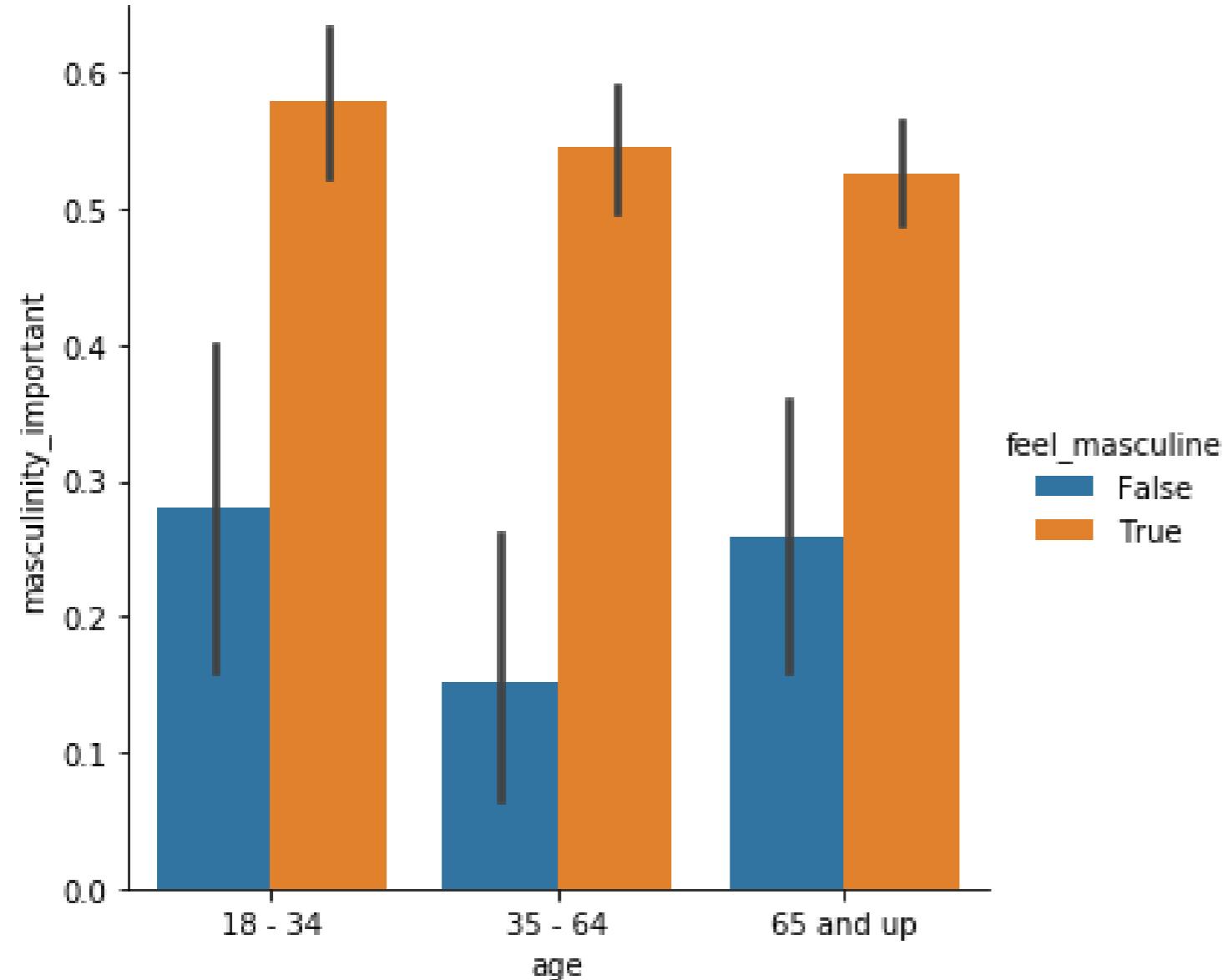
- Line plot has **quantitative** variable (usually time) on x-axis
- Point plot has **categorical** variable on x-axis

Point plots vs. bar plots

Both show:

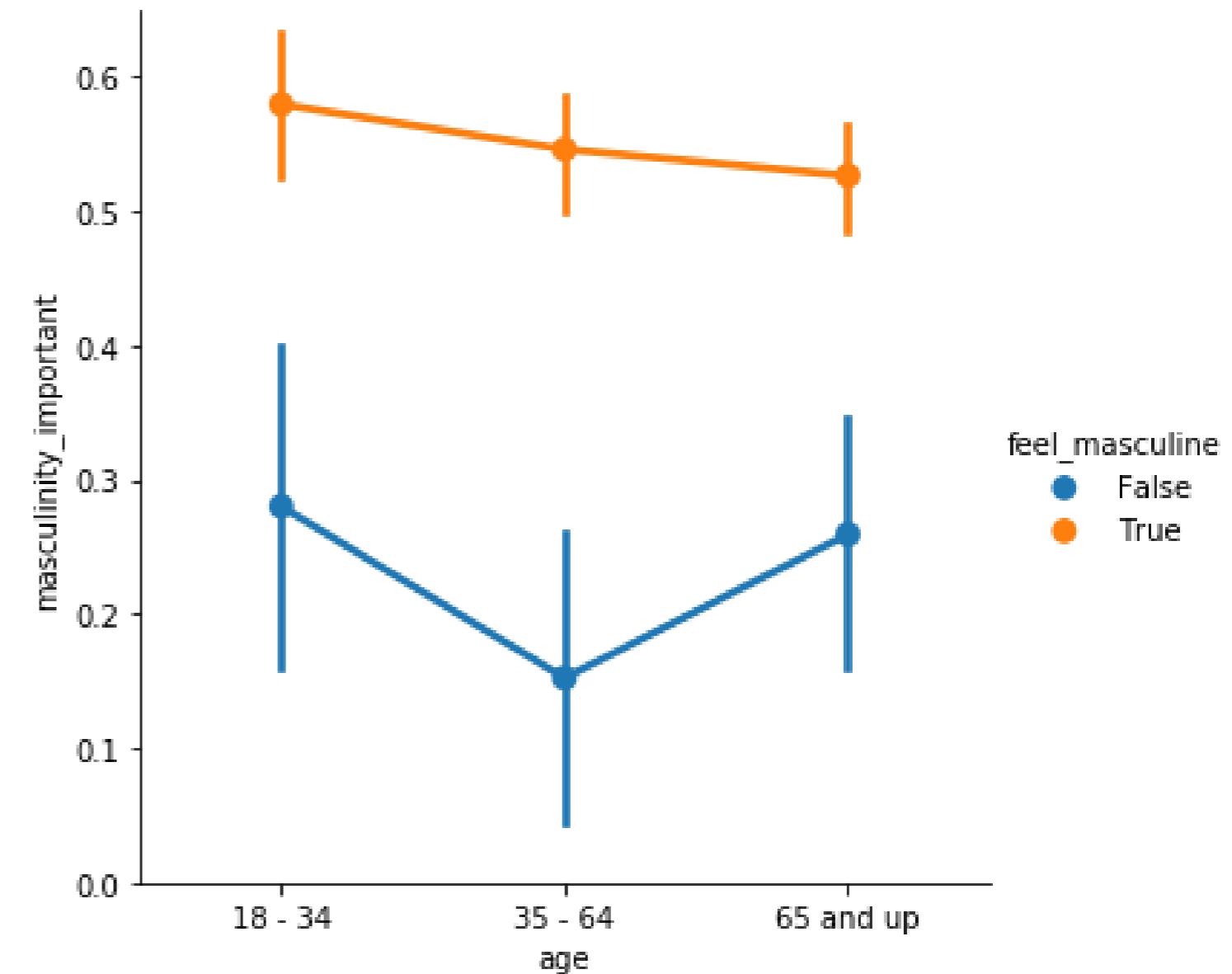
- Mean of quantitative variable
- 95% confidence intervals for the mean

Point plots vs. bar plots



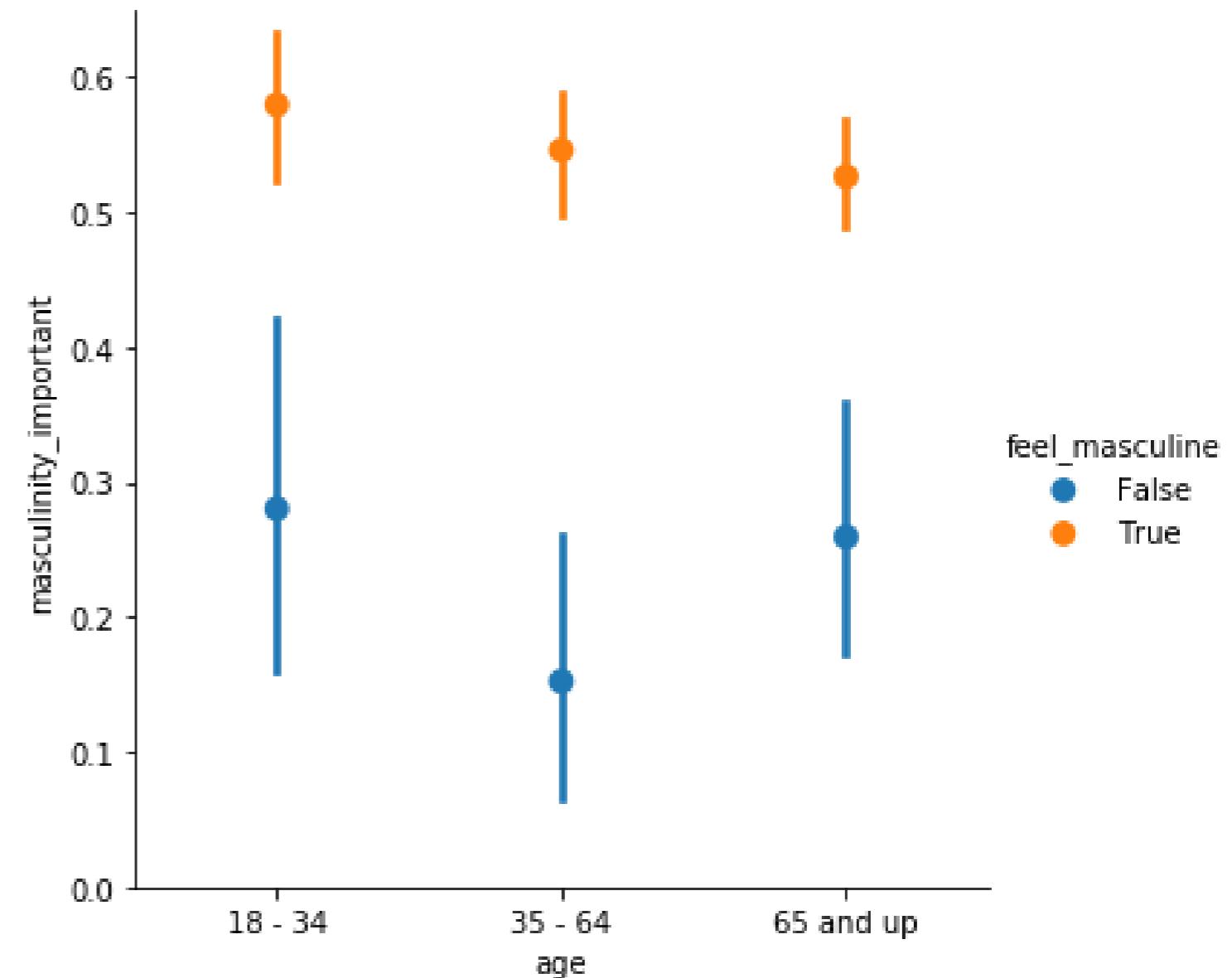
Creating a point plot

```
import matplotlib.pyplot as plt  
import seaborn as sns  
  
sns.catplot(x="age",  
             y="masculinity_important",  
             data=mascularity_data,  
             hue="feel_masculine",  
             kind="point")  
  
plt.show()
```



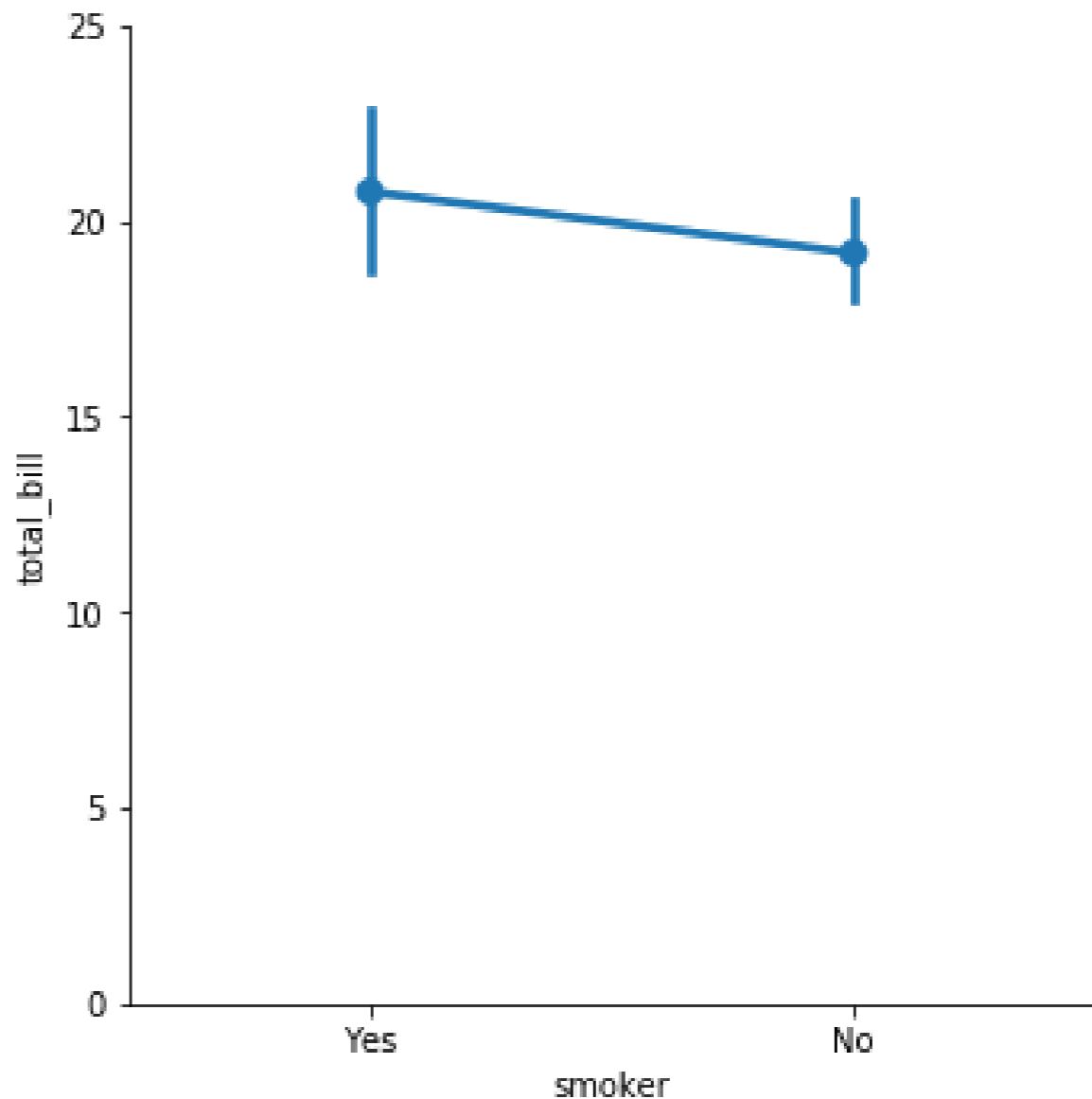
Disconnecting the points

```
import matplotlib.pyplot as plt  
import seaborn as sns  
  
sns.catplot(x="age",  
             y="masculinity_important",  
             data=mascularity_data,  
             hue="feel_masculine",  
             kind="point",  
             join=False)  
  
plt.show()
```



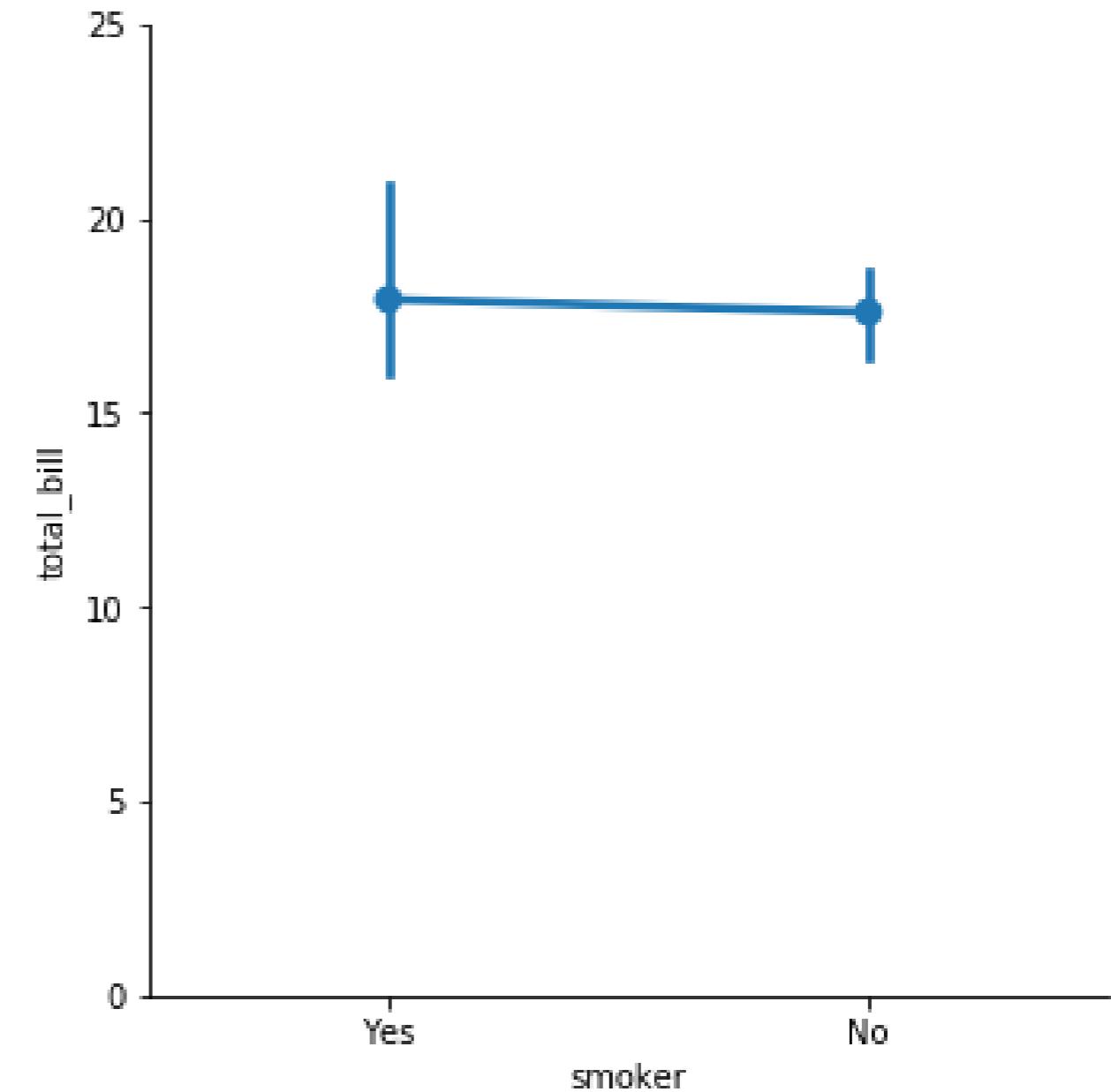
Displaying the median

```
import matplotlib.pyplot as plt  
import seaborn as sns  
  
sns.catplot(x="smoker",  
             y="total_bill",  
             data=tips,  
             kind="point")  
  
plt.show()
```



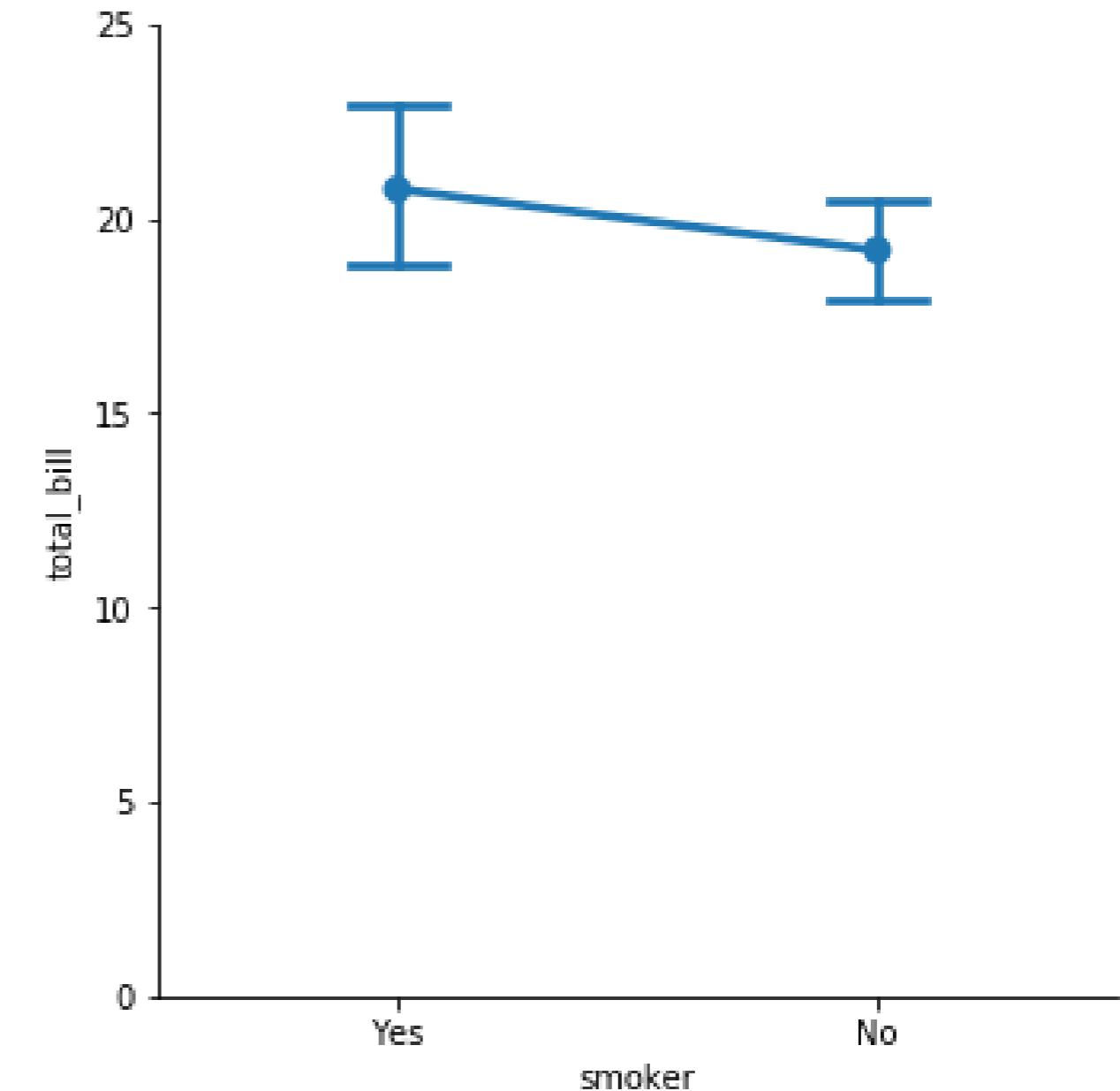
Displaying the median

```
import matplotlib.pyplot as plt  
import seaborn as sns  
from numpy import median  
  
sns.catplot(x="smoker",  
             y="total_bill",  
             data=tips,  
             kind="point",  
             estimator=median)  
  
plt.show()
```



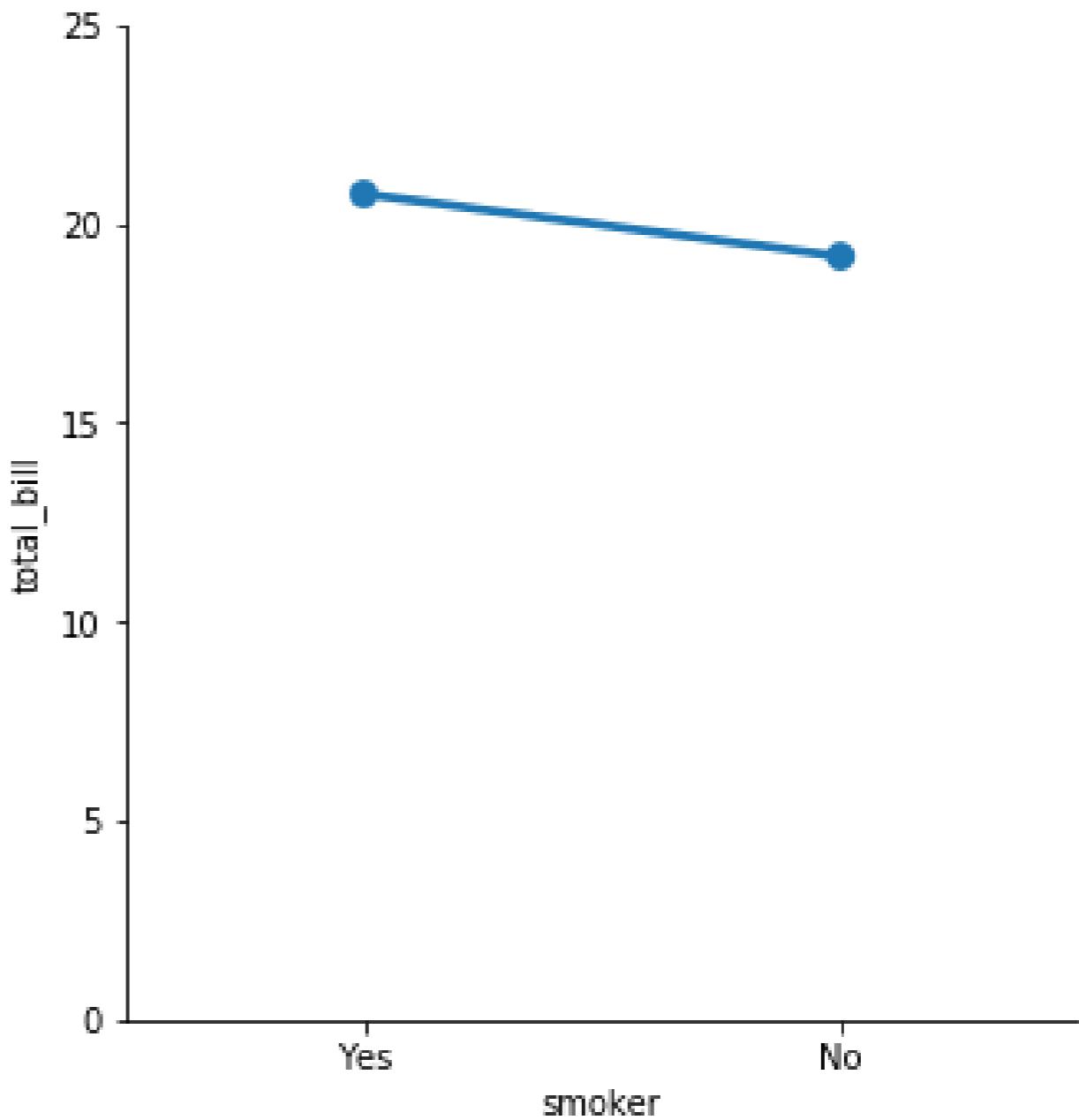
Customizing the confidence intervals

```
import matplotlib.pyplot as plt  
import seaborn as sns  
  
sns.catplot(x="smoker",  
             y="total_bill",  
             data=tips,  
             kind="point",  
             capsize=0.2)  
  
plt.show()
```



Turning off confidence intervals

```
import matplotlib.pyplot as plt  
import seaborn as sns  
  
sns.catplot(x="smoker",  
             y="total_bill",  
             data=tips,  
             kind="point",  
             ci=None)  
  
plt.show()
```

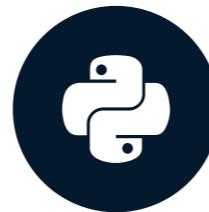


Let's practice!

INTRODUCTION TO DATA VISUALIZATION WITH SEABORN

Changing plot style and color

INTRODUCTION TO DATA VISUALIZATION WITH SEABORN



Erin Case
Data Scientist

Why customize?

Reasons to change style:

- Personal preference
- Improve readability
- Guide interpretation

Changing the figure style

- Figure "style" includes background and axes
- Preset options: "white", "dark", "whitegrid", "darkgrid", "ticks"
- `sns.set_style()`

Default figure style ("white")

```
sns.catplot(x="age",  
            y="masculinity_important",  
            data=mascularity_data,  
            hue="feel_masculine",  
            kind="point")  
  
plt.show()
```

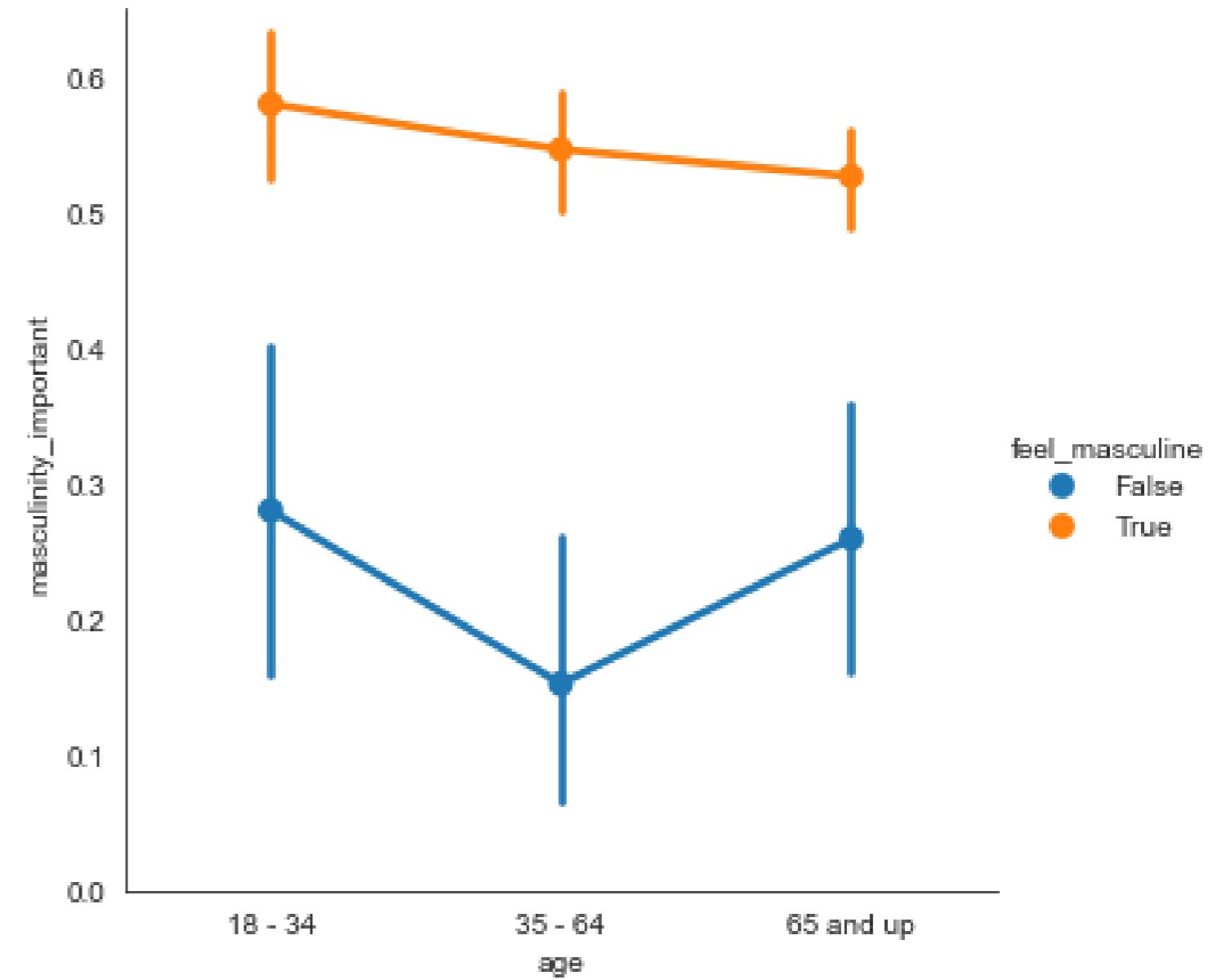
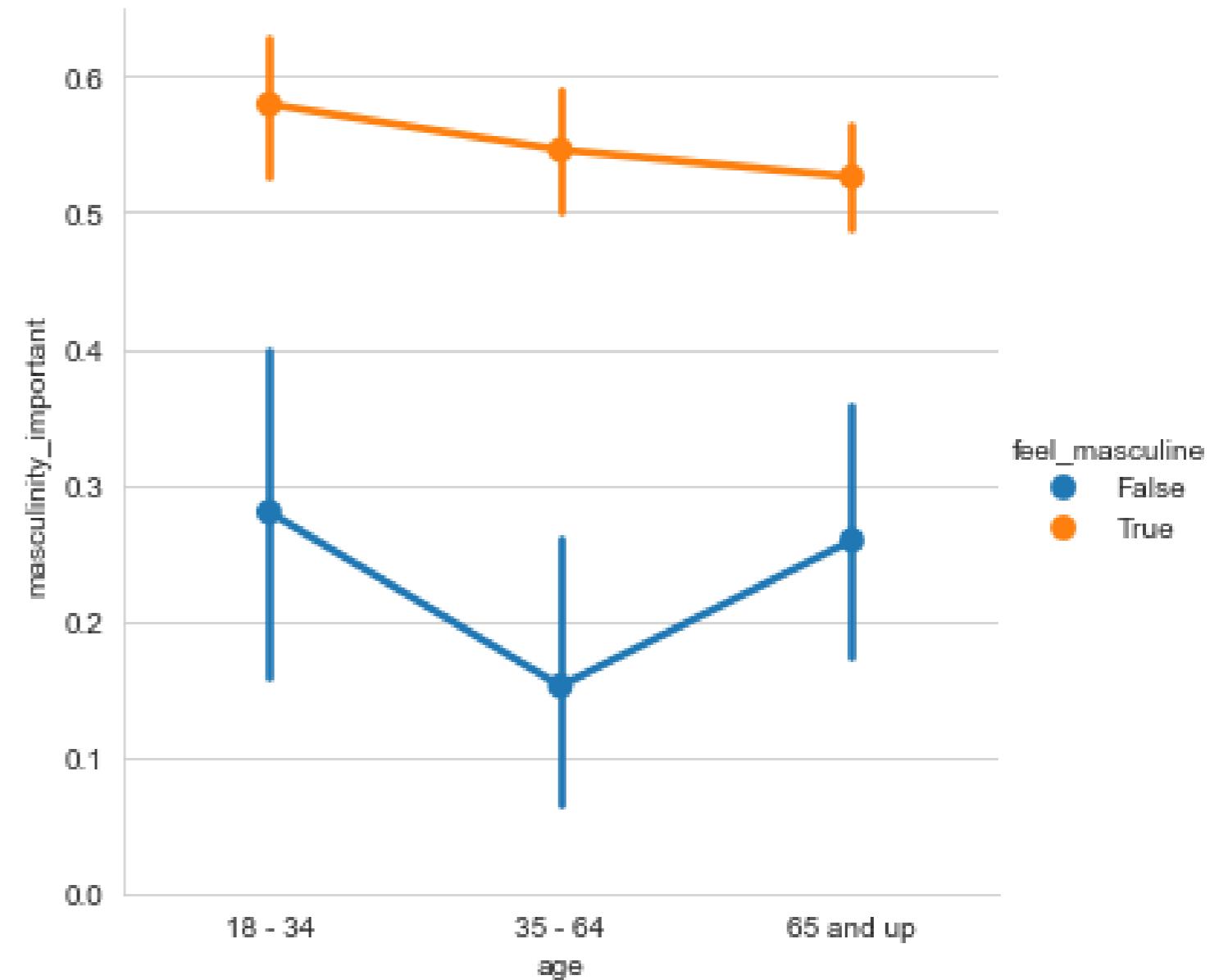


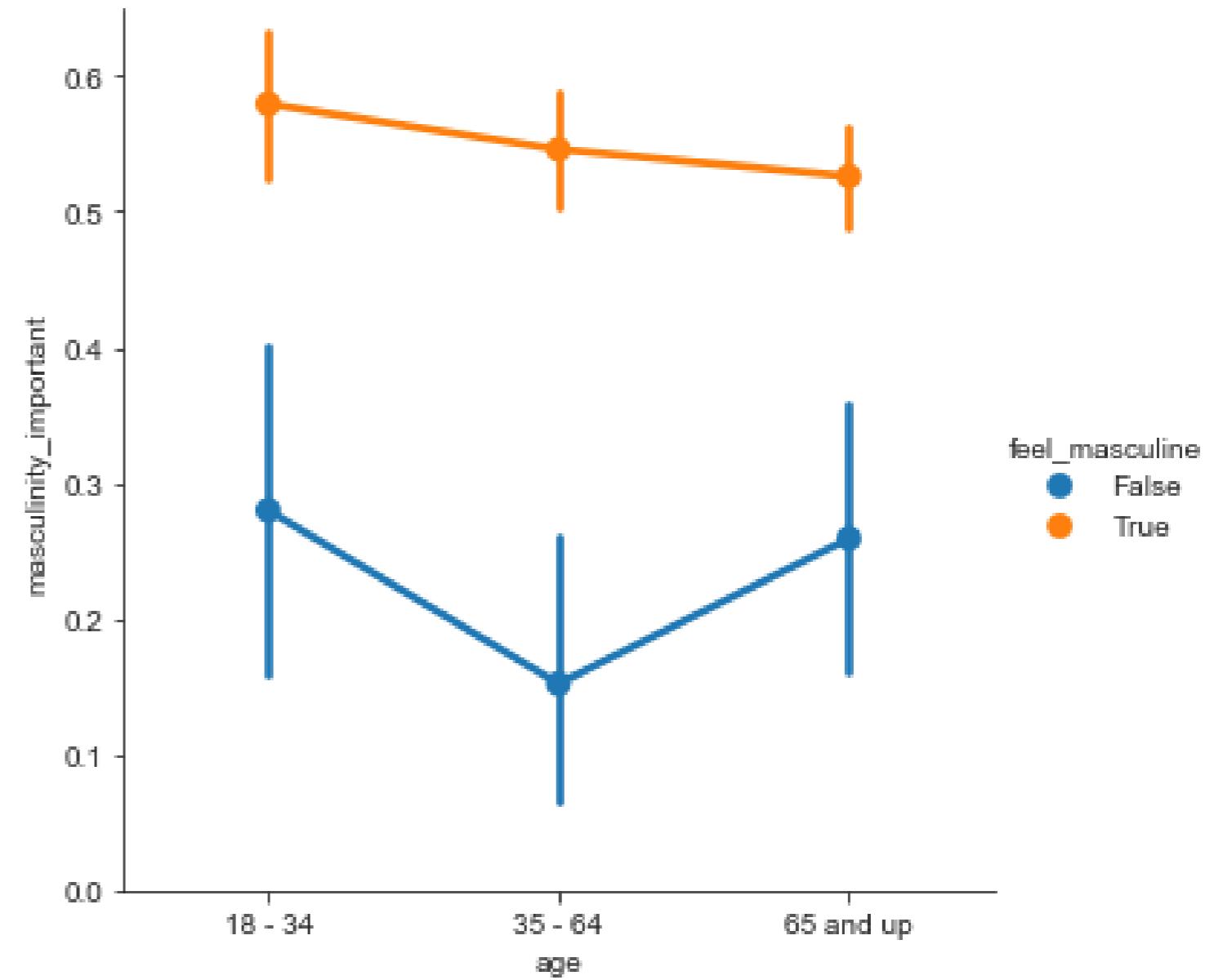
Figure style: "whitegrid"

```
sns.set_style("whitegrid")  
  
sns.catplot(x="age",  
             y="masculinity_important",  
             data=masculinity_data,  
             hue="feel_masculine",  
             kind="point")  
  
plt.show()
```



Other styles

```
sns.set_style("ticks")  
  
sns.catplot(x="age",  
             y="masculinity_important",  
             data=masculinity_data,  
             hue="feel_masculine",  
             kind="point")  
  
plt.show()
```

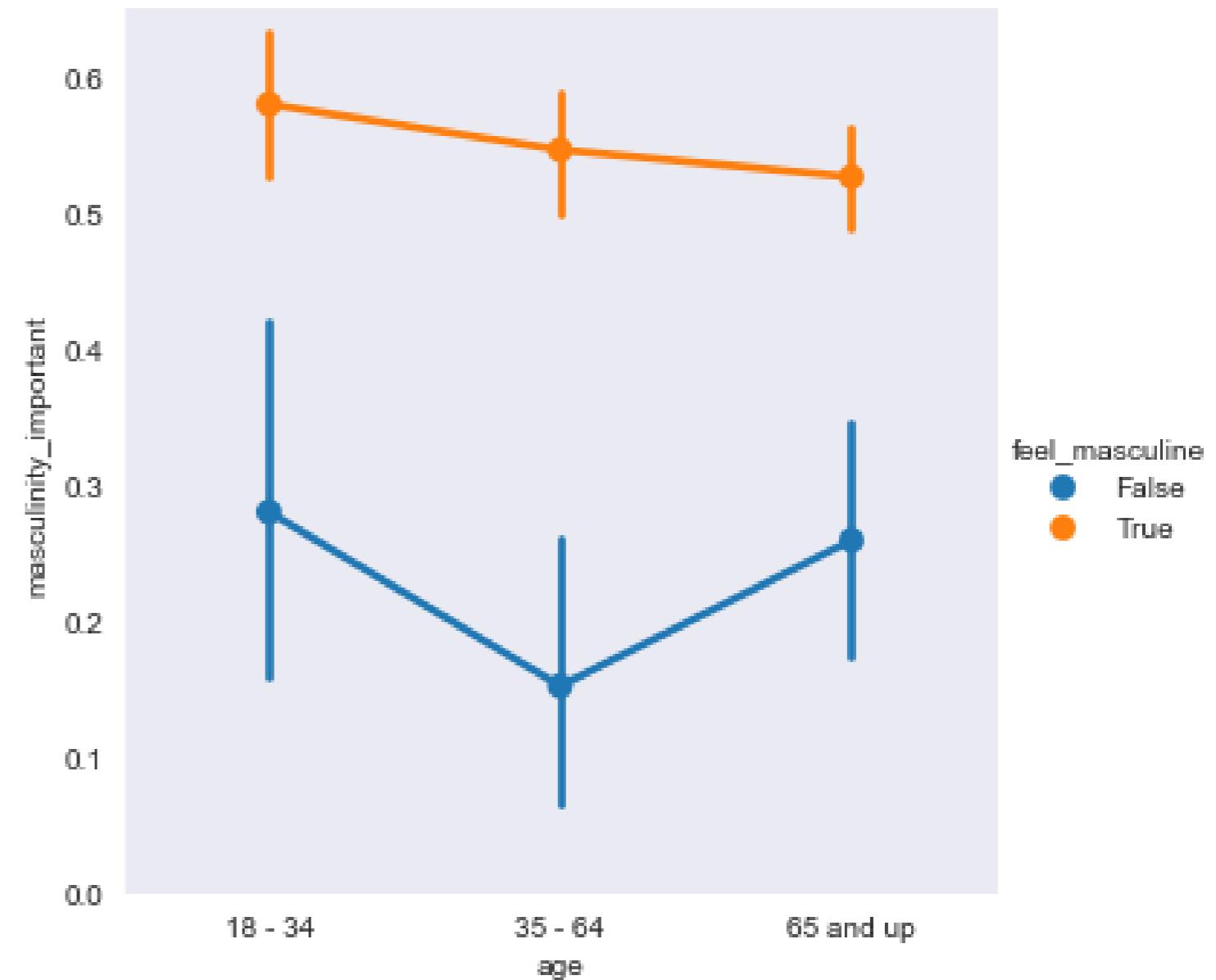


Other styles

```
sns.set_style("dark")

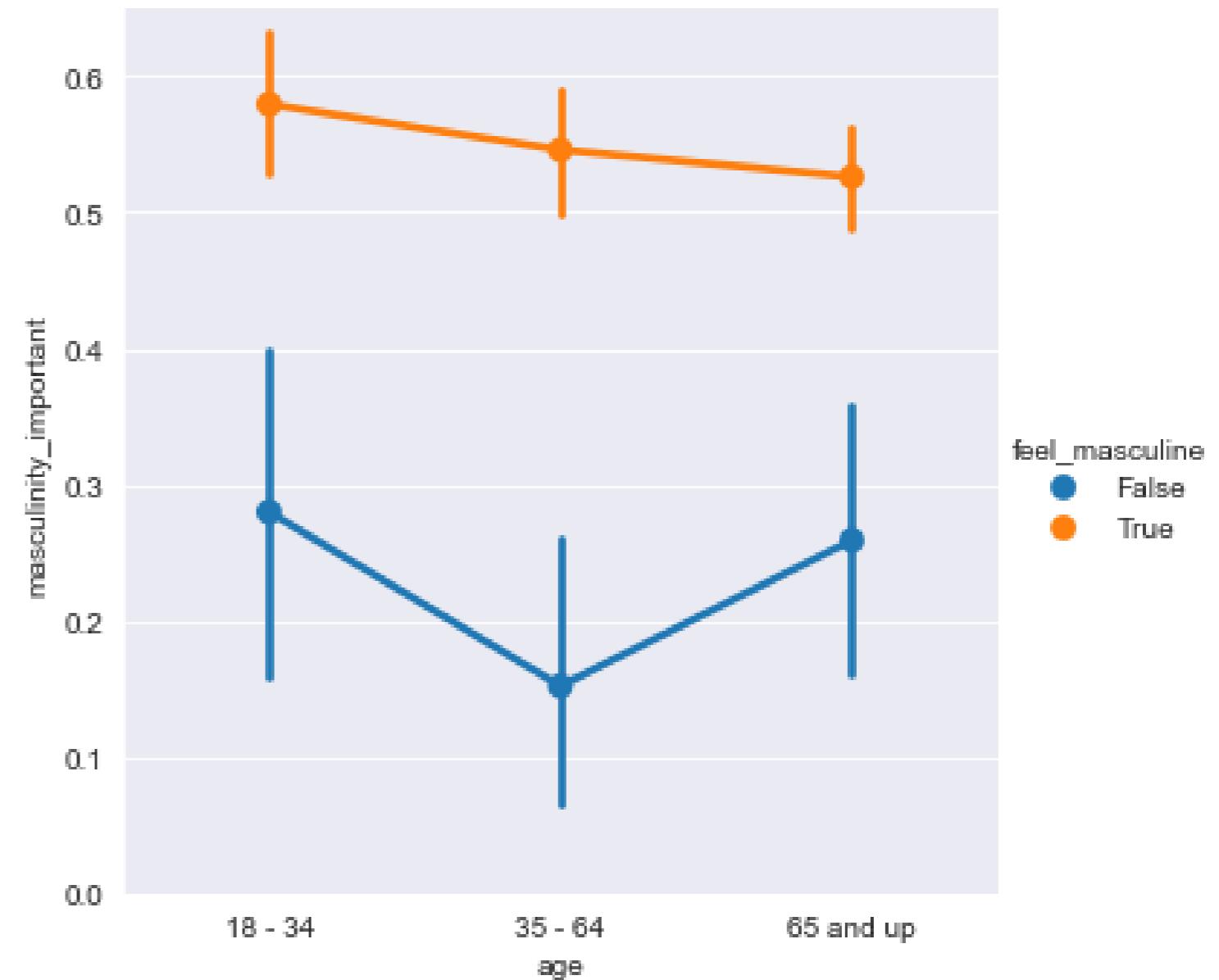
sns.catplot(x="age",
            y="masculinity_important",
            data=masculinity_data,
            hue="feel_masculine",
            kind="point")

plt.show()
```



Other styles

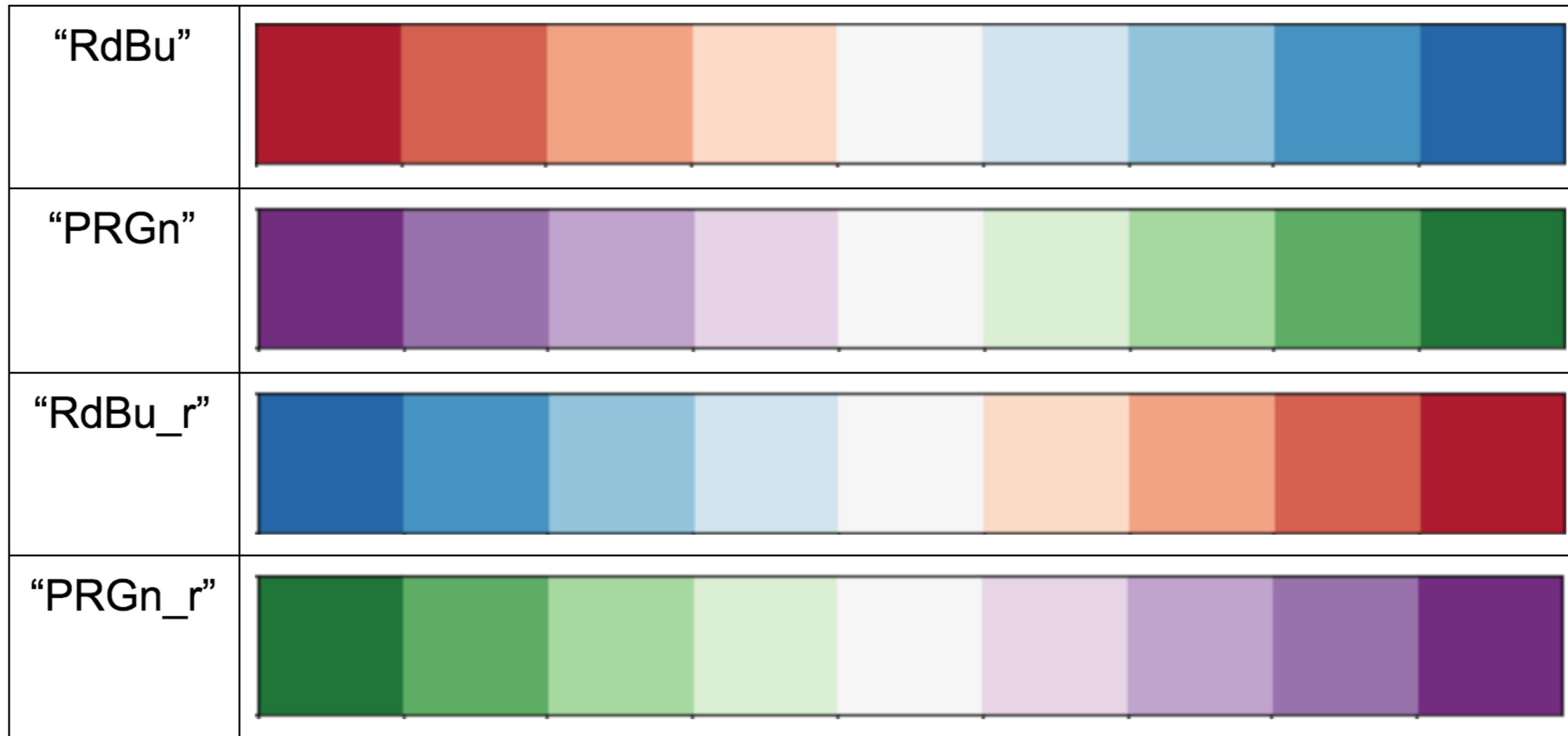
```
sns.set_style("darkgrid")  
  
sns.catplot(x="age",  
             y="masculinity_important",  
             data=masculinity_data,  
             hue="feel_masculine",  
             kind="point")  
  
plt.show()
```



Changing the palette

- Figure "palette" changes the color of the main elements of the plot
- `sns.set_palette()`
- Use preset palettes or create a custom palette

Diverging palettes

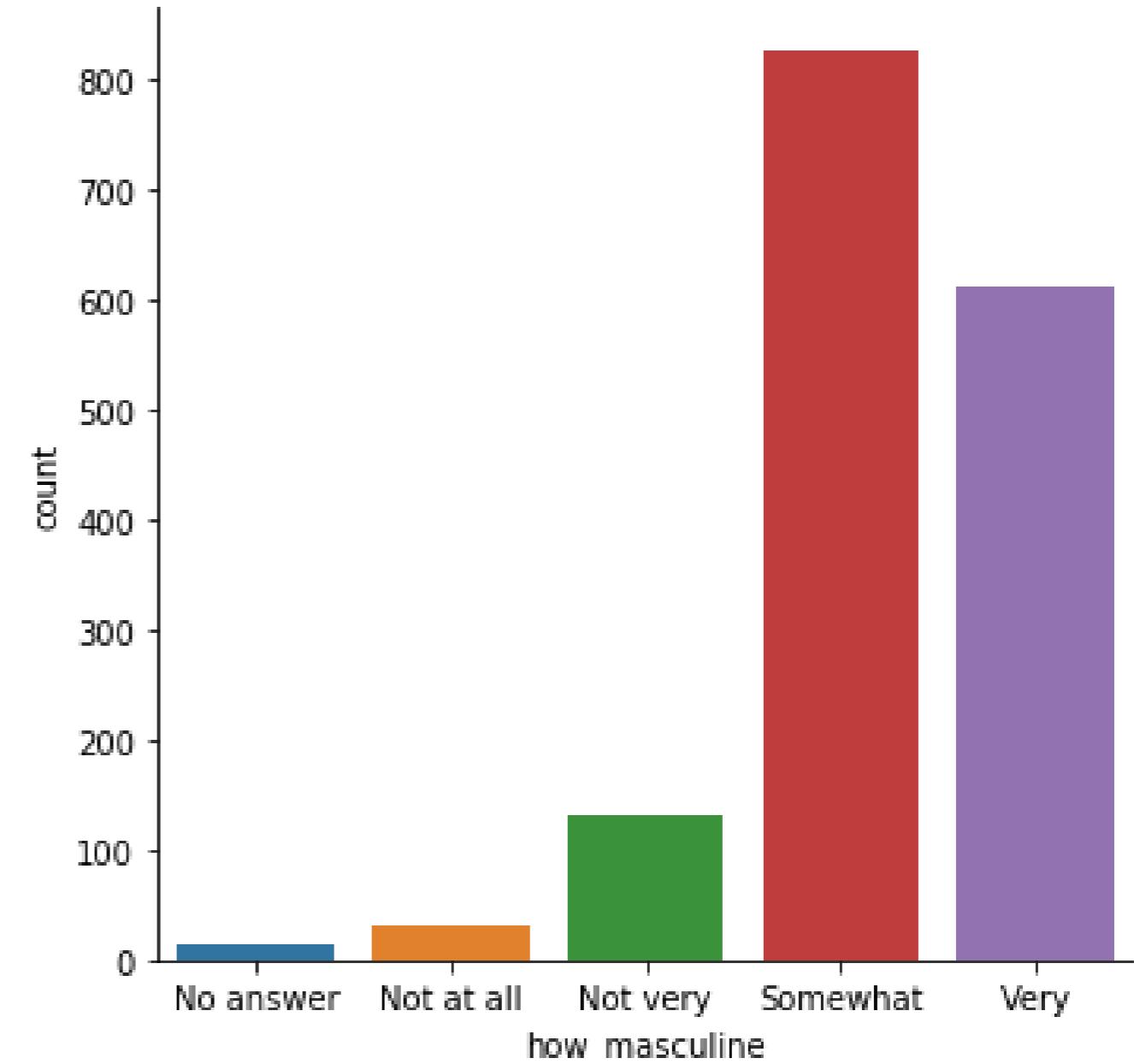


Example (default palette)

```
category_order = ["No answer",
                  "Not at all",
                  "Not very",
                  "Somewhat",
                  "Very"]

sns.catplot(x="how_masculine",
            data=mascularity_data,
            kind="count",
            order=category_order)

plt.show()
```



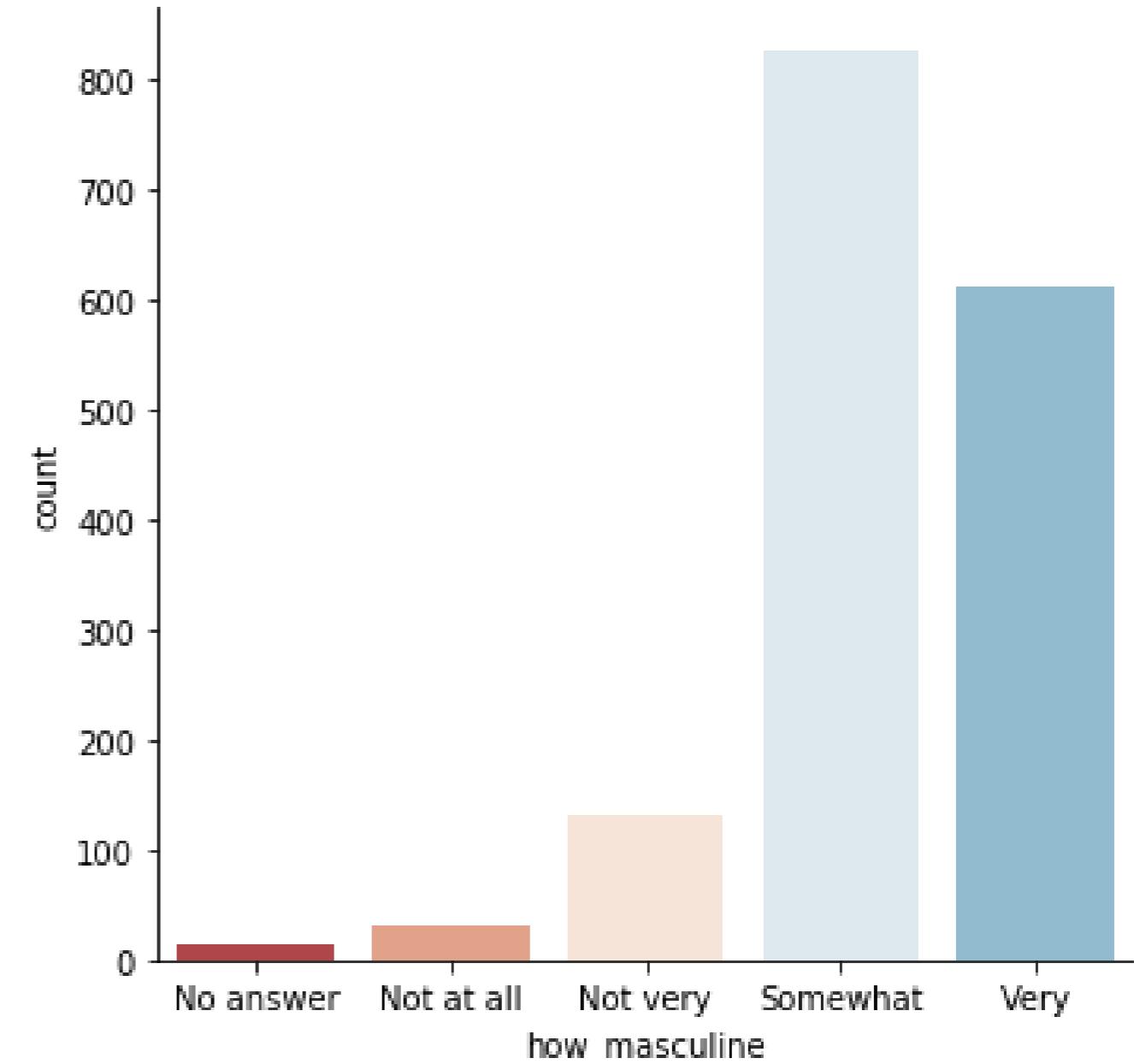
Example (diverging palette)

```
sns.set_palette("RdBu")

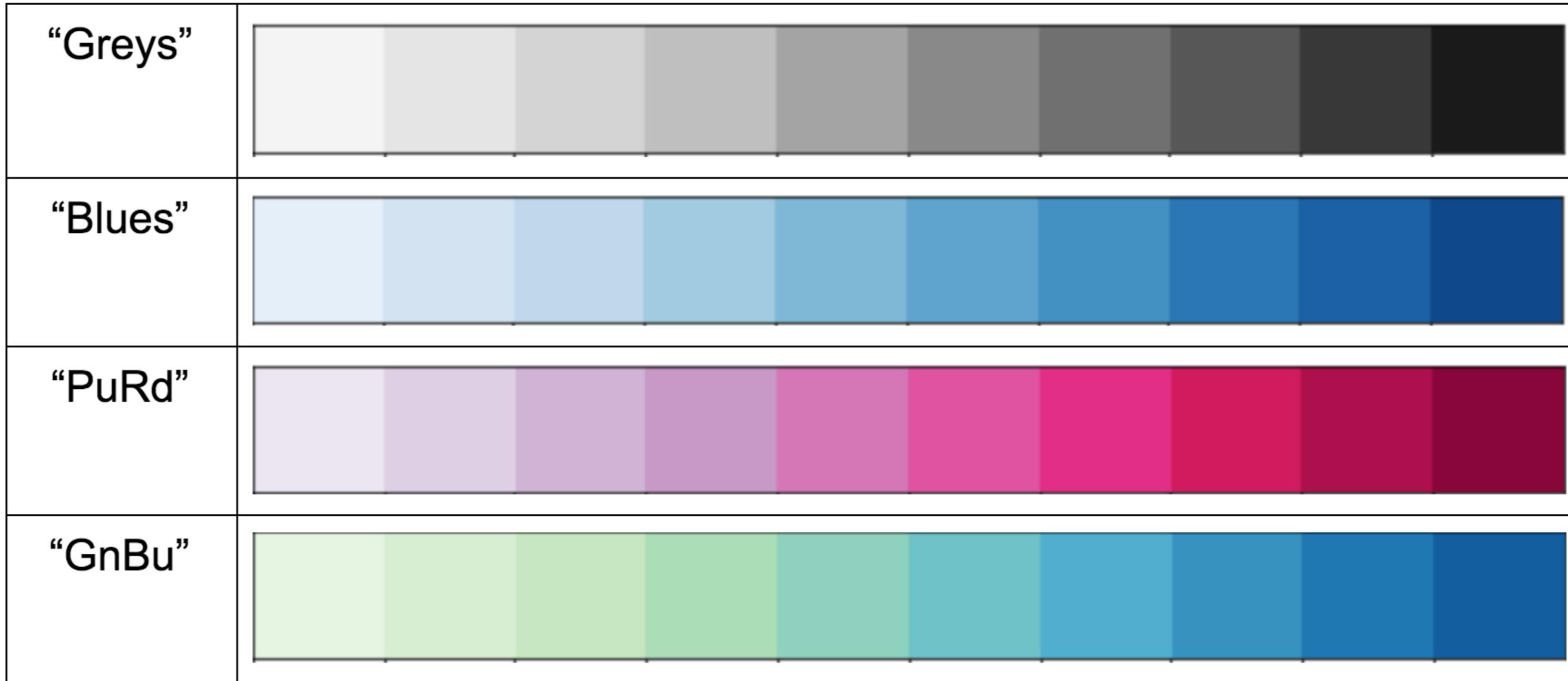
category_order = ["No answer",
                  "Not at all",
                  "Not very",
                  "Somewhat",
                  "Very"]

sns.catplot(x="how_masculine",
            data=masculinity_data,
            kind="count",
            order=category_order)

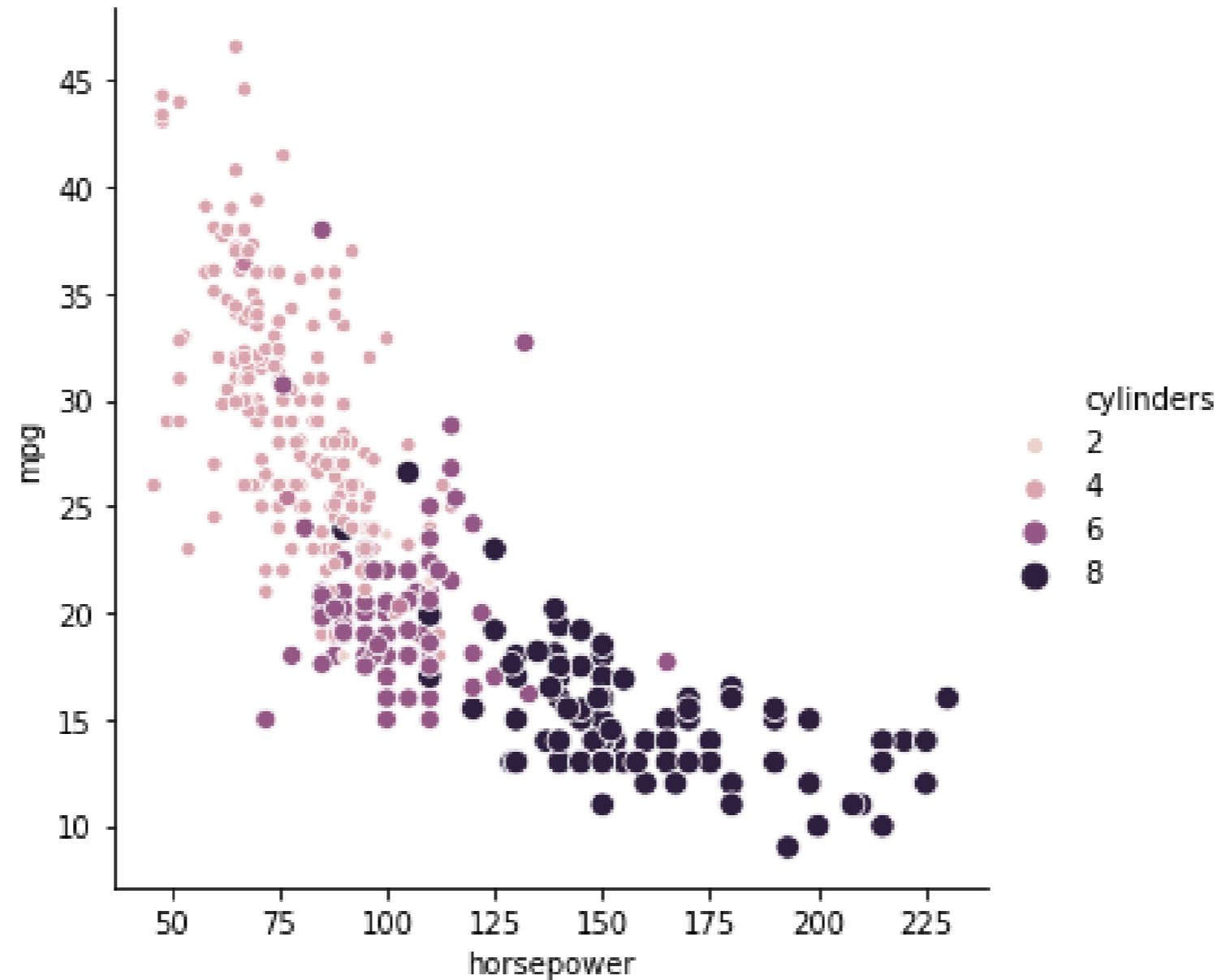
plt.show()
```



Sequential palettes



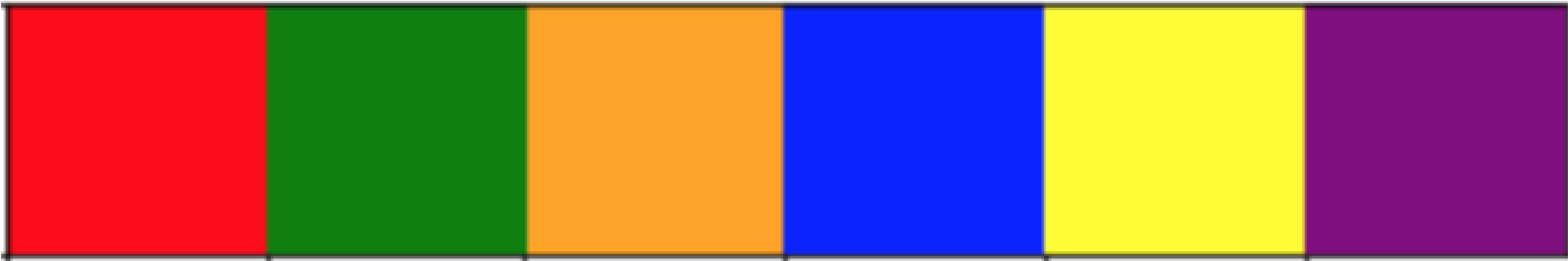
Sequential palette example



Custom palettes

```
custom_palette = ["red", "green", "orange", "blue",  
                  "yellow", "purple"]
```

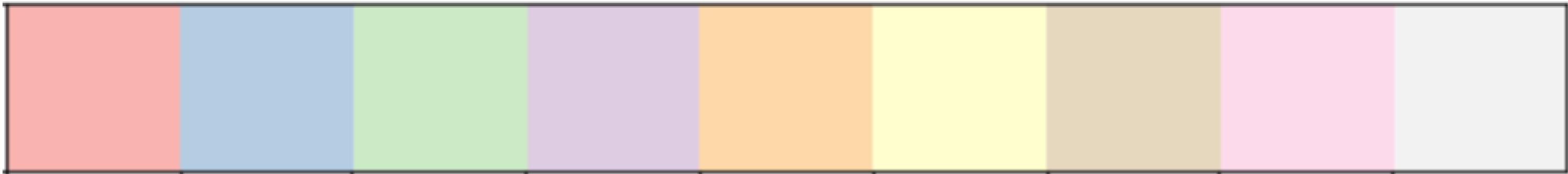
```
sns.set_palette(custom_palette)
```



Custom palettes

```
custom_palette = ['#FBB4AE', '#B3CDE3', '#CCEBC5',
                  '#DECBE4', '#FED9A6', '#FFFFCC',
                  '#E5D8BD', '#FDDAEC', '#F2F2F2']

sns.set_palette(custom_palette)
```

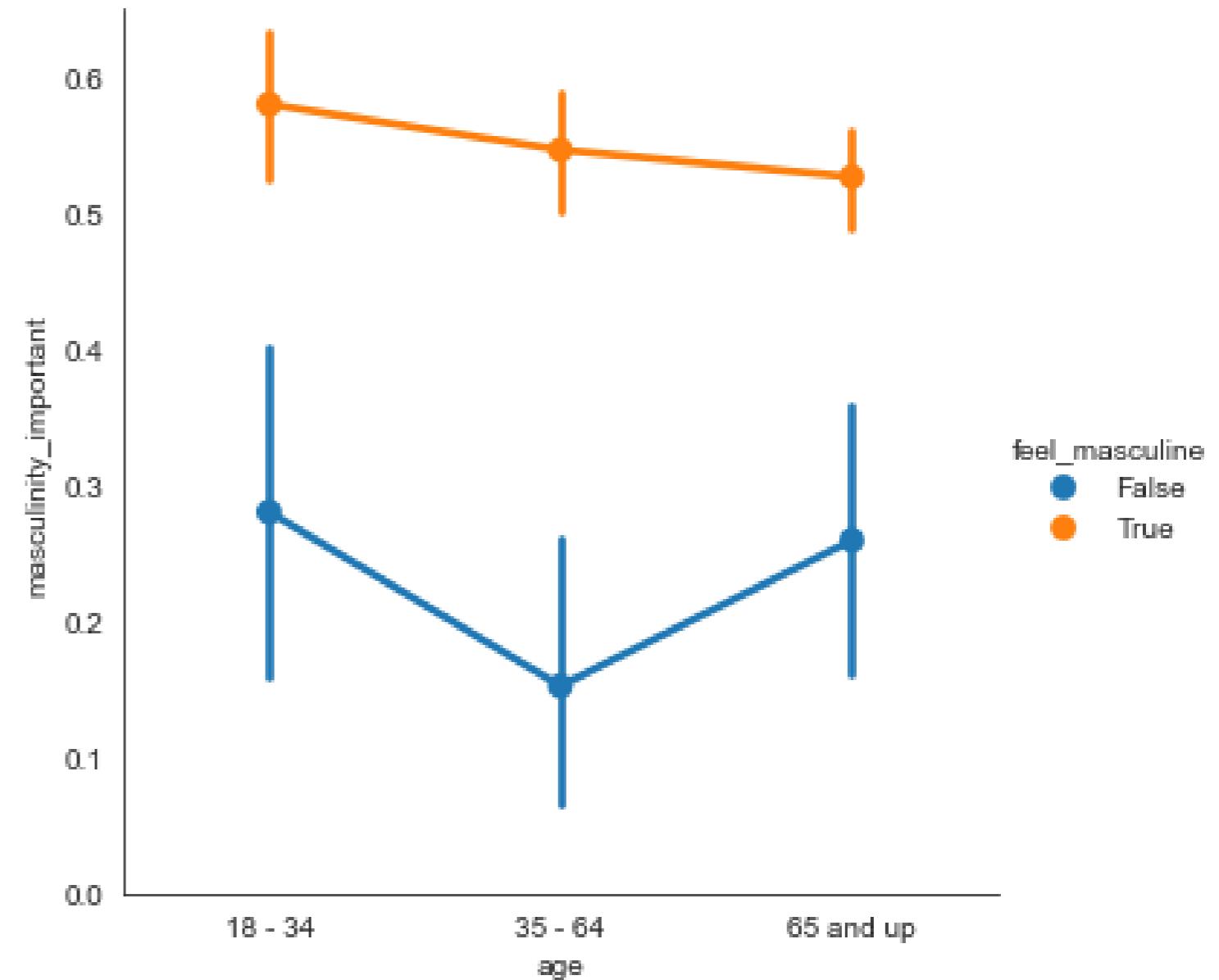


Changing the scale

- Figure "context" changes the scale of the plot elements and labels
- `sns.set_context()`
- Smallest to largest: "paper", "notebook", "talk", "poster"

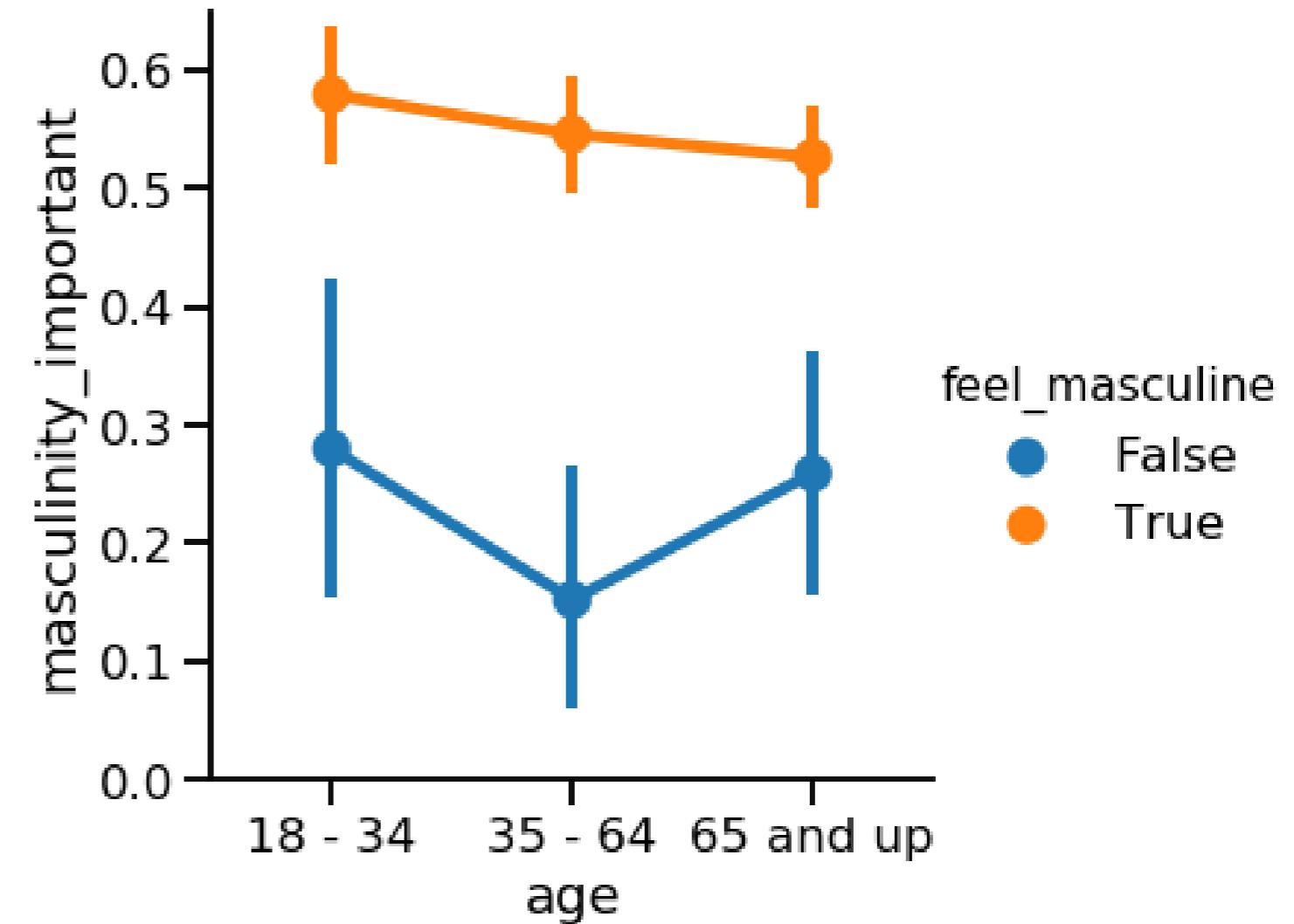
Default context: "paper"

```
sns.catplot(x="age",  
            y="masculinity_important",  
            data=mascularity_data,  
            hue="feel_masculine",  
            kind="point")  
  
plt.show()
```



Larger context: "talk"

```
sns.set_context("talk")  
  
sns.catplot(x="age",  
             y="masculinity_important",  
             data=masculinity_data,  
             hue="feel_masculine",  
             kind="point")  
  
plt.show()
```

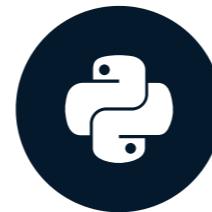


Let's practice!

INTRODUCTION TO DATA VISUALIZATION WITH SEABORN

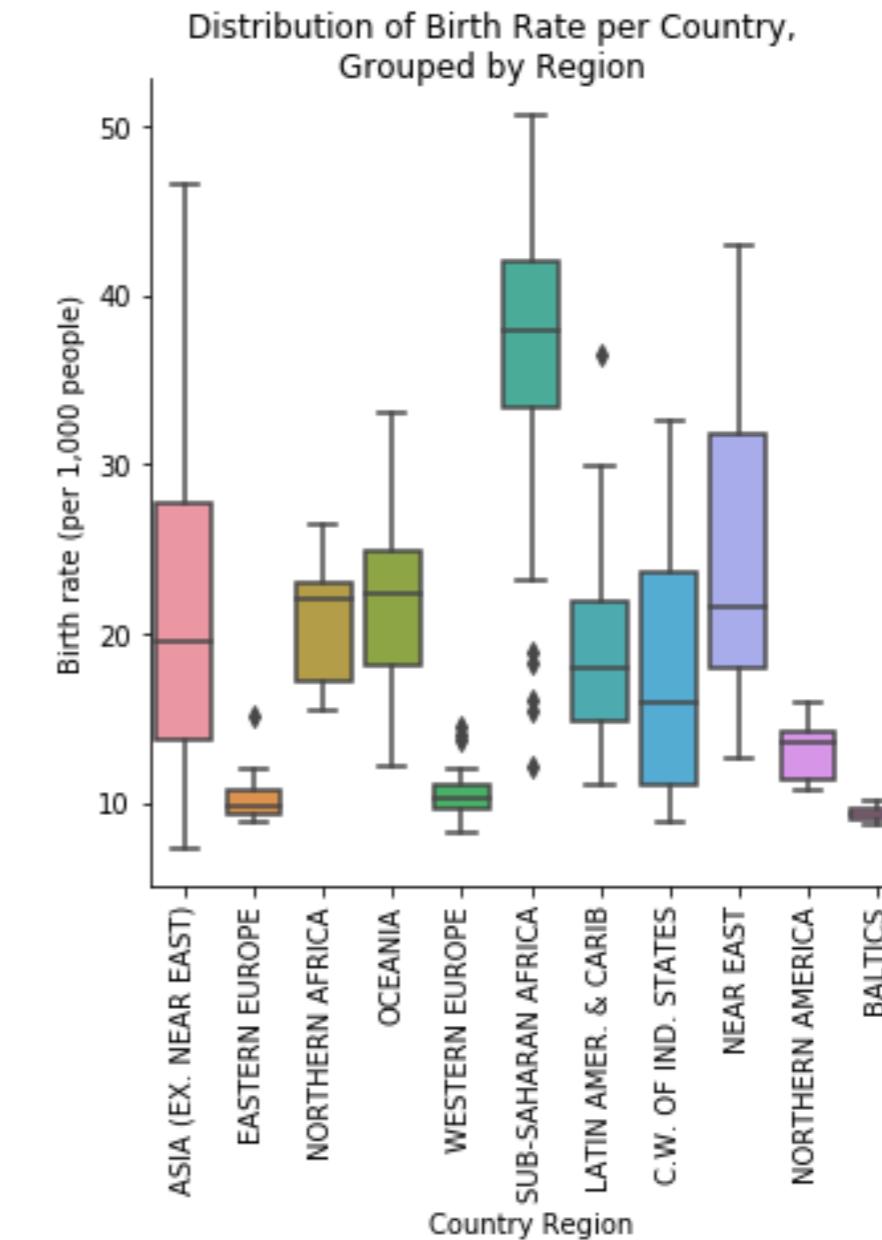
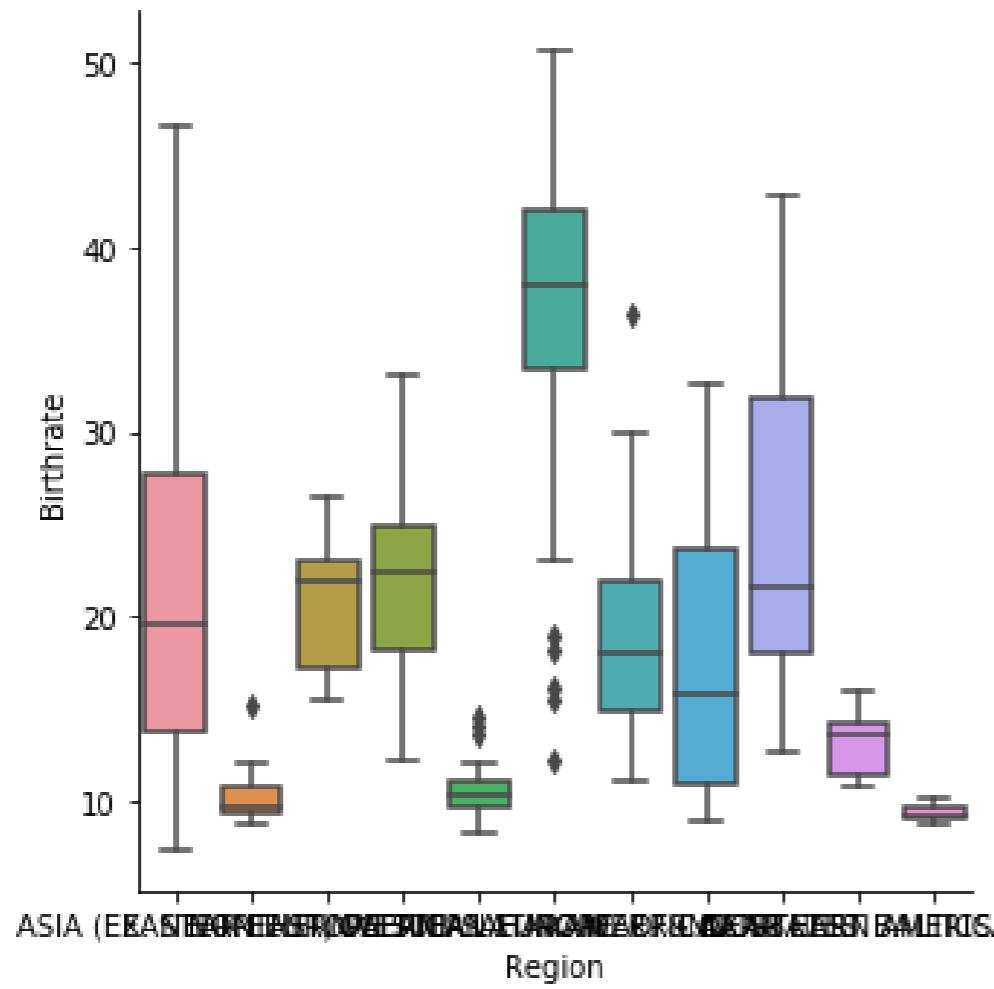
Adding titles and labels: Part 1

INTRODUCTION TO DATA VISUALIZATION WITH SEABORN



Erin Case
Data Scientist

Creating informative visualizations



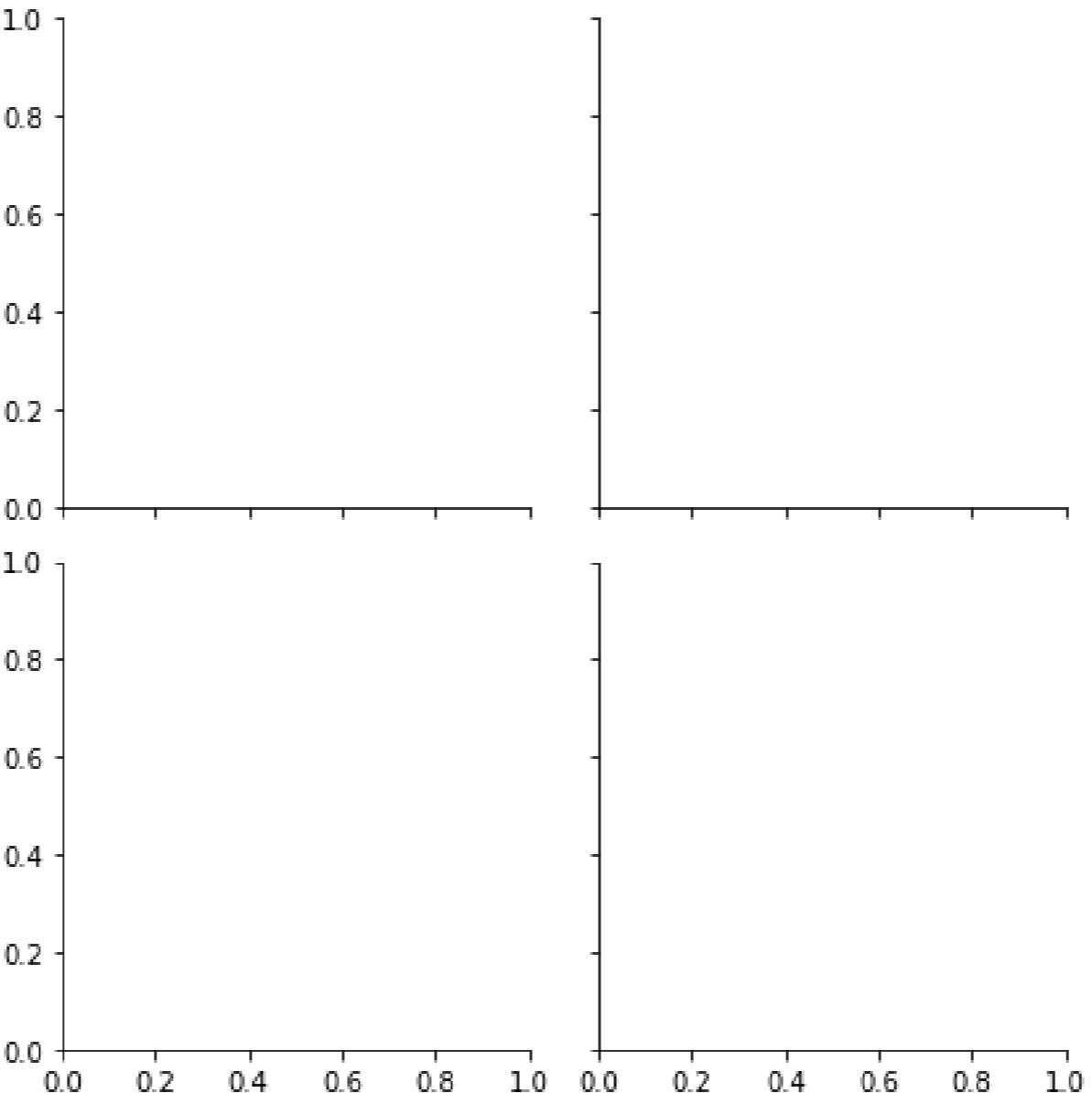
FacetGrid vs. AxesSubplot objects

Seaborn plots create two different types of objects: `FacetGrid` and `AxesSubplot`

```
g = sns.scatterplot(x="height", y="weight", data=df)  
type(g)
```

```
> matplotlib.axes._subplots.AxesSubplot
```

An Empty FacetGrid

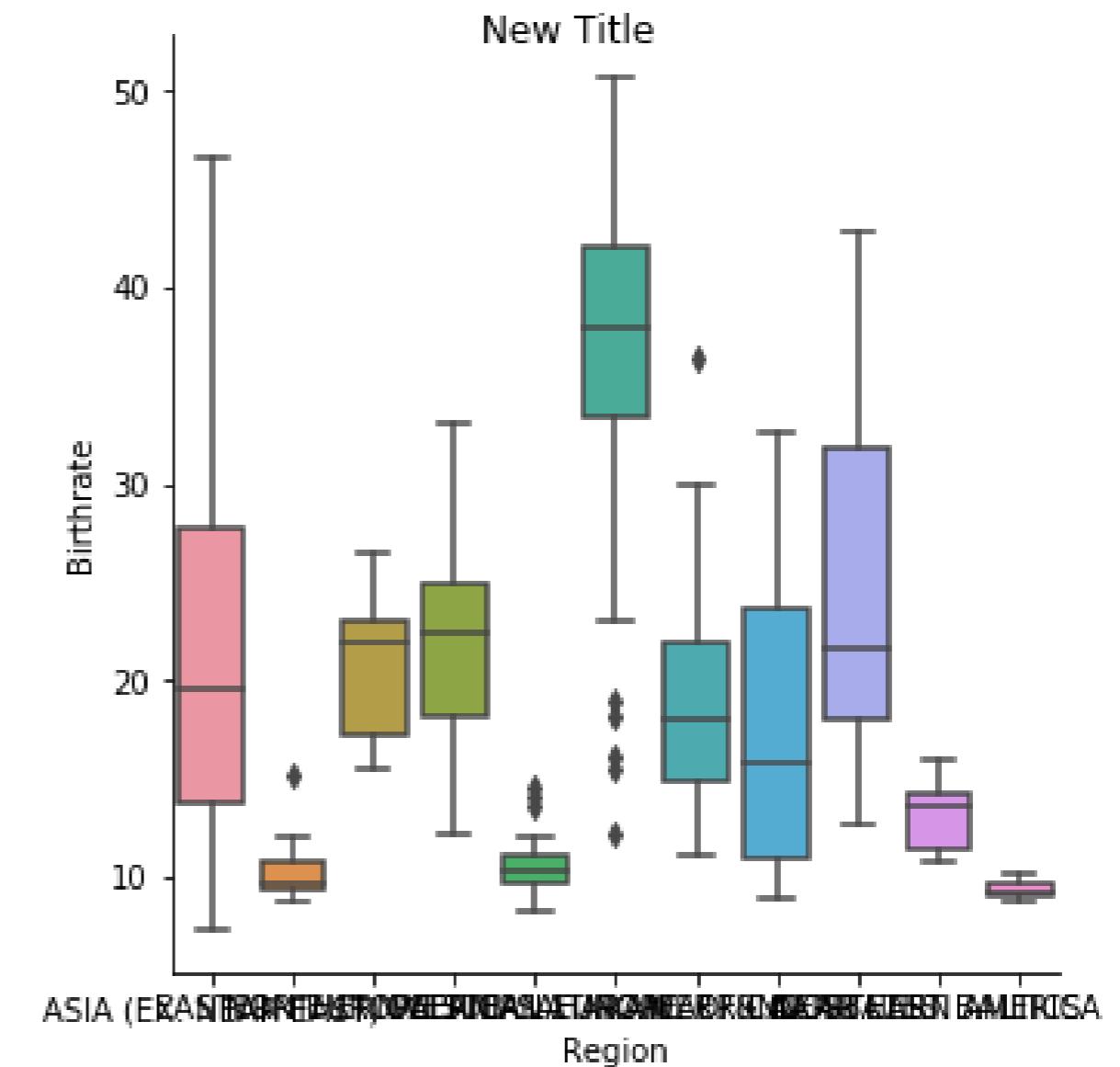


FacetGrid vs. AxesSubplot objects

Object Type	Plot Types	Characteristics
FacetGrid	<code>relplot()</code> , <code>catplot()</code>	Can create subplots
AxesSubplot	<code>scatterplot()</code> , <code>countplot()</code> , etc.	Only creates a single plot

Adding a title to FacetGrid

```
g = sns.catplot(x="Region",
                 y="Birthrate",
                 data=gdp_data,
                 kind="box")
g.fig.suptitle("New Title")
plt.show()
```

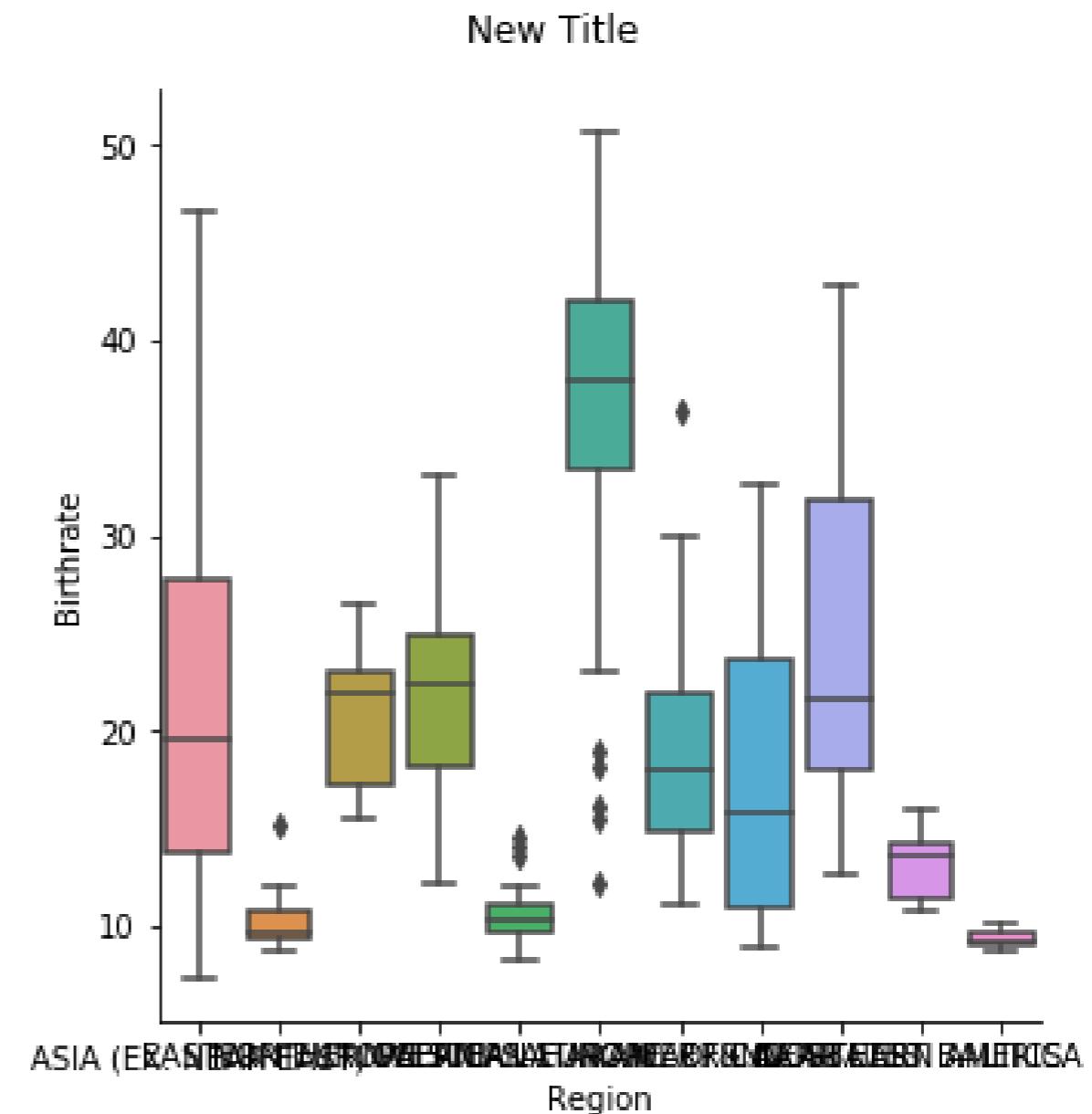


Adjusting height of title in FacetGrid

```
g = sns.catplot(x="Region",
                 y="Birthrate",
                 data=gdp_data,
                 kind="box")

g.fig.suptitle("New Title",
               y=1.03)

plt.show()
```

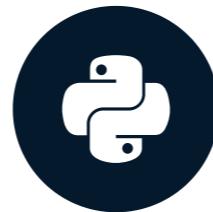


Let's practice!

INTRODUCTION TO DATA VISUALIZATION WITH SEABORN

Adding titles and labels: Part 2

INTRODUCTION TO DATA VISUALIZATION WITH SEABORN



Erin Case
Data Scientist

Adding a title to AxesSubplot

FacetGrid

```
g = sns.catplot(x="Region",  
                 y="Birthrate",  
                 data=gdp_data,  
                 kind="box")
```

```
g.fig.suptitle("New Title",  
               y=1.03)
```

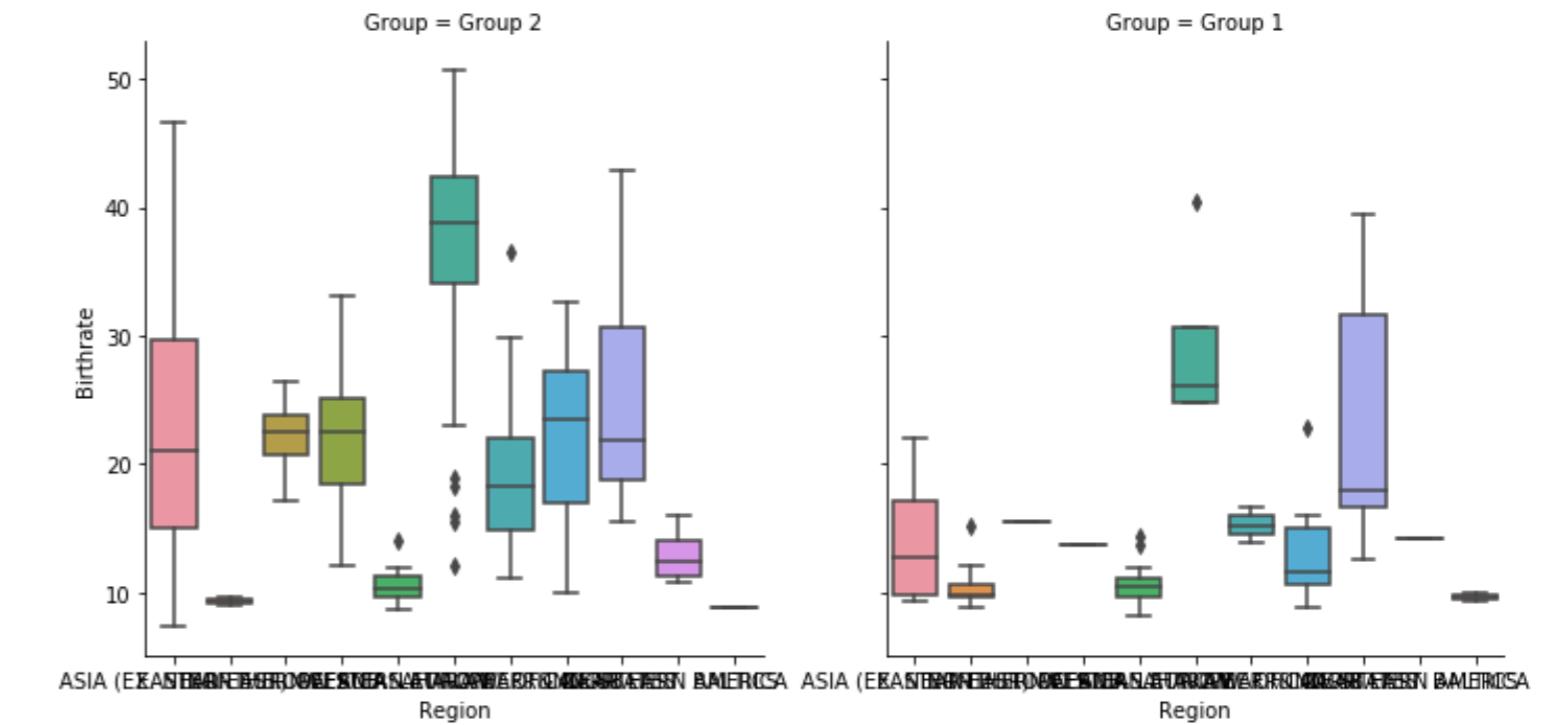
AxesSubplot

```
g = sns.boxplot(x="Region",  
                 y="Birthrate",  
                 data=gdp_data)
```

```
g.set_title("New Title",  
            y=1.03)
```

Titles for subplots

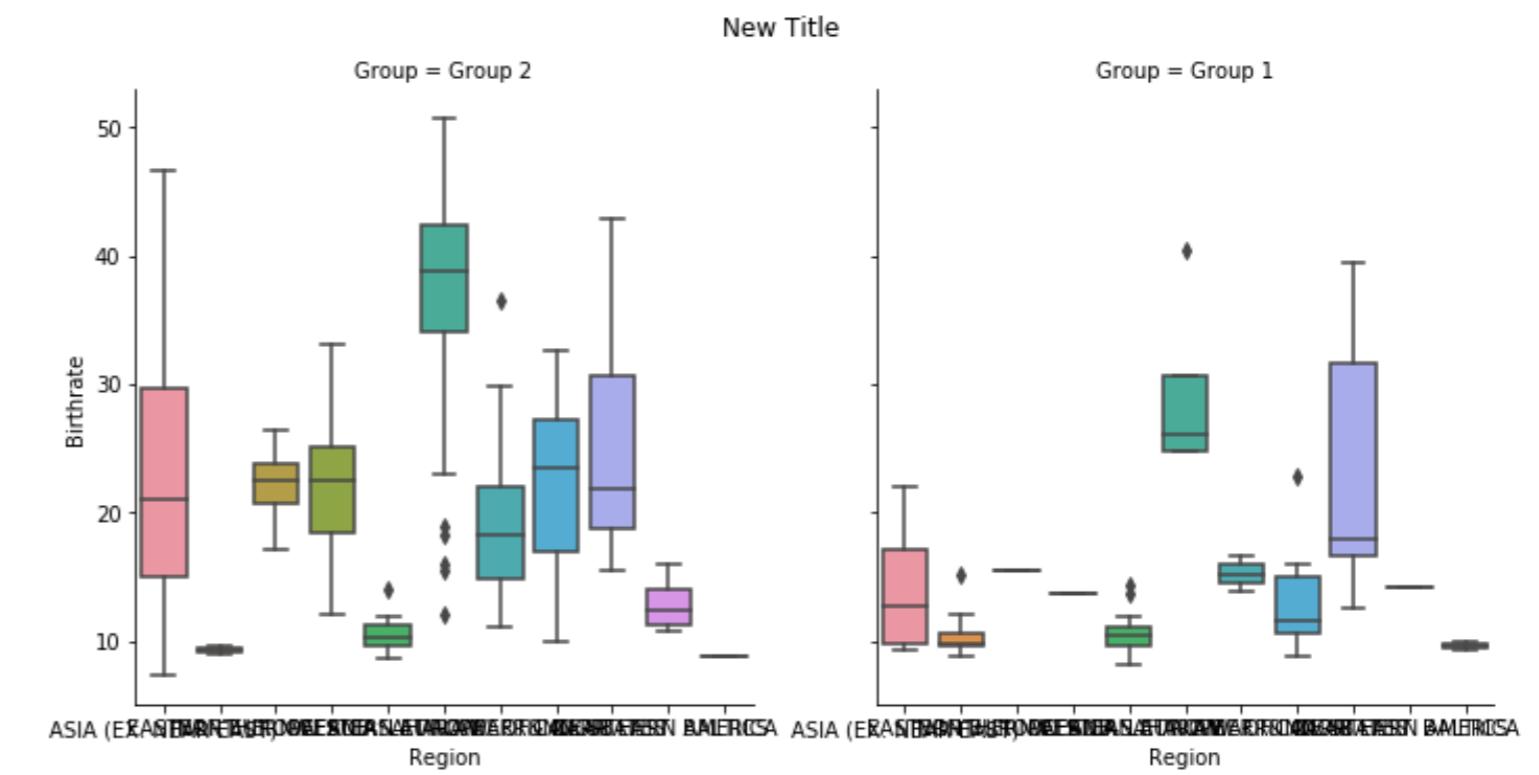
```
g = sns.catplot(x="Region",  
                 y="Birthrate",  
                 data=gdp_data,  
                 kind="box",  
                 col="Group")
```



Titles for subplots

```
g = sns.catplot(x="Region",
                 y="Birthrate",
                 data=gdp_data,
                 kind="box",
                 col="Group")

g.fig.suptitle("New Title",
               y=1.03)
```

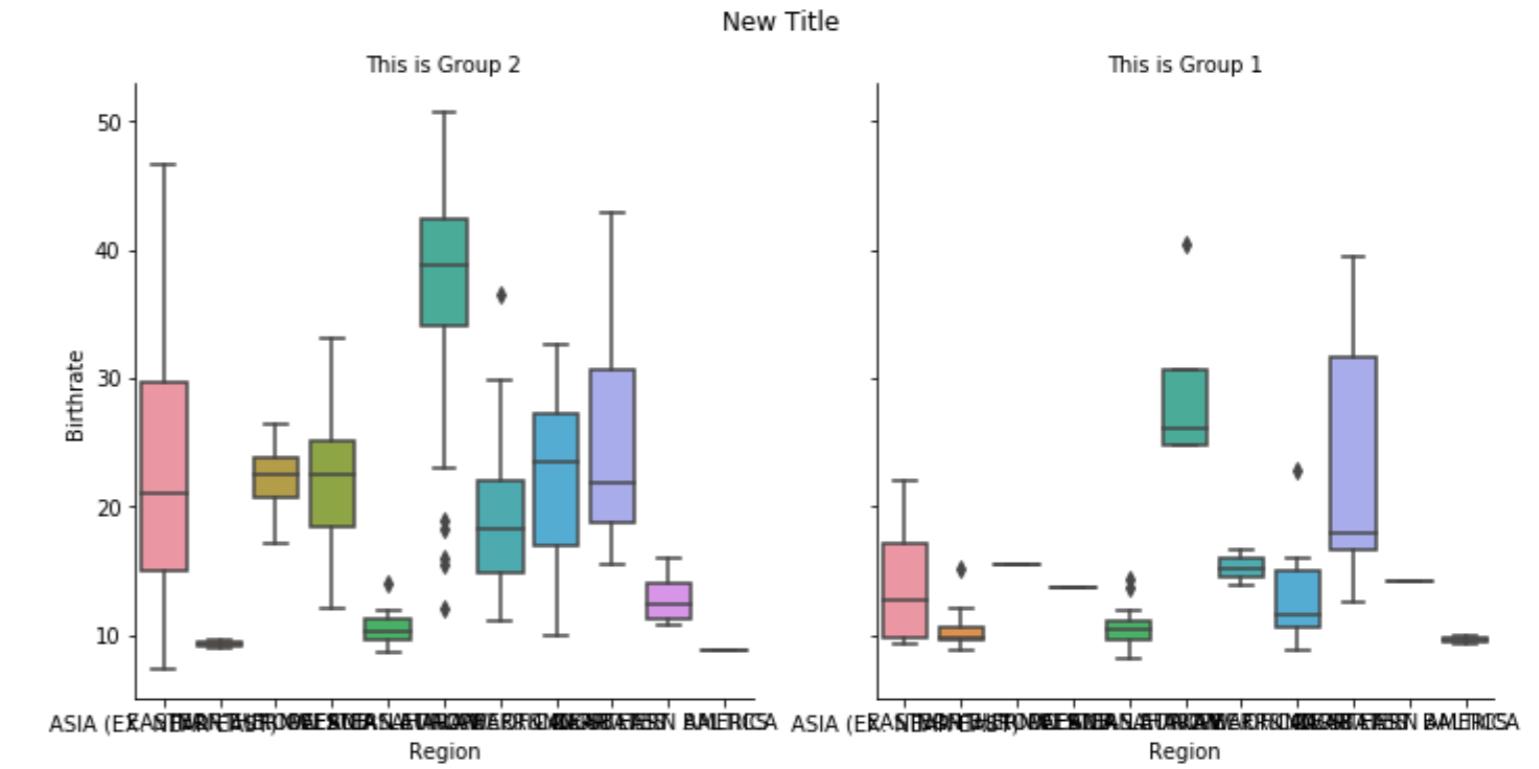


Titles for subplots

```
g = sns.catplot(x="Region",
                 y="Birthrate",
                 data=gdp_data,
                 kind="box",
                 col="Group")

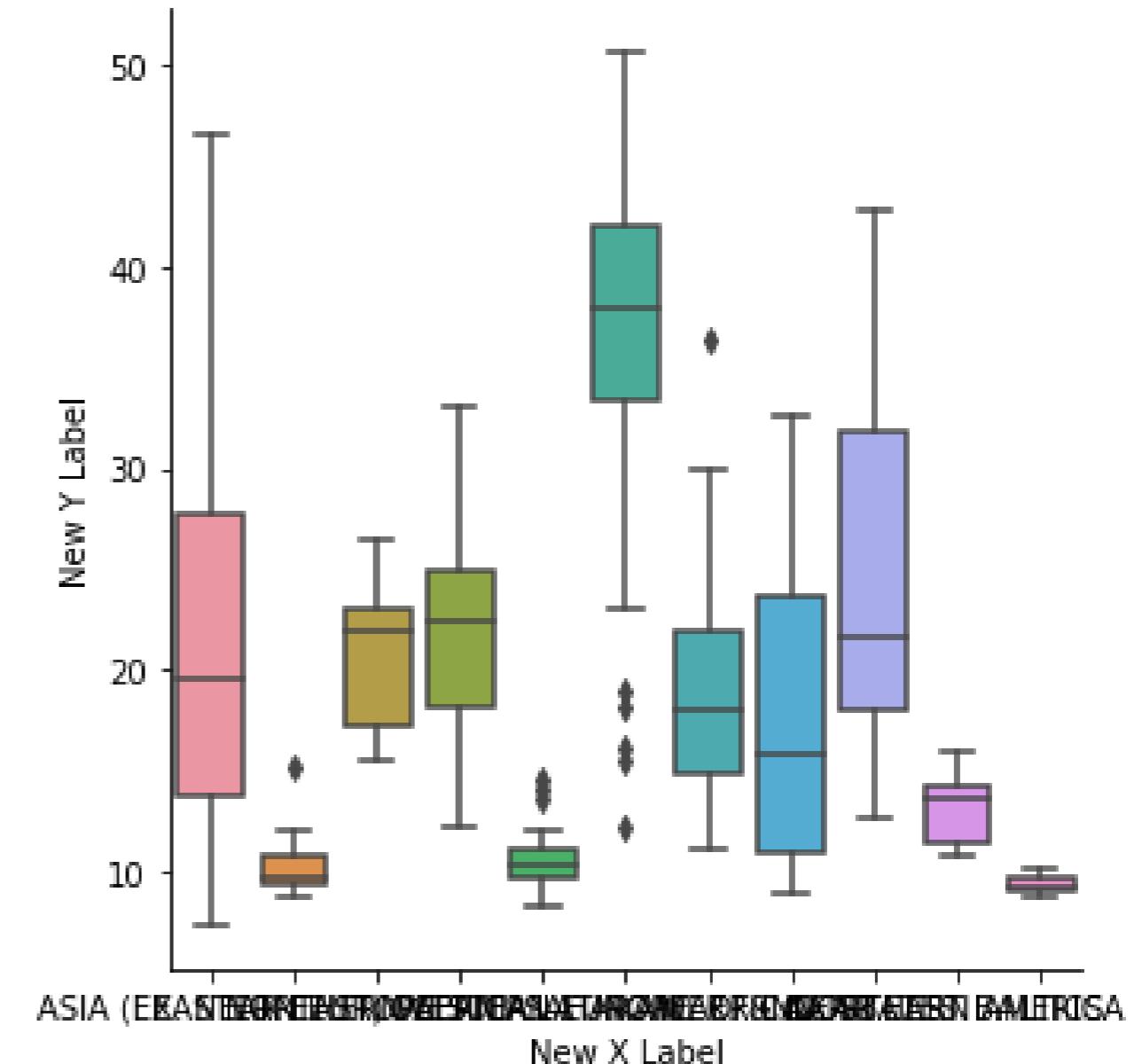
g.fig.suptitle("New Title",
               y=1.03)

g.set_titles("This is {col_name}")
```



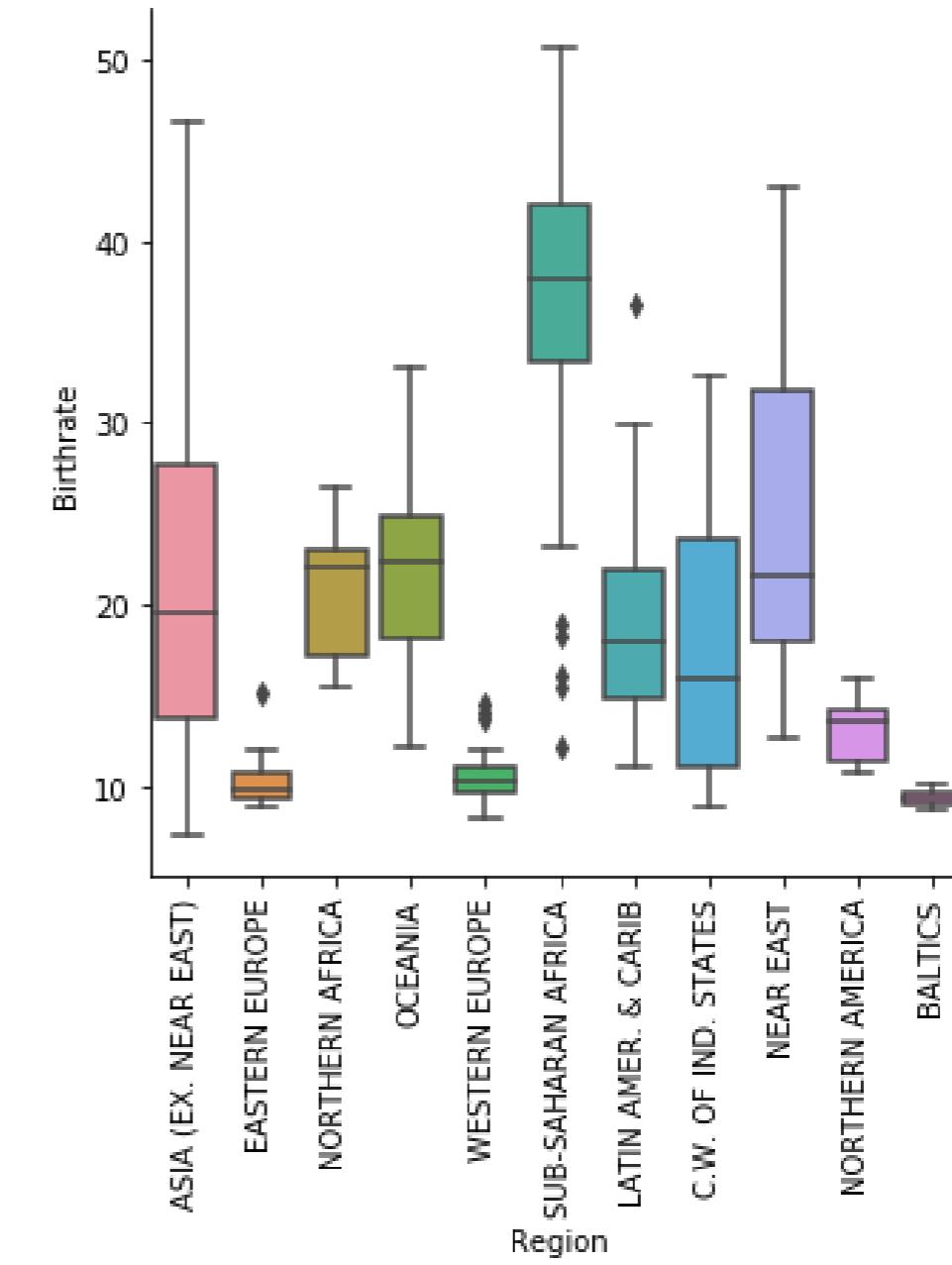
Adding axis labels

```
g = sns.catplot(x="Region",
                 y="Birthrate",
                 data=gdp_data,
                 kind="box")
`  
g.set(xlabel="New X Label",
      ylabel="New Y Label")  
plt.show()
```



Rotating x-axis tick labels

```
g = sns.catplot(x="Region",
                 y="Birthrate",
                 data=gdp_data,
                 kind="box")
plt.xticks(rotation=90)
plt.show()
```

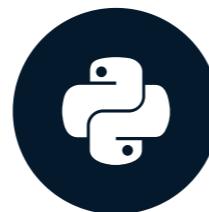


Let's practice!

INTRODUCTION TO DATA VISUALIZATION WITH SEABORN

Putting it all together

INTRODUCTION TO DATA VISUALIZATION WITH SEABORN



Erin Case
Data Scientist

Getting started

To import Seaborn:

```
import seaborn as sns
```

To import Matplotlib:

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

To show a plot:

```
plt.show()
```

Relational plots

- Show the relationship between two quantitative variables
- Examples: scatter plots, line plots

```
sns.relplot(x="x_variable_name",
             y="y_variable_name",
             data=pandas_df,
             kind="scatter")
```

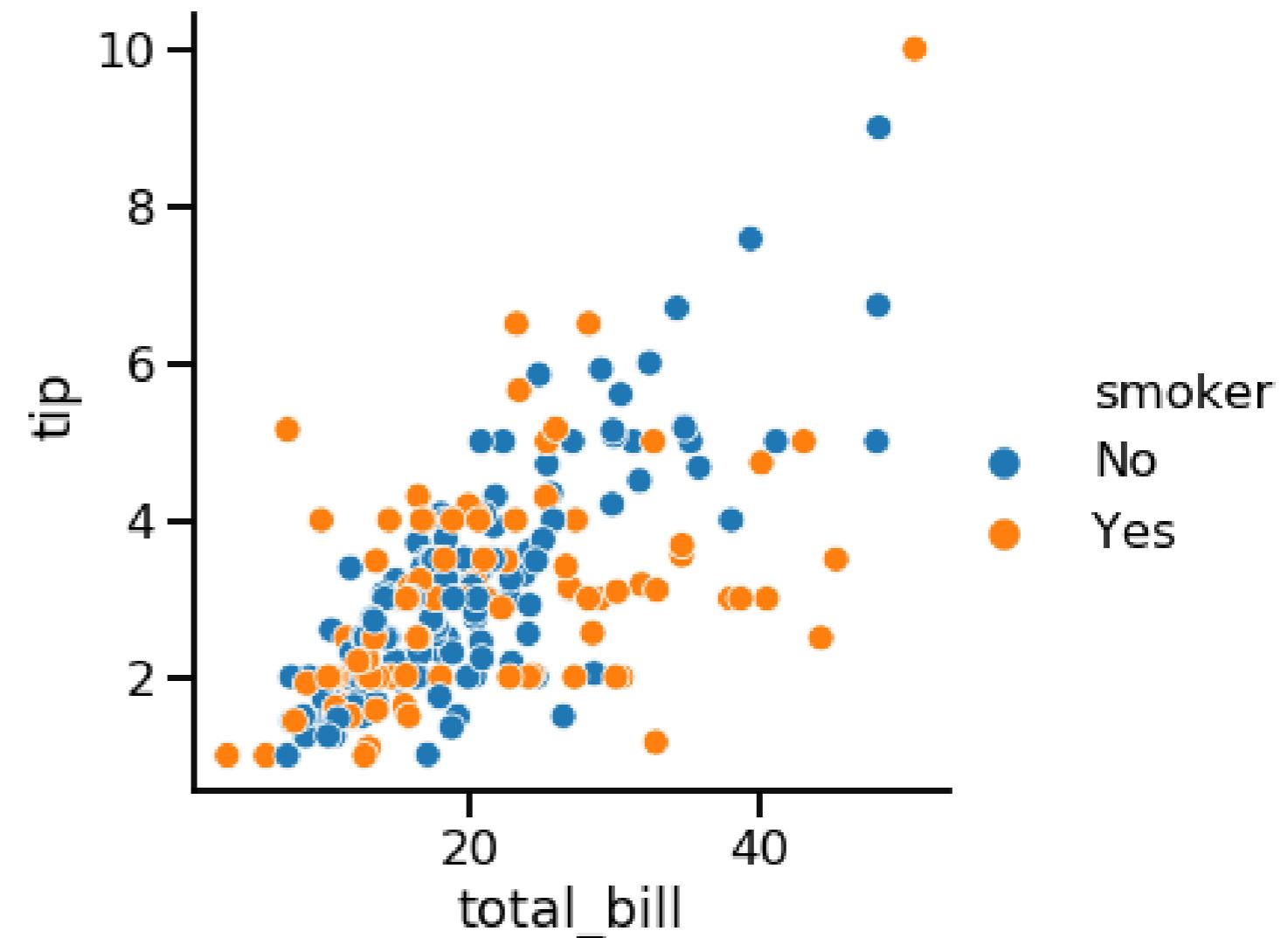
Categorical plots

- Show the distribution of a quantitative variable within categories defined by a categorical variable
- Examples: bar plots, count plots, box plots, point plots

```
sns.catplot(x="x_variable_name",  
             y="y_variable_name",  
             data=pandas_df,  
             kind="bar")
```

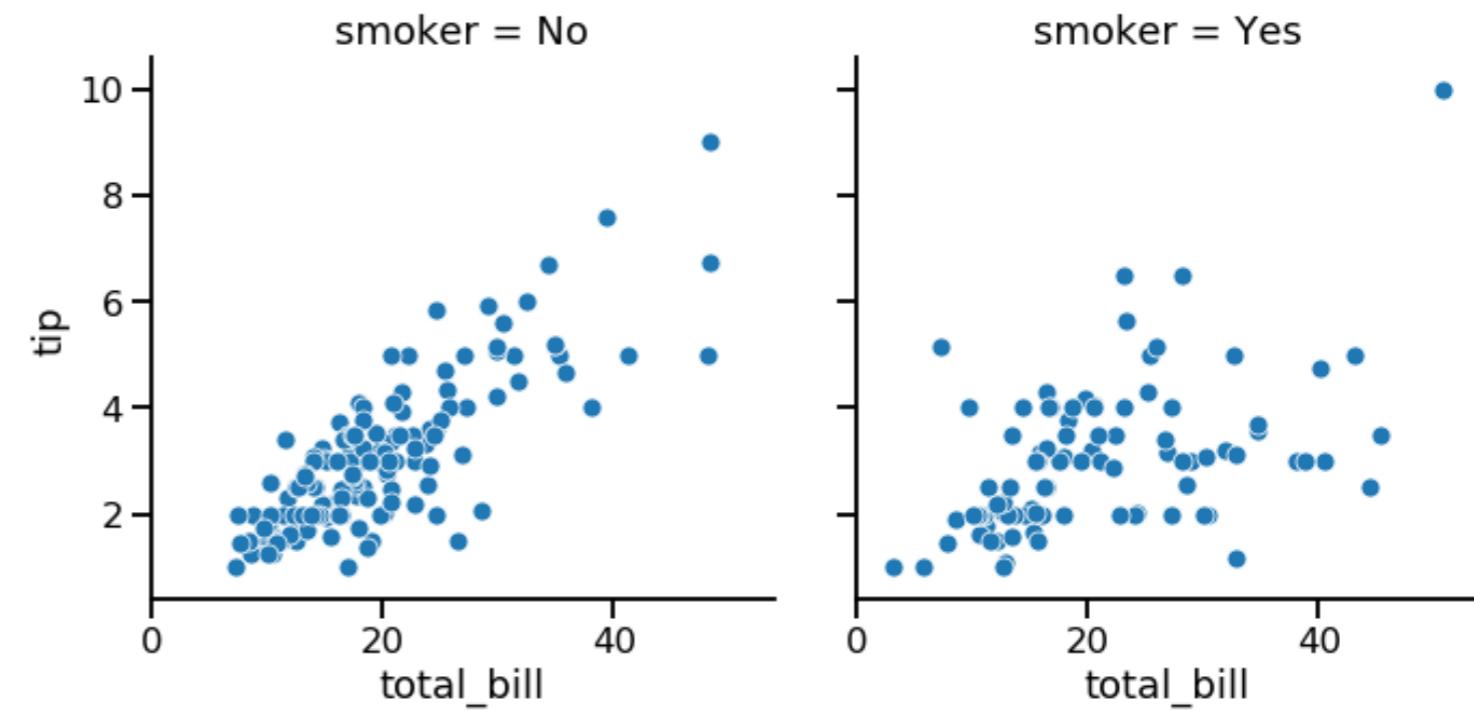
Adding a third variable (hue)

Setting `hue` will create subgroups that are displayed as different colors on a single plot.



Adding a third variable (row/col)

Setting `row` and/or `col` in `relplot()` or `catplot()` will create subgroups that are displayed on separate subplots.



Customization

- Change the background: `sns.set_style()`
- Change the main element colors: `sns.set_palette()`
- Change the scale: `sns.set_context()`

Adding a title

Object Type	Plot Types	How to Add Title
FacetGrid	<code>relplot()</code> , <code>catplot()</code>	<code>g.fig.suptitle()</code>
AxesSubplot	<code>scatterplot()</code> , <code>countplot()</code> , etc.	<code>g.set_title()</code>

Final touches

Add x- and y-axis labels:

```
g.set(xlabel="new x-axis label",  
      ylabel="new y-axis label")
```

Rotate x-tick labels:

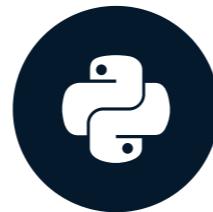
```
plt.xticks(rotation=90)
```

Let's practice!

INTRODUCTION TO DATA VISUALIZATION WITH SEABORN

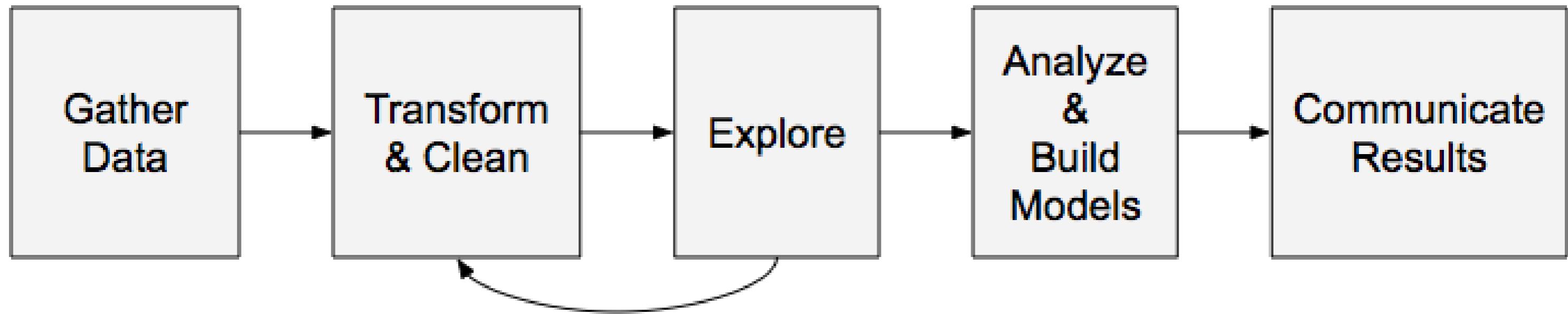
Well done! What's next?

INTRODUCTION TO DATA VISUALIZATION WITH SEABORN

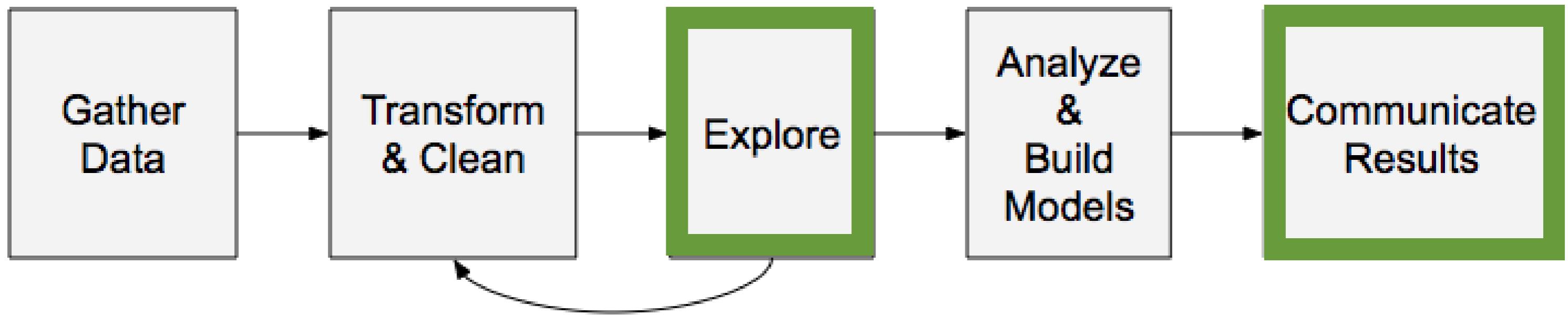


Erin Case
Data Scientist

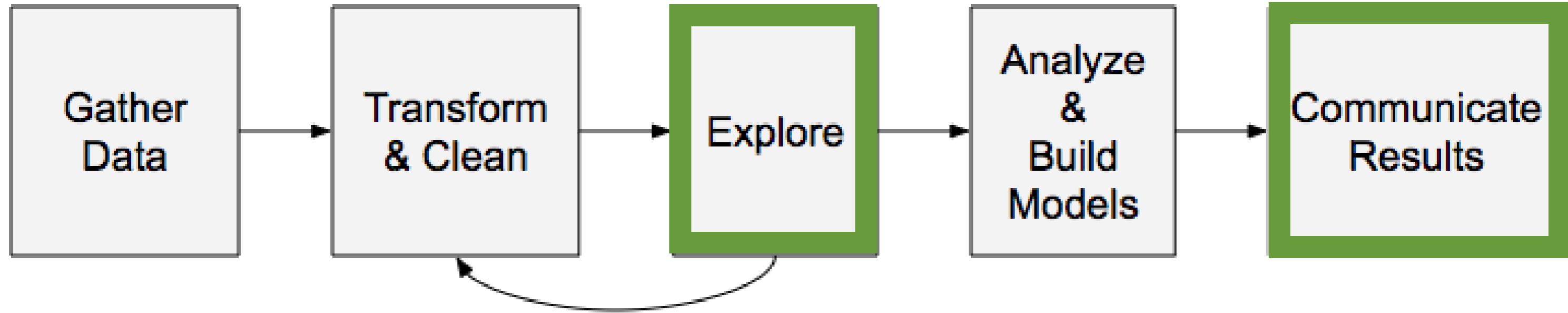
Where does Seaborn fit in?



Where does Seaborn fit in?



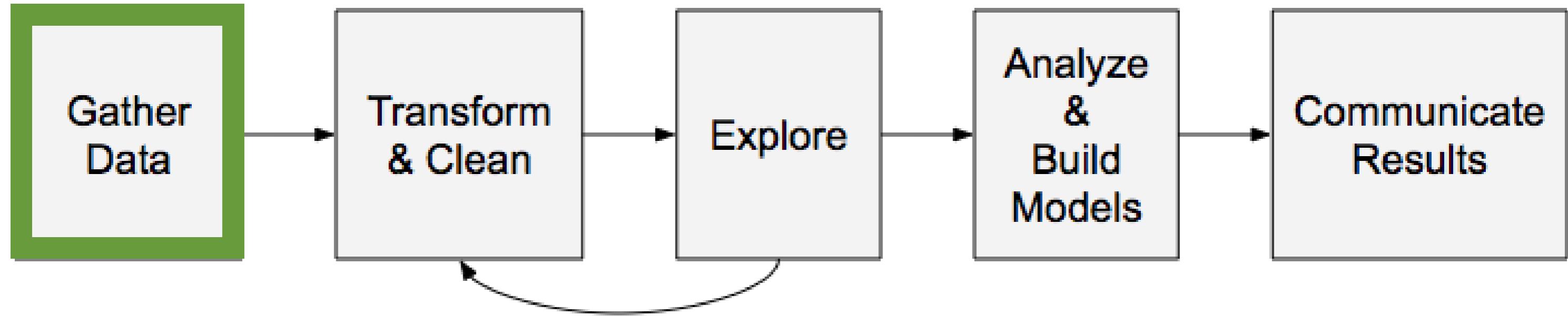
Next Steps: Explore and communicate results



Next steps:

- Seaborn advanced visualizations
- Matplotlib advanced customizations

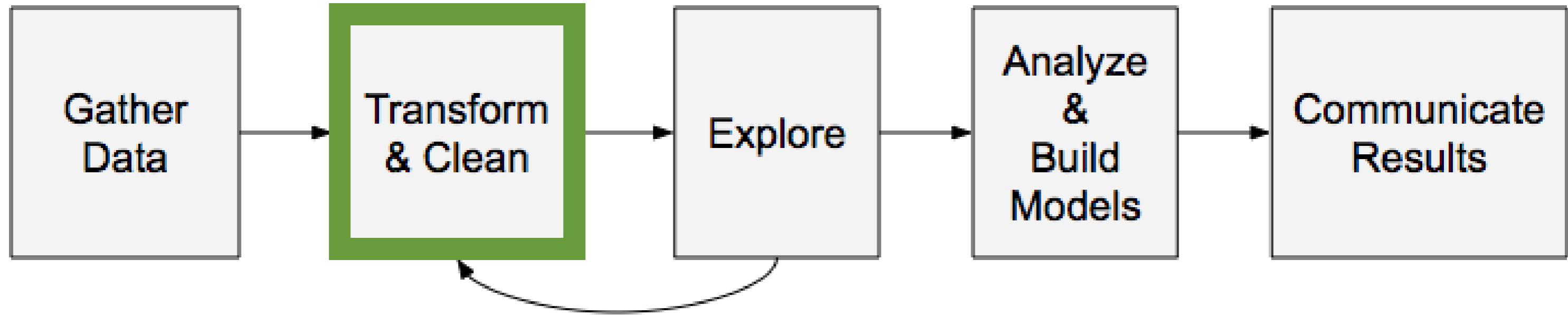
Next steps: Gather data



Next steps:

- Python
- SQL

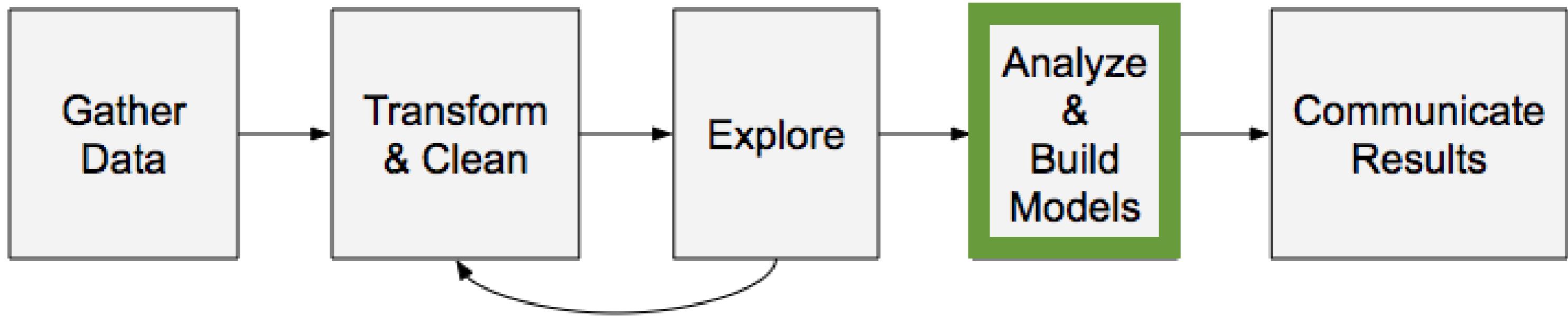
Next steps: Transform and clean



Next steps:

- Getting data into Pandas DataFrames
- Cleaning data
- Transforming into tidy format

Next steps: Analyze and build models



Next steps:

- Statistical analysis
- Calculating and interpreting confidence intervals

Congratulations!

INTRODUCTION TO DATA VISUALIZATION WITH SEABORN

Python For Data Science Cheat Sheet

Matplotlib

Learn Python Interactively at www.DataCamp.com



Matplotlib

Matplotlib is a Python 2D plotting library which produces publication-quality figures in a variety of hardcopy formats and interactive environments across platforms.



1 Prepare The Data

Also see [Lists & NumPy](#)

1D Data

```
>>> import numpy as np  
>>> x = np.linspace(0, 10, 100)  
>>> y = np.cos(x)  
>>> z = np.sin(x)
```

2D Data or Images

```
>>> data = 2 * np.random.random((10, 10))  
>>> data2 = 3 * np.random.random((10, 10))  
>>> Y, X = np.mgrid[-3:3:100j, -3:3:100j]  
>>> U = -1 - X**2 + Y  
>>> V = 1 + X - Y**2  
>>> from matplotlib.cbook import get_sample_data  
>>> img = np.load(get_sample_data('axes_grid/bivariate_normal.npy'))
```

2 Create Plot

```
>>> import matplotlib.pyplot as plt
```

Figure

```
>>> fig = plt.figure()  
>>> fig2 = plt.figure(figsize=plt.figaspect(2.0))
```

Axes

All plotting is done with respect to an Axes. In most cases, a subplot will fit your needs. A subplot is an axes on a grid system.

```
>>> fig.add_axes()  
>>> ax1 = fig.add_subplot(221) # row-col-num  
>>> ax3 = fig.add_subplot(212)  
>>> fig3, axes = plt.subplots(nrows=2, ncols=2)  
>>> fig4, axes2 = plt.subplots(ncols=3)
```

3 Plotting Routines

1D Data

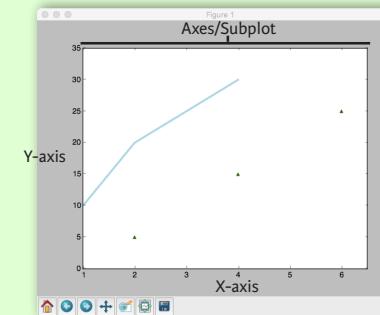
```
>>> fig, ax = plt.subplots()  
>>> lines = ax.plot(x, y)  
>>> ax.scatter(x, y)  
>>> axes[0,0].bar([1,2,3],[3,4,5])  
>>> axes[1,0].barh([0.5,1,2.5],[0,1,2])  
>>> axes[1,1].axhline(0.45)  
>>> axes[0,1].axvline(0.65)  
>>> ax.fill(x,y,color='blue')  
>>> ax.fill_between(x,y,color='yellow')
```

2D Data or Images

```
>>> fig, ax = plt.subplots()  
>>> im = ax.imshow(img,  
                  cmap='gist_earth',  
                  interpolation='nearest',  
                  vmin=-2,  
                  vmax=2)
```

Plot Anatomy & Workflow

Plot Anatomy



Figure

Workflow

The basic steps to creating plots with matplotlib are:

- 1 Prepare data
- 2 Create plot
- 3 Plot
- 4 Customize plot
- 5 Save plot
- 6 Show plot

```
>>> import matplotlib.pyplot as plt  
>>> x = [1,2,3,4]  
>>> y = [10,20,25,30] Step 1  
>>> fig = plt.figure() Step 2  
>>> ax = fig.add_subplot(111) Step 3  
>>> ax.plot(x, y, color='lightblue', linewidth=3) Step 3.4  
>>> ax.scatter([2,4,6],  
             [5,15,25],  
             color='darkgreen',  
             marker='^')  
>>> ax.set_xlim(1, 6.5)  
>>> plt.savefig('foo.png')  
>>> plt.show() Step 6
```

4 Customize Plot

Colors, Color Bars & Color Maps

```
>>> plt.plot(x, x, x, x**2, x, x**3)  
>>> ax.plot(x, y, alpha = 0.4)  
>>> ax.plot(x, y, c='k')  
>>> fig.colorbar(im, orientation='horizontal')  
>>> im = ax.imshow(img,  
                  cmap='seismic')
```

Markers

```
>>> fig, ax = plt.subplots()  
>>> ax.scatter(x,y,marker=".")  
>>> ax.plot(x,y,marker="o")
```

Linestyles

```
>>> plt.plot(x,y,linewidth=4.0)  
>>> plt.plot(x,y,ls='solid')  
>>> plt.plot(x,y,ls='--')  
>>> plt.plot(x,y,'-.',x**2,y**2,'-.')  
>>> plt.setp(lines,color='r',linewidth=4.0)
```

Text & Annotations

```
>>> ax.text(1,-2.1,  
           'Example Graph',  
           style='italic')  
>>> ax.annotate("Sine",  
               xy=(8, 0),  
               xycoords='data',  
               xytext=(10.5, 0),  
               textcoords='data',  
               arrowprops=dict(arrowstyle="->",  
                               connectionstyle="arc3"),)
```

Vector Fields

```
>>> axes[0,1].arrow(0,0,0.5,0.5)  
>>> axes[1,1].quiver(y,z)  
>>> axes[0,1].streamplot(X,Y,U,V)
```

Mathtext

```
>>> plt.title(r'$\sigma_i=15$', fontsize=20)
```

Limits, Legends & Layouts

```
>>> ax.margins(x=0.0,y=0.1)  
>>> ax.axis('equal')  
>>> ax.set(xlim=[0,10.5],ylim=[-1.5,1.5])  
>>> ax.set_xlim(0,10.5)
```

Legends

```
>>> ax.set(title='An Example Axes',  
           ylabel='Y-Axis',  
           xlabel='X-Axis')  
>>> ax.legend(loc='best')
```

Ticks

```
>>> ax.xaxis.set(ticks=range(1,5),  
                  ticklabels=[3,100,-12,"foo"])  
>>> ax.tick_params(axis='y',  
                           direction='inout',  
                           length=10)
```

Subplot Spacing

```
>>> fig3.subplots_adjust(wspace=0.5,  
                           hspace=0.3,  
                           left=0.125,  
                           right=0.9,  
                           top=0.9,  
                           bottom=0.1)
```

```
>>> fig.tight_layout()
```

Axis Spines

```
>>> ax1.spines['top'].set_visible(False)  
>>> ax1.spines['bottom'].set_position(('outward',10))
```

Add padding to a plot
Set the aspect ratio of the plot to 1
Set limits for x-and y-axis
Set limits for x-axis

Set a title and x-and y-axis labels

No overlapping plot elements

Manually set x-ticks

Make y-ticks longer and go in and out

Adjust the spacing between subplots

Fit subplot(s) in to the figure area

Make the top axis line for a plot invisible

Move the bottom axis line outward

5 Save Plot

Save figures

```
>>> plt.savefig('foo.png')
```

Save transparent figures

```
>>> plt.savefig('foo.png', transparent=True)
```

6 Show Plot

```
>>> plt.show()
```

Close & Clear

```
>>> plt.cla()
```

```
>>> plt.clf()
```

```
>>> plt.close()
```

Clear an axis
Clear the entire figure
Close a window



Data Science Career Track

Introduction to Data Science

 Developer Student Clubs
Al-Azhar University



Some Basic Rules Before we Start!



AGENDA



01

What is Data Science

What's Data Science is
All about?

02

Data Science Ecosystem

Data science World and
commercial landscape

03

Data Science Building Blocks

and Team Roles and skills

04

Data Science Methodology

Steps of a Data science
Project

HELLO!

Meet Mohamed

I am here because I love to Start a Career in Data Science!

Join me throughout my journey ☺!



1

What is Data Science



What's Data Science is All about?

Data

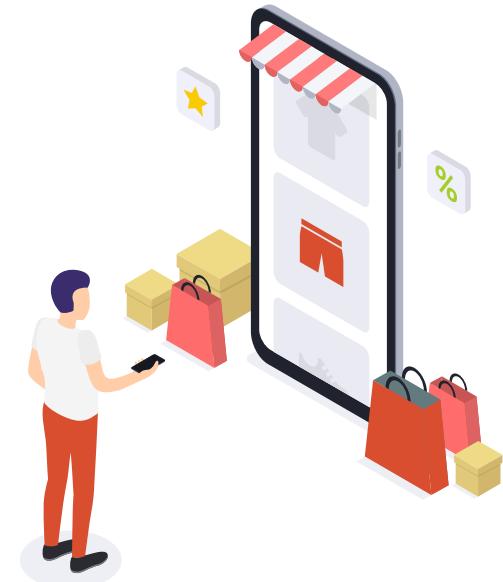
They care about volume and velocity and whatever other buzzwords describe data that is too big for you to analyze in Excel.



Science

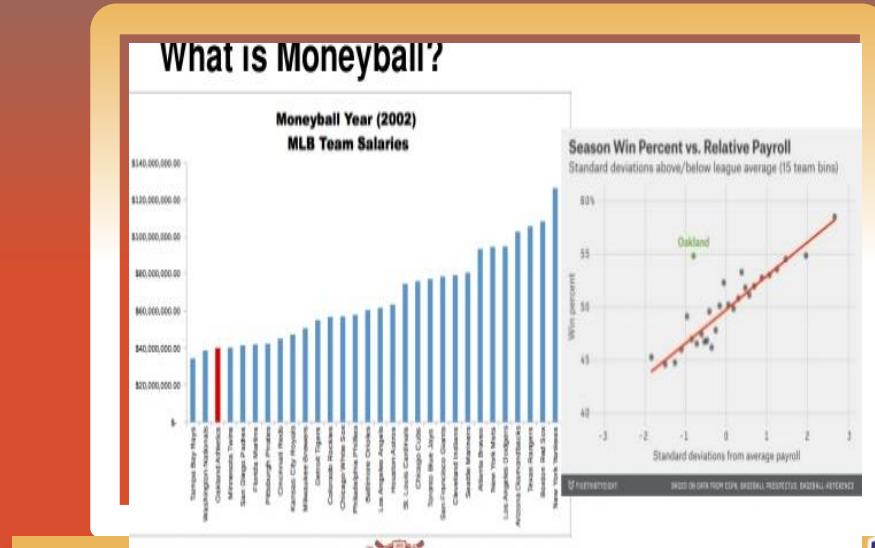
Data science is only useful when the data are used to answer a question.

“I have this really hard question, can I *answer it* with my data?”



Moneyball

can we build a winning baseball team if we have a really limited budget?.



Voter Turnout

“How do we find the people who vote for Barack Obama and make sure that those people end up at the polls on polling day?”



“The goal is to turn data into information, and information into insight.”

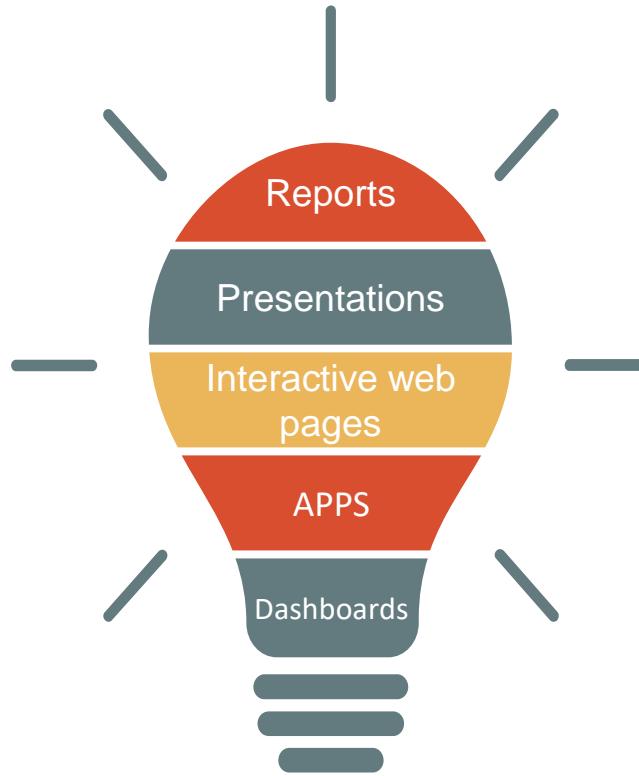
—Carly Fiorina

“Numbers have an important story to tell. They rely on you to give them a voice.”

—Stephen Few



The outputs of a data science Projects



2

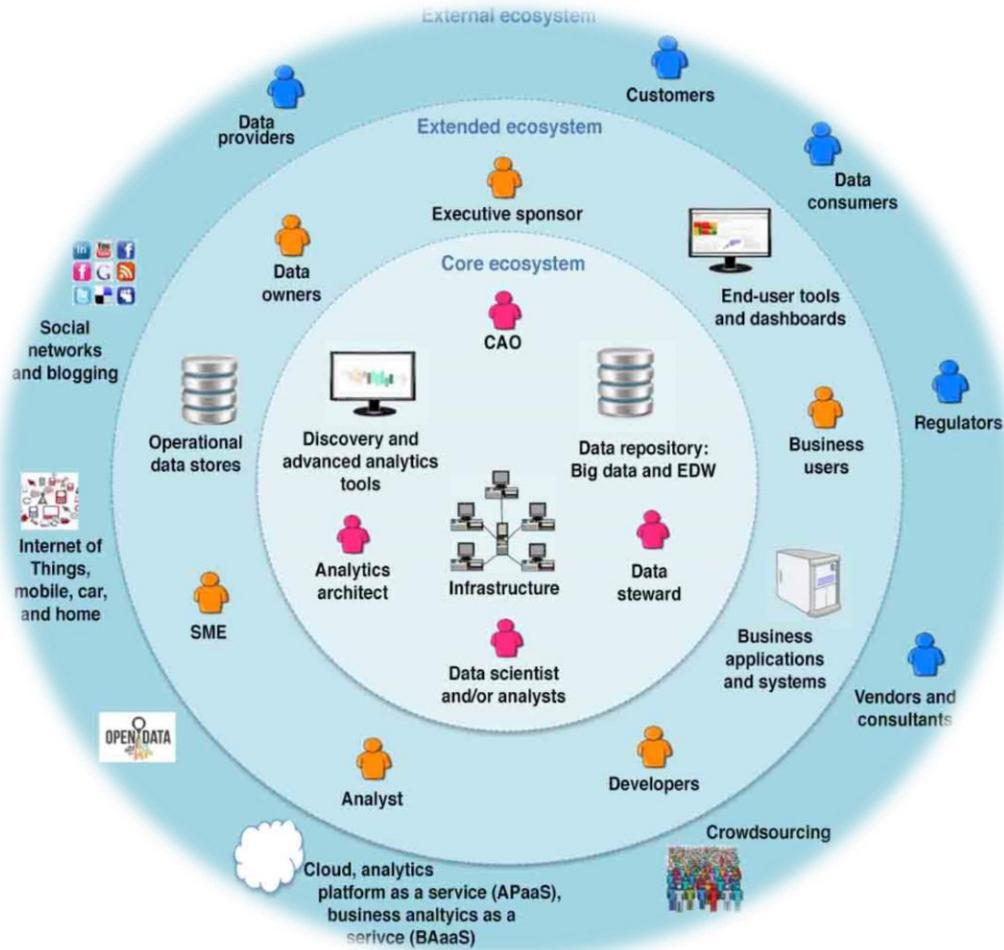
Data Science Ecosystem



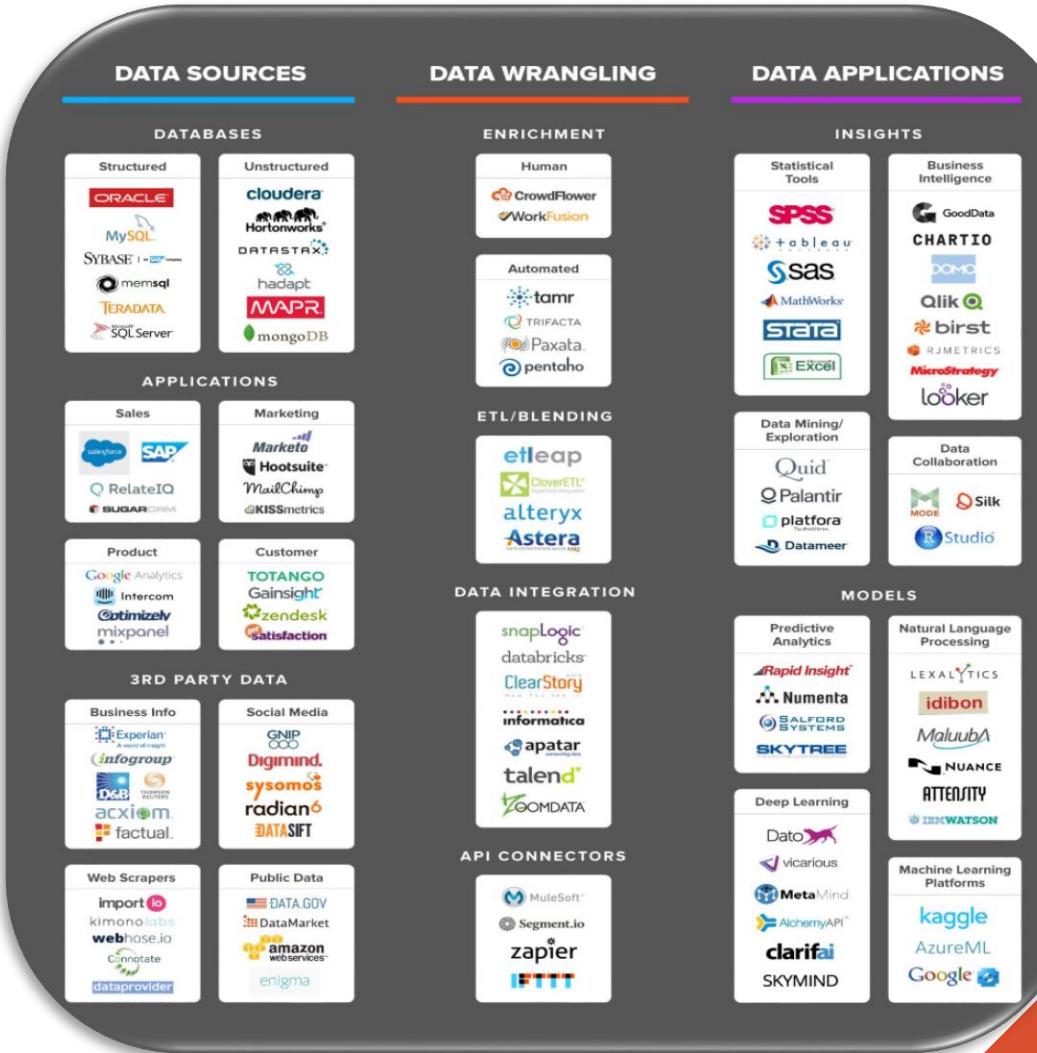
Data Ecosystem

An ecosystem: is the network of organizations—including suppliers, distributors, customers, competitors, government agencies, and so on—involved in the delivery of a specific product or service through both competition and cooperation.

Source: [Investopedia.com](https://www.investopedia.com/terms/e/data_ecosystem.asp)

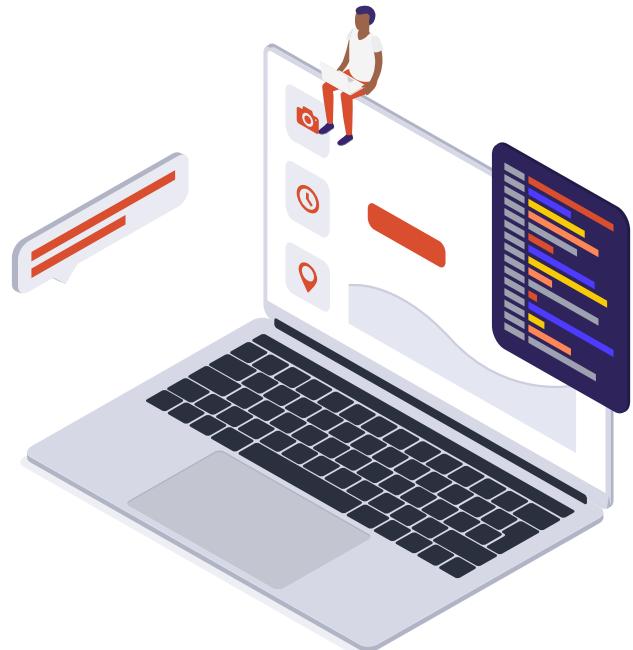


Commercial Landscape



3

Data Science Building Blocks



Data Science Building Blocks

Math And Statistics

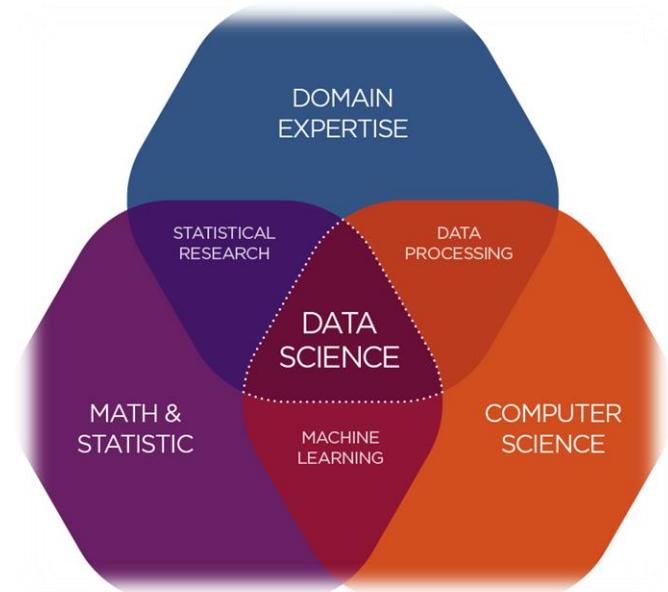
the foundation of knowledge and techniques that are used in Data analysis and inference.

Human Expertise

the knowledge and experience of the business domain problems where data science will try to solve.

Computer Science

knowledge of programming languages, data structures, data bases and algorithms .



Data Analysis

- Feature Engineering
- Data wrangling
- EDA

Machine Learning

- Classification
- Regression
- Reinforcement learning
- Deep learning
- Clustering

Math

- Statistics
- Linear Algebra
- Differential calculus

Programming Language

- Python
- R



Deploy

- AWS
- AZURE

Data Visualization

- Tableau
- Power BI
- Matplotlib
- Seaborn
- ggplot
- D3.js
- Gephi

IDE

- Jupyter
- Colaboratory
- Spyder
- PyCharm

Web Scraping

- Scrapy
- URLLIB
- Beautiful Soup

3

Data Science Team

Data science is a team sport



Data Sciene “Unicorn”

The **data science unicorn** is a somewhat mythical person who is a leader in **data science**, technology, and business. ...



Data Scientist

What does a data scientist do?

Design experiments

Pull and clean data

Analyze data

Communicate results

MODERN DATA SCIENTIST

Data Scientist, the sexiest job of 21th century requires a mixture of multidisciplinary skills ranging from an intersection of mathematics, statistics, computer science, communication and business. Finding a data scientist is hard. Finding people who understand who a data scientist is, is equally hard. So here is a little cheat sheet on who the modern data scientist really is.

MATH & STATISTICS

- ★ Machine learning
- ★ Statistical modeling
- ★ Experiment design
- ★ Bayesian inference
- ★ Supervised learning: decision trees, random forests, logistic regression
- ★ Unsupervised learning: clustering, dimensionality reduction
- ★ Optimization: gradient descent and variants



PROGRAMMING & DATABASE

- ★ Computer science fundamentals
- ★ Scripting language e.g. Python
- ★ Statistical computing package e.g. R
- ★ Databases SQL and NoSQL
- ★ Relational algebra
- ★ Parallel databases and parallel query processing
- ★ MapReduce concepts
- ★ Hadoop and Hive/Pig
- ★ Custom reducers
- ★ Experience with xaaS like AWS

DOMAIN KNOWLEDGE & SOFT SKILLS

- ★ Passionate about the business
- ★ Curious about data
- ★ Influence without authority
- ★ Hacker mindset
- ★ Problem solver
- ★ Strategic, proactive, creative, innovative and collaborative

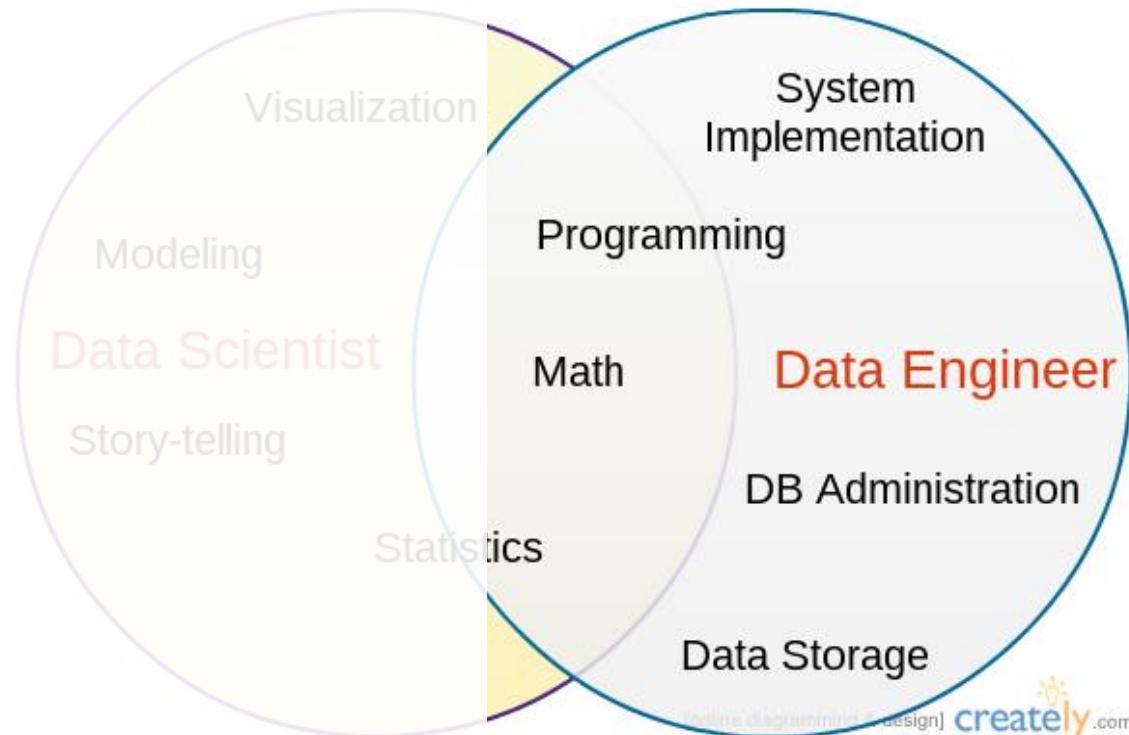
COMMUNICATION & VISUALIZATION

- ★ Able to engage with senior management
- ★ Story telling skills
- ★ Translate data-driven insights into decisions and actions
- ★ Visual art design
- ★ R packages like ggplot or lattice
- ★ Knowledge of any of visualization tools e.g. Flare, D3.js, Tableau

Data Engineer

What does a data engineer do?

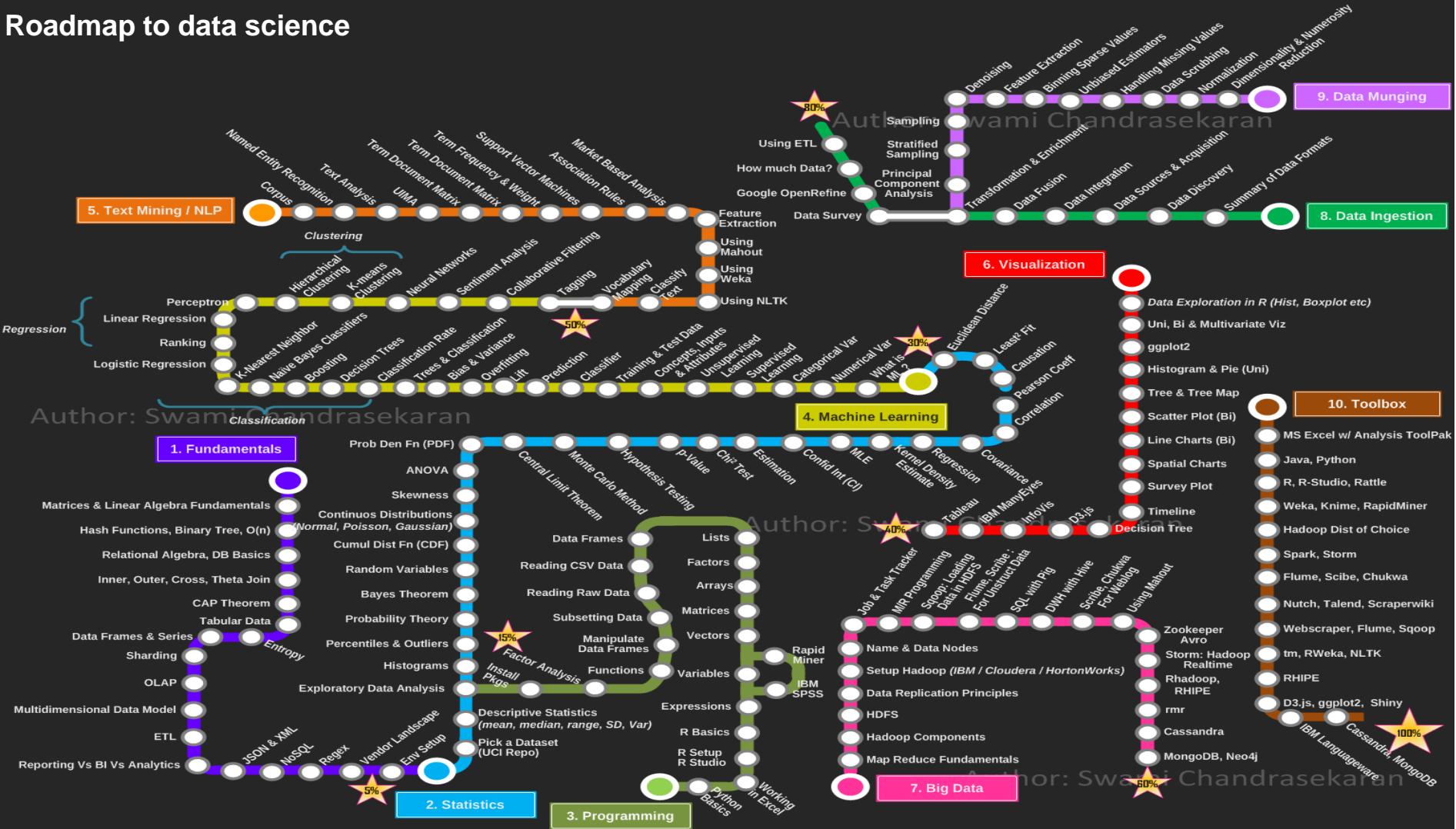
- Build data infrastructure
- Manage data storage and use
- Implement production tools



Data Science is a Marathon



Roadmap to data science

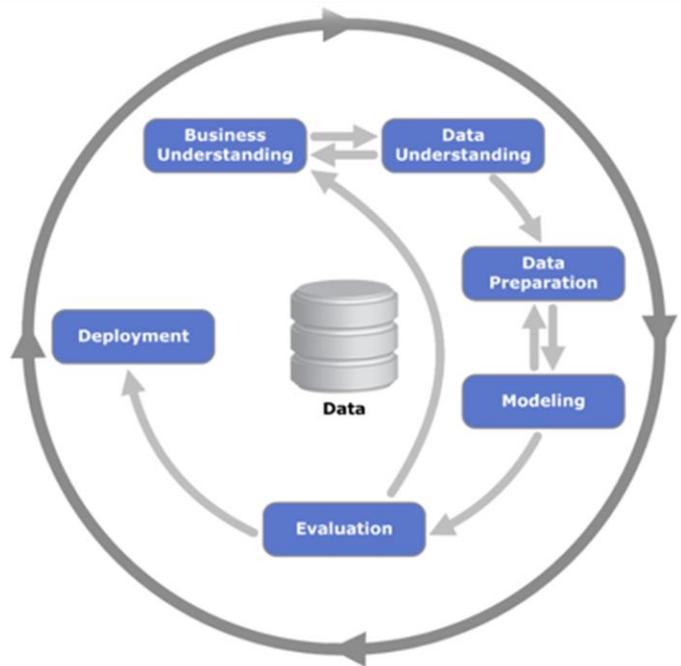


4

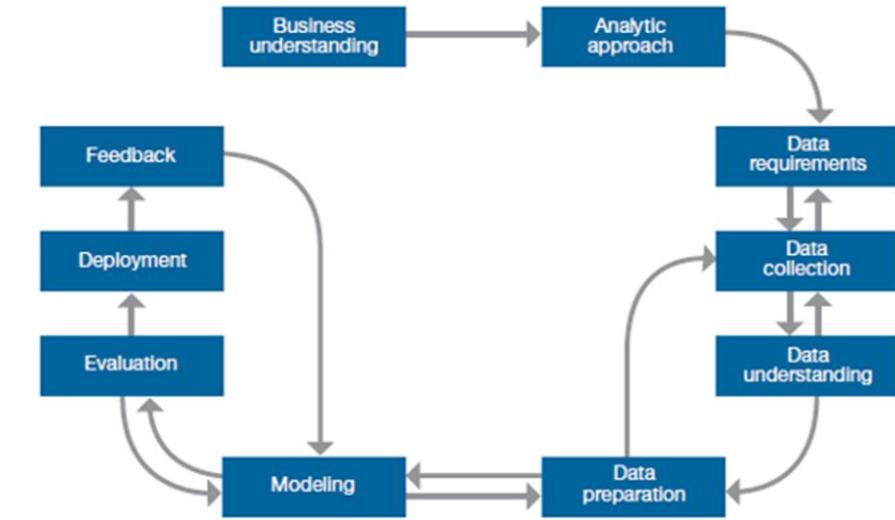
Data Science Methodology



CRISP-DM



Cross Industry Standard Process for Data Mining



IBM Data Science Methodology

From Problem to Approach

Every project begins **with business understanding.**

- ▶ Project objective?
- ▶ Business sponsors play the most critical role
- ▶ What are we trying to do – what is the goal?
- ▶ How do you define “success” and how can you measure it?

Example : Traffic

Problem: Traffic congestion wastes time and money

Clear question: How can we optimize traffic light duration using data on traffic patterns, weather, and pedestrian traffic?

Measurable outcomes:

- % decrease in commute time
- % decrease in length/duration of traffic jams

From Problem to Approach

Analytic Approach

Expand machine learning techniques

Regression:

“Predicting press problem in context of statistical revenue in the next quarter?”

Classification:

“Does this patient have cancer A, cancer B, or are they healthy?”

Clustering:

“Are there groups of users that seem to behave similarly to each other?”

Recommendation/Personalization:

“How can I target discounts to specific customers?”

Outlier Detection

From Requirements to Collection

The chosen analytic approach determines the **data requirements**.

- ▶ Content,
- ▶ formats,
- ▶ representations

Initial **data collection** is performed.

- Available Data?
- Obtain data?
- Revise data requirements or collect more data?

From Understanding to Preparation

Then **data understanding** is gained.

- Initial insights about data
- Descriptive statistics and visualization
- Additional data collection to fill gaps, if needed

Data preparation encompasses all activities to construct and clean the data set.

Data cleaning

- Missing or invalid values
- Eliminating duplicate rows
- Formatting properly

Combining multiple data sources

Transforming data

Feature engineering

Text analysis

Accelerate data preparation by automating common steps

From Modeling to Evaluation

Modeling:

- ▶ Developing predictive or descriptive models
- ▶ May try using multiple algorithms
- ▶ Highly iterative process

Model **evaluation** is performed during model development and before model deployment

- Understand the model's quality
- Ensure that it properly addresses the business problem

Diagnostic measures

- Suitable to the modeling technique used
- Training/Testing set
- Refine model as needed

From Deployment to Feedback

Once finalized, the model is **deployed** into a production environment.

- ▶ • May start in a limited / test environment

Involves other roles:

- ▶ Solution owner
- ▶ Marketing
- ▶ Application developers
- ▶ IT administration

Getting Feedback :

- How well did the model perform?
- Iterative process for model refinement and redeployment
- A/B testing

Let's start our Data science Hands on Journey!

Stay Tuned😊



MacBook Air



THANKS!

Any questions?



Intro to Descriptive Statistics

By: Mahmoud Galal



Developer Student Clubs
Al-Azhar University

AGENDA

01

History



Knowing the past ...
Respecting the present.

03

Summary Statistics



Representing a great sum of
data with just one number !

02

Data Types and Terminologies



Knowing the type of the data
is knowing what to do with it.

04

Exploratory Data Analysis



Using the swords we mastered to
slice down the beast, Just to know
him better (We are not savages !)

-- It's said that data is the biggest beast of our age --

01

History...

Knowing the past ... **Respecting** the present.

How statistics evolved ?



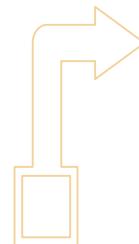
Until the 18th century data was collected and putted into dummy tables.



From the 19th century and onwards, Life got complicated, and people started to think about how to collect, summarize and present the data.



Nowadays, statistics has evolved to contribute in every field of study in our lives and even just Fun !



Founders ...



Abu Youssef Yaaqob Al Kindi Developed the first code breaking algorithm based on **frequency analysis**. He wrote a book entitled "Manuscript on Deciphering Cryptographic Messages", containing detailed discussions on **statistics**.



Check rest of the founder like Bayes , Laplace and Gauss here.



أبو يوسف يعقوب بن إسحاق الصبّاح الكندي



Who got it?

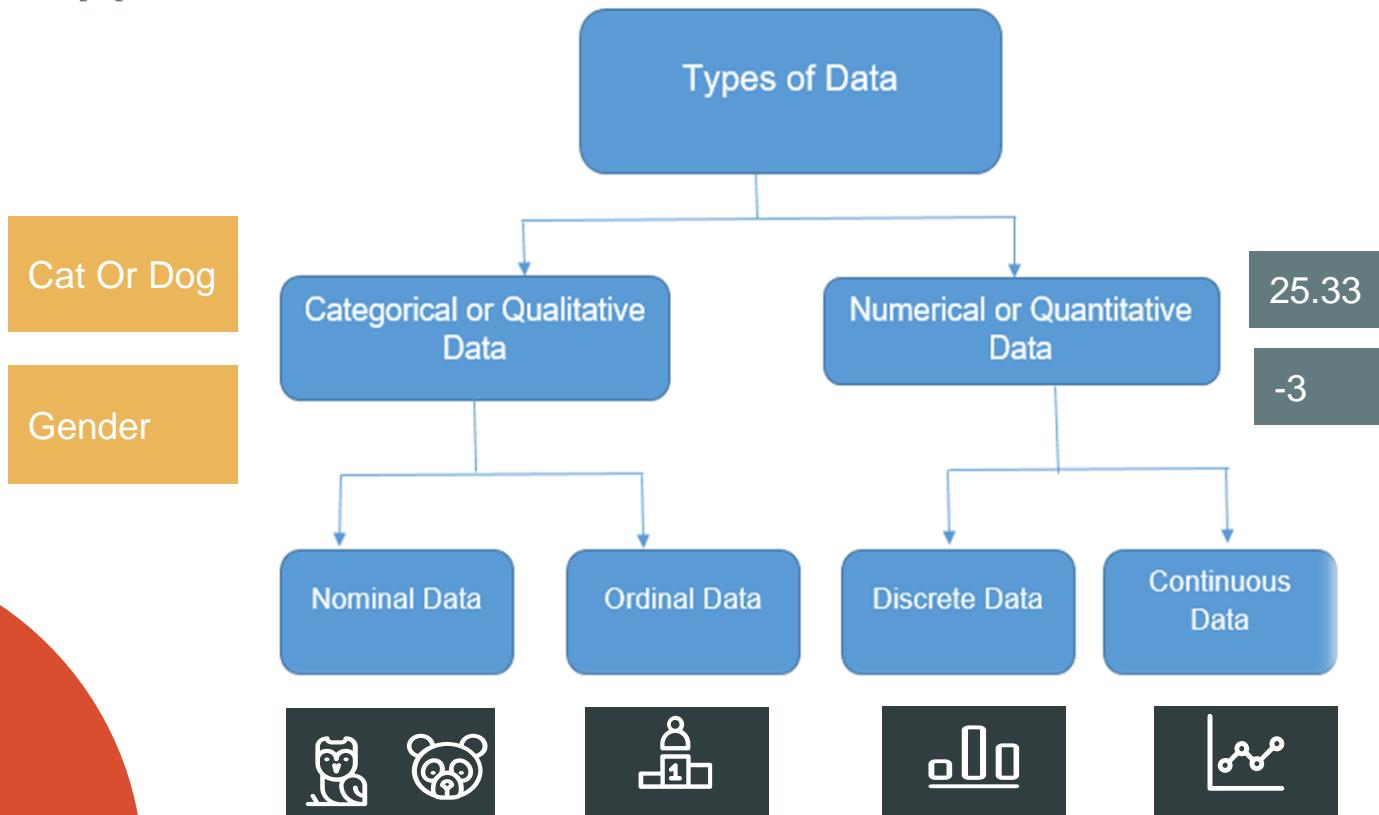


02

Data Types and Terminologies

Knowing the type of the data
is knowing what to do with it.

Data Types



Data Terminologies

- Variable are also called Column, Feature, Dimension, field and Attribute.
- Samples are also called Observations, Records, Instances and rows.
- Variables and Samples make up the term “Data Set” or “Data Frame”.

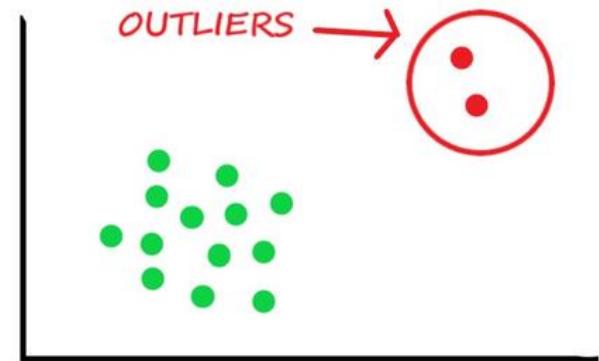
Samples

Variables

Country	Age	Score
Egypt	30	4
Morocco	21	4
Germany	29	3

Outliers

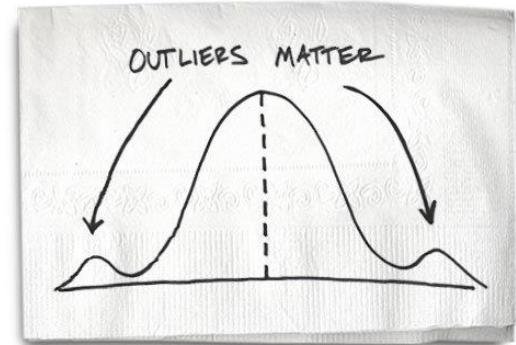
1. An outlier is a data point that differs **significantly** from other observations.
2. We usually tend to remove the outliers to make sure that we are making accurate analysis.
3. **Outliers** can cause serious problems in statistical analyses



Removing Outliers

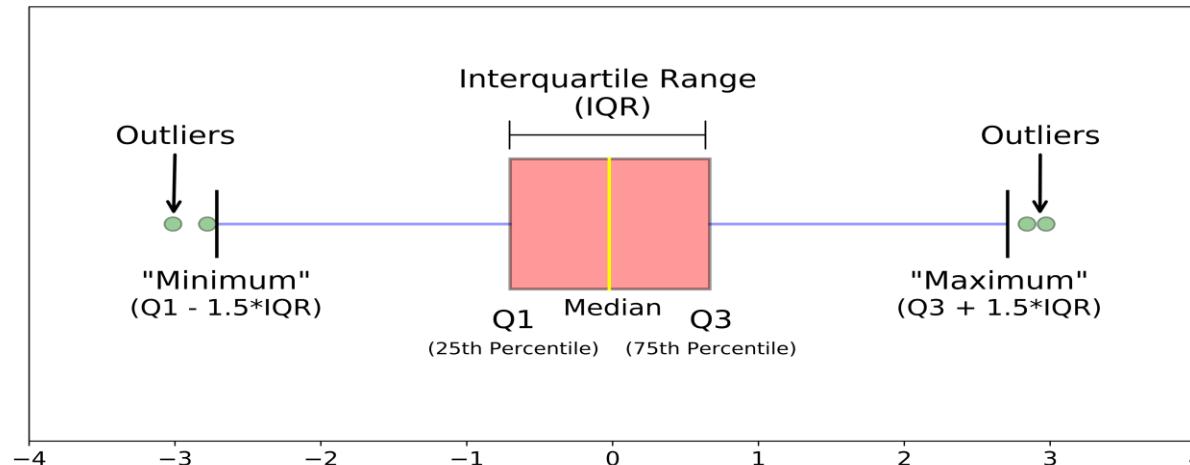
There are different techniques to capture outliers

1. We usually tend to remove the outliers to make sure that we are making accurate analysis, But other times we keep them. Example: **Fraud Detection Applications.**
2. We can **normalize** our outliers to make them look like the majority, not to remove them if their removal will have bad effects like in case of small datasets.
3. So we have to always visualize our data and analyze it properly before making any moves



Removing Outliers – Tukey's Method

- One of the most used methods to detect and remove outliers is the Tukey's method.
- First We calculate the IQR like $IQR = Q3 \text{ value} - Q1 \text{ value}$
- Then any data points $< (Q1 - 1.5 * IQR)$ and $> (Q3 + 1.5 * IQR)$ is considered an outlier.
- Outliers are X , Where $(Q3 + 1.5 * IQR) < X < (Q1 - 1.5 * IQR)$



Real-World Example



03

Summary Statistics

“ONE NUMBER TO RULE THEM ALL ,
ONE NUMBER TO FIND THEM ALL ,
ONE NUMBER TO BRING THEM ALL ”

-Gandalf The Grey

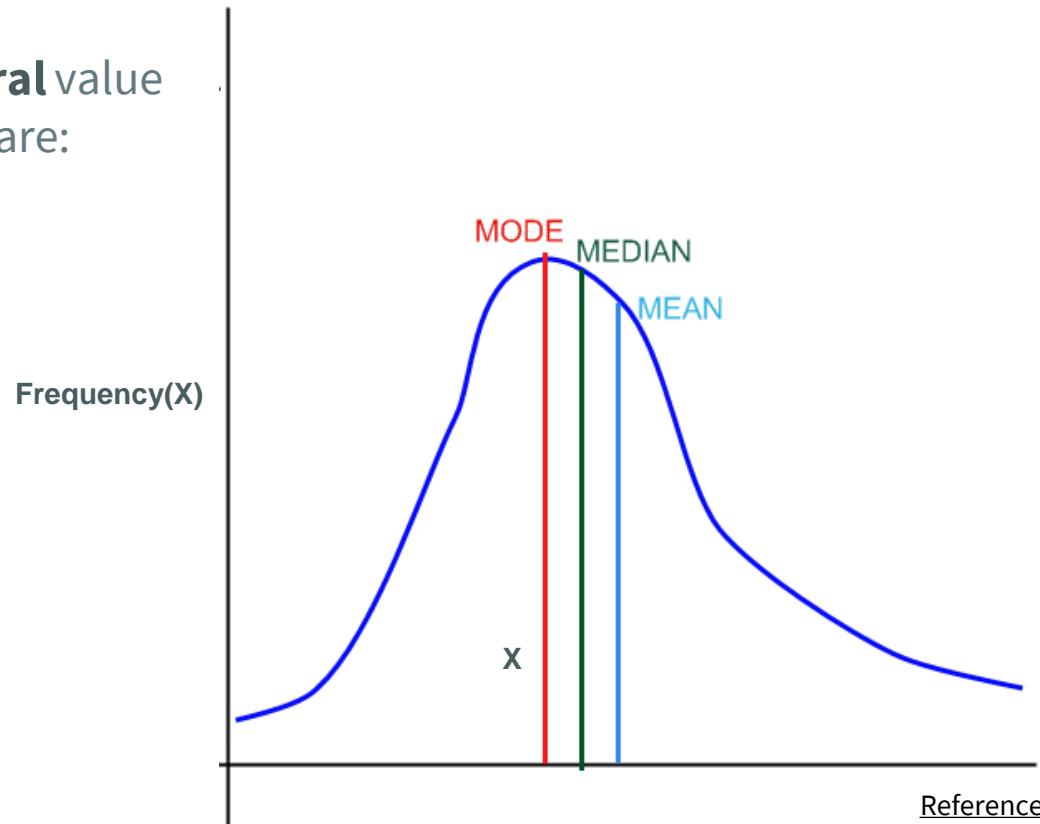
Summary statistics types

- A sample summary is a **Statistic**, A population summary is a **Parameter**.
- We can summarize our data with different measures. Each of them adds a certain power to the analysis.
 1. Measures of location (Mean, Median, mode)
 2. Measures of spread (Min, Max, Variance and Standard Deviation)
 3. Measures of shape (Skewness and Kurtosis)

Measures of location (Center)

It's the measures that describes the **central** value of a data set, And Its most popular forms are:

1. Mean
2. Median
3. Mode



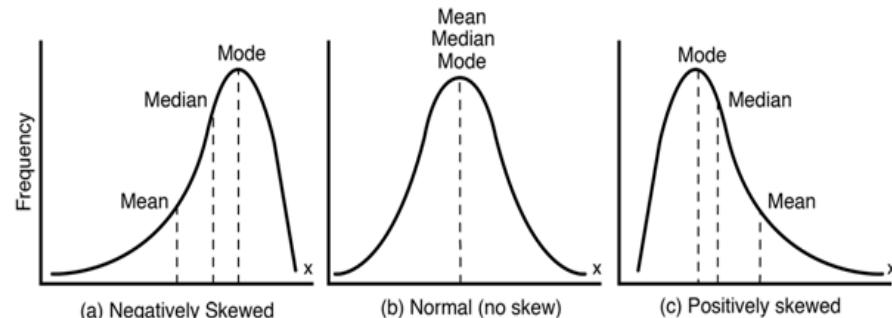
Mean

It's the sum of all values of the data set divided by its records number

1. It's the **simplest** computed summary statistic.
2. Suitable for general-purposed analysis.
3. **Can be computed algebraically.** Median and Mode can not be algebraically manipulated.
4. The mean is more widely used than median and mode.
5. Very Sensitive to outliers and **skewness**.
6. Can't handle **non-numeric** features.
7. **Catches the variability** of data points.

Population Mean	Sample Mean
$\mu = \frac{\sum_{i=1}^N x_i}{N}$	$\bar{X} = \frac{\sum_{i=1}^n x_i}{n}$
N = number of items in the population	n = number of items in the sample

[Reference](#)



Practice

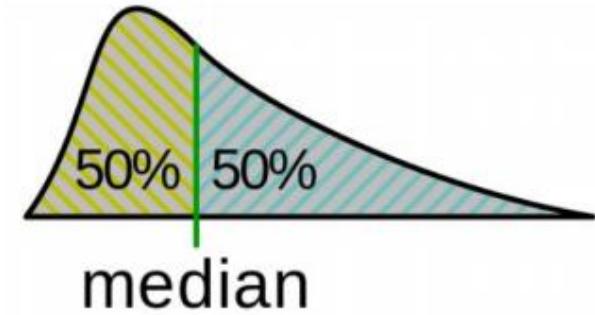
5-Min Break

Please submit your attendance while
Making yourself something hot ☕

Median

It's the middle value of our data set.

1. The median value is fixed by its position and is not reflected by the individual value.
2. Can be used to determine an approximate average if there were outliers in the data.
3. Can't be computed Algebraically.
4. Before applying the law of the median, **the data must be sorted first.**

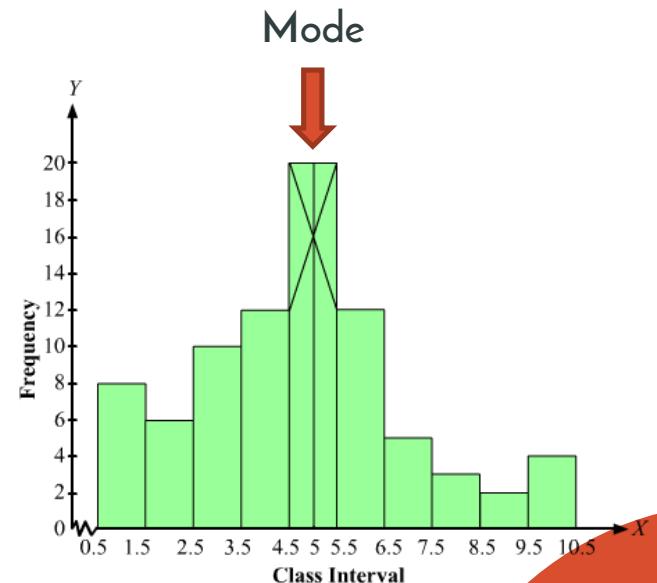


$$\text{Median} = \begin{cases} \frac{(N+1)^{\text{th}}}{2} \text{ term; when } N \text{ is odd} \\ \frac{N^{\text{th}}}{2} \text{ term} + \left(\frac{N}{2} + 1 \right) \text{ term} \\ \hline 2 \end{cases}; \text{when } N \text{ is even}$$

Mode

It's the element that appeared the most in our dataset.

1. We can have **multiple modes** in the dataset.
2. Unlike mean, it has no mathematical property
3. Unlike mean, Mode is **affected by sampling fluctuations**.
4. It's the most suitable measure for **nominal data**.



Practice

Pros and Cons

	Outliers Sensitive?	Algebraic Manipulation	Qualitative Expression	Fluctuations of sampling
Mean	✓	✓	✗	✗
Median	✗	✗	✓	✓
Mode	✗	✗	✓	✓✓

7,540,860,914

Just imagine a dataset with this number of rows...



1

CONCLUSION

To summarize our lecture we can say:

1. It's strongly believed that Arabs are the pioneers of statistics.
2. Outliers are generally bad for our analysis but sometimes they are the most important.
3. Summary statistics is a must when working with data.
4. Mean is the most popular measure of location or center.
5. We can use the other summaries like median and mode in special cases like outliers presence.
6. When data are on interval scale the suitable measure of central tendency is mean. Median is suitable when data are on ordinal scale. Mode is calculated when data are on nominal scale.

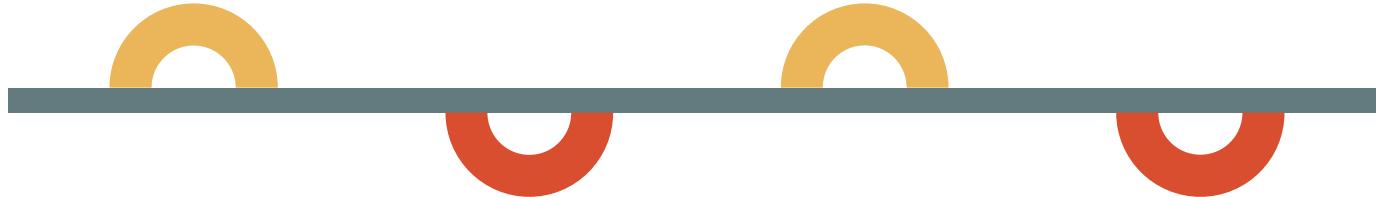
What is coming?

Measures of Spread

Dataset Variability

Visuals

Box plot,
histograms ..etc



Measures of Shape

Skewness and
Kurtosis

Full EDA

Practising what we
have learned.

“Statistics is the grammar of Sceince

–Karl Pearson (Who is he ?)

THANKS

Any Questions? ...

Head to google meet !

Intro to Descriptive Statistics

2

By: Mahmoud Galal



AGENDA

01

Recap



02

Measures of Spread



03

Measures of shape
and Visuals



04

Exploratory Data Analysis



Recap





Summary Statistics

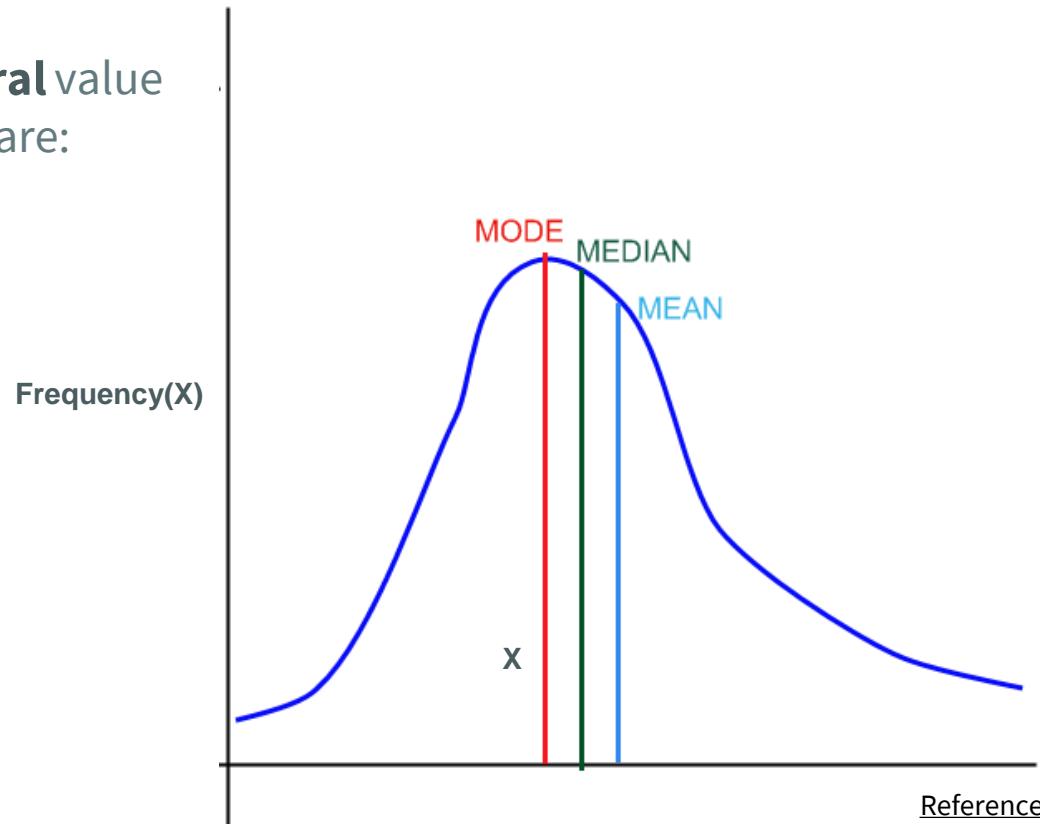
“ONE NUMBER TO RULE THEM ALL ,
ONE NUMBER TO FIND THEM ALL ,
ONE NUMBER TO BRING THEM ALL ”

-Gandalf The Grey

Measures of location (Central Tendency)

It's the measures that describes the **central** value of a data set, And Its most popular forms are:

1. Mean
2. Median
3. Mode



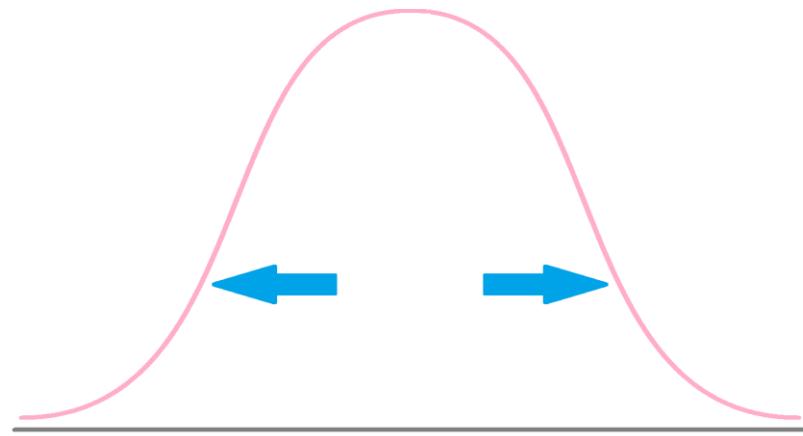
Summary statistics types

- A sample summary is a **Statistic**, A population summary is a **Parameter**.
- We can summarize our data with different measures. Each of them adds a certain power to the analysis.
 1. Measures of location (Mean, Median, mode)
 2. Measures of spread (Min, Max, Variance and Standard Deviation)
 3. Measures of shape (Skewness and Kurtosis)

Measures of Spread

Describe how dispersed or varied data is. We can see measures of spread in these forms:

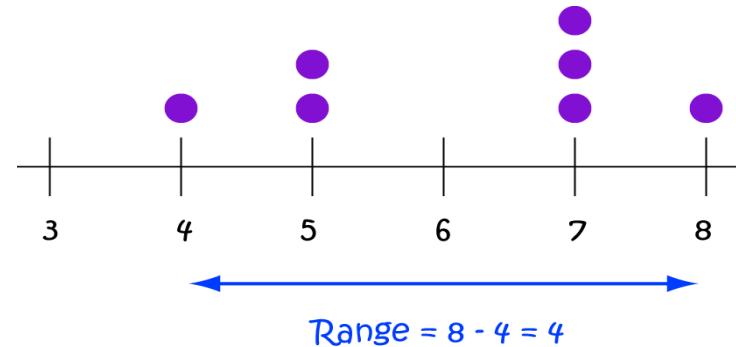
1. Ranges (Maximum - Minimum)
2. IQR
3. Variance
4. Standard Deviations



Ranges

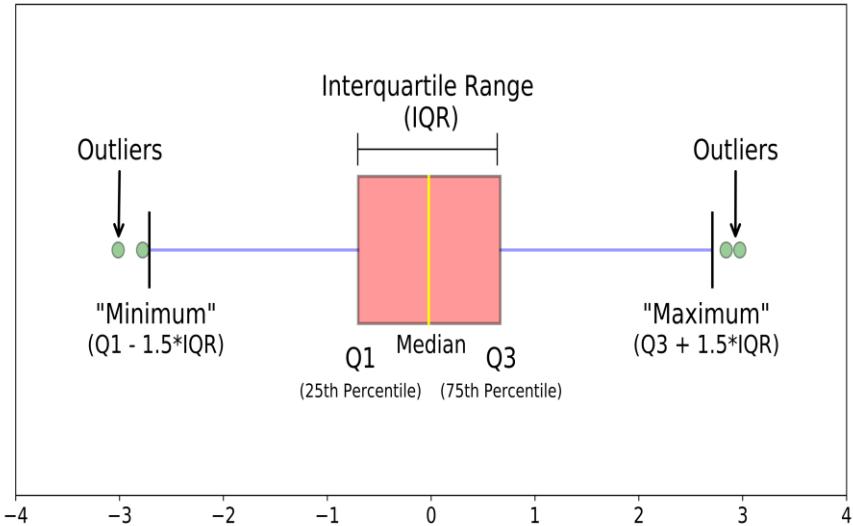
The range is the **simplest** measure of variability to compute, It's simply the **difference between the Maximum Value and the minimum.**

1. It provides a simple view of how varied our data set is.
2. The range can also be used to **estimate** the standard deviation (**The Range Rule**).
3. It's highly sensitive to **outliers**.
4. The range also tells us nothing about the **internal features** of our data set.



The Interquartile range

- We can use the box-plot and the IQR value to represent the variability.
- Remember that $\text{IQR} = Q_3 - Q_1$
- The problem is that IQR doesn't represent the **whole dataset** (Just 50% of it), so it won't be an accurate representation of the dataset's variability.
- It also doesn't tell us nothing about the **internal features** of our data set just like the Range.
- We need to include all of our data sets in the computation ... How ?



Measuring variability using all data points

Ranges doesn't give the full picture, so we have to use all of our data points' values to represent the dataset variability. We can think of multiple ways.

1. Find the average distance between any two value.
 - It's inefficient due to the big amount values
2. Find the average between every minimum and maximum value.
 - It's inefficient due to the lack of standardization.
3. Find the average distance between every value and the mean value.
 - And that is the most effective way.

Variance

Variance (σ^2) a measurement of the spread between each number and the average numbers in the dataset.

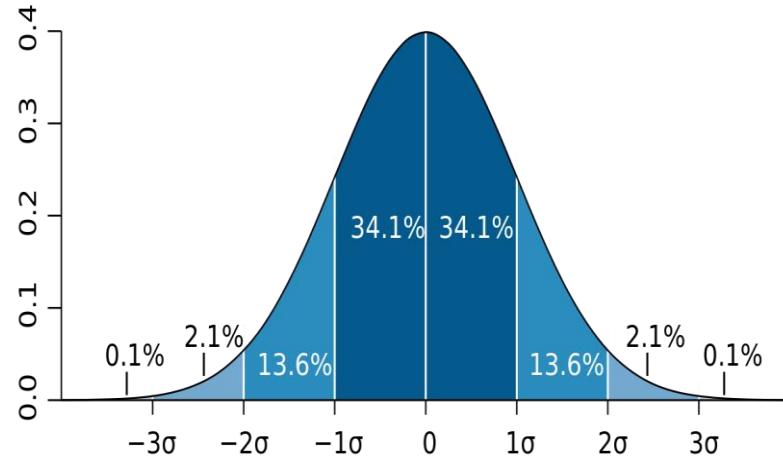
- To calculate the **population variance** (σ^2), we take the sum of distances between each data point and the population mean divided by the total number of data points.
- In case of **sample variance**, there is a tiny difference. That we divide by the total number of data points in the sample – 1 ($n-1$). And that is called **Bessel's Correction**.
- This correction is made to correct for the fact that these **sample statistics tend to underestimate** the actual parameters found in the population.

Population Variance	Sample Variance
$\sigma^2 = \frac{\sum_{i=1}^N (x_i - \mu)^2}{N}$ σ^2 = population variance x_i = value of i^{th} element μ = population mean N = population size	$s^2 = \frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n-1}$ s^2 = sample variance x_i = value of i^{th} element \bar{x} = sample mean n = sample size

Standard Deviation

Standard Deviation (σ) is the most common measure of spread for its robustness and unified measurements.

- It's the same calculations of the Variance except we take the **squared root** of the variance output.
- In a **standardized distribution**, we found that:
 - 68% of data lies in 1σ from the mean.
 - 95% lies in 2σ from the mean.
- So with standard deviation we can know how much data falls in one area. And that helps us making our assumptions and tests.



$$\sigma = \sqrt{\frac{\sum(x_i - \mu)^2}{N}}$$

σ = population standard deviation

N = the size of the population

x_i = each value from the population

μ = the population mean

Statistics Summary

\$33

The middle price
(Median)

\$28

Avg. price (Mean)

\$ 50

Most Frequent Price
(Mode)

\$7

Standard Deviation

\$452

Maximum price

5-Min Break

Please submit your attendance while
Making yourself something hot ☕

Measures of Shape

Skewness

Skewness refers to distortion or asymmetry in a symmetrical bell curve, or normal distribution, in a dataset.

- A symmetrical distribution will have a **skewness** of 0.
- If skewness is **positive**, the data is **right-skewed**, meaning that the right tail of the distribution is longer than the left.
- If skewness is **negative**, the data are negatively skewed or **left-skewed**, meaning that the left tail is longer
- In negative skewness, the value of the skewness is < 0 .
- In positive skewness, the value of the skewness is > 0 .

$$\tilde{\mu}_3 = \frac{\sum_i^N (X_i - \bar{X})^3}{(N - 1) * \sigma^3}$$

$\tilde{\mu}_3$ = skewness

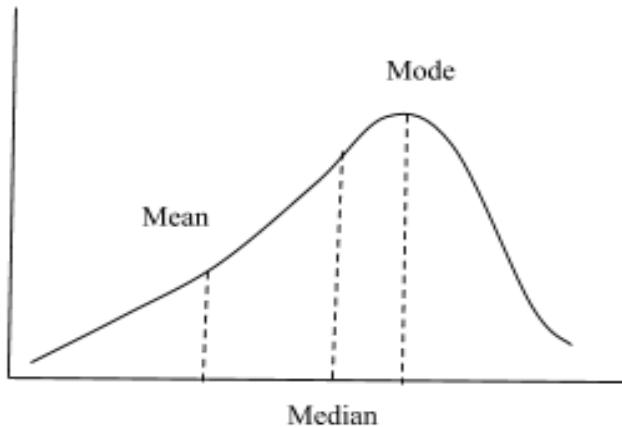
N = number of variables in the distribution

X_i = random variable

\bar{X} = mean of the distribution

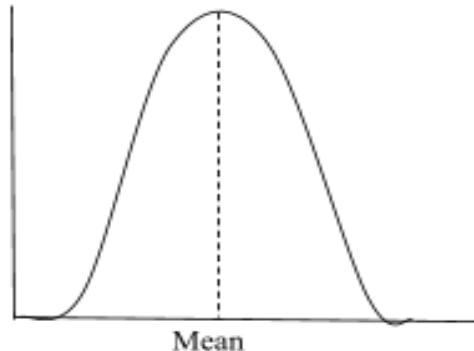
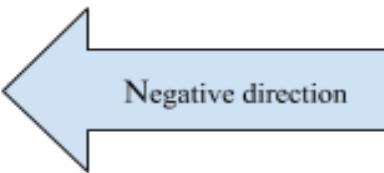
σ = standard deviation

Skewness

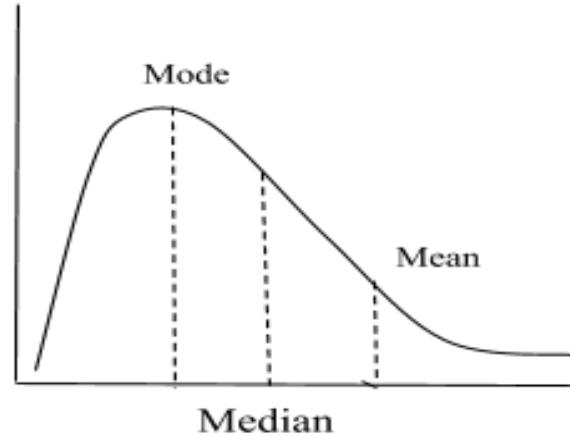


$\text{mean} < \text{median} < \text{mode}$

Negative direction

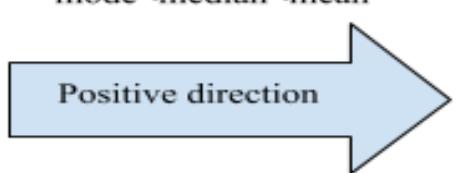


Symmetrical data
 $\text{mean} = \text{median} = \text{mode}$



$\text{mode} < \text{median} < \text{mean}$

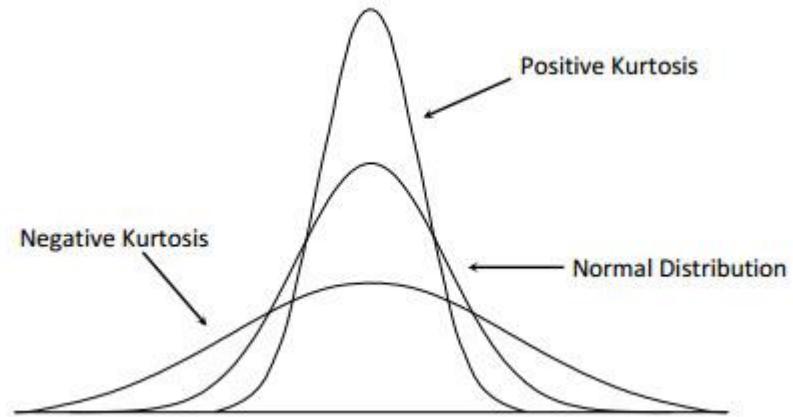
Positive direction



Kurtosis

Kurtosis is a statistical measure used to describe the degree to which scores cluster in the tails or the peak of a frequency distribution.

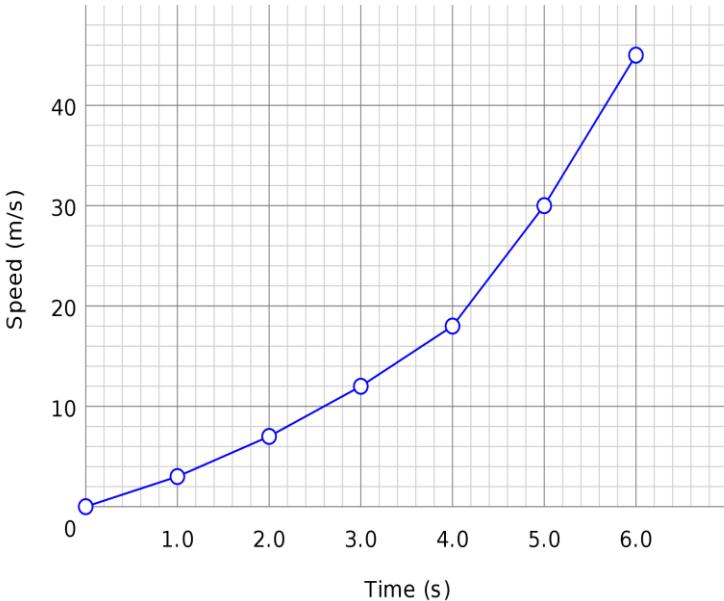
- **Low kurtosis** is an indicator that data has light tails or **lack of outliers**. It's called (**PlatyKurtic**)
- Datasets with **high kurtosis** tend to have heavy tails, and it indicates to **outliers**. It's called (**LeptoKurtic**)
- It's closely related to skewness as both represents measures of distribution shapes.
- **Read More**



The power of Visualization

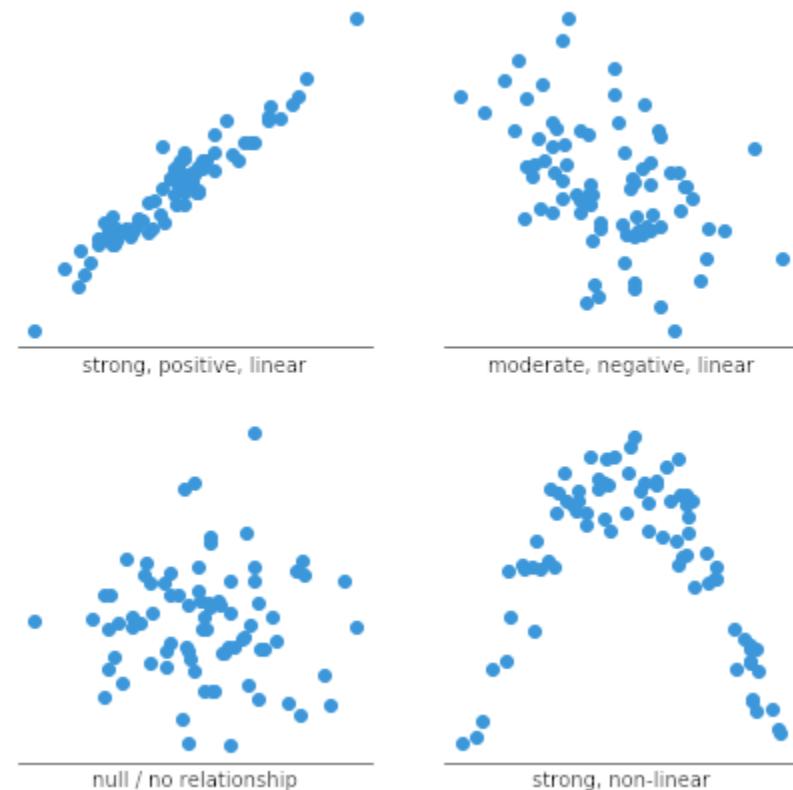
Line Plot

- A **line plot** is a **graph** that shows the frequency of data occurring along a number **line (Usually a timeline)**.
- Line graphs are used to track changes **over short and long periods of time**.
- Line graphs can also be used to compare changes over the same period of time for **more than one group**.
- It's very easy to interpret and use.



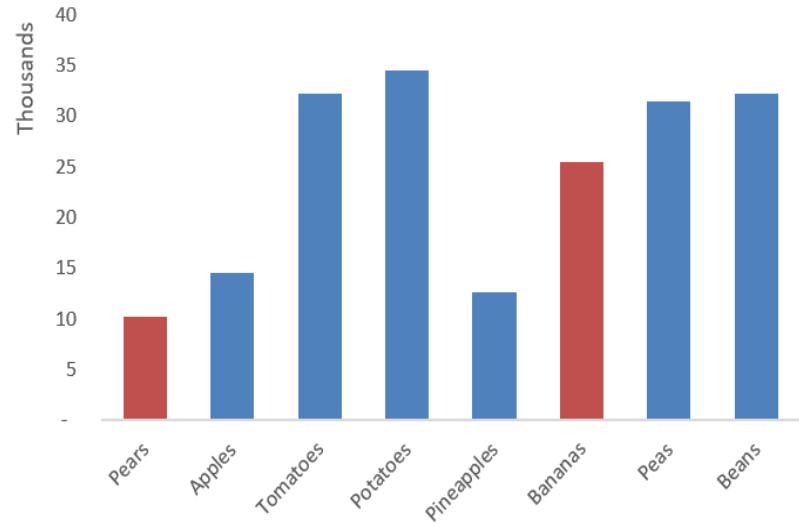
Scatter Plot

- Scatter Plot is a type of plot or mathematical diagram using **Cartesian coordinates** to display values for typically **two variables** for a set of data.
- Scatter plots are used to plot data points on a horizontal and a vertical axis in the attempt to show **how much one variable is affected by another.** (**Correlation test**)
- There are many types of correlation relationships that scatter plots can reveal.



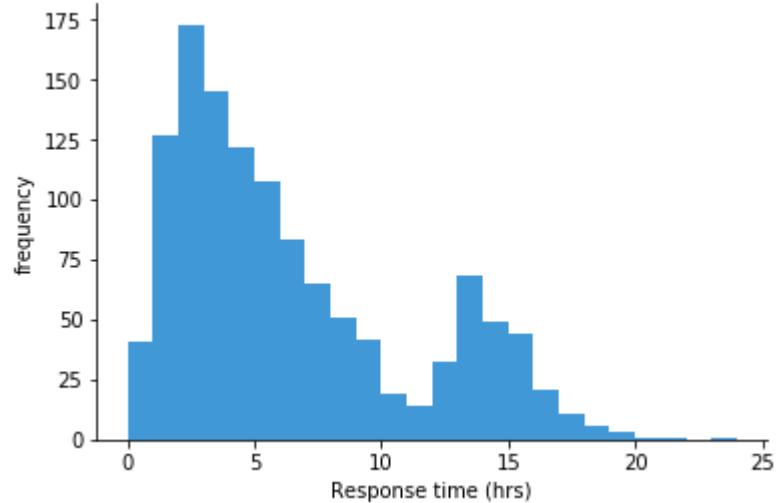
Bar Chart

- A bar chart or bar graph is a chart or graph that presents **categorical data** with rectangular bars.
- Bar graphs are used to compare things between different groups or to track changes over time.
- When trying to measure change over time, **bar graphs** are best when the changes are **larger**. But line plots are more suitable for visualizing changes over time.
- It can be graphed horizontally or vertically.

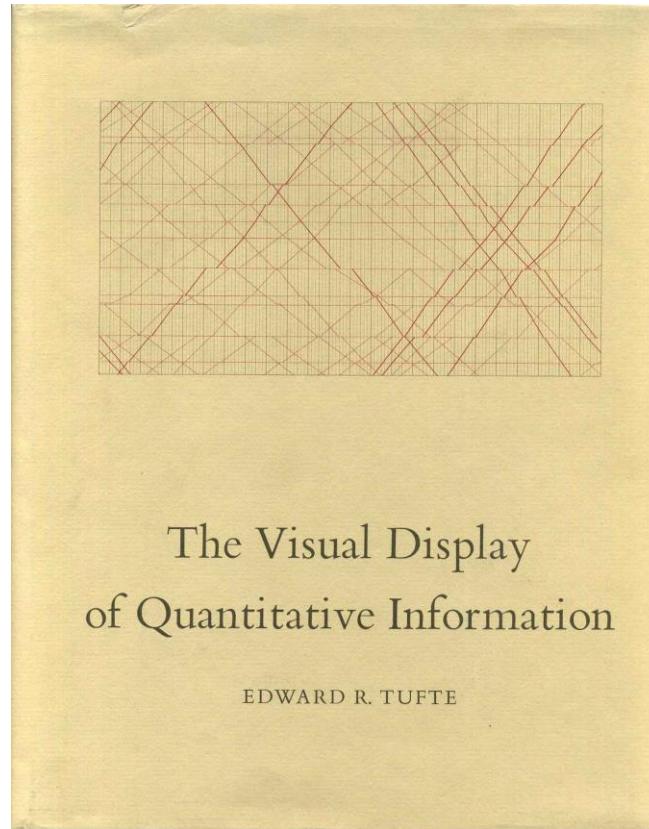


Histogram

- Histograms are the most frequently used chart in frequency distribution.
- Frequency distribution shows how often each different value in a set of data occurs.
- Taller bars show that more data falls in that range.
- A histogram displays the shape and spread of continuous sample data.



Further Reading



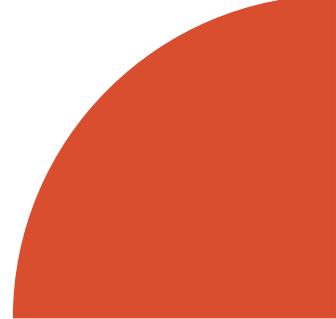
Real-World EDA

CONCLUSION

To summarize our lecture we can say:

1. Ranges and IQRs aren't the best for reporting variability.
2. Variance is less used than Standard Deviation.
3. Summary statistics is a must when working with data.
4. Measures of shape help us determine what transformation and operation should we apply on the data.
5. Without visualization we are **blind** warriors with swords in a battlefield.
6. Every Visual has its power. Choosing when to use it and with type of data activates that power.

Quiz



THANKS

Any Questions? ...

Head to google meet !



Introduction to data visualization with python

“SHOW, DON’T TELL”



Developer Student Clubs
Al-Azhar University

AGENDA

01

Benefits of Data
Visualization

02

Data Type

03

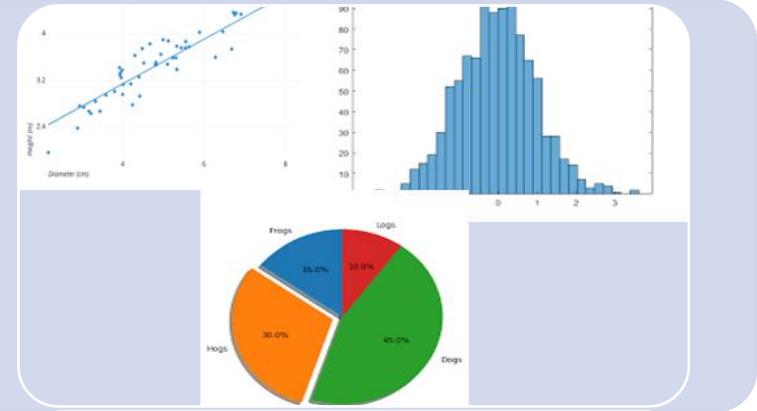
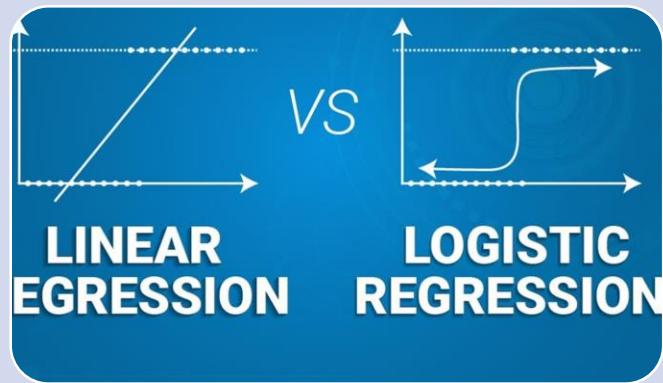
Design of Data
visualization

04

Python library for
data visualization

How to get insights of data

	Scores	Column
2	82	
3	93	Mean 81.21428571
4	91	Standard Error 4.045318243
5	69	Median 85
6	96	Mode 93
7	61	Standard Deviation 15.13619489
8	88	Sample Variance 229.1043956
9	58	Kurtosis -1.426053506
10	59	Skewness -0.402108004
11	100	Range 42
12	93	Minimum 58
13	71	Maximum 100
14	78	Sum 1137
15	98	Count 14



Calculating
summary
statistics

Running
models

Drawing
plots

	1		2		3		4	
	X	Y	X	Y	X	Y	X	Y
	10.0	8.04	10.0	9.14	10.0	7.46	8.0	6.58
	8.0	6.95	8.0	8.14	8.0	6.77	8.0	5.76
	13.0	7.58	13.0	8.74	13.0	12.74	8.0	7.71
	9.0	8.81	9.0	8.77	9.0	7.11	8.0	8.84
	11.0	8.33	11.0	9.26	11.0	7.81	8.0	8.47
	14.0	9.96	14.0	8.10	14.0	8.84	8.0	7.04
	6.0	7.24	6.0	6.13	6.0	6.08	8.0	5.25
	4.0	4.26	4.0	3.10	4.0	5.39	19.0	12.50
	12.0	10.84	12.0	9.13	12.0	8.15	8.0	5.56
	7.0	4.82	7.0	7.26	7.0	6.42	8.0	7.91
	5.0	5.68	5.0	4.74	5.0	5.73	8.0	6.89
Rectangular Snip								
Mean	9.0	7.5	9.0	7.5	9.0	7.5	9.0	7.5
Variance	10.0	3.75	10.0	3.75	10.0	3.75	10.0	3.75
Correlation	0.816		0.816		0.816		0.816	

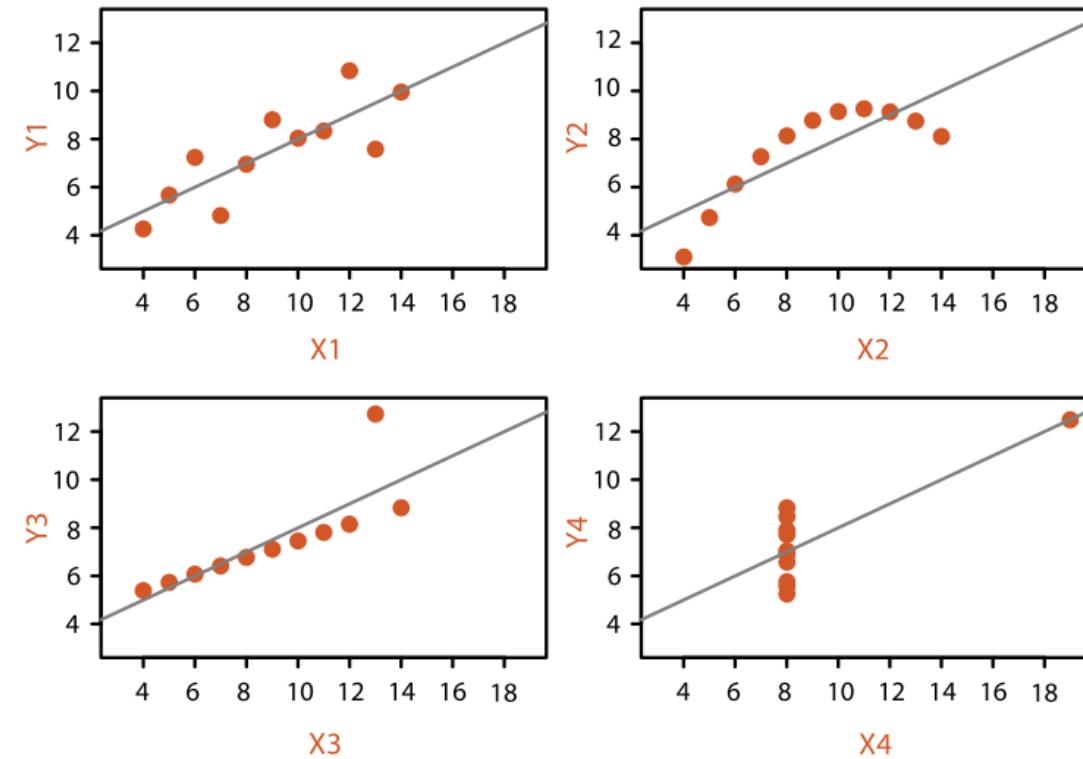


Figure 1.3. Anscombe's Quartet is four datasets with identical simple statistical properties: mean, variance, correlation, and linear regression line. However visual inspection immediately shows how their structures are quite different. After [Anscombe 73, Figures 1–4].

Go

DATA TYPE

Numerical Data

Categorical Data

Data Type

Numerical
Data

Categorical
Data

Continuous
data

Discrete
data

Categorical
Ordinal

Categorical
Nominal

Numerical Data

A. **Continuous** data

- can be split into smaller and smaller units, and still a smaller unit exists.
- could take on any value within an interval, many possible values.
 - Length
 - Weight
 - Temperature
 - Age

B. **Discrete** data

- countable value, finite number of values.

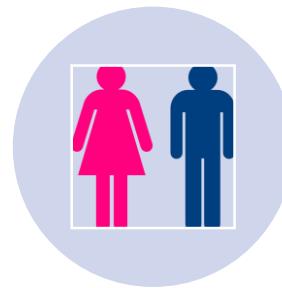


Categorical Data

Classifies items into different groups



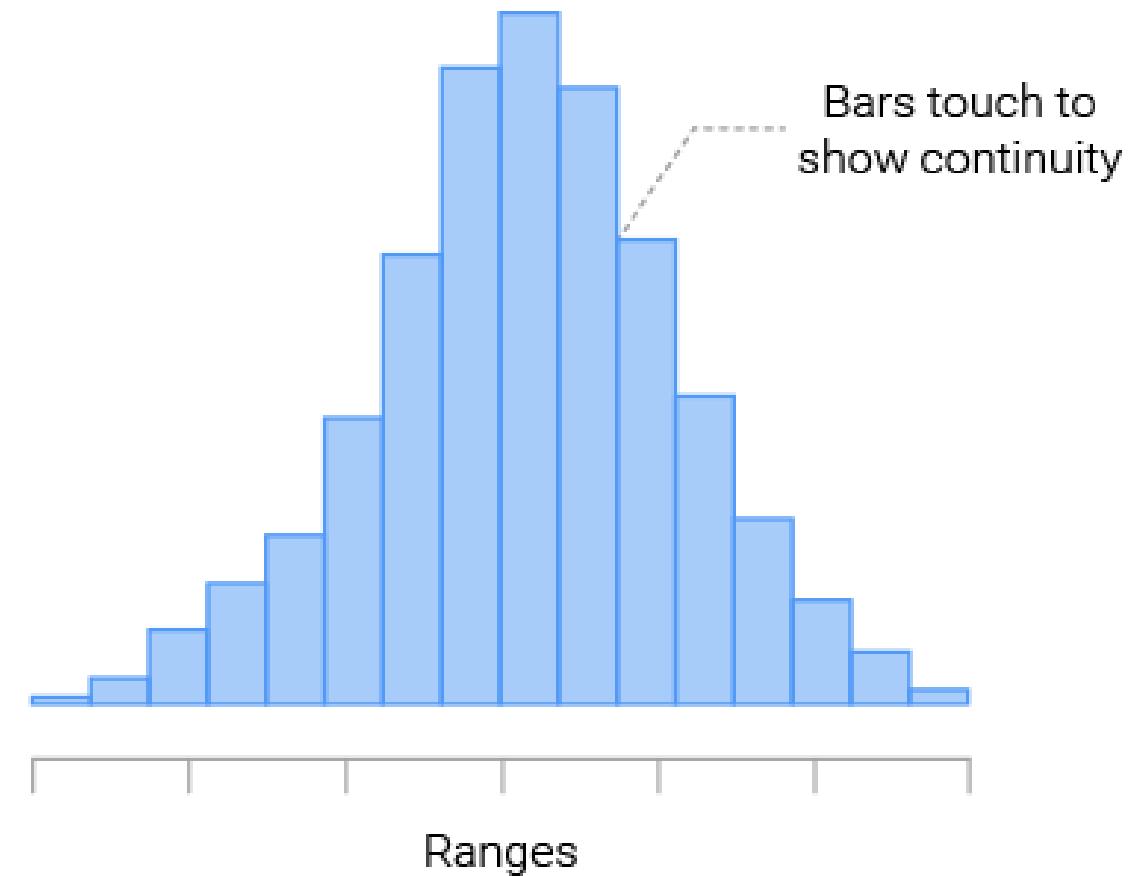
Ordinal
groups have an order or
ranking



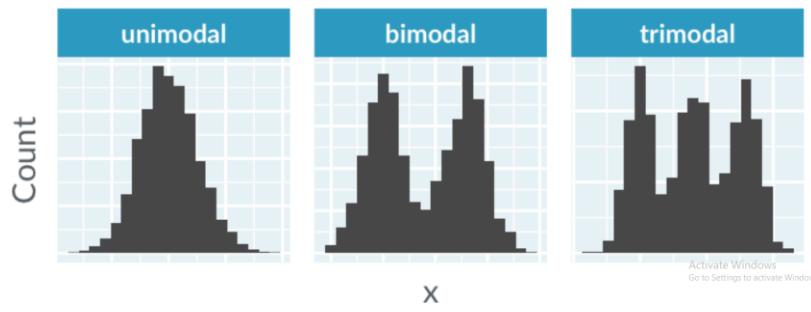
Nominal
groups are merely
names, no ranking.



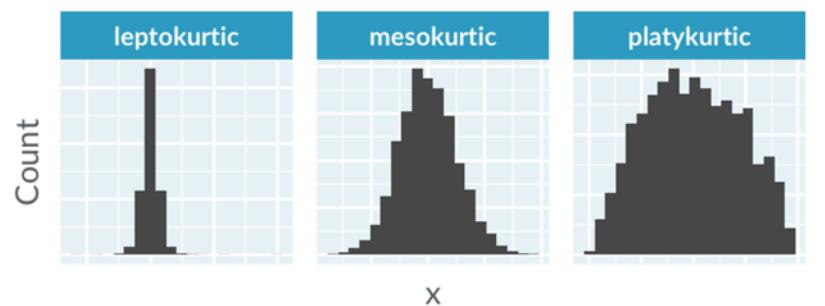
Histogram



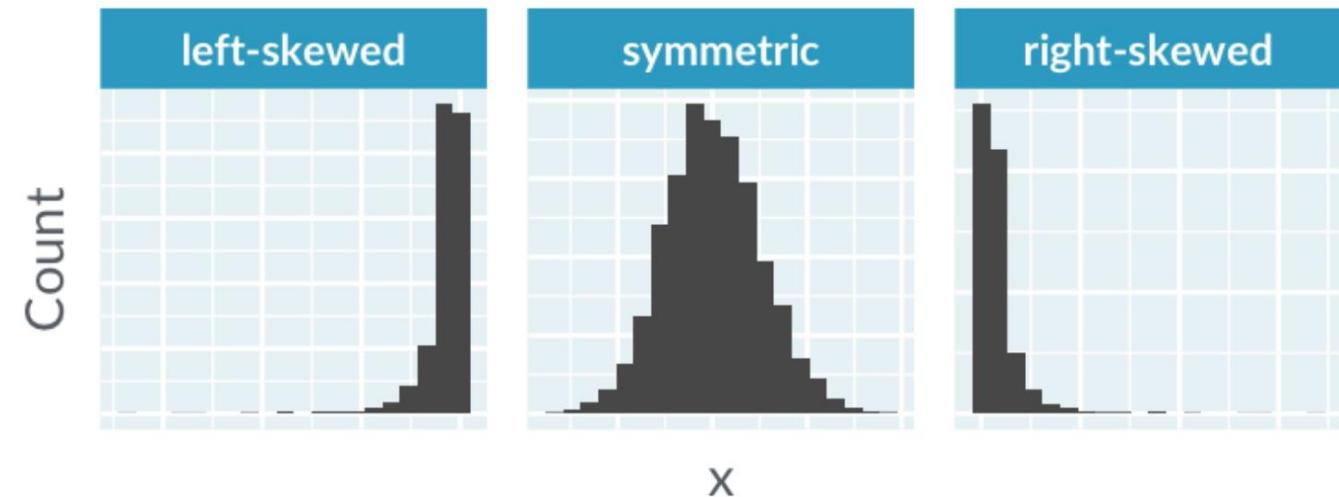
Modality



Kurtosis

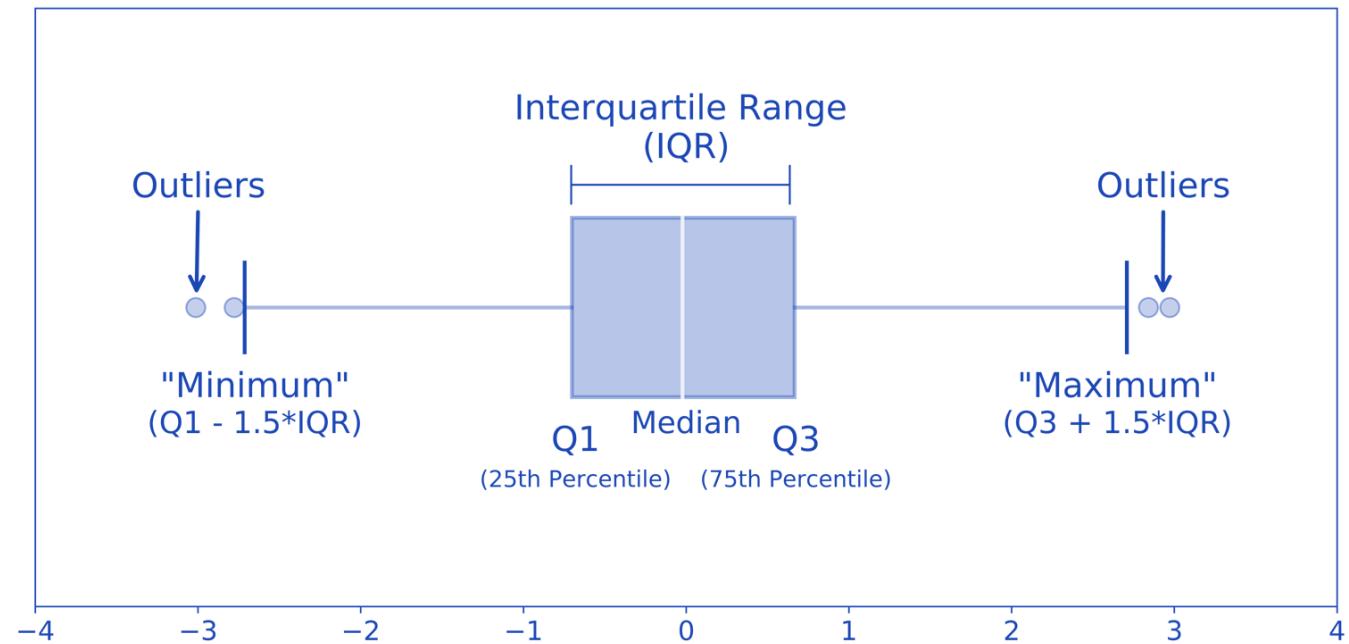


Skewness



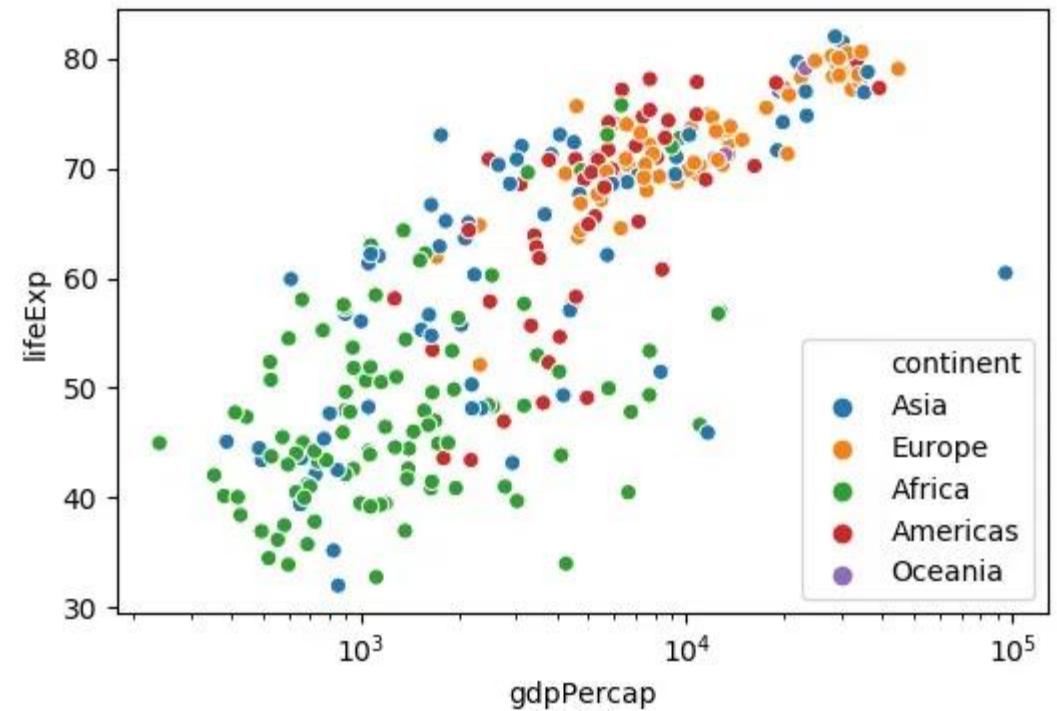
Box plot

- When you want to compare the distributions of the continuous variable for each category.



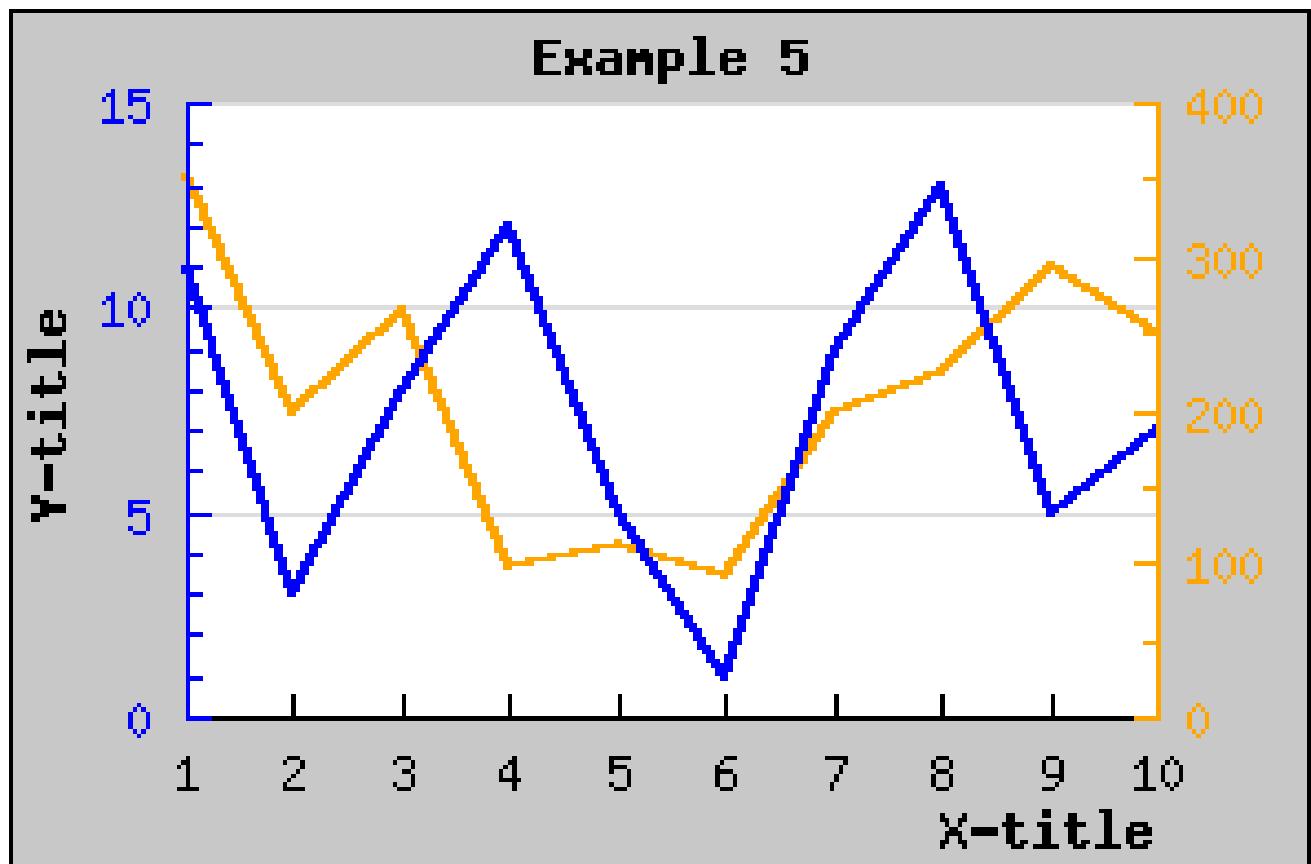
Scatter plots

- 1. You have two continuous variables.
- 2. relationship between the two variables.



Line plots

- 1. You have two continuous variables.
- 2. You want to answer questions about their relationship.
- Usually, but not always, the x-axis is dates or times.



Bar plots

- 1. You have a categorical variable.

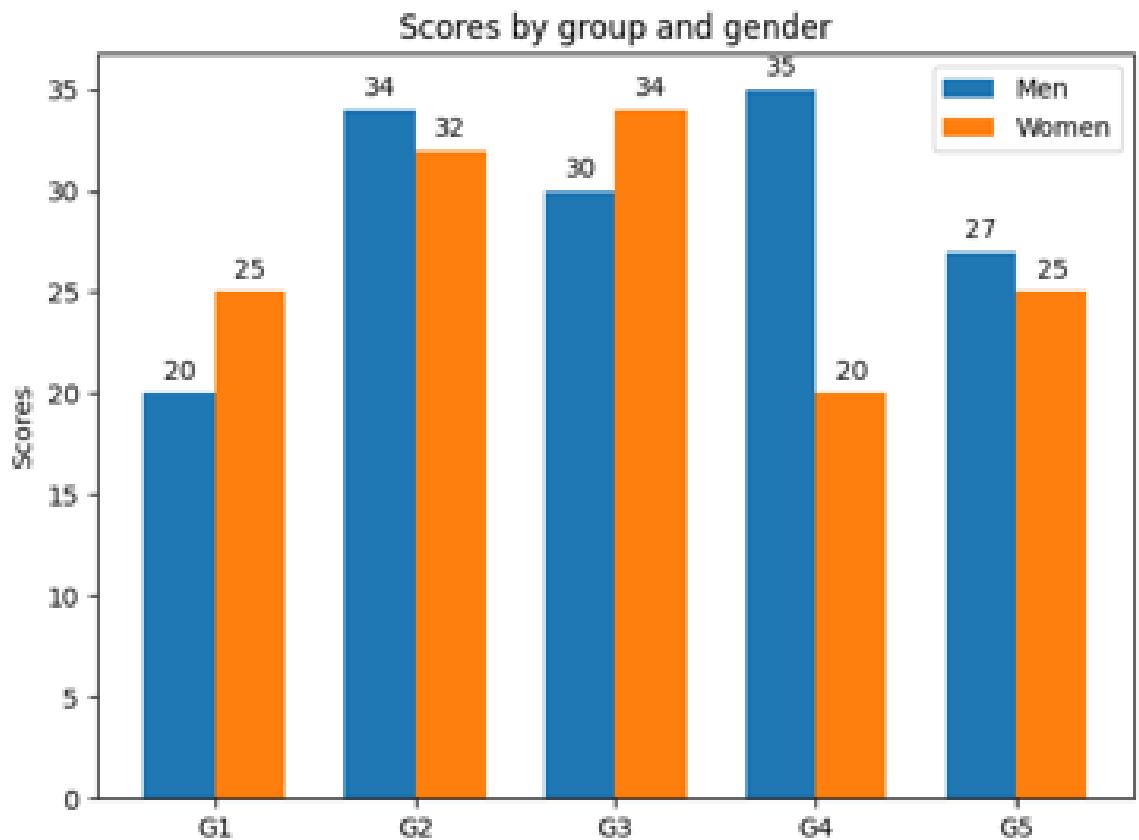
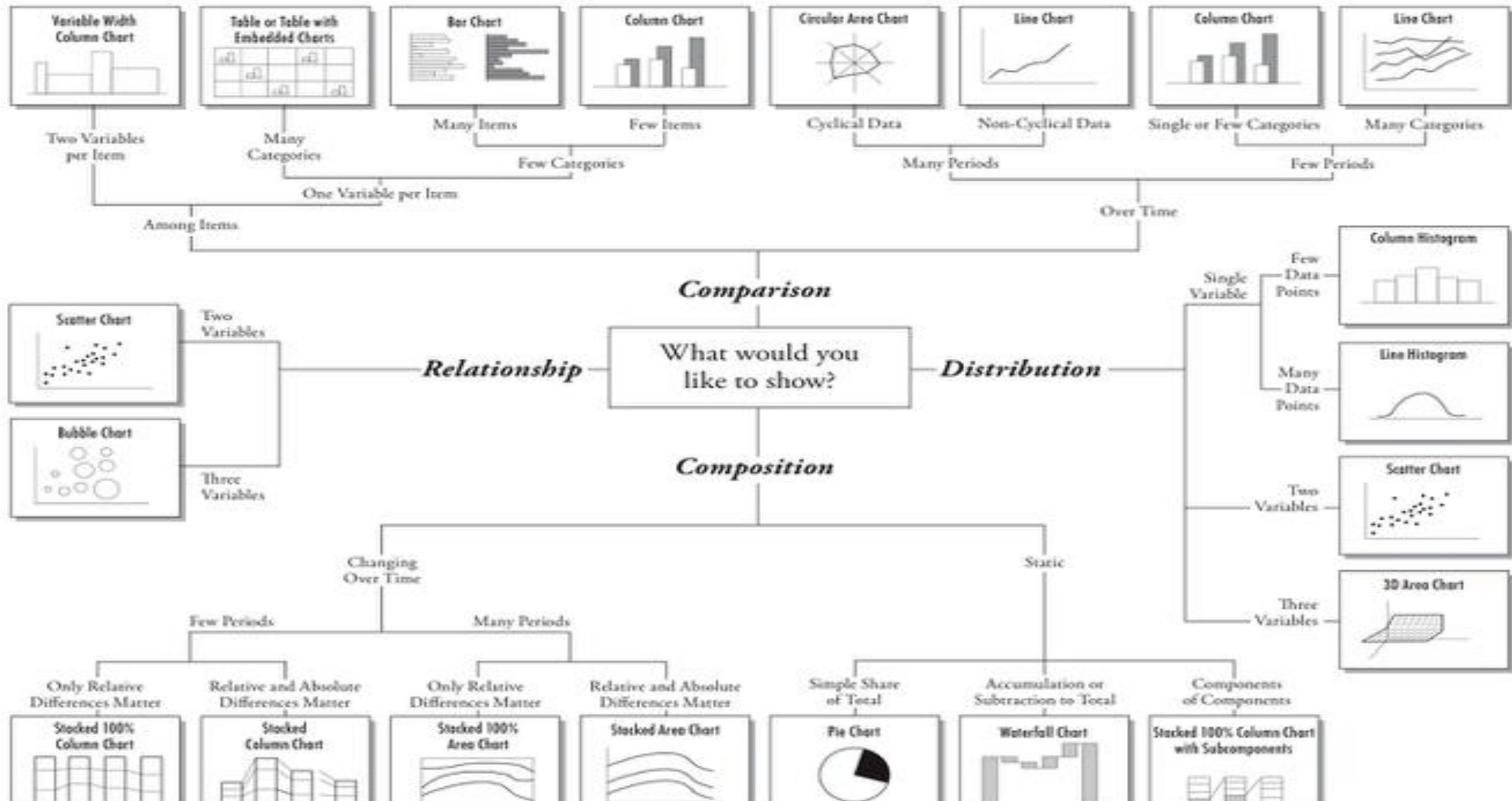


Chart Suggestions—A Thought-Starter

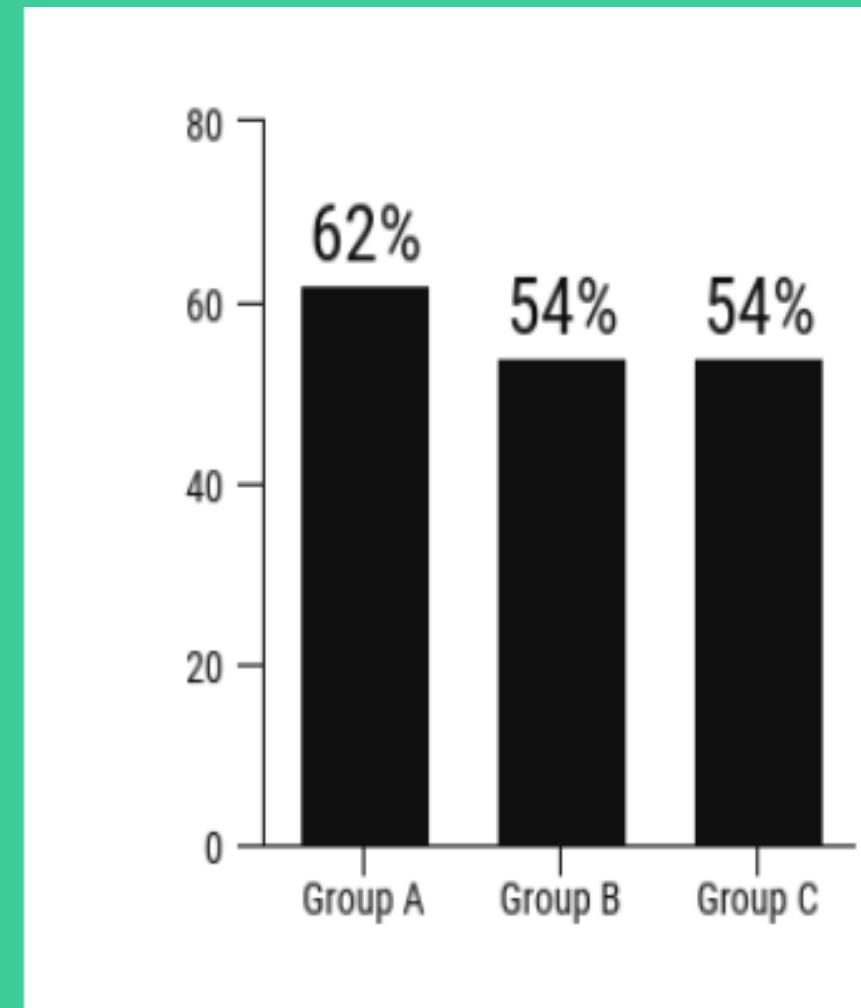
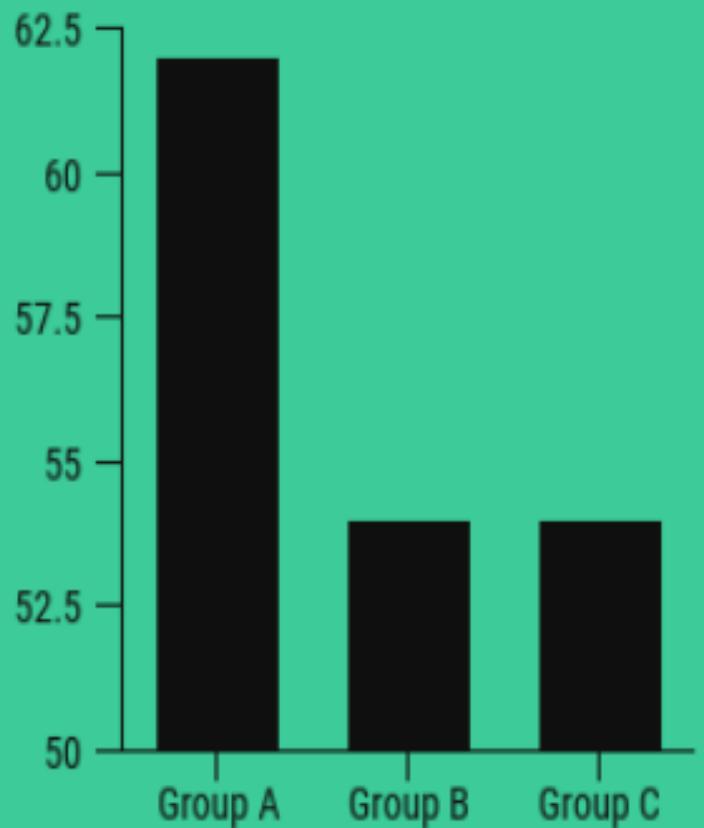


Design of visualization

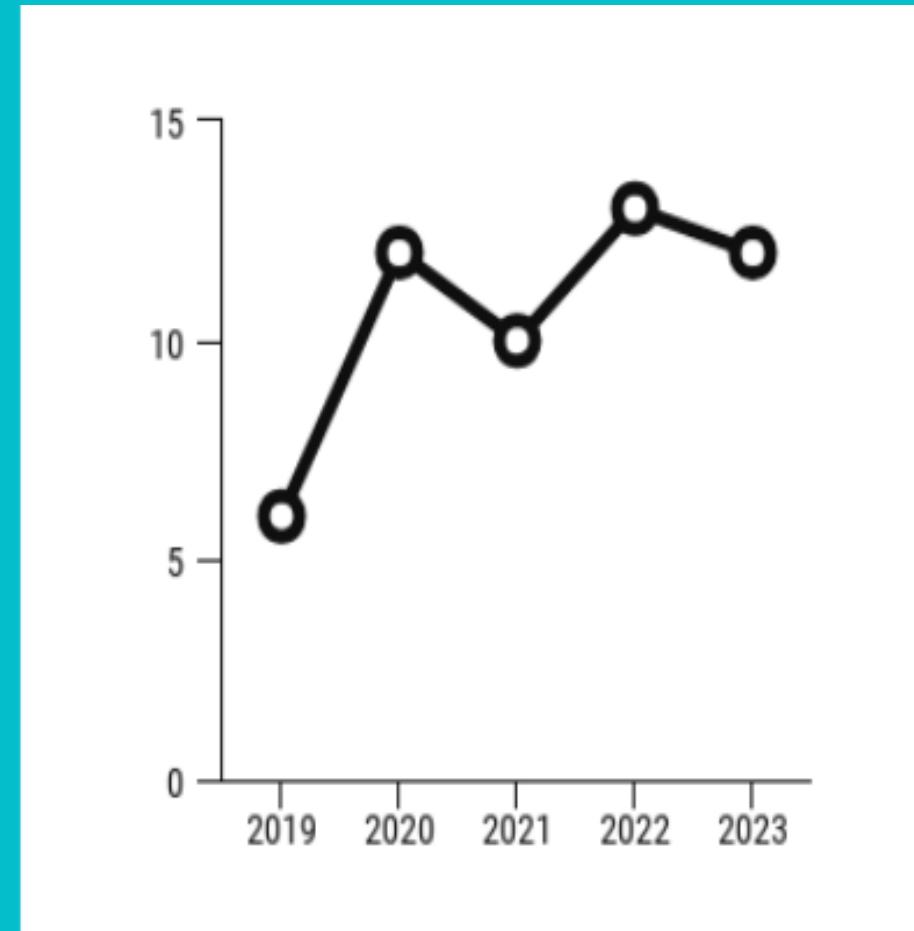
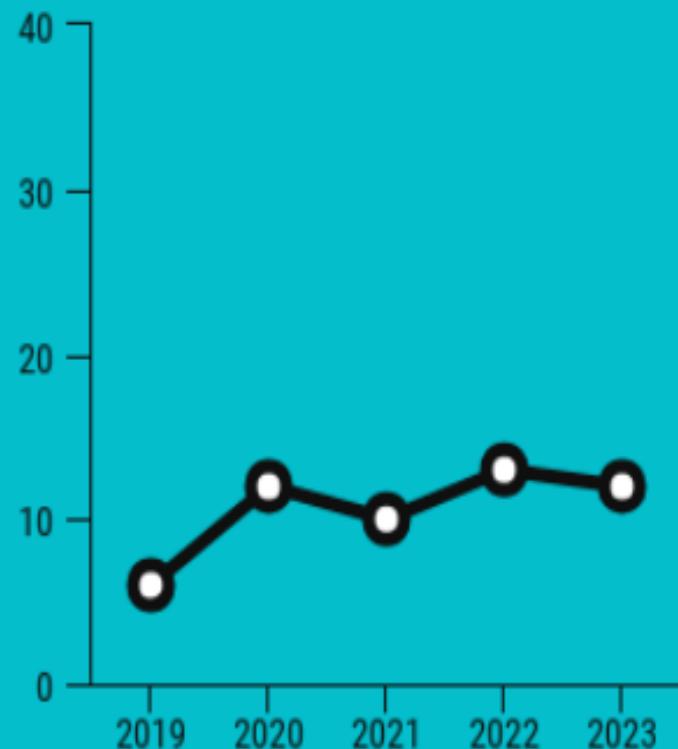


Ommitting the baseline

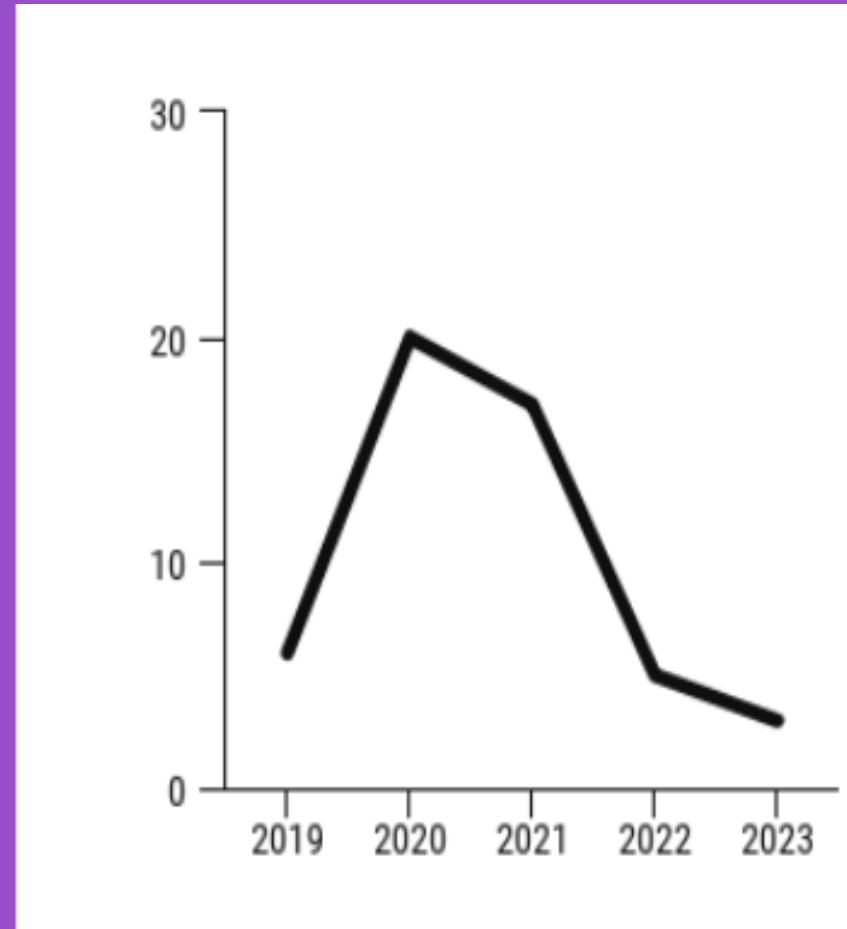
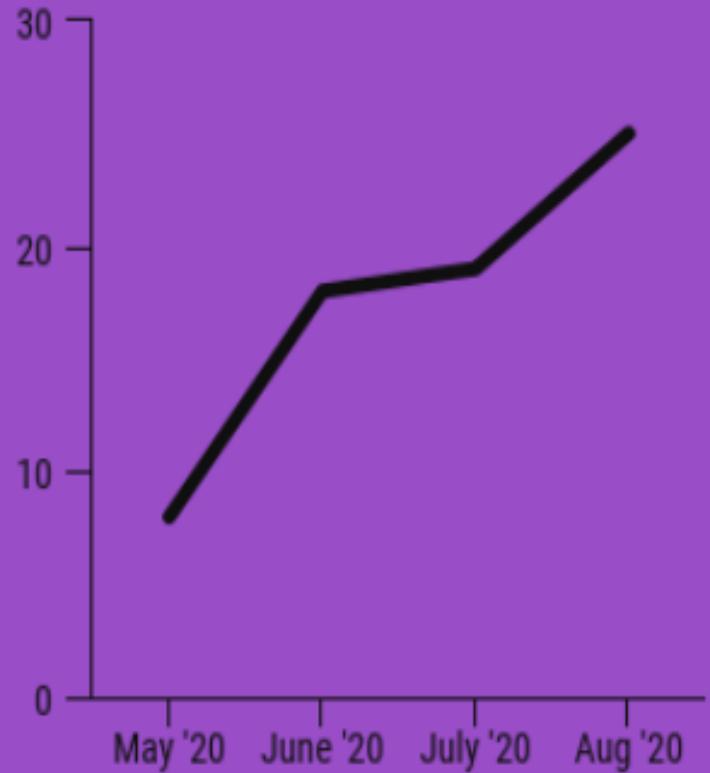
- Base line 0
- Truncated graph



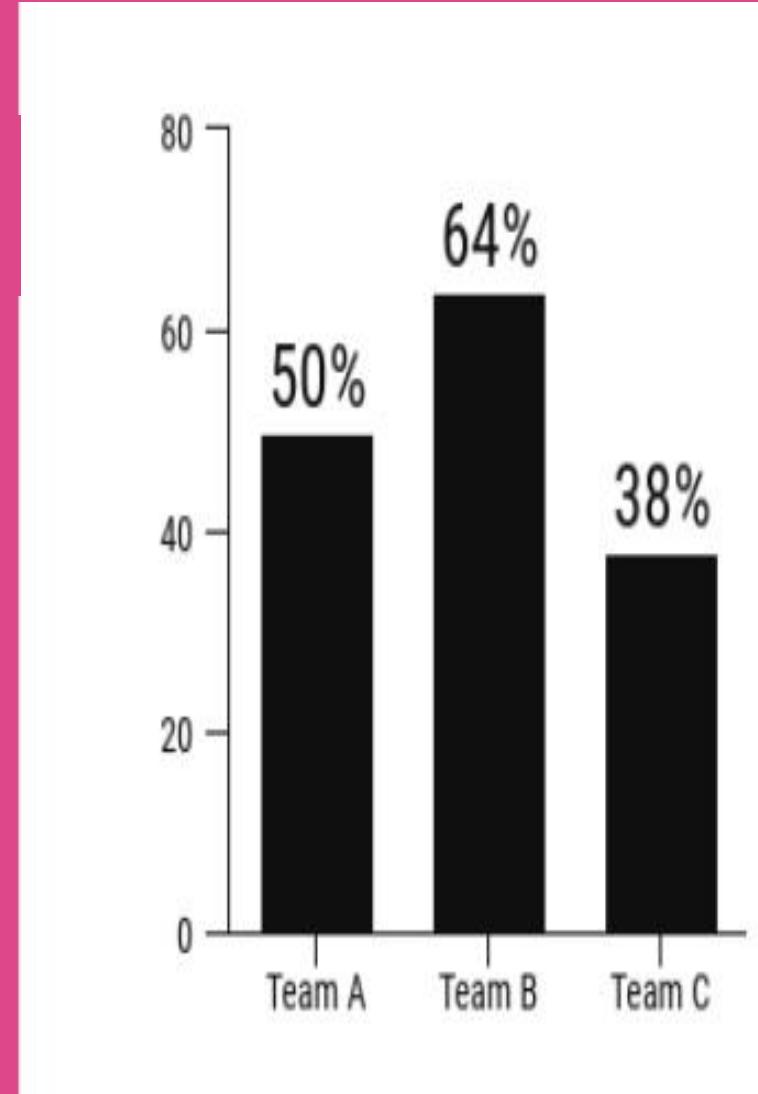
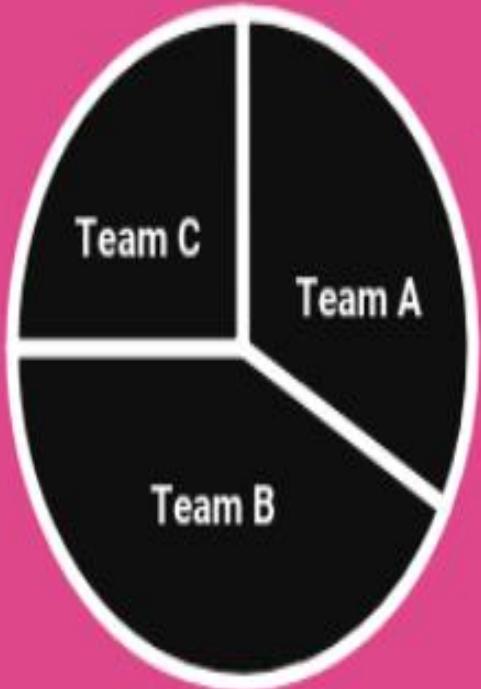
Manipulating the y-access



Cherry picking data

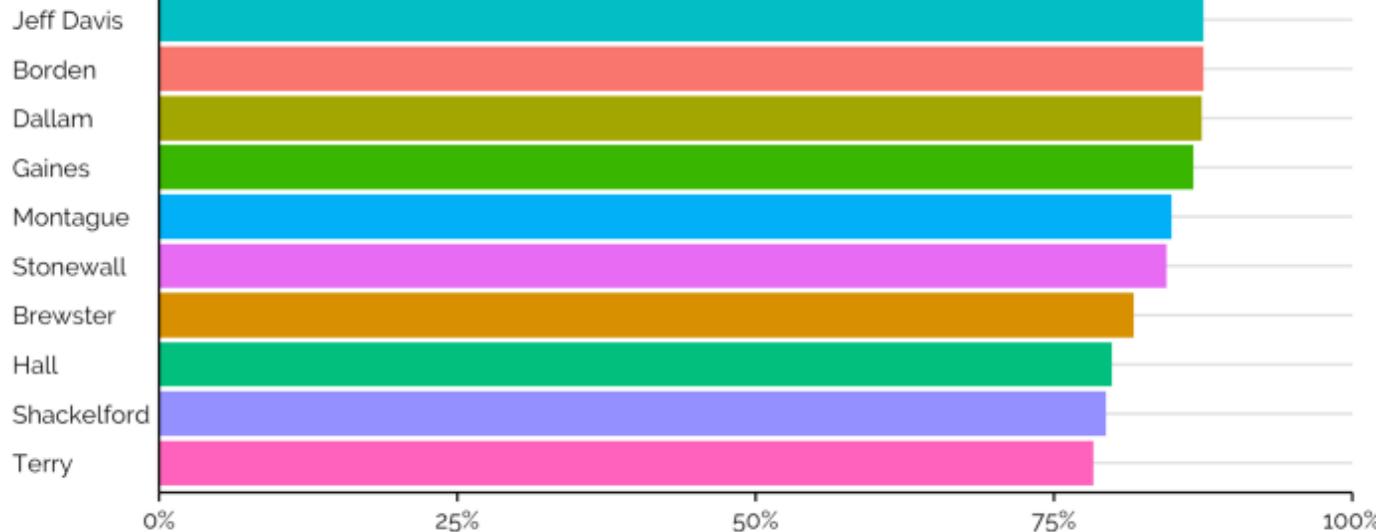


Using the wrong graph



The Least Vaccinated Counties

Terry County has the lowest Kindergarten vaccination rate in Texas



color

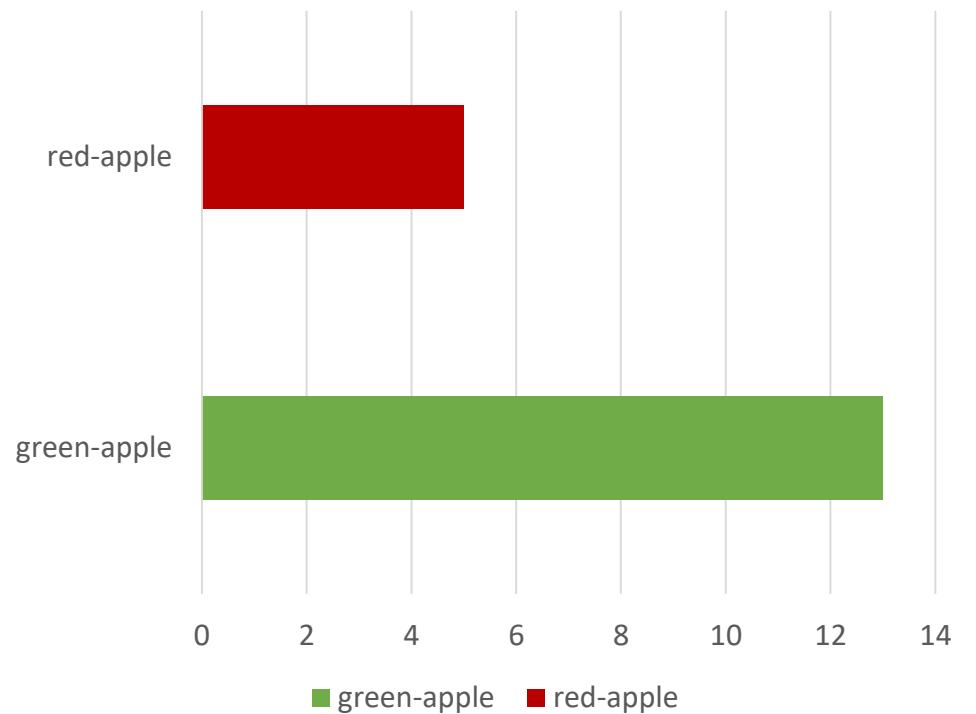
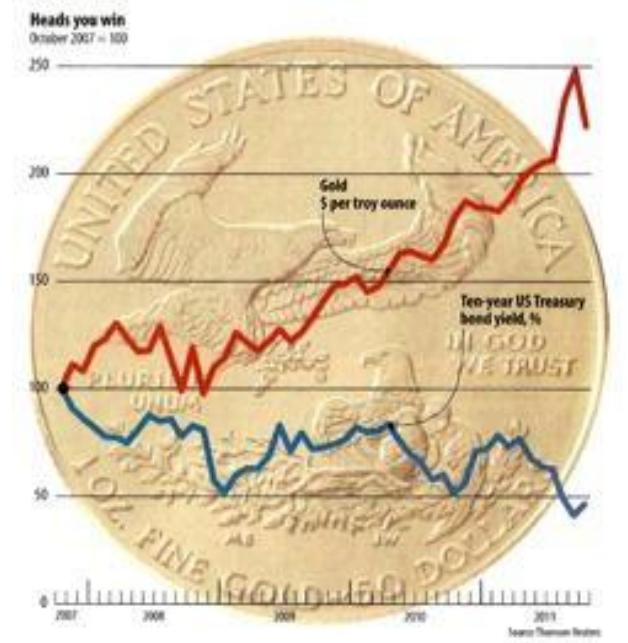
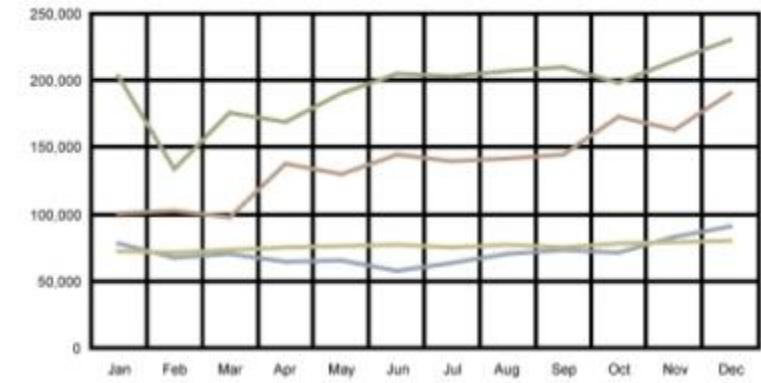
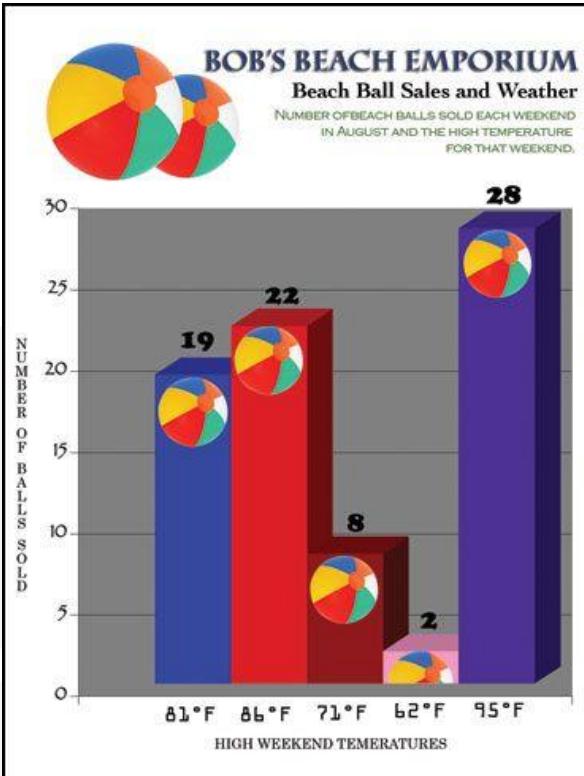
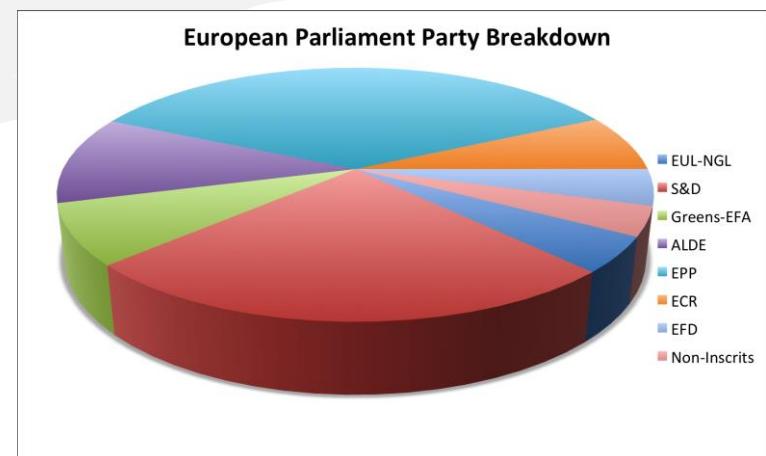


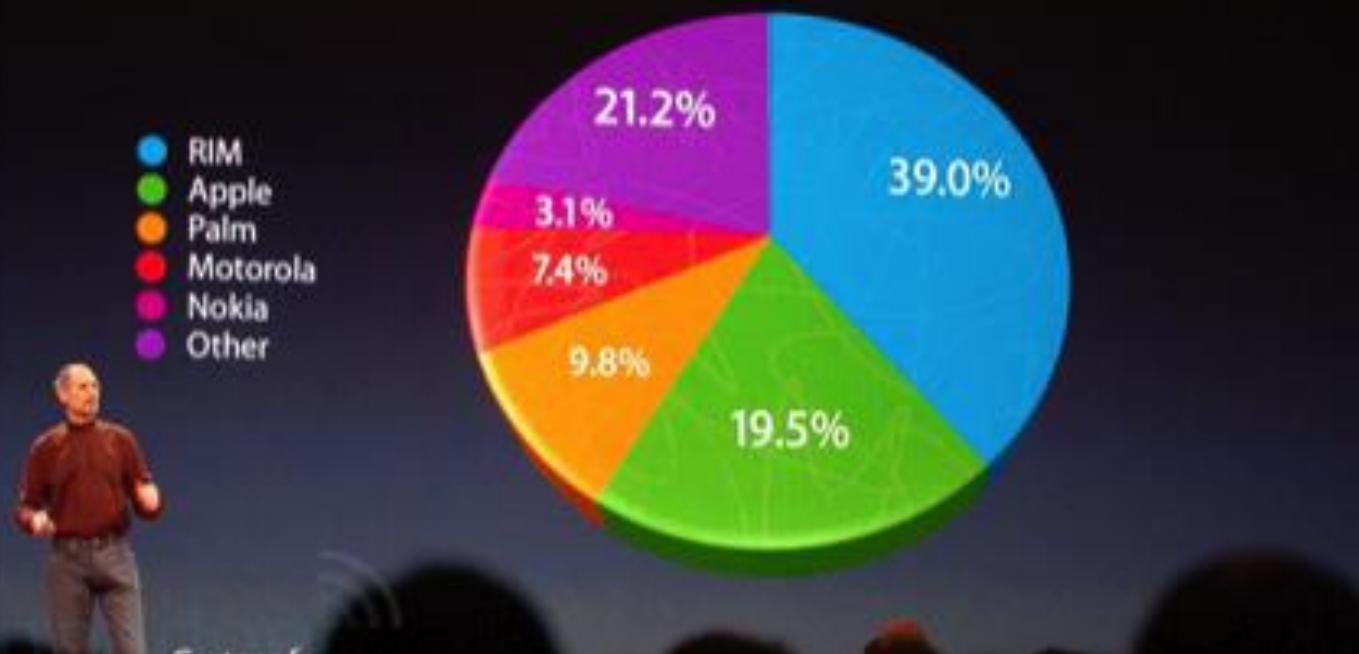
Chart junk

[More.](#)

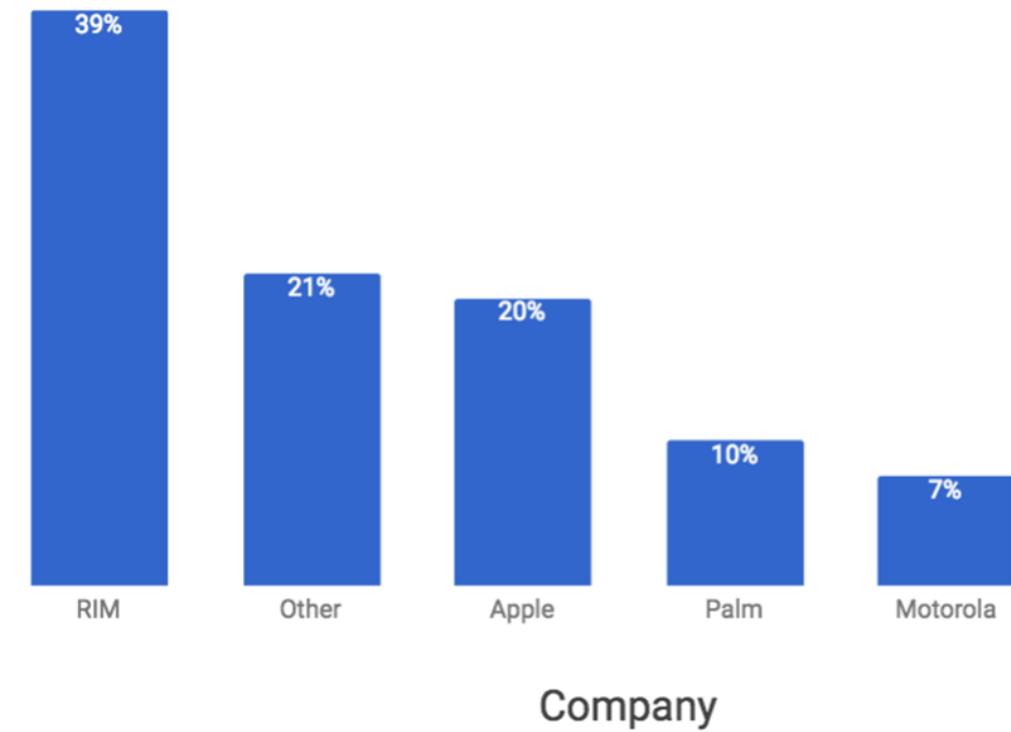


The chart-junk of Steve Jobs [more.](#)

U.S. SmartPhone Marketshare



Percentage of the Phone Market

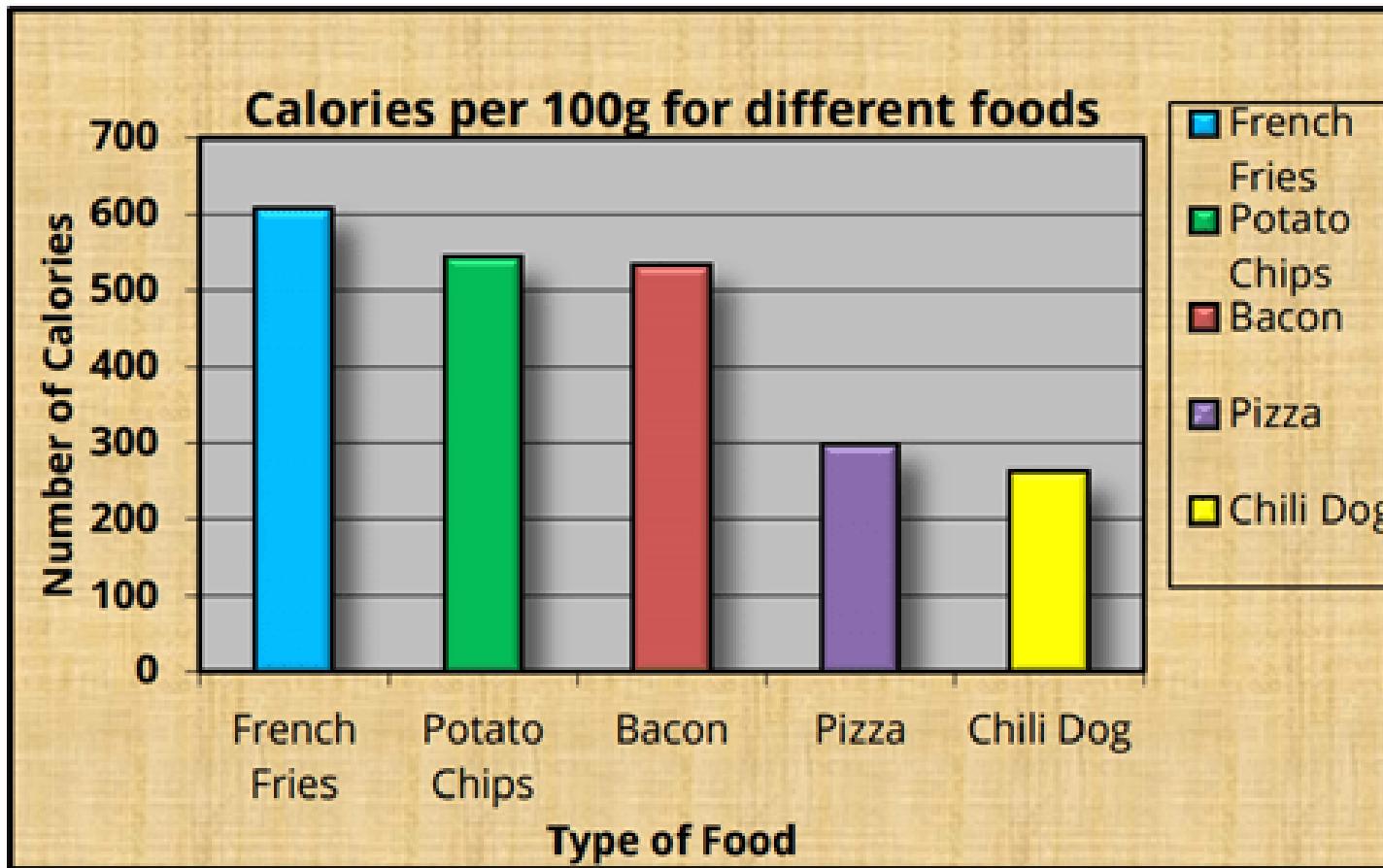


1,000

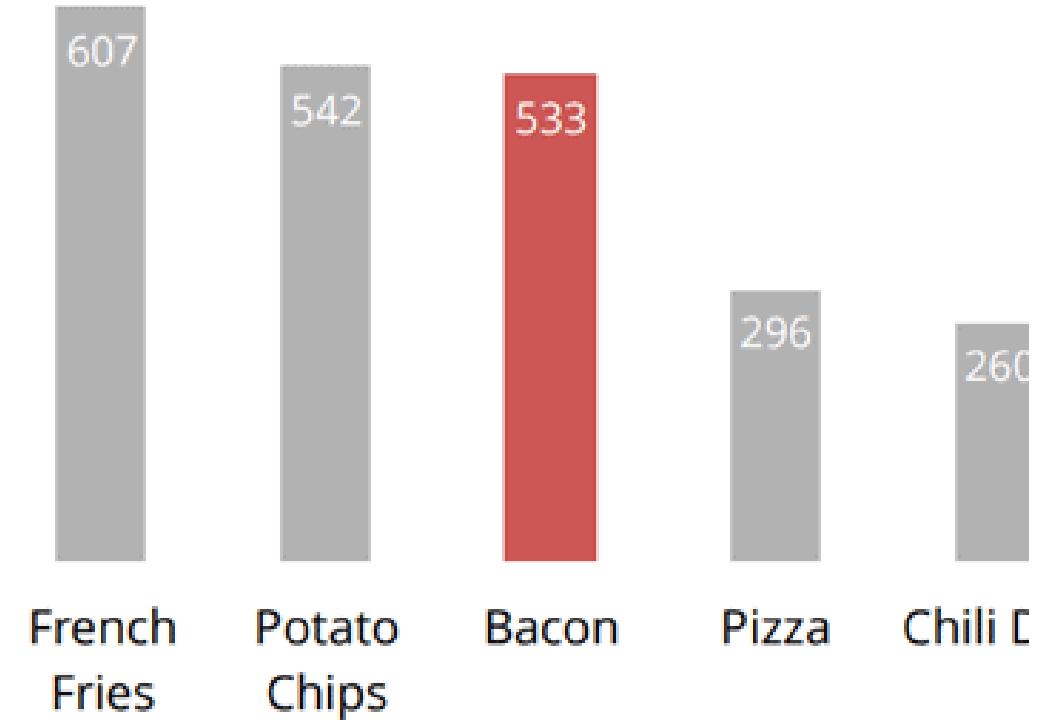
Data-ink ratio



Example of ink-ratio



Calories per 100g



Lie factor

$$\frac{\frac{5.3 - 0.6}{0.6}}{\frac{27.5 - 18}{18}} = 14.8$$

This results in the following formula:

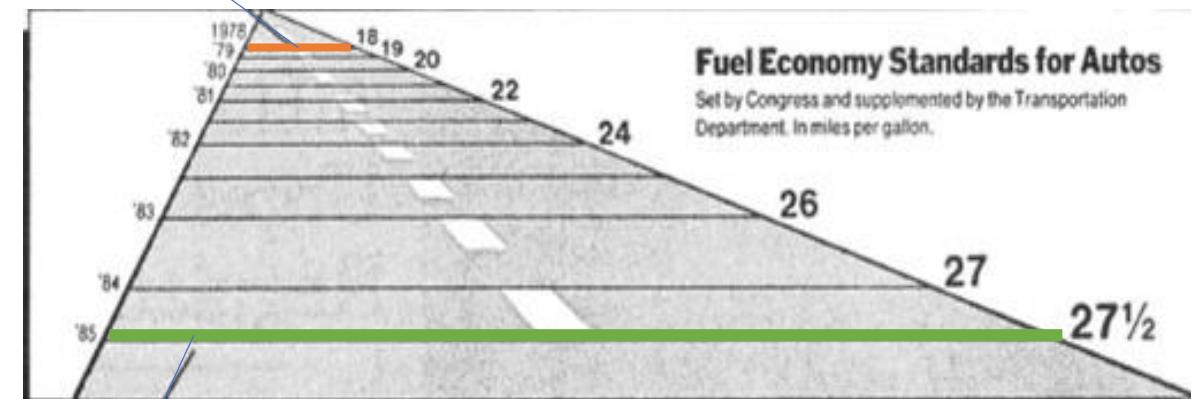
$$\text{Lie Factor} = \frac{\text{size of effect shown in graphic}}{\text{size of effect in data}}$$

where

$$\text{size of effect} = \frac{|\text{second value} - \text{first value}|}{\text{first value}}$$

1978: 18 mile/gallon

0.6 inches



1985: 27.5 mile/gallon

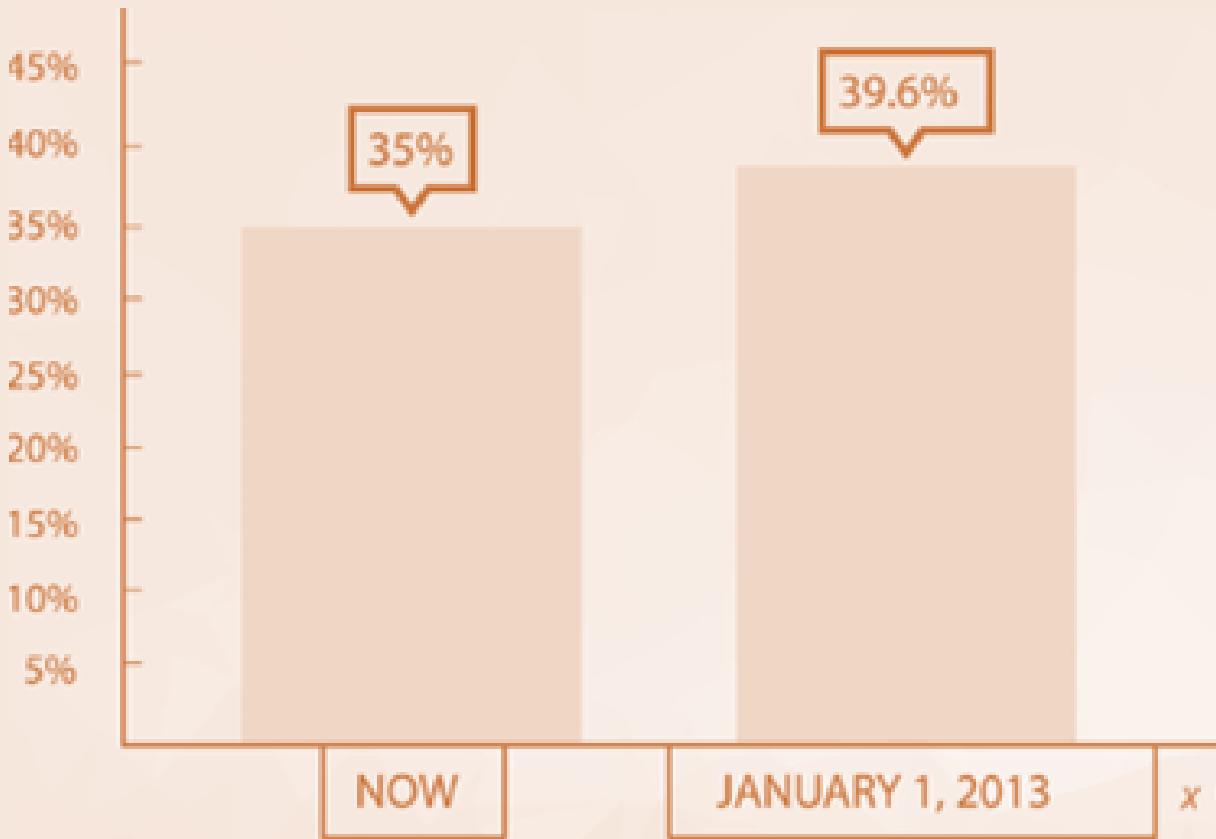
5.3 inches

- The standard required an increase in mileage from 18 to 27.5, an increase of 53%.

$$\frac{5.3 - 0.6}{0.6} * 100 = 53\%$$

The magnitude of increase shown in the graph is 783%,

$$\frac{27.5 - 18}{18} * 100 = 783\%$$

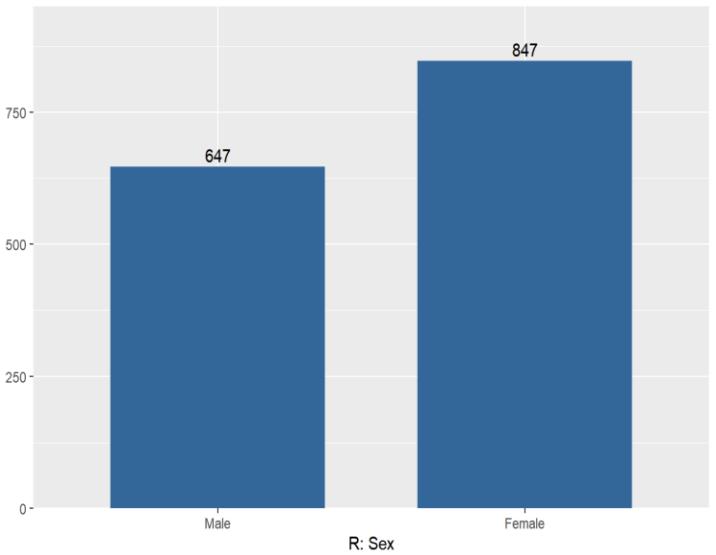




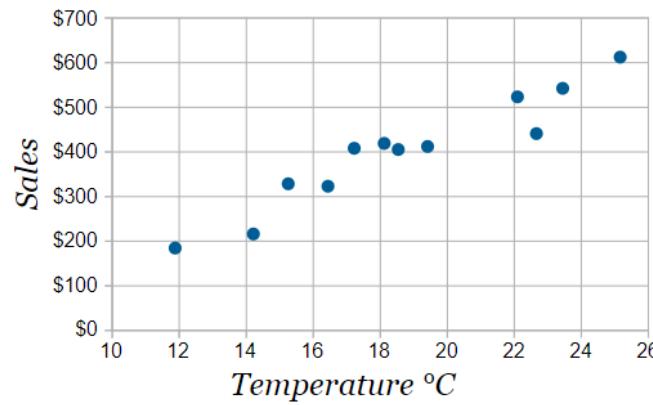
Type of data analysis

Univariate

Gender Distribution

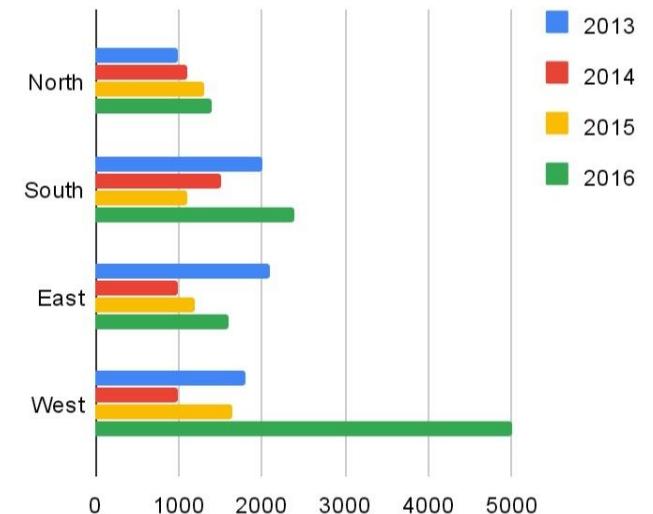


Bivariate



Multivariate

ABC SALES RECORD



Python library for data visualization

Matplotlib

Seaborn

Plotly

ggplot



matplotlib



most popular

2D plotting library

With this library, with just a few lines of code, one can generate plots, bar charts, histograms, power spectra, stemplots, scatterplots, error charts, pie charts and many other types.

<https://matplotlib.org/tutorials/introductory/pyplot.html#sphx-glr-tutorials-introductory-pyplot-py>





seaborn

Seaborn is a library based on Matplotlib





plotly

Plotly is a web-based toolkit to form data visualizations.

Plotly can also be accessed from a Python Notebook and has a great API. With unique functionalities such as contour plots, dendograms, and 3D charts,



It's coding time



The Problem

In a recent survey, researchers have pointed out that 12 million adult patients in the US are misdiagnosed each year and 10% of deaths occur due to diagnostic errors. So, the problem and the question that we ask is that How can we improve the diagnostic Accuracy?

The Purpose of The Project

Save people alive and avoid the diagnostic errors.

Performance's indicators of the project

Decreasing of the percentage of the death or the misdiagnosis issues in specific region.

Analytic approach

Predictive analytics (using data science analytic approach technologies and machine learning algorithms to improve diagnostic accuracy) will be used in this issue to predict the outcomes of the diagnosis using the data that is collected before in the Data Collection Plan.

Data Collection Plan

The data can be collected by primary data collection like surveys and secondary data collection like Data Hub, Kaggle and GitHub. These Data will be historical data including patient data, clinical notes, symptoms, habits, diseases, genome structure, etc.

Python For Data Science Cheat Sheet

Seaborn

Learn Data Science interactively at www.DataCamp.com



Statistical Data Visualization With Seaborn

The Python visualization library **Seaborn** is based on `matplotlib` and provides a high-level interface for drawing attractive statistical graphics.

Make use of the following aliases to import the libraries:

```
>>> import matplotlib.pyplot as plt  
>>> import seaborn as sns
```

The basic steps to creating plots with Seaborn are:

1. Prepare some data
2. Control figure aesthetics
3. Plot with Seaborn
4. Further customize your plot

```
>>> import matplotlib.pyplot as plt  
>>> import seaborn as sns  
>>> tips = sns.load_dataset("tips")  
>>> sns.set_style("whitegrid")  
>>> g = sns.lmplot(x="tip",  
y="total_bill",  
data=tips,  
aspect=2)  
>>> g.set_axis_labels("Tip", "Total bill(USD)")  
set(xlim=(0,10), ylim=(0,100))  
>>> plt.title("title")  
>>> plt.show(g)
```

Step 1
Step 2
Step 3
Step 4
Step 5

1) Data

Also see [Lists, NumPy & Pandas](#)

```
>>> import pandas as pd  
>>> import numpy as np  
>>> uniform_data = np.random.rand(10, 12)  
>>> data = pd.DataFrame({'x':np.arange(1,101),  
y':np.random.normal(0,4,100)})
```

Seaborn also offers built-in data sets:

```
>>> titanic = sns.load_dataset("titanic")  
>>> iris = sns.load_dataset("iris")
```

2) Figure Aesthetics

Seaborn styles

```
>>> sns.set()  
>>> sns.set_style("whitegrid")  
>>> sns.set_style("ticks",  
{"xtick.major.size":8,  
"ytick.major.size":8})  
>>> sns.axes_style("whitegrid")
```

(Re)set the seaborn default
Set the matplotlib parameters
Set the matplotlib parameters
Return a dict of params or use with
with to temporarily set the style

Context Functions

```
>>> sns.set_context("talk")  
>>> sns.set_context("notebook",  
font_scale=1.5,  
rc={"lines.linewidth":2.5})
```

Color Palette

```
>>> sns.set_palette("husl",3)  
>>> sns.color_palette("husl")  
>>> flatui = ["#9b59b6","#3498db","#95a5a6","#e74c3c","#34495e","#2ecc71"]  
>>> sns.set_palette(flatui)
```

3) Plotting With Seaborn

Axis Grids

```
>>> g = sns.FacetGrid(titanic,  
col="survived",  
row="sex")  
>>> g.map(plt.hist,"age")  
>>> sns.factorplot(x="pclass",  
y="survived",  
hue="sex",  
data=titanic)  
>>> sns.lmplot(x="sepal_width",  
y="sepal_length",  
hue="species",  
data=iris)
```

Subplot grid for plotting conditional relationships

Draw a categorical plot onto a Facetgrid

Plot data and regression model fits across a FacetGrid

```
>>> h = sns.PairGrid(iris)  
>>> h.map(plt.scatter)  
>>> sns.pairplot(iris)  
>>> i = sns.JointGrid(x="x",  
y="y",  
data=data)  
>>> i.plot(sns.regplot,  
sns.distplot)  
>>> sns.jointplot("sepal_length",  
"sepal_width",  
data=iris,  
kind='kde')
```

Subplot grid for plotting pairwise relationships
Plot pairwise bivariate distributions
Grid for bivariate plot with marginal univariate plots

Plot bivariate distribution

Categorical Plots

Scatterplot
`>>> sns.stripplot(x="species",
y="petal_length",
data=iris)
>>> sns.swarmplot(x="species",
y="petal_length",
data=iris)`

Bar Chart

```
>>> sns.barplot(x="sex",  
y="survived",  
hue="class",  
data=titanic)
```

Count Plot

```
>>> sns.countplot(x="deck",  
data=titanic,  
palette="Greens_d")
```

Point Plot

```
>>> sns.pointplot(x="class",  
y="survived",  
hue="sex",  
data=titanic,  
palette={"male":"g",  
"female":"m"},  
markers=["^","o"],  
linestyles=["-","--"])
```

Boxplot

```
>>> sns.boxplot(x="alive",  
y="age",  
hue="adult_male",  
data=titanic)
```

Violinplot

```
>>> sns.violinplot(x="age",  
y="sex",  
hue="survived",  
data=titanic)
```

Scatterplot with one categorical variable

Categorical scatterplot with non-overlapping points

Show point estimates and confidence intervals with scatterplot glyphs

Show count of observations

Show point estimates and confidence intervals as rectangular bars

Boxplot

Boxplot with wide-form data

Violin plot

Regression Plots

```
>>> sns.regplot(x="sepal_width",  
y="sepal_length",  
data=iris,  
ax=ax)
```

Plot data and a linear regression model fit

Distribution Plots

```
>>> plot = sns.distplot(data.y,  
kde=False,  
color="b")
```

Plot univariate distribution

Matrix Plots

```
>>> sns.heatmap(uniform_data,vmin=0,vmax=1)
```

Heatmap

4) Further Customizations

Also see [Matplotlib](#)

Axisgrid Objects

```
>>> g.despine(left=True)  
>>> g.set_ylabels("Survived")  
>>> g.set_xticklabels(rotation=45)  
>>> g.set_axis_labels("Survived",  
"Sex")  
>>> h.set(xlim=(0,5),  
ylim=(0,5),  
xticks=[0,2.5,5],  
yticks=[0,2.5,5])
```

Remove left spine
Set the labels of the y-axis
Set the tick labels for x
Set the axis labels

Set the limit and ticks of the x-and y-axis

Plot

```
>>> plt.title("A Title")  
>>> plt.ylabel("Survived")  
>>> plt.xlabel("Sex")  
>>> plt.ylim(0,100)  
>>> plt.xlim(0,10)  
>>> plt.setp(ax,yticks=[0,5])  
>>> plt.tight_layout()
```

Add plot title
Adjust the label of the y-axis
Adjust the label of the x-axis
Adjust the limits of the y-axis
Adjust the limits of the x-axis
Adjust a plot property
Adjust subplot params

5) Show or Save Plot

Also see [Matplotlib](#)

```
>>> plt.show()  
>>> plt.savefig("foo.png")  
>>> plt.savefig("foo.png",  
transparent=True)
```

Show the plot
Save the plot as a figure
Save transparent figure

Close & Clear

```
>>> plt.cla()  
>>> plt.clf()  
>>> plt.close()
```

Clear an axis
Clear an entire figure
Close a window

