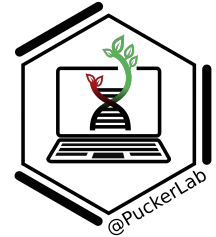
UNIVERSITÄT BONN



Prof. Dr. Boas Pucker

Python - Application examples

UNIVERSITÄT BONN

Availability of slides

- All materials are freely available (CC BY) - after the lectures:
  - GitHub: https://github.com/bpucker/teaching/tree/master/WBIO-A-07

- Questions: Feel free to ask at any time

- Feedback, comments, or questions: pucker[a]uni-bonn.de

UNIVERSITÄT BONN

Reverse complement of nucleotide sequence

What happens here?

```
1  def revcomp( seq ):
2
3      seq = seq.lower()
4
5      #key:value (=dictionary)
6      complement = { 'a':'t', 't':'a', 'c':'g', 'g':'c' }
7
8      new_seq = []
9
10     for nt in seq:
11         new_seq.append( complement[ nt ] )
12
13     #list[::-1] inverts list (last element becomes first)
14     new_seq = "".join( new_seq[::-1] )
15
16     return new_seq
```

Sequence of bases e.g. ATGACATGA

Converts input to lower case: atgacatga

Get complement for each base

Inverts list (=reverse)

UNIVERSITÄT BONN

How to use dictionaries

- Values are accessible via keys
- Keys need to be unique
- Quick access to data based on key
- Higher memory occupation than lists/strings

```
my_dict = {"k1": "v1", "k2": {"x1": "y1"}, 5: ["one", "two", "three"], "hello world": "hello world" }
print(my_dict.keys()) #all keys of a dictionary
print(my_dict.values()) #all values of a dictionary
print(my_dict["k1"])
print(my_dict["k2"]["x1"])
```

```
dict_keys(['k1', 'k2', 5, 'hello world'])
dict_values(['v1', {'x1': 'y1'}, ['one', 'two', 'three'], 'hello world'])
v1
y1
```

UNIVERSITÄT BONN

Exercises - Part 6a

- 6.1) Write a function to get the reverse complement (upper case letters) of a DNA sequence given in upper case letters!
- 6.2) Write a function to translate a DNA sequence into amino acids (first frame only)!

- 6.X1) Write a function to translate DNA sequences in all 6 frames into peptide sequences! The longest peptide sequence per DNA sequence should be returned!
- 6.X2) Write a function to grep a sequence from a FASTA file based on the name of this sequence!

UNIVERSITÄT BONN

How to approach a bioinfo challenge?

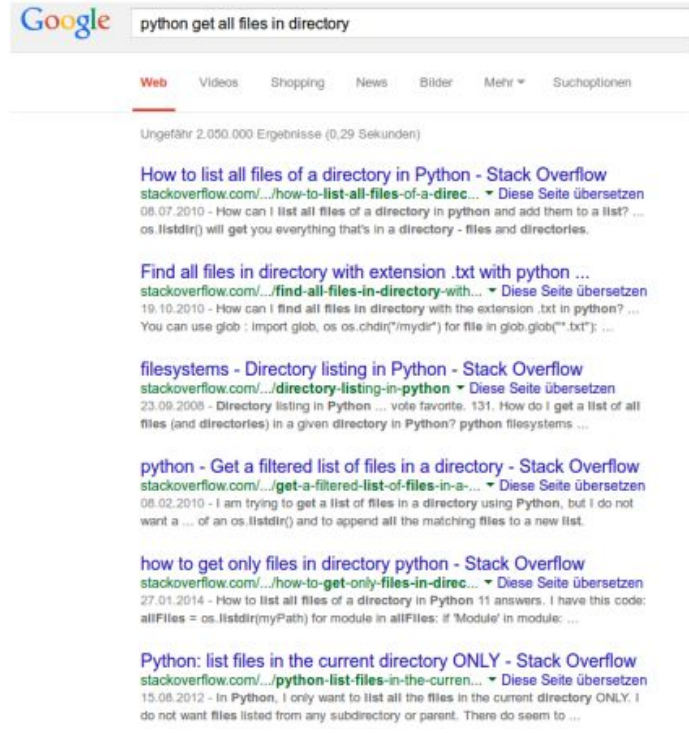
- Identify the problem that needs to be solved
- Split the problem into smallest possible parts
- Solutions for small parts of the problem might be available already
- Precise description of the problem is required for online search
- Best hit often leads to stack overflow:
  - Problem is described by the asking person
  - Multiple solutions are suggested by the community
  - Community votes to identify the best solution
  - Green marking highlights the answer that solved the problem

UNIVERSITÄT BONN

Example

- Problem: get paths of all files in a certain directory

- Search expression: ‘python get all files in directory’



Best hit on stackoverflow

533 user like this answer

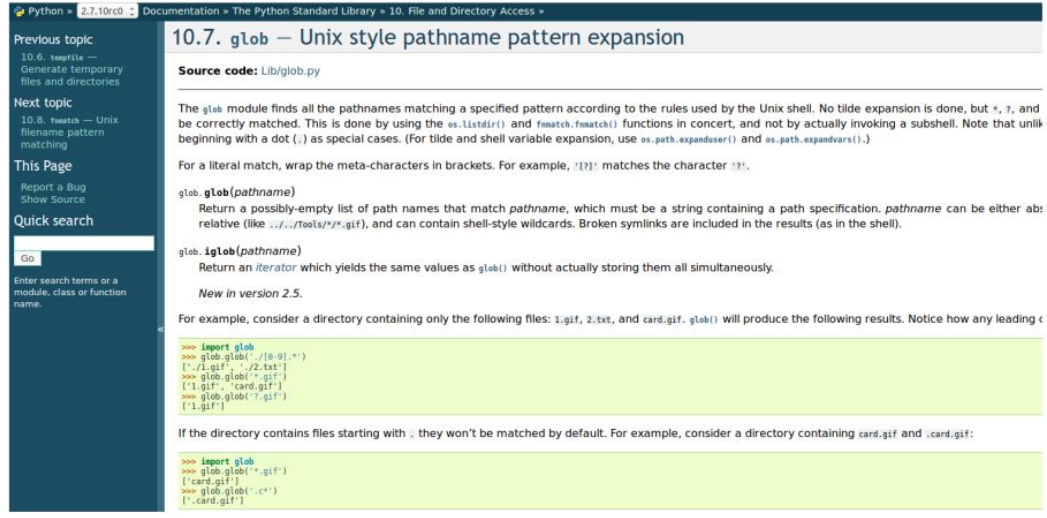
This answers solved the question

Three different ways to solve this problem are described

UNIVERSITÄT BONN

Official Python documentation

- Systematic documentation of all functions in a module with all possible arguments
- Sometimes examples are given



UNIVERSITÄT BONN

Linux (Ubuntu)

- Ubuntu is an operating system with a graphical user interface
- Excellent environment to perform bioinformatics
- Offers a powerful terminal and comes with comprehensive support
- Processing of large data sets is more efficient via command line tools
- Dual boot system with Windows is possible
- Configuration of USB stick for Ubuntu is possible to explore opportunities

https://ubuntu.com/tutorials/create-a-usb-stick-on-ubuntu#1-overview

How to run BLAST?

- Running at the NCBI website does not give you full control over all parameters
- Python can be used to run a local search:

```
blastn \
-query <query_file> \
-subject <subject_file> \
-out <output_file> \
-outfmt 6 \
-evalue 0.01 \
-word_size 4
```

1.	qseqid	query (e.g., gene) sequence id
2.	sseqid	subject (e.g., reference genome) sequence id
3.	pident	percentage of identical matches
4.	length	alignment length
5.	mismatch	number of mismatches
6.	gapopen	number of gap openings
7.	qstart	start of alignment in query
8.	qend	end of alignment in query
9.	sstart	start of alignment in subject
10.	send	end of alignment in subject
11.	evalue	expect value
12.	bitscore	bit score

https://www.metagenomics.wiki/tools/blast/blastn-output-format-6

UNIVERSITÄT BONN

How to execute processes via shell?

- Running shell commands through the subprocess module:

  p = subprocess.Popen(arg='ls -lh', shell=True)

  p.communicate()

- Can be used to run everything via Python

- Python waits until the command is completed

- Example:

```
import subprocess
p = subprocess.Popen( arg="mkdir test && cd test && ls -lh", shell=True )
p.communicate()
```

UNIVERSITÄT BONN

How to process BLAST results?

```python
def load_BLAST_results( input_file ):
    """! @brief load all BLAST results from file """

    data = []
    with open( input_file, "r" ) as f:
        line = f.readline()
        while line:
            parts = line.strip().split('\t')
            data.append( {  'query': parts[0],
                            'subject': parts[1],
                            'query_start': int( parts[6] ),
                            'query_end': int( parts[7] ),
                            'score': float( parts[-1] )
                         } )
            line = f.readline()
    return data
```

```
AT1G01010   NdCChr1.g1.t1   100.00  429   0   0   1   429   1   429   0.0    895
AT1G01010   NdCChr4.g18734.t1       32.84   469   247   15   1   428   2   443   4e-56   194
AT1G01010   NdCChr1.g127.t1 34.23   336   157   10   1   330   1   278   1e-41   152
AT1G01010   NdCChr1.g128.t1 32.38   349   157   14   1   331   1   288   4e-39   146
AT1G01010   NdCChr4.g18730.t1       39.33   178   95   5   1   175   2   169   1e-32   126
AT1G01010   NdCChr3.g12773.t1       39.63   164   89   2   1   162   1   156   1e-28   115
AT1G01010   NdCChr4.g22969.t1       40.74   162   79   5   5   159   11   162   3e-28   117
AT1G01010   NdCChr4.g18733.t1       40.00   165   90   5   1   162   2   160   4e-28   115
AT1G01010   NdCChr3.g17122.t1       42.31   156   74   6   5   153   15   161   4e-27   112
```

UNIVERSITÄT BONN

Exercises - Part6b

- 6.6) Collect the best CHS BLAST result per contig from the CHS_vs_Digitalis.txt file.

- 6.7) Count the number of BLAST hits that show a similarity >80%, an alignment length >200, and an e-value<10^{-10}.

UNIVERSITÄT BONN

How to organize a Python script

- Make a script recognize that it needs to run with Python:

  #!/usr/bin/env python3

- Other information to include:
  - Author
  - Version
  - Usage
  - Imports

```
1  ### Boas Pucker ###
2  ### bpucker@cebitec.uni-bielefeld.de ###
3  ### v0.2 ###
4
5  __usage__ = """
6                  python construct_RNA_seq_coverage_file.py\n
7                  --in <BAM_FILE>
8                  --out <OUTPUT_FILE>
9
10                 --bam_is_sorted <PREVENTS_EXTRA_SORTING_OF_BAM_FILE>
11
12                 feature requests and bug reports: bpucker@cebitec.uni-bielefeld.de
13                 """
14
15 __cite__ = """ Pucker & Brockington, 2018: https://doi.org/10.1186/s12864-018-5360-z """
16
17
18 import os, sys
19
20 # --- end of imports --- #
21
22 def main( arguments ):
```

UNIVERSITÄT BONN

How to pass arguments to a Python script?

```
1
2   __usage__ = """ how to run the script and list of arguments """
3
4   def main( arguments ):
5       """! @brief run everything """
6
7       fasta_file = arguments[ arguments.index( '--fasta' )+1 ]
8       gff3_file = arguments[ arguments.index( '--gff3' )+1 ]
9       species = arguments[ arguments.index( '--species' )+1 ]
10      output_dir = arguments[ arguments.index( '--tmp' )+1 ]
11      hints_file = arguments[ arguments.index( '--hints' )+1 ]
12
13      if '--cutoff' in arguments:
14          cutoff = int( arguments[ arguments.index( '--cutoff' )+1 ] )
15      else:
16          cutoff = 1
17
18      #everything happens here
19
20  if '--fasta' in sys.argv and '--gff3' in sys.argv and '--species' in sys.argv and '--tmp' in sys.argv and '--hints' in sys.argv:
21      main( sys.argv )
22  else:
23      sys.exit( __usage__ )
```

UNIVERSITÄT BONN

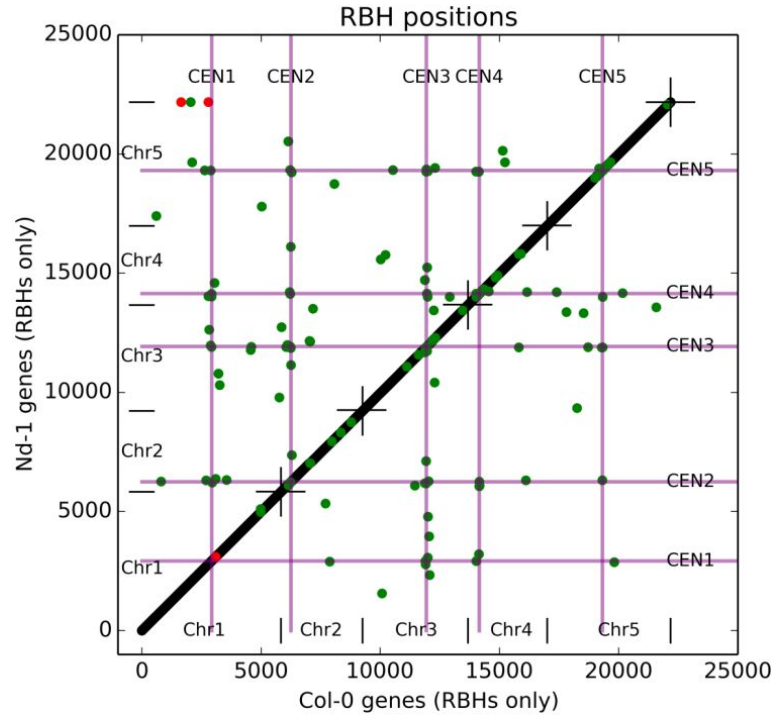
Command line examples

```
python transeq.py \
--in Eucommia_ulmoides.cds.fasta \
--out Eucommia_ulmoides.pep.fasta

python3 construct_anno.py \
--out ./Eucommia_ulmoides_anno/ \
--in ./Eucommia_ulmoides.pep.fasta \
--ref ./Araport11_genes.201606.pep.repr_MOD.fasta \
--anno ./araport11_annotation.updated.txt
```

UNIVERSITÄT BONN

matplotlib

- Importing matplotlib:
  import matplotlib.pyplot as plt
- Visualization of complex data
- Automatic generation of plots
- Unlimited customization options

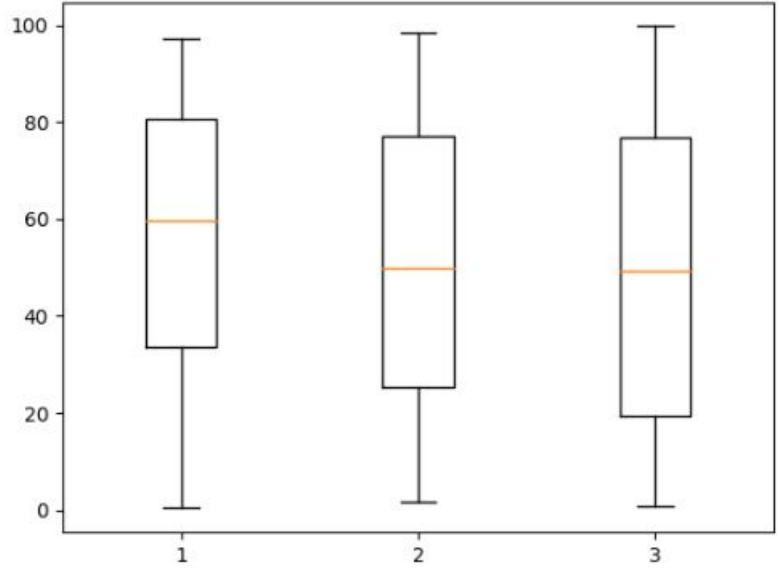


Pucker et al., 2016: 10.1371/journal.pone.0164321

UNIVERSITÄT BONN

Box plot

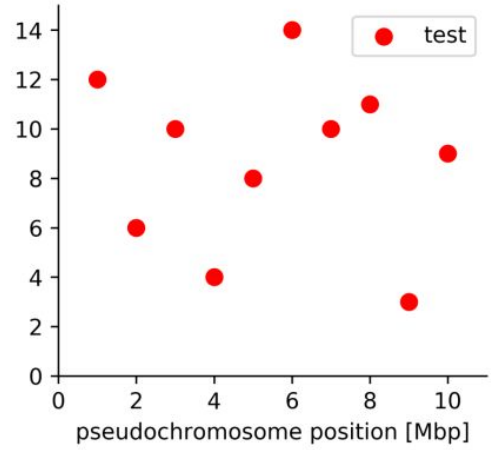
```python
import matplotlib.pyplot as plt
import numpy as np

d1 = np.random.rand(50) * 100   #generate random numbers
d2 = np.random.rand(50) * 100
d3 = np.random.rand(50) * 100

data = [d1, d2, d3] # multiple box plots on one figure

plt.figure()
plt.boxplot(data)
plt.show()
```



UNIVERSITÄT BONN

Scatter plot

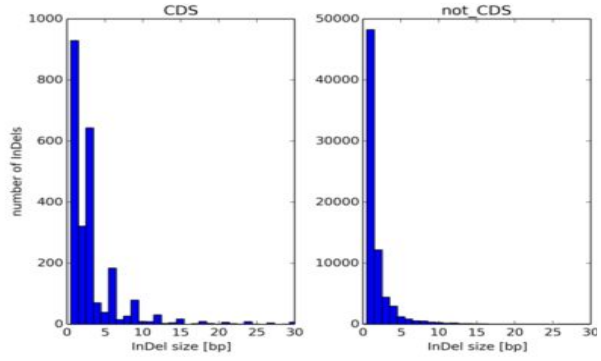
```python
import matplotlib.pyplot as plt

fig, ax = plt.subplots( figsize=( 10, 4 ) )   #defining size of plot

x_values = [ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ]
y_values = [ 12, 6, 10, 4, 8, 15, 10, 11, 3, 9 ]

ax.scatter( x_values, y_values, color="red", s=10, marker="o", label="test" )
#setting color, marker size, marker shape and label of this group

ax.legend( numpoints=1 )
#each group is represented by only one marker in the legend (default=3)

ax.set_xlim( 0, 11 )    #set range of x-axis
ax.set_ylim( 0, 15 )    #set range of y-axis

ax.set_xlabel( "pseudochromosome position [Mbp]" )

ax.spines["top"].set_visible(False)      #remove lines and ticks
ax.spines["right"].set_visible(False)    #remove lines and ticks

plt.subplots_adjust(left=0.05, right=0.99, top=0.97, bottom=0.12)
#adjust size of plot within figure

plt.show()
fig.savefig( "my_plot.png", dpi=600 )   #write figure into output file
plt.close( "all" )  #destroy created figures (cleaning up)
```



UNIVERSITÄT BONN

Histogram

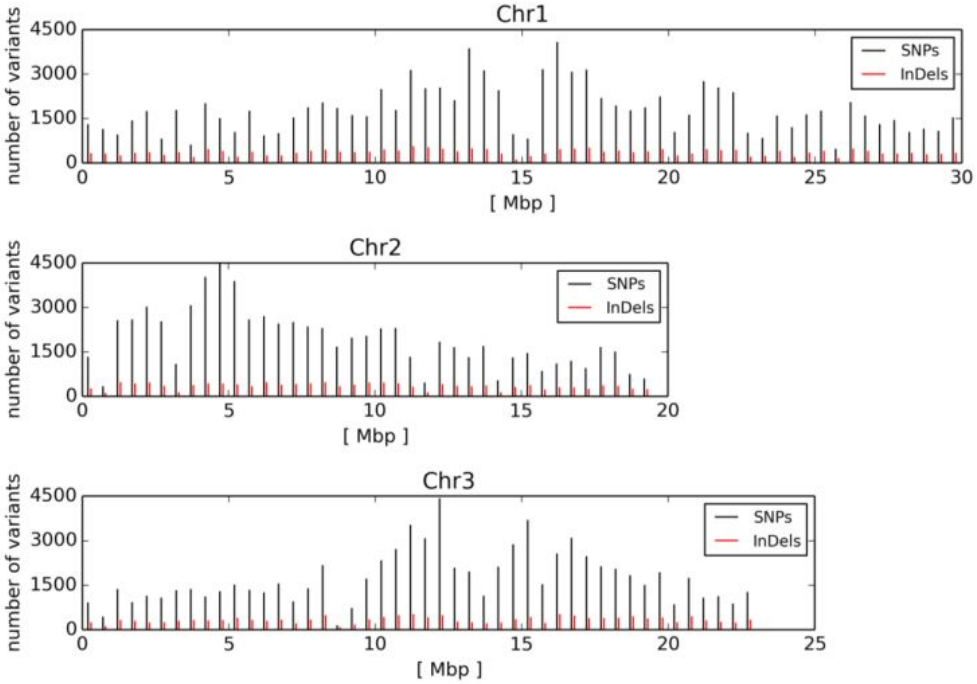
```
1   import matplotlib.pyplot as plt
2
3   # --- end of imports --- #
4
5   gene_space = [  3, 3, 6, 6, 9, 9, 12, 3, 3, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
6                   11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
7                   12, 15, 18, 21, 24, 27, 30 ]
8   intergenic = [ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 1, 2, 3, 4, 5, 6, 7, 8, 9,
9                   1, 2, 3, 4, 5, 6, 7, 1, 2, 3, 4, 1, 2, 1 ]
10
11
12  fig, ( ax1, ax2 ) = plt.subplots( 1, 2, sharey=False)
13  counts, bins, patches = ax1.hist( gene_space, bins=max( gene_space ), align="left" )
14  ax1.set_title( "CDS" )
15  ax1.set_xlim( 0, 30 )
16  ax1.set_xlabel( "InDel size [bp]" )
17  ax1.set_ylabel( "number of InDels" )
18
19  counts, bins, patches = ax2.hist( intergenic, bins=max( intergenic ), align="left" )
20  ax2.set_title( "not_CDS" )
21  ax2.set_xlim( 0, 30 )
22  ax2.set_xlabel( "InDel size [bp]" )
23  plt.subplots_adjust( wspace=0.3 )   #increase space between figures
24
25  plt.show()
26  fig.savefig( prefix + "InDel_size_distribution.png", dpi=300 )
27  plt.close('all')
```



Pucker et al., 2016: 10.1371/journal.pone.0164321

UNIVERSITÄT BONN

Barplot figure



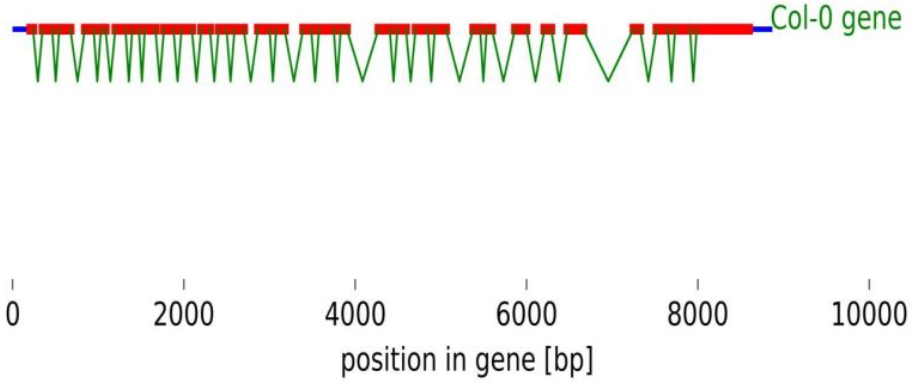
barplots.py generates barplots at specific positions by drawing a normal line

(script is available in course repository)

Pucker et al., 2016: 10.1371/journal.pone.0164321

Gene structure plot

- gene_structure_plot.py generates visualizations of gene/transcript structures based on GFF annotations



Pucker et al., 2017: 10.1186/s13104-017-2985-y

UNIVERSITÄT BONN

Exercises - Part6c

- 6.8) Construct a figure to illustrate the order and orientation of genes in the gum gene cluster of *Xanthomonas campestris* pv. campestris!

- 6.9) Save this figure in different file formats (png, jpg, pdf, svg)!

UNIVERSITÄT BONN

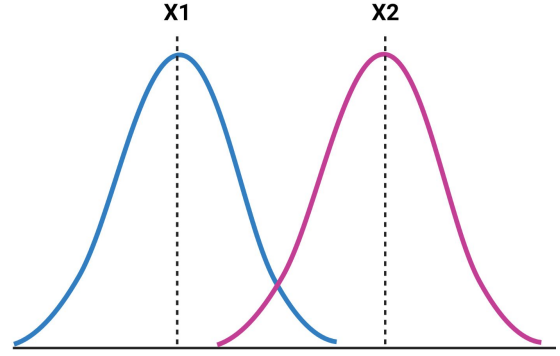
Statistics

- Compare observed sample against the expected distribution
- Check if two samples are derived from same distribution

- H0 = samples were taken from same distribution

- H0 can only be rejected or kept due to insufficient evidence against it

- H0 can NEVER be confirmed



UNIVERSITÄT BONN

Shapiro-Wilk test

- Testing data set for normal distribution

- Important for decision about potential tests

```
from scipy import stats
x = [1, 2, 3, 3, 3, 2, 1]
stats.shapiro(x)
```

UNIVERSITÄT BONN

Correlation

- Pearson correlation coefficient is suitable for data following a normal distribution

- Spearman correlation coefficient is better if data distribution is unknown

```
from scipy import stats
x = [1,2,3,4,5]
y = [2,4,6,8,10]
r,p = stats.pearsonr(x,y)
r,p = stats.spearmanr(x,y)
```

UNIVERSITÄT BONN

t-test

- Samples need to show normal distribution
- Comparison of one sample against a reference value
- Comparison of two samples:
  - Paired samples (ttest_rel)
  - Unpaired samples (ttest_ind)

```
from scipy import stats
x = [1,2,3,4,5]
y = [4,6,8,10,11]
t,p = stats.ttest_ind(x,y) #independent samples
t,p = stats.ttest_rel(x,y) #paired samples
```

UNIVERSITÄT BONN

W-test

- Wilcoxon (W) test compares two paired samples

- Normal distribution is not required

```
from scipy import stats
x = [1,2,3,4,5]
y = [4,6,8,10,11]
w,p = stats.wilcoxon(x,y)
```

UNIVERSITÄT BONN

U-test

- Comparison of unpaired samples

- Normal distribution is not required

```
from scipy import stats
x = [1,2,3,4,5]
y = [4,6,8,10,11]
w,p = stats.mannwhitneyu(x,y)
```

UNIVERSITÄT BONN

Chi square test

- Comparison of an observation against an expectation

- Comparisons of two observations

```
from scipy import stats
obs = [1,2,3,4,5]
exp = [4,6,8,10,11]
x, p = stats.chisquare(obs,exp)
```

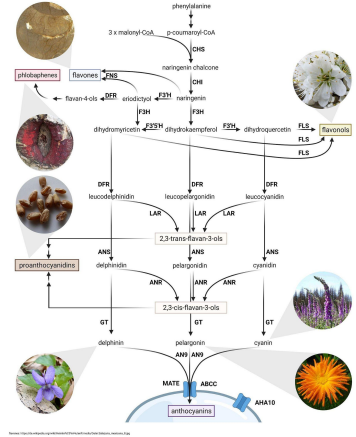
UNIVERSITÄT BONN

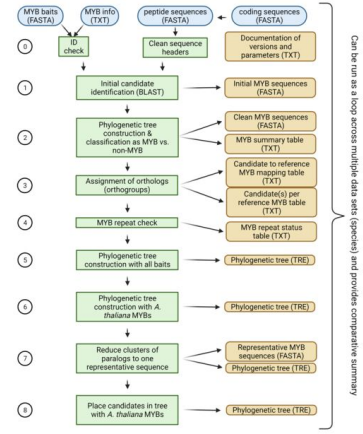
Exercises - Part6c (UNKNOWN_DATA.ods)

- 6.10) Construct a suitable visualization!
- 6.11) Analyze distribution and trends!
- 6.12) Apply statistical test to investigate difference!

Looking for more Python opportunities?

- Plant Biotechnology and Bioinformatics group is working on:
  - Plant genomics
  - Plant transcriptomics (RNA-seq)
  - Specialized plant metabolites
  - Synthetic biology
  - Big data comparative studies
  - Tool development
  - Data reuse





- Details: https://www.pbb.uni-bonn.de/
- Web server: https://pbb-tools.de/

Pucker, 2022: 10.1186/s12864-022-08452-5

UNIVERSITÄT BONN

Time for questions!

UNIVERSITÄT BONN

References

- Nd-1 genome assembly (Pucker *et al*., 2016)
  - https://doi.org/10.1371/journal.pone.0164321
- Non-canonical splice sites (Pucker *et al*., 2017)
  - https://doi.org/10.1186/s13104-017-2985-y
- *Croton tiglium* transcriptome assembly (Haak *et al*., 2018)
  - https://doi.org/10.3389/fmolb.2018.00062
- Genome-wide non-canonical splice sites in plants (Pucker & Brockington, 2018)
  - https://doi.org/10.1186/s12864-018-5360-z
- Chromosome-level Nd-1 genome assembly (Pucker *et al*., 2019)
  - https://doi.org/10.1371/journal.pone.0216233
- NAVIP (Baasner *et al*., 2025)
  - https://doi.org/10.1371/journal.pcbi.1012732