Do It Yourself Data Science Course

Hauke Bartsch, Dr. rer. nat.
University of Bergen, Norway

For PhD students / early-stage researchers, who see a need for DIY data science in their work/research.

100% learning with code.

No certificate.

We might have math.

Do It Yourself - Data Science Course

Schedule

- Mo Class 1

Objective: Introduce participants to the fundamentals of data science, and study design by creating a data collection project, entering and exporting data. Python programming reminders and how to setup Python for data manipulation and analysis.

- Mo Class 2

Objective: Familiarize participants with the essential *panda's* table for structuring data. Table formats and examples on data import, re-coding and data exports. Examples on how to create a Table 1.

- Tue Class 3

Objective: Libraries for plotting data. *Matplotlib* and *plotly* examples including the display of hierarchical data and volumetric data. Plotting using *plotnine*. Merging two tables based on index variables.

Bring your own data Q&A

- Tue Class 4

Objective: How to handle missing data and the use of multiple imputations using chained equations (MICE) to fill in missing values. How to detect outliers.

Do we still need data science (classes) if we have ChatGPT?

Structured or unstructured data?

Is it true that most data is unstructured?

Isn’t structured data limiting as it can only be used for its intended purpose?

Unstructured data (preserve in *data lake*) + assumptions = structured data for a specific purpose (stored in database).

Where do I spend my effort, extracting information from unstructured data or collecting structured data?

first name, last name
date of birth
geo-location data as longitude/latitude
images
sound recordings
essay about how I felt this morning.
drop-down and radio buttons
free text fields as notes
actigraphy data

Experimental design for event-related studies

Events:

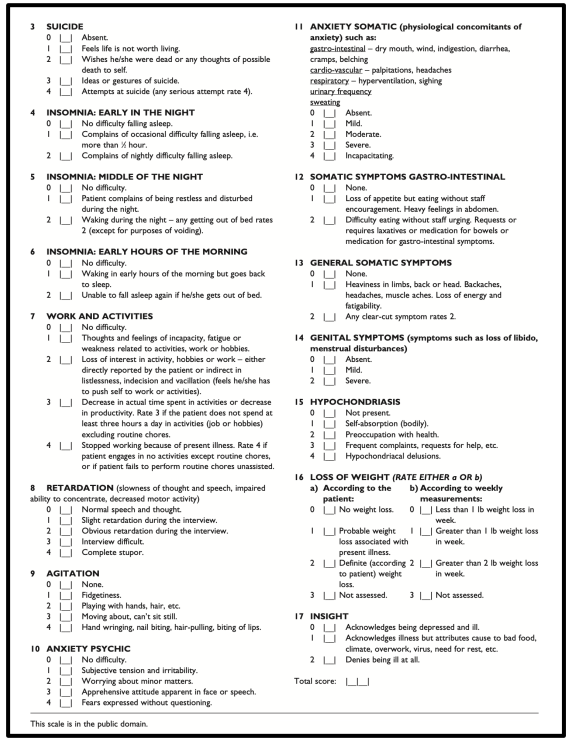
baseline / before treatment	follow up / after treatment	6 month follow up
treatment arm 1	treatment arm 2	treatment arm 3

Instrument/Form:

Class	Grade	Units	GPA
Biology	A-	5	3.7
Calculus	A-	3	3.7
English	B+	3	3.3
History	A	3	4.0

record
record
record

Field / Variable / Measure



3 **SUICIDE**
0 |__| Absent.
1 |__| Feels life is not worth living.
2 |__| Wishes he/she were dead or any thoughts of possible death to self.
3 |__| Ideas or gestures of suicide.
4 |__| Attempts at suicide (any serious attempt rate 4).

4 **INSOMNIA: EARLY IN THE NIGHT**
0 |__| No difficulty falling asleep.
1 |__| Complains of occasional difficulty falling asleep, i.e. more than ½ hour.
2 |__| Complains of nightly difficulty falling asleep.

5 **INSOMNIA: MIDDLE OF THE NIGHT**
0 |__| No difficulty.
1 |__| Patient complains of being restless and disturbed during the night.
2 |__| Waking during the night – any getting out of bed rates 2 (except for purposes of voiding).

6 **INSOMNIA: EARLY HOURS OF THE MORNING**
0 |__| No difficulty.
1 |__| Waking in early hours of the morning but goes back to sleep.
2 |__| Unable to fall asleep again if he/she gets out of bed.

Items measure the test takers' levels of performance on an overall measure of the ability that item was designed to measure.

(item response theory)

3 SUICIDE
0 |__| Absent.
1 |__| Feels life is not worth living.
2 |__| Wishes he/she were dead or any thoughts of possible death to self.
3 |__| Ideas or gestures of suicide.
4 |__| Attempts at suicide (any serious attempt rate 4).

4 INSOMNIA: EARLY IN THE NIGHT
0 |__| No difficulty falling asleep.
1 |__| Complains of occasional difficulty falling asleep, i.e. more than ½ hour.
2 |__| Complains of nightly difficulty falling asleep.

5 INSOMNIA: MIDDLE OF THE NIGHT
0 |__| No difficulty.
1 |__| Patient complains of being restless and disturbed during the night.
2 |__| Waking during the night – any getting out of bed rates 2 (except for purposes of voiding).

6 INSOMNIA: EARLY HOURS OF THE MORNING
0 |__| No difficulty.
1 |__| Waking in early hours of the morning but goes back to sleep.
2 |__| Unable to fall asleep again if he/she gets out of bed.

7 WORK AND ACTIVITIES
0 |__| No difficulty.
1 |__| Thoughts and feelings of incapacity, fatigue or weakness related to activities, work or hobbies.
2 |__| Loss of interest in activity, hobbies or work – either directly reported by the patient or indirect in listlessness, indecision and vacillation (feels he/she has to push self to work or activities).
3 |__| Decrease in actual time spent in activities or decrease in productivity. Rate 3 if the patient does not spend at least three hours a day in activities (job or hobbies) excluding routine chores.
4 |__| Stopped working because of present illness. Rate 4 if patient engages in no activities except routine chores, or if patient fails to perform routine chores unassisted.

8 RETARDATION (slowness of thought and speech, impaired ability to concentrate, decreased motor activity)
0 |__| Normal speech and thought.
1 |__| Slight retardation during the interview.
2 |__| Obvious retardation during the interview.
3 |__| Interview difficult.
4 |__| Complete stupor.

11 ANXIETY SOMATIC (physiological concomitants of anxiety) such as:
gastro-intestinal – dry mouth, wind, indigestion, diarrhea, cramps, belching
cardio-vascular – palpitations, headaches
respiratory – hyperventilation, sighing
urinary frequency
sweating
0 |__| Absent.
1 |__| Mild.
2 |__| Moderate.
3 |__| Severe.
4 |__| Incapacitating.

12 SOMATIC SYMPTOMS GASTRO-INTESTINAL
0 |__| None.
1 |__| Loss of appetite but eating without staff encouragement. Heavy feelings in abdomen.
2 |__| Difficulty eating without staff urging. Requests or requires laxatives or medication for bowels or medication for gastro-intestinal symptoms.

13 GENERAL SOMATIC SYMPTOMS
0 |__| None.
1 |__| Heaviness in limbs, back or head. Backaches, headaches, muscle aches. Loss of energy and fatigability.
2 |__| Any clear-cut symptom rates 2.

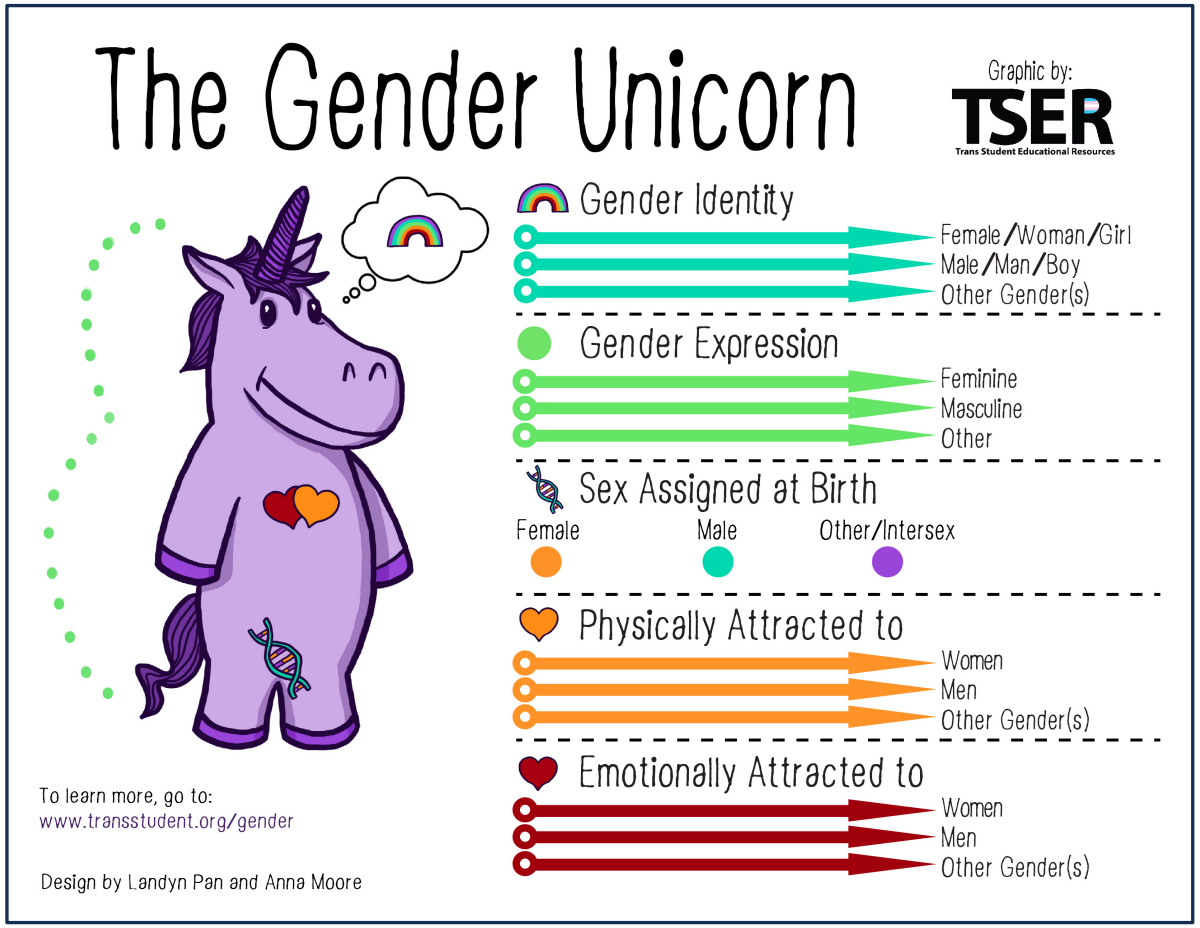
14 GENITAL SYMPTOMS (symptoms such as loss of libido, menstrual disturbances)
0 |__| Absent.
1 |__| Mild.
2 |__| Severe.

15 HYPOCHONDRIASIS
0 |__| Not present.
1 |__| Self-absorption (bodily).
2 |__| Preoccupation with health.
3 |__| Frequent complaints, requests for help, etc.
4 |__| Hypochondriacal delusions.

16 LOSS OF WEIGHT ***(RATE EITHER a OR b)***

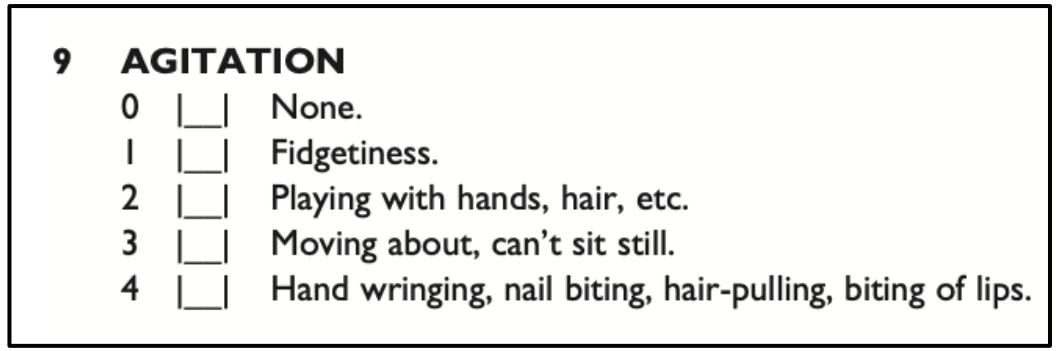
a) According to the patient:	**b) According to weekly measurements:**
0 |__| No weight loss.	0 |__| Less than 1 lb weight loss in week.
1 |__| Probable weight loss associated with present illness.	1 |__| Greater than 1 lb weight loss in week.
2 |__| Definite (according	2 |__| Greater than 2 lb weight loss

HDRS: measure rate of depression



Gender Unicorn: measure gender (gender not the same as sex at birth)

Raw data and raw scores and scores



9 AGITATION

+0	0	|__|	None.
+1	1	|__|	Fidgetiness.
	2	|__|	Playing with hands, hair, etc.
	3	|__|	Moving about, can't sit still.
+4	4	|__|	Hand wringing, nail biting, hair-pulling, biting of lips.

sum of individual scores

t-score table

Score	Male	Female
0	0	0
1-5	10	0
6-10	20	30
11-19	50	60
20-39	80	80
40-59	95	99
60-100	100	100

adjusted score

scaled by covariate of no interest like age, sex at birth, education, income

scaled score

60%

Questions:
Where to store raw data?
Where to store intermediate data?
Where to store scores?

Good experiment documentation is sufficient to re-create the experiment

(and to check if data is valid!)

`redcapdemo.vanderbilt.edu`

sign-up,
create an empty Operational Support project
in Designer edit “Form 1”
add a numeric age value 0 to 120 in years
add a drop-down with 3 choices (1, 2, 3)
add a yes/no field
add a “Why not” text field
Make “Why not” depend on ”no”
Enter some records in “Record status dashboard”
Export data as a spreadsheet (Excel format, raw)
Export data dictionary as Excel and PDF

Instrument by event table

Instrument	Event 01	Event 02	Event 03	Baseline
Grades	✓	✓	✓	
Demographic				✓
Depression rating	✓	✓	✓	

Data table

Record ID	Event name	Measure 1	Measure 2	Measure 3

unique identifier columns (multi-index)

`Data dictionary:`

Variable / Field name	Form name	Section header	Field type	Field label	Choices / Calculations	Field notes	Text validations	Text Validation Min	Text Validation Max	Identifier	Branching logic	Required	…

Summary repeated measures / event related design

- Good experiment description allows us to recreate data capture.
- Essential information
  - instrument by event table for repeated measures
  - data dictionary table to describe all measures (as rows)
  - data table with measures as columns and records as rows

single event Experiment1_DATA_2023-08-02_0535

record_id	age	doing_ok	really	why	form_1_complete
1	33	1	1		0
2	77	2	0	Don't want to say	0
3	1	3	1		0

multiple-event Experiment1_DATA_2023-08-02_0654

record_id	redcap_event_name	age	doing_ok	really	why
1	event_1_arm_1	33	1	1	
1	event_2_arm_1	11	1	1	
2	event_1_arm_1	77	2	0	Don't want to say
2	event_2_arm_1	87	3	0	llll
3	event_1_arm_1	1	3	1	

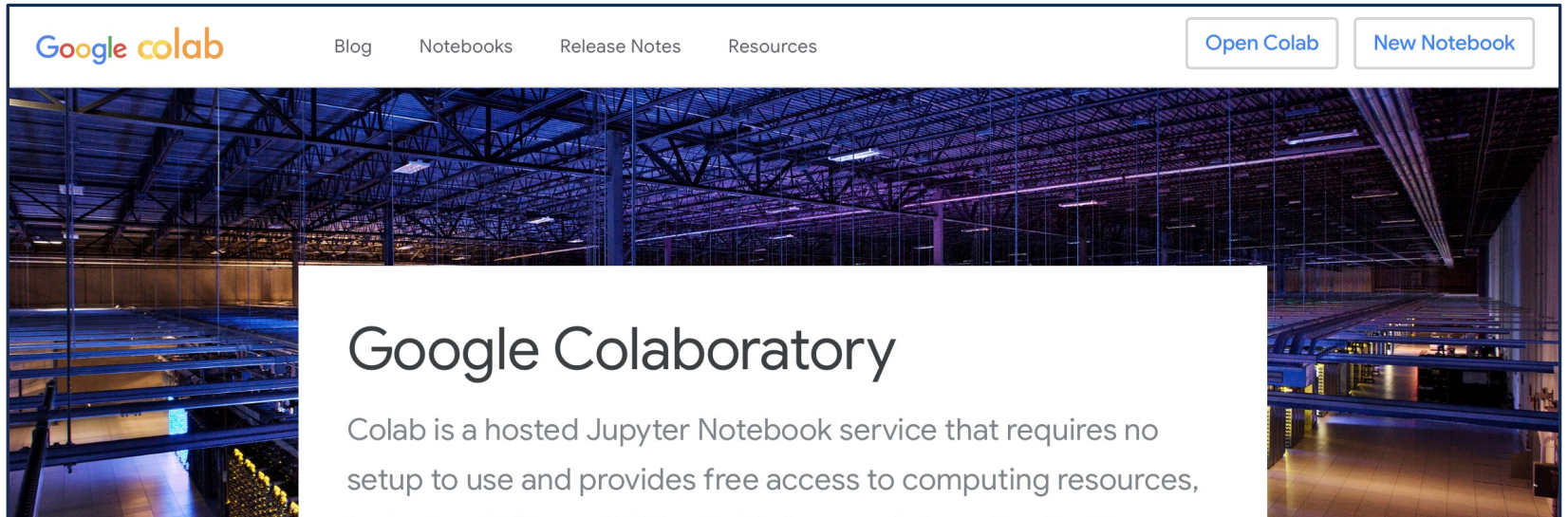
Variable / Field Name	Form Name	Section Header	Field Type	Field Label	Choices, Calculations, OR Slider Labels	Field Note	Text Validation Type
record_id	form_1		text	Record ID			
age	form_1		text	Age		(in years)	integer
doing_ok	form_1		dropdown	Are you doing ok?	1, yes | 2, maybe | 3, no		
really	form_1		yesno	Really?			
why	form_1		text	Why not?			

Python setup as environment with miniconda

DIY — -zsh — 76×19

```
> conda info --env
> conda create --name diy python=3.10
> conda activate diy
> conda install numpy pandas matplotlib
> conda install notebook
> jupyter notebook
```

Some libraries only work with specific versions of other python libraries. Instead of one project per laptop, environments allow us to use one python + libraries per project.
+ Reproducibility



or: colab.google, using google account, use for public data only

End Class 01

Of Pythons and Pandas

Monty Python's Flying Circus +

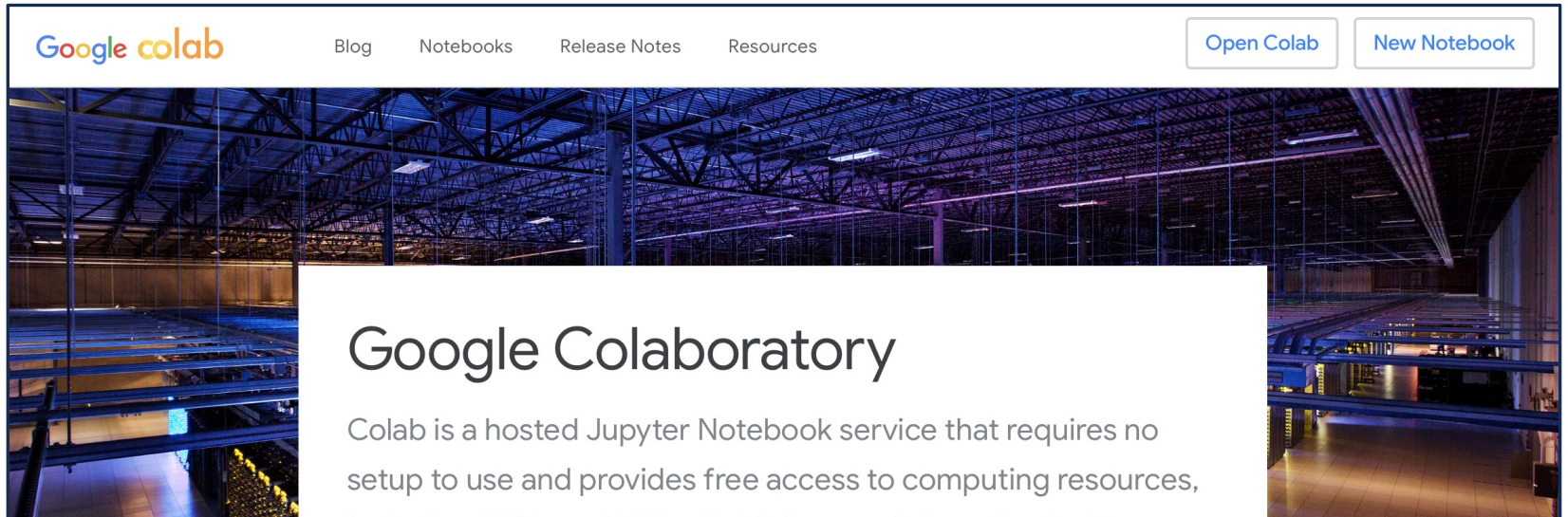
Syntactic Sugar as a feature

Python setup as environment with miniconda

DIY — -zsh — 76×19

```
> conda info --env
> conda create --name diy python=3.10
> conda activate diy
> conda install numpy pandas matplotlib
> conda install notebook
> jupyter notebook
```

Some libraries only work with specific versions of other python libraries. Instead of one project per laptop, environments allow us to use one python + libraries per project.
+ Reproducibility



or: colab.google, using google account, use for public data only

Spreadsheet data

Choices of storing the same information

How many rows per record?

Table 1

record_id	Volume T0	Volume T1
A		
B		
C		
D		
E		
F		

wide format

Table 2

record_id	redcap_event_name	Volume
A	01	
A	02	
B	01	
B	02	
C	01	
C	02	

long format

personal favorite

Table 3

record_id	redcap_event_name	Measure	Value
A	01	Volume	
A	02	Volume	
B	01	Volume	
B	02	Volume	
C	01	Volume	
C	02	Volume	

skinny format

Long to wide format

Input: 4 rows per record
Output: 1 row per record

```
import pandas as pd
df = pd.DataFrame({"record_id": ['A', 'A', 'A', 'A', 'B', 'B', 'B', "B"],
        "redcap_event_name": [1, 2, 3, 4, 1, 2, 3, 4],
        "volume": [11, 8, 10, 6, 12, 5, 9, 4] })
```

```
df
```

	record_id	redcap_event_name	volume
0	A	1	11
1	A	2	8
2	A	3	10
3	A	4	6
4	B	1	12
5	B	2	5
6	B	3	9
7	B	4	4

```
df_wide = pd.pivot(df, index='record_id', columns='redcap_event_name', values='volume')
```

```
df_wide
```

redcap_event_name	1	2	3	4
record_id				
A	11	8	10	6
B	12	5	9	4

Wide to long format

Input: 1 row per record
Output: 4 row per record

```
df_wide
```

redcap_event_name	1	2	3	4
record_id				
A	11	8	10	6
B	12	5	9	4

```
df_wide["id"] = df_wide.index
df_wide
```

redcap_event_name	1	2	3	4	id
record_id					
A	11	8	10	6	A
B	12	5	9	4	B

```
df_long = pd.melt(df_wide, id_vars='id', value_vars=df_wide.keys())
```

```
df_long
```

	id	redcap_event_name	value
0	A	1	11
1	B	1	12
2	A	2	8
3	B	2	5
4	A	3	10
5	B	3	9
6	A	4	6
7	B	4	4

Python reminders

list

```
x = list()
x.append(1)
x.append([1,2,3])
x.extend([1,2,3])
```

set

```
x = set([1, 2])
if 3 in x:
    print("yes")
```

dictionary

```
x = { "A": "bla" }
x["B"] = "blub"
```

int, str, bool

```
x = 1
x = "bla"
x = False
```

loops

```
for x in [1, 2]:
  print(x)
for i in range(0,len(x)):
  print(i)
```

numpy arrays

```
x = np.array([1, 2])
```

tuple

x = ([1, 2], 3)

x = 1, 2, 3

x = (1,)

weird stuff

type(np.nan) == float

0.1+0.2 != 0.3

pandas DataFrame

```
x = pd.read_csv("A.csv")
x.head()
x.tail()
x.iloc[-1]
```

First steps with pandas

Difference between pandas and numpy?

```
import pandas as pd
df = pd.read_csv('Experiment1_DATA_2023-08-02_0535.csv')
df.head()
df.columns, df.index
df["why"]
pd.isna(df["why"])
df.describe()
df.age.describe()
df[(df.age > 10) & (df.age < 77)]
```

Tasks: Import 2 different versions of your data with read_csv. One version with comma, one with semi-colon separated values.

use help(“pandas.read_table”) and help(”pandas.read_excel”)

Task: Find real data and read in using pandas.read_csv()

Example data from PGC consortium:
https://pgc.unc.edu/for-researchers/download-results/

pgcGroup	phenotype	publication	journal	pubMedIDlink	Data DOI	Download link
Attention Deficit Hyperactivity Disorder						
ADHD	ADHD	adhd2022	Nature Genetics	36702997	10.6084/m9.figshare.22564390	download
ADHD	ADHD	adhd2019	Nature Genetics	30478444	10.6084/m9.figshare.14671965	download
ADHD	ADHD	adhd2018_SexSpecific	Biol Psychiatry	29325848	10.6084/m9.figshare.19383299	download
ADHD	ADHD	adhd2010	JAACP	20732625	10.6084/m9.figshare.14671962	download

Third steps in pandas

```
df = pd.read_csv('Experiment1_DATA_2023-08-02_0654.csv', \
      index_col=list(["record_id", "redcap_event_name"]) )


df.grouped = df[["age", "really"]].groupby("redcap_event_name")
df.grouped.describe()


df2 = df.reset_index()
df2 = df2[df2.redcap_event_name == "event_1_arm_1"]
df2.drop(columns=["redcap_event_name", "form_1_complete"])


!pip install tableone
from tableone import TableOne
t = TableOne(df)
print(t.tabulate(tablefmt = "fancy_grid"))
t.to_csv("summary.csv")
```

Data Science Preparation Scripts: Read – Recode – Drop – Write

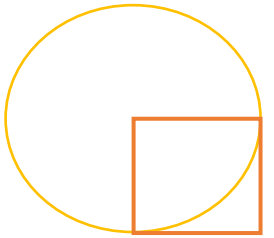
```
df = pd.read_csv('Experiment1_DATA_2023-08-02_0654.csv', \
      index_col=list(["record_id", "redcap_event_name"]) )

df = df.assign(reallyF = df.really.map({ 0: "Yes", 1: "No" }))
df = df.assign(doingokF = df.doing_ok.map({ 1: "Yes", 2: "Maybe", 3: "no" }))

# df = df[["record_id", "redcap_event_name", "age", "reallyF", "doingokF"]]
df.drop(columns=["really", "doing_ok", "form_1_complete"])

# store as pickle for future use without the need for re-coding
df.to_pickle("data_recoded.pickle")
# df_again = pandas.read_pickle("data_recoded.pickle")
```

Special recodes

```
df2 = pd.read_csv('Experiment1_DATA_2023-08-02_0654.csv')

# let’s say a column "Y" contains "A3_1B" but it should be read "A", 3, 1, "B"
df2["Y"] = pd.Series(["A1_2B","B2_3B","B1_2C","A4_1C","A4_3A"],dtype="string")

df2[['Y1', 'Y2']] = df2['Y'].apply(lambda x: pd.Series(str(x).split("_").split("")))
df2[["Y11", "Y12"]] = df2.Y1.apply(lambda x: pd.Series([x for x in str(x)]))
df2[["Y21", "Y22"]] = df2.Y2.apply(lambda x: pd.Series([x for x in str(x)]))

# let’s say a column contains dates "29/08/2022 08:30”
df2["D"] = pd.Series(5 * ["29/08/2022 08:30"], dtype="string")

df2["Date"] = pd.to_datetime(df2['D'], format="%d/%m/%Y %H:%M")
df2.grouped = df2.groupby("record_id")["Date"]
diff_by_id = (df2.grouped.first() - df2.grouped.last()) / np.timedelta64(1, 'D')
df2 = df2.assign(DateDiff = df2["record_id"].map( lambda x: diff_by_id[x]))
```

End Class 02

Simple loops – computing pi using Monte Carlo



```
import numpy as np
n = 10000
X = np.random.rand(n)
Y = np.random.rand(n)

count = 0
for x, y in zip(X, Y):
    if x*x + y*y < 1:
        count += 1
print(4.0*count/n)
```

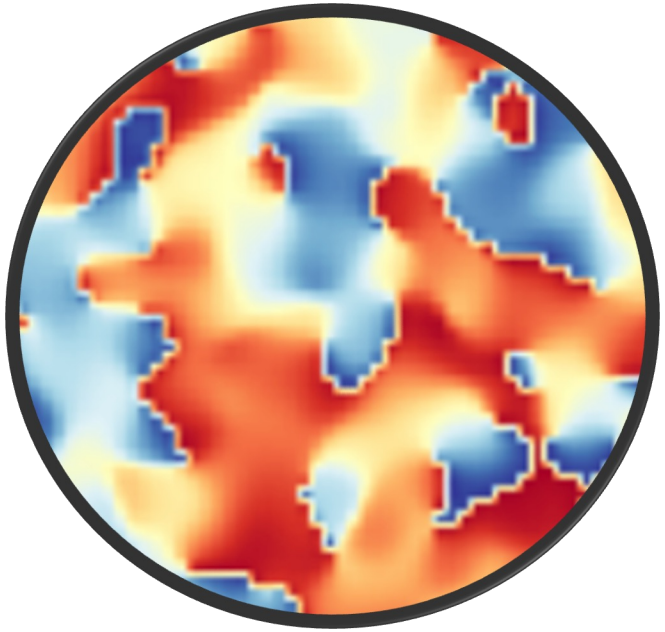
```
import num as np
n = 10000
X, Y = np.random.rand(2,n)
Z = X**2 + Y**2
print("PI approx.: %f" % (4*sum(Z<1)/n))
```

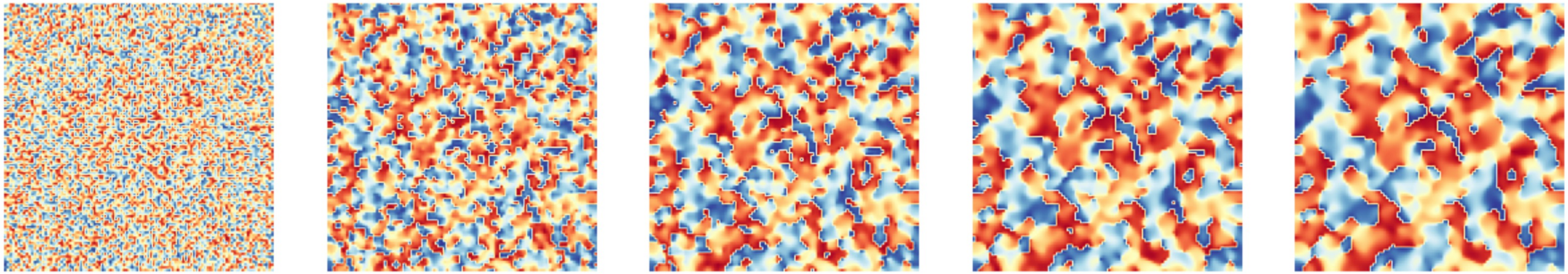
```
import numpy as np
X,Y = np.random.rand(2,10000)
print(4.0*sum((X**2+Y**2)<1)/n)
```

```
import numpy as np
n = 10000
X,Y = np.random.rand(2,n)

def proc(X,Y):
  count = 0
  for x, y in zip(X, Y):
     if x*x + y*y < 1:
        count += 1
  return count

%timeit –r 1 –n 10 4.0*proc(X,Y)/n)
```

Bandpass-filtered white noise example

```
import matplotlib.pyplot as plt
from scipy.ndimage import gaussian_filter

img1 = np.random.rand(512,512)-0.5
img2 = np.random.rand(512,512)-0.5
plt.figure(figsize=(15,15))
for i in range(5):
    plt.subplot(1,5,i+1)
    plt.imshow(np.angle(img1 + 1j * img2), cmap='RdYlBu_r')
    plt.axis('equal')
    plt.axis('image')
    plt.axis('off')
    img1 = gaussian_filter(img1,7)
    img2 = gaussian_filter(img2,7)
```

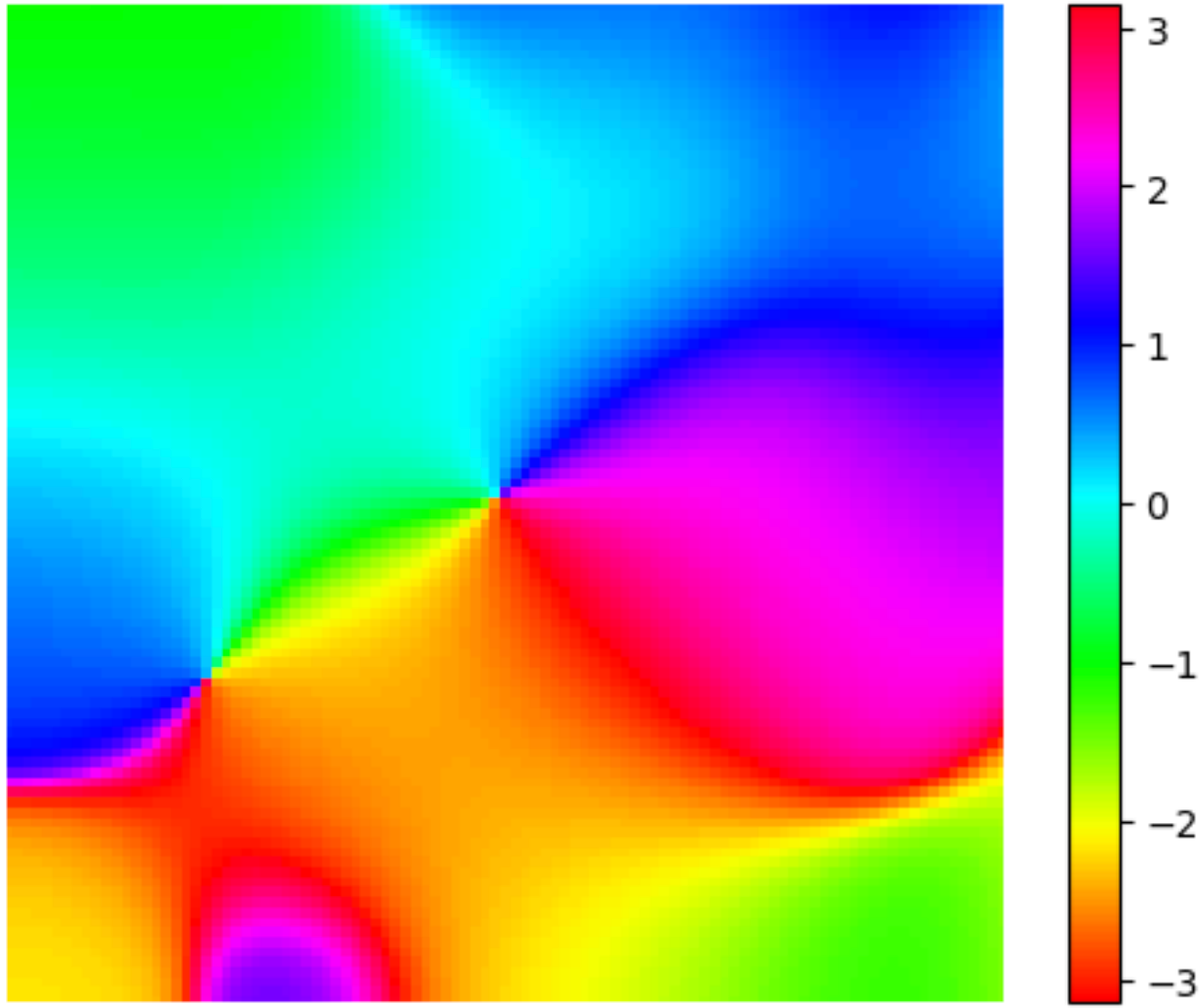


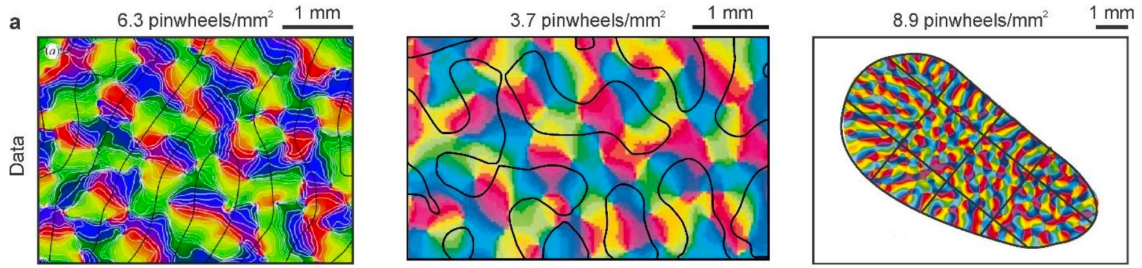


Zooming in and better colormap

https://matplotlib.org/stable/gallery/color/colormap_reference.html

```
plt.imshow(np.angle(img1 + 1j * img2), cmap='hsv')
plt.axis('equal')
plt.axis('image')
plt.axis('off')
plt.xlim([0,100])
plt.ylim([0,100])
plt.colorbar()
```

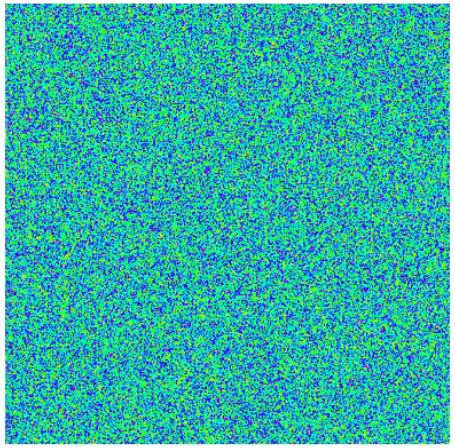




Saving as animation for websites

```
import numpy as np
import imageio
import os
import matplotlib.pyplot as plt
from scipy.ndimage import gaussian_filter

png_dir = 'pngs/'

img1 = np.random.rand(512,512)-0.5
img2 = np.random.rand(512,512)-0.5
plt.figure(figsize=(5,5))
# save them into a folder
for i in range(65):
    plt.imshow(np.angle(img1 + 1j * img2), cmap='hsv')
    plt.axis('equal')
    plt.axis('image')
    plt.axis('off')
    plt.axis('tight')
    img1 = gaussian_filter(img1,1)
    img2 = gaussian_filter(img2,1)
    plt.savefig("%s/img%02d.png" % (png_dir,i))
```

```
images = []
for file_name in sorted(os.listdir(png_dir)):
   if file_name.endswith('.png'):
      file_path = os.path.join(png_dir, file_name)
      images.append(imageio.imread(file_path))

for _ in range(10):
   images.append(imageio.imread(file_path))

imageio.mimsave('movie.gif', images, duration=120)
```



Strategies for missing data

Values used for missing:
<empty>, “NaN”, 999, 777, …

Why is missing data a problem? Because algorithms do not work with missing data.

```
# keep only complete cases
df = df[~df.isna().any(axis=1)]
```

Example: Ask for income but all high-income people refuse to answer (MAR, difficult to correct).

Example: Forget to ask for income (MCAR, there is hope).

good type of missing

Missing at random (MAR) or missing completely at random (MCAR)

Missing not at random (structurally missing)

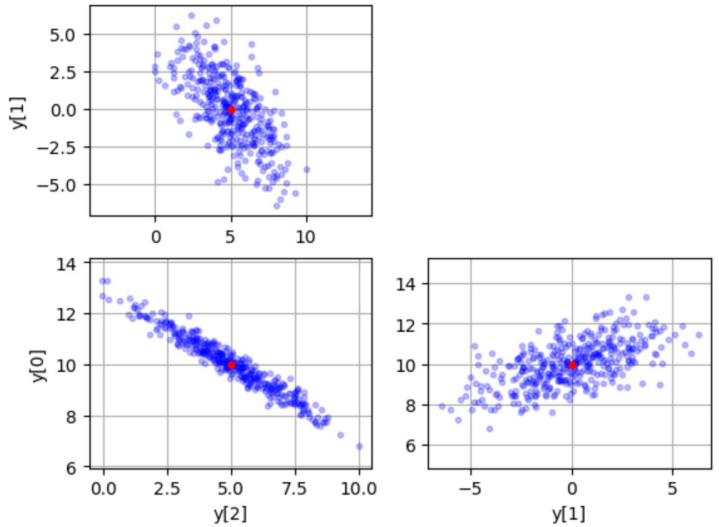
Rule of thumb: If less than 10% of the data is missing and missingness is of type MCAR we can use *imputation* to recover values for missing entries.

Imputation: Invent missing values based on pattern in the data.

Create test data with known structure

If we set some values to nan, we can test imputation

```
import numpy as np
n = 400
mu = np.array([5.0, 0.0, 10.0])

# The desired variance-covariance matrix.
r = np.array([   [  3.40,-2.75,-2.00],
                 [ -2.75, 5.50, 1.50],
                 [ -2.00, 1.50, 1.25] ])


rng = np.random.default_rng()
y = rng.multivariate_normal(mu, r, size=n)


import pandas as pd
data = pd.DataFrame(y) # y_hat = data.to_numpy()

print(data[2][0])
data[2][0] = np.nan
data[0][2] = np.nan
```




```
import matplotlib.pyplot as plt

# Plot various projections of the samples.
plt.subplot(2,2,1)
plt.plot(y[:,0], y[:,1], 'b.', alpha=0.25)
plt.plot(mu[0], mu[1], 'ro', ms=3.5)
plt.ylabel('y[1]’)
plt.axis('equal’)
plt.grid(True)

plt.subplot(2,2,3)
plt.plot(y[:,0], y[:,2], 'b.', alpha=0.25)
plt.plot(mu[0], mu[2], 'ro', ms=3.5)
plt.xlabel('y[0]') plt.ylabel('y[2]’)
plt.axis('equal’)
plt.grid(True)

plt.subplot(2,2,4)
plt.plot(y[:,1], y[:,2], 'b.', alpha=0.25)
plt.plot(mu[1], mu[2], 'ro', ms=3.5)
plt.xlabel('y[1]’)
plt.axis('equal’)
plt.grid(True)

plt.show()
```

Multiple Imputation using Chained Equations (MICE)

```
!pip install sklearn
import numpy as np
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
from sklearn.linear_model import LinearRegression

lr = LinearRegression()
imp = IterativeImputer(
        estimator=lr,
        missing_values=np.nan,
        max_iter=10,
        verbose=2,
        imputation_order='roman’,
        random_state=0)
O = imp.fit_transform(impute_data.to_numpy())
```

before

	0	1	2
0	4.241496	1.465245	10.331013
1	5.922603	-1.889773	9.345456
2	5.203843	2.373977	9.761845
3	3.103197	1.995237	11.310436
4	4.128783	-1.918453	10.272100

set to NaN

	0	1	2
0	4.241496	1.465245	NaN
1	5.922603	-1.889773	9.345456
2	NaN	2.373977	9.761845
3	3.103197	1.995237	11.310436
4	4.128783	-1.918453	10.272100

imputed with MICE

	0	1	2
0	4.241496	1.465245	10.429647
1	5.922603	-1.889773	9.345456
2	5.164974	2.373977	9.761845
3	3.103197	1.995237	11.310436
4	4.128783	-1.918453	10.272100

Merging of tables – left/right vs. inner/outer

```
table1 = pd.DataFrame({ "record_id": [ "patient_001", "patient_002", "patient_003", "patient_004"],
                        "A": [ 1, 3, 5, 7],
                        "B": [ 2, 4, 6, 8] })
table2 = pd.DataFrame({ "record_id": [ "patient_001", "patient_005" ], "C": [9, 11], "D": [10,12] })
```

Table 1

record_id	A	B
patient_001	1	2
patient_002	3	4
patient_003	5	6
patient_004	7	8

Table 2

record_id	C	D
patient_001	9	10
patient_005	11	12

```
pd.merge(left= table1, right= table2)
```

	record_id	A	B	C	D
0	patient_001	1	2	9	10

```
pd.merge(left=table2, right=table1)
```

	record_id	C	D	A	B
0	patient_001	9	10	1	2

```
pd.merge(left=table1, right=table2, how='outer')
```

	record_id	A	B	C	D
0	patient_001	1.0	2.0	9.0	10.0
1	patient_002	3.0	4.0	NaN	NaN
2	patient_003	5.0	6.0	NaN	NaN
3	patient_004	7.0	8.0	NaN	NaN
4	patient_005	NaN	NaN	11.0	12.0

Python – more like R’s dplyr

```
!pip install Dplython

from dplython import (DplyFrame, X, diamonds, select, sift,
                      sample_n, sample_frac, head, arrange,
                      mutate, group_by, summarize, DelayFunction)

data = DplyFrame(pd.read_pickle('pickled_df2.pickle'))

data >> select(X.record_id, X.redcap_event_name, X.age) >> head(5)
data >> sample_n(3) >> arrange(X.age) >> select(X.record_id, X.redcap_event_name)

(
data
  >> sample_n(3)
  >> arrange(X.age)
  >> select(X.record_id, X.redcap_event_name)
)
```

Plotting using plotnine – more like R's ggplot

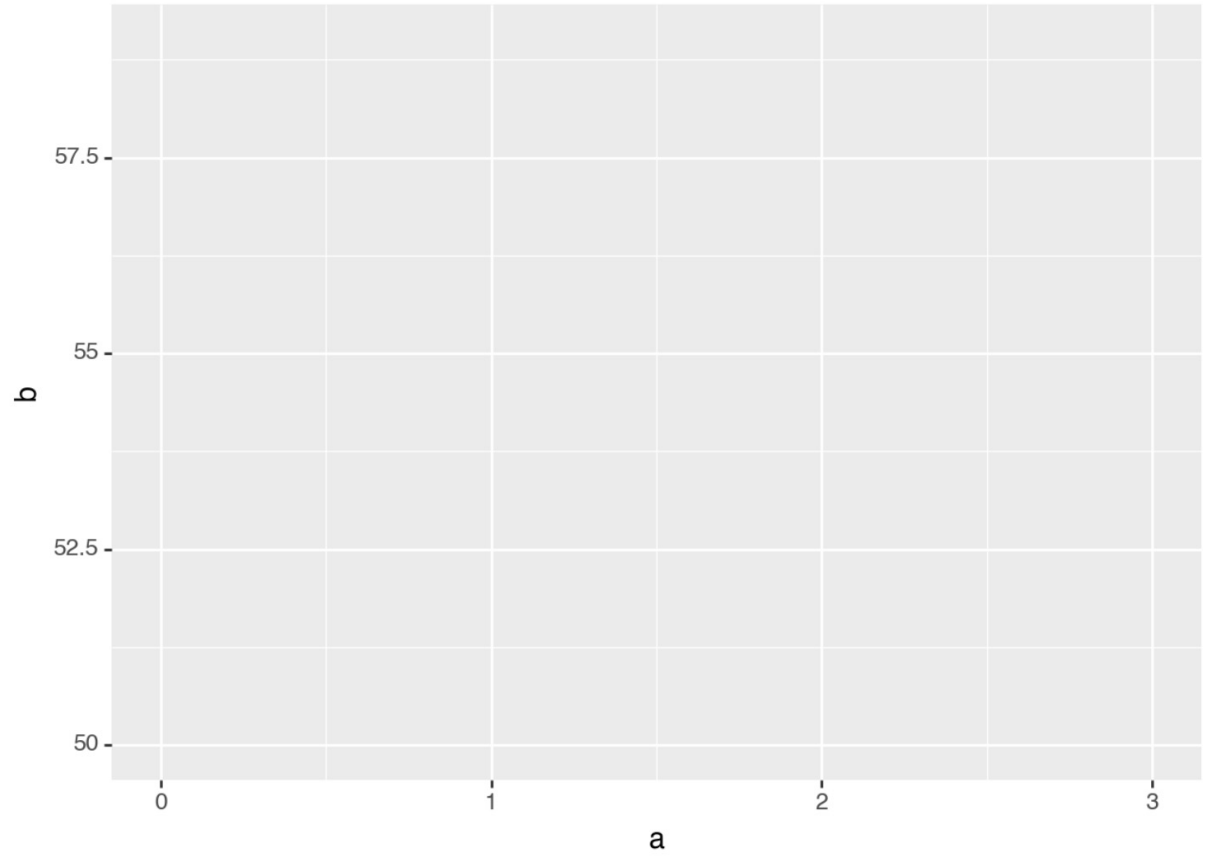
ggplot, a "grammar for graphics" mapping variables to aesthetics, specify what graphical primitives to use, and it takes care of the details.

```
import plotnine
df = pd.DataFrame({ "A": [ 1, 3, 5, 7], "B": [ 2, 4, 6, 8] })
(
  ggplot(df)                          # What data to use
       + aes(x="A", y="B")            # What variables to use
       + geom_line()                  # Geometric object to use for drawing
)
```

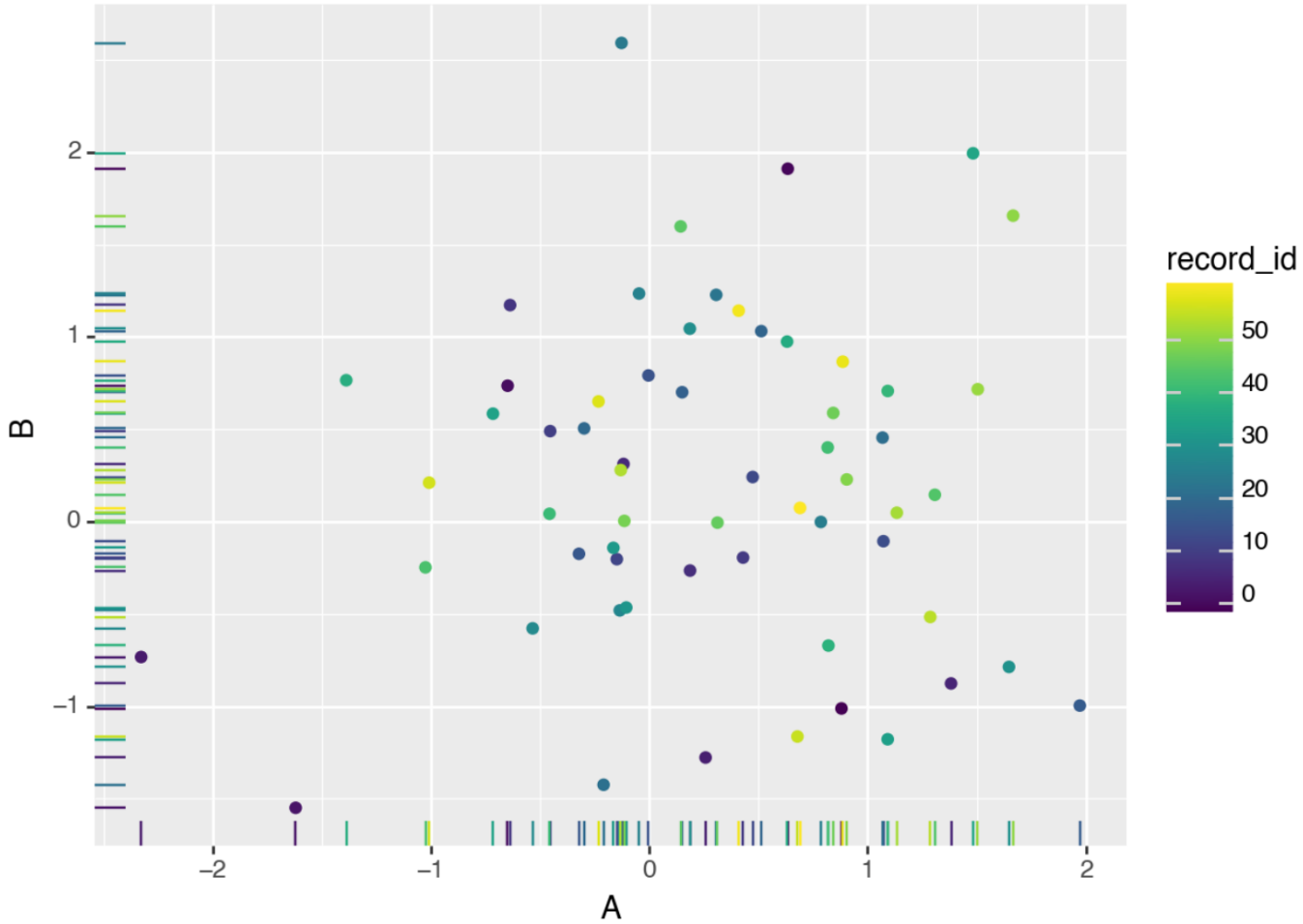
```
df_empty = pd.DataFrame({ "a": [0, 1, 2, 3], "b": [55, 50, 53, 59] })
(
  ggplot(df_empty)
)
```



```
df_empty = pd.DataFrame({ "a": [0, 1, 2, 3], "b": [55, 50, 53, 59] })
(
  ggplot(df_empty)
    + aes("a", "b")
)
```



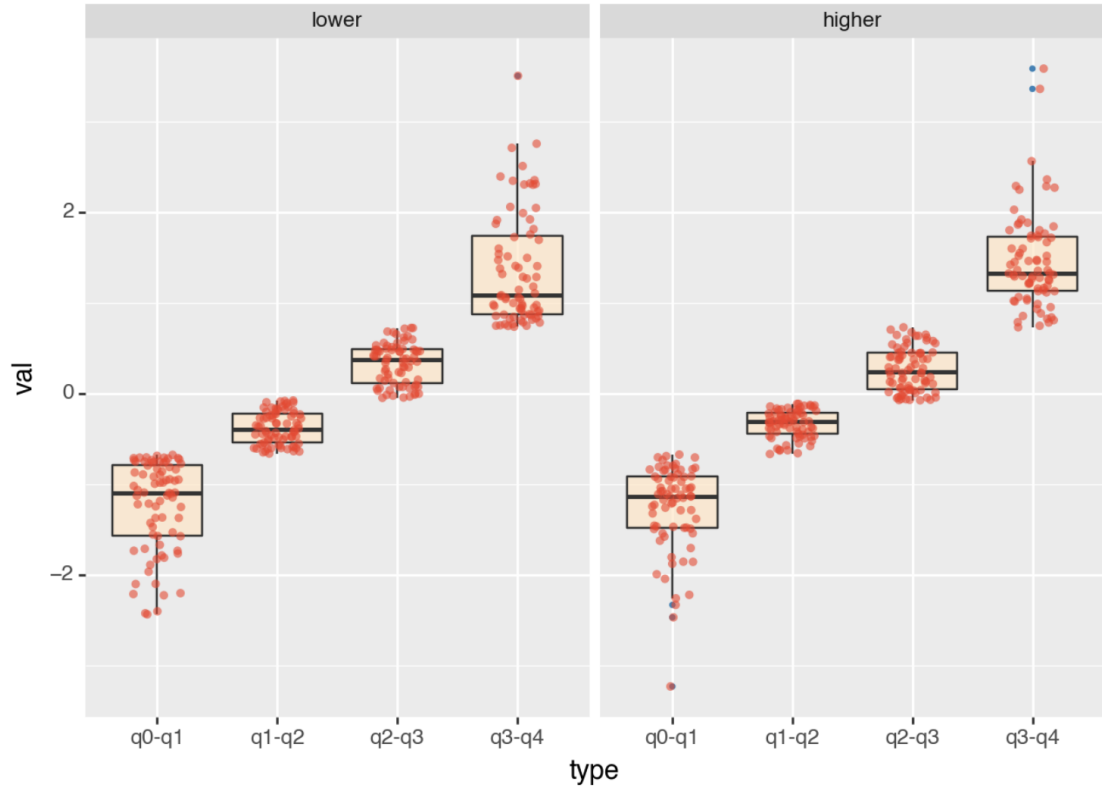
```
import plotnine
import numpy as np
df = pd.DataFrame({ "A": np.random.randn(60), "B": np.random.randn(60), "record_id": list(range(0,60,1)) })
(
  ggplot(df)
       + aes(x="A", y="B", color="record_id")
       + geom_point()
       + geom_rug()
)
```



```
from plotnine import (aes, ggplot, geom_boxplot, geom_jitter, facet_grid)
import numpy as np

df = pd.DataFrame({ "A": np.random.randn(600), "B": np.random.randn(600), "val": np.random.randn(600) })
q = np.quantile(df["val"], [0,0.25,0.5,0.75,1])
df["type"] = pd.cut(df["val"], bins=q, labels=['q0-q1', 'q1-q2', 'q2-q3', 'q3-q4'], include_lowest=True)
df["type2"] = pd.qcut(df["B"], 2, labels=["lower", "higher"])
```

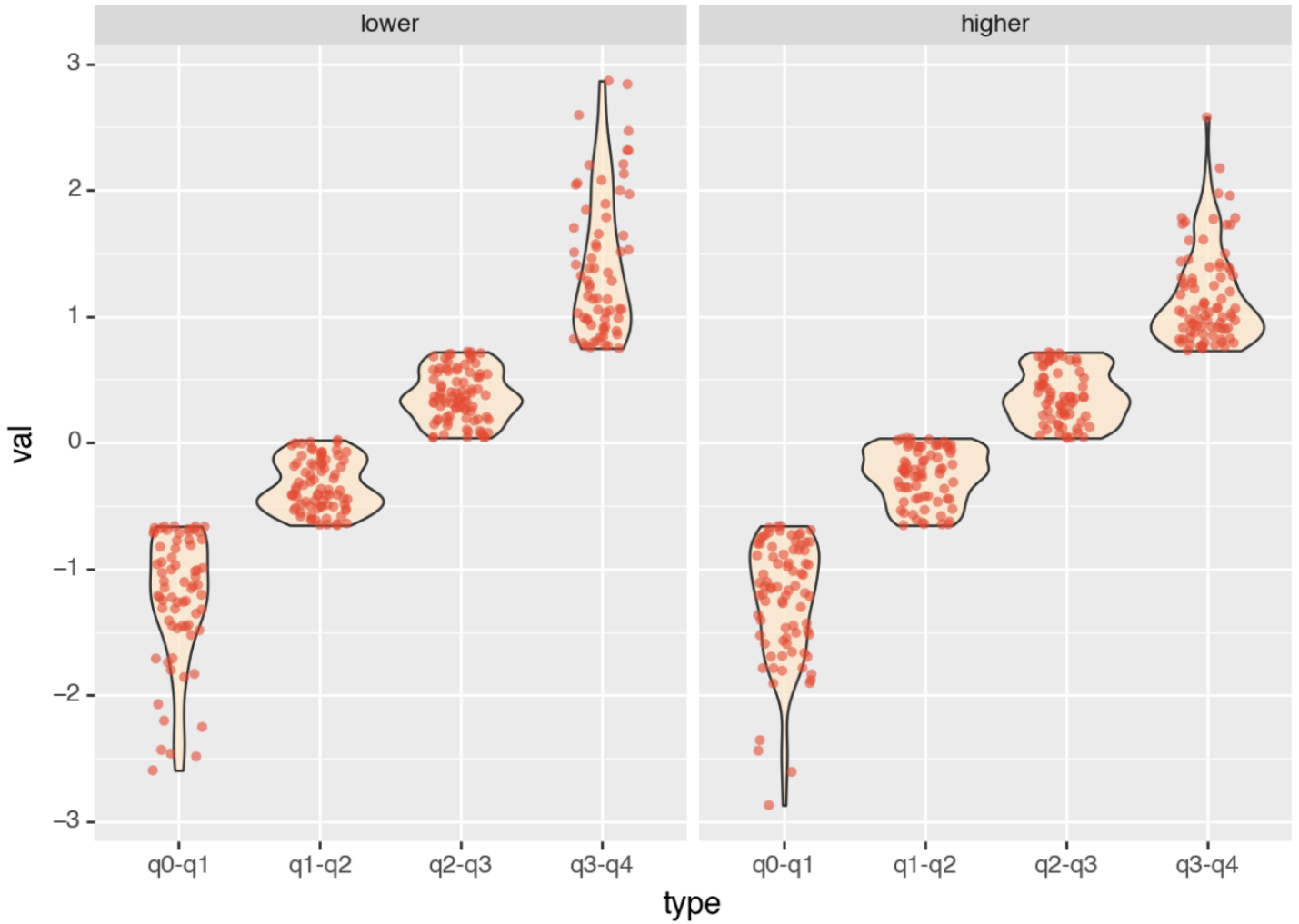
```
df.head()
```

	A	B	val	type	type2
0	0.732601	-1.366299	-1.181452	q0-q1	lower
1	-0.065415	-1.632031	0.454027	q2-q3	lower
2	-0.837869	-0.846796	-0.802154	q0-q1	lower
3	-0.156724	-0.221354	-0.247113	q1-q2	lower
4	-1.108569	0.276229	-0.379684	q1-q2	higher

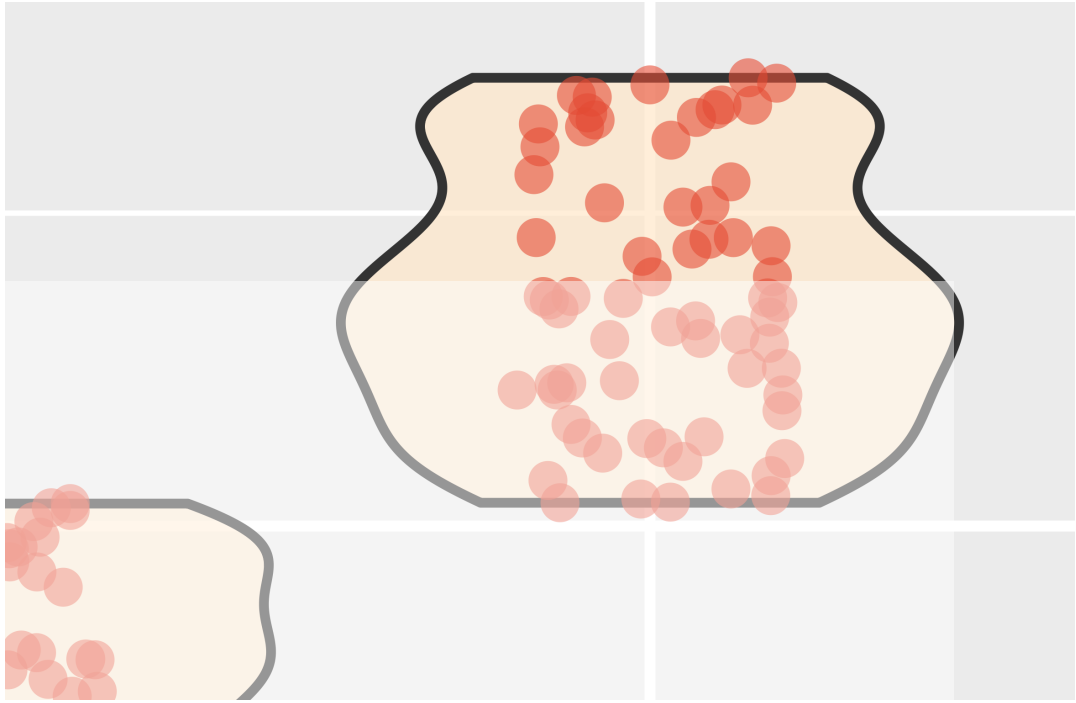



```
(
  ggplot(df)
    + aes("type", "val")
    + geom_boxplot(fill="#fee8c8",alpha=0.7, outlier_shape=".", outlier_colour="steelblue",)
    + geom_jitter(width=0.2, alpha=0.6, fill="#e34a33", stroke=0, size=2)
    + facet_grid(". ~ type2")
)
```

```
(
  ggplot(df)
    + aes("type", "val")
    + geom_violin(fill="#fee8c8",alpha=0.7)
    + geom_jitter(width=0.2, alpha=0.6, fill="#e34a33", stroke=0, size=2)
    + facet_grid(". ~ type2")
)
```



Export plots for publications



```
from plotnine import (save_as_pdf_pages, theme)


plot = (
  ggplot(df)
    + aes("type", "val")
    + geom_violin(fill="#fee8c8",alpha=0.7)
    + geom_jitter(width=0.2, alpha=0.6, fill="#e34a33", stroke=0, size=2)
    + facet_grid(". ~ type2")
)
save_as_pdf_pages([plot + theme(figure_size=(8, 6))])
```

End Class 03

Data quality improvement

Improving data quality is a process known as data curation. Data curation begins when you start collecting data, and it never ends.

Make a (trivial) assumption about your data, create an algorithm to test that assumption, and finally fix any wrong data in-place before starting again.

Keep only a single version/location for your data.
Any exported version of your data is out-of-date.
Backups are nice to have but do not trust them.
Use semantic versioning and DOI’s for data resources.

What are trivial assumptions?

- With an index column there should be no duplicate records.
- The date of the second visit should be after the date of the first visit.
- There are no records without any data.
- All calculated fields are present and correct.
- There are no fields with values and hidden by branching logic.
- There are no field validation errors.
- For all missing data I can tell you why they are missing.

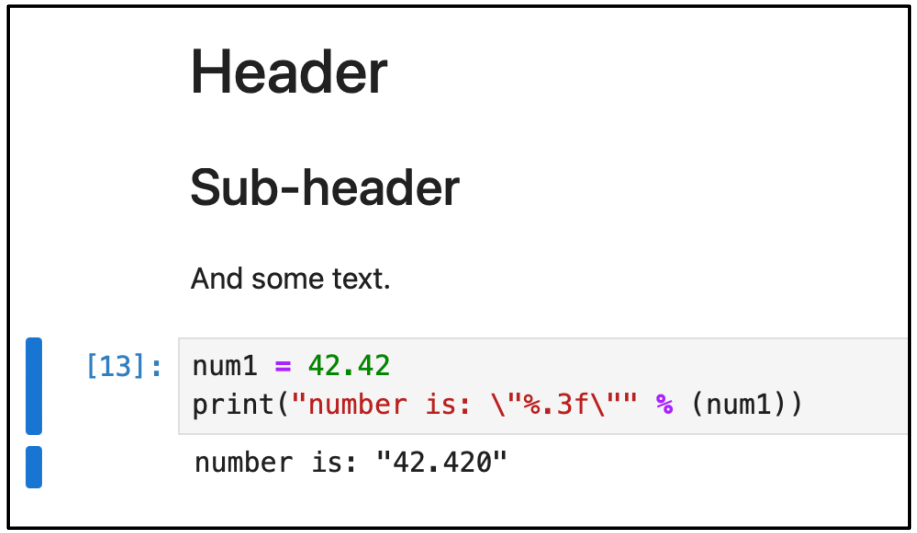
Notebook or script or program

How about proto-typing, debugging, organization, reproducibility?

script.py

```
"""A script to print a number"""
num1 = 42.42
print("number is: %.3f" % (num1))
```

notebook.ipynb



program.py

```
#!/usr/bin/env python

def main():
    """Our main function"""
    num1 = 42.42
    print("number is: %.3f" % (num1))


if __name__=="__main__":
    main()
```

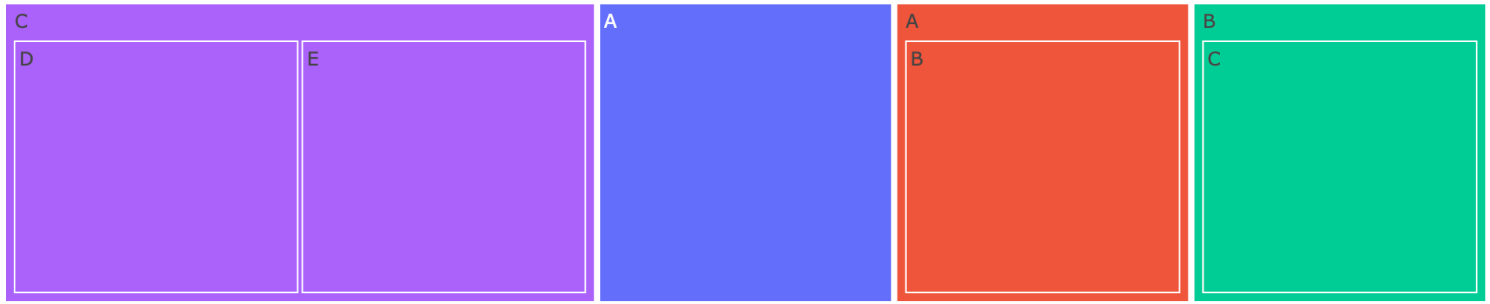
A program in python

```
#!/usr/bin/env python
import getopt, sys


def main():
    # Remove 1st argument from the list of command line arguments
    args = sys.argv[1:]

    options = "hmo:"
    long_options = ["Help", "My_file", "Output="]

    try:
        # Parsing argument
        arguments, values = getopt.getopt(args, options, long_options)

        # checking each argument
        for currentArgument, currentValue in arguments:
            if currentArgument in ("-h", "--Help"):
                print ("Displaying Help")
            elif currentArgument in ("-m", "--My_file"):
                print ("Displaying file_name:", sys.argv[0])
            elif currentArgument in ("-o", "--Output"):
                print (("Enabling special output mode (% s)") % (currentValue))

    except getopt.error as err:
        # output error, and return with an error code
        print (str(err))
    num1 = 42.42
    print("number is: %.3f" % (num1))


if __name__=="__main__":
    main()
```

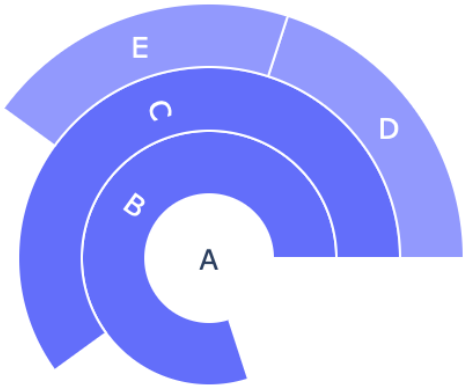
Hierarchies

```
import pandas as pd
h = pd.DataFrame( {
        "level": [0, 1, 2, 3, 3],
        "name": ["A", "B", "C", "D", "E"],
        "w": [1, 1, 1, 1, 1], "Parent": ""
} )
for row in h.iterrows():
    idx = row[0]
    l = row[1]["level"]
    for c in range(idx, 0, -1):
        entry = h.loc[c-1]
        entry_level = entry["level"]
        entry_name = entry["name"]
         if entry_level == l-1:
             h.loc[idx,"Parent"] = entry_name
             break
```

level	name	w	Parent
0	A	1	
1	B	1	A
2	C	1	B
3	D	1	C
3	E	1	C

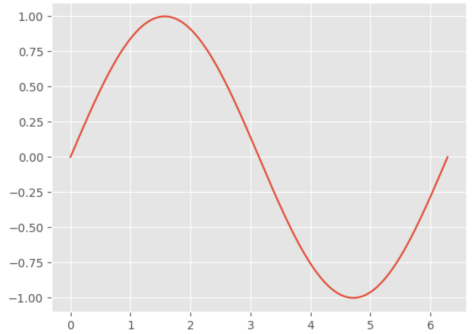



```
import plotly.express as px
fig3 = px.treemap(h, path=['Parent', 'name'], values='w', color='Parent')
fig3.show()
```

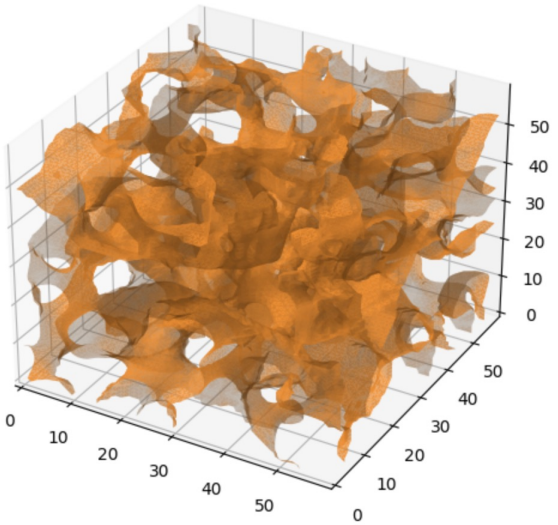


Matplotlib

```
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(0, 2 * np.pi, 200)
y = np.sin(x)

# print(plt.style.available)
plt.style.use('ggplot')
fig, ax = plt.subplots()
ax.plot(x, y)
plt.show()
```



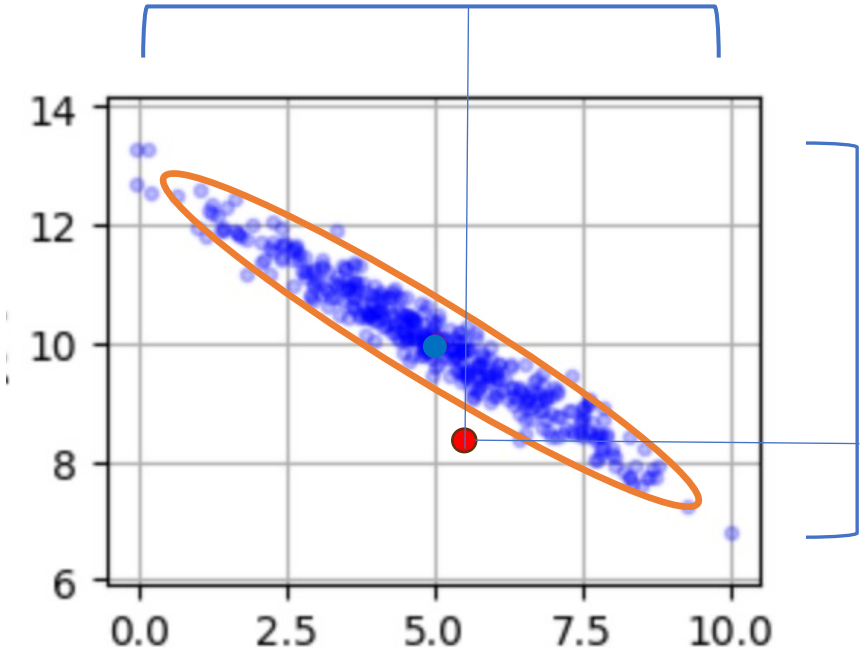

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d.art3d import Poly3DCollection
from matplotlib.colors import LightSource
from scipy.ndimage import gaussian_filter
from skimage import measure

data = np.random.randn(60,60,60)
data = gaussian_filter(data, sigma=3)

verts, faces, normals, values = measure.marching_cubes(data, 0)

fig = plt.figure(figsize=(5, 5))
ax = fig.add_subplot(111, projection="3d")

ax.set_xlim(np.min(verts[:,0]), np.max(verts[:,0]))
ax.set_ylim(np.min(verts[:,1]), np.max(verts[:,1]))
ax.set_zlim(np.min(verts[:,2]), np.max(verts[:,2]))

mesh = Poly3DCollection(verts[faces])
ls = LightSource(azdeg=225.0, altdeg=45.0)
normalsarray = np.array([np.array((np.sum(normals[face[:], 0]/3),  np.sum(normals[face[:], 1]/3),
                        np.sum(normals[face[:], 2]/3))/np.sqrt(np.sum(normals[face[:], 0]/3)**2
                        + np.sum(normals[face[:], 1]/3)**2 + np.sum(normals[face[:], 2]/3)**2)) for face in faces])
min = np.min(ls.shade_normals(normalsarray, fraction=1.0)) # min shade value
max = np.max(ls.shade_normals(normalsarray, fraction=1.0)) # max shade value
diff = max-min
newMin = 0.3
newMax = 0.95
newdiff = newMax-newMin

# Using a constant color, put in desired RGB values here.
colourRGB = np.array((255.0/255.0, 154.0/255.0, 57/255.0, 1.0))

rgbNew = np.array([colourRGB*(newMin + newdiff*((shade-min)/diff)) for shade in ls.shade_normals(normalsarray, fraction=1.0)])
mesh.set_facecolor(rgbNew)

ax.add_collection3d(mesh)
plt.tight_layout()
plt.show()
```



Outlier detection

In a random sample an outlier is far away from other values. Outliers can be highlighted, or censored, or set to some border value (Winsorize).

```
# !pip install scikit-learn
import pandas as pd
from sklearn.covariance import EllipticEnvelope
from sklearn.metrics import mean_absolute_error

df=pd.read_csv('Datasets.csv')
data = df.to_numpy()

ee = EllipticEnvelope(contamination=0.01)
yhat = ee.fit_predict(data)

mask = yhat != -1
cleaned = data[mask]
cleaned.shape, data.shape
```



Sample of sample, sample of cohort, sample of population

Question: Is it the same if I do an analysis on my data or on your data?

Hint: Two samples are more than one sample.

Propensity scoring

Make measurements independent of sample and depended on population by correcting for sample bias relative to population.