

Data Visualization

01

Prof. Dr. Phillipp Torkler

Data Visualization Provides Data Exploration and Communication

Data Visualization serves **two** major purposes:

1. Data Exploration:

- Familiarize with a data set
- Look for patterns in data
- Patterns or regularities in the data set are expected but not known in advance
- Data exploration can guide the scientific process
- New scientific ideas can emerge from visualizations from (laboratory) data. Thus, data visualization is often more than ‘just showing data’.

2. Communication / Presentation:

- Interesting findings have been made and need to get communicated clearly to readers
- Focus to tell a clear message
- A reader does not need to get through the same process as a researcher that tries to find patterns during data exploration. In contrast, key findings should be communicated without overwhelming a reader with unnecessary data

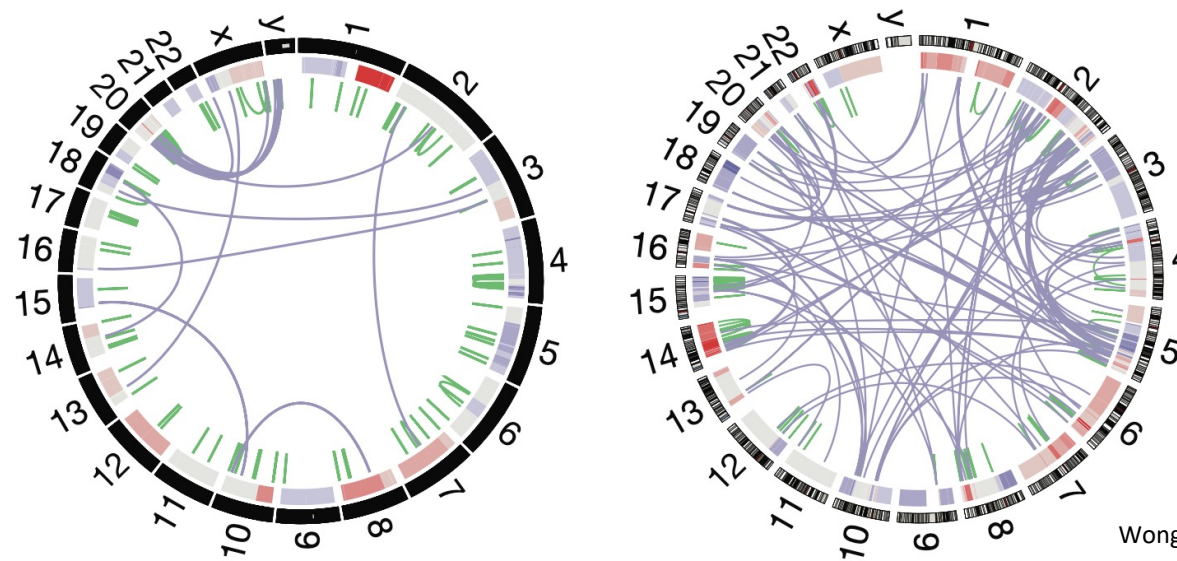
In either case, the purpose of any figure is to transport a message!

Why Do We Need Exploration and Communication?

Today, the challenge for researchers is to take benefit from large data sets without getting drawn in too much data. Thus, both exploration and communication is key for finding and sharing new insights.

Goals of data visualization:

- enable researchers to explore and explain their data through (interactive) visualizations
- take advantage of the human's ability to recognize patterns
- data types and research questions evolve rapidly in the scientific community. Likewise, data visualizations need to adapt to new techniques to provide insights.
- Visualization as a complement for algorithmic approaches to provide a mental image of what happens (see example)



Technical and Design Aspects of Data Visualization

A figure can be **any** visual representation: graphs, photos, drawings, schematics, cartoons, maps etc.. Despite their variety, the purpose of any figure is to support a message. **Good figures will show the data AND the message** you want to tell!

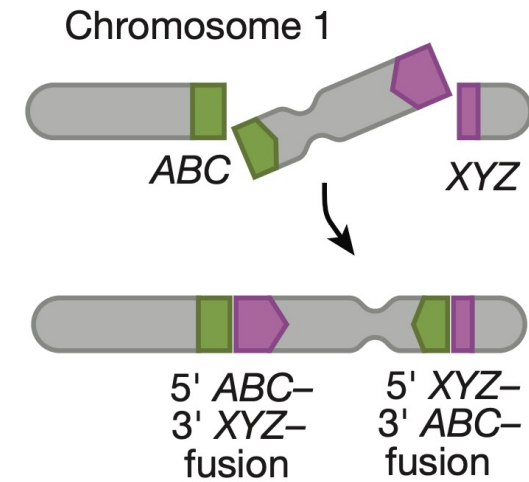
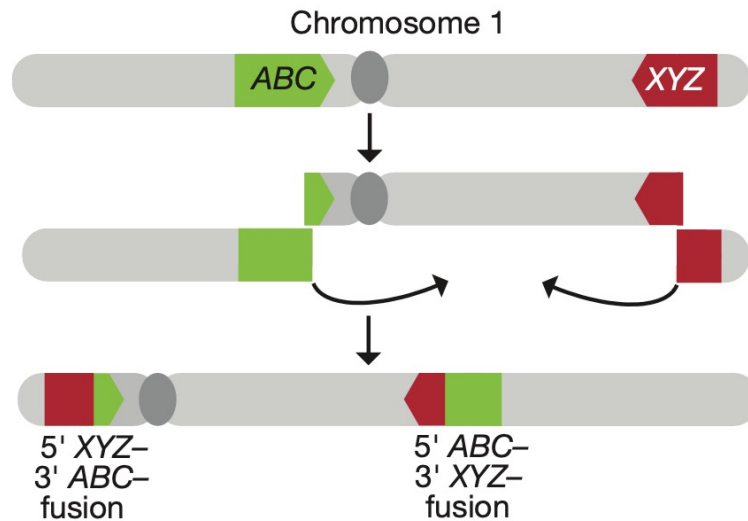
Figure creation consists of two major building blocks:

1. The 'technical' generation of a figure (the doing):
 - e.g. use of a programming language and corresponding graphic library
 - choice and usage of graphics software
 - photography etc...
2. The 'design' of a figure (the theory and idea):
 - use graphic design principles
 - biology of the human visual system
 - psychology

In this course we want to train 'the doing' and introduce a couple of design principles to generate meaningful and clear figures.

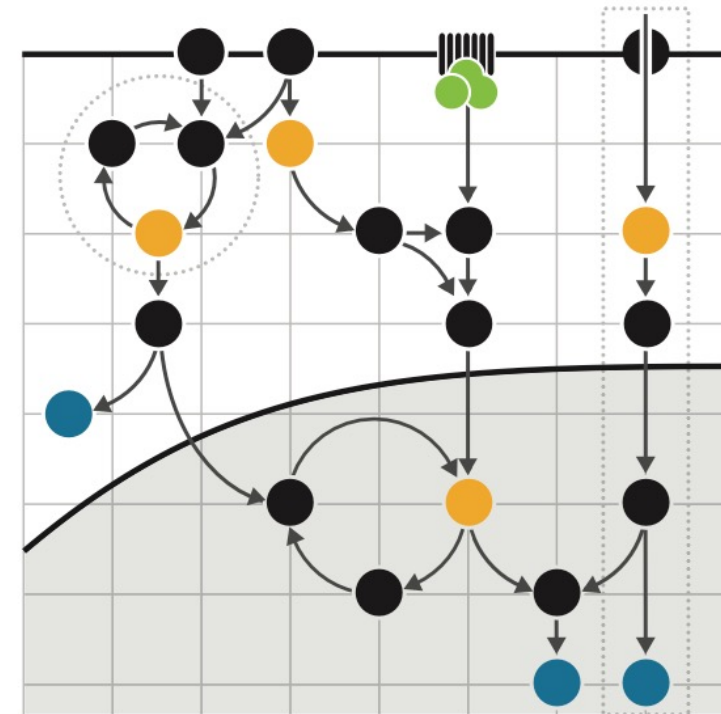
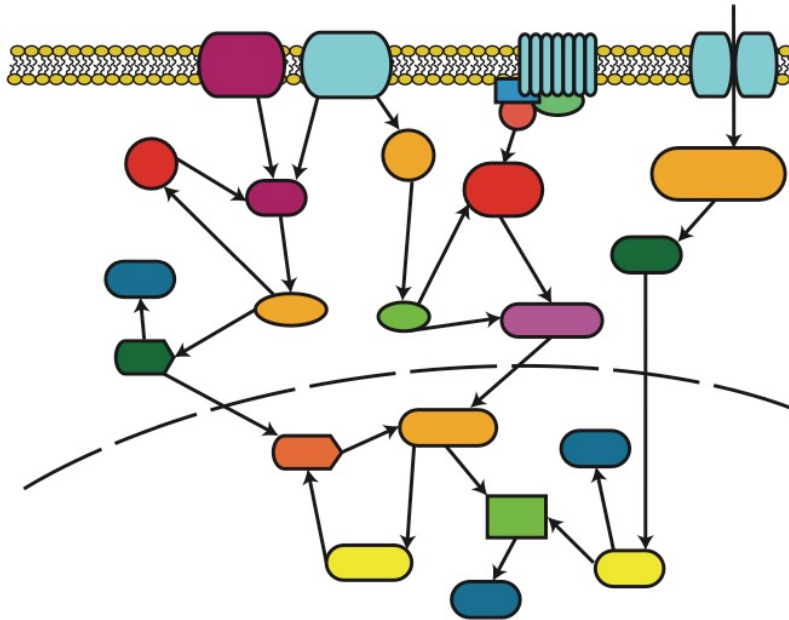
Examples For Clearly Structured Figures (1)

Visualizing chromosomal inversion that results in two fusion genes:



Examples For Clearly Structured Figures (2)

Examples for pathway diagrams



- redundant visual encodings removed and main points emphasized by visual grouping
- Color and shape variations have been removed except for those highlighting a molecule of interest (orange), the products of the pathway (blue)

Reading Exercise

- What is salience?
- Why do we need to consider the concept of salience when designing (scientific) visualizations?

Notes

Figure Generation

Vector vs. Raster Graphics



Vector vs. Raster Graphics



Raster Graphic



Vector Graphic



<https://inkscape.org/~ozant/star-bot>

Vector vs. Raster Graphics



<https://inkscape.org/~ozant/★art-bot>

A raster graphic of width n and height m is described by an mn array where each position i, j stores the color information of the corresponding pixel of the graphic.

In contrast, vector graphics are described by graphical primitives like lines (described by two points) and circles (described by center and radius). Primitives can be described by mathematical functions, and they can be combined to build complex objects.

Objects in a vector graphic are fully described by their primitives and as a consequence of that vector graphics can be scaled without loss in quality.

Take home message: schematic drawings or data visualizations are ideally generated in a vector format.



Tools For Creating and Editing Vector Graphics

There are many tools available, but two prominent examples are Adobe Illustrator (if you have money and want to pay) and Inkscape (free and open-source). Both programs can be used to create or edit vector graphics.

<https://inkscape.org>



<https://www.adobe.com/products/illustrator.html>



Knowledge in these tools can be quite helpful even as a computer scientist. Composing bigger figures from multiple single plots, quickly adjust or align colors, add explanations or explanatory drawings can help to create better figures. Depending on the scenario, compositions can be done more easily in these programs rather than doing these edits via programming.

Figure Generation via Programming

Of course, as computer scientists we want to generate figures from data automatically via programming languages. In the first part of the course, you have already been introduced into *base R* and *ggplot2*. For the sake of comparison, let's look at a few others:

Base R [<https://cran.r-project.org>]

ggplot2 (R) [<https://ggplot2.tidyverse.org>]

matplotlib (python) [<https://matplotlib.org>]

seaborn (Python) [<https://seaborn.pydata.org/examples/index.html>]

D3 (JavaScript) [<https://d3js.org>]

plotly (R, Python, JavaScript) [<https://plotly.com>]

bokeh (Python) [<https://docs.bokeh.org/en/latest/index.html>]

and so on.... there are so many...

If There Are So Many, Which Should I Learn?

Typically, it makes sense to focus on a plotting library that belongs to the language that is used in the project. Since most data science is performed via R or Python it is beneficial to have experience in both. For R good starting points are base R and ggplot2 for Python matplotlib is very popular.

Comparing the left and right list, what do you think is the difference between them?

Base R [<https://cran.r-project.org>]

matplotlib (python) [<https://matplotlib.org>]

D3 (JavaScript) [<https://d3js.org>]

ggplot2 (R) [<https://ggplot2.tidyverse.org>]

seaborn (Python) [<https://seaborn.pydata.org/examples/index.html>]

plotly (R, Python, JavaScript) [<https://plotly.com>]

bokeh (Python) [<https://docs.bokeh.org/en/latest/index.html>]

If There Are So Many, Which Should I Learn?

Typically, it makes sense to focus on a plotting library that belongs to the language that is used in the project. Since most data science is performed via R or Python it is beneficial to have experience in both. For R good starting points are base R and ggplot2 for Python matplotlib is very popular.

Comparing the left and right list, what do you think is the difference between them?

Base R [<https://cran.r-project.org>]

matplotlib (python) [<https://matplotlib.org>]

D3 (JavaScript) [<https://d3js.org>]

ggplot2 (R) [<https://ggplot2.tidyverse.org>]

seaborn (Python) [<https://seaborn.pydata.org/examples/index.html>]

plotly (R, Python, JavaScript) [<https://plotly.com>]

bokeh (Python) [<https://docs.bokeh.org/en/latest/index.html>]

It's hard to draw a clear line, but some libraries are more low-level (e.g. base R, D3, matplotlib), whereas others are more high-level (e.g. seaborn, ggplot2). It's a trade-off. The more high-level a library is, the more you need to stick to the decisions and the offered functionality. Extending or changing is often cumbersome. Low-level offers more flexibility at the cost of spending more time to create visual pleasing plots.

Technical Requirements

Conda, Python, Matplotlib

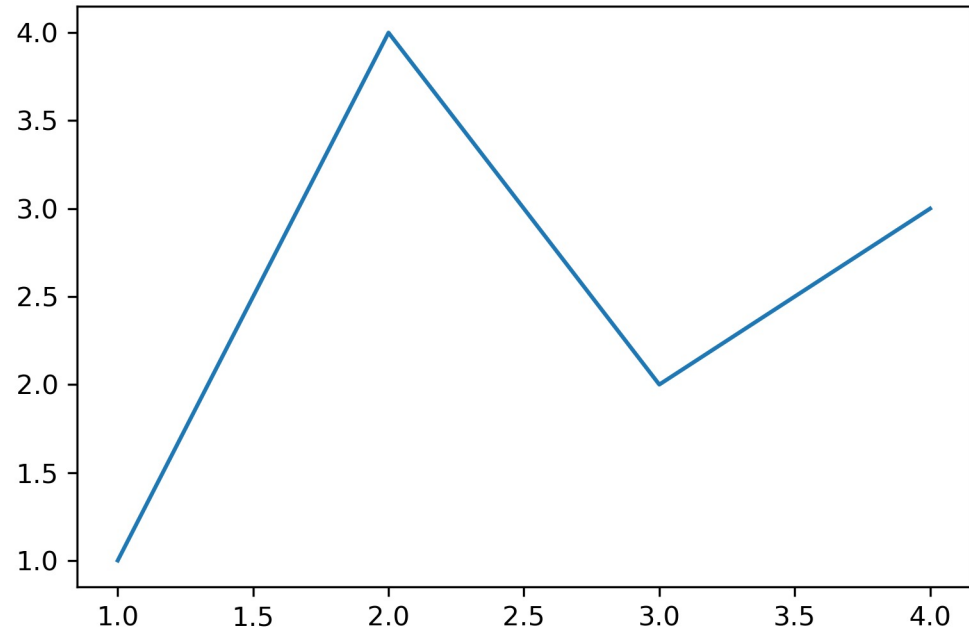
In this course we will use Python to generate data visualizations. The LSI students should have a running system since this setup is used in other courses as well.

Follow the instructions on the [DataVis_Setup.pdf](#) to install the required software.

Matplotlib

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

fig, ax = plt.subplots() # Create a figure containing a single axes.
ax.plot([1, 2, 3, 4], [1, 4, 2, 3]) # Plot some data on the axes.
```



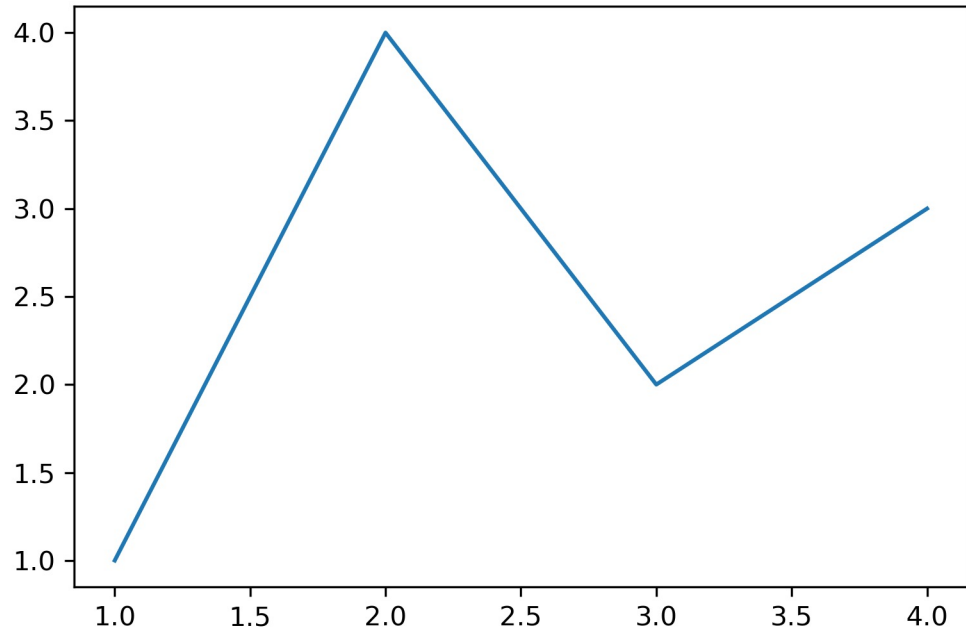
matplotlib user guide:

<https://matplotlib.org/stable/tutorials/introductory/usage.html>

Matplotlib

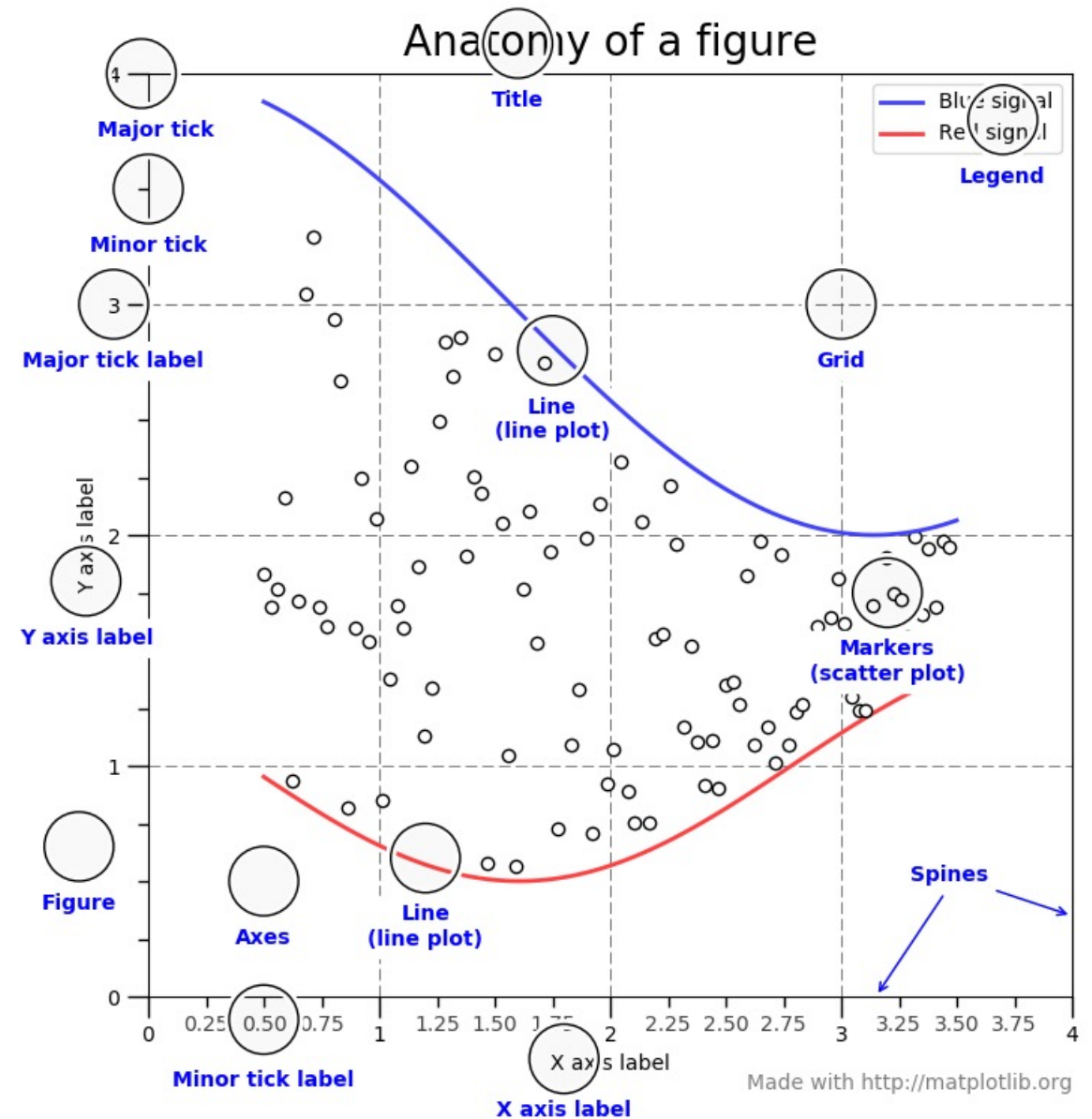
```
import matplotlib.pyplot as plt
import numpy as np

fig, ax = plt.subplots() # Create a figure containing a single axes.
ax.plot([1, 2, 3, 4], [1, 4, 2, 3]) # Plot some data on the axes.
```



matplotlib user guide:

<https://matplotlib.org/stable/tutorials/introductory/usage.html>



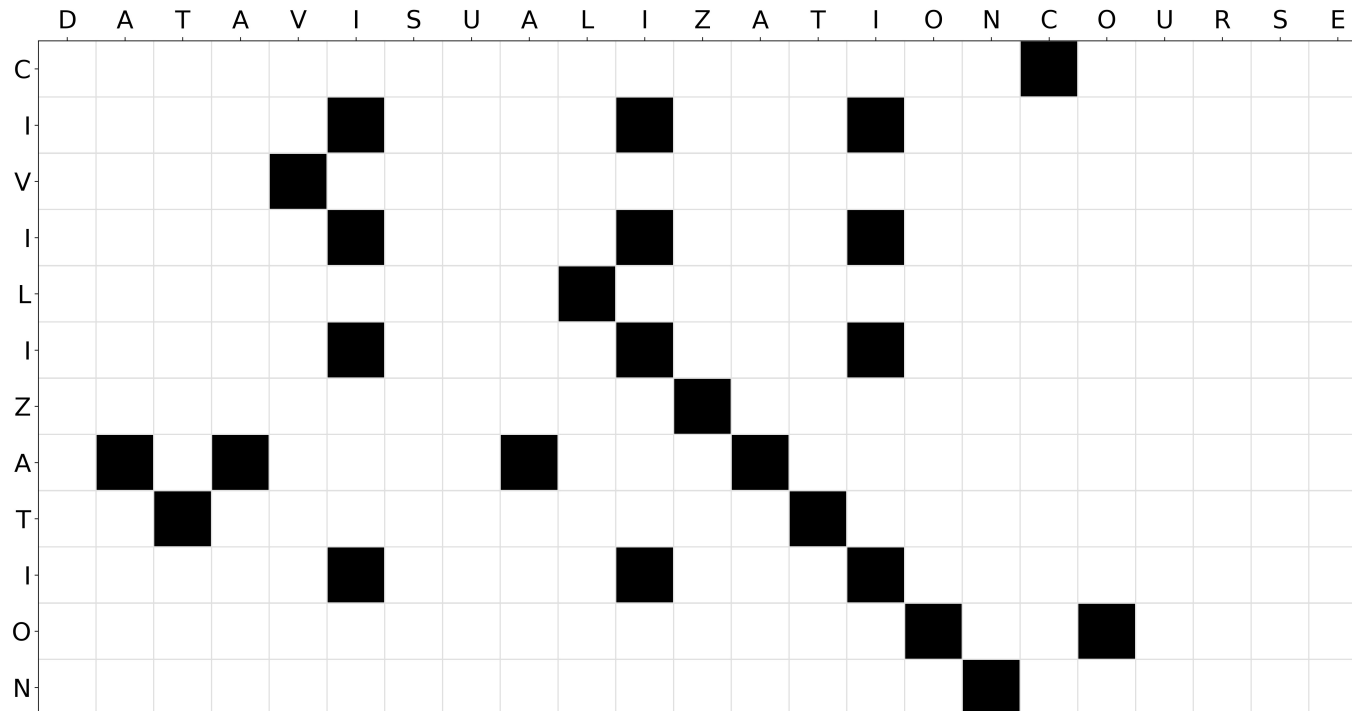
Programming Exercise

- Create A Dot Plot From Scratch Using Python/matplotlib

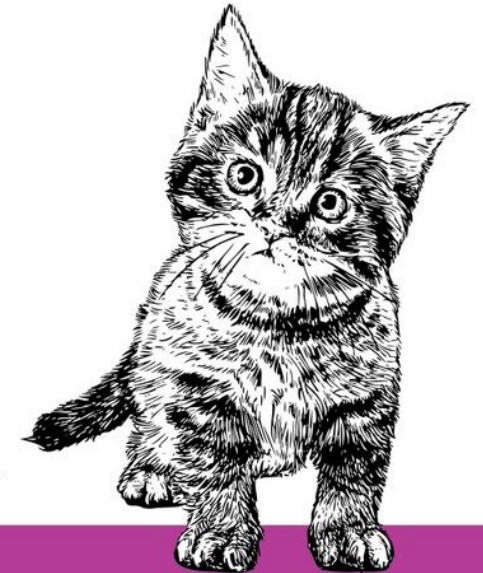
A Dot Plot Visualizes Sequence Similarity

Matplotlib user guide:

<https://matplotlib.org/stable/tutorials/introductory/usage.html>



How to actually learn any new programming concept



Essential

Changing Stuff and
Seeing What Happens

ORLY?

@ThePracticalDev

<https://twitter.com/ThePracticalDev/status/720257210161311744>

Data Visualization

02

Prof. Dr. Phillipp Torkler

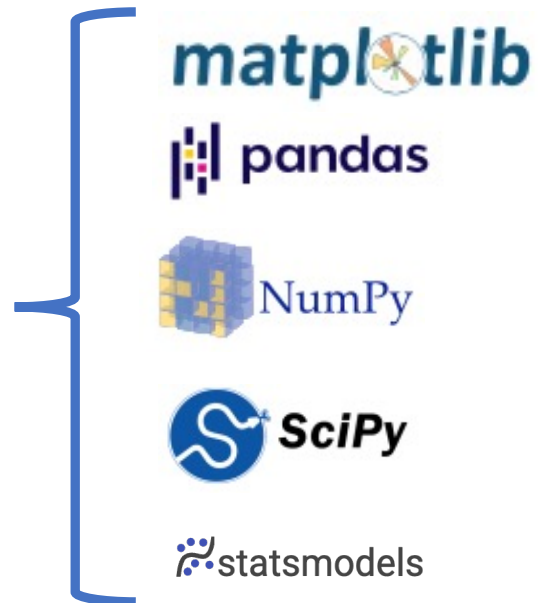
Technical Requirements

Python And R Packages In Comparison

R



Python



Keeping Your Packages Organized



<https://docs.conda.io/en/latest/index.html>

Package, dependency and environment management for any language—Python, R, Ruby, Lua, Scala, Java, JavaScript, C/C++, FORTRAN, and more.

Conda is an open source package management system and environment management system that runs on Windows, macOS and Linux. Conda quickly installs, runs and updates packages and their dependencies. Conda easily creates, saves, loads and switches between environments on your local computer. It was created for Python programs, but it can package and distribute software for any language.

A package and environment management system helps you to manage your environments for different projects!

Reproducible Research / Product Development

Reproducible research is based upon the concept that research results and in particular the underlying data analysis of scientific results are published together with scientific publications. As a consequence, the raw data as well as the analysis are published and shared with the community so that findings and claims that are made in publications can be verified.

As data analysis is going to get more and more complex the need for reproducible research is increased.



knitr/markdown as jupyter notebooks/lab serve the need to make data analysis easily accessible and to agree upon a standard format how data analysis shall be shared and documented.

Guides Are Always Worth A Look! Again and Again

Both, the matplotlib as well as the NumPy guides are written very well. The NumPy guide also has visual representations of the commands that help to understand the behaviour. It is always good to go back to these guides if you forgot what a function really does!



https://numpy.org/doc/stable/user/absolute_beginners.html



<https://matplotlib.org/stable/tutorials/introductory/usage.html>

```
>>> data - ones
array([0, 1])
>>> data * data
array([1, 4])
>>> data / data
array([1., 1.])
```

$$\begin{array}{|c|} \hline \text{data} \\ \hline 1 \\ \hline 2 \\ \hline \end{array} - \begin{array}{|c|} \hline \text{ones} \\ \hline 1 \\ \hline 1 \\ \hline \end{array} = \begin{array}{|c|} \hline 0 \\ \hline 1 \\ \hline \end{array}$$

$$\begin{array}{|c|} \hline \text{data} \\ \hline 1 \\ \hline 2 \\ \hline \end{array} * \begin{array}{|c|} \hline \text{data} \\ \hline 1 \\ \hline 2 \\ \hline \end{array} = \begin{array}{|c|} \hline 1 \\ \hline 4 \\ \hline \end{array}$$

$$\begin{array}{|c|} \hline \text{data} \\ \hline 1 \\ \hline 2 \\ \hline \end{array} / \begin{array}{|c|} \hline \text{data} \\ \hline 1 \\ \hline 2 \\ \hline \end{array} = \begin{array}{|c|} \hline 1 \\ \hline 1 \\ \hline \end{array}$$

NumPy and Matplotlib

NumPy is typically imported via:

```
import numpy as np
```

Generation of 1D Arrays

```
a = np.array([2,1,3,5])  
a
```

```
array([2, 1, 3, 5])
```

.ndim gives the number of dimension/axes of an array

```
a.ndim
```

```
1
```

.shape gives the lengths of the corresponding array dimensions.

```
a.shape
```

```
(4,)
```

.size gives the total number of elements in an array

```
a.size
```

```
4
```

Generation of 2D Arrays

```
a = np.array([[2,1,3,5],[5,6,7,8]])  
a
```

```
array([[2, 1, 3, 5],  
       [5, 6, 7, 8]])
```

.ndim gives the number of dimension/axes of an array

```
a.ndim
```

```
2
```

.shape gives the lengths of the corresponding array dimensions.

```
a.shape
```

```
(2, 4)
```

.size gives the total number of elements in an array

```
a.size
```

```
8
```

Numpy Array Generation with Functions

Numpy has built-in functions to generate multidimensional arrays:

`np.zeros()`, `np.ones()`, `np.full()`

All three functions take a list or array of the shape of the array as first argument. `np.full()` also takes another argument used as fill value:

```
np.zeros(5, dtype=int)
```

```
array([0, 0, 0, 0, 0])
```

```
np.ones([3,2])
```

```
array([[1., 1.],  
       [1., 1.],  
       [1., 1.]])
```

```
np.full([2,5], 10, dtype=int)
```

```
array([[10, 10, 10, 10, 10],  
       [10, 10, 10, 10, 10]])
```

Array Indexing, Slicing And Operations

```
ar
```

```
array([[0, 1, 2, 3, 4],  
       [5, 6, 7, 8, 9]])
```

```
print('ar[0,0]: '+str(ar[0,0]))  
print('ar[1,2]: '+str(ar[1,2]))  
print('ar[0,4]: '+str(ar[0,4]))
```

```
ar[0,0]: 0  
ar[1,2]: 7  
ar[0,4]: 4
```

```
print(ar[0, 2:])  
print(ar[1, 1:3])
```

```
[2 3 4]  
[6 7]
```

```
ar+5
```

```
array([[ 5,  6,  7,  8,  9],  
       [10, 11, 12, 13, 14]])
```

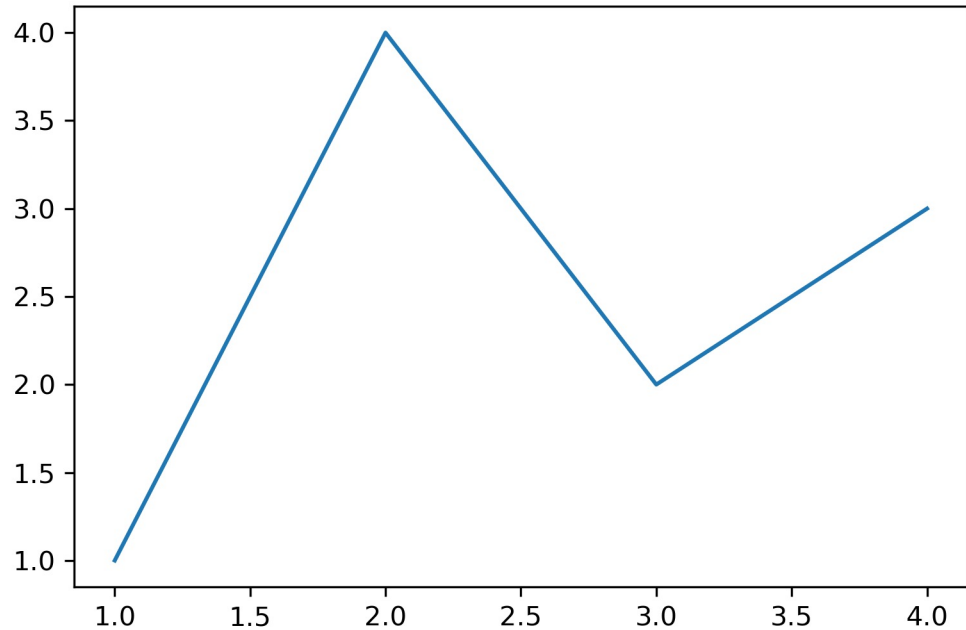
```
ar*2
```

```
array([[ 0,  2,  4,  6,  8],  
       [10, 12, 14, 16, 18]])
```


Matplotlib

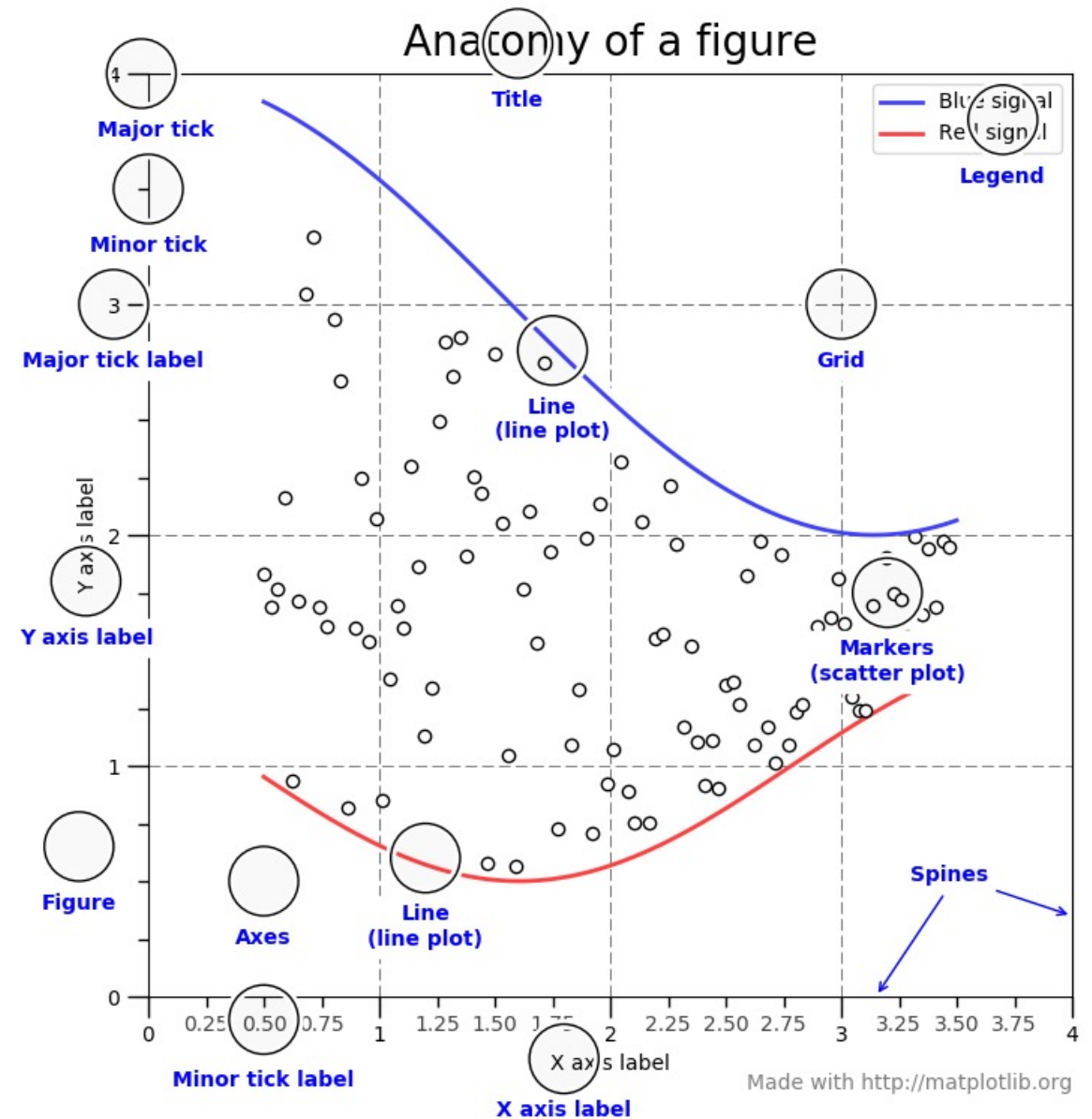
```
import matplotlib.pyplot as plt
import numpy as np

fig, ax = plt.subplots() # Create a figure containing a single axes.
ax.plot([1, 2, 3, 4], [1, 4, 2, 3]) # Plot some data on the axes.
```



matplotlib user guide:

<https://matplotlib.org/stable/tutorials/introductory/usage.html>



Matplotlib

```
x = np.linspace(0,10,100)
print(np.round(x[0:10],2))

[0.  0.1  0.2  0.3  0.4  0.51 0.61 0.71 0.81 0.91]
```

```
y1 = np.sin(x)
y2 = np.sin(x)-np.sin(2*x)+np.cos(x)

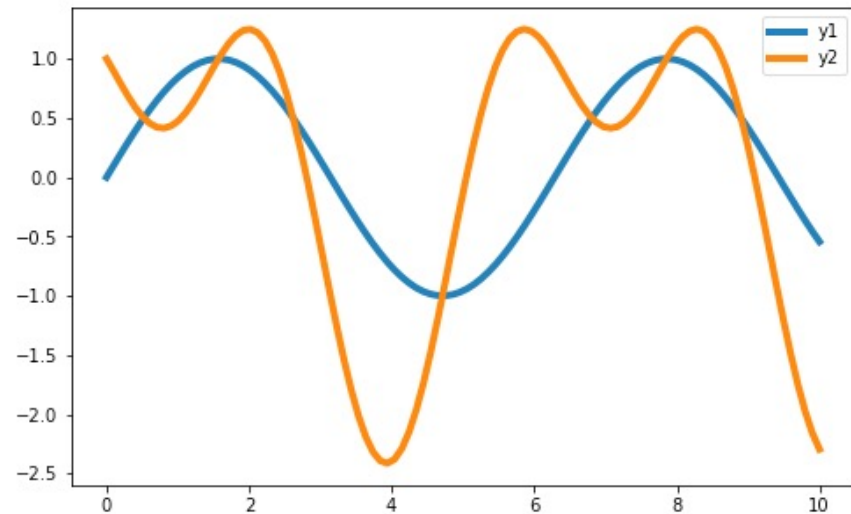
fig, ax = plt.subplots()

fig.set_size_inches(8,5)

ax.plot(x,y1, linewidth=4, label='y1')
ax.plot(x, y2, linewidth=4, label='y2')

ax.legend()
```

<matplotlib.legend.Legend at 0x7fcd68103af0>



```
Z = np.full([6,10], 0, dtype=int)
Z[0,0:3] = np.array([0,0,0])
Z[0,3:6] = np.array([1,1,1])
Z[1,0:6] = np.array([2,2,2,2,2,2])
Z[5,8:10] = np.array([3,3])

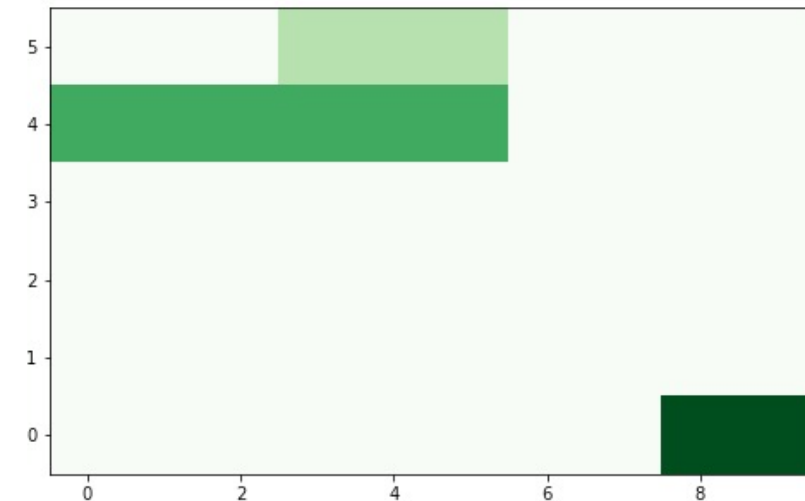
x = np.arange(0, Z.shape[1], 1) # len = 10
y = np.arange(0, Z.shape[0], 1) # len = 6

fig, ax = plt.subplots()
fig.set_size_inches(8,5)

ax.pcolormesh(x, y[:-1], Z, shading='nearest', cmap='Greens', zorder=0)

print(Z)
```

```
[[0 0 0 1 1 1 0 0 0 0]
 [2 2 2 2 2 2 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 3 3]]
```



Programming Exercise

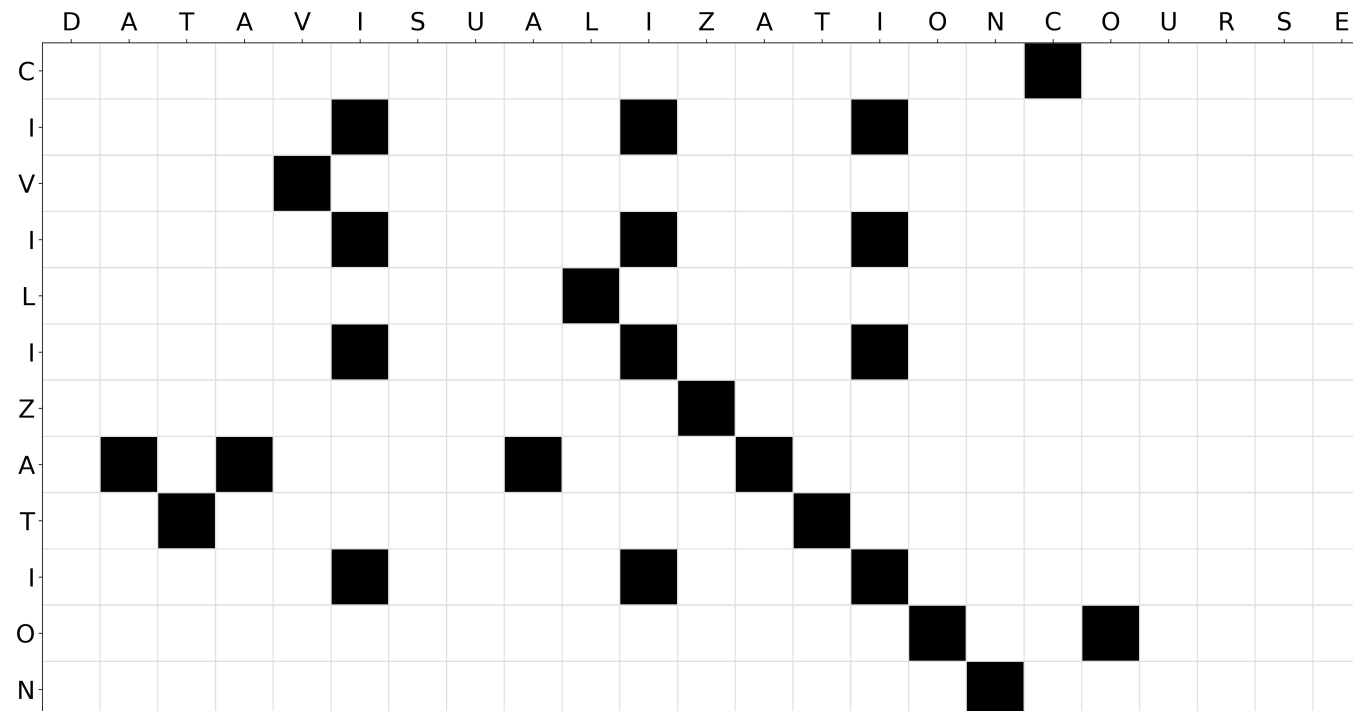
- Create A Dot Plot From Scratch Using Python/matplotlib

A Dot Plot Visualizes Sequence Similarity

A simple and visual way of comparing two sequences s_1 and s_2 of length n and m with each other is a dot plot. The sequences s_1 and s_2 span a $n \times m$ matrix M where a dot is plotted for $M_{i,j}$ if $s_1[i] = s_2[j]$.

$$M_{i,j} = \begin{cases} 1, & \text{if } s_1[i] = s_2[j] \\ 0, & \text{if } s_1[i] \neq s_2[j] \end{cases}$$

The figure below shows a dot plot for $s_1 = \text{CIVILIZATION}$ and $s_2 = \text{DATAVISUALIZATIONCOURSE}$.



Matplotlib user guide:

<https://matplotlib.org/stable/tutorials/introductory/usage.html>

How to actually learn any new programming concept



Essential

Changing Stuff and
Seeing What Happens

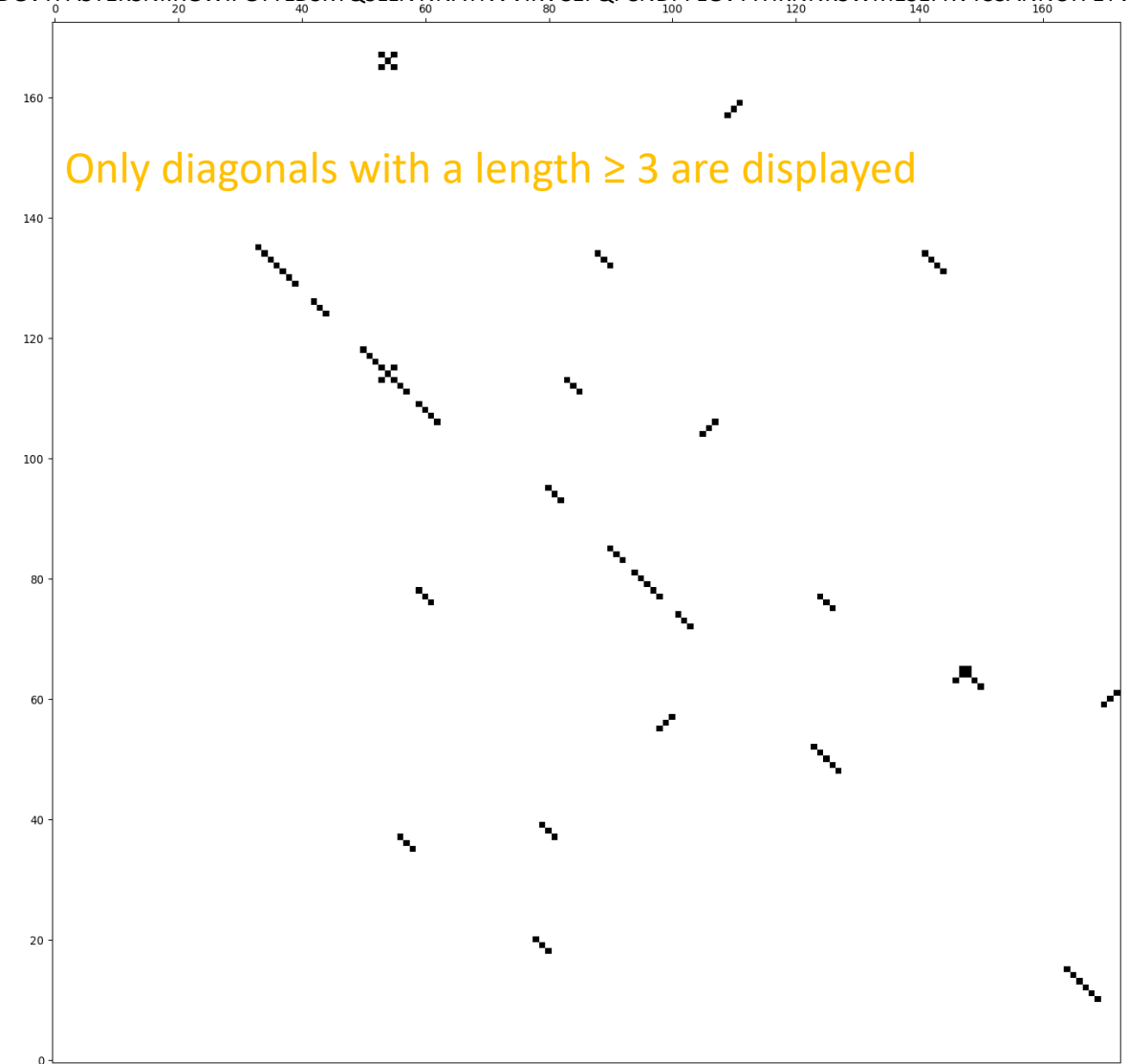
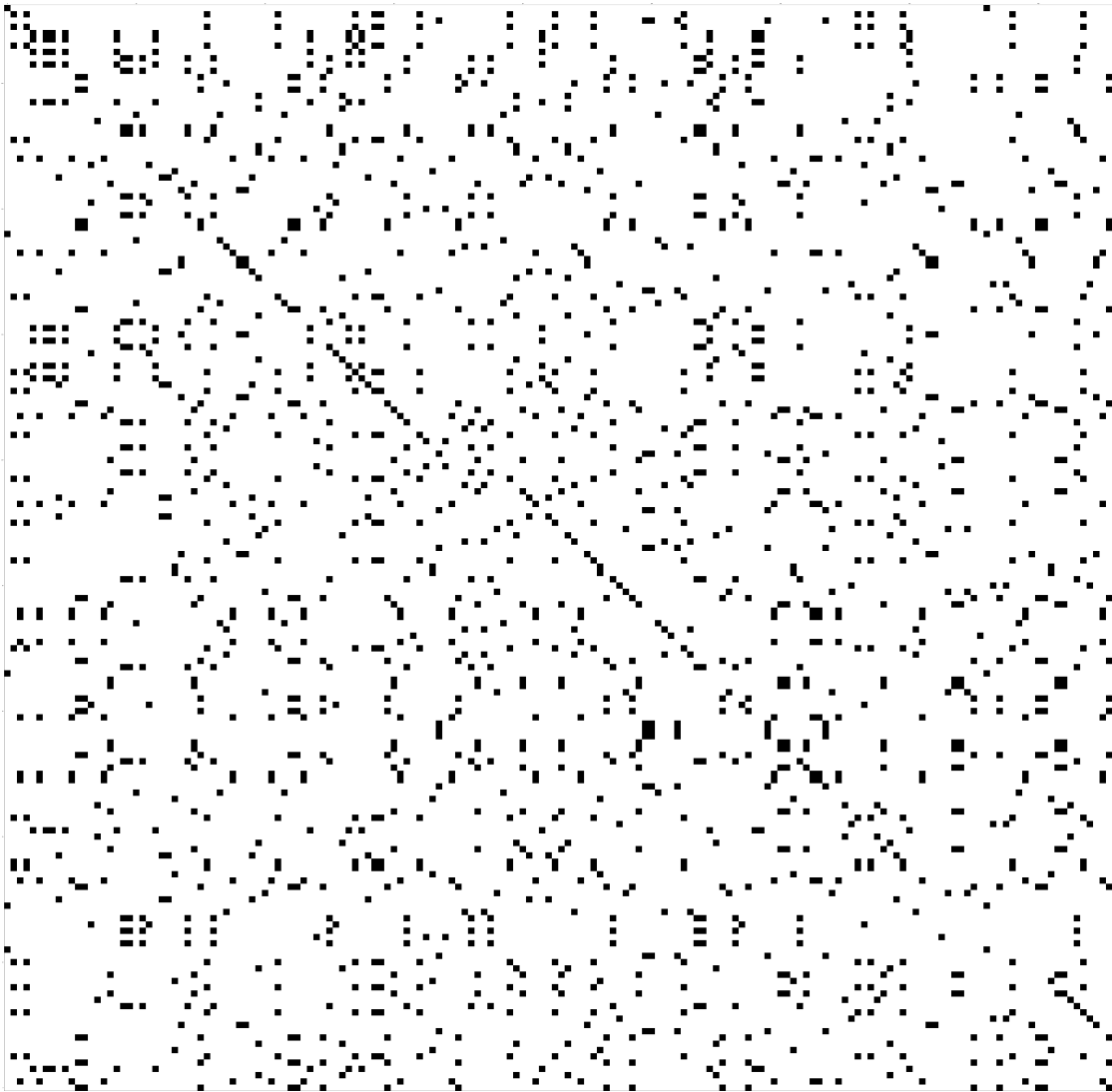
ORLY?

@ThePracticalDev

<https://twitter.com/ThePracticalDev/>

Dot Plots Without Length Threshold Are Overcrowded

```
s1 = 'MFIFLLFLTSTGSDLRCTTFDDVQAPNYTQHTSSMRGVVYPDEFIRSDTLYLTDLFLPFYSNVTGFHTINHTFGNPVVPFKDGIYFAATEKSNVVRGWVFGSTMNNKSQSVIIINNSTNVVIRACNFELCDNPFFAVSKPMGTQHTMIFDNAFNCTFEYISDAFSLDVS'  
s2 = 'MFVFLVLLPLVSSQCYNLTTTRTQLPPAYTNSFTRGVVYPDKVFRSSVLHSTQDLFLPFFSNVTWFHAIHVSGTNGTKRFDNPVLPFNDGVYFASTEKSNIIRGWIFGTTLD SKTQSLIVN NATNVV IKVCFQFCNDPFLGVVYHKNNKSWMESEFRVYSSANNCTFEYVSQ'
```



Reading Exercise

1. Describe the impact of graphical elements to the ability to assess the relative magnitude between elements.
2. Why do we need to know about the impact of graphical elements?

Notes

Notes

Data Visualization

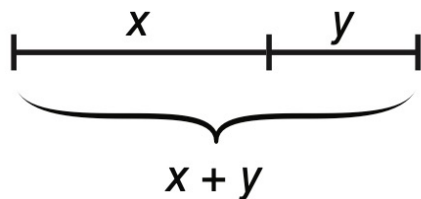
02

Prof. Dr. Phillipp Torkler

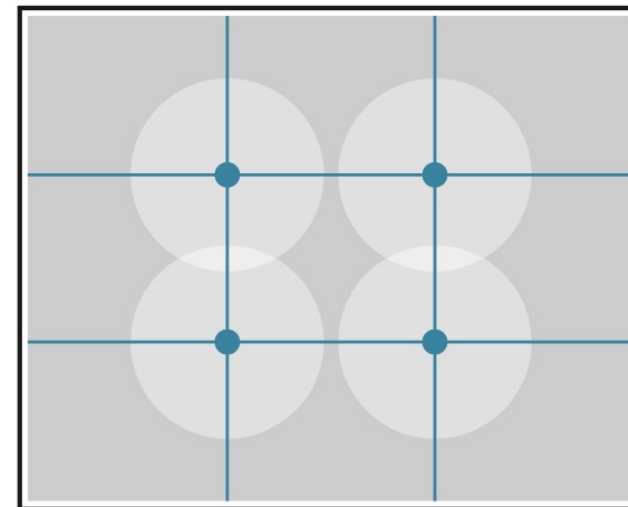
Layout and White Space

Layout and the Golden Ratio

- What is layout? Layout is the act of arranging text and images on the page/slide/poster according to an overall aesthetic scheme and for the purpose of clarifying a presentation.
- Why do we need layout? Well-structured content can guide readers through complex information and unstructured material can confuse or, worse, it needs more time for the reader arrange the items into correct order rather than focus on the message.
- A helpful guidedance for the arrangement of elements in a figure is the 'rule of thirds' also known as the 'golden ratio' that became popular during artists of the renaissance (and remains popular since).

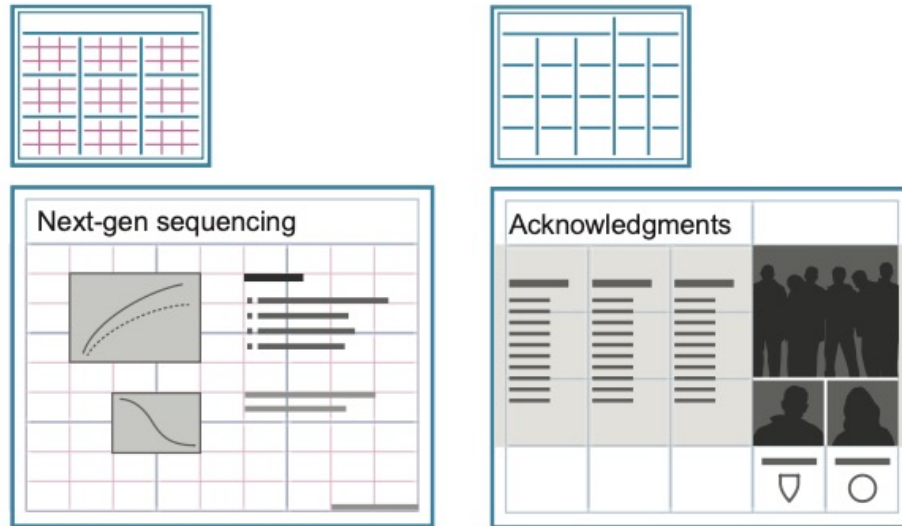


The golden section is a line segment divided by the golden ratio 13:8 such that $(x + y)$ is to x as x is to y .

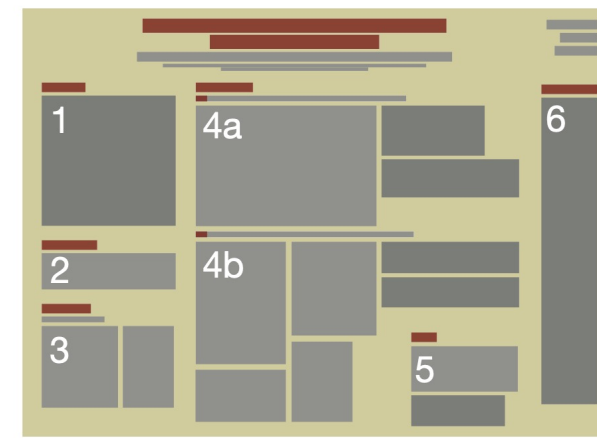
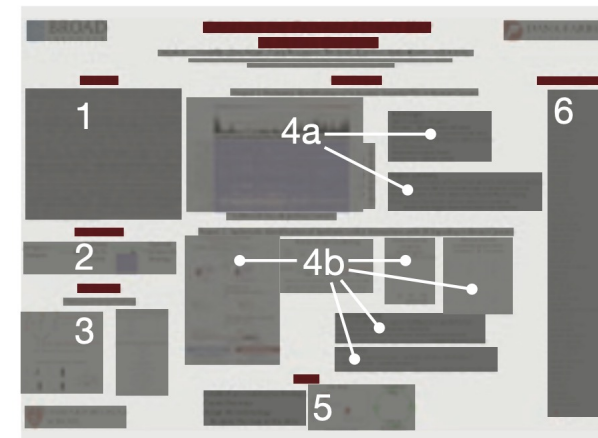


Using Grid Lines Helps To Arrange And Structure Content

- Grids help you to anchor elements and to align elements properly.
- Gestalt principles can help you to group elements and give structure.
- White spaces are an important tool to properly structure content.



Wong, B. "Layout" *Nature Methods* (2011)

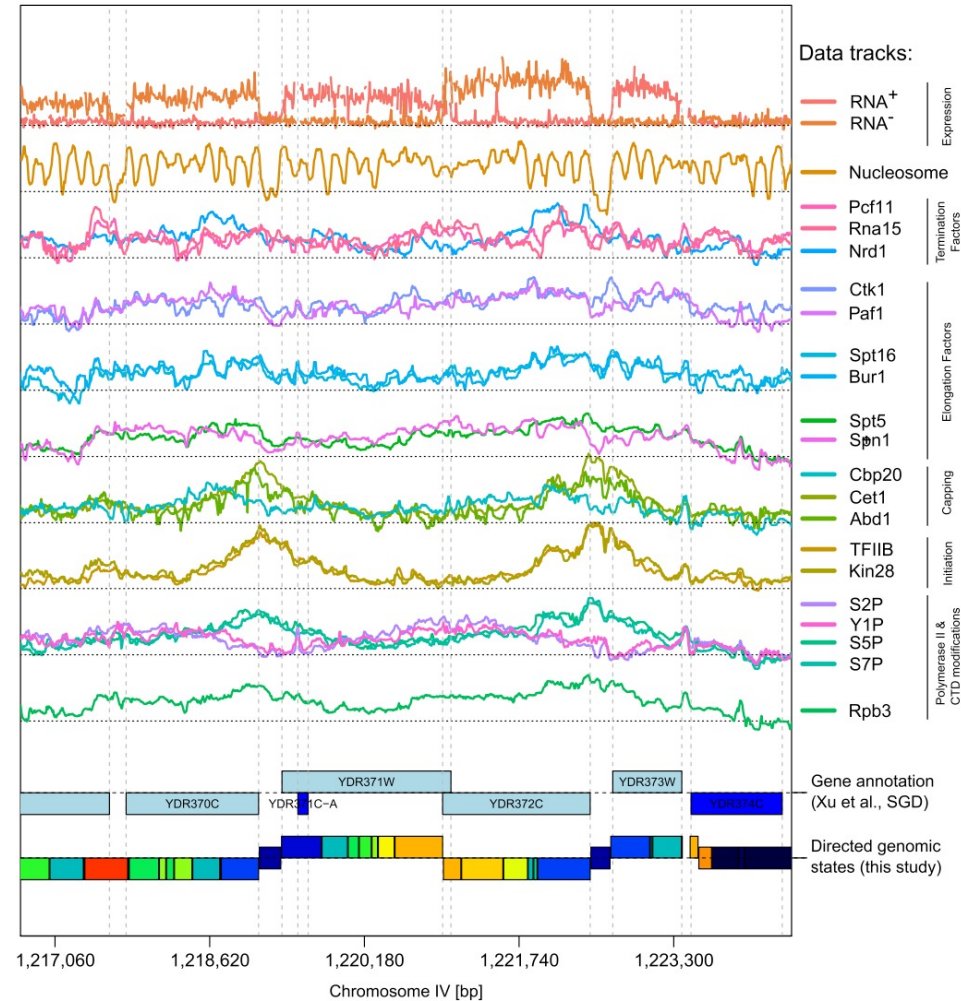


Wong, B. "Negative space" *Nature Methods* (2011)

Reading Exercise

1. Why is data visualization challenging in modern life science research?
2. What can be done to reduce visual complexity and what are potential drawbacks?

Summarization Example



Chromatin immunoprecipitation in conjunction with hidden Markov models (HMMs) segment the genome into discrete states that can be related to DNA-associated protein complexes.

Directed genomic states:



Programming Exercise

- Visualize k-mer Counts For Genomic Regions

Visualize k-mer Counts For Genomic Regions



Essential

Googling the Error Message

ORLY?

The Practical Developer @ThePracticalDev

<https://github.com/thepracticaldev>

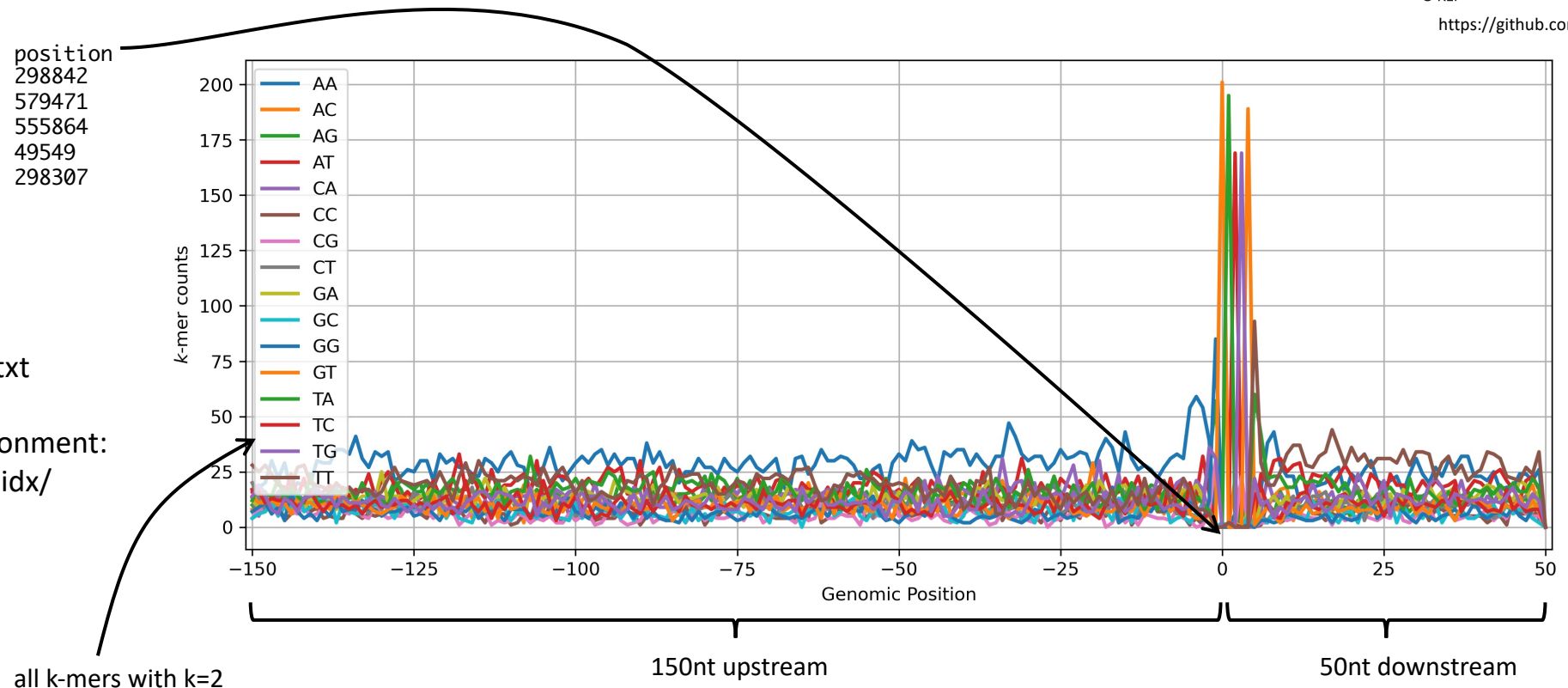
Inputs:

- A text file with genomic locations
- A genome stored in a multiple fasta file
- A value for k to define the k-mer length
- An integer that defines the number of nucleotides that are shown upstream of the genomic locations
- An integer that defines the number of nucleotides that are shown downstream of the genomic locations

chr	strand	position
chrXIII	+	298842
chrXIV	+	579471
chrX	+	555864
chrVII	+	49549
chrXIII	+	298307

Go to iLearn and download:

- Genome: genome_yeast.fsa
- Positions: 03_genomic_data.txt
- Install pyfaidx into your environment:
<https://pypi.org/project/pyfaidx/>



Data Visualization

04

Prof. Dr. Phillipp Torkler

Programming Exercise

- Visualize k-mer Counts For Genomic Regions

Visualize k-mer Counts For Genomic Regions



Essential

Googling the Error Message

ORLY?

The Practical Developer @ThePracticalDev

<https://github.com/thepracticaldev>

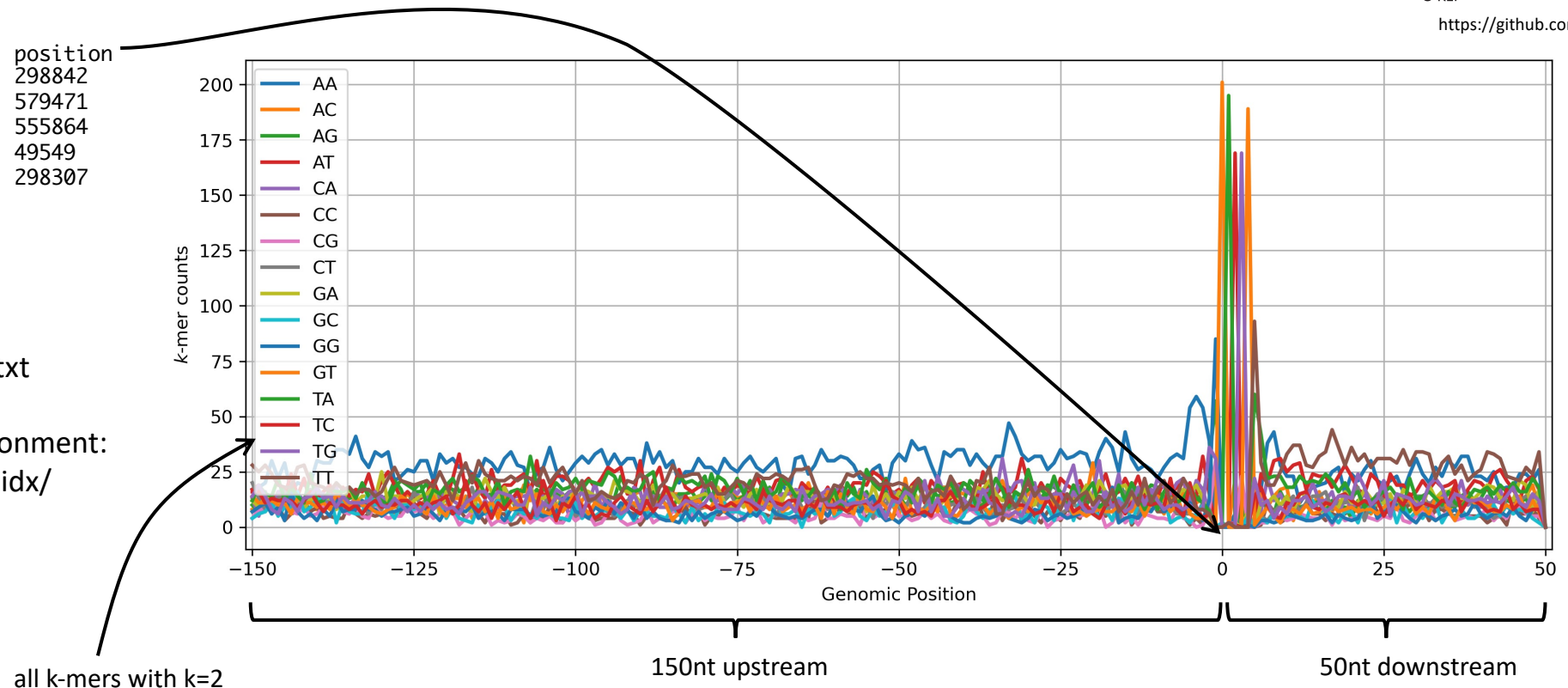
Inputs:

- A text file with genomic locations
- A genome stored in a multiple fasta file
- A value for k to define the k-mer length
- An integer that defines the number of nucleotides that are shown upstream of the genomic locations
- An integer that defines the number of nucleotides that are shown downstream of the genomic locations

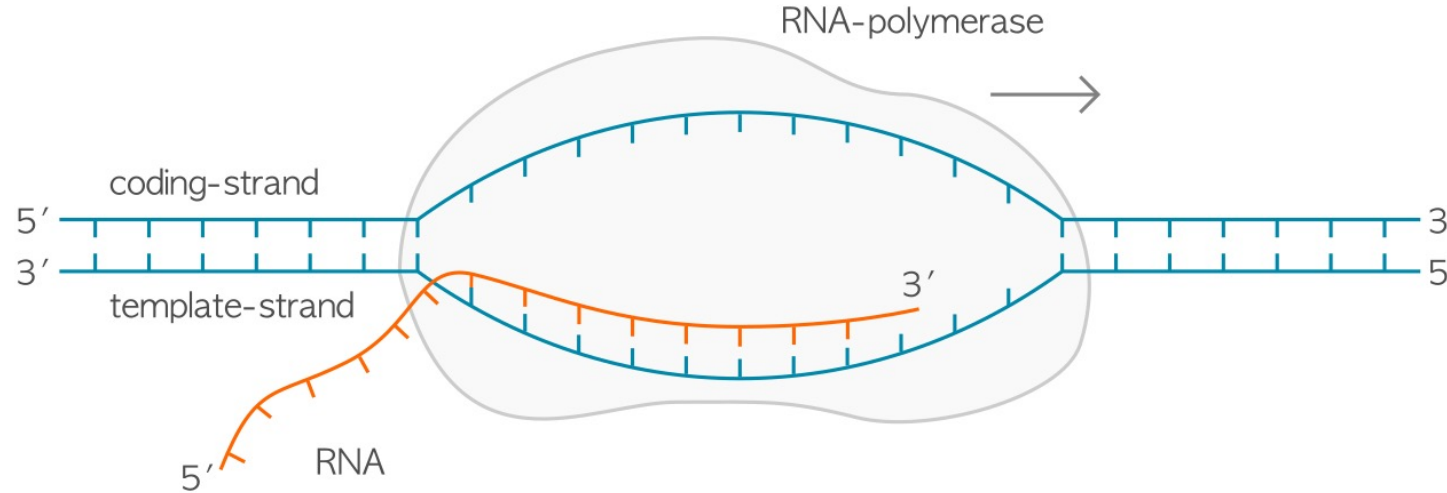
chr	strand	position
chrXIII	+	298842
chrXIV	+	579471
chrX	+	555864
chrVII	+	49549
chrXIII	+	298307

Go to iLearn and download:

- Genome: genome_yeast.fsa
- Positions: 03_genomic_data.txt
- Install pyfaidx into your environment:
<https://pypi.org/project/pyfaidx/>



The Strand Matters

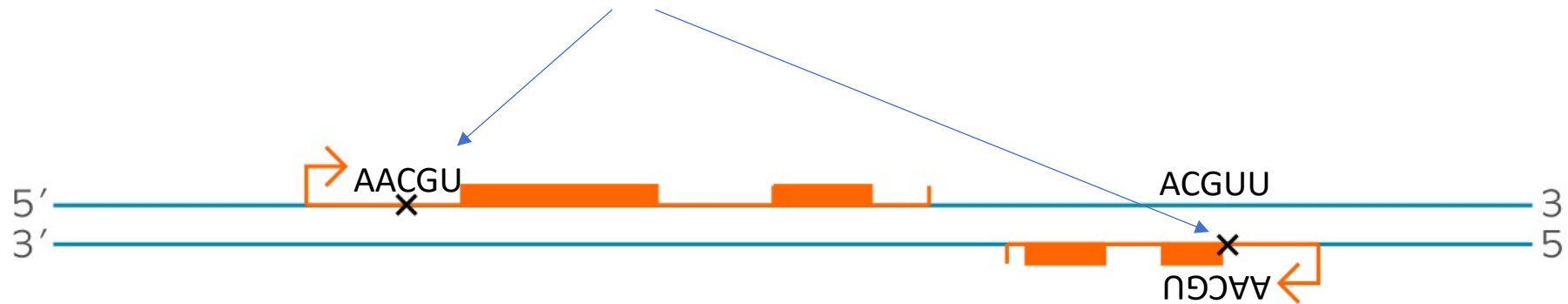


The RNA-polymerase reads in 3'-5' direction of the template-strand (minus strand). The resulting RNA molecule that grows in 5'-3' direction has the same nucleotide sequence as the coding-strand (plus strand) except for the T/U change. Consequently, the strand-information matters to obtain the correct sequence information.

Typically, the coding-strand sequences are saved as genome sequences.

The Strand Matters: Example

Let's assume the marked positions are binding sites of an RNA binding protein that binds to the specific RNA sequence AACGU



Reading Exercise

1. Explain what “Gestalt principles” are.
2. What is meant by “the whole is ‘other’ than the sum of its parts”?
3. What differentiates connection and enclosure from similarity and proximity?

Programming Exercise

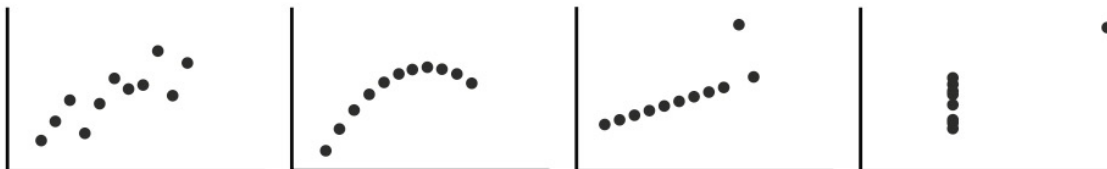
Read And Plot Data

- Go to iLearn and download '04_data.csv'
- Load the data into a Jupyter notebook
- Plot the data in 1D. Use $y=0$ for the x1 column and $y=1$ for the x2 column and plot the data.
 - Do you spot a difference between x1 and x2?
- Calculate the means from x1 and x2 and compare them. Do they appear to be different?
- Go to <https://seaborn.pydata.org/installing.html> and install the seaborn package.
- Create a boxplot of x1 and x2 using seaborn. Do the distributions appear to be different?
 - Check <https://seaborn.pydata.org/generated/seaborn.boxplot.html#seaborn.boxplot> for further guidance

Ancombe's Quartet

I		II		III		IV	
x	y	x	y	x	y	x	y
10	8.04	10	9.14	10	7.46	8	6.58
8	6.95	8	8.14	8	6.77	8	5.76
13	7.58	13	8.74	13	12.74	8	7.71
9	8.81	9	8.77	9	7.11	8	8.84
11	8.33	11	9.26	11	7.81	8	8.47
14	9.96	14	8.10	14	8.84	8	7.04
6	7.24	6	6.13	6	6.08	8	5.25
4	4.26	4	3.10	4	5.39	19	12.5
12	10.84	12	9.13	12	8.15	8	5.56
7	4.82	7	7.26	7	6.42	8	7.91
5	5.68	5	4.74	5	5.73	8	6.89

Raw data as well as summary statistics can be misleading if you rely exclusively on these. The shown four data sets known as Ancombe's quartet have the same means, standard deviations, correlation coefficients, regression lines etc. but still the data sets are fundamentally different as can be seen by visualization. During data exploration don't rely on a single perspective.

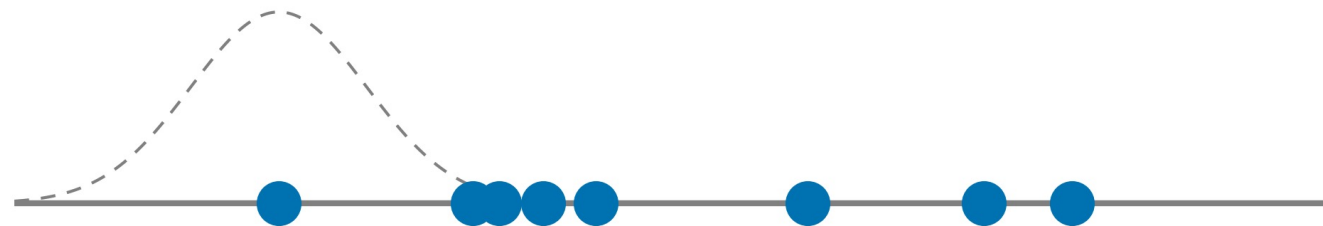


Kernel Density Estimation In A Nutshell



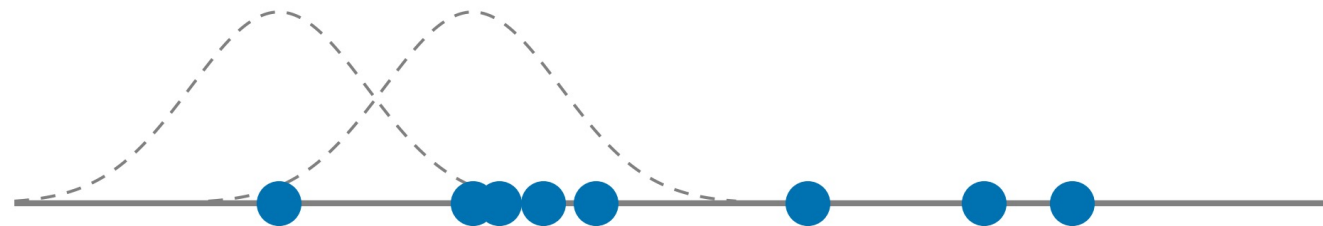
The density estimate at each point is the average contribution from each of the kernels at that point.

Kernel Density Estimation In A Nutshell



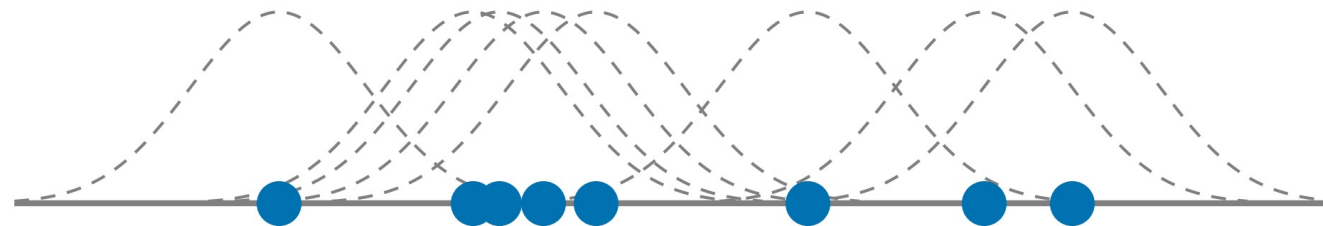
The density estimate at each point is the average contribution from each of the kernels at that point.

Kernel Density Estimation In A Nutshell



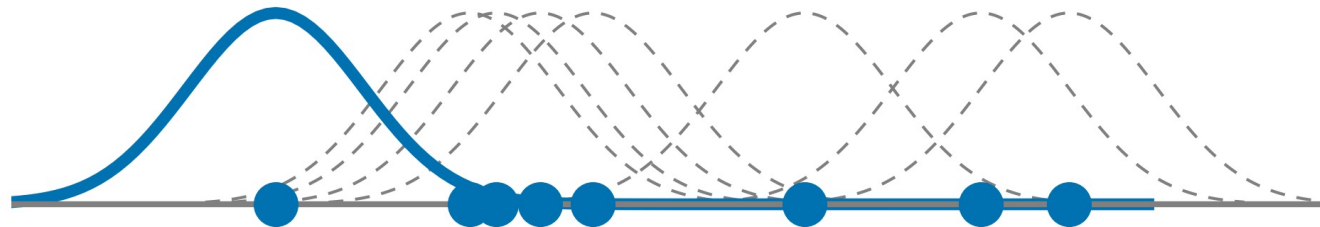
The density estimate at each point is the average contribution from each of the kernels at that point.

Kernel Density Estimation In A Nutshell



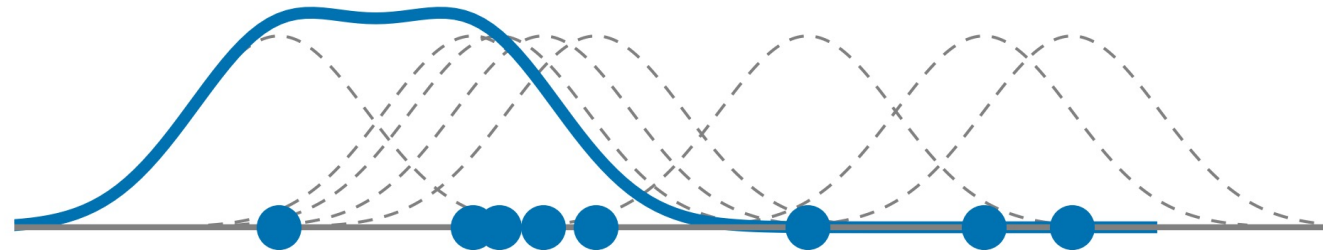
The density estimate at each point is the average contribution from each of the kernels at that point.

Kernel Density Estimation In A Nutshell



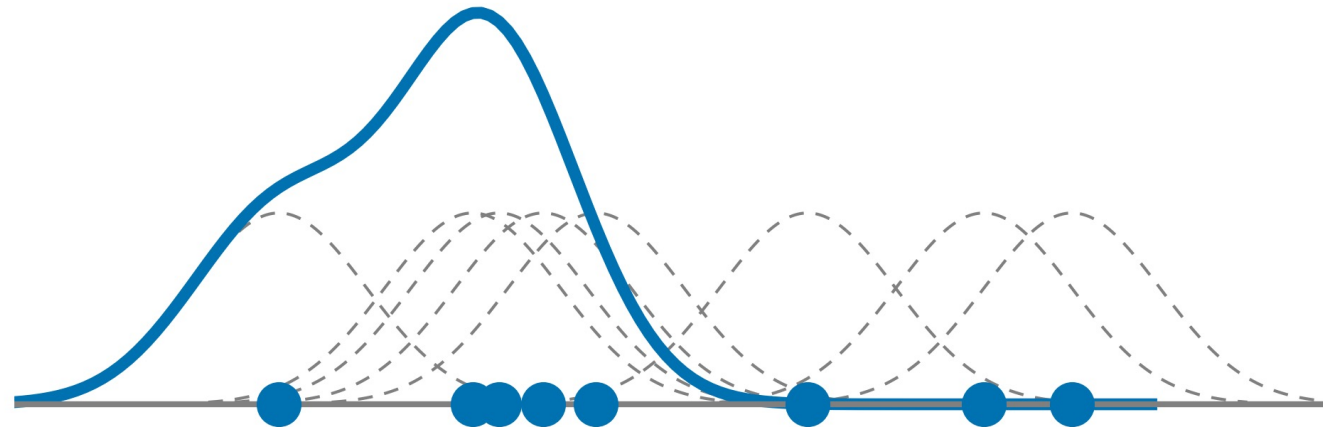
The density estimate at each point is the average contribution from each of the kernels at that point.

Kernel Density Estimation In A Nutshell



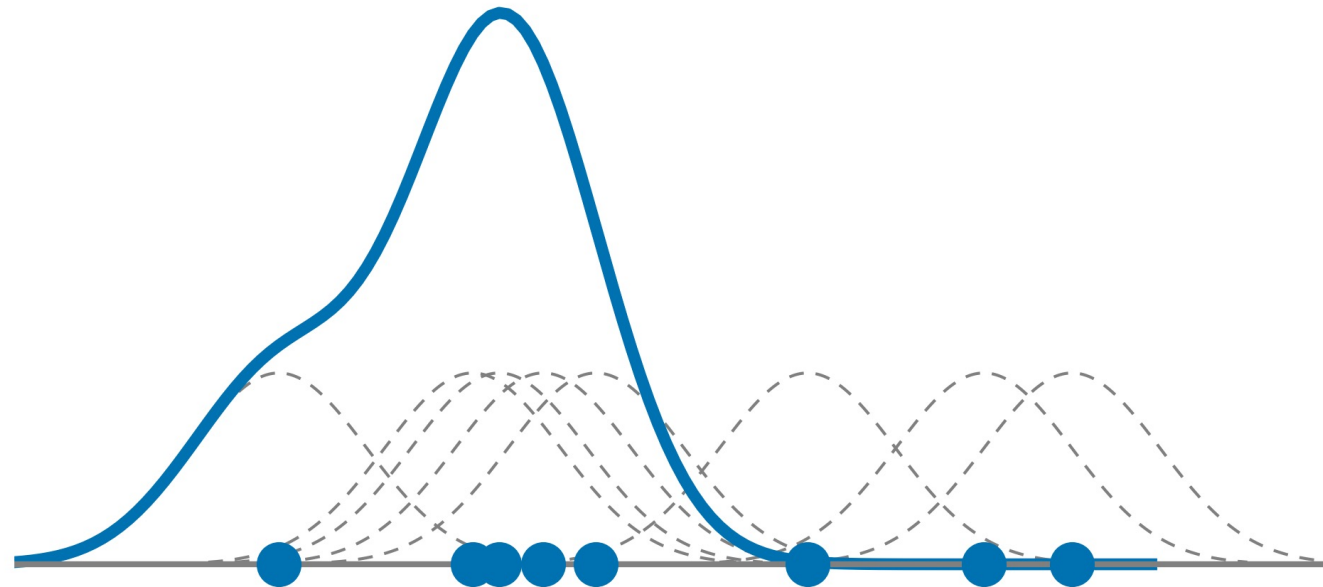
The density estimate at each point is the average contribution from each of the kernels at that point.

Kernel Density Estimation In A Nutshell



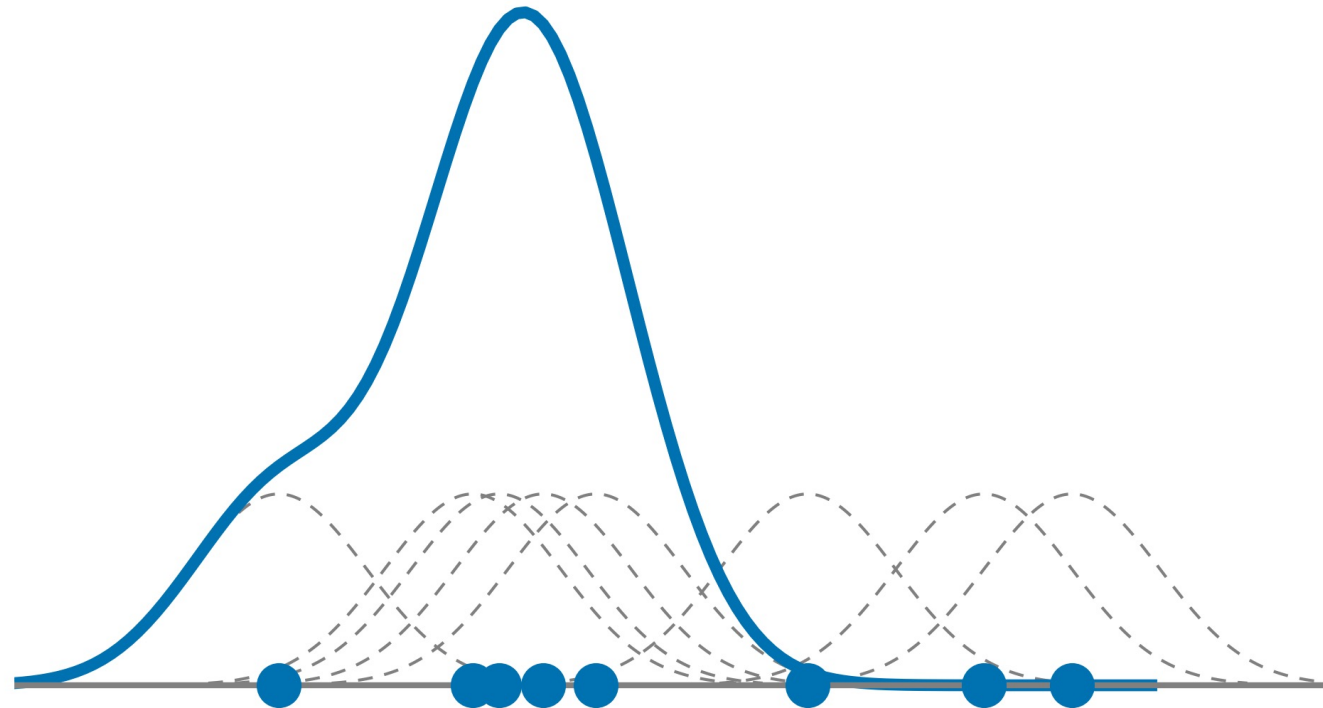
The density estimate at each point is the average contribution from each of the kernels at that point.

Kernel Density Estimation In A Nutshell



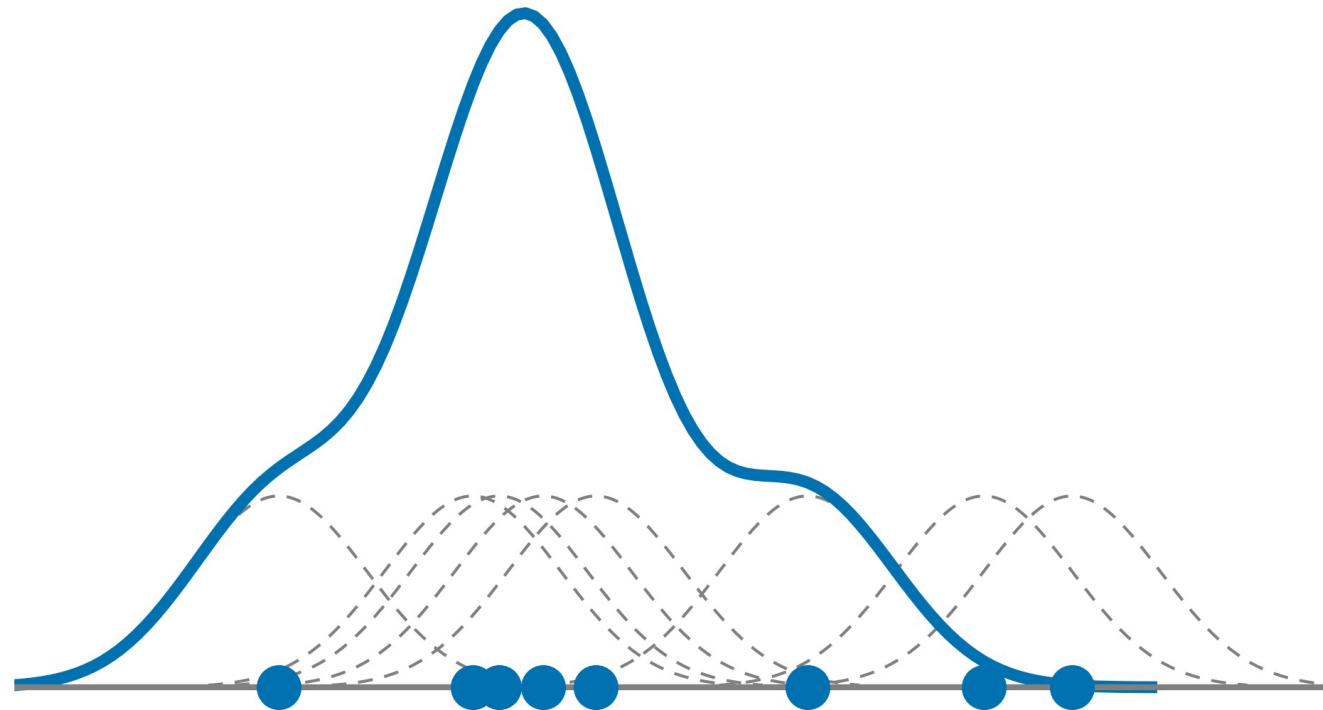
The density estimate at each point is the average contribution from each of the kernels at that point.

Kernel Density Estimation In A Nutshell



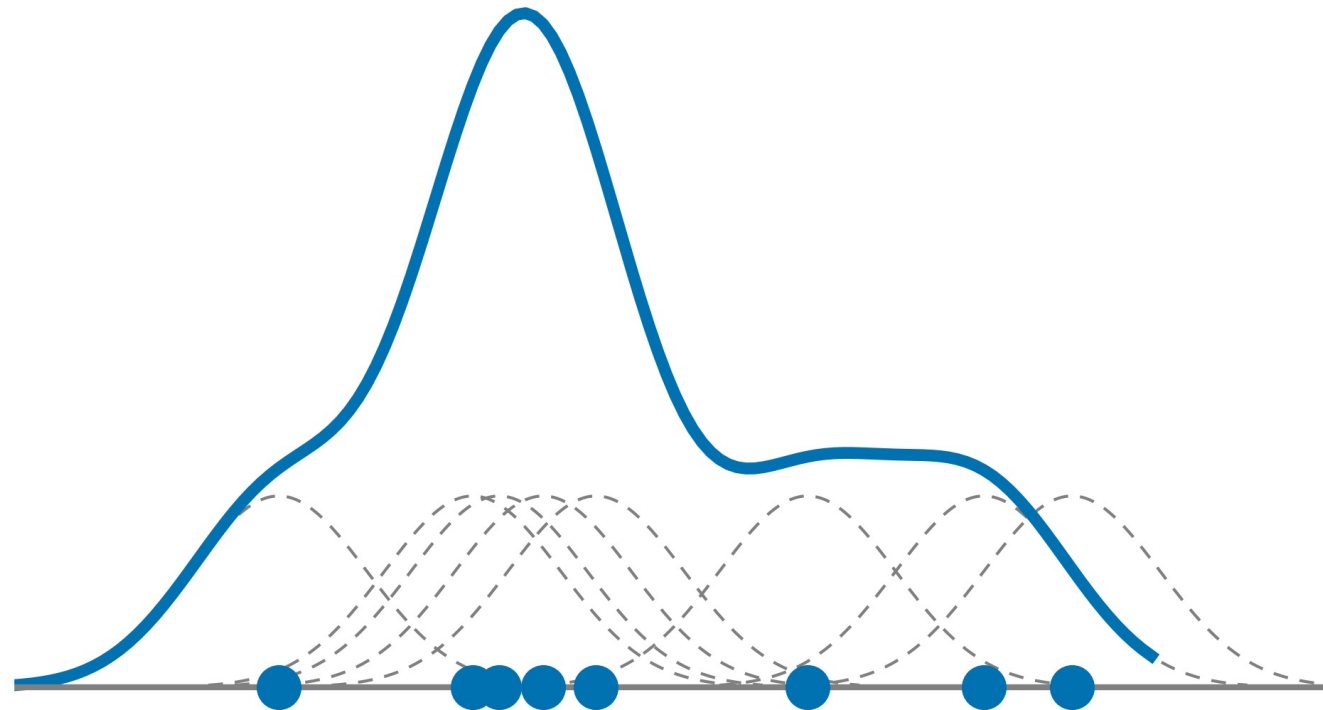
The density estimate at each point is the average contribution from each of the kernels at that point.

Kernel Density Estimation In A Nutshell



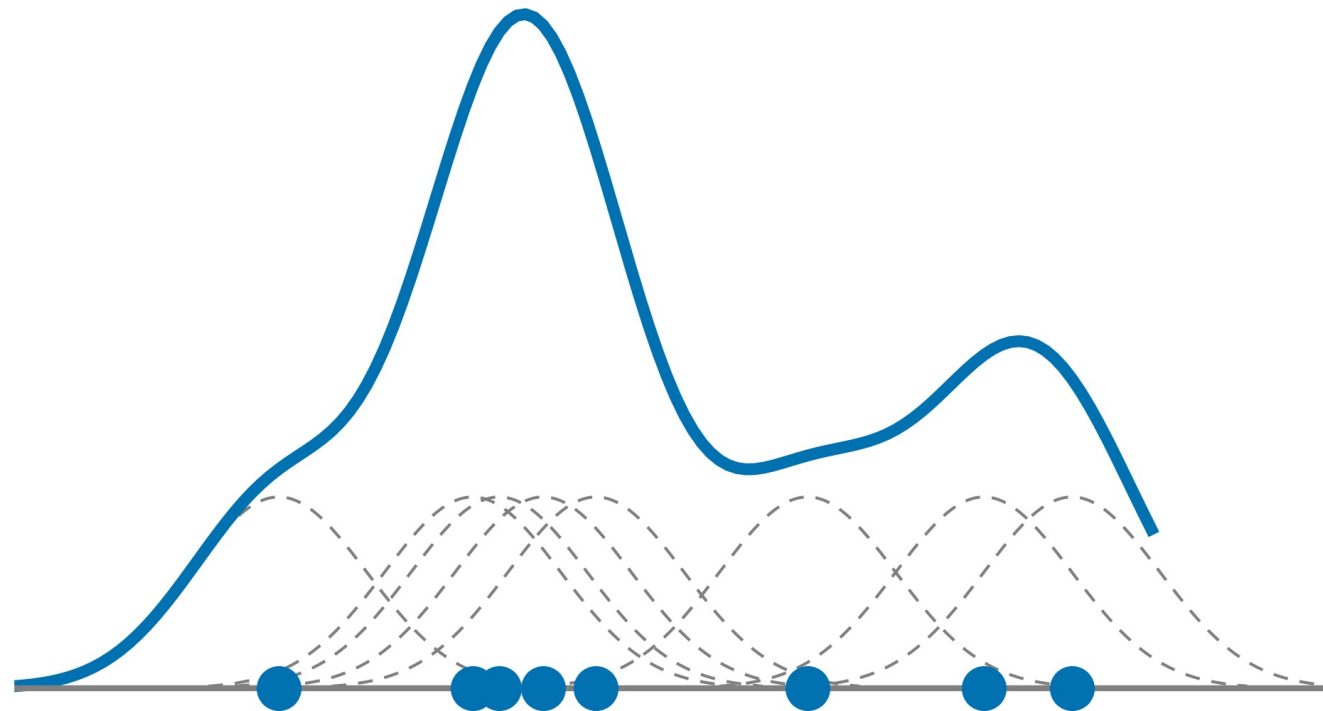
The density estimate at each point is the average contribution from each of the kernels at that point.

Kernel Density Estimation In A Nutshell



The density estimate at each point is the average contribution from each of the kernels at that point.

Kernel Density Estimation In A Nutshell



The density estimate at each point is the average contribution from each of the kernels at that point.

Read And Plot Data

- Go to iLearn and download '04_data.csv'
- Load the data into a Jupyter notebook
- Plot the data in 1D. Use $y=0$ for the x1 column and $y=1$ for the x2 column and plot the data.
 - Do you spot a difference between x1 and x2?
- Calculate the means from x1 and x2 and compare them. Do they appear to be different?
- Go to <https://seaborn.pydata.org/installing.html> and install the seaborn package.
- Create a boxplot of x1 and x2 using seaborn. Do the distributions appear to be different?
 - Check <https://seaborn.pydata.org/generated/seaborn.boxplot.html#seaborn.boxplot> for further guidance
- Create a violin or kde plot of x1 and x2 using seaborn. What's the difference compared to the previously produced boxplot?

Data Visualization

05

Prof. Dr. Phillipp Torkler

Color

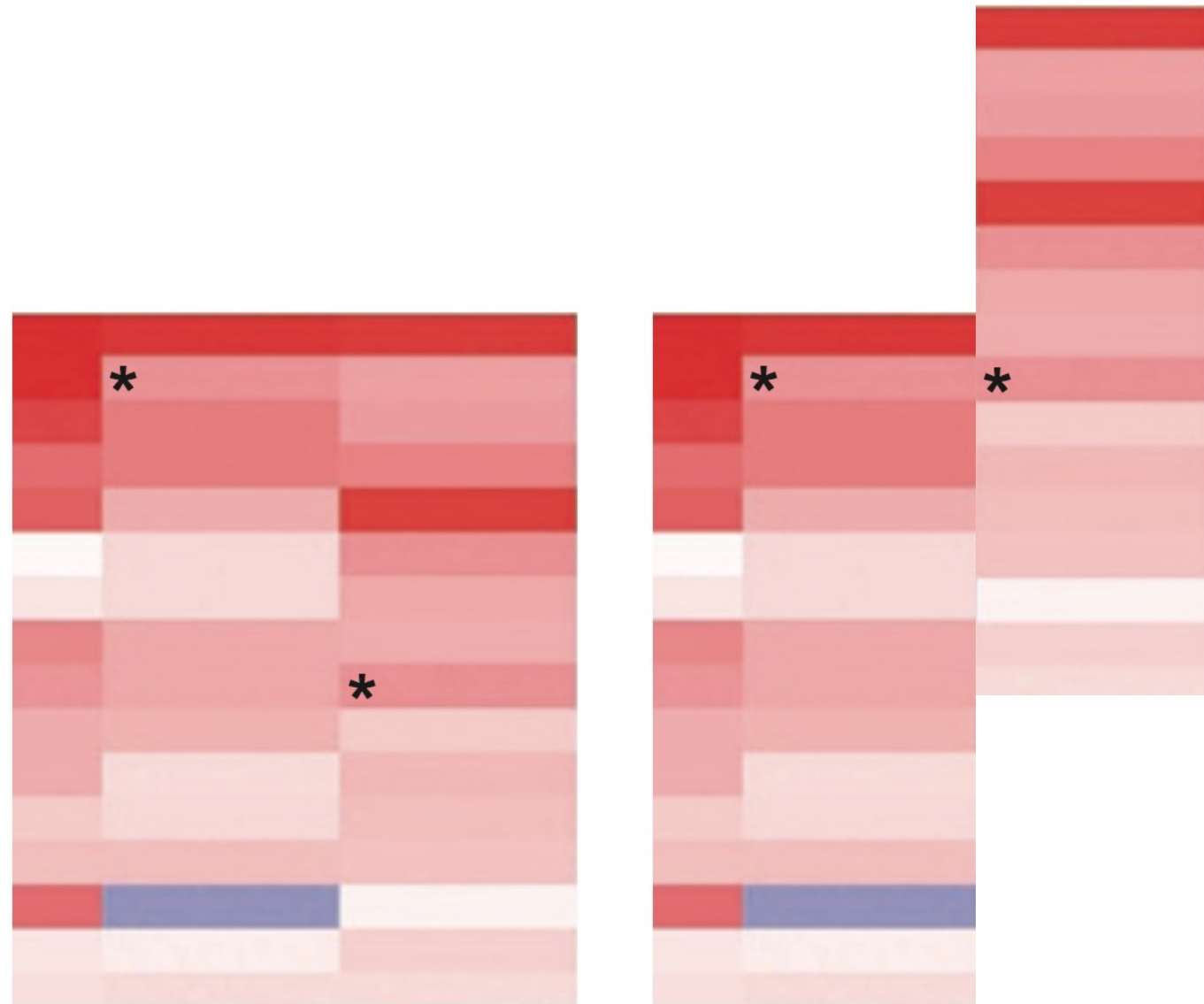
Color



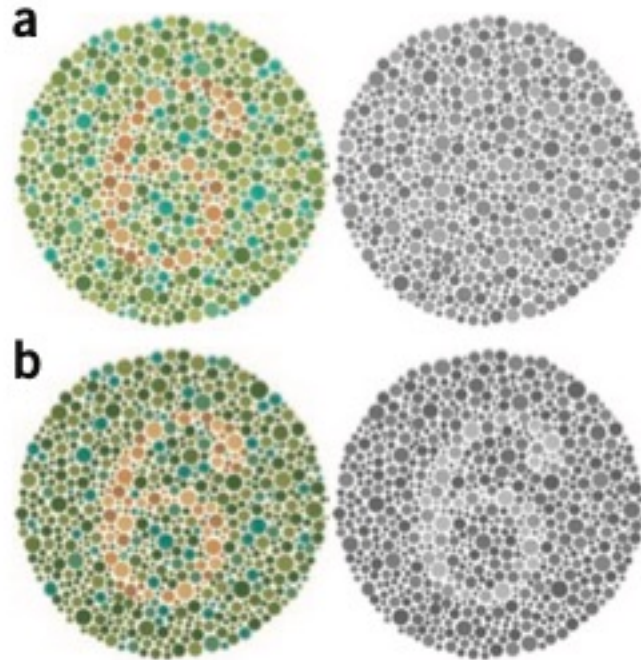
Color



Color



























Color Blindness



Wong, B. "Color blindness" *Nature Methods* (2011)

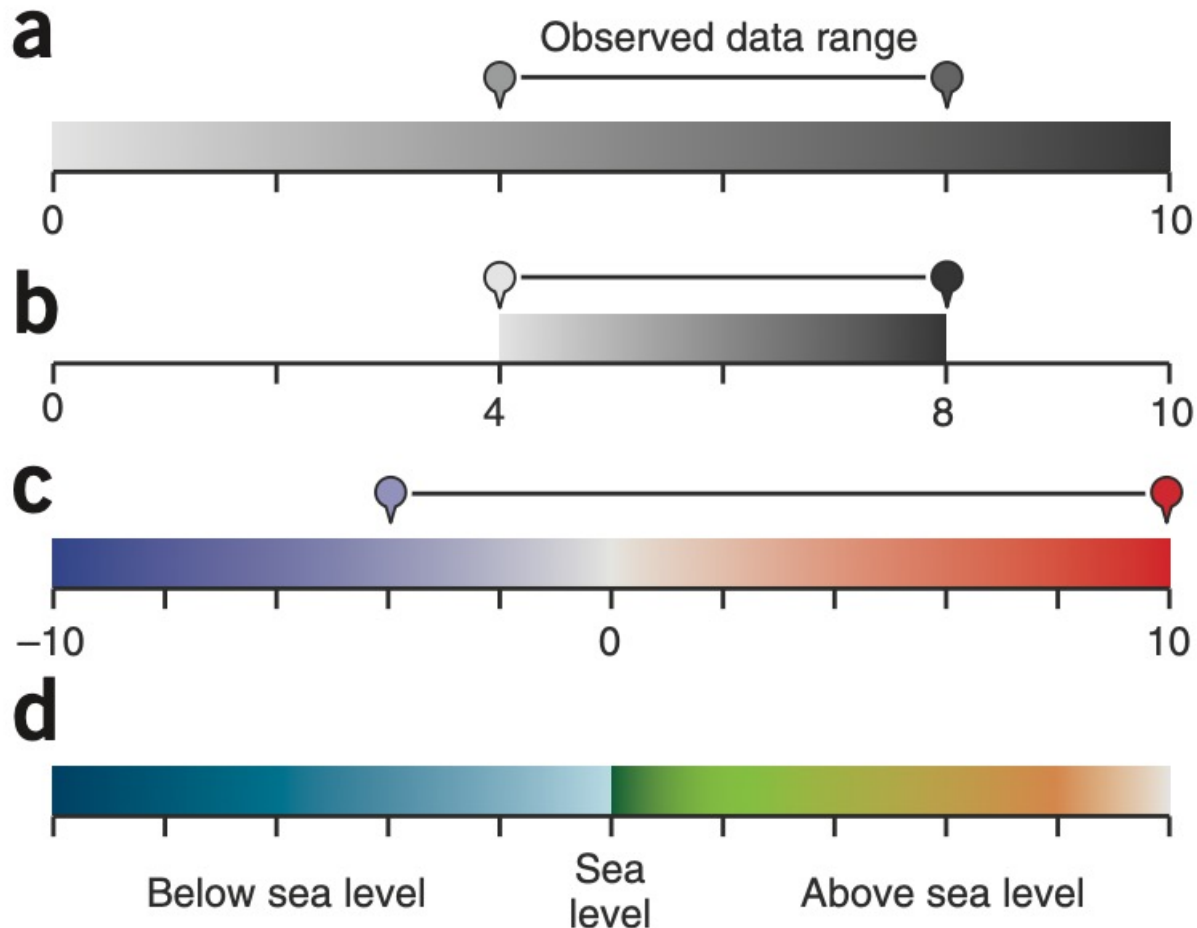
Ishihara color vision test plates. **a)** colors are only different in hue. **b)** colors can be separated easier if the colors also change in lightness and saturation

Color	Color name	RGB (1–255)	CMYK (%)	P	D
	Black	0, 0, 0	0, 0, 0, 100		
	Orange	230, 159, 0	0, 50, 100, 0		
	Sky blue	86, 180, 233	80, 0, 0, 0		
	Bluish green	0, 158, 115	97, 0, 75, 0		
	Yellow	240, 228, 66	10, 5, 90, 0		
	Blue	0, 114, 178	100, 50, 0, 0		
	Vermillion	213, 94, 0	0, 80, 100, 0		
	Reddish purple	204, 121, 167	10, 70, 0, 0		

Wong, B. "Color blindness" *Nature Methods* (2011)

Colors optimized for color-blind individuals. P and D indicate simulated colors as seen by individuals with protanopia and deuteranopia.

Mapping quantitative data to color



(a) Plotting data with only positive or negative values, an intuitive encoding is a sequential color map that varies only the lightness from 10% to 90% black.

(b) Rescaling the color map to the observed minimum and maximum. This is applicable if the theoretical range (e.g. 0,10) is not of interest.

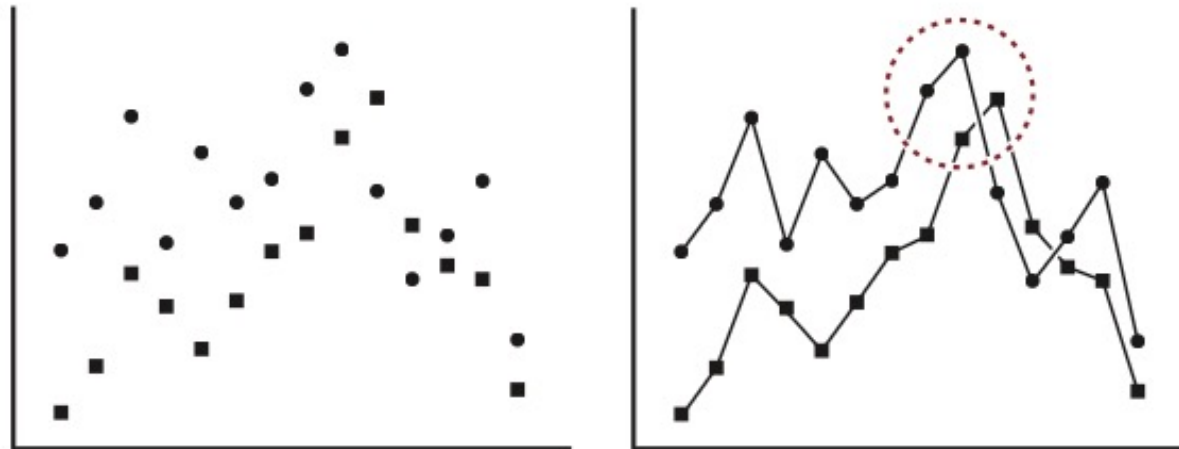
(c) Scenario where data containing both positive and negative values and in which the lower and upper ends of the distribution as well as zero need to be distinguished. A diverging (or bipolar) color schema that employs both color hue and color saturation is effective. Color hue makes a distinction between positive and negative values (for example, red and blue) and color saturation indicates the relative scale and no saturation represents zero. (d) The interpretation of zero or other key values can further influence the choice of color keys.

Reading Exercise

1. What is the connection between writing and figure generation?
2. List and explain the shown examples of effective written communication applied to the process of figure creation.

Programming Exercise

Creating Multiple Subplots



Wong "Gestalt principles (Part 1)" *Nature Methods* (2010)

Creating Multiple Subplots

rows

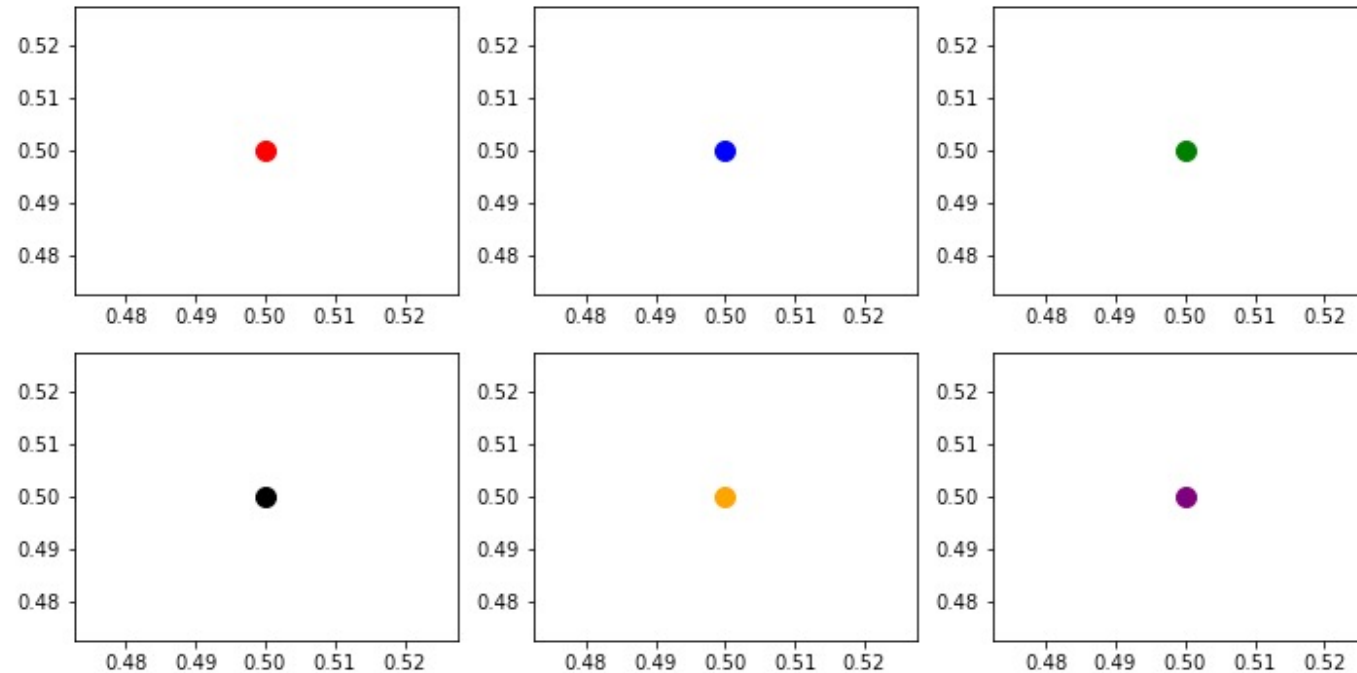
columns

```
fig, axs = plt.subplots(2,3)

fig.set_size_inches(12,6)
axs[0,0].scatter(0.5,0.5,s=100,c='red')
axs[0,1].scatter(0.5,0.5,s=100,c='blue')
axs[0,2].scatter(0.5,0.5,s=100,c='green')

axs[1,0].scatter(0.5,0.5,s=100,c='black')
axs[1,1].scatter(0.5,0.5,s=100,c='orange')
axs[1,2].scatter(0.5,0.5,s=100,c='purple')
```

<matplotlib.collections.PathCollection at 0x7fd0c329d8b0>



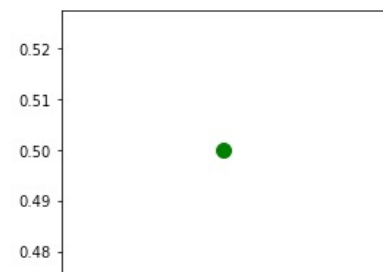
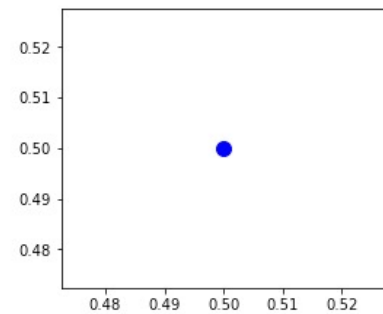
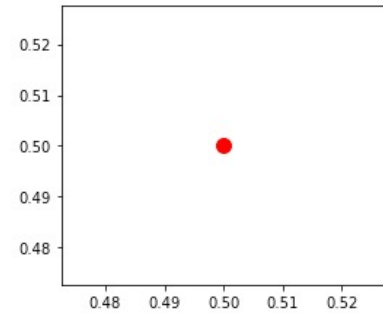
Creating Multiple Subplots

Vertical

```
fig, axs = plt.subplots(3,1)

fig.set_size_inches(4,12)
axs[0].scatter(0.5,0.5,s=100,c='red')
axs[1].scatter(0.5,0.5,s=100,c='blue')
axs[2].scatter(0.5,0.5,s=100,c='green')
```

<matplotlib.collections.PathCollection at 0x7fd0d7b0a5b0>

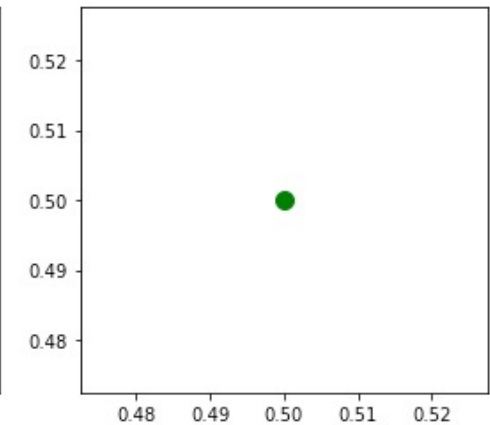
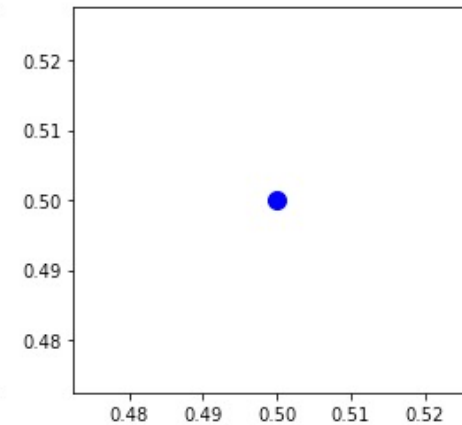
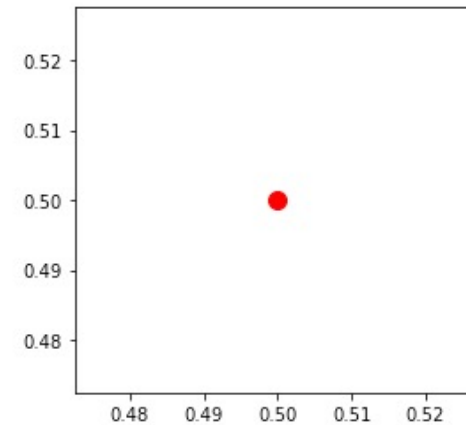


Horizontal

```
fig, axs = plt.subplots(1,3)

fig.set_size_inches(14,4)
axs[0].scatter(0.5,0.5,s=100,c='red')
axs[1].scatter(0.5,0.5,s=100,c='blue')
axs[2].scatter(0.5,0.5,s=100,c='green')
```

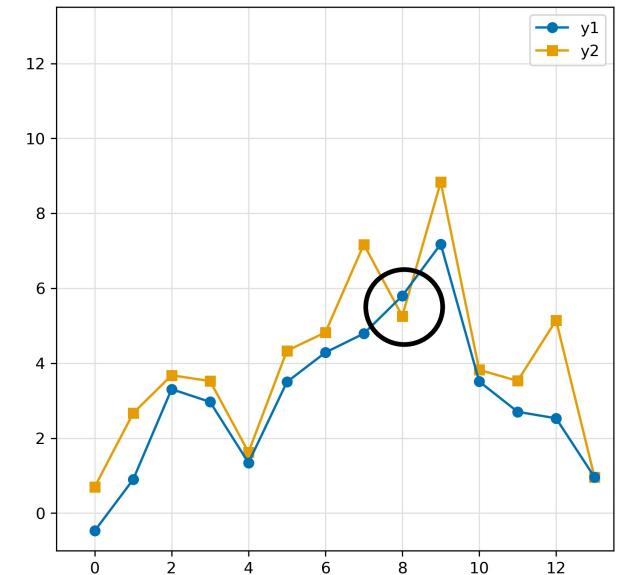
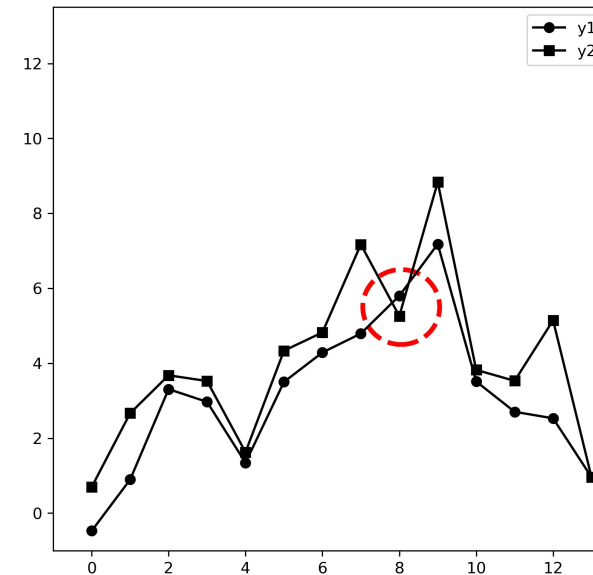
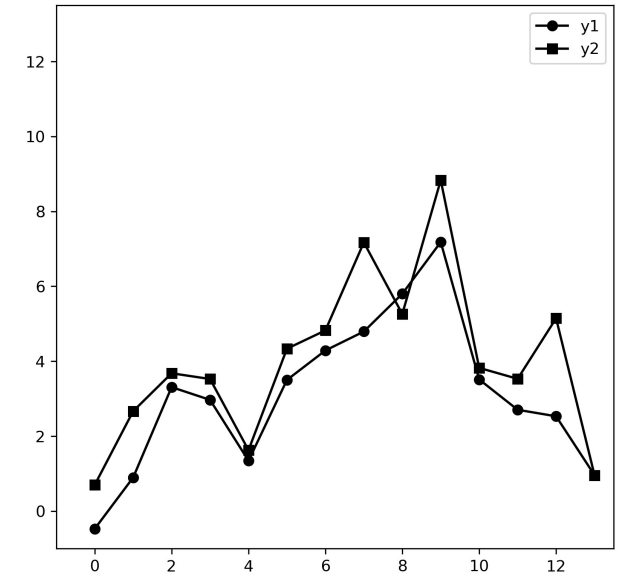
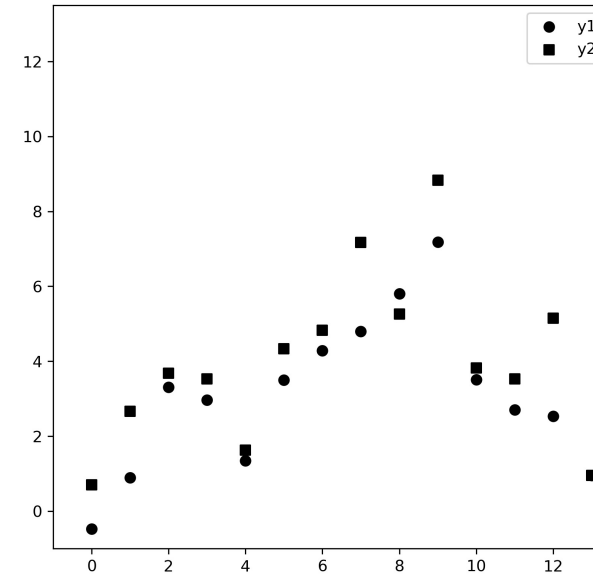
<matplotlib.collections.PathCollection at 0x7fd0b28fc160>



Creating Multiple Subplots

- go to iLearn and download '05_data.csv'
- create the shown figure with 4 subplots
- figure out how to draw circles with Matplotlib

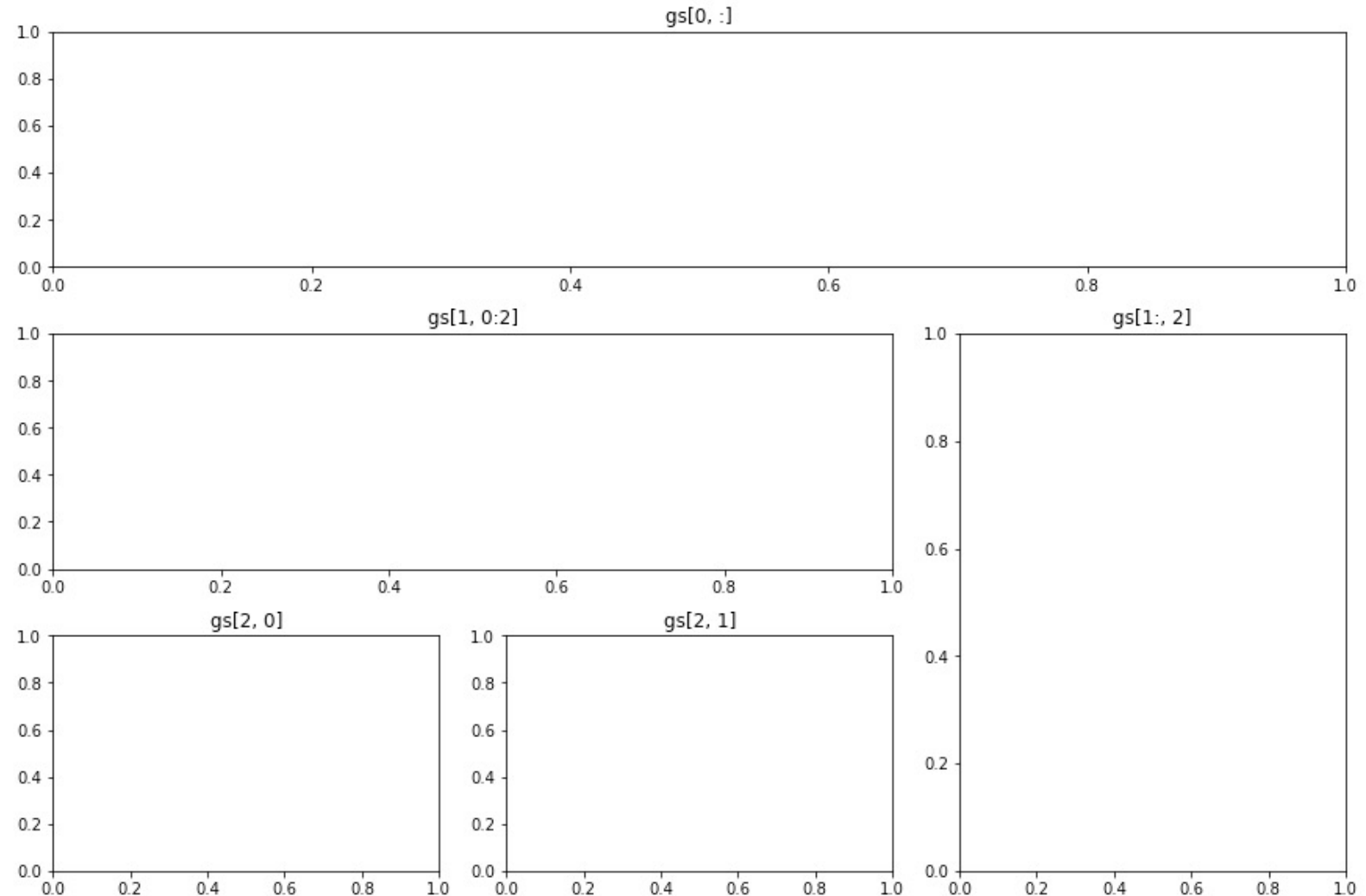
Matplotlib cheatsheets and hand-outs:
<https://github.com/matplotlib/cheatsheets#cheatsheets>



Use GridSpec for Custom Layouts

```
import matplotlib.gridspec as gridspec
fig = plt.figure(constrained_layout=True)
fig.set_size_inches(12,8)
gs = fig.add_gridspec(3, 3)

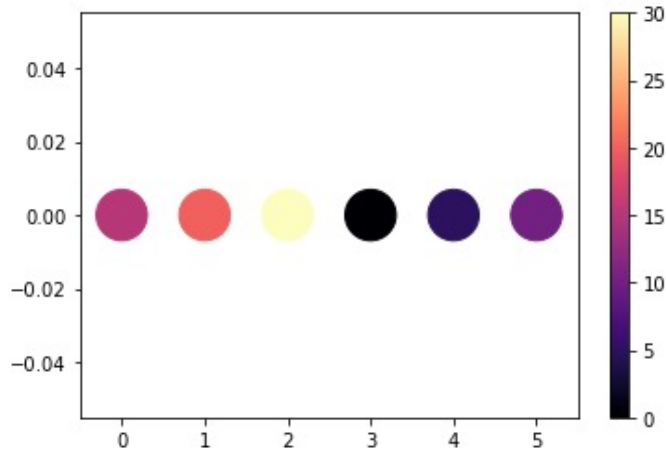
f_ax1 = fig.add_subplot(gs[0, :])
f_ax1.set_title('gs[0, :]')
f_ax2 = fig.add_subplot(gs[1, 0:2])
f_ax2.set_title('gs[1, 0:2]')
f_ax3 = fig.add_subplot(gs[1:, 2])
f_ax3.set_title('gs[1:, 2]')
f_ax4 = fig.add_subplot(gs[2, 0])
f_ax4.set_title('gs[2, 0]')
f_ax5 = fig.add_subplot(gs[2, 1])
f_ax5.set_title('gs[2, 1]')
```



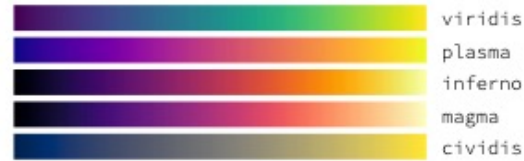
Colormaps

```
fig, ax = plt.subplots()
ax.set_xlim([-0.5, 5.5])
points = ax.scatter(np.arange(0, 6, 1), np.zeros(6), c=[15, 20, 30, 0, 5, 10],
                   s=750, cmap=plt.get_cmap('magma'), vmin=0, vmax=30)
fig.colorbar(points, ax=ax)
```

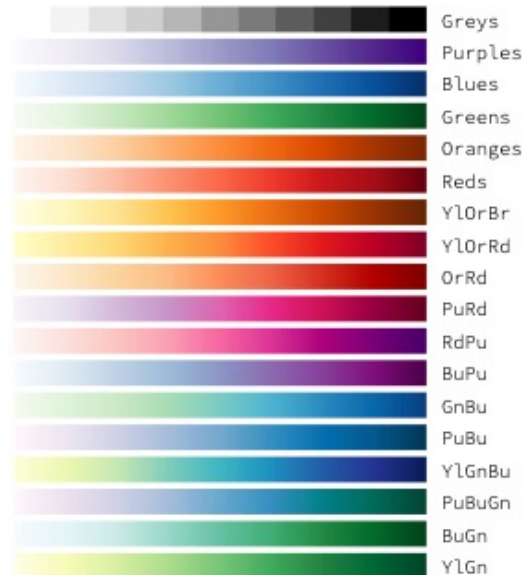
<matplotlib.colorbar.Colorbar at 0x7fece0ef2100>



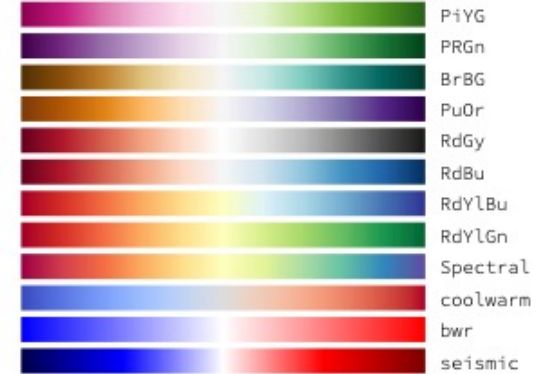
Uniform colormaps



Sequential colormaps



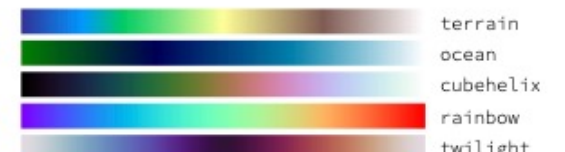
Diverging colormaps



Qualitative colormaps



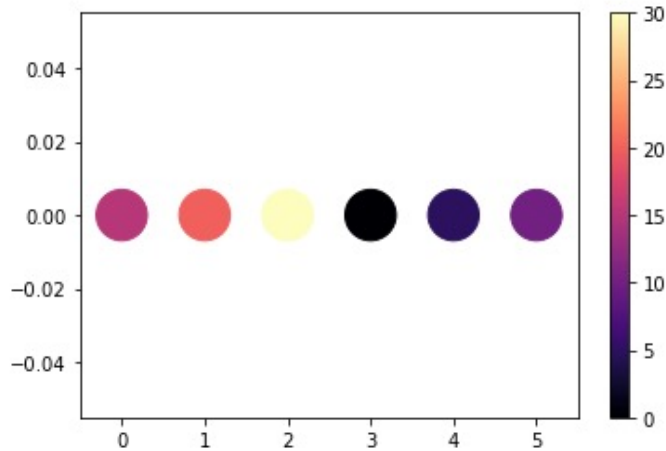
Miscellaneous colormaps



Colormaps

```
fig, ax = plt.subplots()
ax.set_xlim([-0.5, 5.5])
points = ax.scatter(np.arange(0, 6, 1), np.zeros(6), c=[15, 20, 30, 0, 5, 10],
                   s=750, cmap=plt.get_cmap('magma'), vmin=0, vmax=30)
fig.colorbar(points, ax=ax)
```

<matplotlib.colorbar.Colorbar at 0x7fece0ef2100>



- (go to iLearn and download '05_data.csv')
- create the shown figure with 2 subplots
- familiarize with the `pcolormesh()` function

Matplotlib cheatsheets and hand-outs:
<https://github.com/matplotlib/cheatsheets#cheatsheets>

