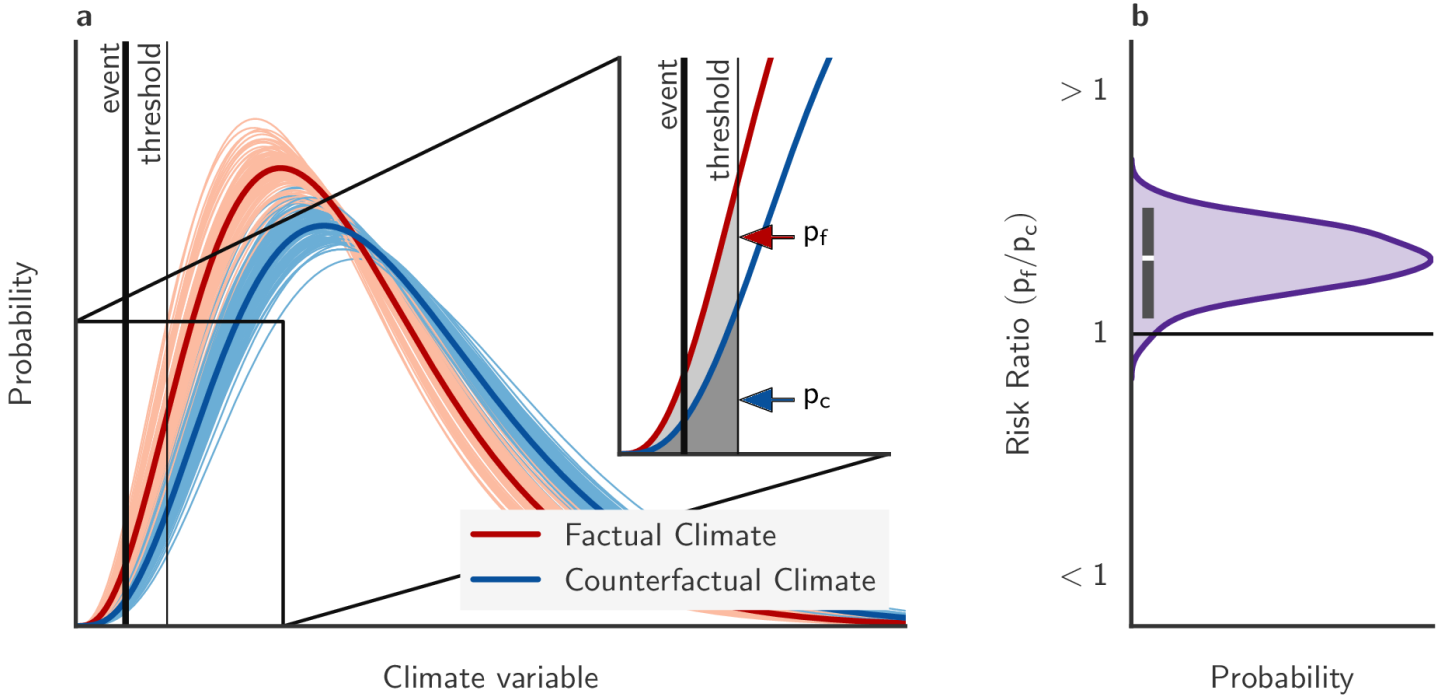
Python Visualization Workshop



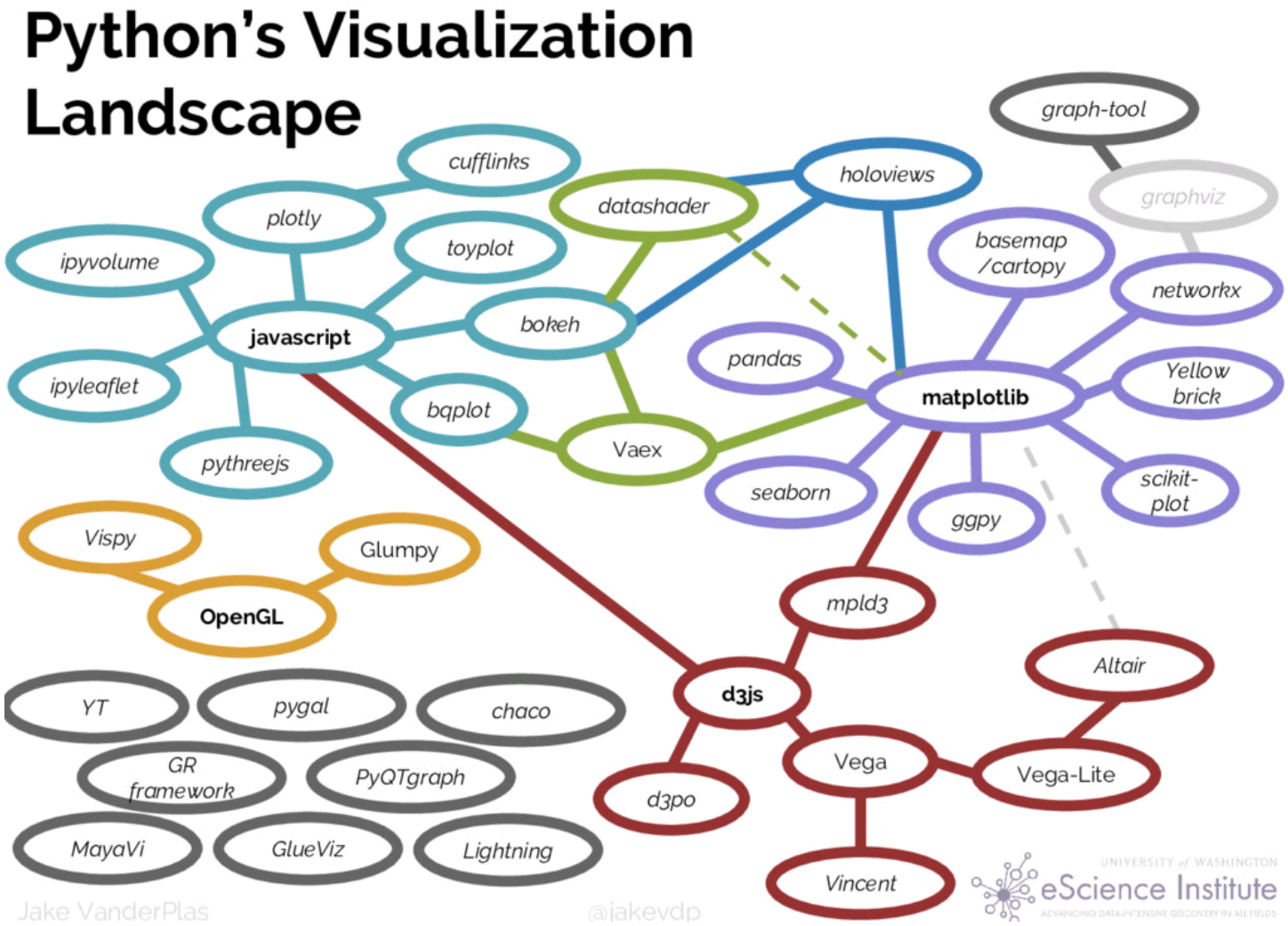
C2SM – Mathias Hauser, Tarun Chadha

ETH zürich

18 – 19 June 2019

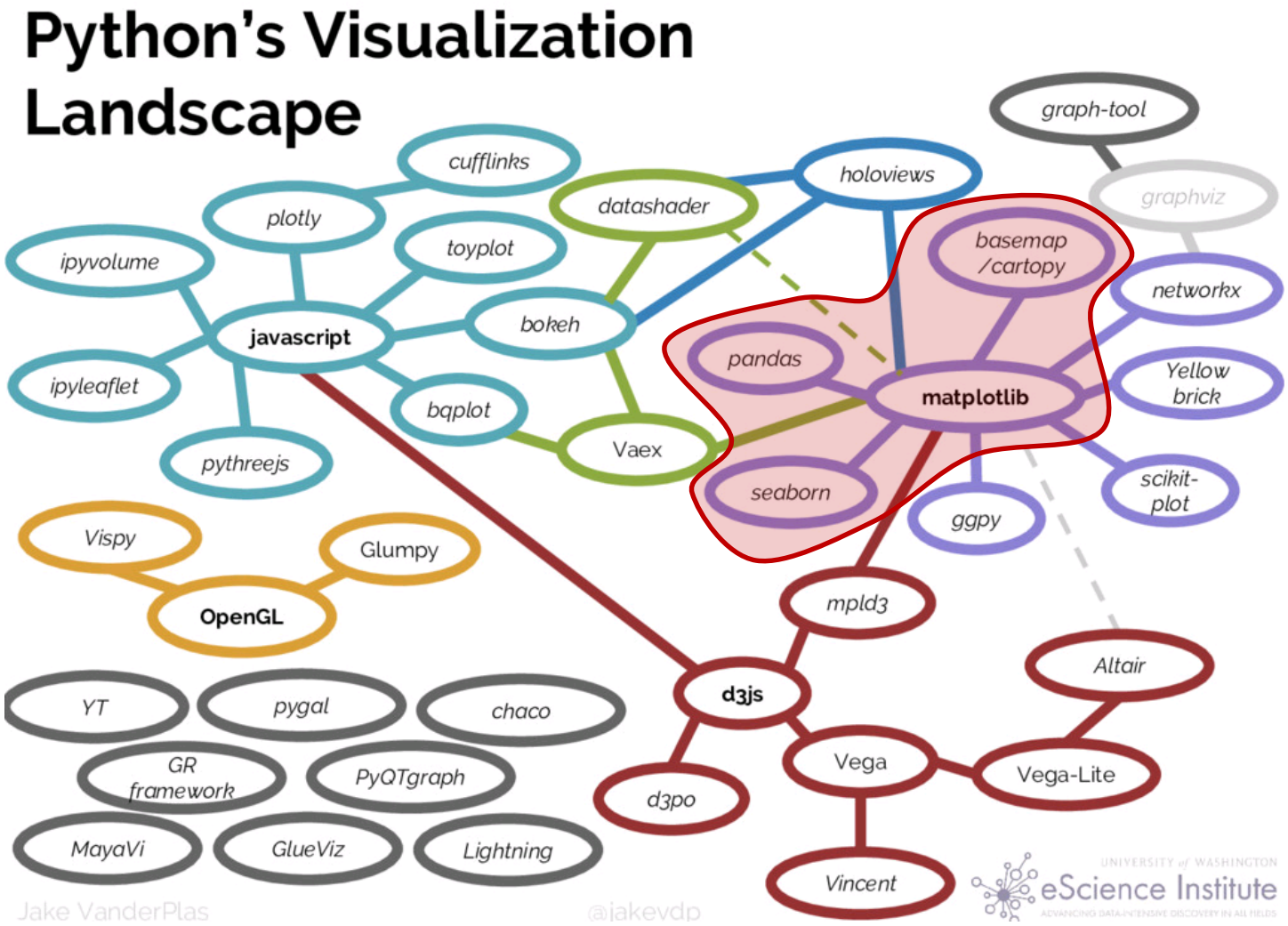
C2SM Center for Climate Systems Modeling

Python Plotting Libraries



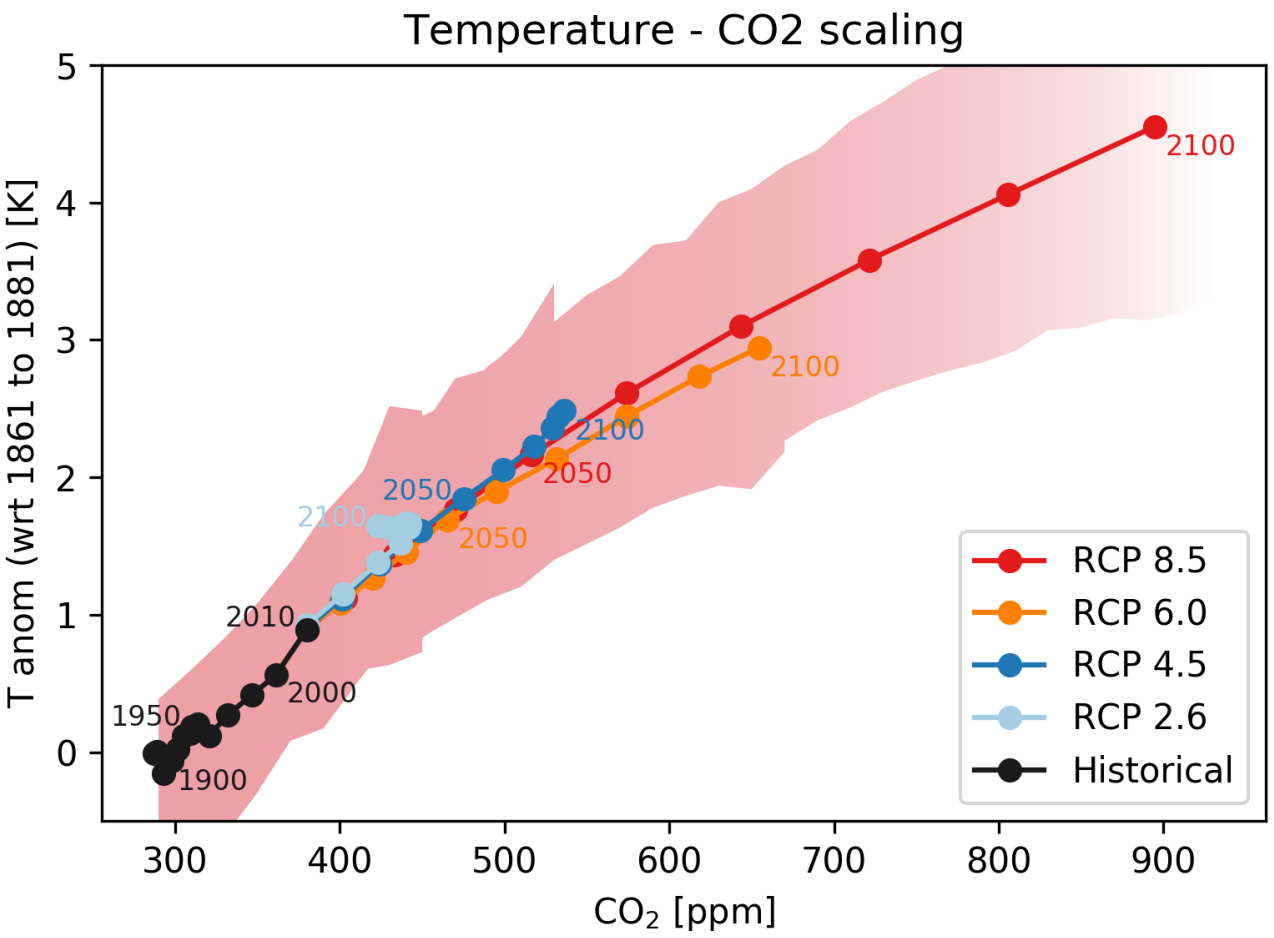
www.matplotlib.org

Python Plotting Libraries

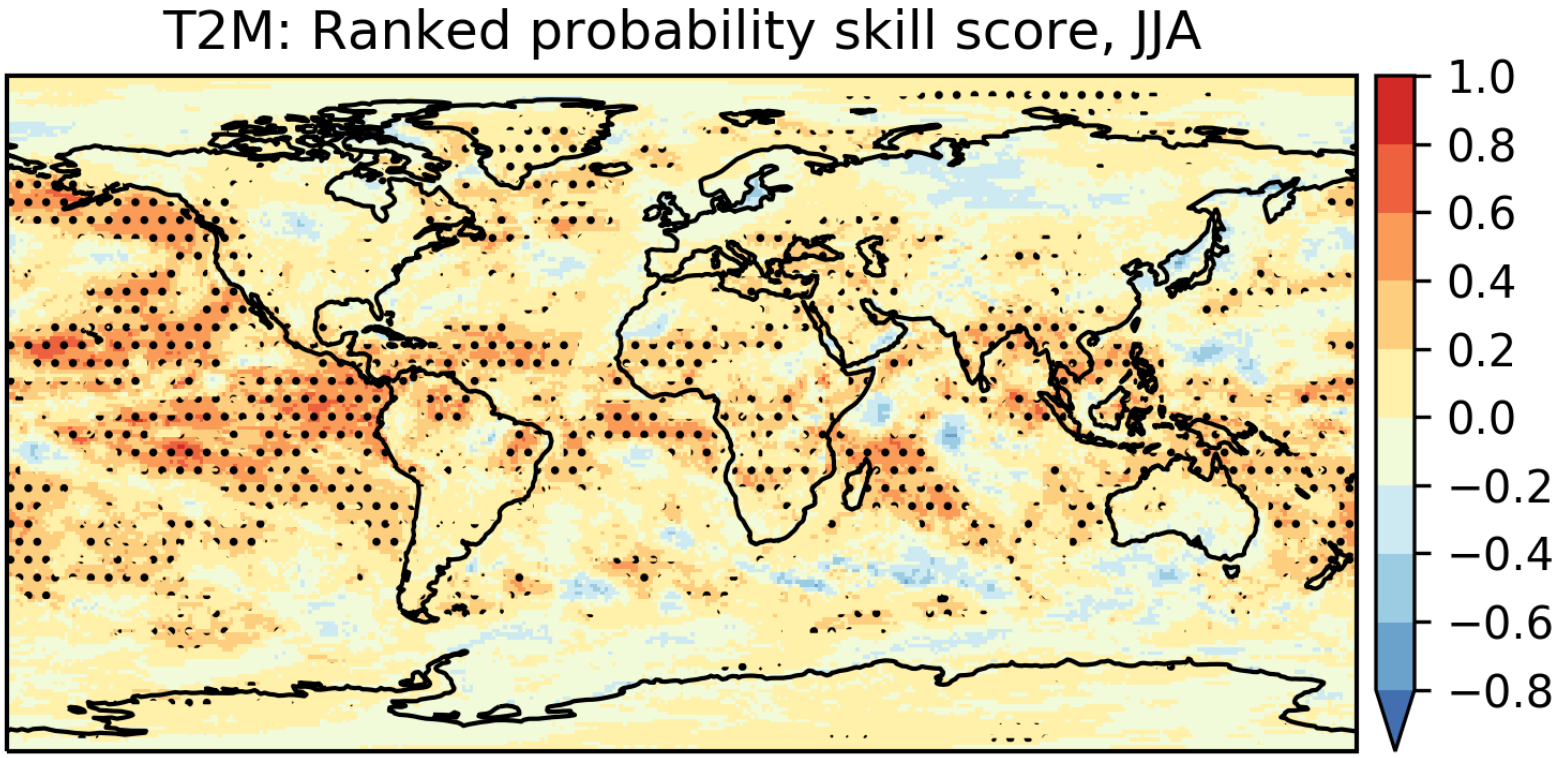


www.matplotlib.org

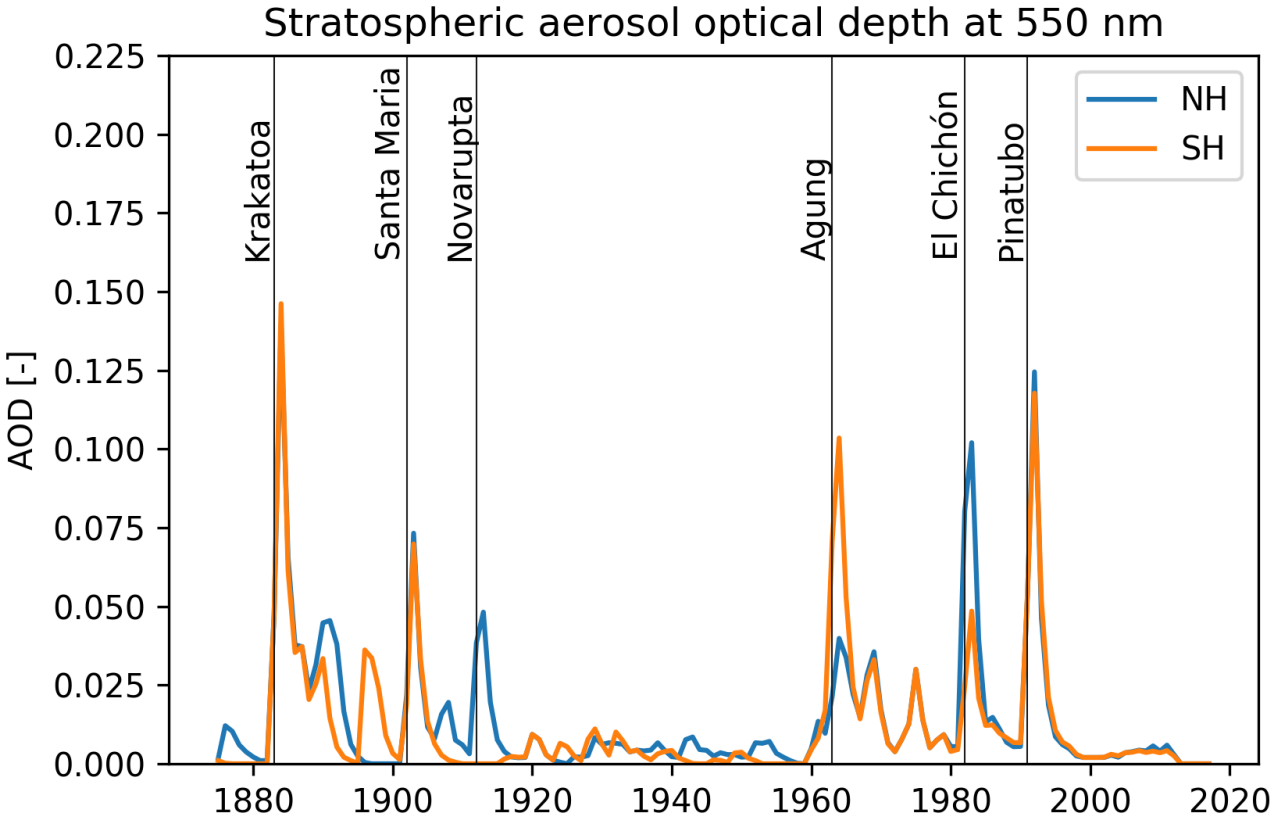
Teaser


Temperature - CO2 scaling
T anom (wrt 1861 to 1881) [K]
CO_2 [ppm]
RCP 8.5
RCP 6.0
RCP 4.5
RCP 2.6
Historical
1900
1950
2000
2010
2050
2100
2050
2100
2050
2100
2100

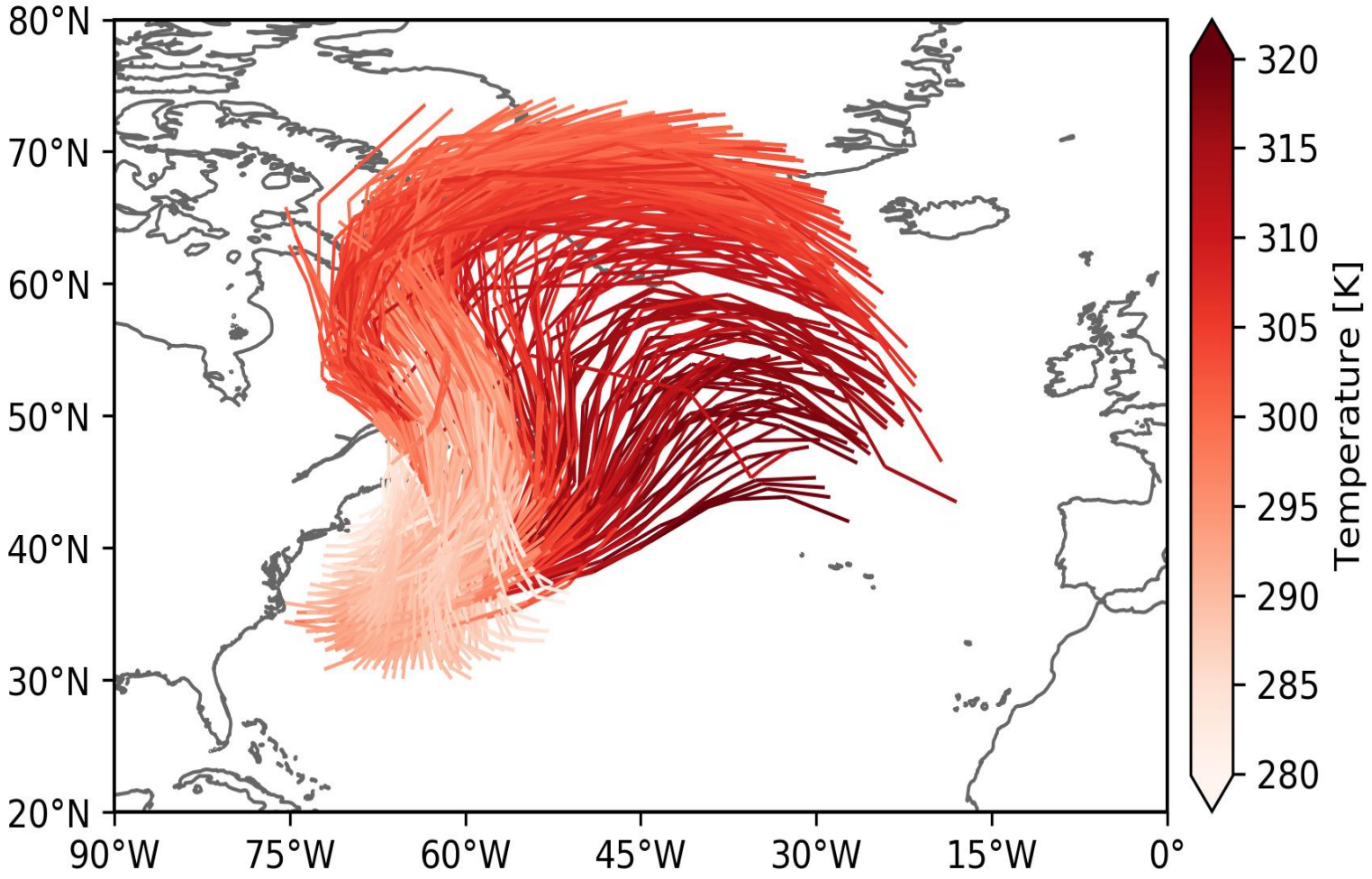
Teaser


T2M: Ranked probability skill score, JJA
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8

Teaser


Stratospheric aerosol optical depth at 550 nm
AOD [-]
0.225
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
1880
1900
1920
1940
1960
1980
2000
2020
Krakatoa
Santa Maria
Novarupta
Agung
El Chichón
Pinatubo
NH
SH

Teaser


80°N
70°N
60°N
50°N
40°N
30°N
20°N
90°W
75°W
60°W
45°W
30°W
15°W
0°
320
315
310
305
300
295
290
285
280
Temperature [K]

Schedule – Python Visualisation Course

Day 1	
09:00 – 10:30	**Part 0**: Introduction
10:30 – 11:00	Coffee Break
11:00 – 12:30	**Part 1**: Plotting in Python with matplotlib
12:30 – 13:30	Lunch Break
13:30 – 15:00	**Part 1**: Plotting in Python with matplotlib
15:00 – 15:30	Coffee Break
15:30 – 17:00	**Part 2**: Advanced plotting

Day 2	
09:00 – 10:30	**Part 3**: Plotting georeferenced data
10:30 – 11:00	Coffee Break
11:00 – 12:30	**Part 3**: Plotting georeferenced data
12:30 – 13:30	Lunch Break
13:30 – 15:00	**Part 3**: Plotting georeferenced data
15:00 – 15:30	Coffee Break
15:30 – 17:00	**Part 4**: User contributions

Schedule – Python Visualisation Course

Day 1	
09:00 – 10:30	**Part 0**: Introduction
10:30 – 11:00	Coffee Break
11:00 – 12:30	**Part 1**: Plotting in Python with matplotlib
12:30 – 13:30	Lunch Break
13:30 – 15:00	**Part 1**: Plotting in Python with matplotlib
15:00 – 15:30	Coffee Break
15:30 – 17:00	**Part 2**: Advanced plotting

Day 2	
09:00 – 10:30	**Part 3**: Plotting georeferenced data
10:30 – 11:00	Coffee Break
11:00 – 12:30	**Part 3**: Plotting georeferenced data
12:30 – 13:30	Lunch Break
13:30 – 15:00	**Part 3**: Plotting georeferenced data
15:00 – 15:30	Coffee Break
15:30 – 17:00	**Part 4**: User contributions

Part 0: Introduction

- No plotting, yet (sorry)
- integrated development environment
  - ipython notebooks (Exercise 0.1)
- Data handling libraries
  - NumPy (Exercise 0.2)
  - netCDF4 (Exercise 0.3)
  - xarray (Exercise 0.4)
  - pandas (Exercise 0.5)

Part 0: NumPy

- “NumPy is the fundamental package for scientific computing with python” – www.numpy.org

- N-dimensional array

```
>>> import numpy as np
>>> a = np.arange(15).reshape(3, 5)
>>> a
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]])
>>> a.shape
(3, 5)
```

Part 0: netCDF4

- Python interface to the netCDF C library
- Module to read & write netCDF (both ver. 3 & 4) files

Part 0: netCDF4

In [14]:
```
fN = '../data/HadEX2_GSL.nc'

ncf = nc.Dataset(fN)

print(ncf)
```

```
<class 'netCDF4._netCDF4.Dataset'>
root group (NETCDF4_CLASSIC data model, file format HDF5):
    data: Growing season length
    source: HadEX2 (http://www.climdex.org/)
    reference: Donat et al., 2013
    dimensions(sizes): lon(96), lat(73), time(50)
    variables(dimensions): float64 lon(lon), float64 lat(lat), int32 time(time), float32 GSL(time,lat,lon), float64
 trend(lat,lon), float64 p_val(lat,lon)
    groups:
```

In [13]:
```
trend = ncf.variables['trend']

print(trend)

print(trend[30:40, 30:40])
```

```
<class 'netCDF4._netCDF4.Variable'>
float64 trend(lat, lon)
    _FillValue: nan
unlimited dimensions:
current shape = (73, 96)
filling on
[[-- -- -- -- 0.0020566136610928876 0.0020273329058180245
  0.0021492478827482 84 -- 0.01941575946785279 --]
 [-- -- -- -- -- -- 0.00016648749343484148 -- -- --]
 [-- -- -- -- -- -- -- -- -- --]
 [-- -- -- -- -- -- -- -- -- --]
 [-- -- -- -- -- -- -- -- -- --]
 [-- -- -- -- -- -- -- -- -- --]
 [-- -- -- -- -- -- -- -- -- --]
 [-- -- -- -- -- -- -- -- -- --]
 [-- -- -- -- -- -- -- -- -- --]]
```

Part 0: xarray

- Multidimensional labeled arrays
- combines a netCDF-like data model with capabilities of pandas

Part 0: xarray

In [15]:
```
fN = '../data/HadEX2_GSL.nc'

ds = xr.open_dataset(fN)

print(ds)
```

```
<xarray.Dataset>
Dimensions:  (lat: 73, lon: 96, time: 50)
Coordinates:
  * lon      (lon) float64 0.0 3.75 7.5 11.25 15.0 18.75 22.5 26.25 30.0 ...
  * lat      (lat) float64 -90.0 -87.5 -85.0 -82.5 -80.0 -77.5 -75.0 -72.5 ...
  * time     (time) datetime64[ns] 1956-01-01 1957-01-01 1958-01-01 ...
Data variables:
    GSL      (time, lat, lon) float32 ...
    trend    (lat, lon) float64 ...
    p_val    (lat, lon) float64 ...
Attributes:
    data:       Growing season length
    source:     HadEX2 (http://www.climdex.org/)
    reference:  Donat et al., 2013
```

In [16]:
```
ds.trend
```

Out[16]:
```
<xarray.DataArray 'trend' (lat: 73, lon: 96)>
array([[nan, nan, nan, ..., nan, nan, nan],
       [nan, nan, nan, ..., nan, nan, nan],
       [nan, nan, nan, ..., nan, nan, nan],
       ...,
       [nan, nan, nan, ..., nan, nan, nan],
       [nan, nan, nan, ..., nan, nan, nan],
       [nan, nan, nan, ..., nan, nan, nan]])
Coordinates:
  * lon      (lon) float64 0.0 3.75 7.5 11.25 15.0 18.75 22.5 26.25 30.0 ...
  * lat      (lat) float64 -90.0 -87.5 -85.0 -82.5 -80.0 -77.5 -75.0 -72.5 ...
```

Part 0: xarray

In [17]:
```
lat = slice(30, 50)
lon = slice(360 - 105, 360 - 85)

ds.trend.sel(lat=lat, lon=lon)
```

Out[17]:
```
<xarray.DataArray 'trend' (lat: 9, lon: 6)>
array([[-0.013822, -0.017993,  0.028642,  0.068076,  0.050725,  0.052645],
       [ 0.042555, -0.08705 ,  0.044582,  0.072828,  0.033329,  0.08891 ],
       [ 0.286903, -0.121509, -0.163817, -0.209337, -0.071513,  0.080229],
       [ 0.20771 , -0.051205, -0.129159, -0.217335, -0.049765,  0.085104],
       [ 0.126969,  0.000905,  0.014469,  0.052174,  0.166199,  0.24968 ],
       [ 0.080082,  0.036576,  0.072964,  0.082449,  0.183254,  0.271937],
       [ 0.209573,  0.201362,  0.147796,  0.125882,  0.153077,       nan],
       [ 0.191687,  0.171211,  0.084009,  0.078607, -0.008008,       nan],
       [ 0.13068 ,  0.106646,  0.065728,  0.041136, -0.049716, -0.080854]])
Coordinates:
  * lon      (lon) float64 255.0 258.8 262.5 266.2 270.0 273.8
  * lat      (lat) float64 30.0 32.5 35.0 37.5 40.0 42.5 45.0 47.5 50.0
```

Part 0: pandas

- Statistical package for “labeled” data

- Provides an R-like DataFrame

```
In [6]: dates = pd.date_range('20130101', periods=6)

In [7]: dates
Out[7]:
DatetimeIndex(['2013-01-01', '2013-01-02', '2013-01-03', '2013-01-04',
               '2013-01-05', '2013-01-06'],
              dtype='datetime64[ns]', freq='D')

In [8]: df = pd.DataFrame(np.random.randn(6,4), index=dates, columns=list('ABCD'))

In [9]: df
Out[9]:
                   A         B         C         D
2013-01-01  0.469112 -0.282863 -1.509059 -1.135632
2013-01-02  1.212112 -0.173215  0.119209 -1.044236
2013-01-03 -0.861849 -2.104569 -0.494929  1.071804
2013-01-04  0.721555 -0.706771 -1.039575  0.271860
2013-01-05 -0.424972  0.567020  0.276232 -1.087401
2013-01-06 -0.673690  0.113648 -1.478427  0.524988
```

Part 0: Setup

- www.github.com/c2sm/pyvis

- Setup in HG E26.1
  - git clone https://github.com/c2sm/pyvis.git
  - If the terminal prompt shows "bash-4.4" instead of your username then
    - Execute: **source /etc/bashrc**

      OR
    - Add following lines to .bashrc:

```
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi
```

  - conda activate /opt/kunden/chadha/conda/envs/pyvis
  - jupyter notebook

Schedule – Python Visualisation Course

Day 1	
09:00 – 10:30	**Part 0**: Introduction
10:30 – 11:00	Coffee Break
11:00 – 12:30	**Part 1**: Plotting in Python with matplotlib
12:30 – 13:30	Lunch Break
13:30 – 15:00	**Part 1**: Plotting in Python with matplotlib
15:00 – 15:30	Coffee Break
15:30 – 17:00	**Part 2**: Advanced plotting

Day 2	
09:00 – 10:30	**Part 3**: Plotting georeferenced data
10:30 – 11:00	Coffee Break
11:00 – 12:30	**Part 3**: Plotting georeferenced data
12:30 – 13:30	Lunch Break
13:30 – 15:00	**Part 3**: Plotting georeferenced data
15:00 – 15:30	Coffee Break
15:30 – 17:00	**Part 4**: User contributions

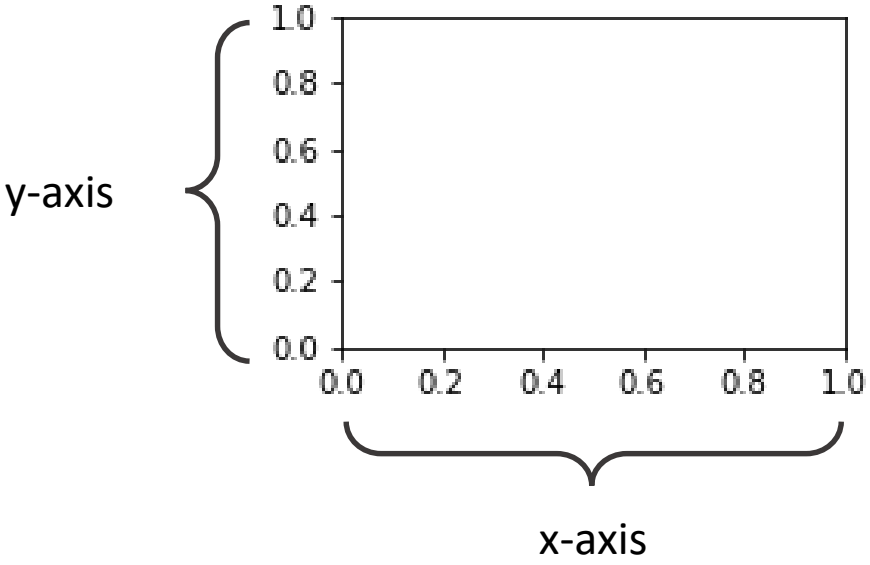
Part 1: Matplotlib

- Matplotlib is a Python 2D plotting library
- Similar syntax as MATLAB

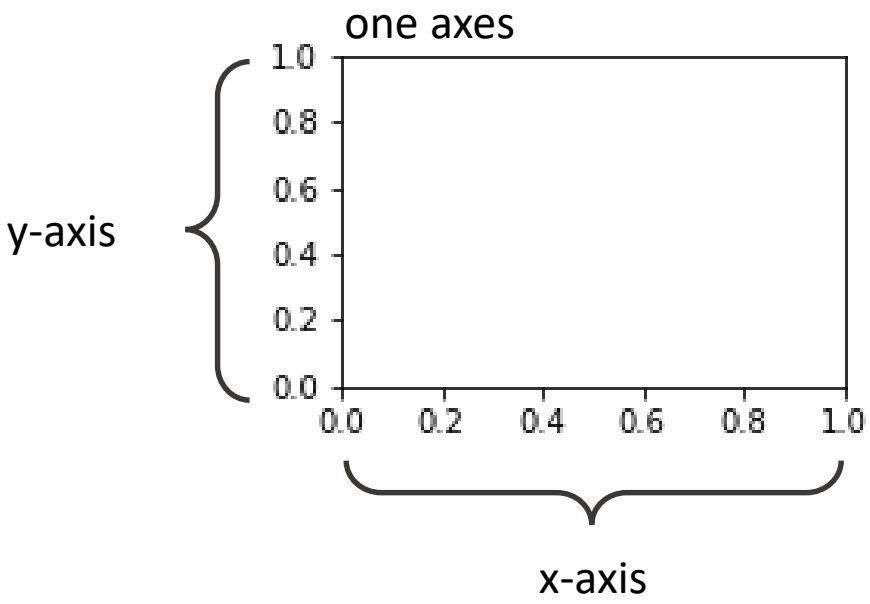
Part 1: Plotting in Python with matplotlib

- line plots
- uncertainty
- scatter plots
- subplots
- histograms
- annotations

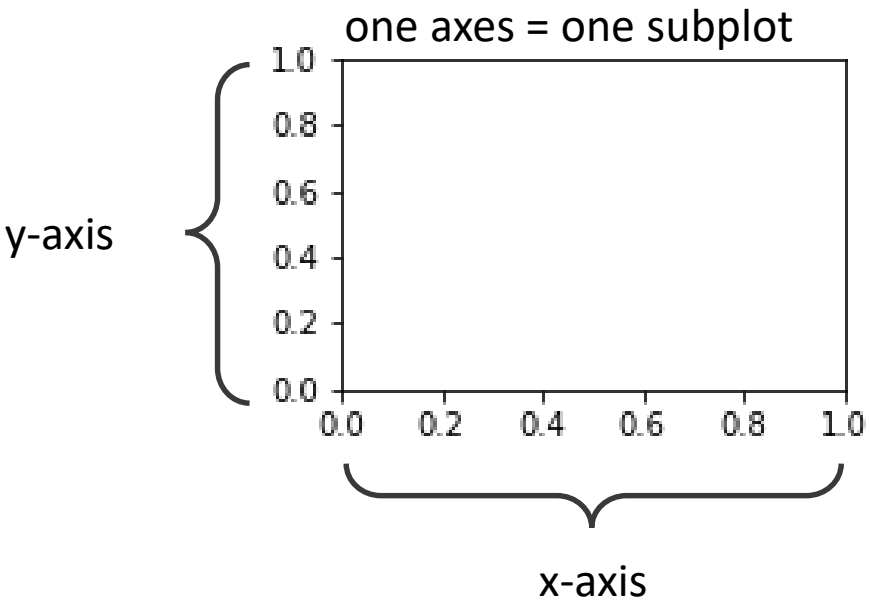
axis, axes, subplots, figure, spines, …



axis, axes, subplots, figure, spines, …

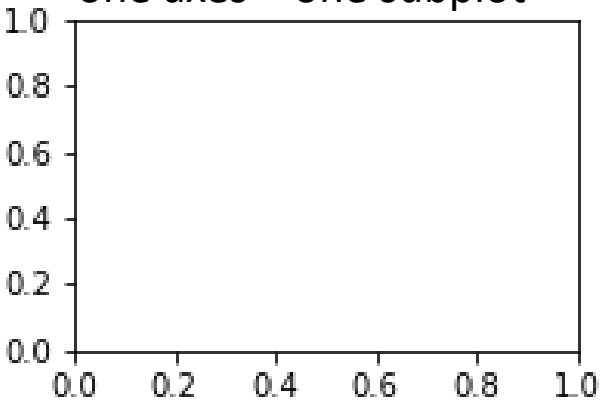


axis, axes, subplots, figure, spines, …

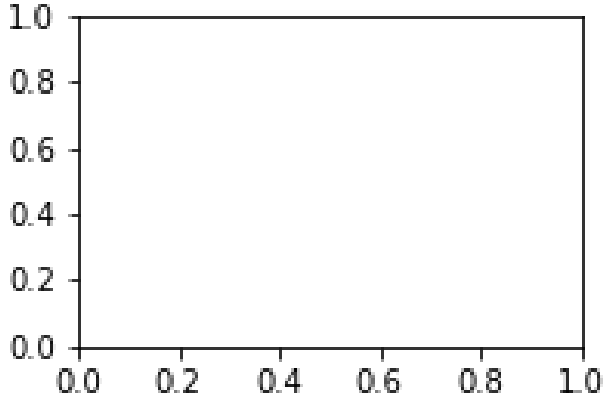


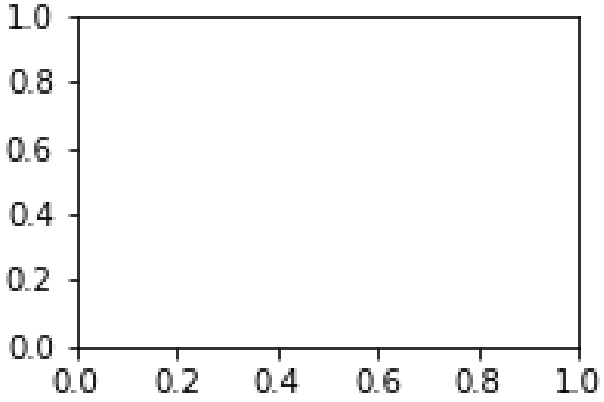
axis, axes, subplots, figure, spines, …

one axes = one subplot



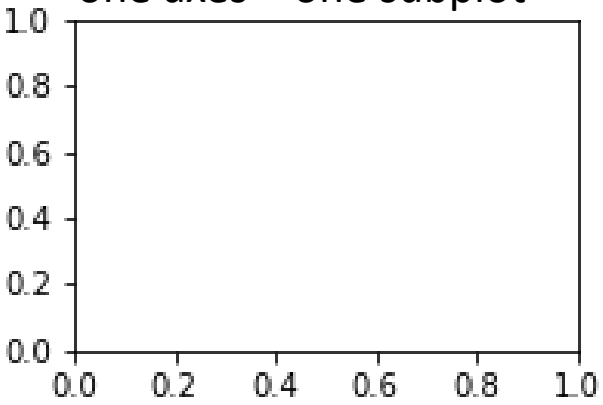
two axes = two subplots



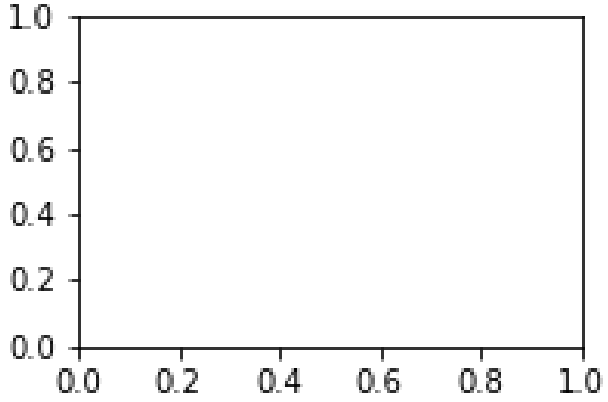


axis, axes, subplots, figure, spines, …

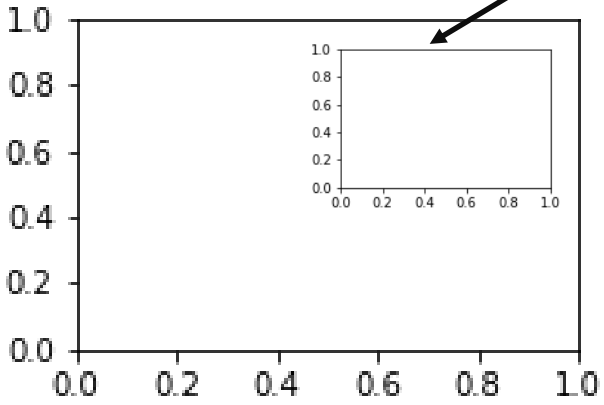
one axes = one subplot



two axes = two subplots

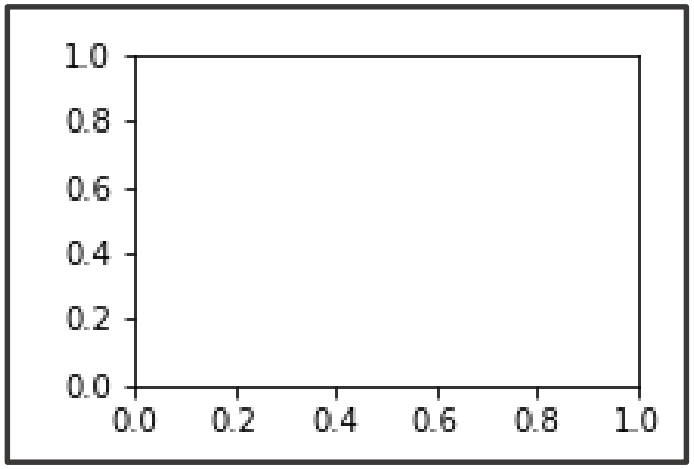


one axes; not a subplot

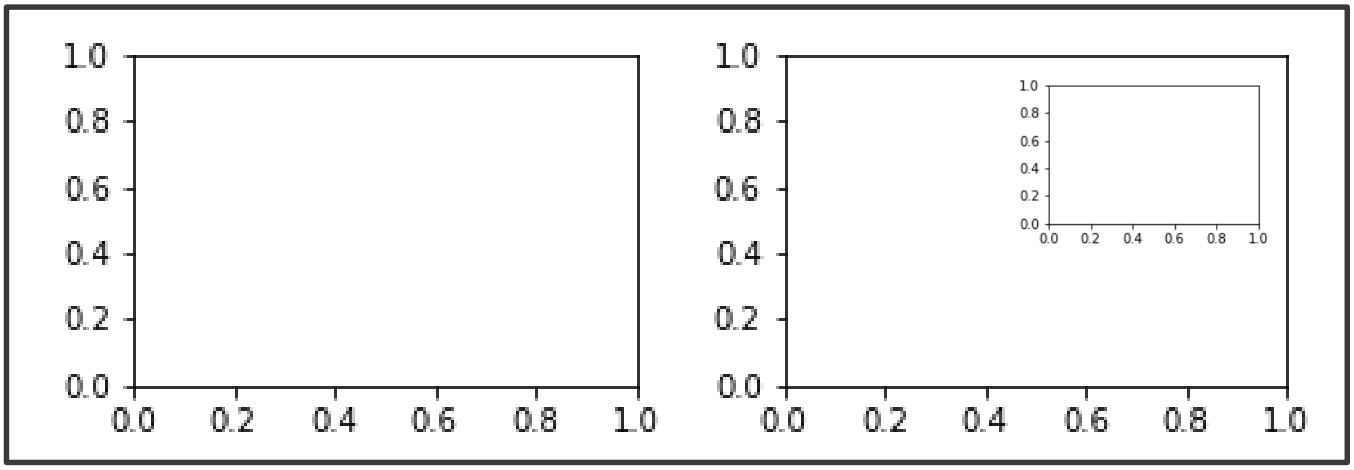


axis, axes, subplots, figure, spines, …

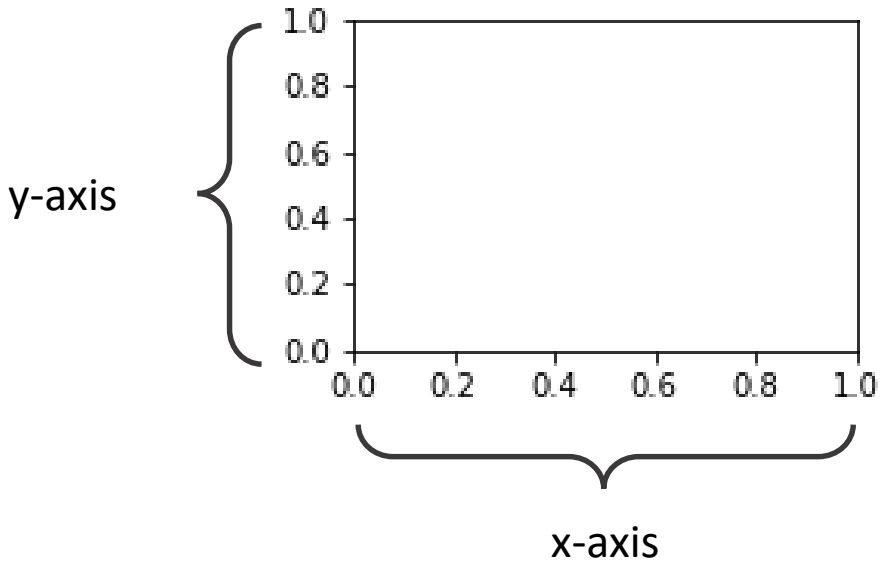
figure



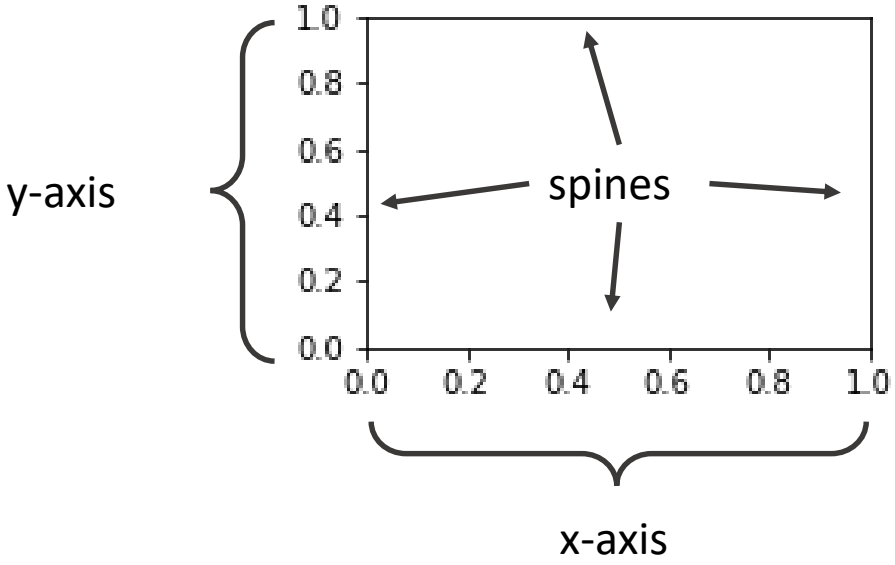
figure

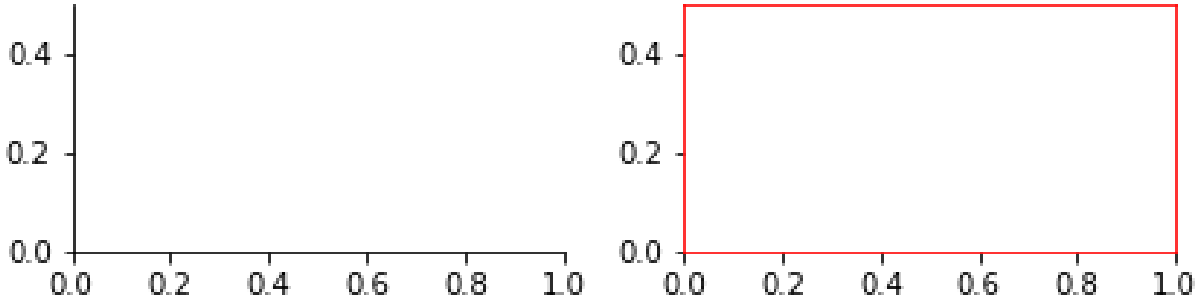


axis, axes, subplots, figure, spines, …

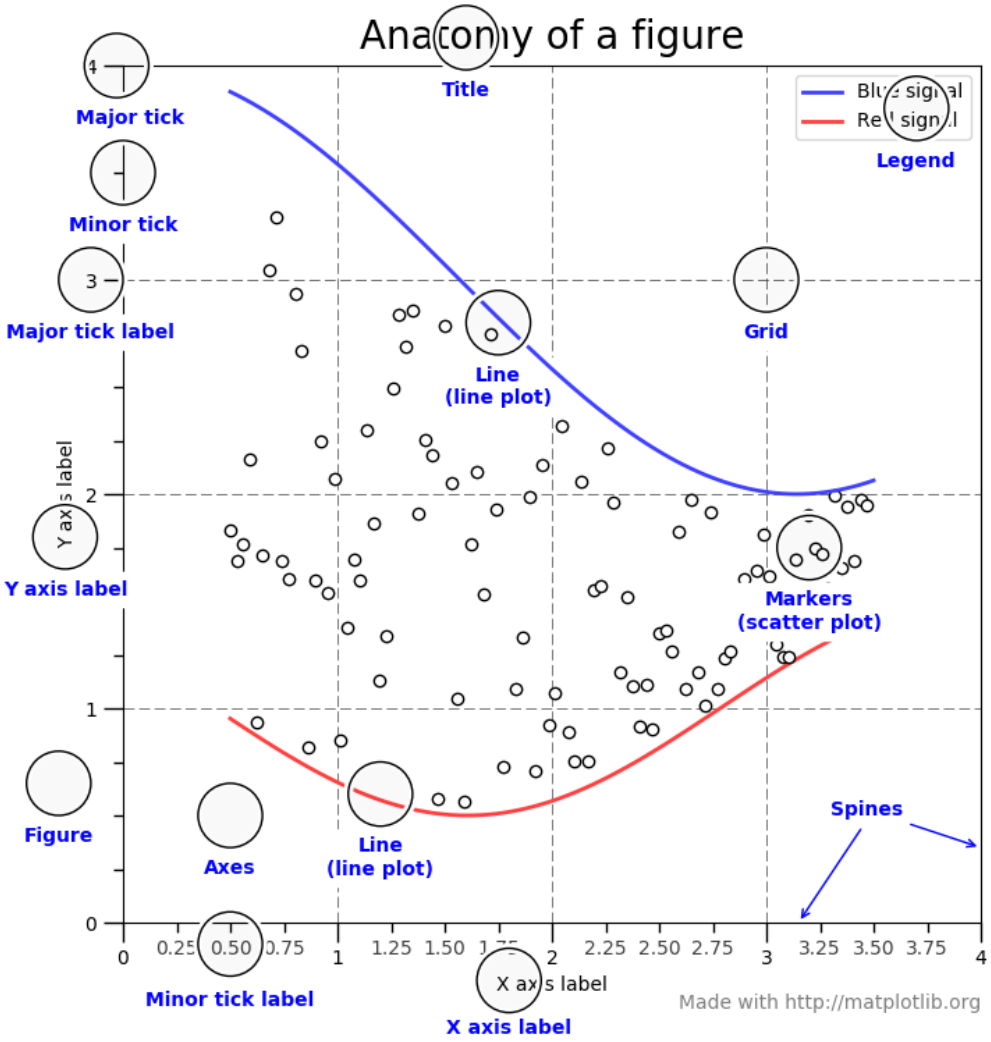


axis, axes, subplots, figure, spines, …





Part 1: Anatomy of a Figure



www.matplotlib.org

Schedule – Python Visualisation Course

Day 1	
09:00 – 10:30	**Part 0**: Introduction
10:30 – 11:00	Coffee Break
11:00 – 12:30	**Part 1**: Plotting in Python with matplotlib
12:30 – 13:30	Lunch Break
13:30 – 15:00	**Part 1**: Plotting in Python with matplotlib
15:00 – 15:30	Coffee Break
15:30 – 17:00	**Part 2**: Advanced plotting

Day 2	
09:00 – 10:30	**Part 3**: Plotting georeferenced data
10:30 – 11:00	Coffee Break
11:00 – 12:30	**Part 3**: Plotting georeferenced data
12:30 – 13:30	Lunch Break
13:30 – 15:00	**Part 3**: Plotting georeferenced data
15:00 – 15:30	Coffee Break
15:30 – 17:00	**Part 4**: User contributions

Schedule – Python Visualisation Course

Day 1	
09:00 – 10:30	**Part 0**: Introduction
10:30 – 11:00	Coffee Break
11:00 – 12:30	**Part 1**: Plotting in Python with matplotlib
12:30 – 13:30	Lunch Break
13:30 – 15:00	**Part 1**: Plotting in Python with matplotlib
15:00 – 15:30	Coffee Break
15:30 – 17:00	**Part 2**: Advanced plotting

Day 2	
09:00 – 10:30	**Part 3**: Plotting georeferenced data
10:30 – 11:00	Coffee Break
11:00 – 12:30	**Part 3**: Plotting georeferenced data
12:30 – 13:30	Lunch Break
13:30 – 15:00	**Part 3**: Plotting georeferenced data
15:00 – 15:30	Coffee Break
15:30 – 17:00	**Part 4**: User contributions

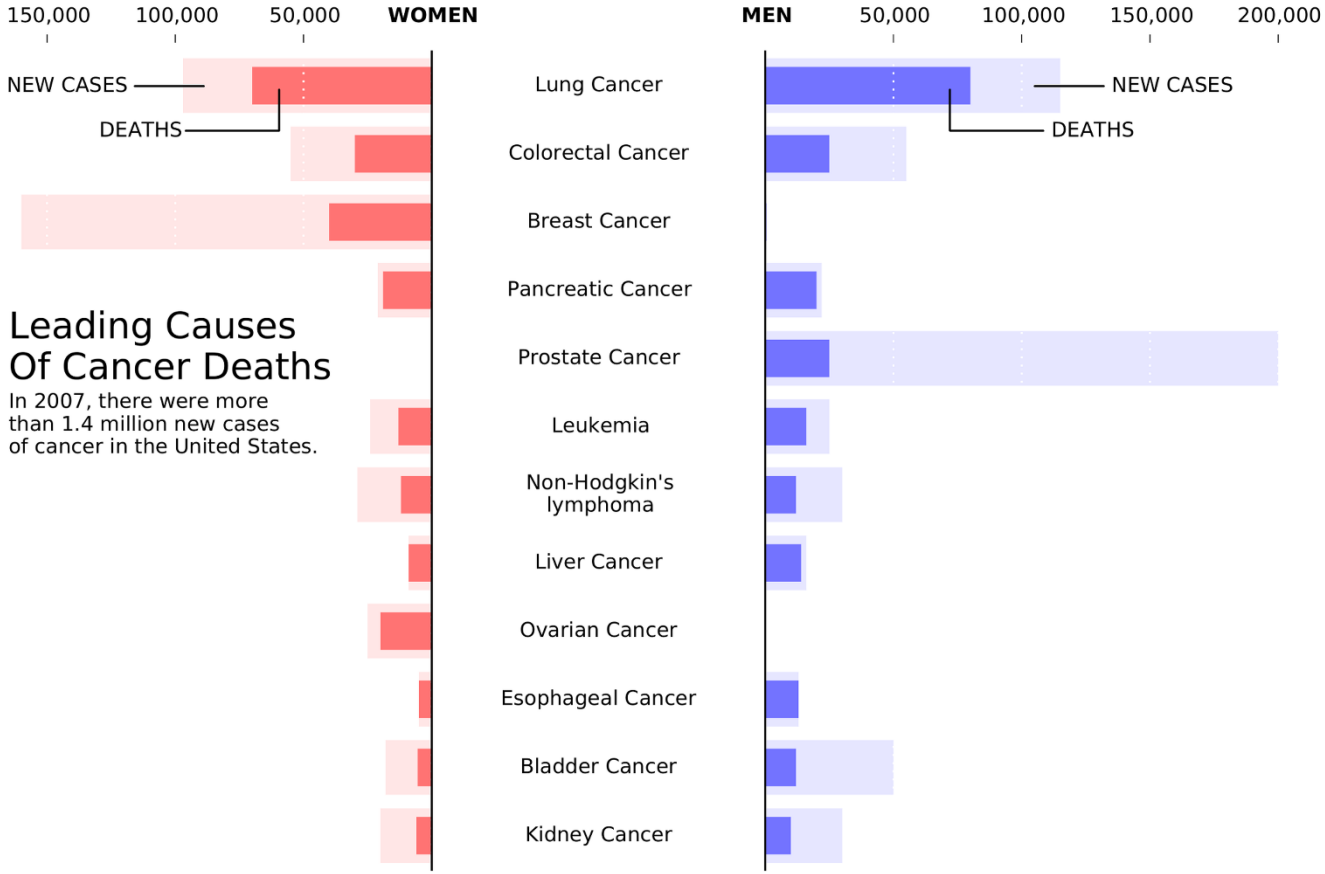
Advanced Plotting

- Styles
- Helper functions
- seaborn (statistical plots)

Ten Simple Rules for Better Figures

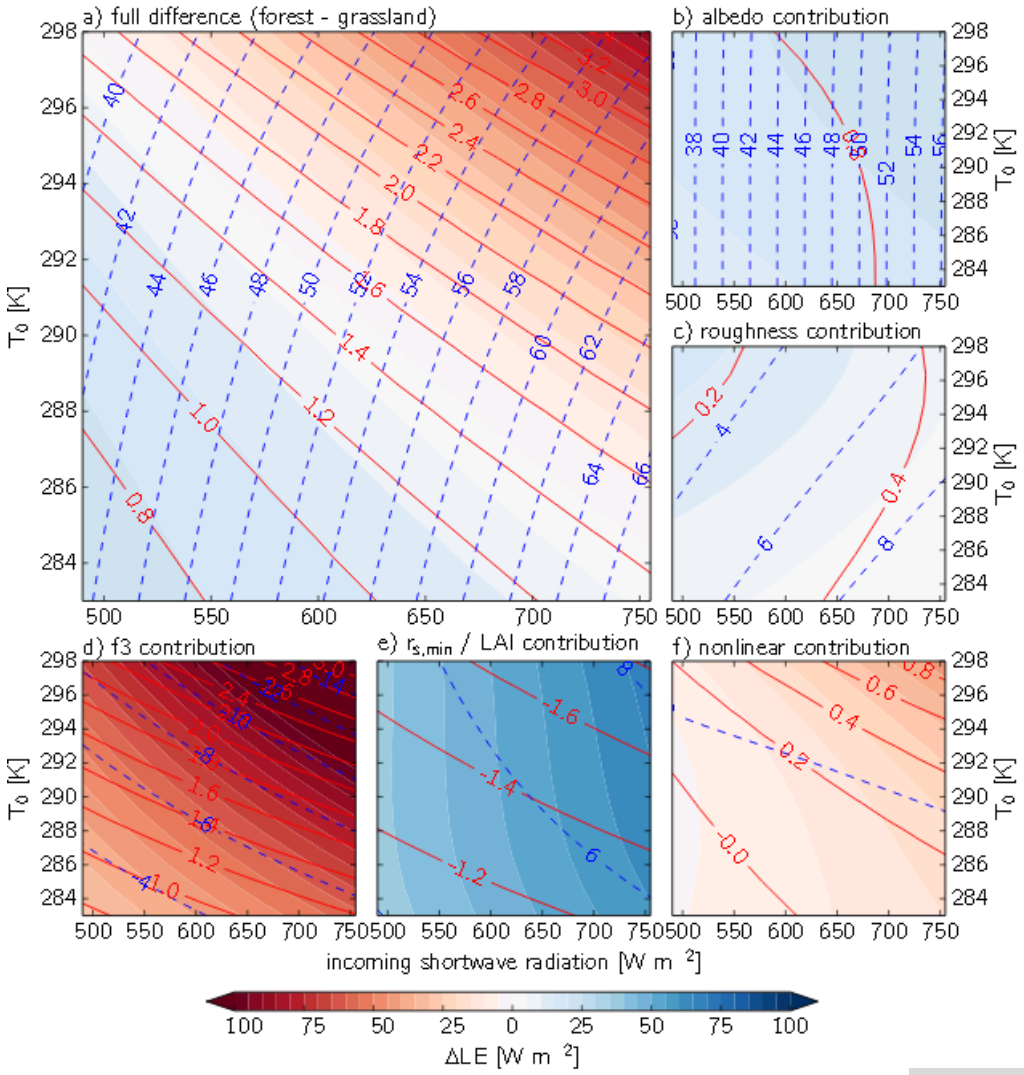
Rougier et al., 2014 PLOS

Rule 1: Know your audience



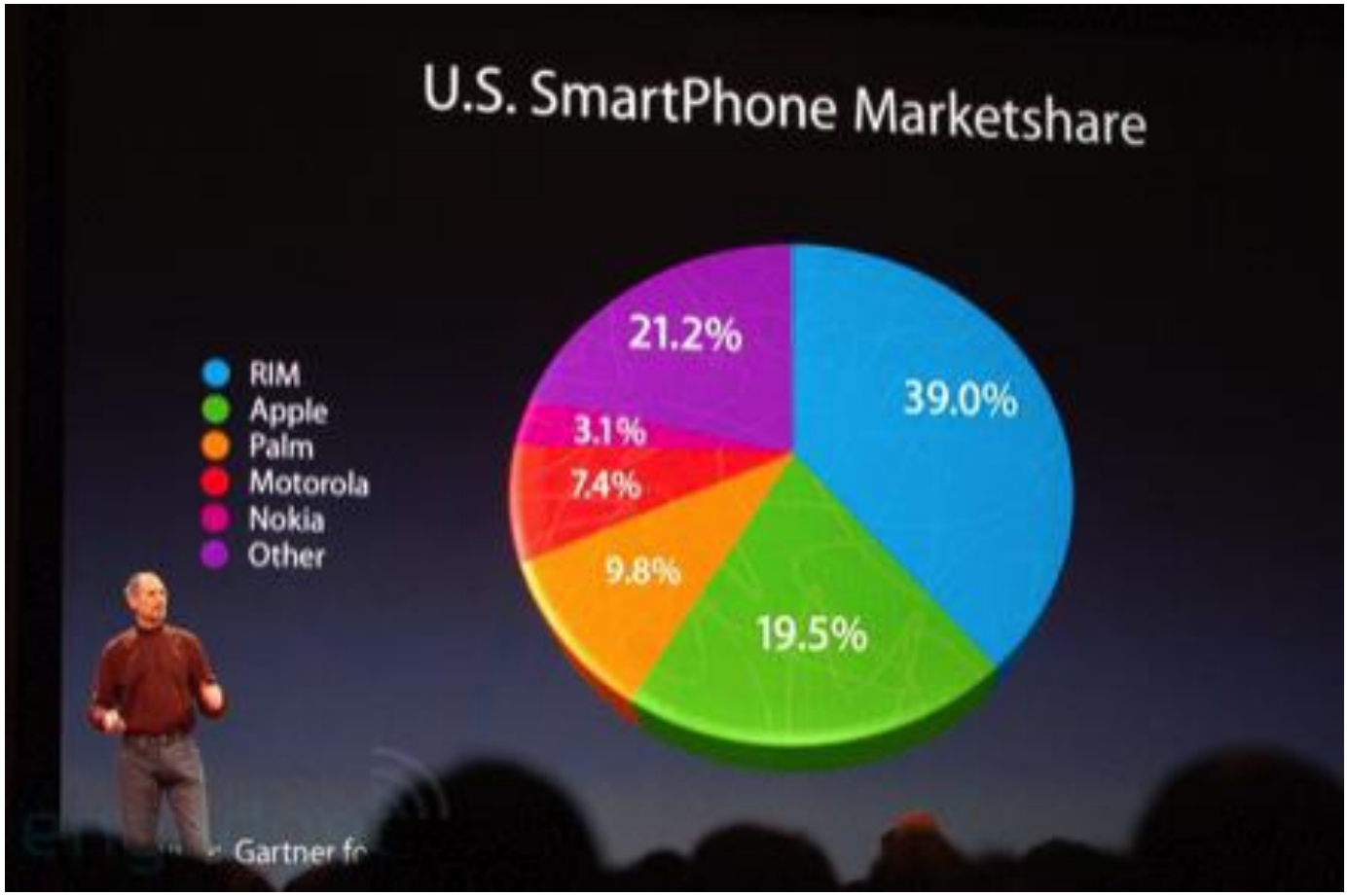
Rougier et al., 2014, PLOS

Rule 1: Know your audience

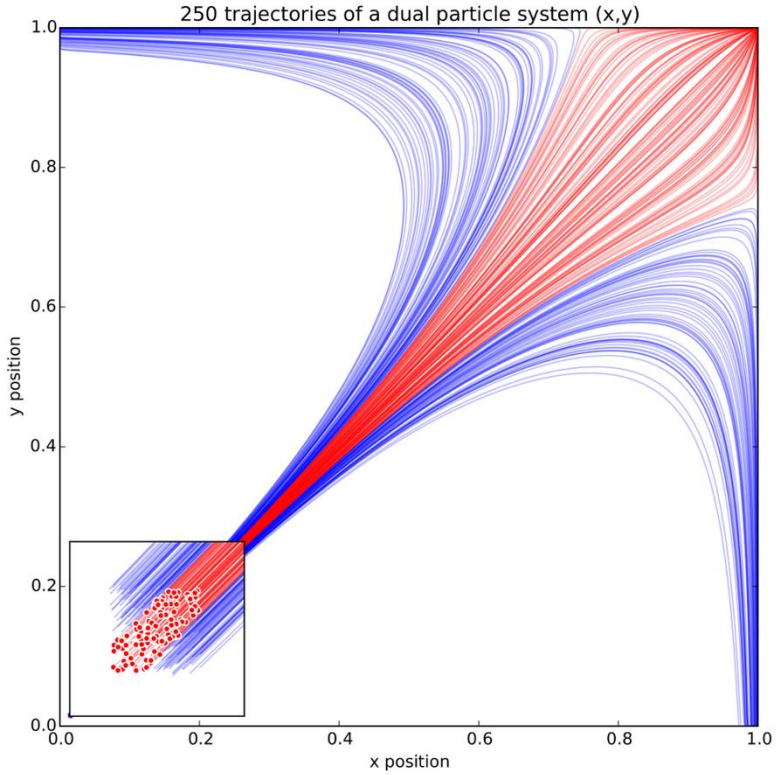


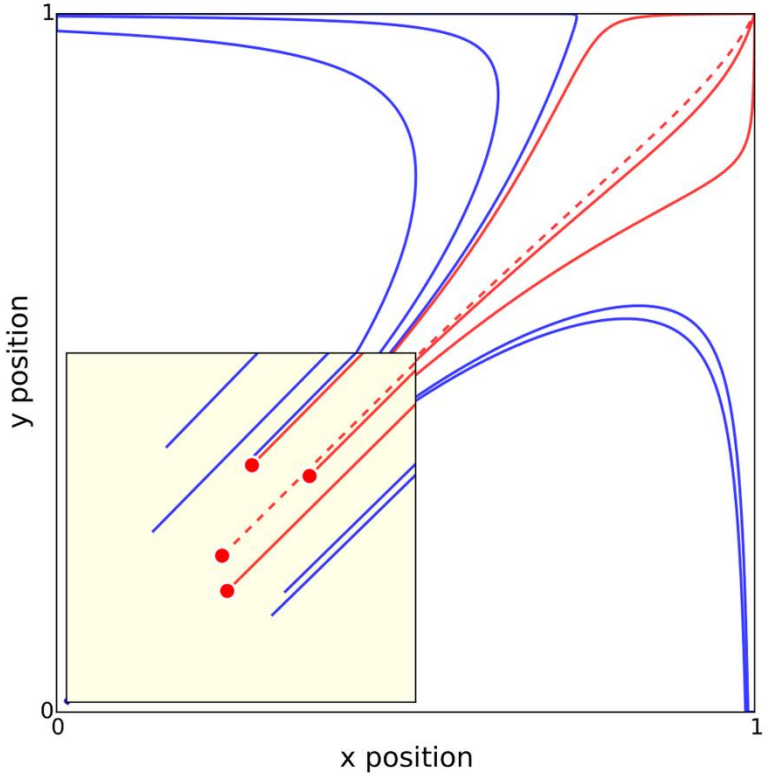
Van Heerwaarden et al., 2014

Rule 2: Identify Your Message



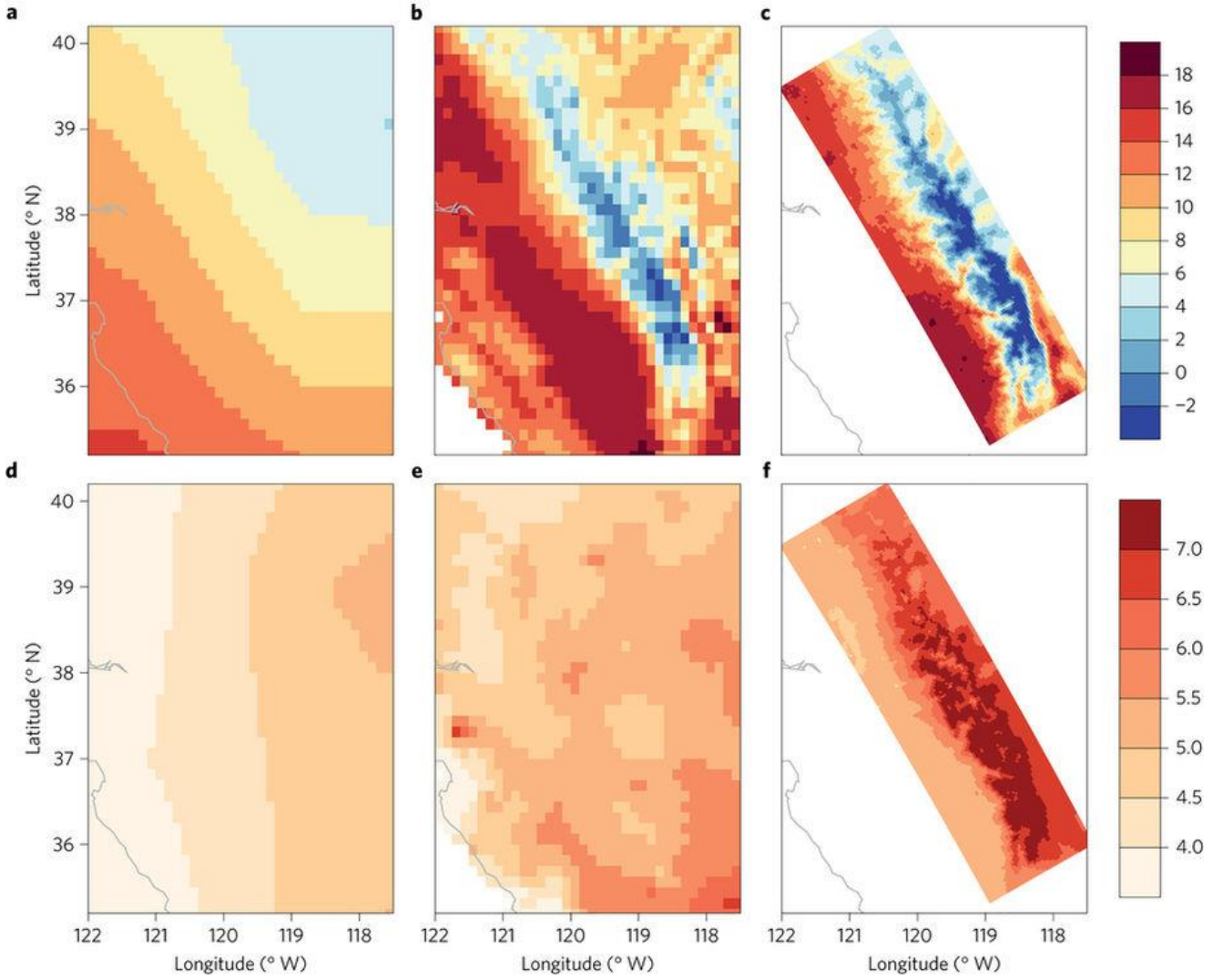
Rule 3: Adapt the Figure to the Support Medium





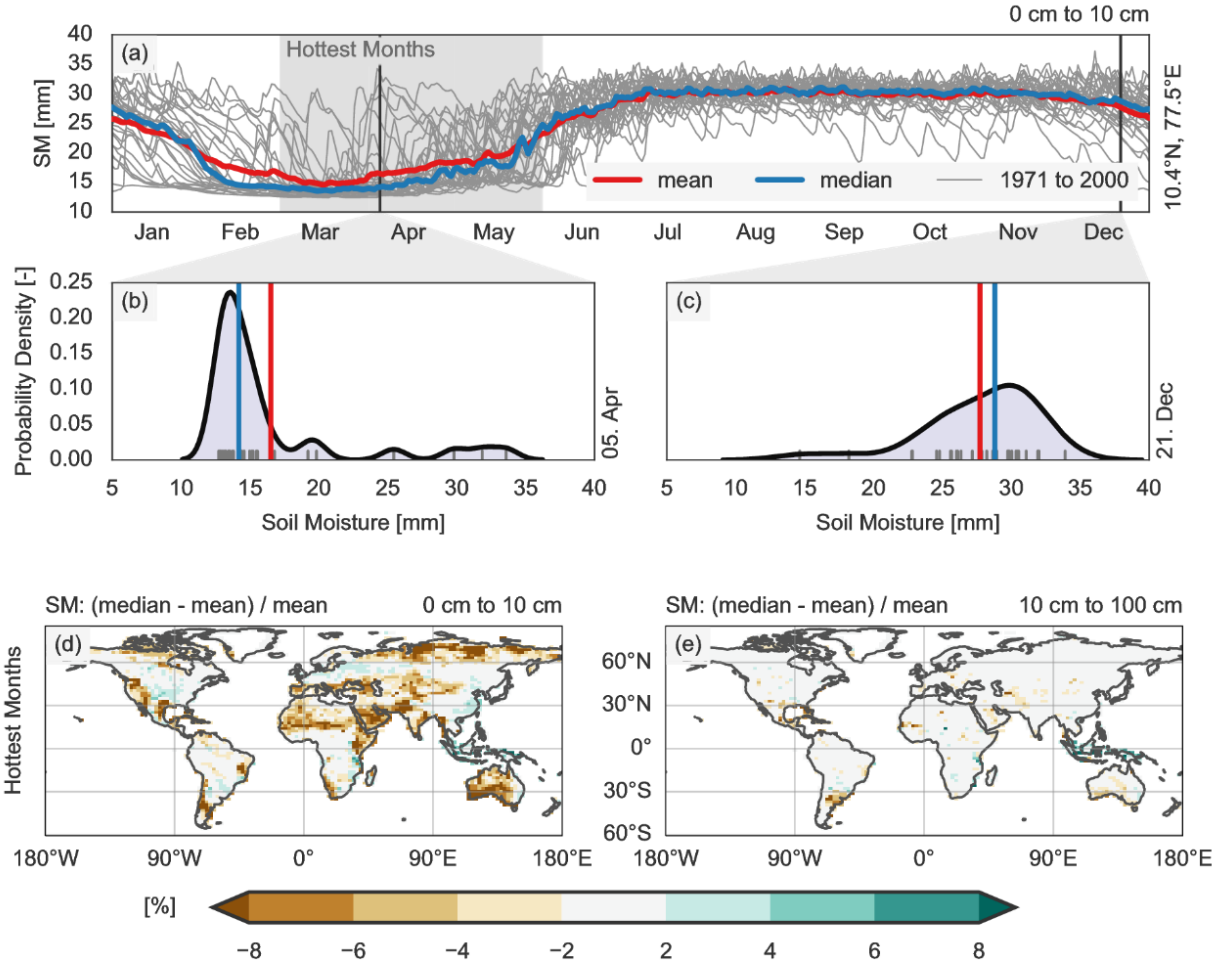
Rougier et al., 2014, PLOS

Rule 4: Labels, Titles, … Are Not Optional



Maraun et al., 2017 NCC

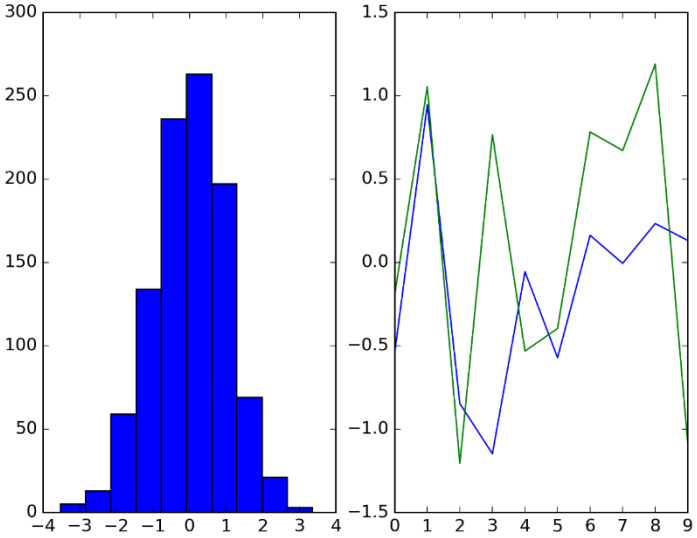
Rule 4: Labels, Titles, … Are Not Optional



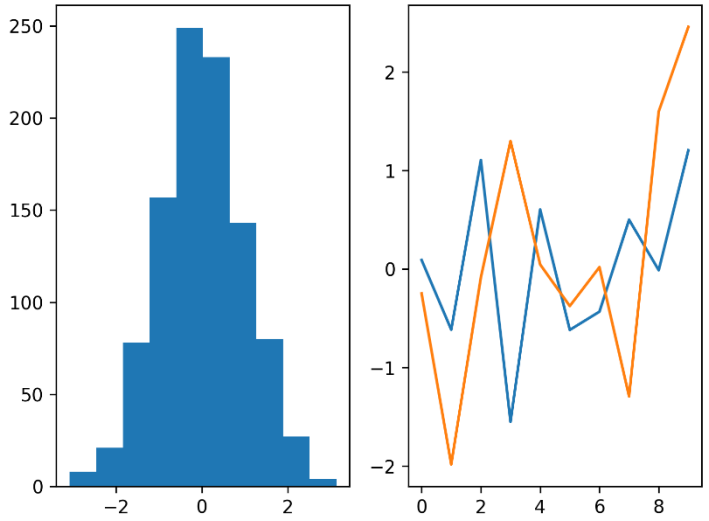
Hauser et al., 2017, GMD

Rule 5: Do Not Trust the Defaults

Version 1.5



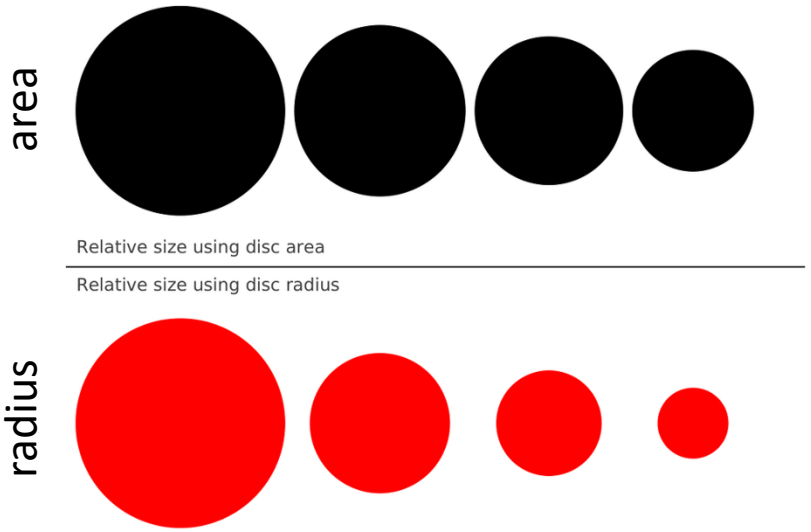
Version 2.1



Rule 6: Use Color Effectively

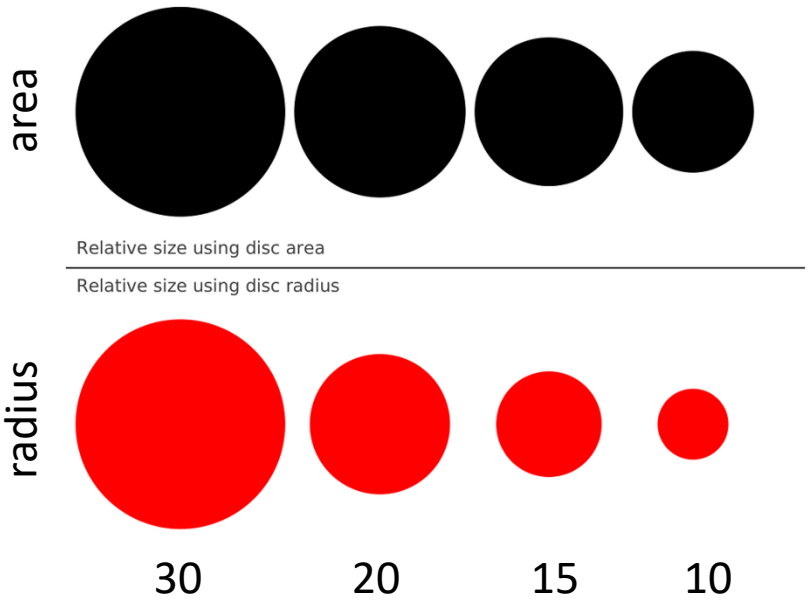
- See later

Rule 7: Do Not Mislead the Reader



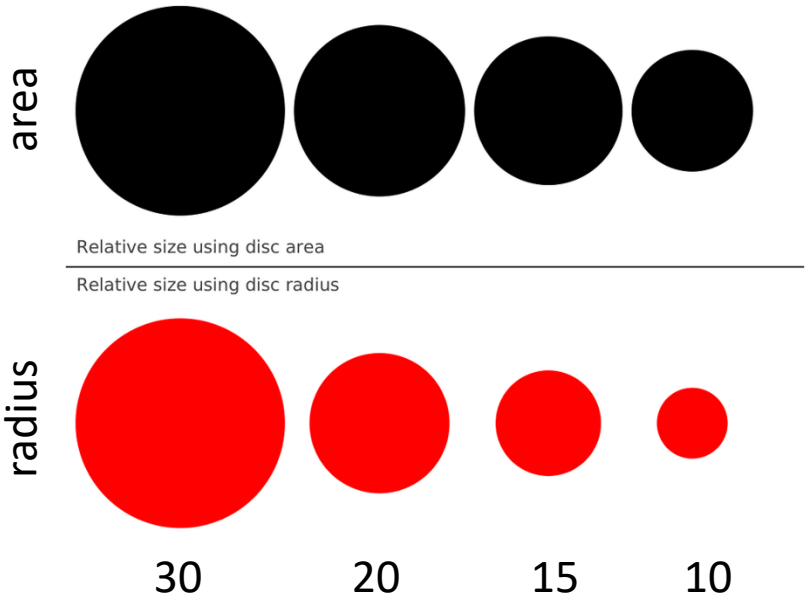
Rougier et al., 2014

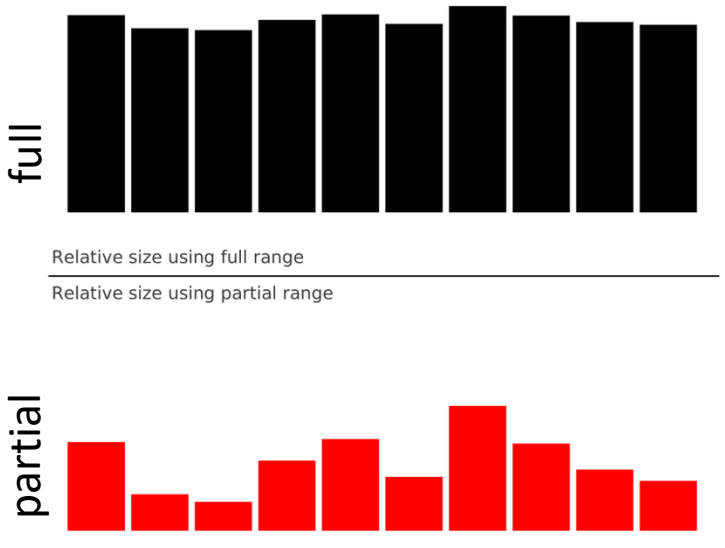
Rule 7: Do Not Mislead the Reader



Rougier et al., 2014

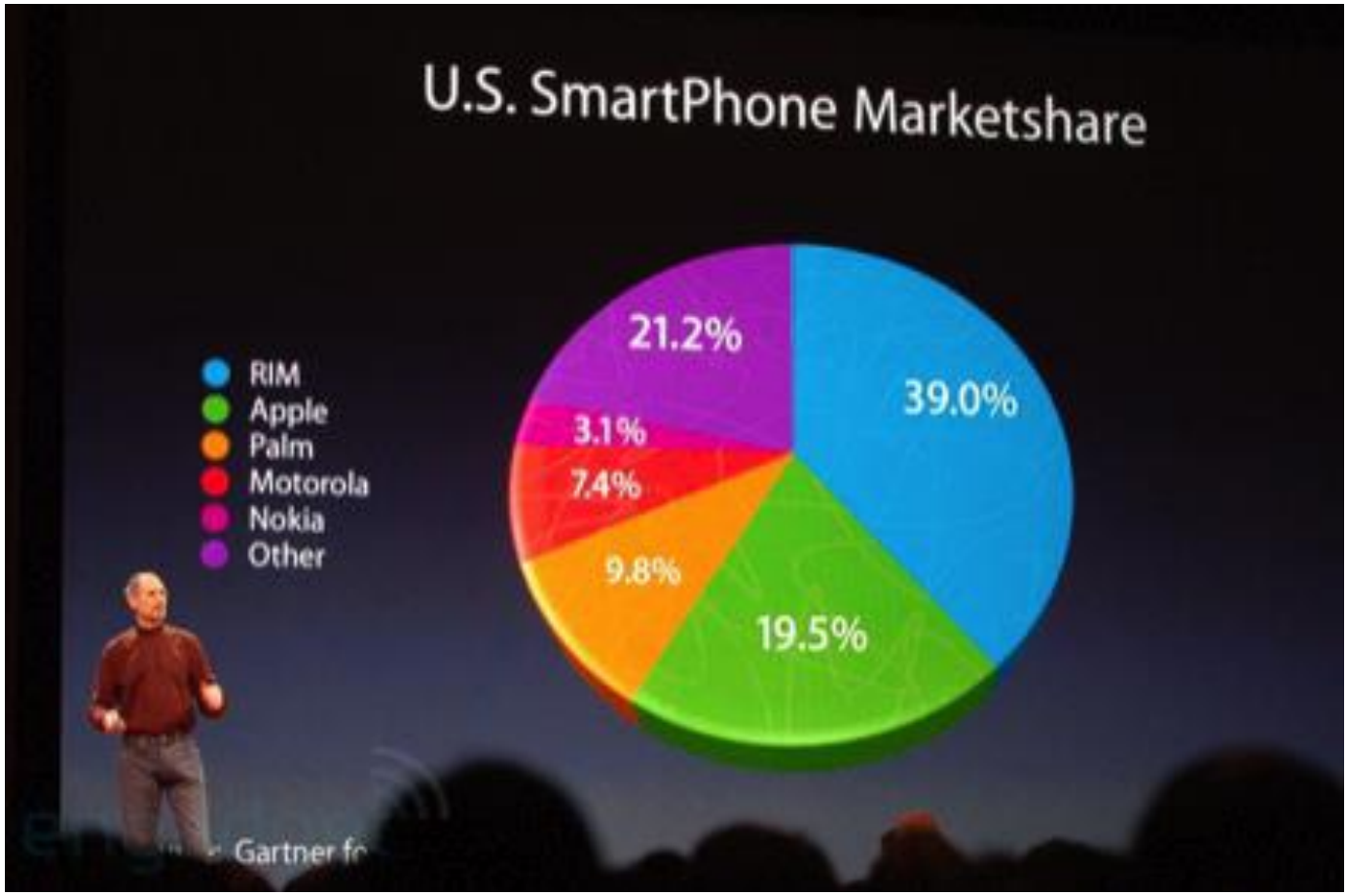
Rule 7: Do Not Mislead the Reader



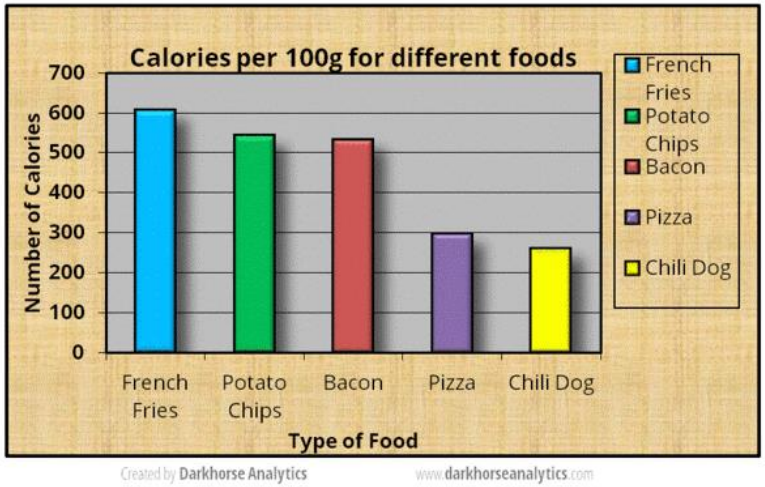


Rougier et al., 2014

Rule 7: Do Not Mislead the Reader

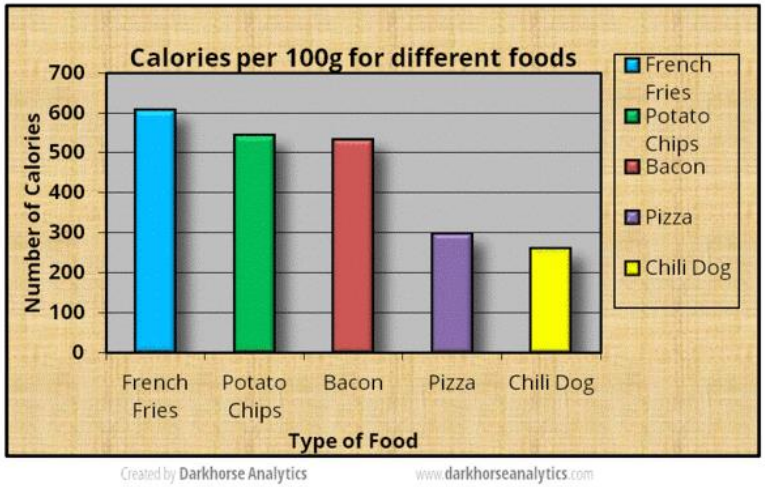


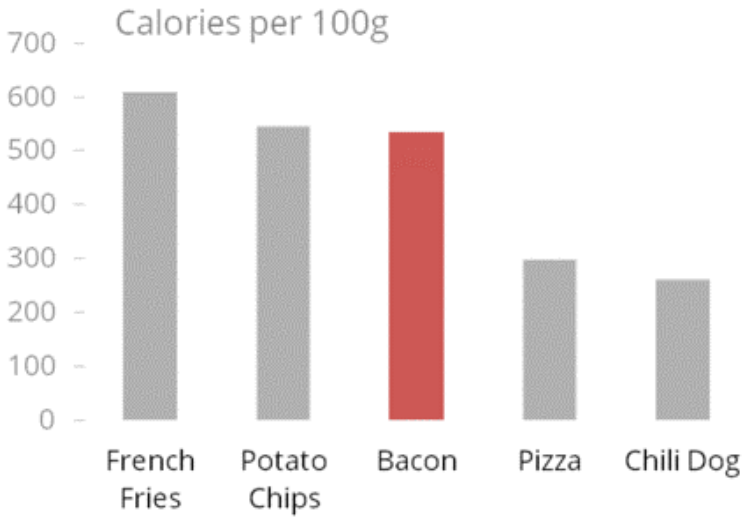
Rule 8: Avoid “Chartjunk”



darkhorseanalytics

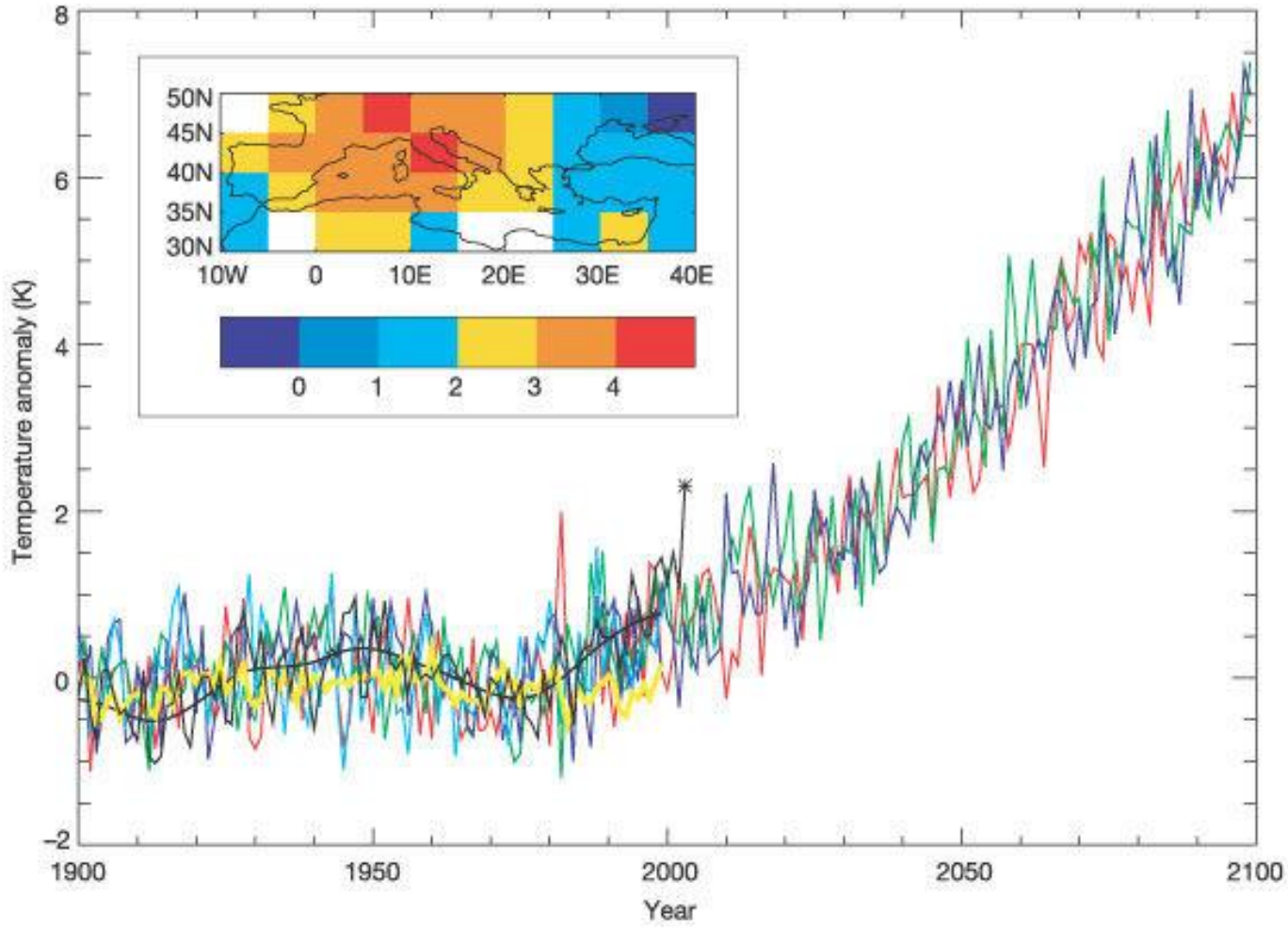
Rule 8: Avoid “Chartjunk”





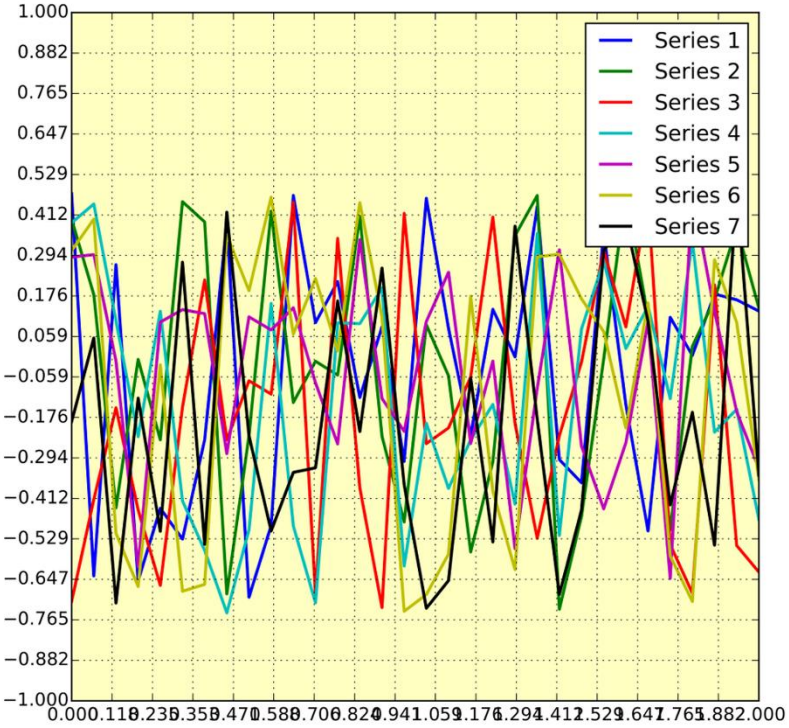
darkhorseanalytics

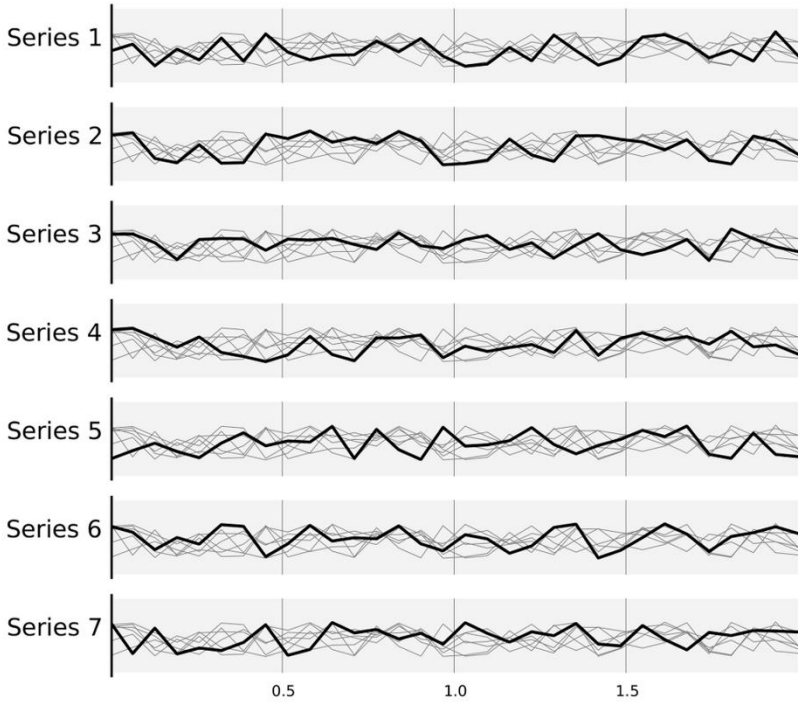
Rule 8: Avoid “Chartjunk”



Stott et al., 2004, Nature

Rule 8: Avoid “Chartjunk”



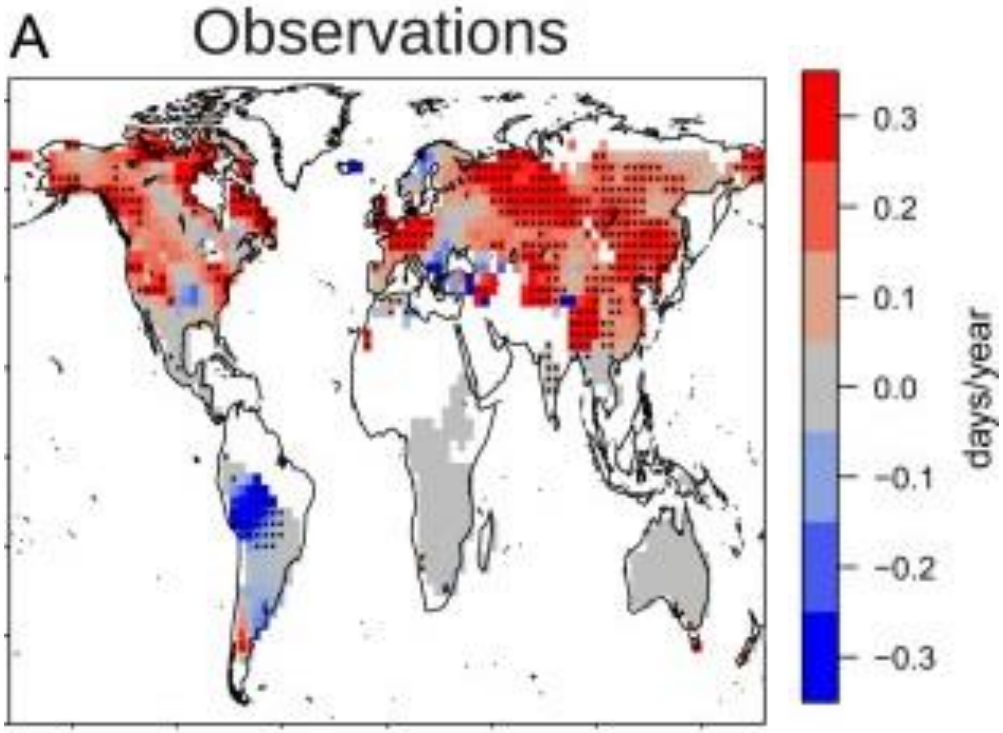


Rougier et al., 2014

Rule 9: Message Trumps Beauty

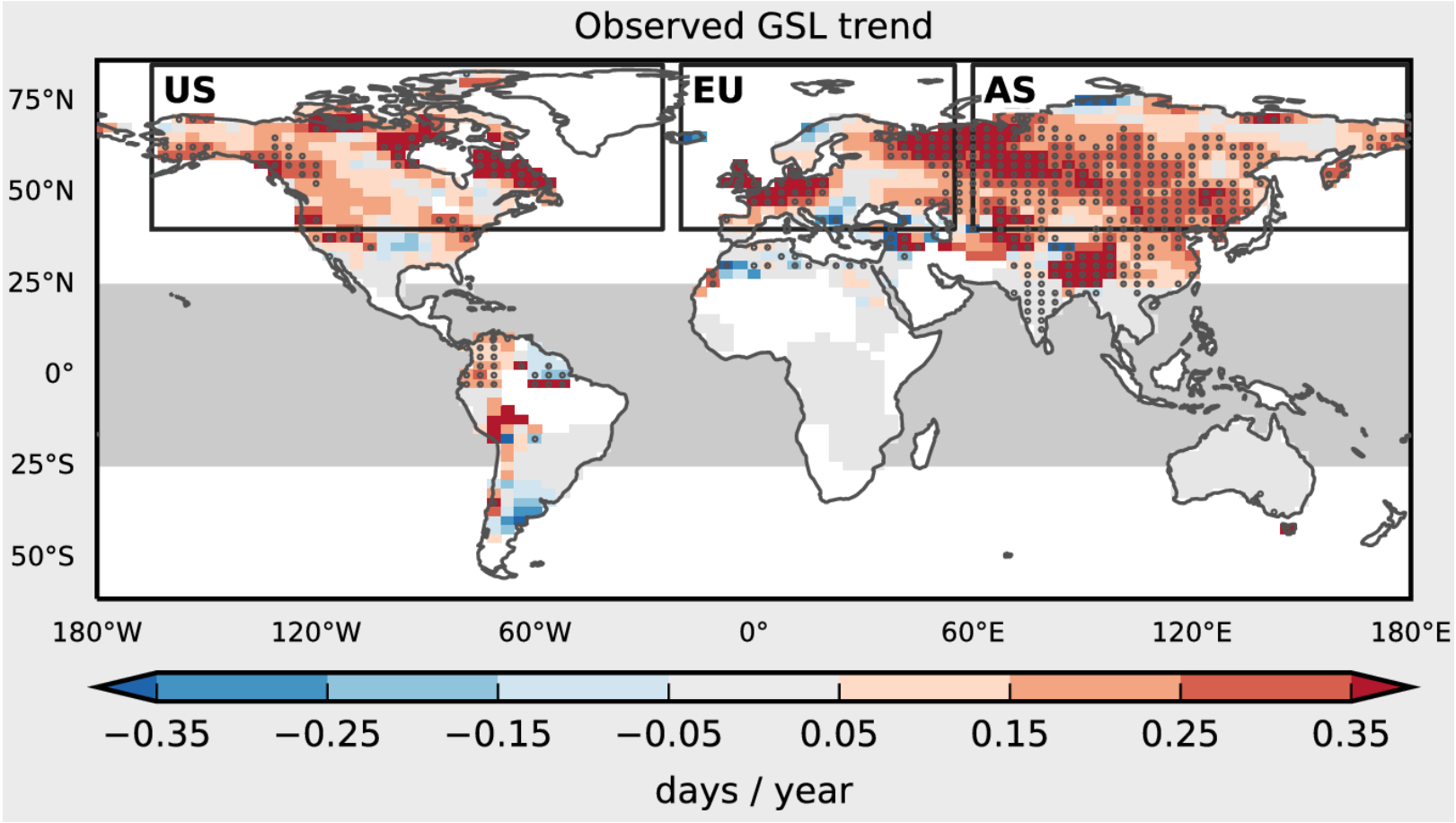
Rule 10: Get the Right Tool

Can you do better?



Müller et al., 2015

Can you do better?



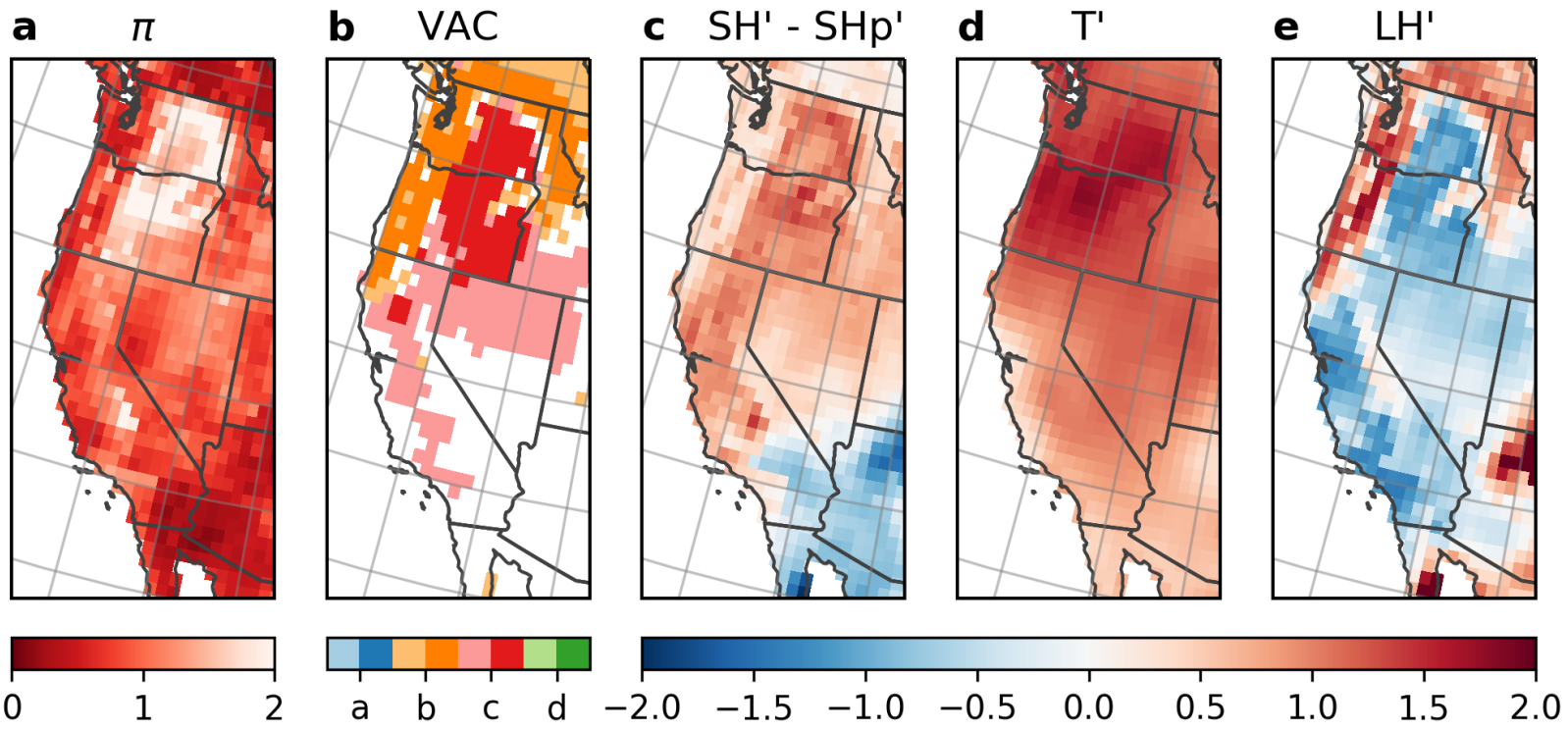
Hauser 2017

Schedule – Python Visualisation Course

Day 1	
09:00 – 10:30	**Part 0**: Introduction
10:30 – 11:00	Coffee Break
11:00 – 12:30	**Part 1**: Plotting in Python with matplotlib
12:30 – 13:30	Lunch Break
13:30 – 15:00	**Part 1**: Plotting in Python with matplotlib
15:00 – 15:30	Coffee Break
15:30 – 17:00	**Part 2**: Advanced plotting

Day 2	
09:00 – 10:30	**Part 3**: Plotting georeferenced data
10:30 – 11:00	Coffee Break
11:00 – 12:30	**Part 3**: Plotting georeferenced data
12:30 – 13:30	Lunch Break
13:30 – 15:00	**Part 3**: Plotting georeferenced data
15:00 – 15:30	Coffee Break
15:30 – 17:00	**Part 4**: User contributions

Georeferenced data with cartopy



Georeferenced data with cartopy

- Cartopy
- Basemap

Georeferenced data with cartopy

- Cartopy

- ~~Basemap~~

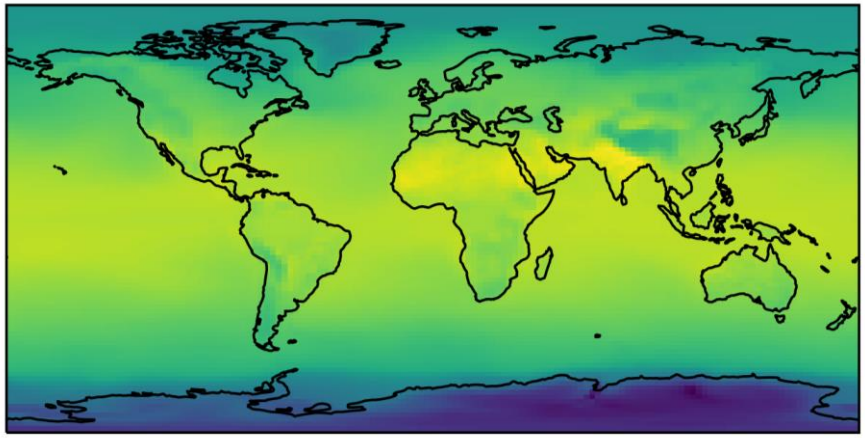
Georeferenced data with cartopy

```
ax = plt.axes(projection=ccrs.PlateCarree())

ax.coastlines()

h = ax.pcolormesh(LON, LAT, cesm.temp - 273.15,
                  transform=ccrs.PlateCarree())

ax.set_global()
```



Georeferenced data with cartopy

```
ax = plt.axes(projection=ccrs.PlateCarree())

ax.coastlines()

h = ax.pcolormesh(LON, LAT, cesm.temp - 273.15,
                  transform=ccrs.PlateCarree())
```

- projection: type of map
  - can change

- transform: coordinate system of data
  - Hardly ever changes
  - PlateCarree -> coordinates in lon/ lat

Plotting georeferenced data

- scatter points data on maps
- contour and pcolormesh
- colorbars
- stippling
- multiple map plots
- trajectories
- rotated pole data

Schedule – Python Visualisation Course

Day 1	
09:00 – 10:30	**Part 0**: Introduction
10:30 – 11:00	Coffee Break
11:00 – 12:30	**Part 1**: Plotting in Python with matplotlib
12:30 – 13:30	Lunch Break
13:30 – 15:00	**Part 1**: Plotting in Python with matplotlib
15:00 – 15:30	Coffee Break
15:30 – 17:00	**Part 2**: Advanced plotting

Day 2	
09:00 – 10:30	**Part 3**: Plotting georeferenced data
10:30 – 11:00	Coffee Break
11:00 – 12:30	**Part 3**: Plotting georeferenced data
12:30 – 13:30	Lunch Break
13:30 – 15:00	**Part 3**: Plotting georeferenced data
15:00 – 15:30	Coffee Break
15:30 – 17:00	**Part 4**: User contributions

Colormaps

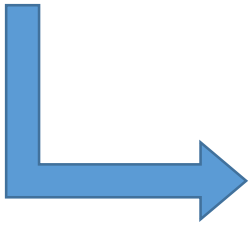
```
([[ 0.40784314,  0.40784314,  0.40784314, ...,  0.42745098,
    0.42745098,  0.42745098],
  [ 0.41176471,  0.41176471,  0.41176471, ...,  0.42745098,
    0.42745098,  0.42745098],
  [ 0.41960785,  0.41568628,  0.41568628, ...,  0.43137255,
    0.43137255,  0.43137255],
  ...,
  [ 0.43921569,  0.43529412,  0.43137255, ...,  0.45490196,
    0.4509804 ,  0.4509804 ],
  [ 0.44313726,  0.44313726,  0.43921569, ...,  0.4509804 ,
    0.44705883,  0.44705883],
  [ 0.44313726,  0.4509804 ,  0.4509804 , ...,  0.44705883,
    0.44705883,  0.44313726]], dtype=float32)
```

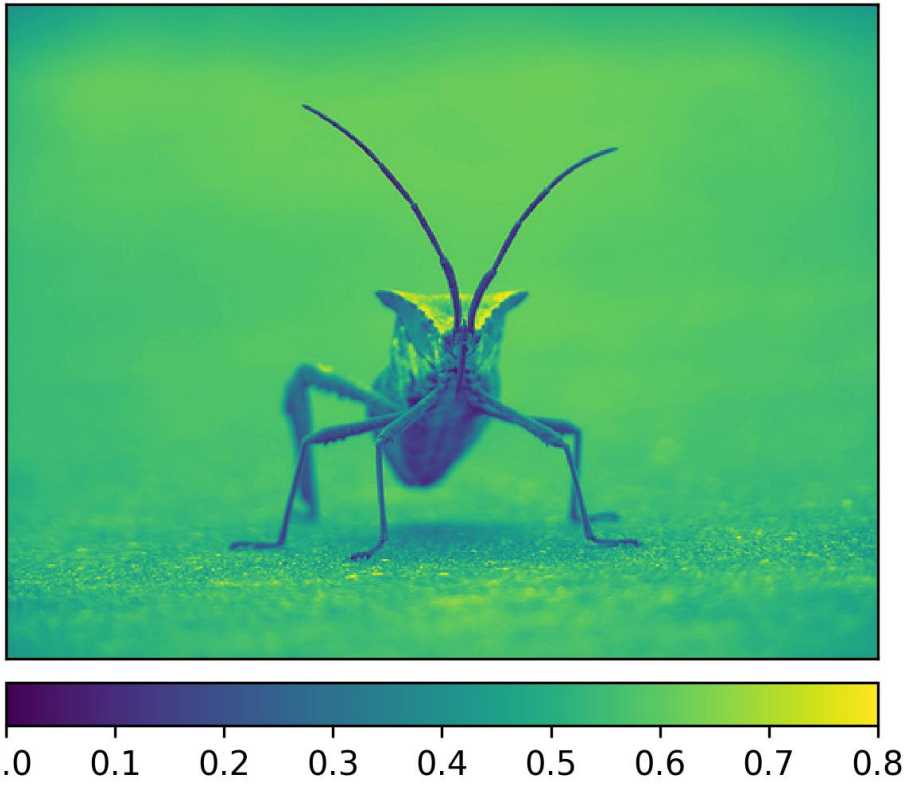
Colormaps

```
([[ 0.40784314,  0.40784314,  0.40784314, ...,  0.42745098,
    0.42745098,  0.42745098],
  [ 0.41176471,  0.41176471,  0.41176471, ...,  0.42745098,
    0.42745098,  0.42745098],
  [ 0.41960785,  0.41568628,  0.41568628, ...,  0.43137255,
    0.43137255,  0.43137255],
  ...,
  [ 0.43921569,  0.43529412,  0.43137255, ...,  0.45490196,
    0.4509804 ,  0.4509804 ],
  [ 0.44313726,  0.44313726,  0.43921569, ...,  0.4509804 ,
    0.44705883,  0.44705883],
  [ 0.44313726,  0.4509804 ,  0.4509804 , ...,  0.44705883,
    0.44705883,  0.44313726]], dtype=float32)
```

Colormaps

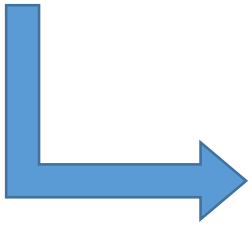
```
([[ 0.40784314,  0.40784314,  0.40784314, ...,  0.42745098,
    0.42745098,  0.42745098],
  [ 0.41176471,  0.41176471,  0.41176471, ...,  0.42745098,
    0.42745098,  0.42745098],
  [ 0.41960785,  0.41568628,  0.41568628, ...,  0.43137255,
    0.43137255,  0.43137255],
  ...,
  [ 0.43921569,  0.43529412,  0.43137255, ...,
    0.4509804 ,  0.4509804 ],
  [ 0.44313726,  0.44313726,  0.43921569, ...,
    0.44705883,  0.44705883],
  [ 0.44313726,  0.4509804 ,  0.4509804 , ...,
    0.44705883,  0.44313726]], dtype=float32)
```

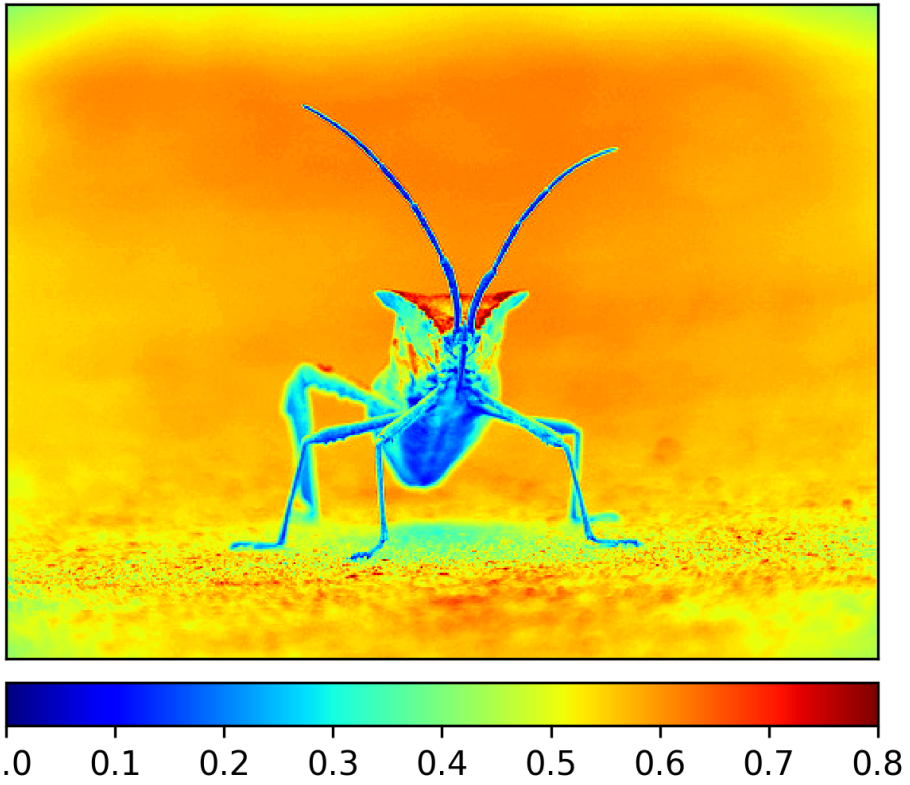




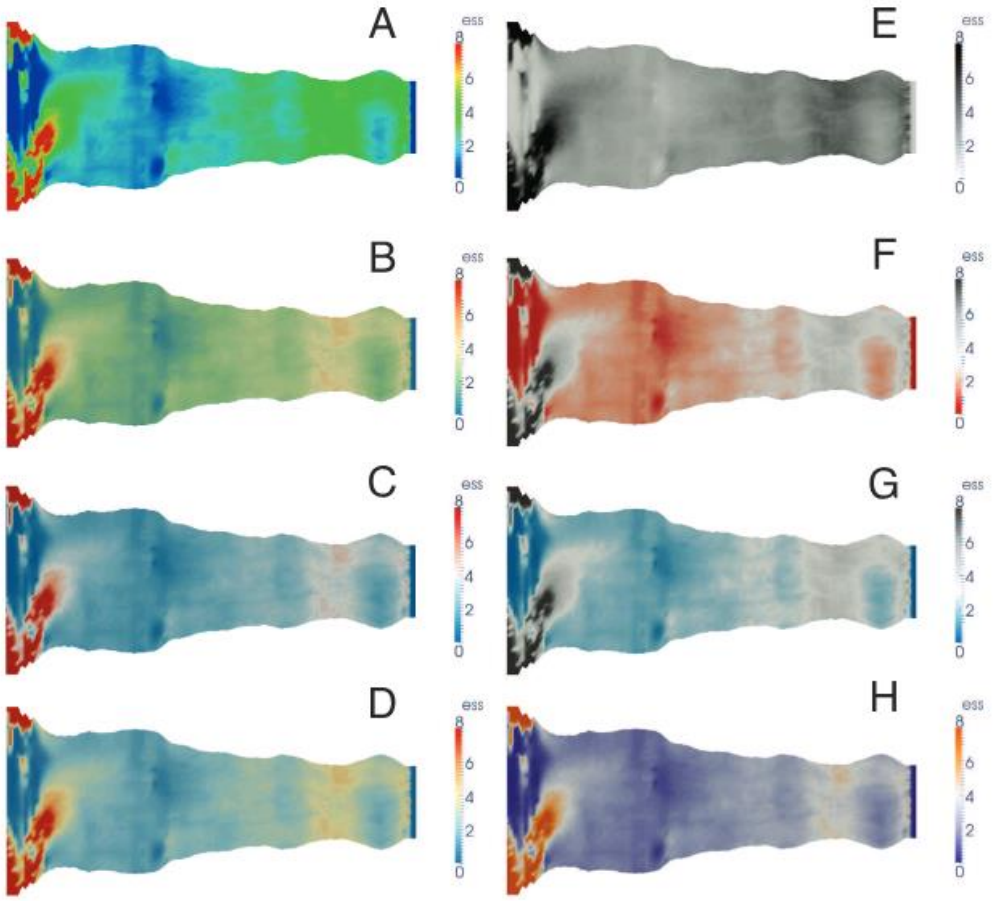
Colormaps: choice matters

```
([[ 0.40784314,  0.40784314,  0.40784314, ...,  0.42745098,
    0.42745098,  0.42745098],
  [ 0.41176471,  0.41176471,  0.41176471, ...,  0.42745098,
    0.42745098,  0.42745098],
  [ 0.41960785,  0.41568628,  0.41568628, ...,  0.43137255,
    0.43137255,  0.43137255],
  ...,
  [ 0.43921569,  0.43529412,  0.43137255, ...,
    0.4509804 ,  0.4509804 ],
  [ 0.44313726,  0.44313726,  0.43921569, ...,
    0.44705883,  0.44705883],
  [ 0.44313726,  0.4509804 ,  0.4509804 , ...,
    0.44705883,  0.44313726]], dtype=float32)
```





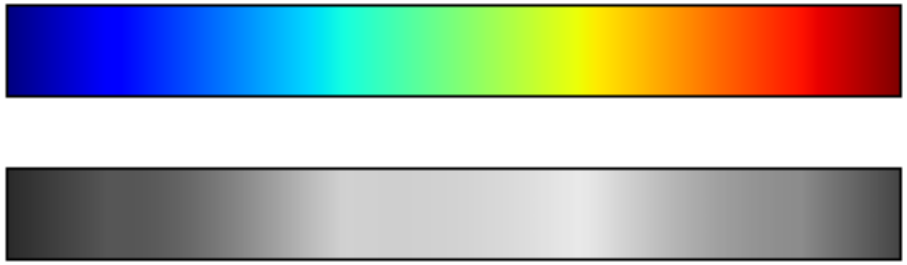
Colormaps: choice matters



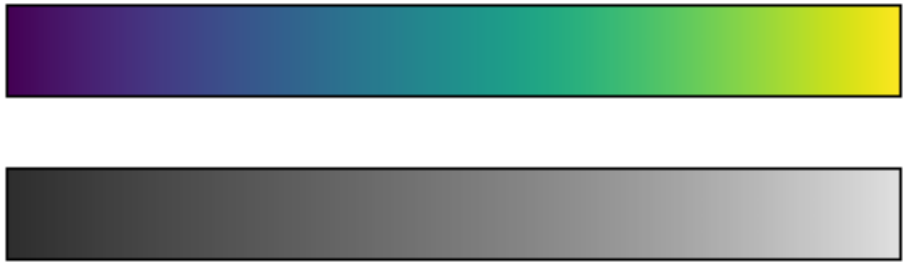
Borkin et al., 2011

Colormaps: choice matters

Jet



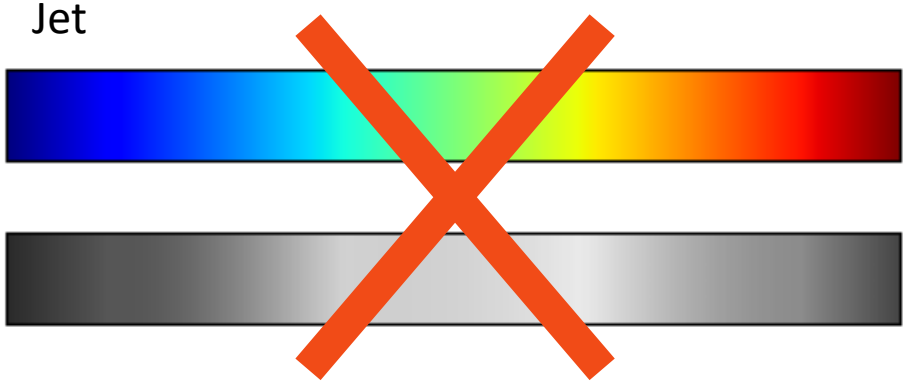
Viridis



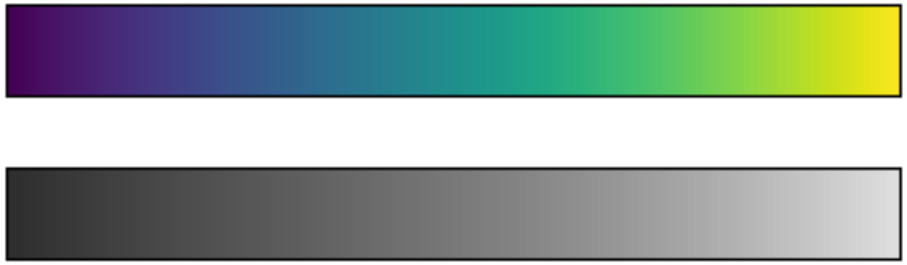
jakevdp

Colormaps: choice matters

Jet



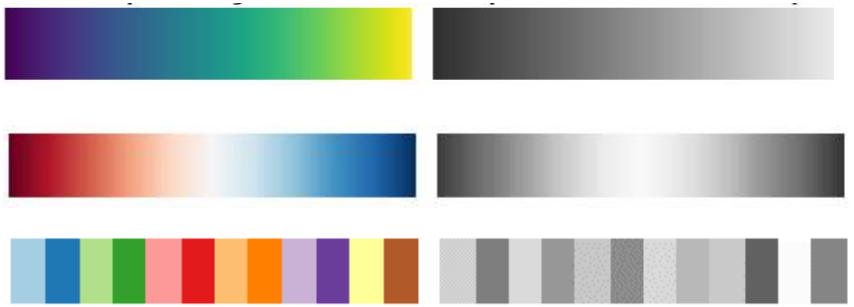
Viridis



jakevdp

Colormaps: type of data

- Sequential
- Diverging
- Qualitative



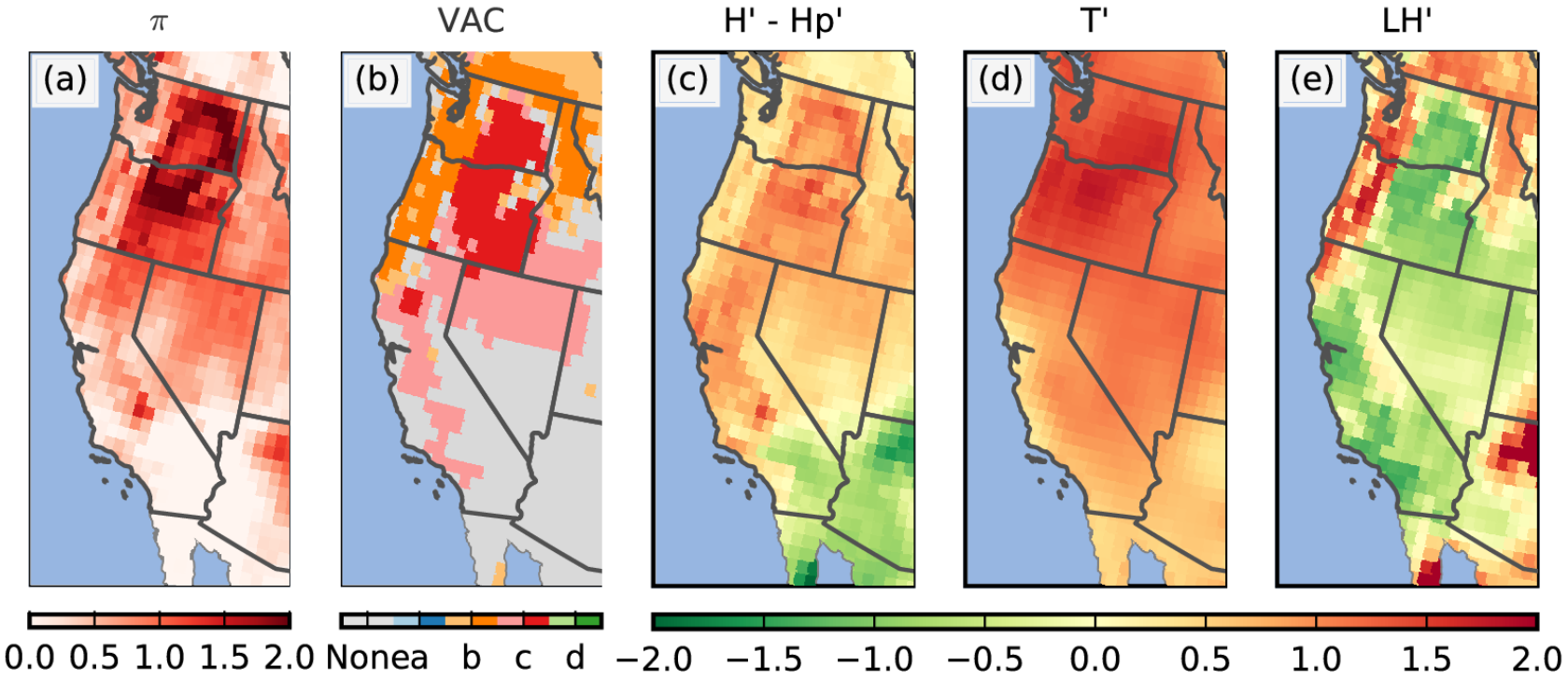
matplotlib.org

Colormaps: type of data

Sequential

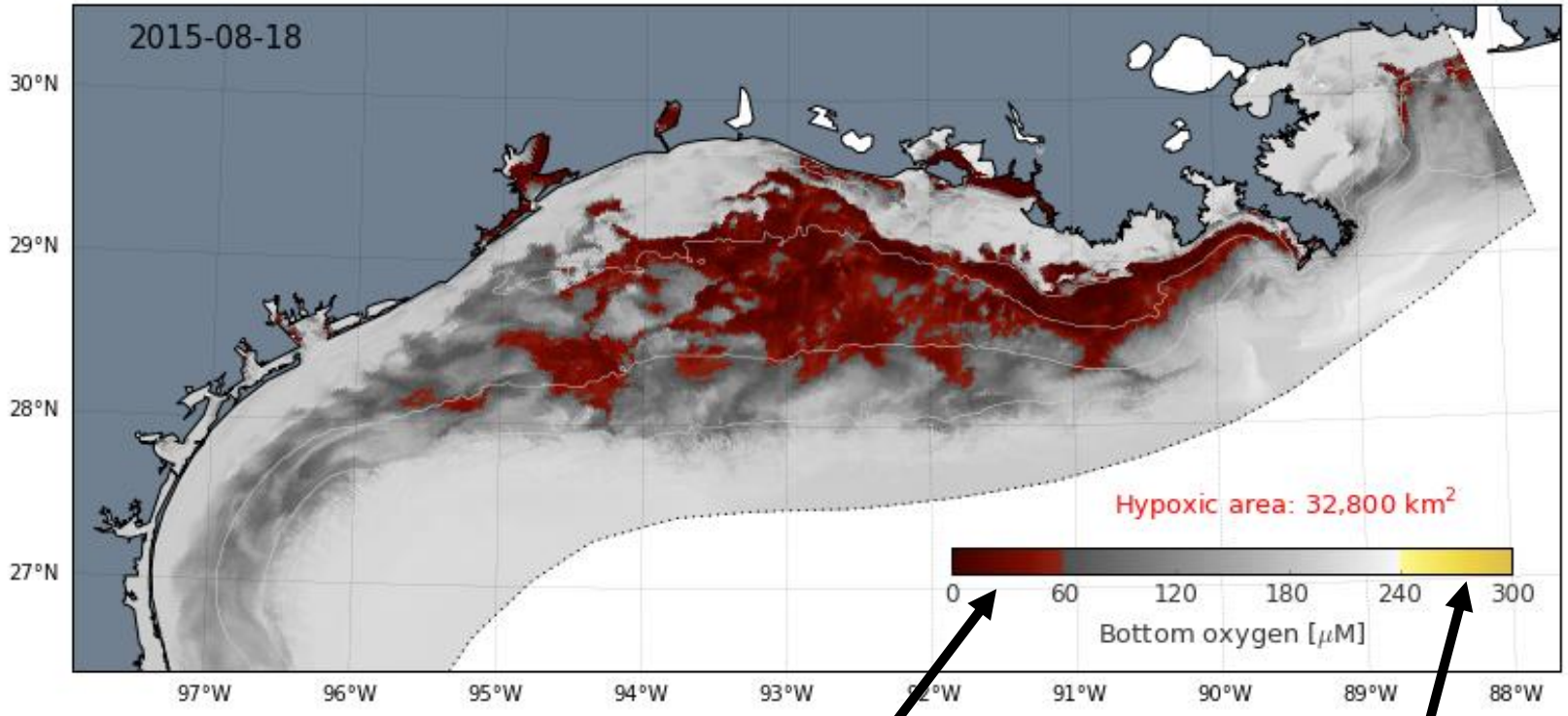
Qualitative

Diverging



Philip et al., 2017

Colormaps: non-diverging breakpoints

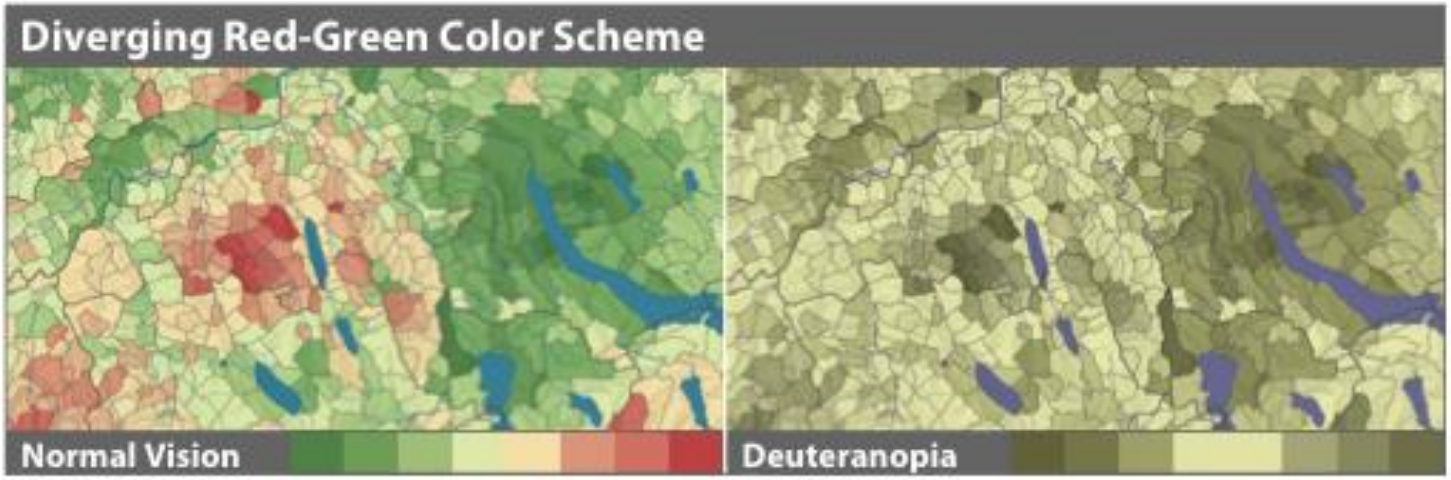


hypoxic

supersaturated

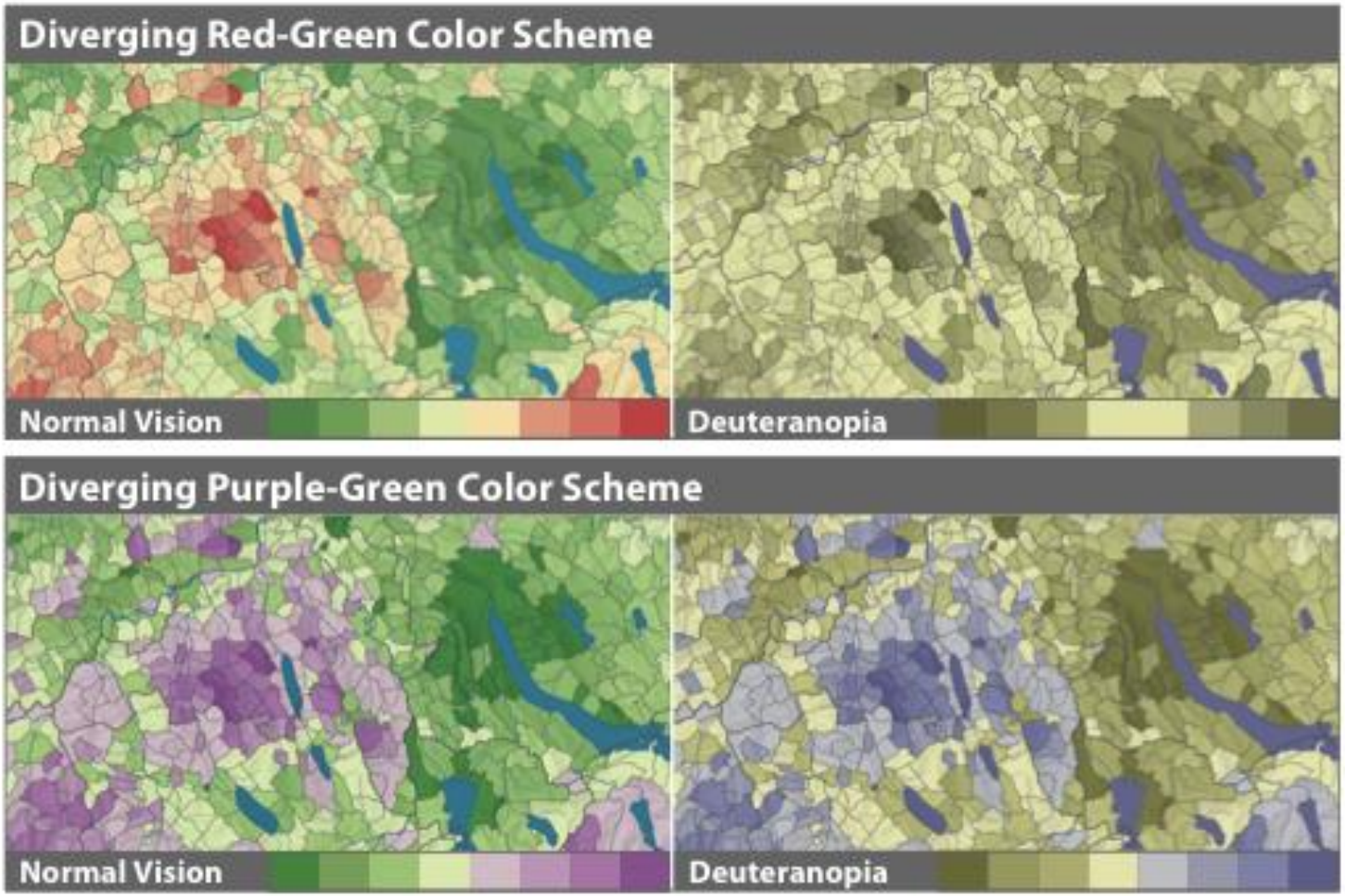
cmocean

Colormaps: color blindness



vis4.net

Colormaps: color blindness



vis4.net

Resources

- Choosing Colormaps
  - Color Brewer
  - Cmocean
- Color blindness
  - vischeck.com

Schedule – Python Visualisation Course

Day 1	
09:00 – 10:30	**Part 0**: Introduction
10:30 – 11:00	Coffee Break
11:00 – 12:30	**Part 1**: Plotting in Python with matplotlib
12:30 – 13:30	Lunch Break
13:30 – 15:00	**Part 1**: Plotting in Python with matplotlib
15:00 – 15:30	Coffee Break
15:30 – 17:00	**Part 2**: Advanced plotting

Day 2	
09:00 – 10:30	**Part 3**: Plotting georeferenced data
10:30 – 11:00	Coffee Break
11:00 – 12:30	**Part 3**: Plotting georeferenced data
12:30 – 13:30	Lunch Break
13:30 – 15:00	**Part 3**: Plotting georeferenced data
15:00 – 15:30	Coffee Break
15:30 – 17:00	**Part 4**: User contributions

Schedule – Python Visualisation Course

Day 1	
09:00 – 10:30	**Part 0**: Introduction
10:30 – 11:00	Coffee Break
11:00 – 12:30	**Part 1**: Plotting in Python with matplotlib
12:30 – 13:30	Lunch Break
13:30 – 15:00	**Part 1**: Plotting in Python with matplotlib
15:00 – 15:30	Coffee Break
15:30 – 17:00	**Part 2**: Advanced plotting

Day 2	
09:00 – 10:30	**Part 3**: Plotting georeferenced data
10:30 – 11:00	Coffee Break
11:00 – 12:30	**Part 3**: Plotting georeferenced data
12:30 – 13:30	Lunch Break
13:30 – 15:00	**Part 3**: Plotting georeferenced data
15:00 – 15:30	Coffee Break
15:30 – 17:00	**Part 4**: User contributions

User contributions

- Advanced plots from the community

References

- Hauser et al., 2016, GRL (https://doi.org/10.1002/2016GL068036)
- Hauser et al., 2017, GMD (https://doi.org/10.5194/gmd-10-1665-2017, 2017)
- Hauser, 2017 (https://doi.org/10.3929/ethz-b-000179749)
- Maraun et al., 2017 NCC (https://doi.org/10.1038/nclimate3418)
- Müller et al., 2015, WACE (https://doi.org/10.1016/j.wace.2015.04.001)
- Rougier et al., 2014, PLOS (https://doi.org/10.1371/journal.pcbi.1003833)
- Stott et al., 2004, Nature (https://doi.org/10.1038/nature03089)
- Van Heerwaarden et al., 2014, Biogeosciences (https://doi.org/doi:10.5194/bg-11-6159-2014)