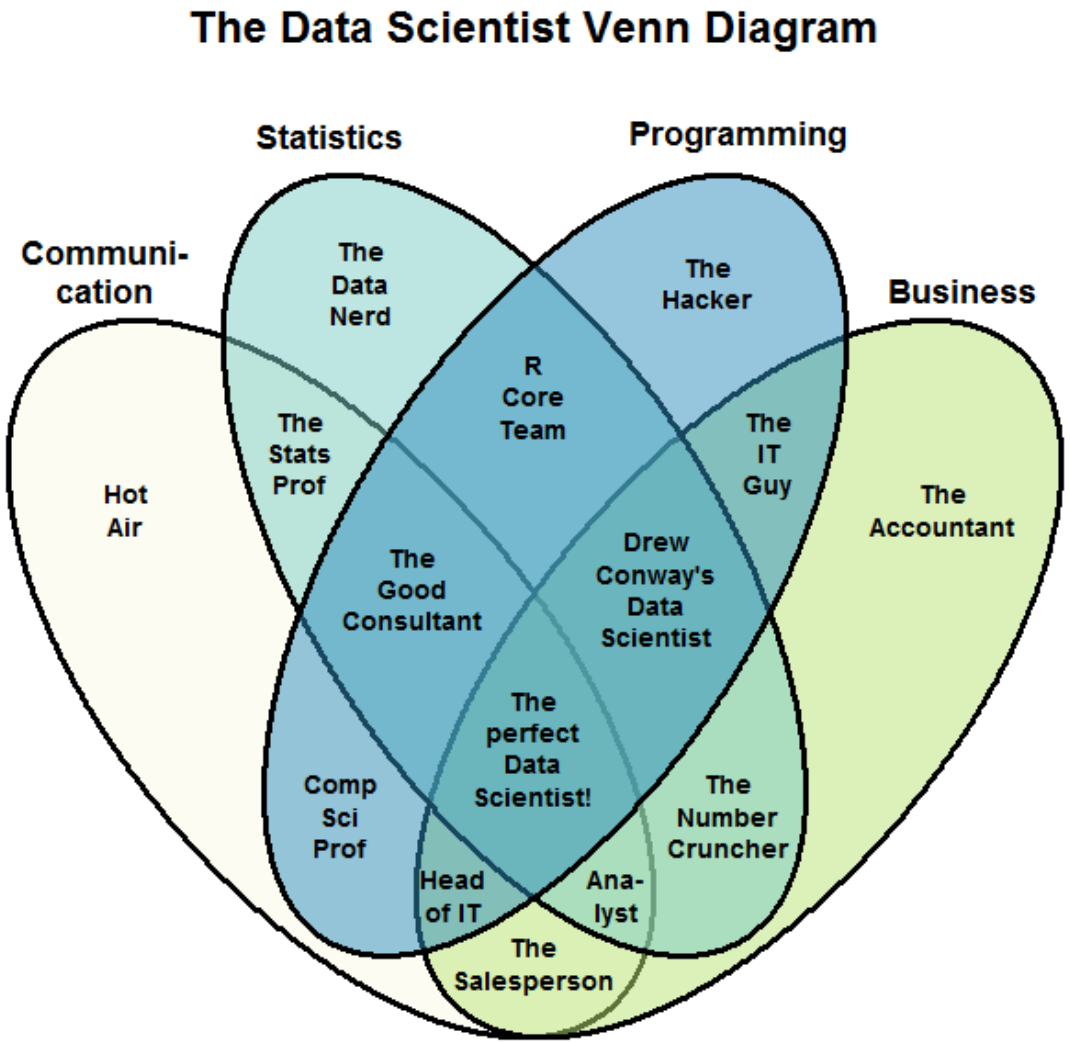
The Python Ecosystem for Data Science: A Guided Tour

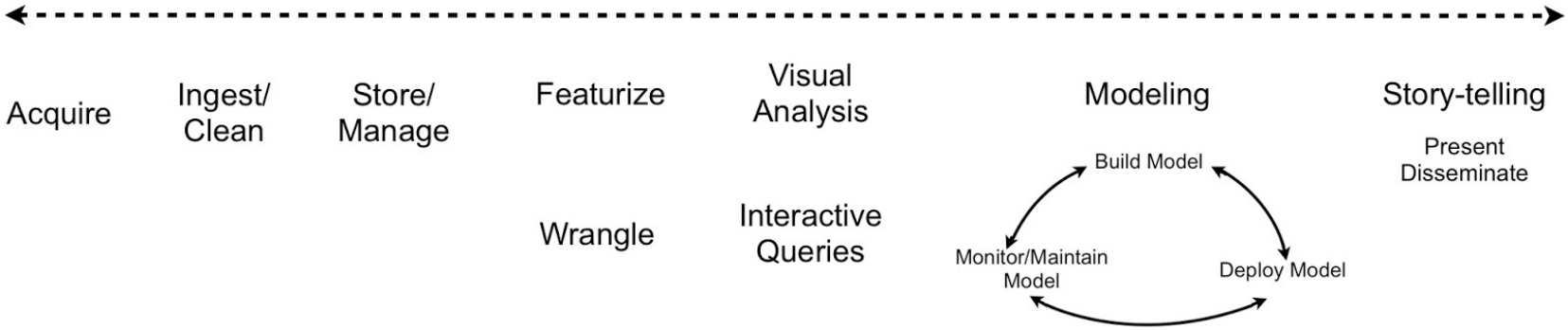
PyData Warsaw 2017 | at the Copernicus Science Centre | 19-20 October 2017

Christian Staudt | Independent Data Scientist



Source: Stephan Kolassa @ Stackexchange (https://datascience.stackexchange.com/questions/2403/data-science-without-knowledge-of-a-specific-topic-is-it-worth-pursuing-as-a-ca)

The (?) Data Science Workflow



Source: Ben Lorica @ O'Reilly (https://www.oreilly.com/ideas/data-analysis-just-one-component-of-the-data-science-workflow)

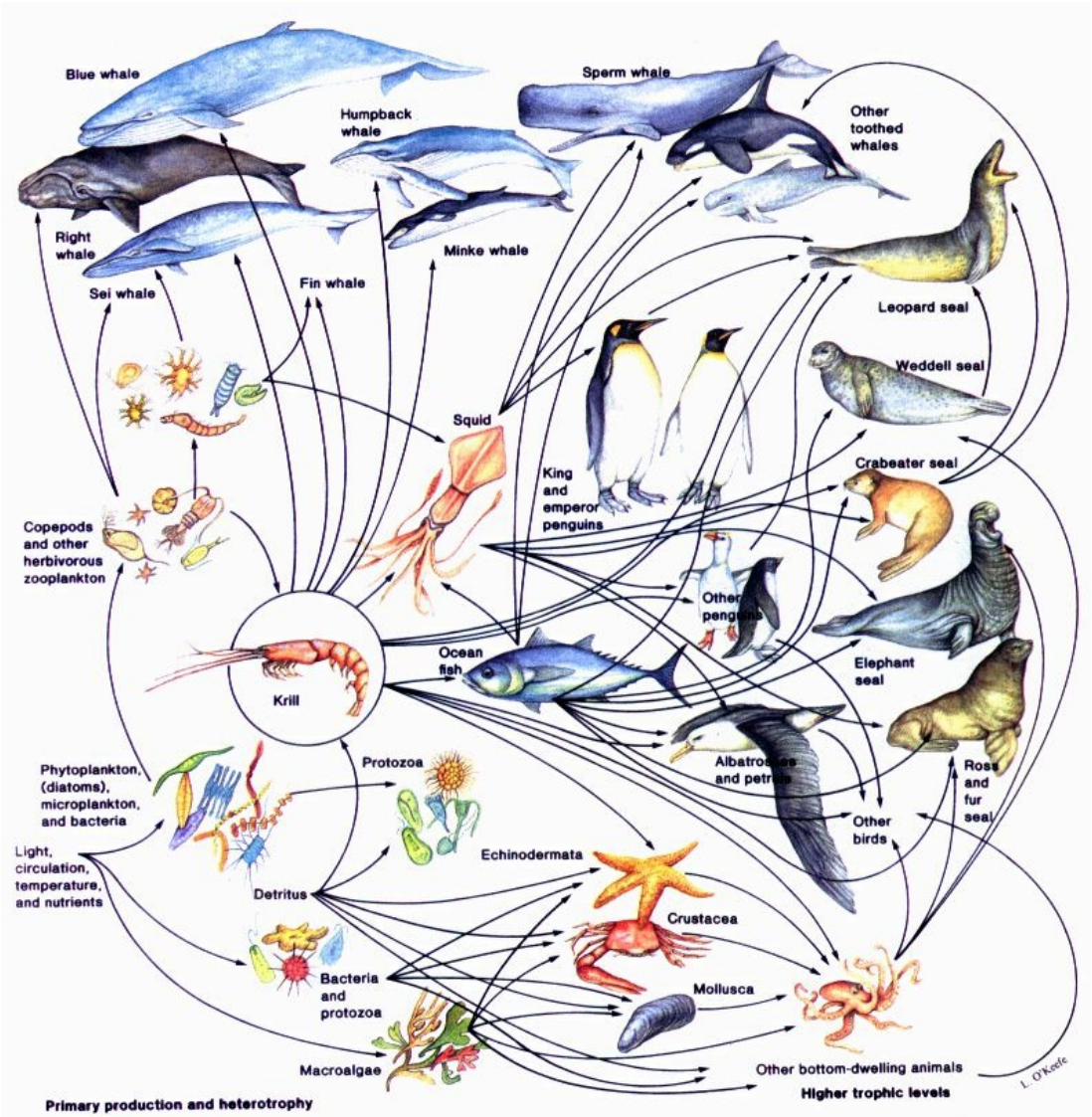
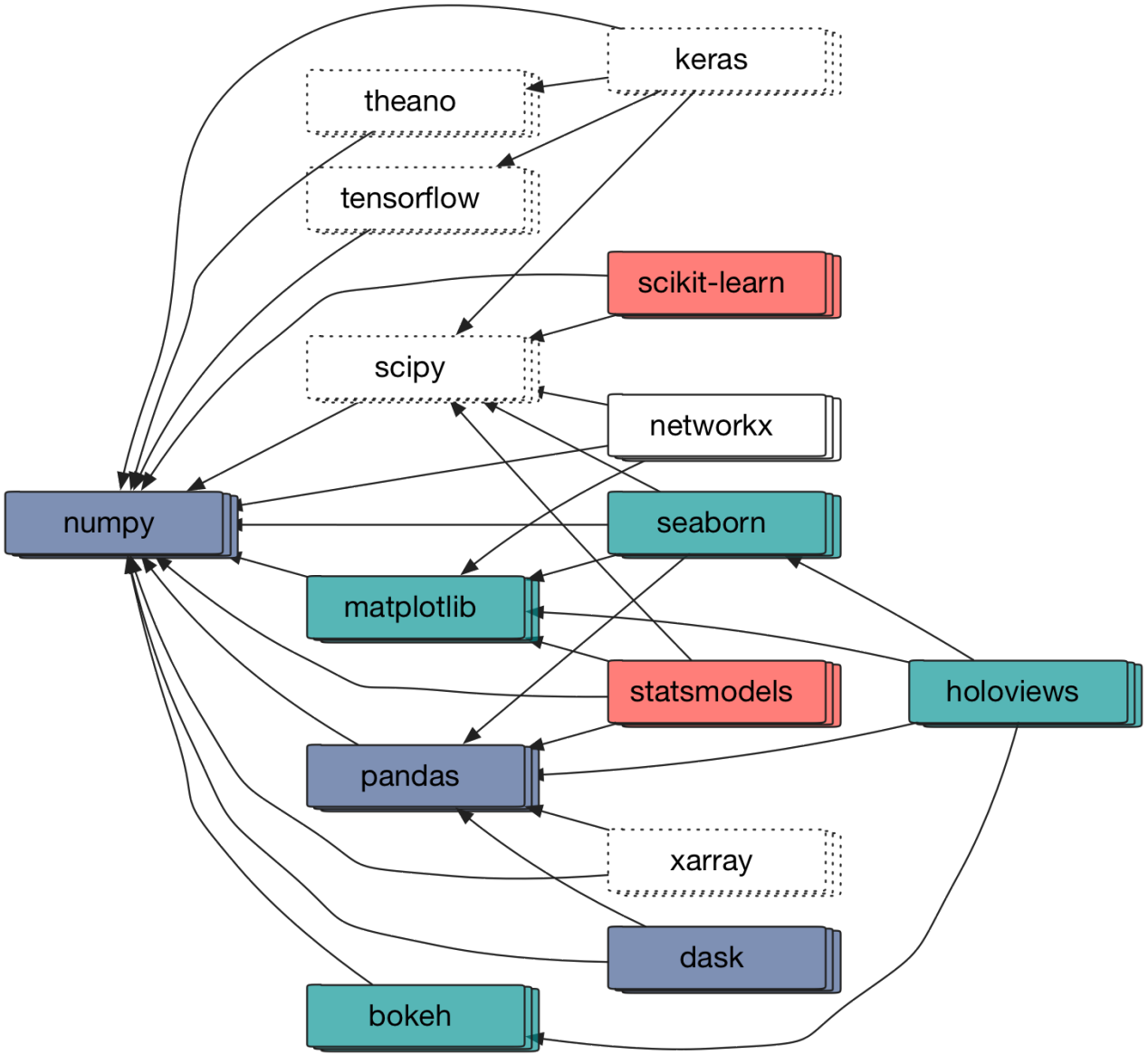


FIGURE 15.3 Food Webs. An Antarctic food web. Small crustaceans called krill support nearly all life in Antarctica. Krill are eaten by 6 species of baleen whales, 20 species of squid, over 100 species of fish, 35 species of birds, and 7 species of seals. Krill feed on algae, protozoa, other small crustaceans, and various larvae.


keras
theano
tensorflow
scikit-learn
scipy
networkx
numpy
seaborn
matplotlib
statsmodels
holoviews
pandas
xarray
dask
bokeh

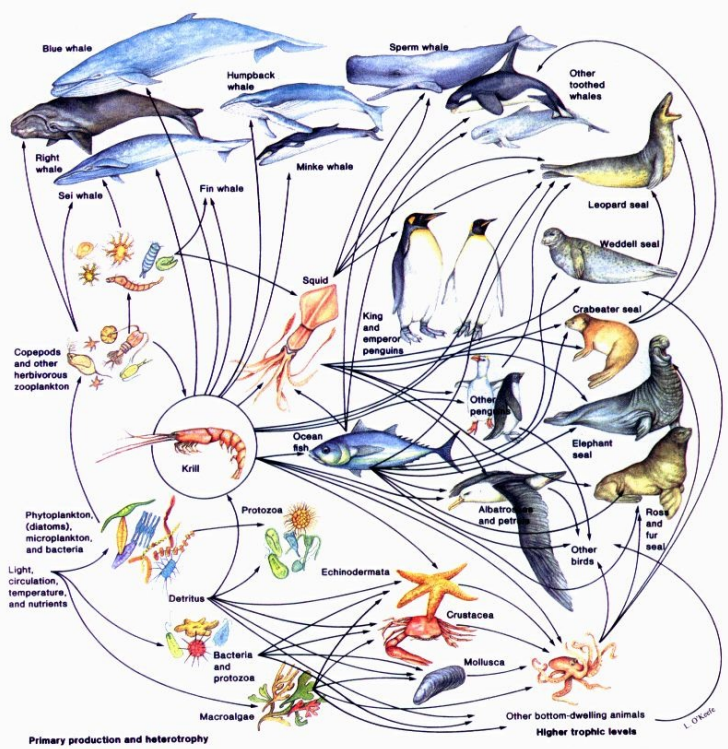
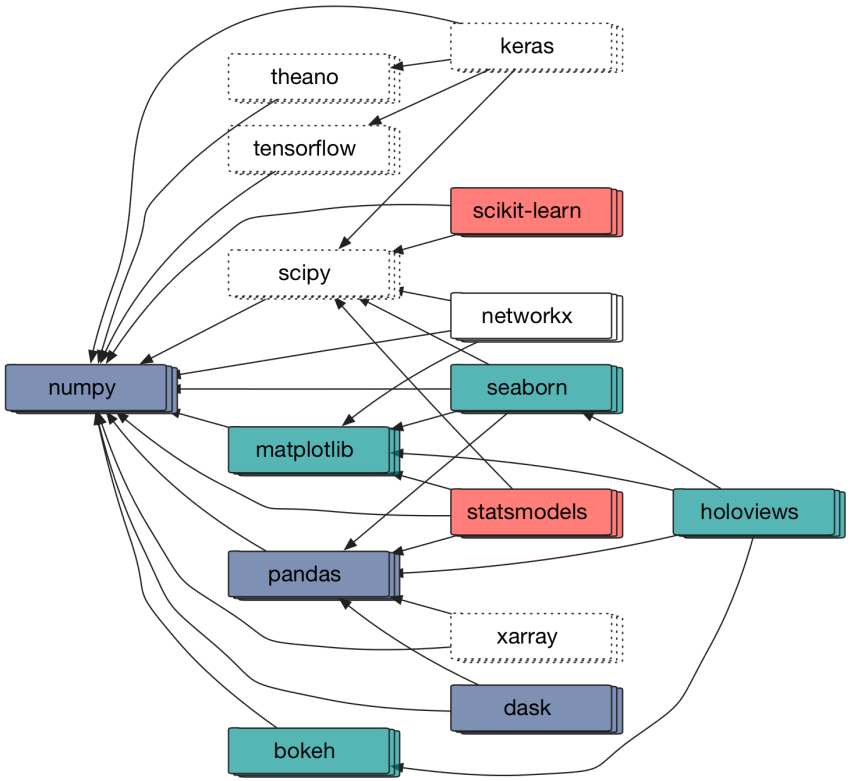
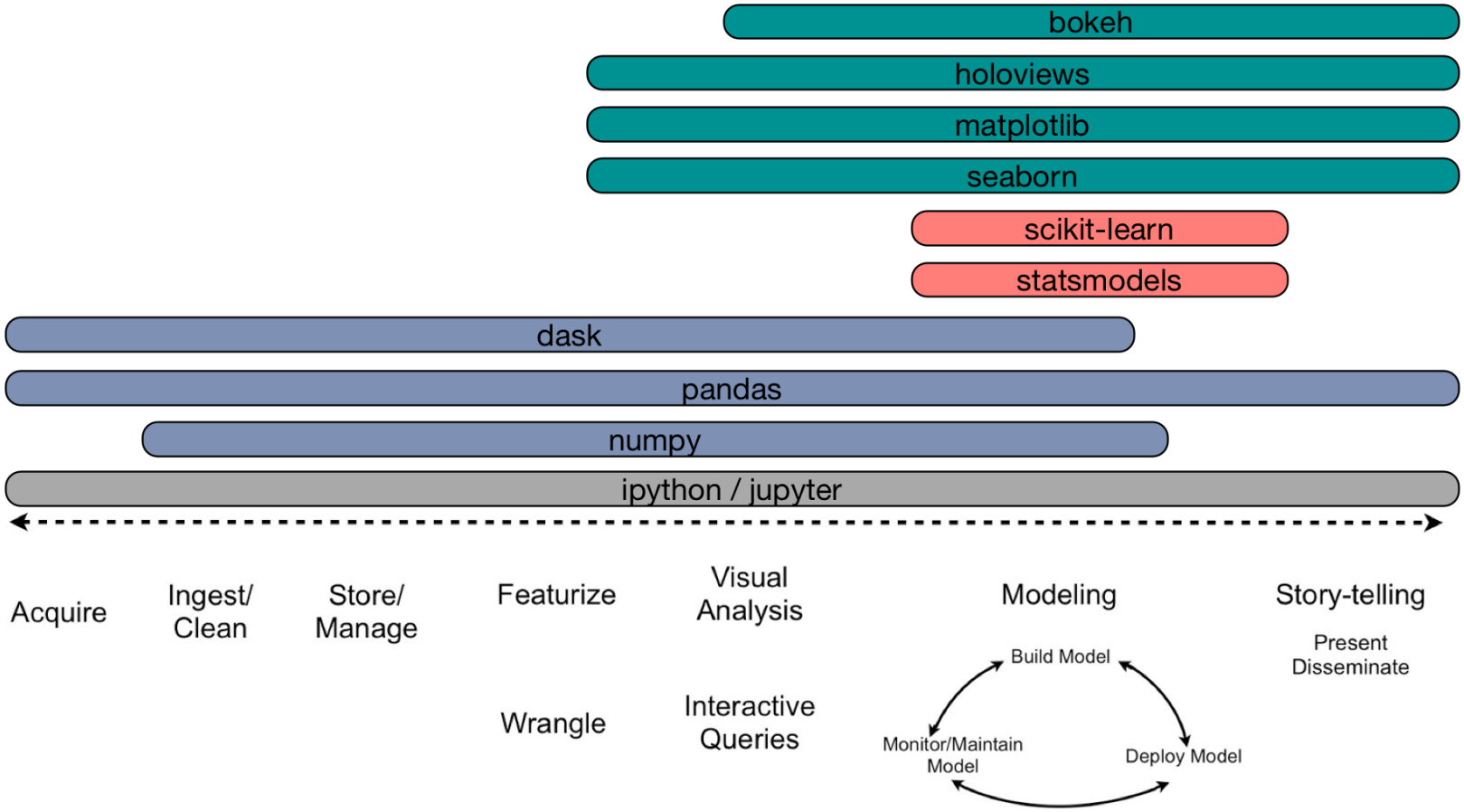


FIGURE 15.3 Food Webs. An Antarctic food web. Small crustaceans called krill support nearly all life in Antarctica. Krill are eaten by 6 species of baleen whales, 20 species of squid, over 100 species of fish, 35 species of birds, and 7 species of seals. Krill feed on algae, protozoa, other small crustaceans, and various larvae.



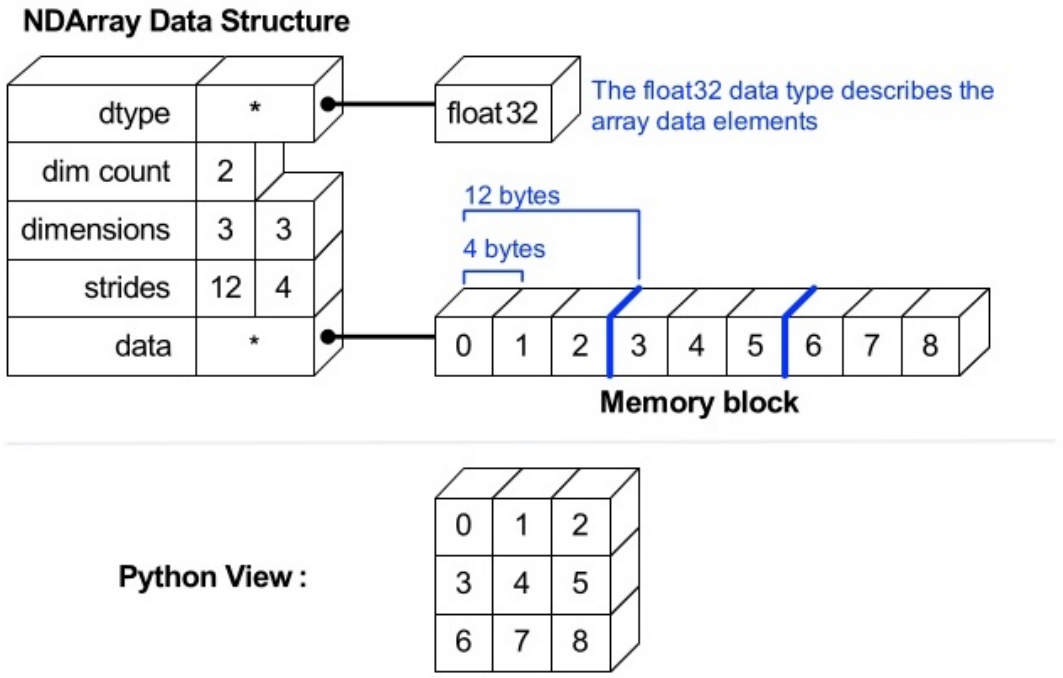
Wrangling


bokeh
holoviews
matplotlib
seaborn
scikit-learn
statsmodels
dask
pandas
numpy
ipython / jupyter
Acquire
Ingest/ Clean
Store/ Manage
Featurize
Wrangle
Visual Analysis
Interactive Queries
Modeling
Build Model
Deploy Model
Monitor/Maintain Model
Story-telling
Present
Disseminate

numpy

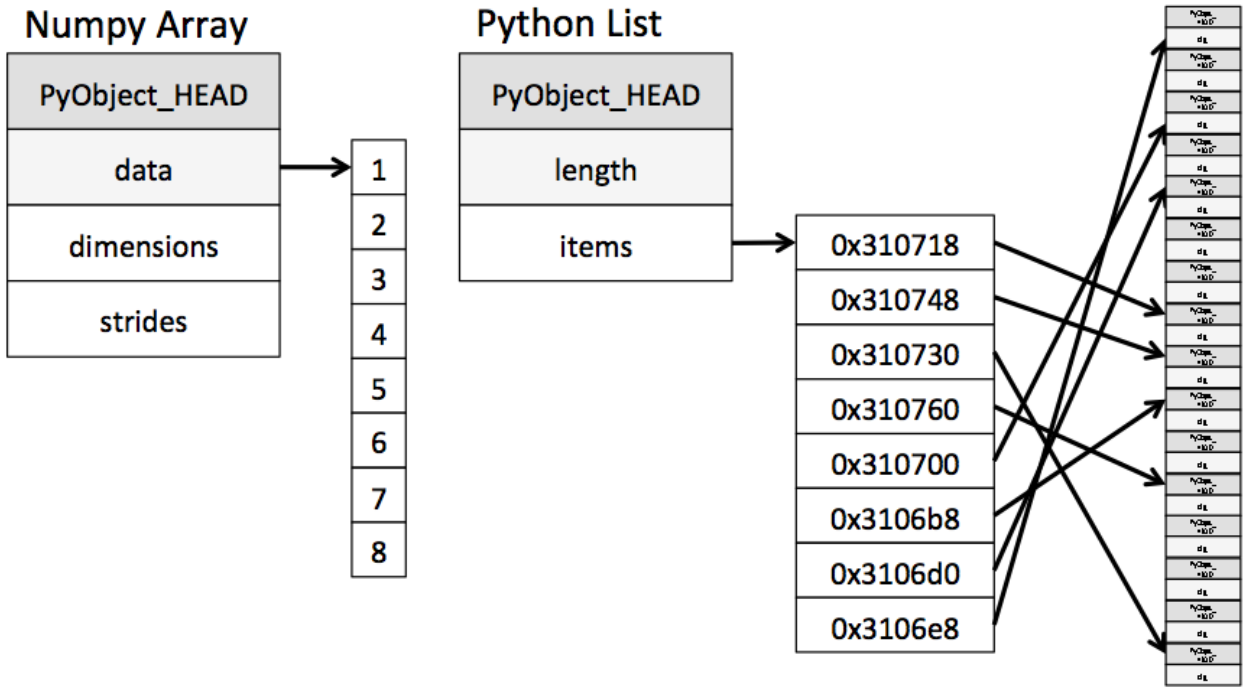
- the fundamental package for numeric computing in Python
- provides
  - n-dimensional array object
  - powerful array functions
  - math: linear algebra, random numbers, ...

numpy ndarray



Source: Travis Oliphant @ SIAM 2011 (https://www.slideshare.net/enthought/numpy-talk-at-siam)

numpy array vs python list



Source: Python Data Science Handbook by Jake VanderPlas (https://jakevdp.github.io/PythonDataScienceHandbook/02.01-understanding-data-types.html)

understand numpy - lose your loops

In [5]:
```
n = int(1e6)
```

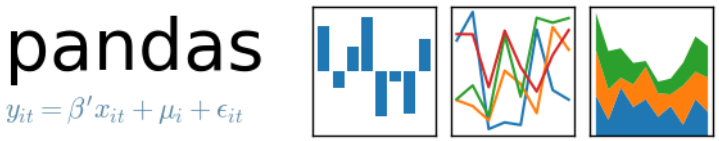
In [6]:
```
%%timeit
a = [random.random() for i in range(n)]
b = [math.log(x) for x in a]
```

```
440 ms ± 66.3 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

In [7]:
```
%%timeit
a = numpy.random.rand(n)
b = numpy.log(a)
```

```
22.2 ms ± 904 µs per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

pandas



- labeled, indexed array data structures (e.g. Series, DataFrame)
- operations (e.g. `join`, `groupby`, ...)
- time series support (e.g. selection by date range)
- input/output tools (e.g. CSV, Excel, ...)
- some statistics

pandas example

task: find the correlation between inhabitants and number of museums of the departements of France

In [8]:
```
ls heresthedata/
```

```
Departements.csv            Liste_musees_de_France.xls
```

In [9]:
```
import pandas
```

In [10]:
```
departements = pandas.read_csv("heresthedata/Departements.csv", sep=";")
departements.head()
```

Out[10]:

	Nom du département	Nombre d'arrondissements	Nombre de cantons	Nombre de communes	Population municipale	Populat to
0	Ain	4	23.0	410	626.127	643.30
1	Aisne	5	21.0	805	539.783	554.04
2	Allier	3	19.0	318	343.062	353.26
3	Alpes-de-Haute-Provence	4	15.0	199	161.588	166.29
4	Hautes-Alpes	2	15.0	168	139.883	145.21

In [11]:
```
departements = departements[["Nom du département", "Population totale"]]
```

In [12]:
```
museums = pandas.read_excel("heresthedata/Liste_musees_de_France.xls")
museums.head(2)
```

Out[12]:

	NOMREG	NOMDEP	DATEAPPELLATION	FERME	ANNREOUV	ANNEXI
0	ALSACE	BAS-RHIN	01/02/2003	NON	NaN	NaN
1	ALSACE	BAS-RHIN	01/02/2003	NON	NaN	NaN

In [13]:
```
museum_count = museums.groupby("NOMDEP").size()
museum_count.head(5)
```

Out[13]:
```
NOMDEP
AIN                          14
AISNE                        15
ALLIER                        9
ALPES DE HAUTE PROVENCE       9
ALPES-MARITIMES              33
dtype: int64
```

In [14]:
```
departements["Nom du département"] = departements["Nom du département"].apply(lamb
da s: s.upper())
departements["Nom du département"] = departements["Nom du département"].apply(lamb
da s: s.replace("-", " "))
```

In [15]:
```
departements.index = departements["Nom du département"]
departements.drop(["Nom du département"], axis=1, inplace=True)
```

In [16]:
```
joined = departements.join(pandas.DataFrame(museum_count,
index=museum_count.index, columns=["number of museums"]))
joined.head(3)
```

Out[16]:

	Population totale	number of museums
Nom du département		
AIN	643.309	14.0
AISNE	554.040	15.0
ALLIER	353.262	9.0

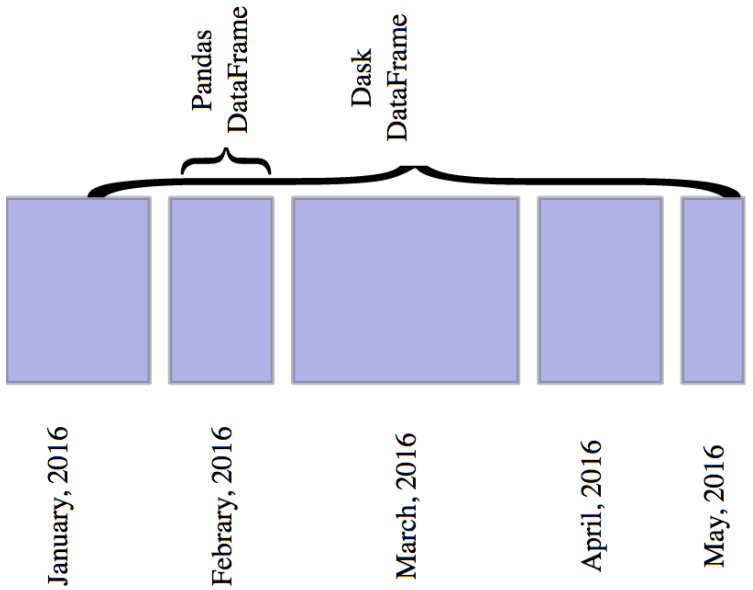
In [17]:
```
joined["Population totale"] = joined["Population totale"].apply(lambda s: pandas.to_numeric(s, errors="drop"))
joined.corr()
```

Out[17]:

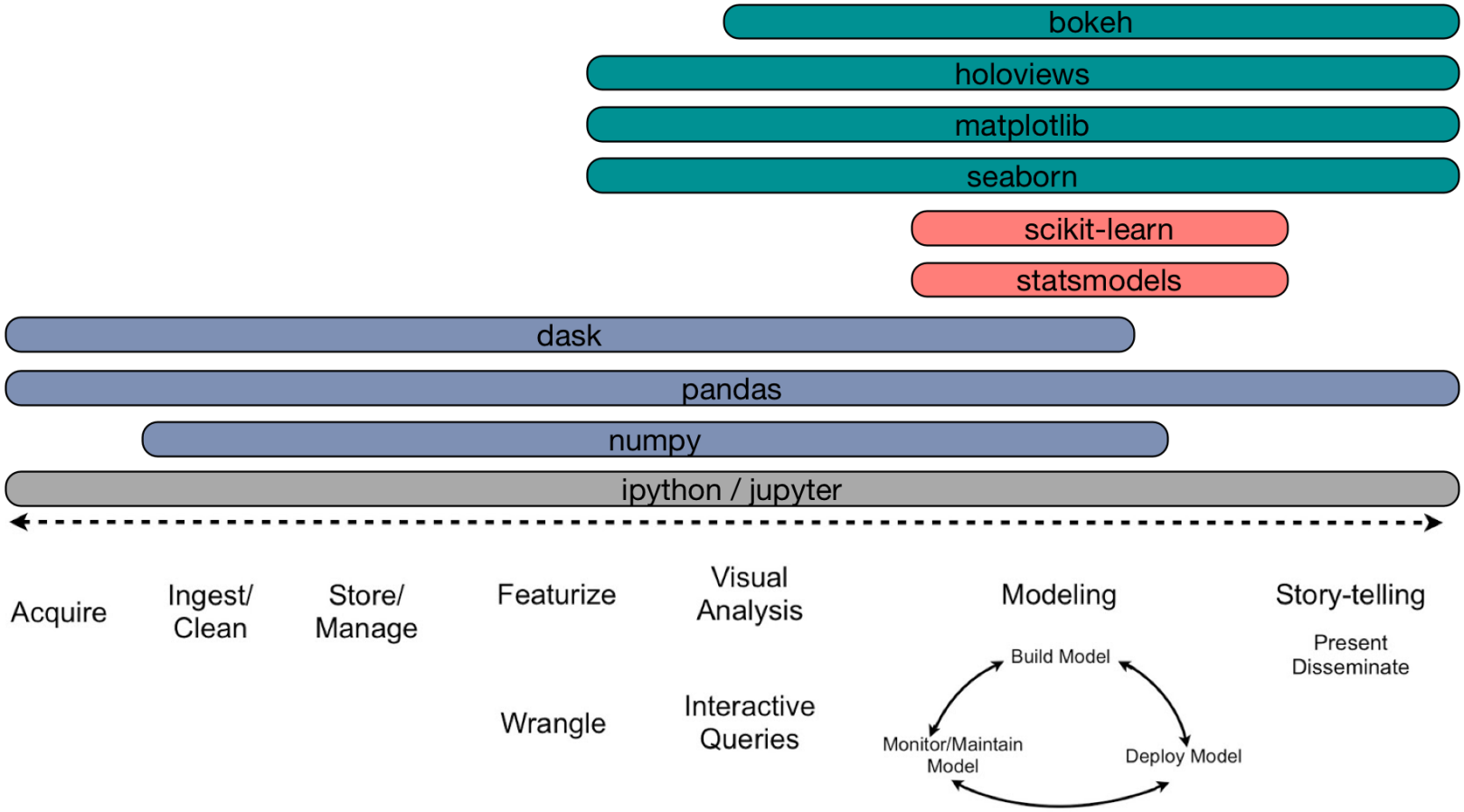
	Population totale	number of museums
Population totale	1.000000	0.601027
number of museums	0.601027	1.000000

dask

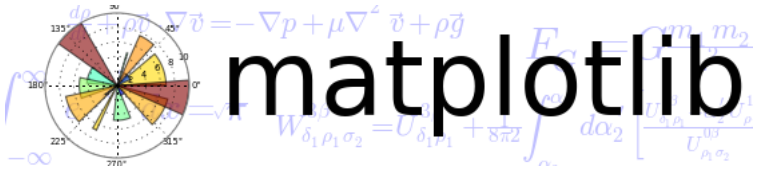
- **dask dataframe**
  - combines many pandas dataframes (split along the index), **mimic pandas API**
  - use cases
    - manipulating **datasets not fitting comfortably into memory** on a single machine
    - **parallelizing** many pandas operations across **many cores**
    - **distributed computing** of very large tables (e.g. stored in parallel file systems)



Visual Exploration & Presentation



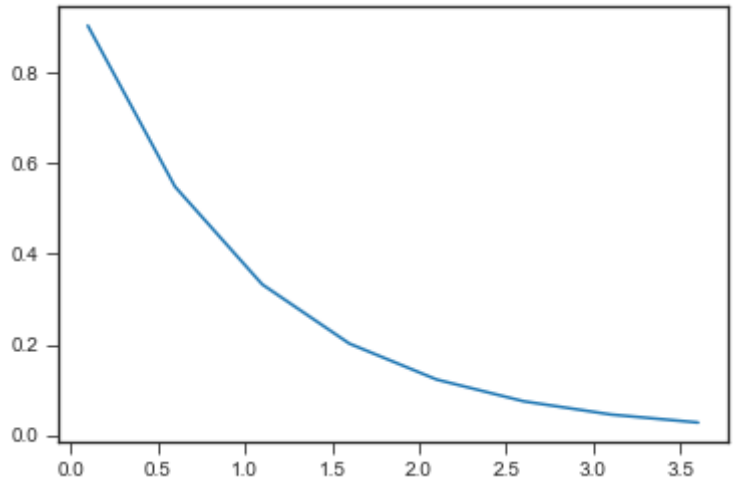
matplotlib



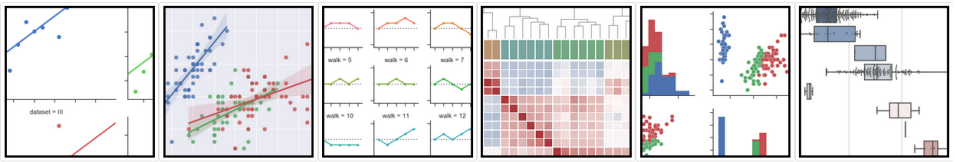
- 2D plotting library
- provides
  - MATLAB-like interface via the `pyplot` API

In [18]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

In [19]:
```
x = numpy.arange(0.1, 4, 0.5)
y = numpy.exp(-x)

fig, ax = plt.subplots()
ax.plot(x, y)
plt.show()
```



seaborn



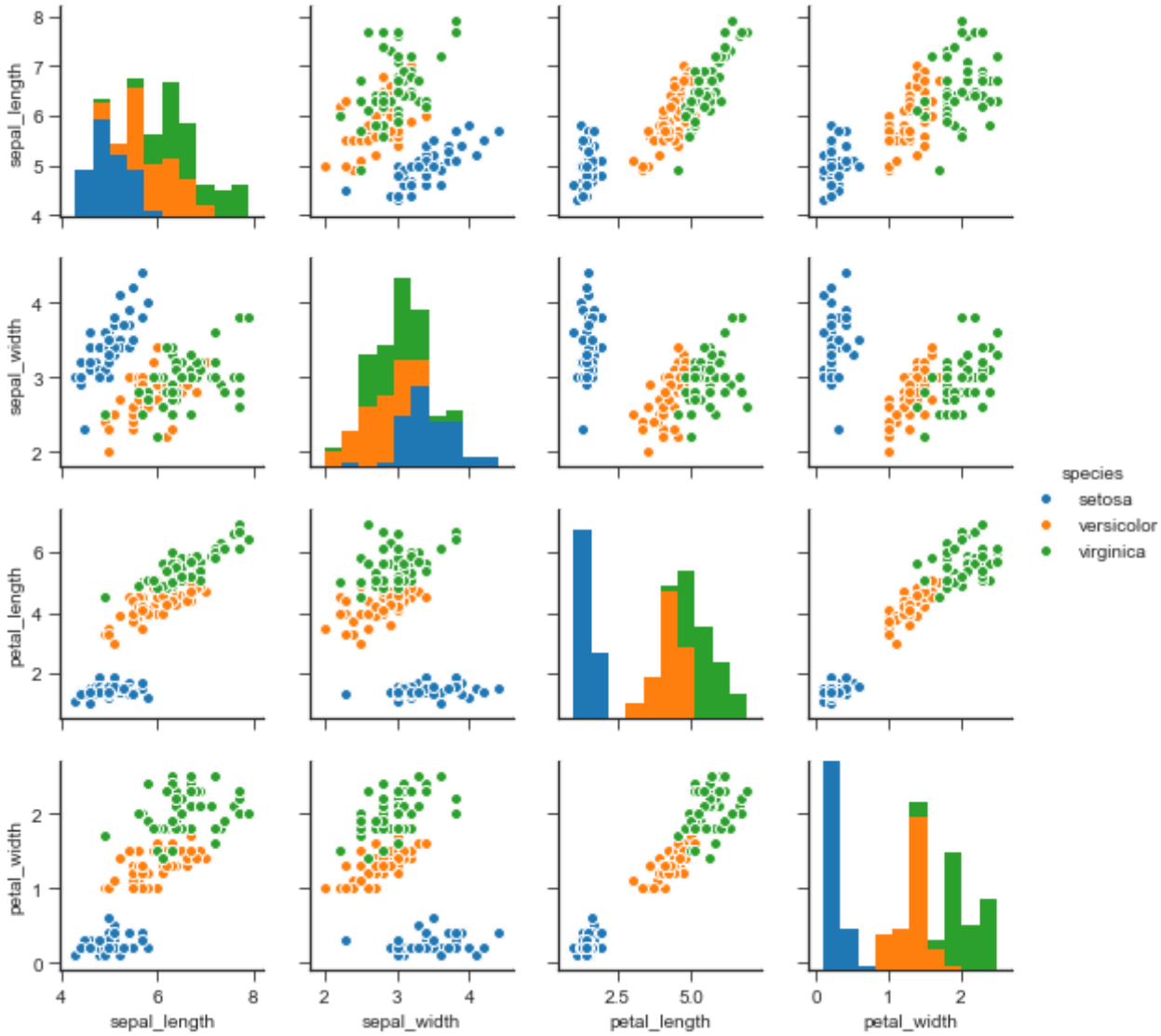
- production-ready **statistical graphics** on top of matplotlib
  - fit and visualize linear regression models
  - visualize and cluster matrix data
  - plot time series data
  - structuring grids of plots
- support for pandas and numpy data structures
- improved styling of matplotlib graphics (themes, color palettes, ...)

In [20]:
```
irisData = seaborn.load_dataset("iris")
irisData.head()
```

Out[20]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [21]:
```
seaborn.pairplot(irisData, hue="species", size=2);
```



bokeh

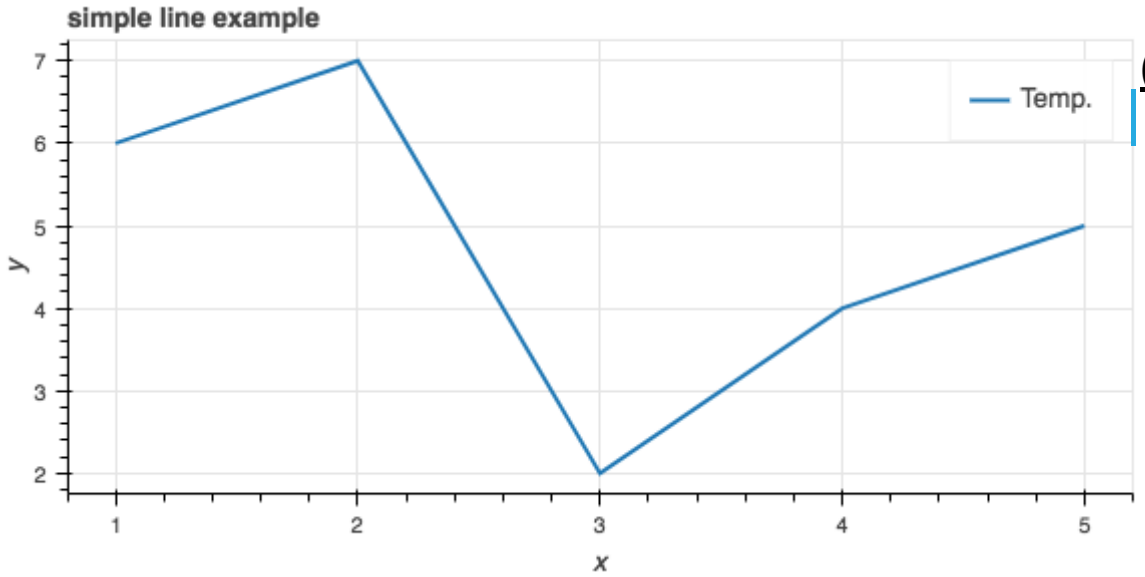
- interactive visualization library that targets modern web browsers for presentation
  - build e.g. interactive dashboards, data applications, ...
  - inspired by D3.js

In [22]:

```
from bokeh import plotting

x = [1, 2, 3, 4, 5]
y = [6, 7, 2, 4, 5]

plotting.output_notebook()
plot = plotting.figure(title="simple line example", x_axis_label='x',
y_axis_label='y', width=600, height=300)
plot.line(x, y, legend="Temp.", line_width=2)
plotting.show(plot)
```

(https://bokeh.pydata.org)

Bokeh JS 0.12.9 successfully loaded.



(https://bokeh.pydata.o

holoviews

- "focus on what you are trying to explore and convey, not on the process of plotting"
- annotate data with semantic metadata, then "let it plot itself"
- use **matplotlib** or **bokeh** as backend (and easily switch between them)

In [24]:
```
macro_df = pandas.read_csv('data/macro.csv', '\t')
macro_df.head()
```

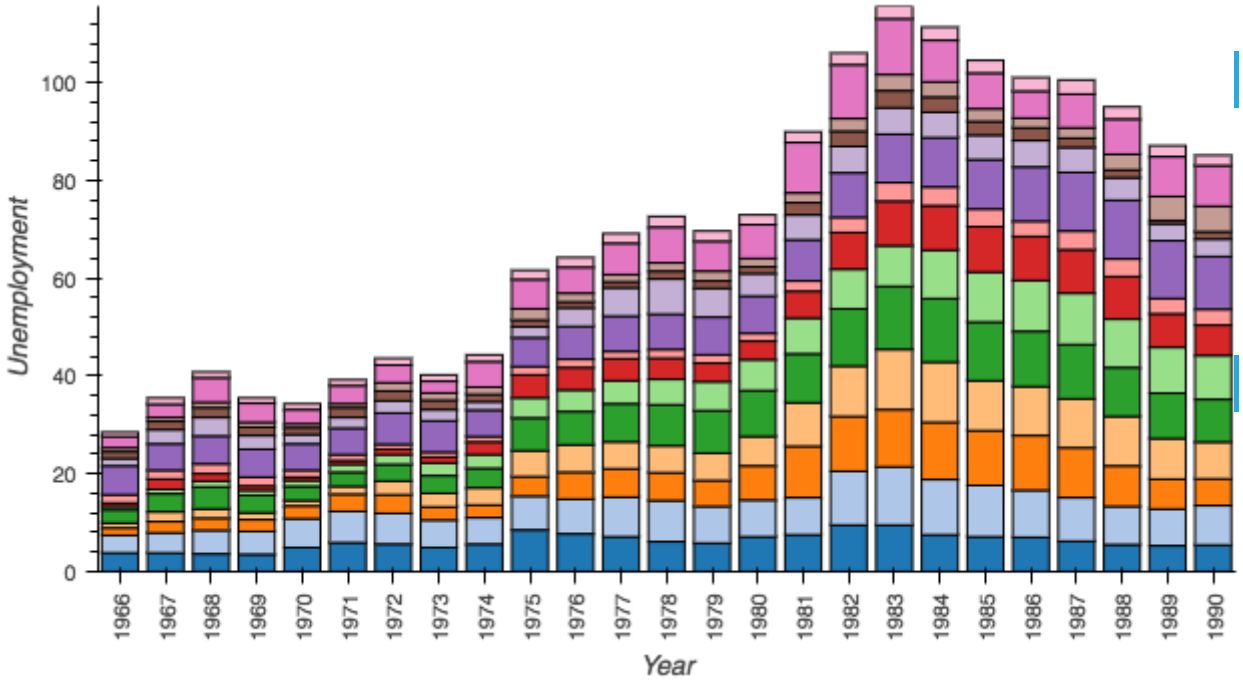
Out[24]:

	country	year	gdp	unem	capmob	trade
0	United States	1966	5.111141	3.8	0	9.622906
1	United States	1967	2.277283	3.8	0	9.983546
2	United States	1968	4.700000	3.6	0	10.089120
3	United States	1969	2.800000	3.5	0	10.435930
4	United States	1970	-0.200000	4.9	0	10.495350

In [25]:
```
import holoviews
holoviews.extension('bokeh')

key_dimensions   = [('year', 'Year'), ('country', 'Country')]
value_dimensions = [('unem', 'Unemployment'), ('capmob', 'Capital Mobility'),
                    ('gdp', 'GDP Growth'), ('trade', 'Trade')]
macro = holoviews.Table(macro_df, kdims=key_dimensions, vdims=value_dimensions)
```

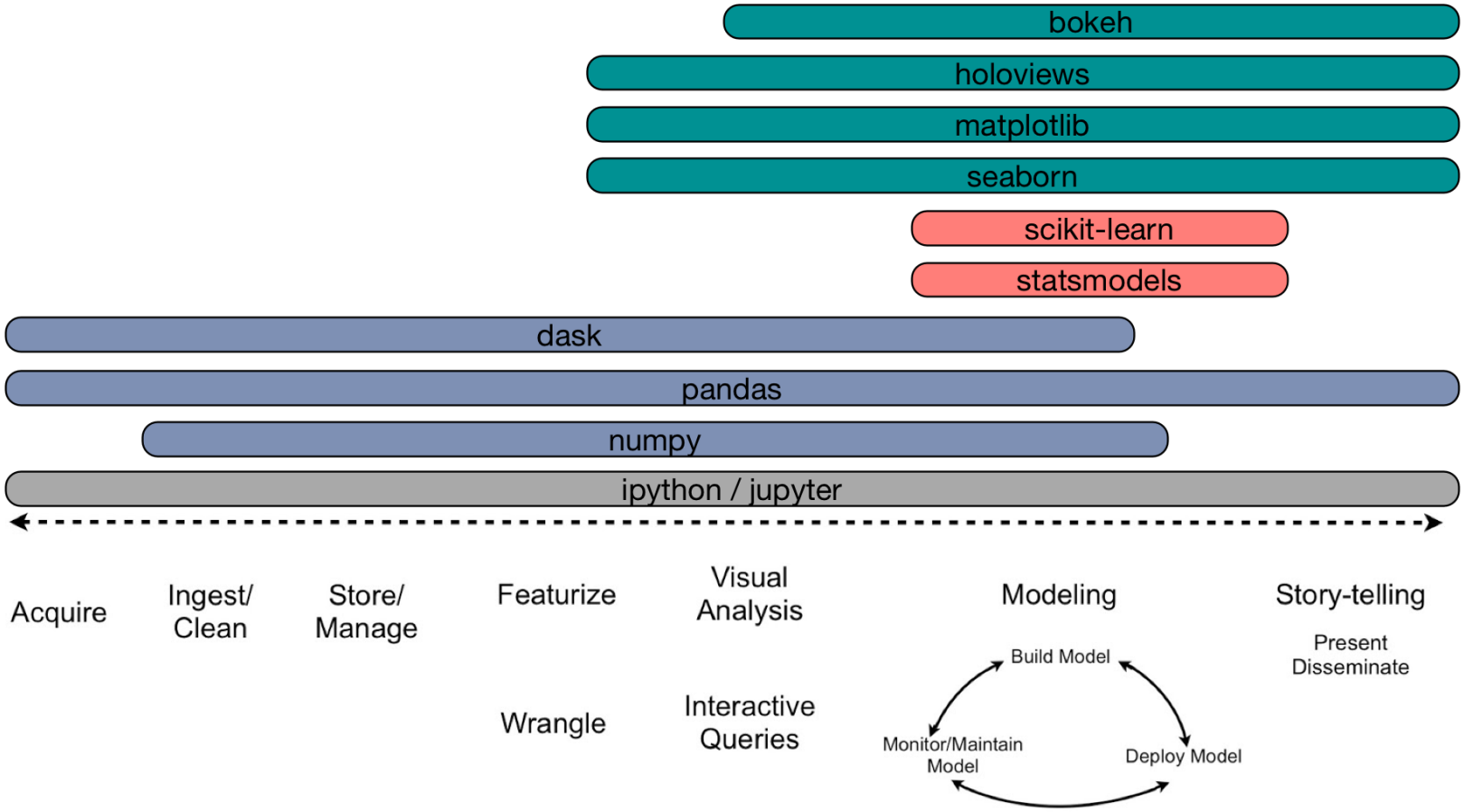
In [26]:
```
%%opts Bars [stack_index=1 xrotation=90 legend_cols=7 show_legend=False  show_fram
e=False tools=['hover']]
%%opts Bars (color=Cycle('Category20'))
%%opts Bars [width=650 height=350]

macro.to.bars([ 'Year', 'Country'], 'Unemployment', [])
```

Out[26]:



(https://bok

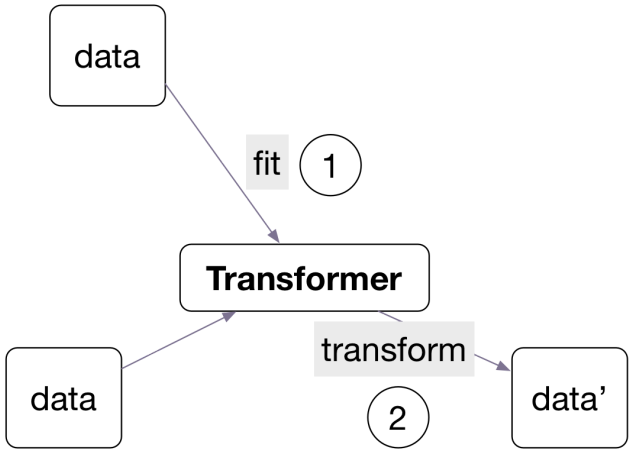
Modeling

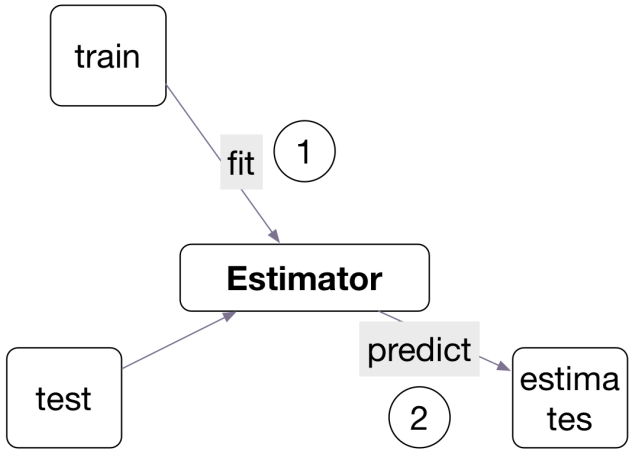

bokeh
holoviews
matplotlib
seaborn
scikit-learn
statsmodels
dask
pandas
numpy
ipython / jupyter
Acquire
Ingest/ Clean
Store/ Manage
Featurize
Wrangle
Visual Analysis
Interactive Queries
Modeling
Build Model
Deploy Model
Monitor/Maintain Model
Story-telling
Present
Disseminate

scikit-learn

- machine learning in Python
- provides
  - machine learning algorithms for classification, regression, clustering, dimensionality reduction, ...
  - building blocks of preprocessing and model selection workflows

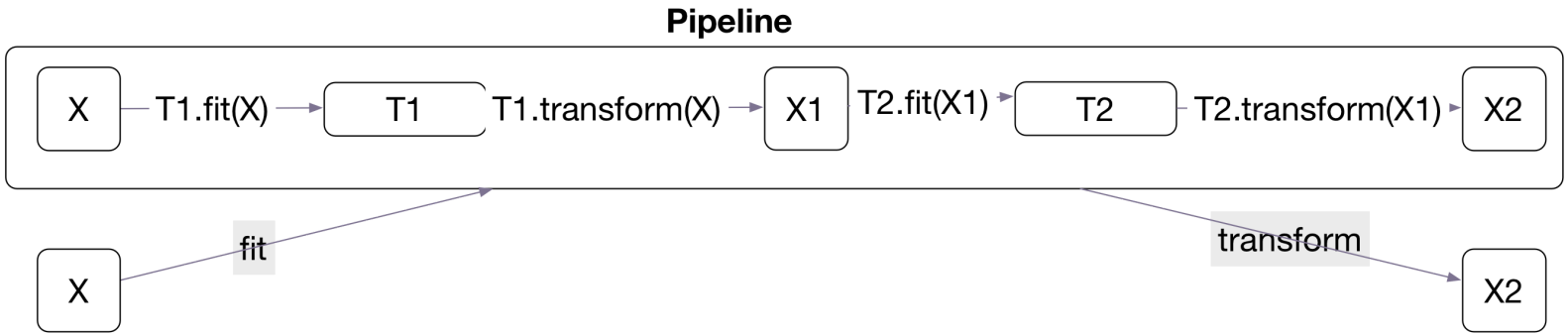
scikit-learn's modular approach: estimators, transformers, pipelines






```
transformer.fit(X)
X_t = transformer.transform(X)
```

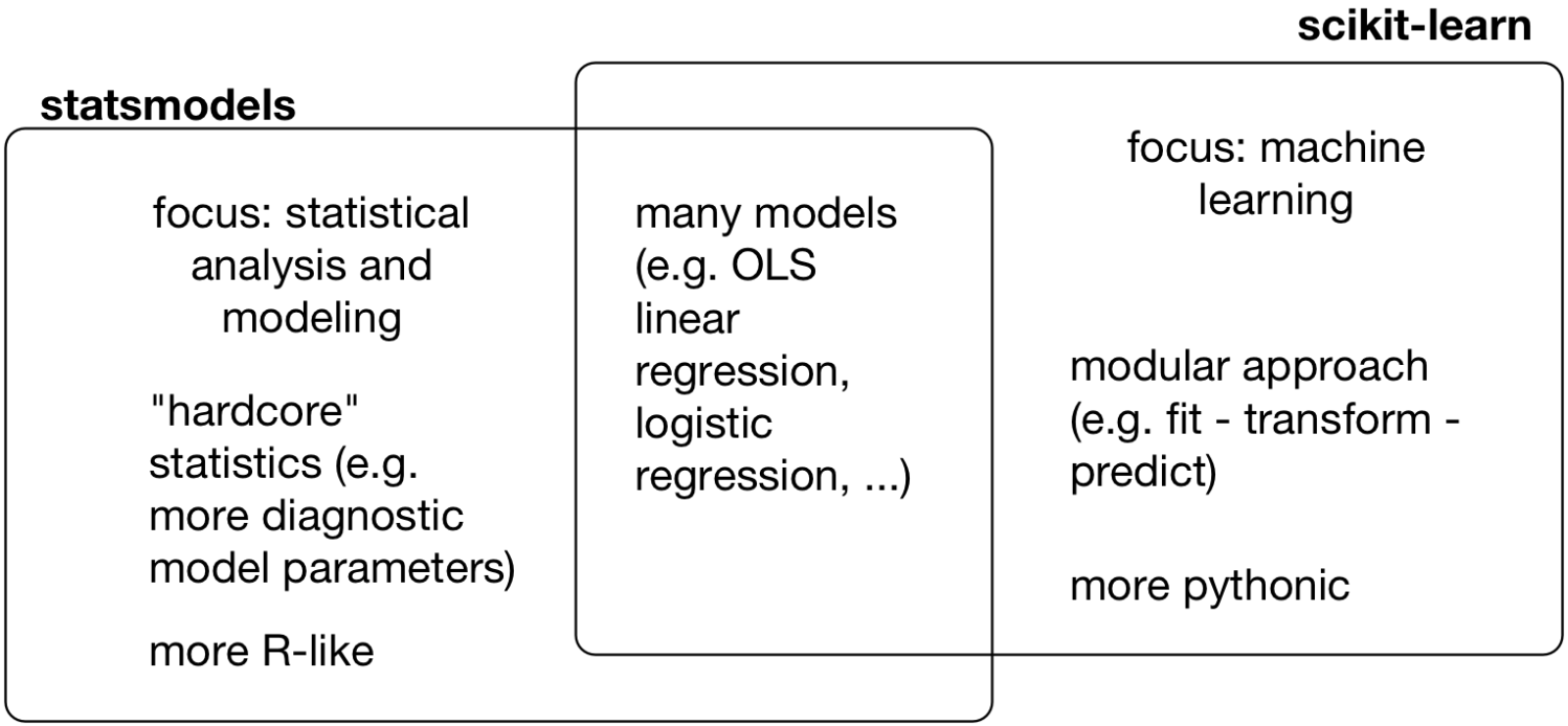
```
estimator.fit(X_train)
p = estimator.predict(X_test)
```



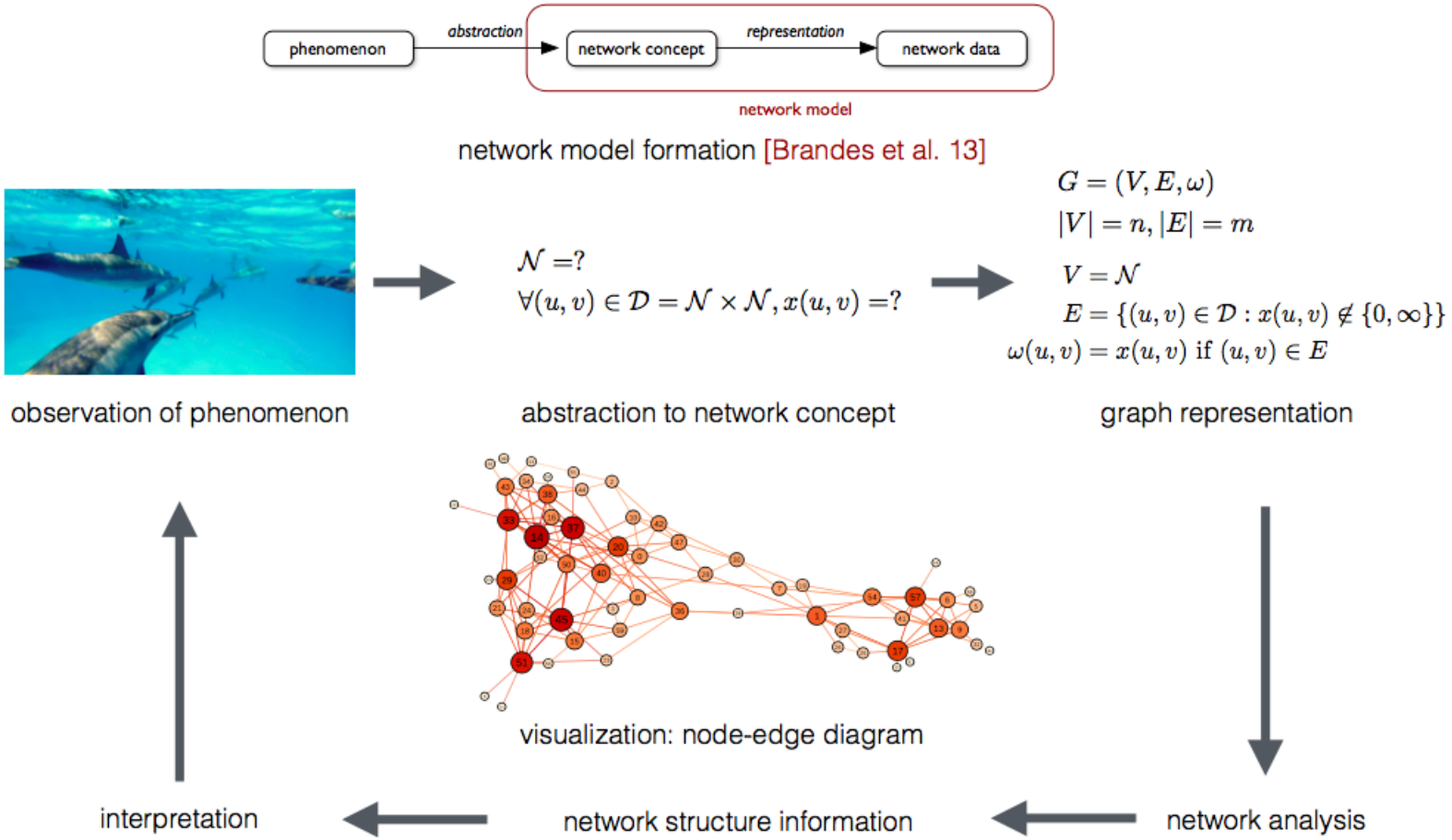
statsmodels

- statistical models, test, and exploration
  - R replacement
- contents, e.g.
  - regression models
  - time series analysis (e.g. ARIMA)
  - statistical test (e.g. t-test)

statsmodels vs scikit-learn



and now for something completely different: network analysis

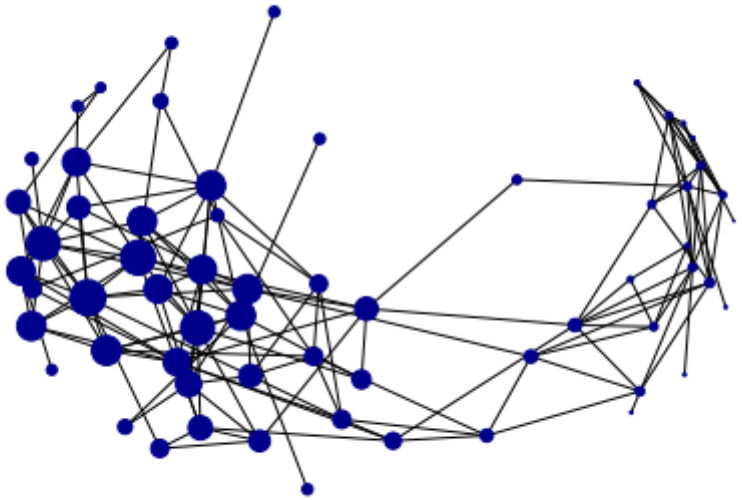


networkx

- creation, manipulation, and study of the structure of complex networks
- provides
  - algorithms for network analysis (centrality, diameter, ...)
  - algorithms for constructing graphs
  - ...
- pure Python
  - any object can be a node

In [27]:

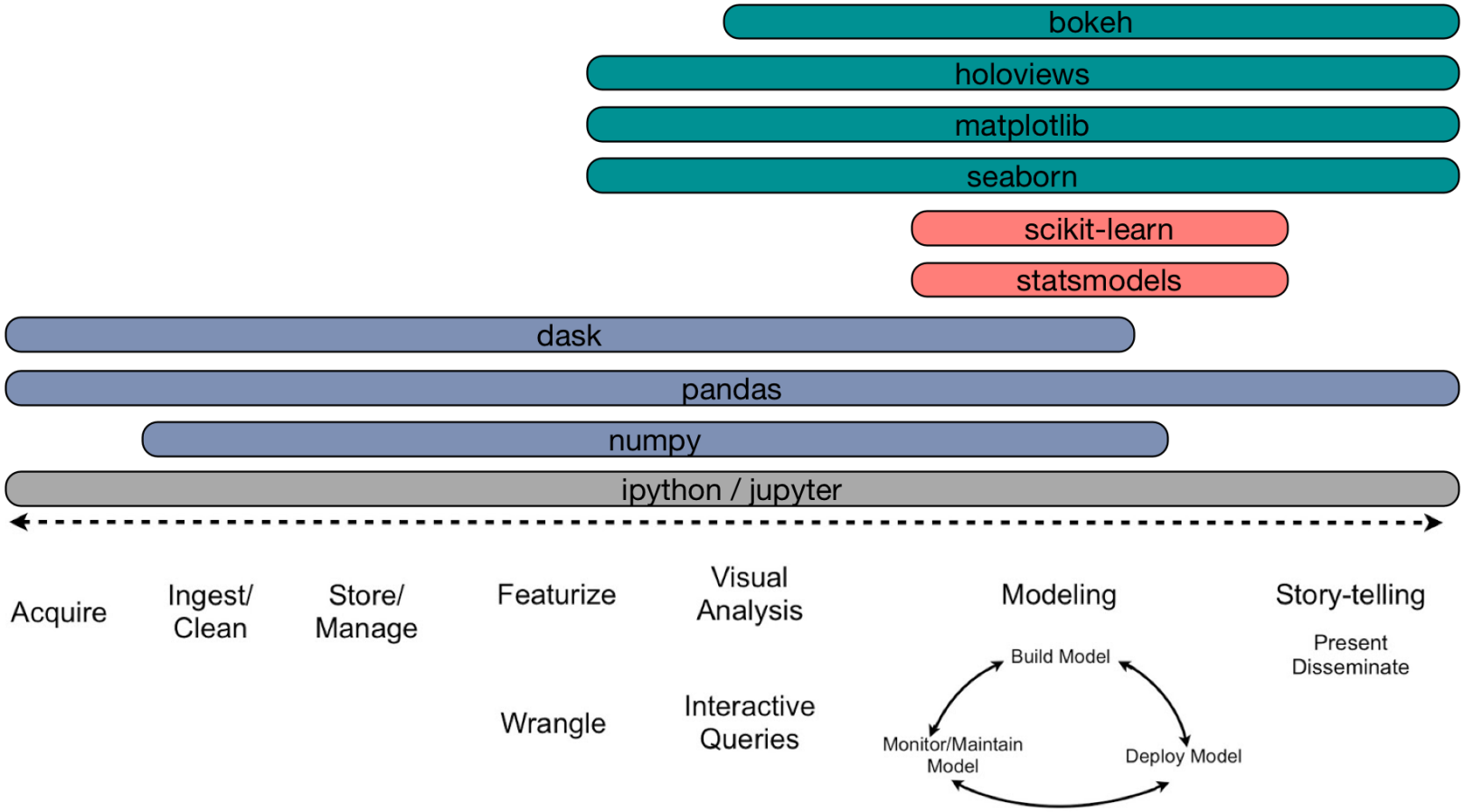
```
import networkx

G = networkx.read_edgelist("data/dolphins.edgelist")
ec = networkx.eigenvector_centrality(G)
networkx.draw(G,
              node_size=numpy.fromiter(iter(ec.values()), dtype=float) * 1000,
              node_color='darkblue',  pos=networkx.spring_layout(G))
```



need to do graph data analysis at scale?

- again, put algorithms and data structures into compiled code with
  - **igraph**
  - **graph-tool**
  - **networkit**

meta-tools


bokeh
holoviews
matplotlib
seaborn
scikit-learn
statsmodels
dask
pandas
numpy
ipython / jupyter
Acquire
Ingest/ Clean
Store/ Manage
Featurize
Wrangle
Visual Analysis
Interactive Queries
Modeling
Build Model
Deploy Model
Monitor/Maintain Model
Story-telling
Present
Disseminate

ipython

IP[y]

- powerful interactive Python shell
- tools for parallel computing (ipyparallel)

ipython extension: rpy2.ipython (formerly known as rmagic)

- seamless conversion of R and pandas dataframes between cells

In [29]:
```
%load_ext rpy2.ipython
```

In [30]:
```
df = pandas.read_csv("data/iris.csv", sep=";")
```

In [31]:
```
%%R -i df
head(df)
```

```
  Unnamed..0 sepal_length sepal_width petal_length petal_width species
0          0          5.1         3.5          1.4         0.2  setosa
1          1          4.9         3.0          1.4         0.2  setosa
2          2          4.7         3.2          1.3         0.2  setosa
3          3          4.6         3.1          1.5         0.2  setosa
4          4          5.0         3.6          1.4         0.2  setosa
5          5          5.4         3.9          1.7         0.4  setosa
```

In [32]:
```
%%R -o df2
df2 <- read.csv(file="data/iris.csv", header=TRUE, sep=";")
```

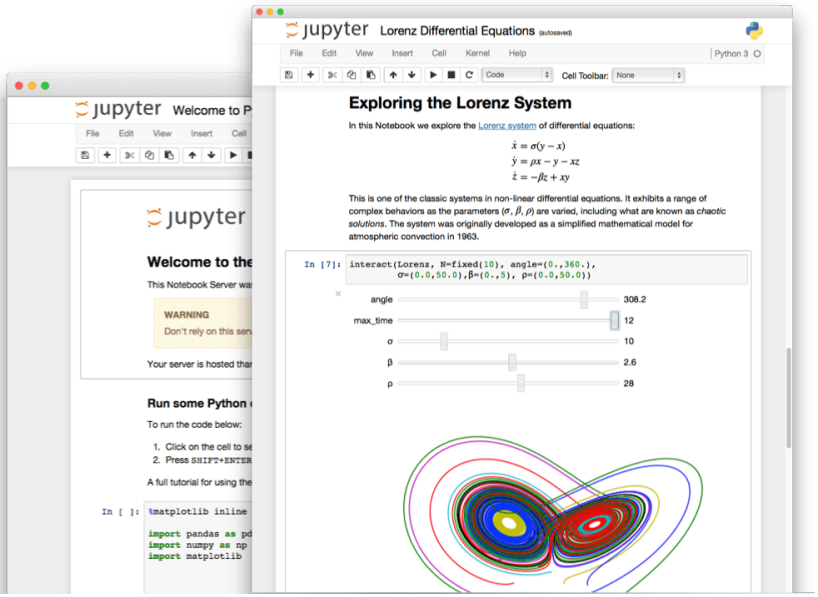
In [33]:
```
df2.describe()
```

Out[33]:

	X	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	74.500000	5.843333	3.057333	3.758000	1.199333
std	43.445368	0.828066	0.435866	1.765298	0.762238
min	0.000000	4.300000	2.000000	1.000000	0.100000
25%	37.250000	5.100000	2.800000	1.600000	0.300000
50%	74.500000	5.800000	3.000000	4.350000	1.300000
75%	111.750000	6.400000	3.300000	5.100000	1.800000
max	149.000000	7.900000	4.400000	6.900000	2.500000

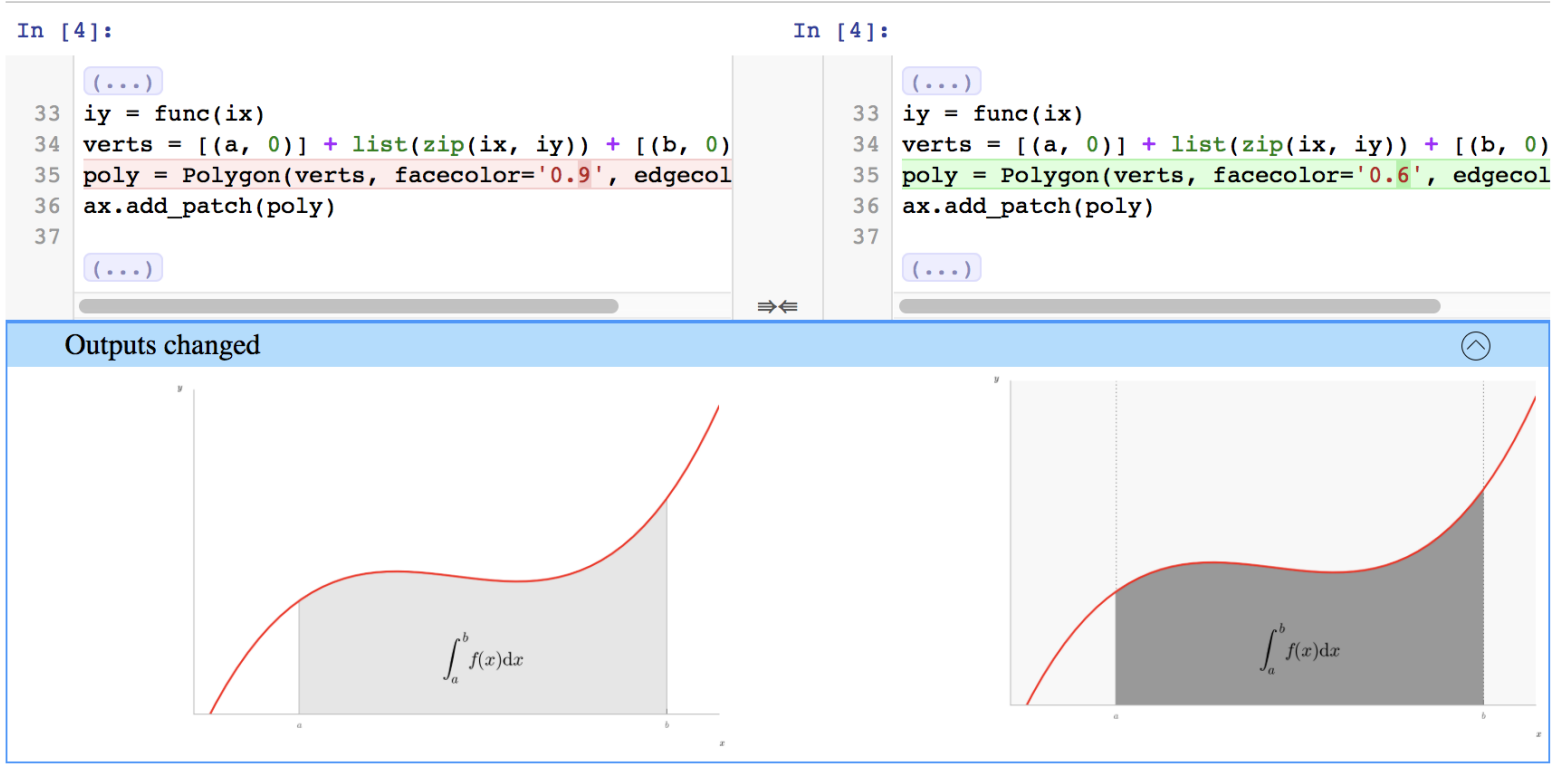
jupyter

- interactive notebooks, combining code, documentation, visualizations
  - Donald Knuth's literate programming (1992)
- language-agnostic, with support for e.g. R, Julia, Scala
- **nbconvert**: export notebooks to PDF, HTML, slides (e.g. this presentation)...



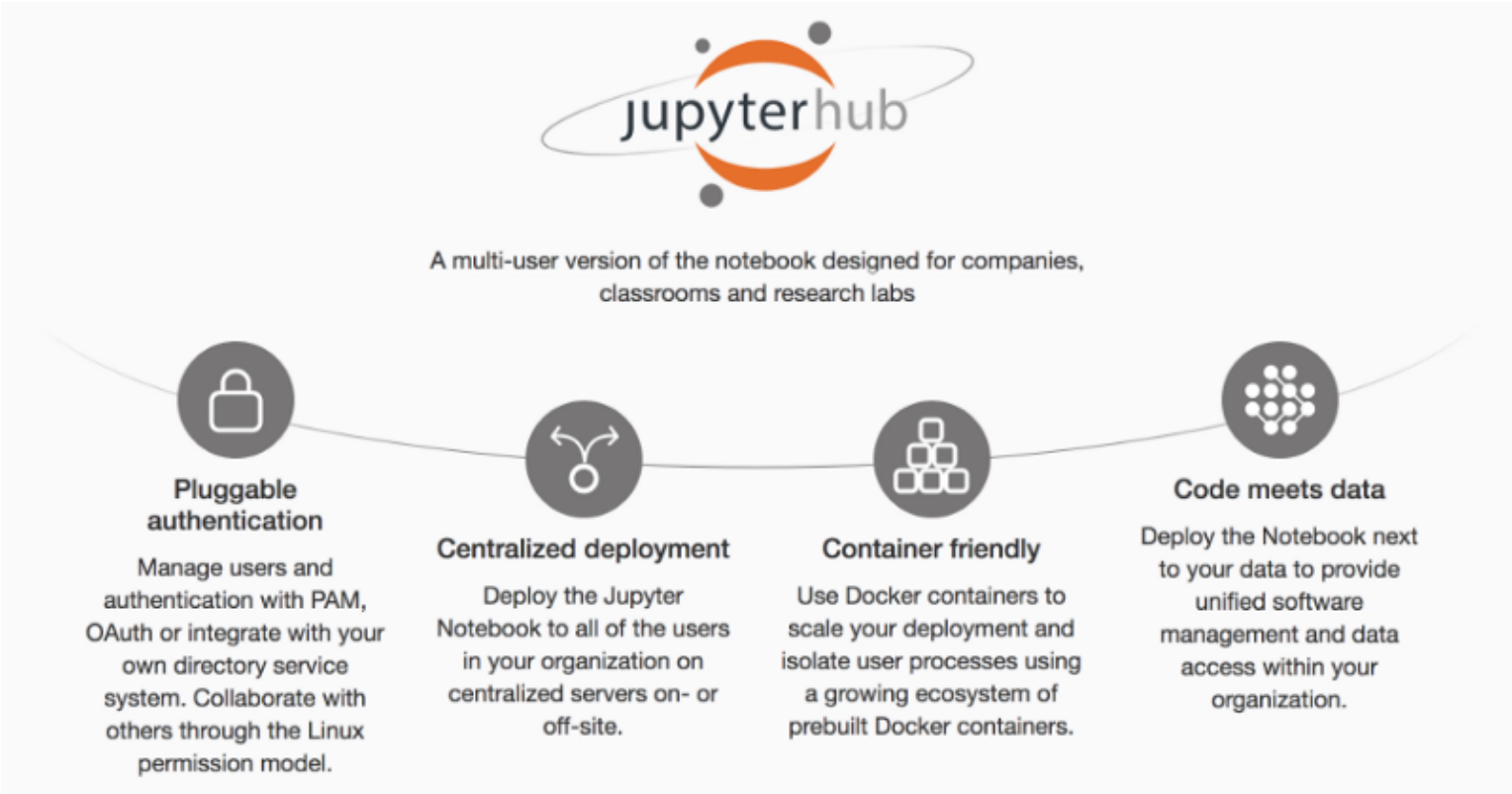
nbdime

- diff and merge tools for jupyter notebooks
- git integration!



jupyterhub

- multi-user server for jupyter notebooks



thank you for your attention!