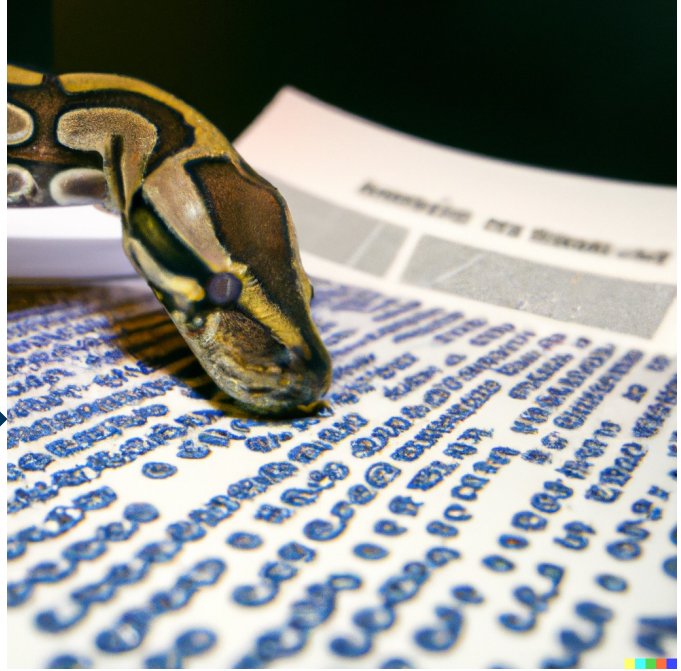
elixir GERMANY

AI generated



RUB

deNBI
GERMAN NETWORK FOR BIOINFORMATICS INFRASTRUCTURE

RUHR-UNIVERSITÄT BOCHUM

DATA EXPLORATION WITH PYTHON

The young researcher's toolkit

CUBiMed.RUB
CORE UNIT BIOINFORMATICS
MEDIZINISCHE FAKULTÄT
DER RUHR-UNIVERSITÄT BOCHUM

mpc
MEDIZINISCHESPROTEOMCENTER

Why Python?

- **Easy to use**
- **Common in scientific setting**
- **Lot of 3rd party support**
- **Works on Linux, MacOS and Windows**
- **General purpose**
  - Automating (Ansible)
  - Web application (streamlit)
  - Machine and Deep Learning (Keras & PyTorch)
  - Data analysis & visualization (Pandas, Plotly)
  - Game development (Pygame)

Similar languages

- **R**
  - Not a general purpose language
  - Focused more on scientific applications like statistics, algebra, machine learning
- **Ruby**
  - Not much support for scientific libraries

Source file

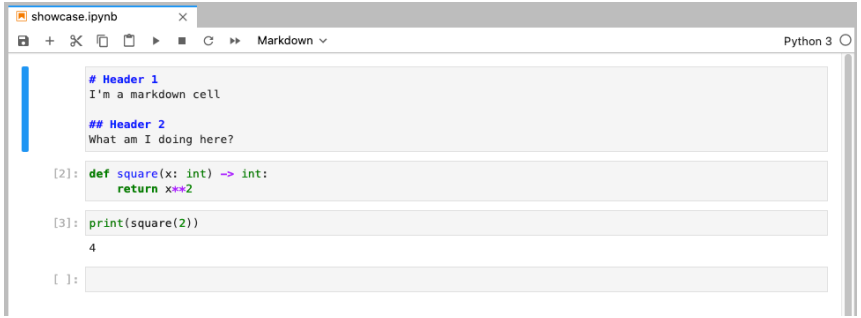
- **Normally your code lives in Python files ending with `.py`**
- **Which are executed using the command:**
  python do_what_i_say.py
- **Python starts to execute with the first line and goes through the file line by line**

```python
# std imports
import argparse
import importlib
from pathlib import Path
from types import ModuleType

# internal imports
import sdrf_convert.config_generators as config_generators
import sdrf_convert.sdrf_generators as sdrf_generators

def import_submodules(sub_module: ModuleType, python_file_wildcard: str):
    """Import the submodules of a module dynamically
    """
    for converter_path in Path(sub_module.__file__).parent.glob(python_file_wildcard):
        importlib.import_module(f"{sub_module.__name__}.{converter_path.stem}")

# Import all subclassed generators automatically so they are available in Abstract*Generator.__subclasses__()
# and can be automatically added to the CLI
import_submodules(config_generators, '*_config_generator.py')
import_submodules(sdrf_generators, '*_sdrf_generator.py')


You, yesterday | 2 authors (You and others)
class CommandLineInterface:
    """
```

Jupyter Notebooks

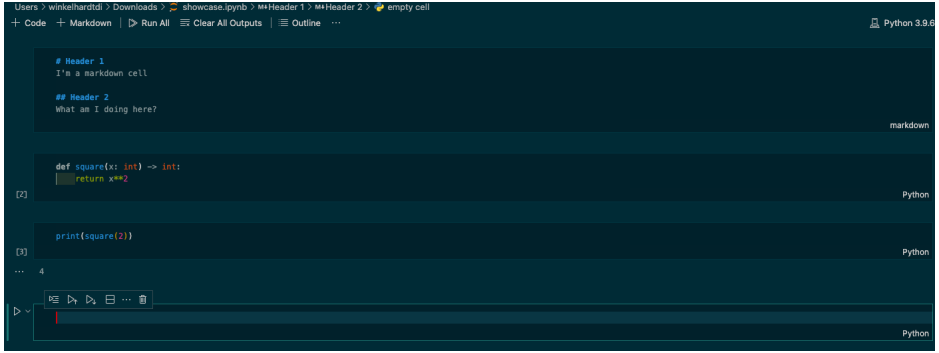
- **Jupyer Notebooks are a <u>frontend</u> for combining code and text blocks**
  - They can display results immediatly (plots, tables, text)
  - Export to PDF, HTML, TeX, ...
  - Runs
    - Visual Studio Code (local)
    - Galaxy (Browser)
      - Which basically starts Jupyter Lab (self hosted / browser)
    - Google Colab

Jupyter Notebooks



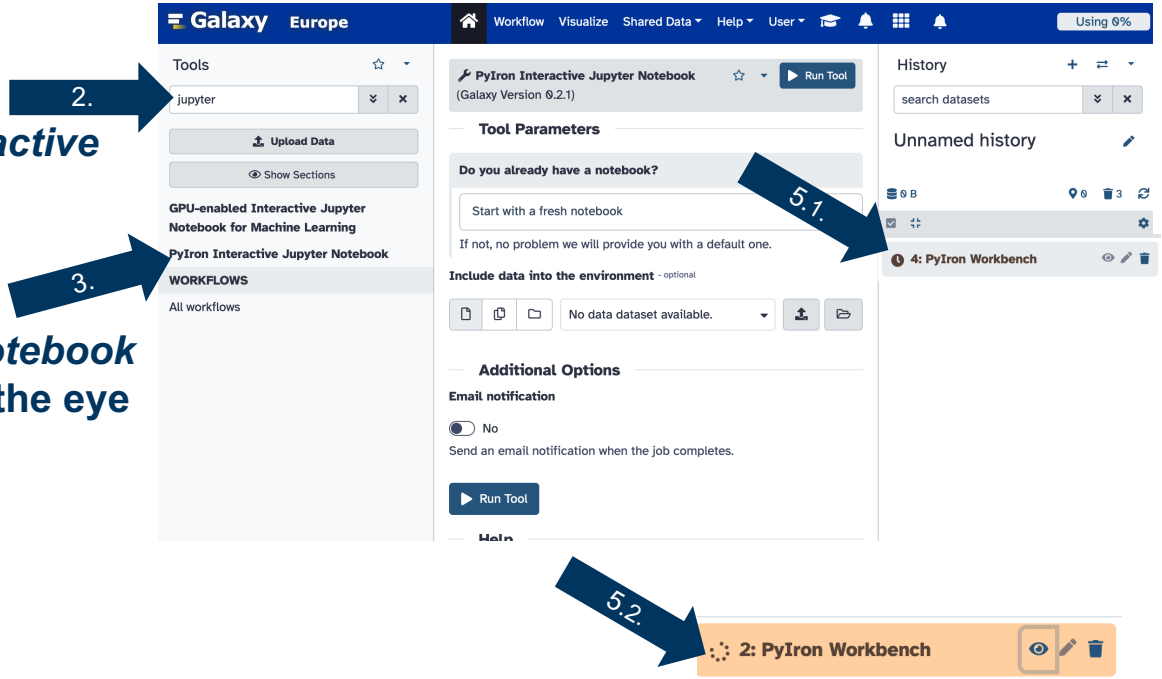
Jupyter Lab (browser)

Visual Studio Code (local)



Jupyter Notebooks in Galaxy

1. **Go to usegalaxy.eu and log in**
2. **Left side: Search for *jupyter***
3. **Left side: Click on *PyIron Interactive Jupyter Notebook***
4. **Middle: Click *Run Job***
5. **Right side: Wait until *PyIron Notebook* becomes orange then click on the eye symbol**



Primitive data, standard operators and comments

Primitive datatypes

Type	Description	Examples
Integer		... -2, -1, -0, -1, -2 ...
Float	Numbers of various **precision** (rounding errors included)	..., -0.2, 0.0001, 0.0, 0.3, 0.333335, ...
Boolean	Value about **truthiness**	True & False

Primitive data, standard operators and comments

Standard operator

Operator	Usage	Examples
Assignment	=	i = 1
Equals	==	1 == 1
Not equals	!=	1 != 2

Primitive data, standard operators and comments

Numeric standard operator

Operator	Usage	Examples
Addition	+	1 + 1 == 2
Substraction	-	1 - 1 == 0
Multiplication	*	2 * 3 == 6
Division	/	6 / 3 == 2
Modulo	%	5 % 2 == 1
Exponential	**	5 ** 3 == 125

Primitive data, standard operators and comments

Boolean standard operator

Operator	Usage	Examples
and	and	True and True == True True and False == False False and True == False False and False == False
or	or	True and True == True True and False == True False and True == True False and False == False
negation	not	not True == False not False == True not not True == True not not not not not True == False

Syntax

```
# This is a comment, it‘s start with ‚#‘ and is not evaluated by Python
“““
This is also a comment but spanning over
multiple lines
“““
# Your future you will appreciate to see some of them in your code ... Believe me


x = 1 # This is an assignment
y = 3 # This is also an assignment
x + y # This is an operation, adds y to x
z = x + y # Option and assignment
print(“z is:”, z)
=> z is: 4
```

Exercise 1

https://l.rub.de/74423d59

Functions

- **Represents a calculation / method / process / user interaction ...**
- **Reusable**

argument list, arguments also in lower_snake_case

function name in lower_snake_case

type of return (optional)

keyword

```
def i_am_a_function(arg_1: int, arg_2: float = 0.1) -> return_type:
    # something is happening here
    #
    ...
```

function body

4 spaces, not 1 tab, not 2 spaces

optional default value

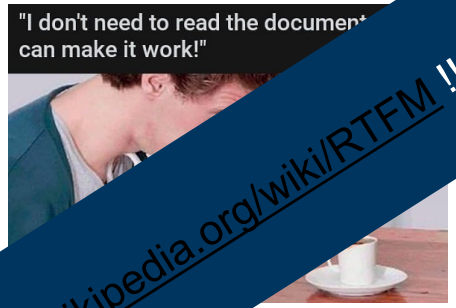
Functions

- **Built-in functions**
  - `pow(base, exp, mod=None)` - Returns *base* to the power of *exp*
  - `round(number, ndigits)` - Rounds *number* to *ndigits*, if *ndigits* is *None* (default) it rounds to nearest integer
  - `print(*objects, sep=' ', end='\n', file=None, flush=False)` - Prints the given *objects*, using the provided separator and end character. *file* & *flush* is for advanced purpose.

  - More: https://docs.python.org/3/library/functions.html

RTFM – Read the ******* manual (docs)

You can find the Python documentation here: https://docs.p[illegible]n.org/3/



!!! https://en.wikipedia.org/wiki/RTFM !!!

Function usage

```
print(“I am“, “Groot“)
=> “I am Groot“
print(“I am“, “Groot“, sep=“__“)
=> “I am__Groot“
round(3.14159265359)
=> 3
round(3.14159265359, 2)
=> 3.14
pow(2, 3)
=> 8

# save the result of a function
res = pow(2,3)
```

Imports

- **Python contain more functionality then the built-in function and additional constants like the number Pi**
- **Make them accessible via *imports***

```
import math
import random

print(math.pi)
=> 3.141592653589793
math.ceil(math.pi)
=> 4
random.randint(1, 10)
=> ???
```

Complex data types & data collections

Type	Description	Examples
List	A list of **elements**	[1,2,3,4,5], [„I“, „am“, „Groot“], [True, False, True, True]
String	List of character or simply: a **text**.	„“I am Groot“
Tuple	**Immutable** list of elements	(1,2,3,4), („I“, „am“, „Groot“), („only one element requires a comma at the end“**,**)
Set	A list of **unique** elements	{„Every“, „value“, „only“, „once“}
Dictionary	A list of key-value pairs. Each key is **unique**.	{„DS9“: „Deep Space 9“, „Greece“: „Athen“, 1: 4, „a list“: [1,2,3,4,5]}

https://docs.python.org/3/tutorial/datastructures.html

Complex data types & data collections

```
a_list = [1, 2, 3, “4”]
# zero based indexing
print(a_list[0])
=> 1

# we can append new elements
a_list.append(5)
print(a_list)
=> [1, 2, 3, “4”, 5]

# removing the first element
a_list.pop()
```

https://docs.python.org/3/tutorial/datastructures.html#more-on-lists

Complex data types & data collections

```
a_dict = {“Groot”, “I am Groot”, ”This is”: ”Sparta”, 7: “of 9?”}
# Accessing by key
print(a_dict[“Groot”])
=> I am Groot
print(a_dict[7])
=> of 9?

# Adding a new element
a_dict[“Yoda”] = ”Do or do not, there is nor try”

# Removing an element
a_dict.remove(“Yoda”)
```

https://docs.python.org/3/tutorial/datastructures.html#dictionaries

Exercise 2

https://l.rub.de/74423d59

Control flow

- **if-else-statement**

```
surname = input("Enter surname:")
if surname[0].lower() < "m":
    print("Your case worker is Person X")
else:
    print("Your case worker is Person Y"
```

- **if-elif-else-statement**

```
surname = input("Enter surname:")
if surname[0].lower() < "j":
    print("Your case worker is person X")
elif surname[0].lower() >= "j" and surname[0].lower() < "q":
    print("Your case worker is person >")
elif ... :
else:
    print("No idea who your case worker is: ¯\_(ツ)_/¯ "
```

Control flow

- **`match`-statement**
  - Introduced in Python 3.10

```
def get_mass_of_amino_acid(amino_acid: str) -> float:
    match amino_acid.lower():
        case “a”:
            return 71.037113805
        case “c”:
            return 103.009184505
        ...
        case “w”:
            return 186.079312980
        case _:
            # Default case
            raise Error(f”’{}’ not found”)
```

Control flow

- **`try-catch`-statement**

```
amino_acid = input("Amino acid: ")

try:
    get_mass_of_amino_acid(amino_acid)
except Error as err:
    print("Oops! Something went wrong. You might want to have a look:", err)
```

Control flow

- **Loops**

```
# Simple for loop counting from 0 to 9
for i in range(10):
    print(i)
```

```
# A while loop repeats until a given condition is false
i = 0
while i < 10:
    print(i)
```

- **You can skip a step in the loop by calling `continue`**
- **Leaving a loop is possible with `break`**

Control flow

- **Loops**

```
numbers = [1,2,3,4,5,6,7,8,9,110]

for i in numbers :
    print(i)
```

- **Types which can be iterated called *iterables*!**

Control flow

- **Loops**

```
dictionary = {“I am”: “Groot”, “This is”: “Sparta”}
for i in dictionary:
    print(i)
```

```
dictionary = {“I am”: “Groot”, “This is”: “Sparta”}
for k, v in dictionary.items():
    print(k, v)
```

Exercise 3

https://l.rub.de/74423d59

Unleash the force

- **Next to the built-in functions and modules you can use 3rd party imports**
- **You need to install them first**

```
> pip install <PACKAGE_NAME>
```

- **You can find them**
  - **Python Package Index https://pypi.org/**
  - **GitHub (you can install directly from a repository)**
- **Be aware that this packages can also contain harmful code**
  - **Never install a package from a ZIP archive or similar**
  - **Look up the developers on GitHub**
  - **Check the stars on GitHub**

Object oriented programming

How would you represent a Lab with name, number of postdocs and students?

Object oriented programming

- **Let‘s abstract something by creating a template, a so called *class***

```
class Laboratory:
    # constructor
    def __init__(self, name: str, num_postdocs: int, num_students: int) -> Laboratory:
        self.name = name
        self.num_postdocs = num_postdocs
        self.num_students = num_students


    def decrease_students(self):
        self.num_students -= 1
```

Object oriented programming

- **Lets build some labs from our template**

```
# Create a lab
mann_lab = Lab(“Mann“, 4, 20)
# Create another lab
kuster_lab = Lab(“Küster“, 6, 40)

# We can also arrange them in a list
labs = [mann_lab, kuster_lab]

# Or directly
labs = [
    Lab(“Mann“, 4, 20),
    Lab(“Küster“, 4, 20)
]
```

Object oriented programming

- **Lets build some labs from our template**

```
# print all
for lab in labs:
    print(lab.name) # here we accessing the attribute name
=> Mann
=> Küster
```

Object oriented programming

- **Working with objects before**
  - List, dictionary, set, string
  - They just have special constructors

Pandas

- **Open source python package for data analysis**
- **Used for data exploration, cleaning and transformation/manipulation**
  - "… a more sophisticated Excel"
- **Works with data series (one dimensional array with indices) and *dataframes* (table 2D)**
- **Pandas documentation can be found here: https://pandas.pydata.org/docs/index.html**
- **And some starter tutorial: https://www.w3schools.com/python/pandas/default.asp**

Pandas - Dataframe

Column names

Columns
axis=1

Index
label

Index
axis=0

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
0	Avery Bradley	Boston Celtics	0.0	PG	25.0	6-2	180.0	Texas	7730337.0
1	John Holland	Boston Celtics	30.0	SG	27.0	6-5	205.0	Boston Uniersity	NaN
2	Jonas Jerebko	Boston Celtics	8.0	PF	29.0	6-10	231.0	NaN	5000000.0
3	Jordan Mickey	Boston Celtics	NaN	PF	21.0	6-8	235.0	LSU	1170960.0
4	Terry Rozier	Boston Celtics	12.0	PG	22.0	6-2	190.0	Louisville	1824360.0
5	Jared Sullinger	Boston Celtics	7.0	C	NaN	6-9	260.0	Ohio State	2569260.0
6	Evan Turner	Boston Celtics	11.0	SG	27.0	6-7	220.0	Ohio State	3425510.0

Missing
value

Data

Pandas – load your data

- **Pandas supports reading of a variety of file formats like excel, csv, tsv, json, hdf, html, sql, ...)**
- **We focus on the three most used:**

```
# Excel
df_sheet = pd.read_excel(„sample.xlsx“)
# CSV
df_csv = pd.read_csv(“sample.csv“)
# JSON
df_json = pd.read_json(“sample.json“)
```

- **The „read“ functions comes with a lot of parameters:**
  - Excel: `sheet_name(1)` or `sheet_name(“sheet_name1“)` or `sheet_name(None)`
  - All: `decimal = “,“` or `decimal = “.“`
  - Read all possibilities up in the docs!

Pandas – check your data

- **Always check if your file was loaded correctly.**
- **Either look at the whole dataframe:**

```
print(df)
```

- **But much more recommended:**

```
print(df.head()) # prints first 5 rows
```

- **Check if the columns are right:**

```
print(df.columns)
```

- **Check what your dataframe contains:**

```
print(df.info())
```

Pandas - Dropping

- **Dropping missing values from the whole dataframe:**

```
df.dropna()
```

- **Drop missing values from specific columns:**

```
df.dropna(subset = [“col1“, “col2“, “col3“, ...]
```

- **Remove all columns containing more than 20% NaNs**

```
df_20_missing = df.dropna(axis=1, thresh = int(0.2*df.shape[0]))
```

- **Delete rows with specific value in a specific column:**

```
new_df = df[df.column1 != 0]
```

- **Delete a specific column:**

```
new_df = df.drop(„column1“, axis = 1)
```

Pandas - Imputation

- **We can impute all missing values with the function fillna()**
- **With scalar:**

```
df.fillna(0)
df.fillna("spongebob")
```

- **With object:**

```
df.fillna(df.mean()) # (median, sum, ,...)
```

- **Impute only some columns:**

```
df.fillna(df.mean()["col1": "col5"])
# or
df.fillna(df.mean()[1:])
```

Pandas - Correlation

- **Pandas can calculate correlation on the whole dataframe, automatically excluding all NA/null values**
- **Different methods available: pearson , kendall and spearman**
- **Returns a dataframe with the correlation matrix**
- **Correlation within a dataframe:**

```
corr = df.corr(method="pearson", numeric_only=False)
```

- **Correlation with another dataframe:**

```
df.corrwith(another_df, method="pearson", numeric_only = False)
```

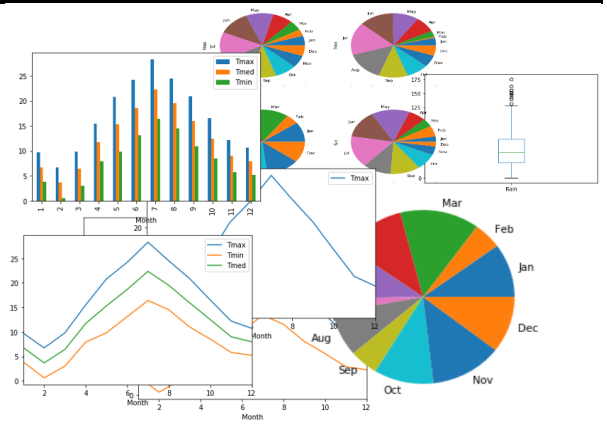
- **In jupyter notebook, we can visualize the matrix as a heatmap with**

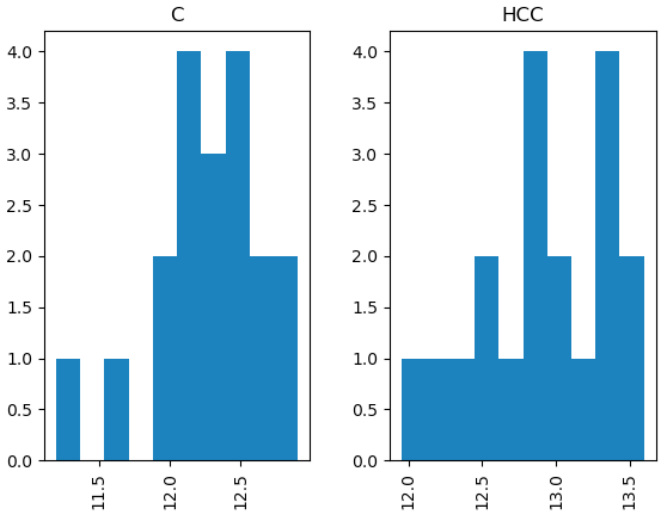
```
corr.style.background_gradient(cmap="coolwarm")
```

Pandas – Plots

- **Pandas supports multiple plot types like bar, hist, box, density, area, scatter and pie plots:**

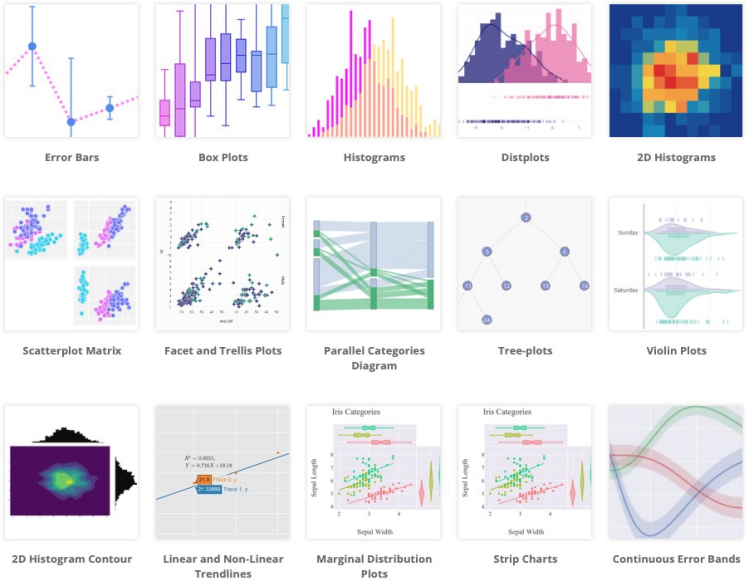
```
df.plot.hist()
df.plot.box()
df.plot.pie(subplots=True, figsize=(8,4));
```





Plotly (Why? ´Cause it is beautiful)

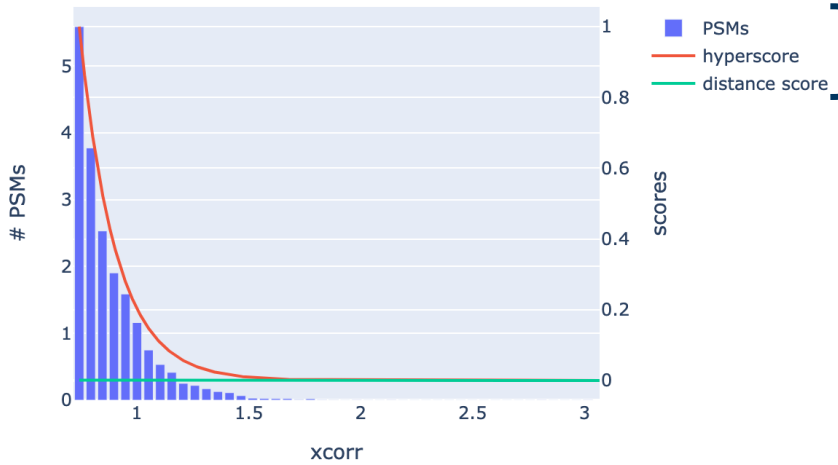
- **Plotly is a graphical library that supports multiple programming languages like Python, R, Julia, Javascript, F#...**
- **Can create**
  - Interactive plots (Jupyter Notebook, HTML/JS)
  - Static images (SVG, PNG, JPEG, PDF, …)
  - JSON for interchange
    - Languages (Python ⇔ R)
    - Server ⇔ Client
- **Can also be created from gglots2!**



Plotly (Why? ´Cause it is beautiful)

Interactivity!

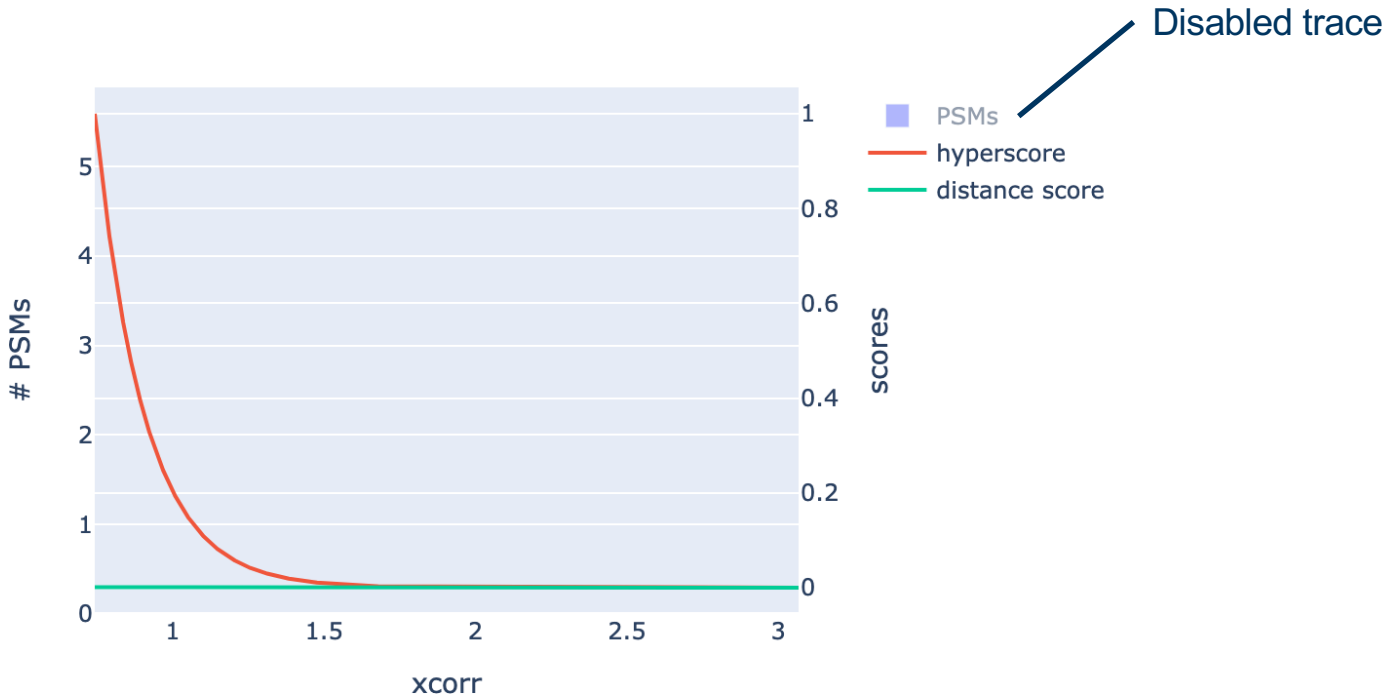
Controls (zoom, download, …)



Traces/ Datasets

Plotly (Why? ´Cause it is beautiful)

Interactivity!



Plotly (Why? ´Cause it is beautiful)

```
import plotly.graph_objects as go
fig = go.Figure()
fig.add_trace(go.Bar(
    x = df[“a_col”],
    y = df[“another_col”]
))
fig.show()
fig.write_image("fig1.png")
```

Sweetviz

- **Open-source Python library to „kickstart“ Exploratory Data Analysis (EDA)**
- **High-density visualization**
- **HTML or jupyter notebook**
- **Compare 2 Dataframes or 2 subsets of the same dataframe**
- **More libraries like this:**
  - Dtale
  - Autoviz
  - YData Profiling

Exercise 4

https://l.rub.de/74423d59

Scientific libraries

- **scipy**
  - **Mathematical functions (algebra, interpolation, distributions, ...)**
- **scikit-learn**
  - **„Classic“ machine learning (Random Forest, SVM, ...)**
- **PyTorch / Keras**
  - **Deep Learning**
    - **Layers types: Dense, Convolution, LSTMs, …**
    - **Activation functions: ReLU, Sigmoid, …**
    - **Optimizers: SDG, Adam, Nadam, …**

Block 3 nach Pause

Shifting gears - Rust integration

- **Why?**
  - Python is slow!
  - No worries: It is actually as fast as an interpreter language can be and 1000x faster than you
  - Why is it so used for machine / deep learning
  - The real power of Python lies in *native modules* like Pandas, Numpy etc.
    - Written in C, C++ and Rust, which are?
      - Compiler languages!

Shifting gears - Rust integration

- **Interpreter language**

```
i = 1
i += 1
for x in range(10):
    print(i**x)
```

- **They have advantages**
  - Easy to program
  - Flexible
  - Able to reprogram itself during runtime

Shifting gears - Rust integration

- **Compiler language**
  - Strict
  - Typed
  - Once finished it is compiled into machine code
  - Need to be recompiled after changes

```
fn main() {
    let mut i: i64 = 1;
    i += 1;
    for x in 0..10 {
        println!("{}", i**x);
    }
}
```

Shifting gears - Rust integration

- **Compiler language**
  - Why not C or C++? Why Rust?
  - Speed is the same
  - Package ecosystem
    - https://crates.io (Python equivalent https://pypi.com)
    - `cargo` (Python equivalent: `pip`)
  - Memory safe

Python vs. Rust

```python
i = 1
i += 1
for x in range(10):
    print(i**x)
```

```rust
fn main() {
    let mut i: i64 = 1;
    i += 1;
    for x in 0..10 {
        println!("{}", i**x);
    }
}
```

Preparation

- **Install rustup by following instructions on https://rustup.rs/**

```
# Terminal
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
rustup install stable
pip install maturin
apt update
apt install build-essentials
```

First we create a mixed project

```
# Terminal
maturin init denbi_summerschool2023_rs -mixed
# Select pyo3
```

Name	Last Modified
python	11 minutes ago
src	11 minutes ago
Cargo.toml	11 minutes ago
pyproject.toml	11 minutes ago

First we create a mixed project

```rust
// src/lib.rs
use pyo3::prelude::*;
/// Formats the sum of two numbers as string.
#[pyfunction]
fn sum_as_string(a: usize, b: usize) -> PyResult<String> {
    Ok((a + b).to_string())
}

/// A Python module implemented in Rust.
#[pymodule]
fn summerschoo2023_rs(_py: Python, m: &PyModule) -> PyResult<()> {
    m.add_function(wrap_pyfunction!(sum_as_string, m)?)?;
    Ok(())
}
```

Compile

```
# Terminal

maturin develop -r
```

Try it

- **Try it in a interactive Python shell**
- **Just type `python` in your terminal**
- **And type in**

```
from summerschoo2023_rs import *
sum_as_string(1, 2)
```

Exercise 5

- **Reimplement the function from exercise 3 in Rust**
  - https://doc.rust-lang.org/book/
  - https://pyo3.rs/v0.19.2/
- **Read the mass spectrum from exercise 3 again using Python and check if your Rust implementation works**
- **Time your rust implementation**

Exercise 5

https://l.rub.de/74423d59

Where to go?

- **Get more proficient in object oriented programming**
  - Dataclasses & slots
  - Properties
- **Dependency management**
  - You need to make your code installable and reproducible. Very important for reviewers
- **GIT**
  - Helps you to keep track of your changes as code will improve iteratively
  - Make your code collaboratively available on GitHub / GitLab etc.
  - The wrong way: Nextcloud, Google Cloud, Dropbox, Word …

Where to go?

- **(Unit-) Tests**
  - Automatic tests will help you keep your code functional after changes
- **Practice practice practice**
  - Automate small task
  - Leet code: https://leetcode.com/
    - Provides you with small code challenges and checks your results