Affective User Research & Human-AI Interaction

Seminar Summer 2024, Karlsruhe Institute of Technology
Dr. Ivo Benke, BioNTech
Dr. Lennard Schmidt, Google

Karlsruhe Institute of Technology

human-centered systems lab

This image was created with the assistance of DALL·E 2

Affective User Research & Human-AI Interaction

Seminar #2 Part: Dataset introduction & Data Exploration Methods

Dr. Ivo Benke, Dr. Lennard Schmidt

Agenda

Dataset Experience

Dataset collection experience

“Eat your own dogfood”

Try the dataset collection and understand user experience

For the question for your prolific ID please answer with **2024-07_kit_seminar**

Visit: https://survey.iism.kit.edu/index.php/715885?newtest=Y&lang=de

Dataset Description

Dataset Introduction

Karlsruhe Institute of Technology

Dataset “Affective Experiences in LLM Interaction”

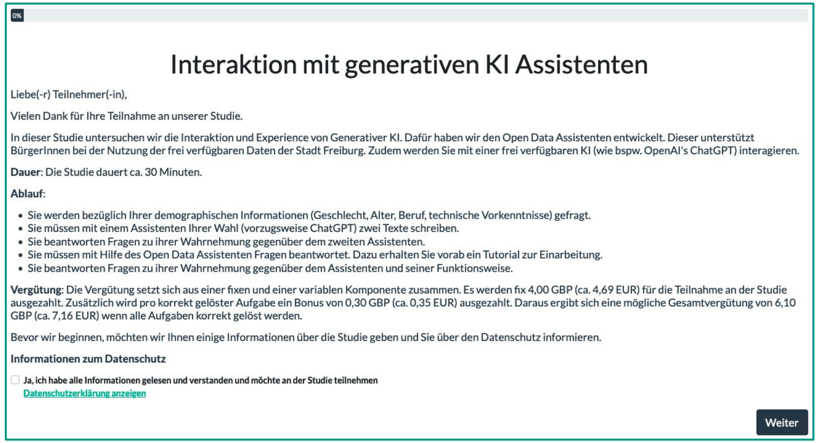
Objective: Understanding how users interact with LLM-based generative AI assistants and how this behavior in form of user prompts is influenced by emotions and how the prompts and answers by the generative AI assistant influence emotions of the users.

Dataset contains:
- User prompts and generative AI assistant answers
- User emotions in form of valence and arousal
- User click interaction with generative AI assistant tool
- User perceptions

Data collection for this seminar via limesurvey and prolific.com with a study for around 30 minutes:
- 83 individual subjects
- 1244 labels for valence and arousal
- >2559 messages

Two generative AI assistants in use: OpenAI’s ChatGPT, Self-developed “Open Assistant”

OpenAI



Interaktion mit generativen KI Assistenten

Liebe(-r) Teilnehmer(-in),

Vielen Dank für Ihre Teilnahme an unserer Studie.

In dieser Studie untersuchen wir die Interaktion und Experience von Generativer KI. Dafür haben wir den Open Data Assistenten entwickelt. Dieser unterstützt BürgerInnen bei der Nutzung der frei verfügbaren Daten der Stadt Freiburg. Zudem werden Sie mit einer frei verfügbaren KI (wie bspw. OpenAI's ChatGPT) interagieren.

Dauer: Die Studie dauert ca. 30 Minuten.

Ablauf:

- Sie werden bezüglich Ihrer demographischen Informationen (Geschlecht, Alter, Beruf, technische Vorkenntnisse) gefragt.
- Sie müssen mit einem Assistenten Ihrer Wahl (vorzugsweise ChatGPT) zwei Texte schreiben.
- Sie beantworten Fragen zu ihrer Wahrnehmung gegenüber dem zweiten Assistenten.
- Sie müssen mit Hilfe des Open Data Assistenten Fragen beantwortet. Dazu erhalten Sie vorab ein Tutorial zur Einarbeitung.
- Sie beantworten Fragen zu ihrer Wahrnehmung gegenüber dem Assistenten und seiner Funktionsweise.

Vergütung: Die Vergütung setzt sich aus einer fixen und einer variablen Komponente zusammen. Es werden fix 4,00 GBP (ca. 4,69 EUR) für die Teilnahme an der Studie ausgezahlt. Zusätzlich wird pro korrekt gelöster Aufgabe ein Bonus von 0,30 GBP (ca. 0,35 EUR) ausgezahlt. Daraus ergibt sich eine mögliche Gesamtvergütung von 6,10 GBP (ca. 7,16 EUR) wenn alle Aufgaben korrekt gelöst werden.

Bevor wir beginnen, möchten wir Ihnen einige Informationen über die Studie geben und Sie über den Datenschutz informieren.

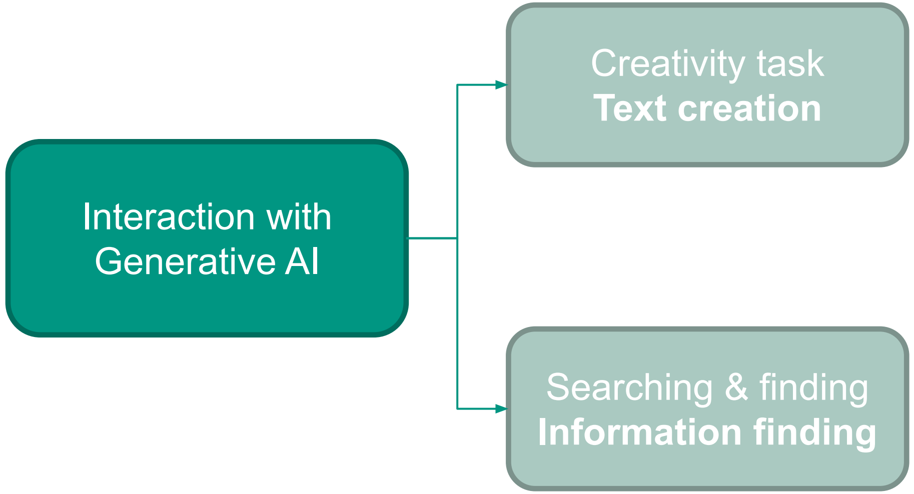
Informationen zum Datenschutz

☐ **Ja, ich habe alle Informationen gelesen und verstanden und möchte an der Studie teilnehmen**
Datenschutzerklärung anzeigen

Image from https://www.flaticon.com/free-icons/ai-assistant created by Freepik - Flaticon

Dataset Tasks

Two tasks for human-AI interaction in dataset:



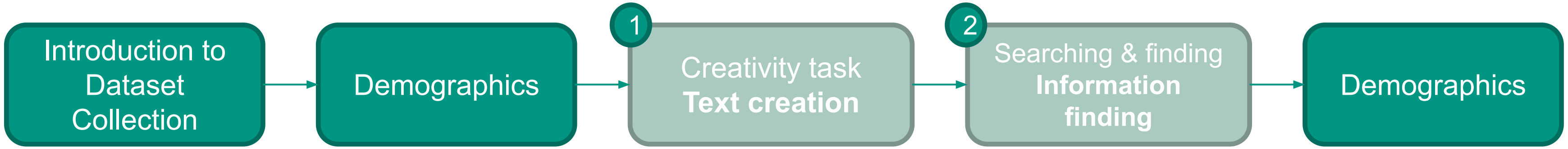
Subjects had to create a text of their choice for a specific task using an generative AI assistant of their choice. They were able to refine the text as much as they want using the generative AI assistant.

OpenAI

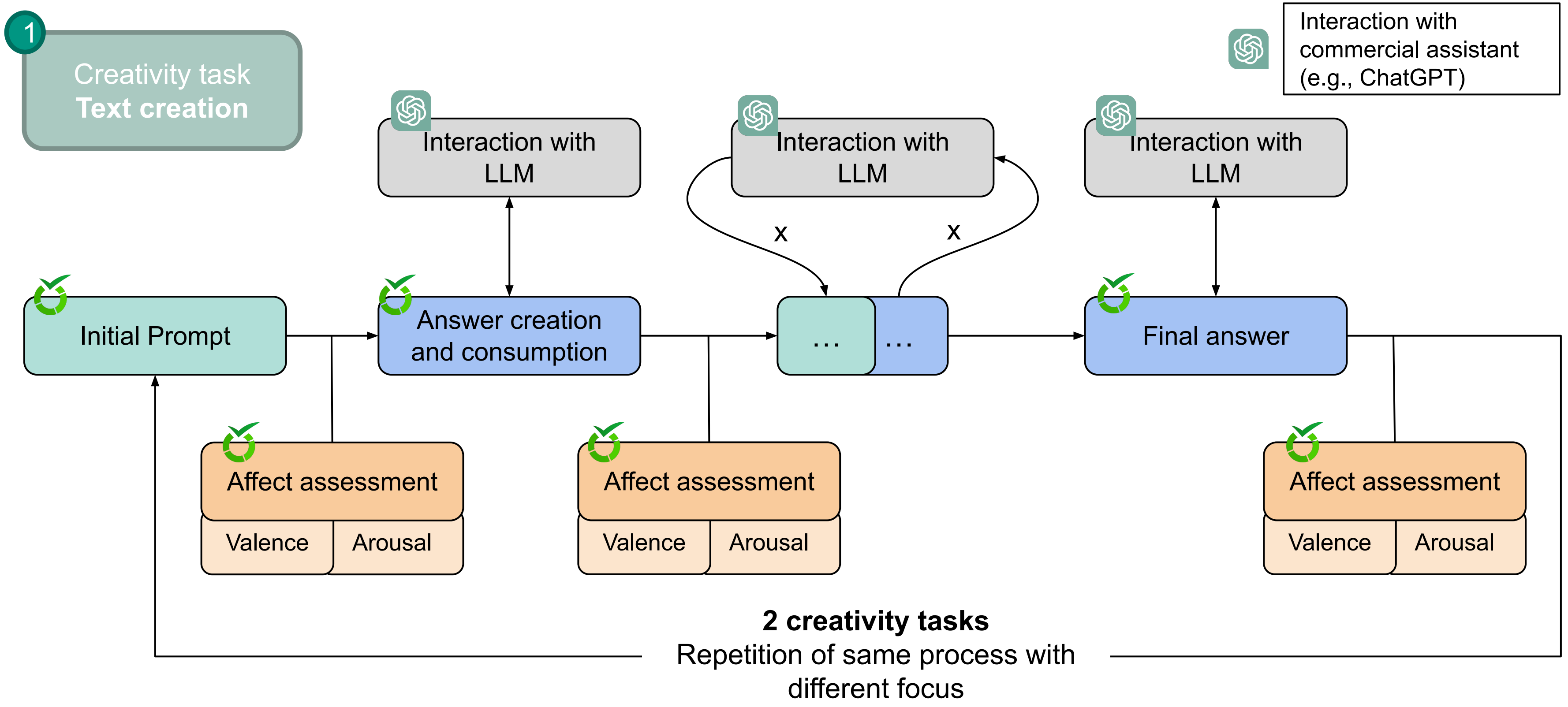
Subjects had to find information about the city of Freiburg like the inhabitants of a suburb or the amount of child care spaces in the neighbourhood. They were using a self-developed assistant called “Open Assistant”.

Image from https://www.flaticon.com/free-icons/ai-assistant created by Freepik - Flaticon

Dataset: Data Collection



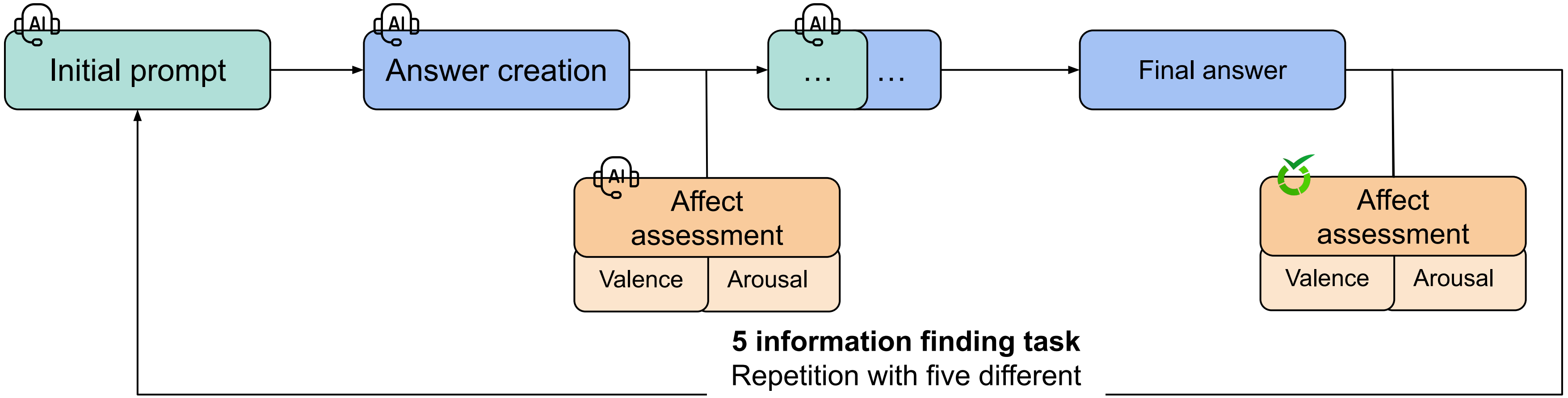
Dataset: Data Collection



2 creativity tasks
Repetition of same process with different focus

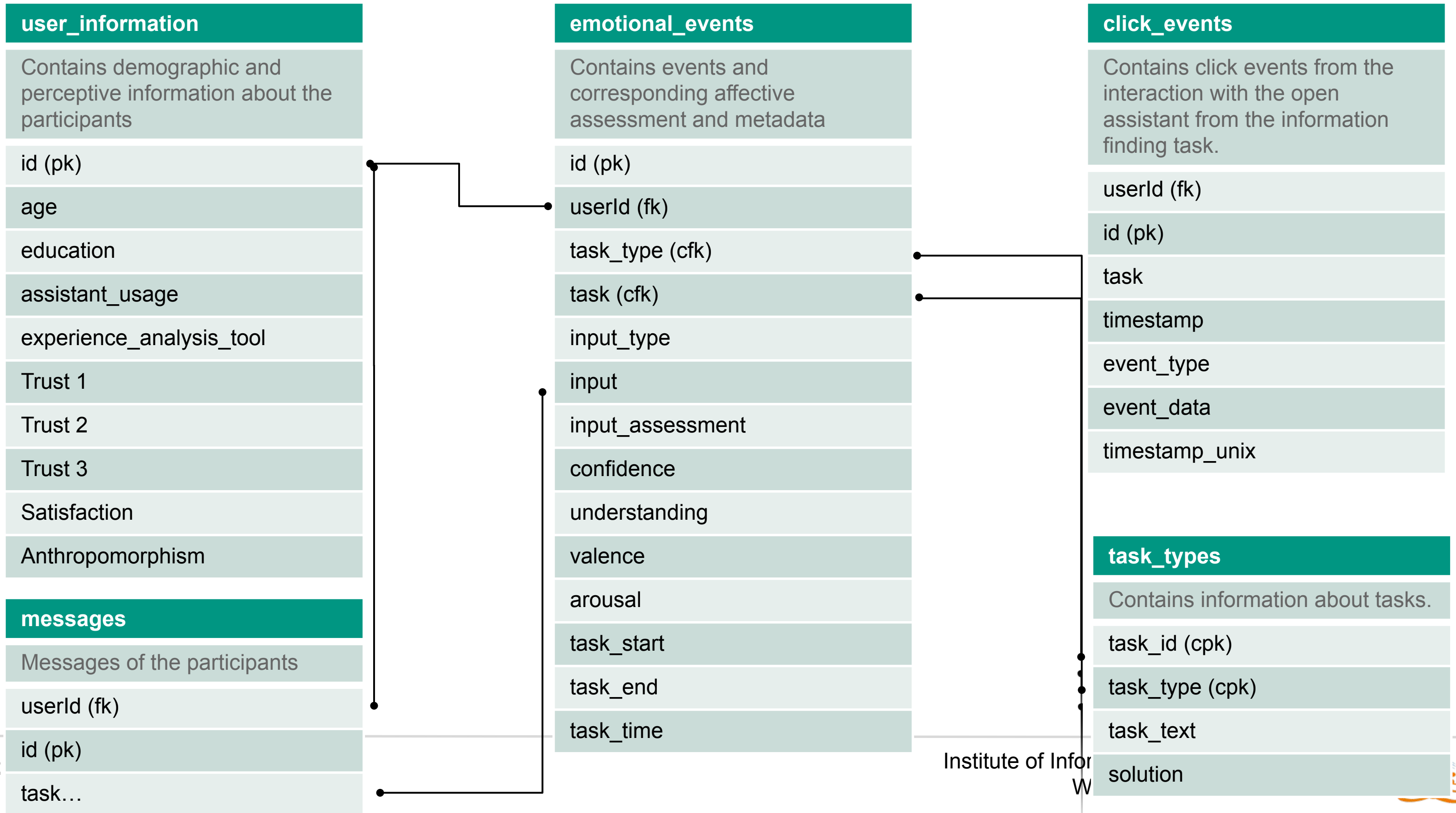
Dataset: Data Collection

2 Searching & finding **Information finding**



5 information finding task
Repetition with five different information finding tasks of different complexity

Data model



user_information

Contains demographic and perceptive information about the participants

id (pk)
age
education
assistant_usage
experience_analysis_tool
Trust 1
Trust 2
Trust 3
Satisfaction
Anthropomorphism

messages

Messages of the participants

userId (fk)
id (pk)
task…

emotional_events

Contains events and corresponding affective assessment and metadata

id (pk)
userId (fk)
task_type (cfk)
task (cfk)
input_type
input
input_assessment
confidence
understanding
valence
arousal
task_start
task_end
task_time

click_events

Contains click events from the interaction with the open assistant from the information finding task.

userId (fk)
id (pk)
task
timestamp
event_type
event_data
timestamp_unix

task_types

Contains information about tasks.

task_id (cpk)
task_type (cpk)
task_text
solution

Data model (2)

user_information

Attribute name	*Description*	*Value type*	*Value range / example*
id (pk)	Identifier of participant	int	112
age	age of participant	int	34
education	level of education of participant	string	middle_school, abitur, bachelor, master, phd
assistant_usage	experience in using generative AI assistants	string	daily, weekly, monthly, yearly
experience_analysis_tool	experience in using data and analytics solutions	int	1 (very low) - 7 (very much)
Trust 1	question about trust level of participant: “Ich glaube, dass generative KI Assistenten meine Fragen ehrlich und transparent beantworten.”	int	1 (Do not agree) - 7 (Highly agree)
Trust 2	question about trust level of participant: “Ich vertraue darauf, dass generative KI Assistenten meine Informationen sicher und vertraulich behandelt.”	int	1 (Do not agree) - 7 (Highly agree)
Trust 3	question about trust level of participant: “Ich habe das Gefühl, dass generative KI Assistenten zuverlässig und konsistent in seinen Antworten ist.”	int	1 (Do not agree) - 7 (Highly agree)
Satisfaction	question about satisfaction with generative AI assistants: “Ich bin insgesamt zufrieden mit der Leistung von generativen KI Assistenten.”	int	1 (Do not agree) - 7 (Highly agree)
Anthropomorphism	question about perceived anthropomorphism of generative AI assistant: “Ich habe das Gefühl, dass generative KI Assistenten menschliche Eigenschaften oder Gefühle haben.”	int	1 (Do not agree) - 7 (Highly agree)

Data model (3)

Karlsruhe Institute of Technology

emotional_events			
Attribute name	*Description*	*Value type*	*Value range / example*
id (pk)	Identifier of event	int	3
userId (fk)	user ID of participant. Foreign key for user table.	int	112
task_type (cfk)	type of task. Foreign key for task table.	string	information_finding, text_creation
task (cfk)	task order	string	daily, weekly, monthly, yearly
input_type	type of input of event such as the prompt of the task, the answer by the llm or the final input. This is dependent on the task	string	prompt, llm_answer, final_output
input	input to the emotional event. Can be the prompt message, the final answer to the task by the participant or the llm answer	string	-
input_assessment	Empty column to provide possibility to assess input. Open for interpretation by seminar participant.	object	-
confidence	confidence in correctnes of personal answer to task. Applies only to task_type information_finding	int	1 (Not at all) - 7 (Absolutely)
understanding	understanding of answer behavior of open assistant. Applies only to task_type information_finding	int	1 (Not at all) - 7 (Absolutely)
valence	valence assessment after event.	int	1 (Very negative) - 6 (Very positive)
arousal	arousal assessment after event.	int	1 (Very low activation) - 6 (Very high activation)
task_start	time when task started. Only present in information_finding task.	int	Format: unix timestamp
task_end	time when task ended. Only present in information_finding task.	int	Format: unix timestamp
task_time	Time for task in seconds.	float	76.76

Data model (4)

Notes for table emotional_events:

- Content of rows depends on task_type
- For information_finding:
  - There are no values for arousal and valence for rows with input_type prompt
  - The entry for input for rows with input_type “final_output” is the answer to the question.
  - The entry for input for rows with input_type “prompt” is the initial prompt.
  - The entry for input for rows with input_type “llm_answer” is the first answer prompt.
- For text_creation:
  - There are no values for confidence and understanding.
  - There is no value for time_start and time_end.

Data model (5)

click_events

Attribute name	*Description*	*Value type*	*Value range / example*
id (pk)	id of individual event	int	112
userId (fk)	Identifier of participant	int	22
task	task number	int	1 - 5
timestamp	timestamp of event in date format	date	-
event_type	experience in using generative AI assistants	string	daily, weekly, monthly, yearly
event_data	experience in using data and analytics solutions	int	1 (very low) - 7 (very much)
timestamp_unix	timestamp of event in unix format	int	-

Data model (6)

task_types			
Attribute name	*Description*	*Value type*	*Value range / example*
task_id (cpk)	Identifier of task	int	112
task_type (cpk)	type of task.	int	34
task_text	task description as presented to participants	string	-
solution	solution to information_finding tasks. Does not apply to text_creation task.	json	-

Data model (7)

messages			
Attribute name	*Description*	*Value type*	*Value range / example*
userId (cpk)	identifier of participant	int	112
task	type of task	int	3
message_type	task description as presented to participants	string	human, agent_finish
timestamp	timestamp of the message in date format	date	-
input	message of the human participant or the open assistant	json/string	-
timestamp_unix	timestamp of the message in unix format	int	-

Notes:

- Messages in this table are only for information finding task using the open assistant.

Open Assistant Introduction

Agenda

Data Storytelling

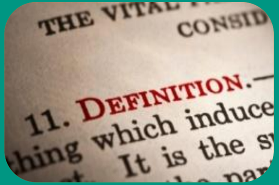
The audience is 22 times more likely to remember a fact when told a story!

Forbes, 2016

Data Storytelling

- Data storytelling is a powerful tool for communicating insights.
- Focus on emotional resonance and connecting with your audience.
- Experiment with different narrative approaches to find what works best for you.
- We’ll look into two specific frameworks that help in crafting compelling data stories:
  - Hero's journey
  - Narrative structure

Data Storytelling



Data storytelling is the ability to effectively communicate insights from a dataset using narratives and visualizations. It can be used to put data insights into context for and inspire action from the target audience.

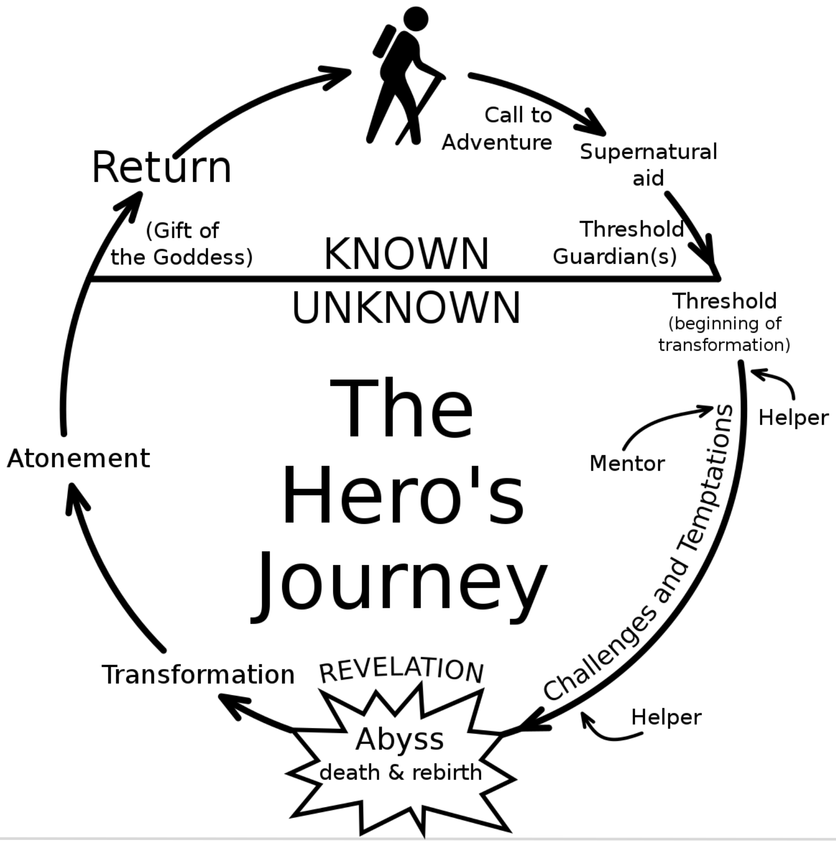
Three components:

1. **Data**: Thorough analysis of accurate, complete data serves as the foundation of a data story
2. **Narrative**: A verbal or written narrative, also called a storyline, is used to communicate insights gleaned from data, the context surrounding it, and the recommended actions
3. **Visualizations**: Visual representations of the data and narrative can be useful for communicating its story clearly and memorably

https://online.hbs.edu/blog/post/data-storytelling

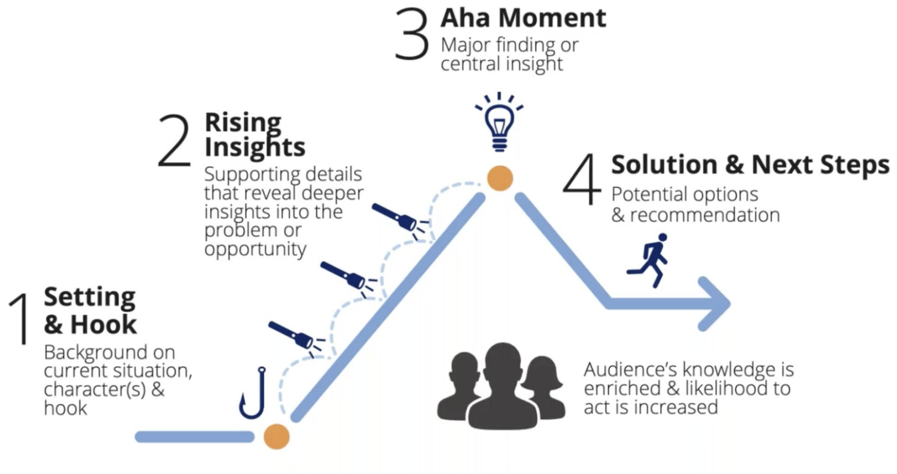
Hero’s Journey

- **Ordinary World:** Introduce the current situation or status quo.
- **Call to Adventure:** Present a problem or challenge that needs to be addressed.
- **Challenges & Trials:** Explore the obstacles faced and the data-driven insights that guide the way.
- **Transformation:** Reveal the key insight or solution that changes the narrative.
- **Return with the Elixir:** Share the impact or outcome of the transformation, highlighting the value of the data-driven decision.



Narrative Structure

- **Hook:** A captivating opening that grabs attention.
- **Rising Insight:** Building context and introducing the problem.
- **Aha Moment:** The turning point where the key insight is revealed.
- **Resolution:** The outcome and impact of the data-driven decision.
- **Call to Action:** (Optional) Encourage the audience to take action based on the insights.



3 Aha Moment
Major finding or central insight
2 Rising Insights
Supporting details that reveal deeper insights into the problem or opportunity
4 Solution & Next Steps
Potential options & recommendation
1 Setting & Hook
Background on current situation, character(s) & hook
Audience's knowledge is enriched & likelihood to act is increased

Situation-Problem-Solution-Next Steps (SPSN) Framework

Slide 1: Situation	Slide 2: Problem
Describe the current state to your audience What is the status quo you're trying to change?	Picture the problem. What's the issue with the situation? What is the pain you're trying to solve?
Slide 3: Solution	**Slide 4: Next Steps**
Present the solution. How do you solve the problem? How do you cure the pain?	You convinced the audience. What are the next steps you need to take? Which actions need to be taken?

https://towardsdatascience.com/storytelling-for-data-scientists-317c2723aa31

SPSN Framework – Example

Slide 1: Situation

- Consumers leave data traces when browsing our website
- We store and collect data for every user
- We don't offer personalized recommendations

Slide 2: Problem

- Consumers expect recommendations, because our competitors offer it
- We’re missing out on potential revenue
- Consumers switch to our competitors for product browsing

Slide 3: Solution

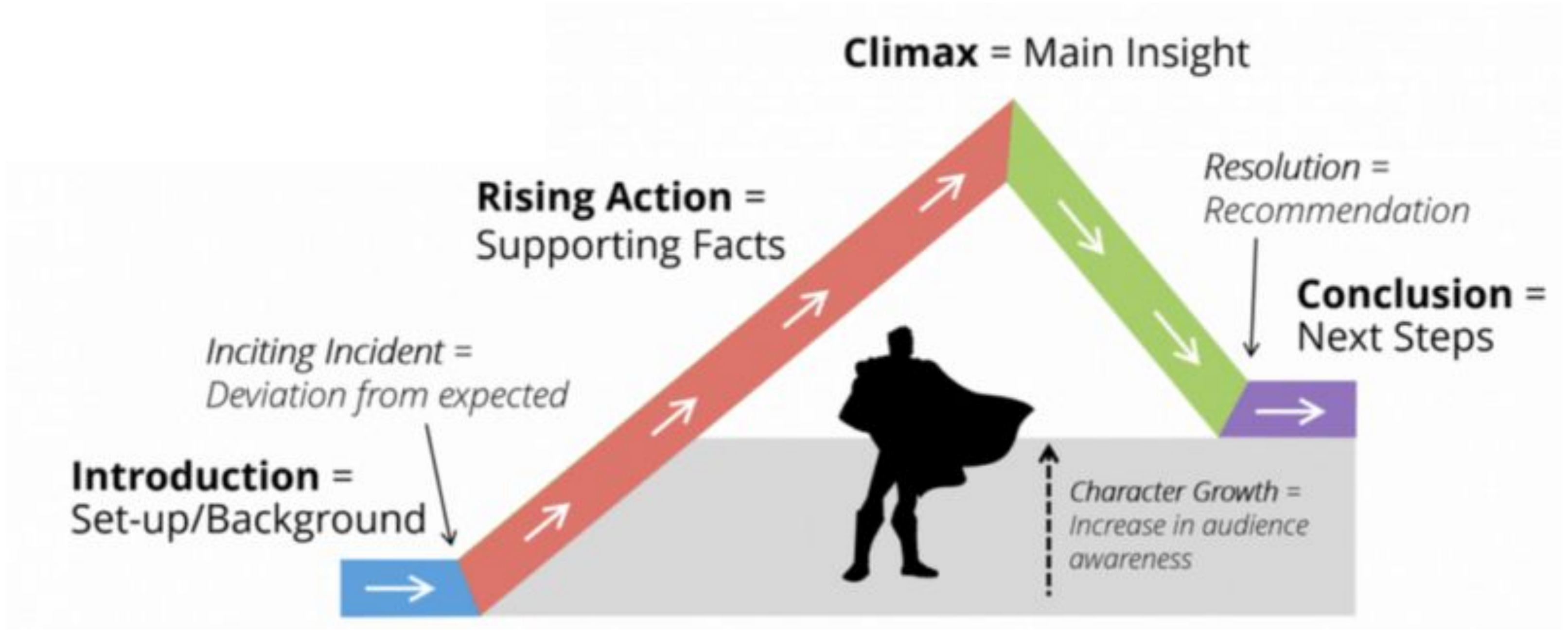
- Create personalized item recommender
- Train state-of-the-art recommendation algorithms
- Roll out recommender to all users

Slide 4: Next Steps

- Create recommendation project team of 6 data engineers & scientists and a Product Owner
- Invest 100k in cloud resources
- A/B tested recommender will be ready for rollout in 6 months

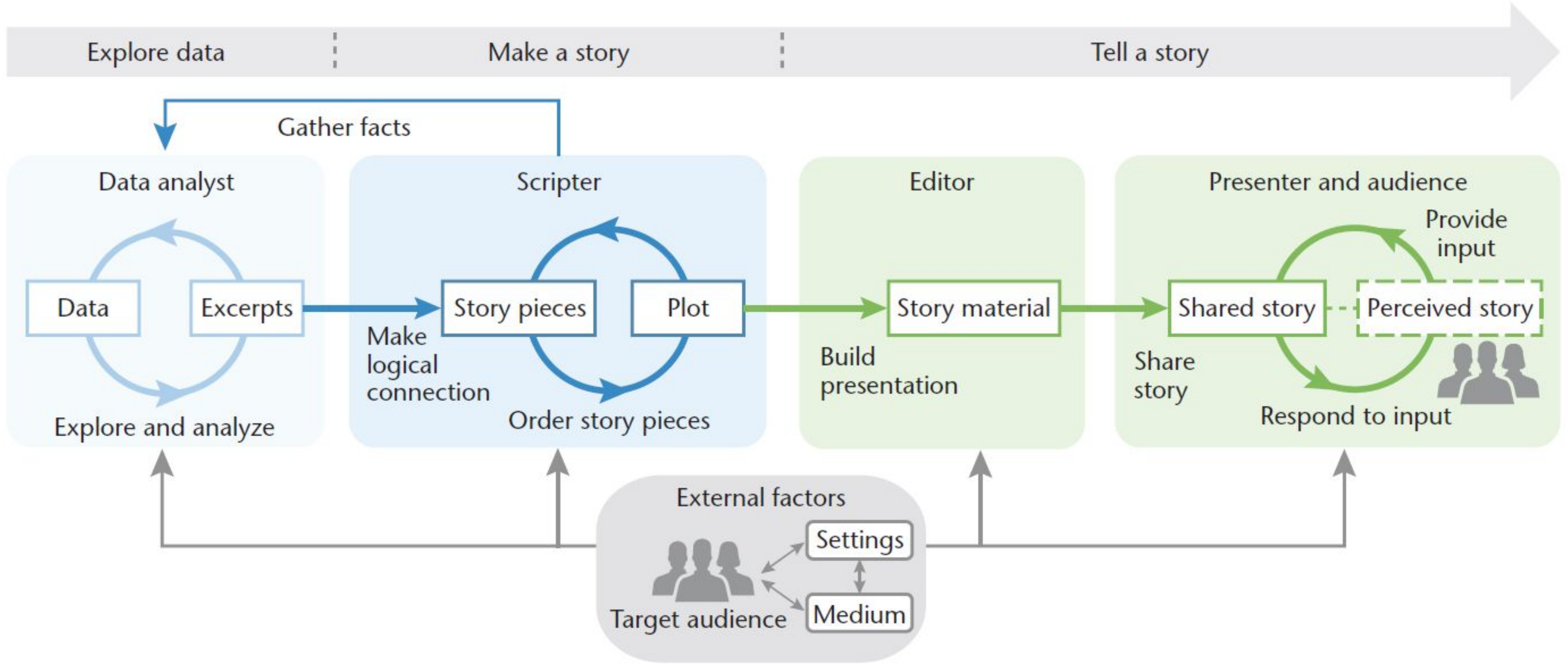
https://towardsdatascience.com/storytelling-for-data-scientists-317c2723aa31

Freytag's Pyramid


Climax = Main Insight
Rising Action = Supporting Facts
Resolution = Recommendation
Conclusion = Next Steps
Inciting Incident = Deviation from expected
Introduction = Set-up/Background
Character Growth = Increase in audience awareness

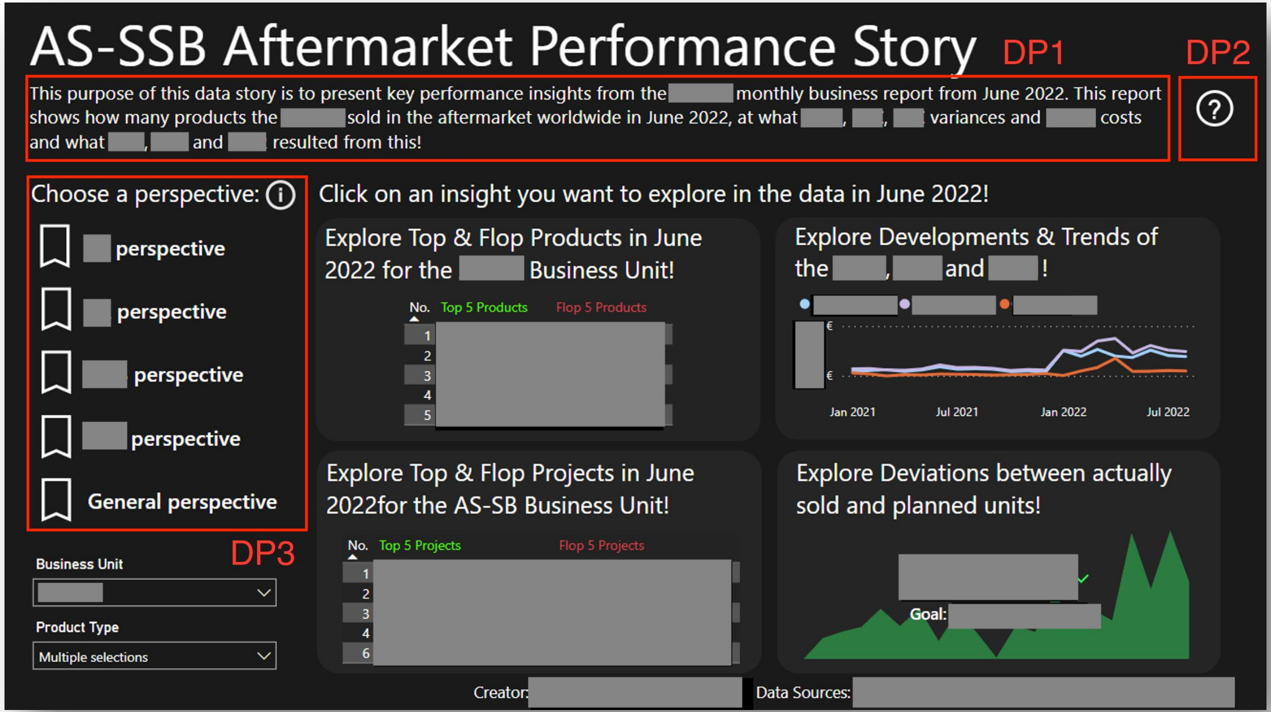
https://www.blastanalytics.com/blog/persuasive-storytelling-with-data-visualization

Data Storytelling Process



Lee et al. (2015)

Research Example: Integrating Data Stories in a Dashboard



AS-SSB Aftermarket Performance Story DP1 DP2

This purpose of this data story is to present key performance insights from the [redacted] monthly business report from June 2022. This report shows how many products the [redacted] sold in the aftermarket worldwide in June 2022, at what [redacted], [redacted], [redacted] variances and [redacted] costs and what [redacted], [redacted] and [redacted] resulted from this!

Choose a perspective:

- [redacted] perspective
- [redacted] perspective
- [redacted] perspective
- [redacted] perspective
- General perspective

DP3

Business Unit: [redacted]

Product Type: Multiple selections

Click on an insight you want to explore in the data in June 2022!

Explore Top & Flop Products in June 2022 for the [redacted] Business Unit!

No.	Top 5 Products	Flop 5 Products
1	[redacted]	
2		
3		
4		
5		

Explore Developments & Trends of the [redacted], [redacted] and [redacted]!

Explore Top & Flop Projects in June 2022for the AS-SB Business Unit!

No.	Top 5 Projects	Flop 5 Projects
1	[redacted]	
2		
3		
4		
6		

Explore Deviations between actually sold and planned units!

Creator: [redacted] Data Sources: [redacted]

https://aisel.aisnet.org/ecis2023_rp/327/

Models and Packages

Matplotlib

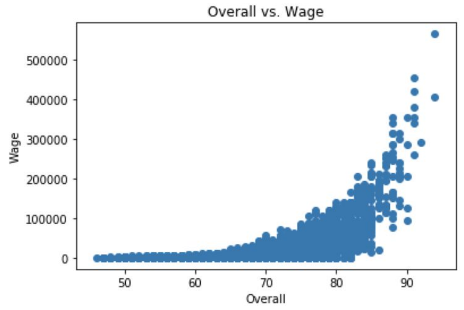
- Matplotlib is a library in Python that enables users to generate visualizations like histograms, scatter plots, bar charts, pie charts and much more.

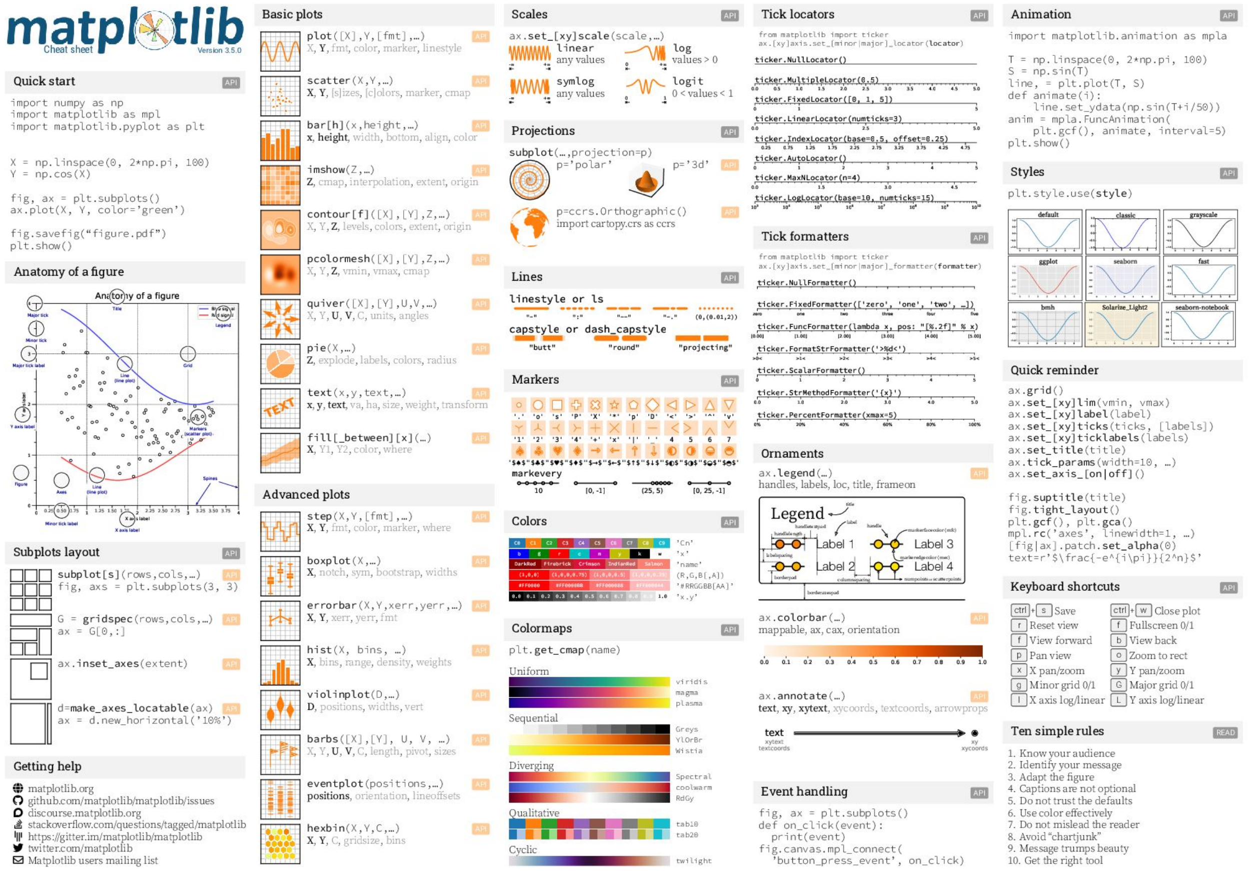
- Methods for different chart types:
  - hist() -> histogram
  - scatter() -> scatter plot
  - bar() -> bar chart
  - … see documentation

- Labels:
  - title()
  - ylabel()
  - xlabel()
  - axis()

- Usage

```
import matplotlib.pyplot as plt
plt.scatter(df['Overall'], df['wage_euro']) plt.title('Overall vs. Wage')
plt.ylabel('Wage')
plt.xlabel('Overall')
plt.show()
```





matplotlib Cheat sheet Version 3.5.0

Quick start

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


X = np.linspace(0, 2*np.pi, 100)
Y = np.cos(X)

fig, ax = plt.subplots()
ax.plot(X, Y, color='green')

fig.savefig("figure.pdf")
plt.show()
```

Anatomy of a figure

Anatomy of a figure
Title
Major tick
Minor tick
Major tick label
Grid
Legend
Line (line plot)
Y axis label
Markers (scatter plot)
Figure
Axes
Line (line plot)
Spines
Minor tick label
X axis label

Subplots layout

```
subplot[s](rows,cols,…)
fig, axs = plt.subplots(3, 3)

G = gridspec(rows,cols,…)
ax = G[0,:]

ax.inset_axes(extent)

d=make_axes_locatable(ax)
ax = d.new_horizontal('10%')
```

Getting help

- matplotlib.org
- github.com/matplotlib/matplotlib/issues
- discourse.matplotlib.org
- stackoverflow.com/questions/tagged/matplotlib
- https://gitter.im/matplotlib/matplotlib
- twitter.com/matplotlib
- Matplotlib users mailing list

Basic plots

- `plot([X],Y,[fmt],…)` X, Y, fmt, color, marker, linestyle
- `scatter(X,Y,…)` X, Y, [s]izes, [c]olors, marker, cmap
- `bar[h](x,height,…)` x, height, width, bottom, align, color
- `imshow(Z,…)` Z, cmap, interpolation, extent, origin
- `contour[f]([X],[Y],Z,…)` X, Y, Z, levels, colors, extent, origin
- `pcolormesh([X],[Y],Z,…)` X, Y, Z, vmin, vmax, cmap
- `quiver([X],[Y],U,V,…)` X, Y, U, V, C, units, angles
- `pie(X,…)` Z, explode, labels, colors, radius
- `text(x,y,text,…)` x, y, text, va, ha, size, weight, transform
- `fill[_between][x](…)` X, Y1, Y2, color, where

Advanced plots

- `step(X,Y,[fmt],…)` X, Y, fmt, color, marker, where
- `boxplot(X,…)` X, notch, sym, bootstrap, widths
- `errorbar(X,Y,xerr,yerr,…)` X, Y, xerr, yerr, fmt
- `hist(X, bins, …)` X, bins, range, density, weights
- `violinplot(D,…)` D, positions, widths, vert
- `barbs([X],[Y], U, V, …)` X, Y, U, V, C, length, pivot, sizes
- `eventplot(positions,…)` positions, orientation, lineoffsets
- `hexbin(X,Y,C,…)` X, Y, C, gridsize, bins

Scales

`ax.set_[xy]scale(scale,…)`

`linear` any values	`log` values > 0
`symlog` any values	`logit` 0 < values < 1

Projections

`subplot(…,projection=p)`

p='polar' p='3d'

p=ccrs.Orthographic()
import cartopy.crs as ccrs

Lines

`linestyle or ls`

"-" ":" "--" "-." (0,(0.01,2))

`capstyle or dash_capstyle`

"butt" "round" "projecting"

Markers

'.' 'o' 's' 'P' 'X' '*' 'p' 'D' '<' '>' '^' 'v'
'1' '2' '3' '4' '+' 'x' '|' '_' 4 5 6 7
'$♠$' '$♣$' '$♥$' '$♦$' '$→$' '$←$' '$↑$' '$↓$' '$◐$' '$◑$' '$◒$' '$◓$'

`markevery`

10 [0, -1] (25, 5) [0, 25, -1]

Colors

C0 C1 C2 C3 C4 C5 C6 C7 C8 C9	'Cn'
b g r c m y k w	'x'
DarkRed Firebrick Crimson IndianRed Salmon	'name'
(1,0,0) (1,0,0,0.75) (1,0,0,0.5) (1,0,0,0.25)	(R,G,B[,A])
#FF0000 #FF0000BB #FF000088 #FF000044	'#RRGGBB[AA]'
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0	'x.y'

Colormaps

`plt.get_cmap(name)`

Uniform: viridis, magma, plasma
Sequential: Greys, YlOrBr, Wistia
Diverging: Spectral, coolwarm, RdGy
Qualitative: tab10, tab20
Cyclic: twilight

Tick locators

```
from matplotlib import ticker
ax.[xy]axis.set_[minor|major]_locator(locator)
```

```
ticker.NullLocator()
ticker.MultipleLocator(0.5)
ticker.FixedLocator([0, 1, 5])
ticker.LinearLocator(numticks=3)
ticker.IndexLocator(base=0.5, offset=0.25)
ticker.AutoLocator()
ticker.MaxNLocator(n=4)
ticker.LogLocator(base=10, numticks=15)
```

Tick formatters

```
from matplotlib import ticker
ax.[xy]axis.set_[minor|major]_formatter(formatter)
```

```
ticker.NullFormatter()
ticker.FixedFormatter(['zero', 'one', 'two', …])
ticker.FuncFormatter(lambda x, pos: "[%.2f]" % x)
ticker.FormatStrFormatter('>%d<')
ticker.ScalarFormatter()
ticker.StrMethodFormatter('{x}')
ticker.PercentFormatter(xmax=5)
```

Ornaments

`ax.legend(…)`
handles, labels, loc, title, frameon

`ax.colorbar(…)`
mappable, ax, cax, orientation

`ax.annotate(…)`
text, xy, xytext, xycoords, textcoords, arrowprops

Event handling

```
fig, ax = plt.subplots()
def on_click(event):
  print(event)
fig.canvas.mpl_connect(
  'button_press_event', on_click)
```

Animation

```
import matplotlib.animation as mpla

T = np.linspace(0, 2*np.pi, 100)
S = np.sin(T)
line, = plt.plot(T, S)
def animate(i):
    line.set_ydata(np.sin(T+i/50))
anim = mpla.FuncAnimation(
    plt.gcf(), animate, interval=5)
plt.show()
```

Styles

`plt.style.use(style)`

default, classic, grayscale, ggplot, seaborn, fast, bmh, Solarize_Light2, seaborn-notebook

Quick reminder

```
ax.grid()
ax.set_[xy]lim(vmin, vmax)
ax.set_[xy]label(label)
ax.set_[xy]ticks(ticks, [labels])
ax.set_[xy]ticklabels(labels)
ax.set_title(title)
ax.tick_params(width=10, …)
ax.set_axis_[on|off]()

fig.suptitle(title)
fig.tight_layout()
plt.gcf(), plt.gca()
mpl.rc('axes', linewidth=1, …)
[fig|ax].patch.set_alpha(0)
text=r'$\frac{-e^{i\pi}}{2^n}$'
```

Keyboard shortcuts

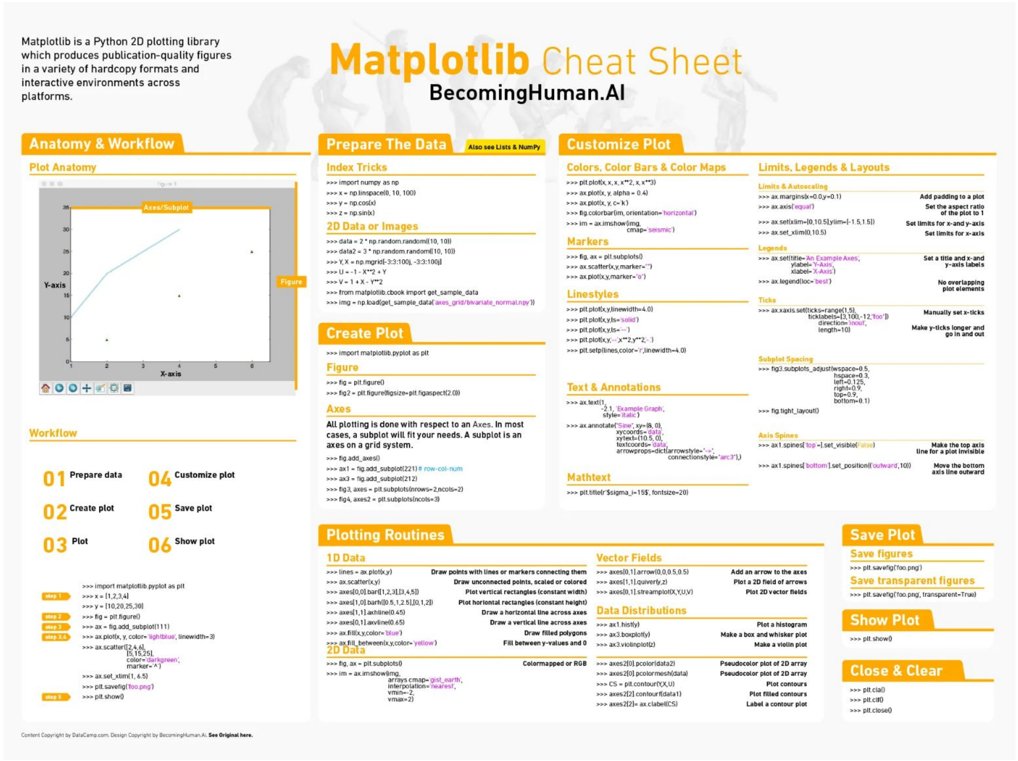
ctrl + s Save	ctrl + w Close plot
r Reset view	f Fullscreen 0/1
f View forward	b View back
p Pan view	o Zoom to rect
x X pan/zoom	y Y pan/zoom
g Minor grid 0/1	G Major grid 0/1
l X axis log/linear	L Y axis log/linear

Ten simple rules

1. Know your audience
2. Identify your message
3. Adapt the figure
4. Captions are not optional
5. Do not trust the defaults
6. Use color effectively
7. Do not mislead the reader
8. Avoid "chartjunk"
9. Message trumps beauty
10. Get the right tool

https://matplotlib.org/cheatsheets/_images/cheatsheets-1.png

Matplotlib Cheat Sheet



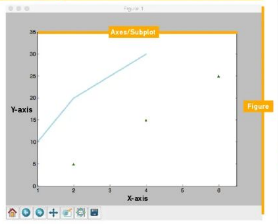
Matplotlib is a Python 2D plotting library which produces publication-quality figures in a variety of hardcopy formats and interactive environments across platforms.

Matplotlib Cheat Sheet

BecomingHuman.AI

Anatomy & Workflow

Plot Anatomy



Workflow

01 Prepare data
02 Create plot
03 Plot
04 Customize plot
05 Save plot
06 Show plot

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

Prepare The Data

Also see Lists & NumPy

Index Tricks

```
>>> 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'))
```

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)
```

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"),)
```

Mathtext

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

Limits, Legends & Layouts

Limits & Autoscaling

```
>>> ax.margins(x=0.0,y=0.1)                      Add padding to a plot
>>> ax.axis('equal')                             Set the aspect ratio
                                                       of the plot to 1
>>> ax.set(xlim=[0,10.5],ylim=[-1.5,1.5])   Set limits for x-and y-axis
>>> ax.set_xlim(0,10.5)                          Set limits for x-axis
```

Legends

```
>>> ax.set(title='An Example Axes',              Set a title and x-and
           ylabel='Y-Axis',                          y-axis labels
           xlabel='X-Axis')
>>> ax.legend(loc='best')                          No overlapping
                                                     plot elements
```

Ticks

```
>>> ax.xaxis.set(ticks=range(1,5),               Manually set x-ticks
                 ticklabels=[3,100,-12,'foo'])
                   direction='inout',         Make y-ticks longer and
                   length=10)                          go in and out
```

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)          Make the top axis
                                             line for a plot invisible
>>> ax1.spines['bottom'].set_position(('outward',10))  Move the bottom
                                                  axis line outward
```

Plotting Routines

1D Data

```
>>> lines = ax.plot(x,y)          Draw points with lines or markers connecting them
>>> ax.scatter(x,y)                    Draw unconnected points, scaled or colored
>>> axes[0,0].bar([1,2,3],[3,4,5])     Plot vertical rectangles (constant width)
>>> axes[1,0].barh([0.5,1,2.5],[0,1,2]) Plot horiontal rectangles (constant height)
>>> axes[1,1].axhline(0.45)                 Draw a horizontal line across axes
>>> axes[0,1].axvline(0.65)                   Draw a vertical line across axes
>>> ax.fill(x,y,color='blue')                             Draw filled polygons
>>> ax.fill_between(x,y,color='yellow')             Fill between y-values and 0
```

2D Data

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

Vector Fields

```
>>> axes[0,1].arrow(0,0,0.5,0.5)             Add an arrow to the axes
>>> axes[1,1].quiver(y,z)                   Plot a 2D field of arrows
>>> axes[0,1].streamplot(X,Y,U,V)              Plot 2D vector fields
```

Data Distributions

```
>>> ax1.hist(y)                                      Plot a histogram
>>> ax3.boxplot(y)                       Make a box and whisker plot
>>> ax3.violinplot(z)                              Make a violin plot
```

```
>>> axes2[0].pcolor(data2)               Pseudocolor plot of 2D array
>>> axes2[0].pcolormesh(data)            Pseudocolor plot of 2D array
>>> CS = plt.contour(Y,X,U)                             Plot contours
>>> axes2[2].contourf(data1)                     Plot filled contours
>>> axes2[2]= ax.clabel(CS)                     Label a contour plot
```

Save Plot

Save figures

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

Save transparent figures

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

Show Plot

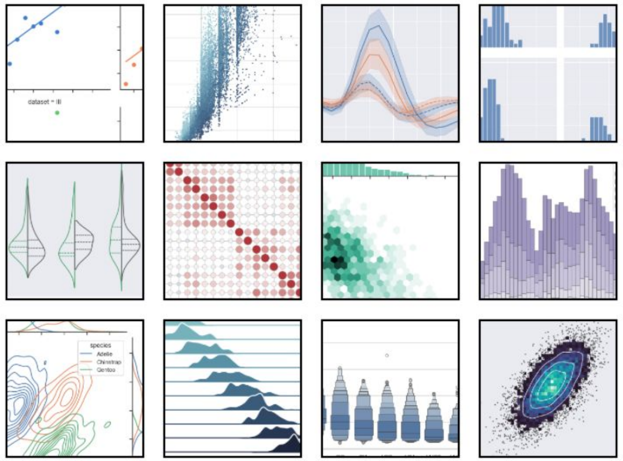
```
>>> plt.show()
```

Close & Clear

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

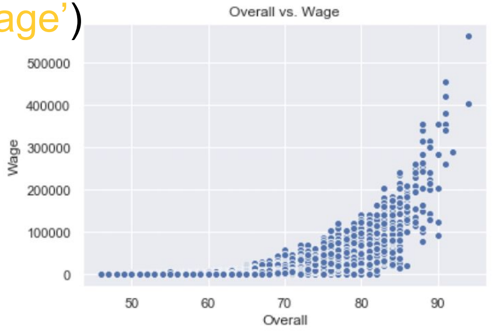
Seaborn

- Seaborn is a visualization library that is built on top of Matplotlib. It provides data visualizations that are typically more aesthetic and statistically sophisticated.



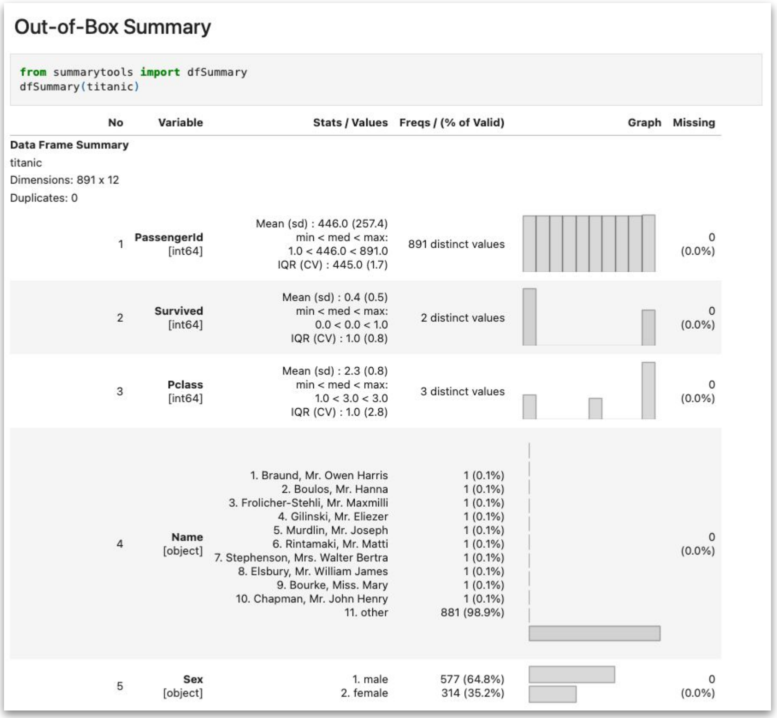
- Usage:

```
import matplotlib.pyplot as plt
import seaborn as sns
sns.scatterplot(df['Overall'], df['wage_euro'])
plt.title('Overall vs. Wage')
plt.ylabel('Wage')
plt.xlabel('Overall')
plt.show()
```



Summarytools

- **Automated Exploratory Data Analysis (EDA)** in Jupyter Notebook:
  - Streamlines the EDA process for faster insights.
  - Works directly within your Jupyter Notebook environment.
- Comprehensive **Data Frame Summaries**:
  - **Descriptive Statistics:** Calculates mean, interquartile range (IQR), and other essential measures.
  - **Frequency Distributions:** Shows the frequency of different values in your data.
  - **Missing Data Analysis:** Identifies and quantifies missing values in each column.
- **Data Visualization:** Generates graphs and tables to visualize distributions, patterns, and relationships within your data.



Out-of-Box Summary

```
from summarytools import dfSummary
dfSummary(titanic)
```

Data Frame Summary
titanic
Dimensions: 891 x 12
Duplicates: 0

No	Variable	Stats / Values	Freqs / (% of Valid)	Graph	Missing
1	**PassengerId** [int64]	Mean (sd) : 446.0 (257.4) min < med < max: 1.0 < 446.0 < 891.0 IQR (CV) : 445.0 (1.7)	891 distinct values		0 (0.0%)
2	**Survived** [int64]	Mean (sd) : 0.4 (0.5) min < med < max: 0.0 < 0.0 < 1.0 IQR (CV) : 1.0 (0.8)	2 distinct values		0 (0.0%)
3	**Pclass** [int64]	Mean (sd) : 2.3 (0.8) min < med < max: 1.0 < 3.0 < 3.0 IQR (CV) : 1.0 (2.8)	3 distinct values		0 (0.0%)
4	**Name** [object]	1. Braund, Mr. Owen Harris 2. Boulos, Mr. Hanna 3. Frolicher-Stehli, Mr. Maxmilli 4. Gilinski, Mr. Eliezer 5. Murdlin, Mr. Joseph 6. Rintamaki, Mr. Matti 7. Stephenson, Mrs. Walter Bertra 8. Elsbury, Mr. William James 9. Bourke, Miss. Mary 10. Chapman, Mr. John Henry 11. other	1 (0.1%) 1 (0.1%) 1 (0.1%) 1 (0.1%) 1 (0.1%) 1 (0.1%) 1 (0.1%) 1 (0.1%) 1 (0.1%) 1 (0.1%) 881 (98.9%)		0 (0.0%)
5	**Sex** [object]	1. male 2. female	577 (64.8%) 314 (35.2%)		0 (0.0%)

Questions, Comments, Observations