Visualizing hierarchies

UNSUPERVISED LEARNING IN PYTHON

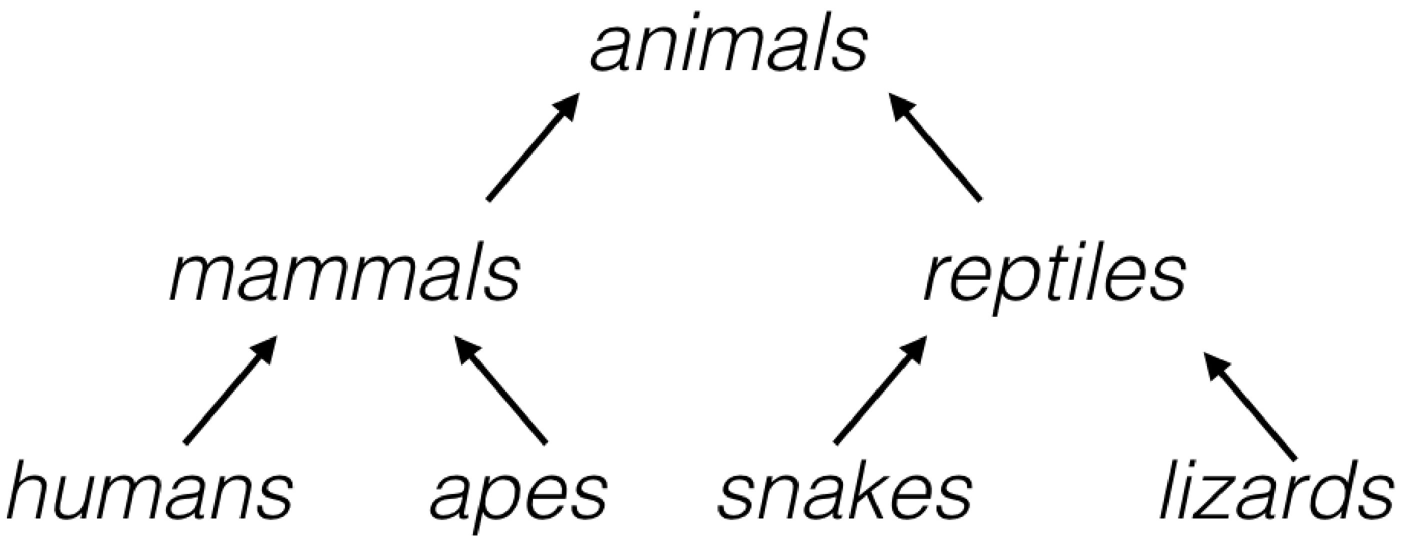
Benjamin Wilson
Director of Research at lateral.io

Visualizations communicate insight

- "t-SNE" : Creates a 2D map of a dataset (later)
- "Hierarchical clustering" (this video)

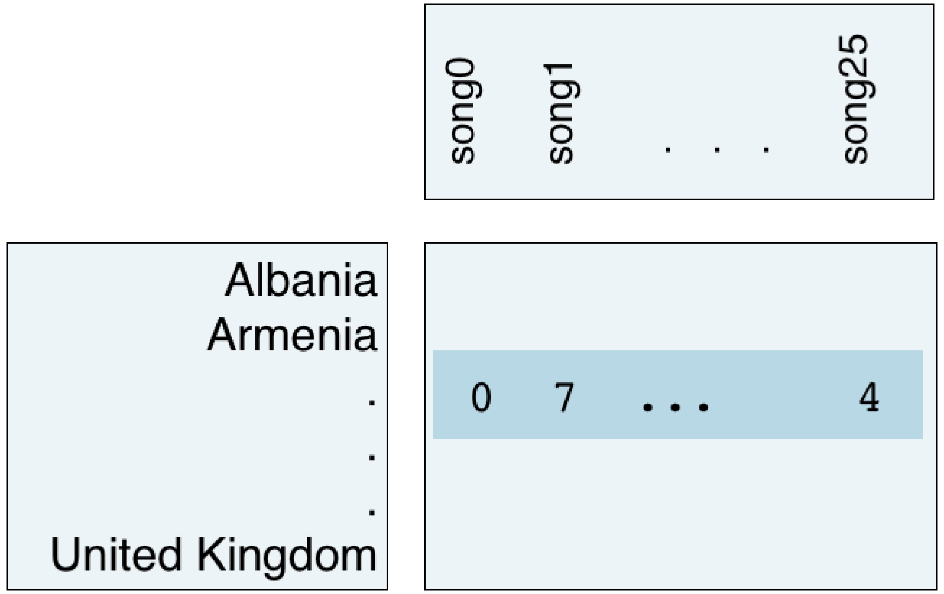
A hierarchy of groups

- Groups of living things can form a hierarchy
- Clusters are contained in one another



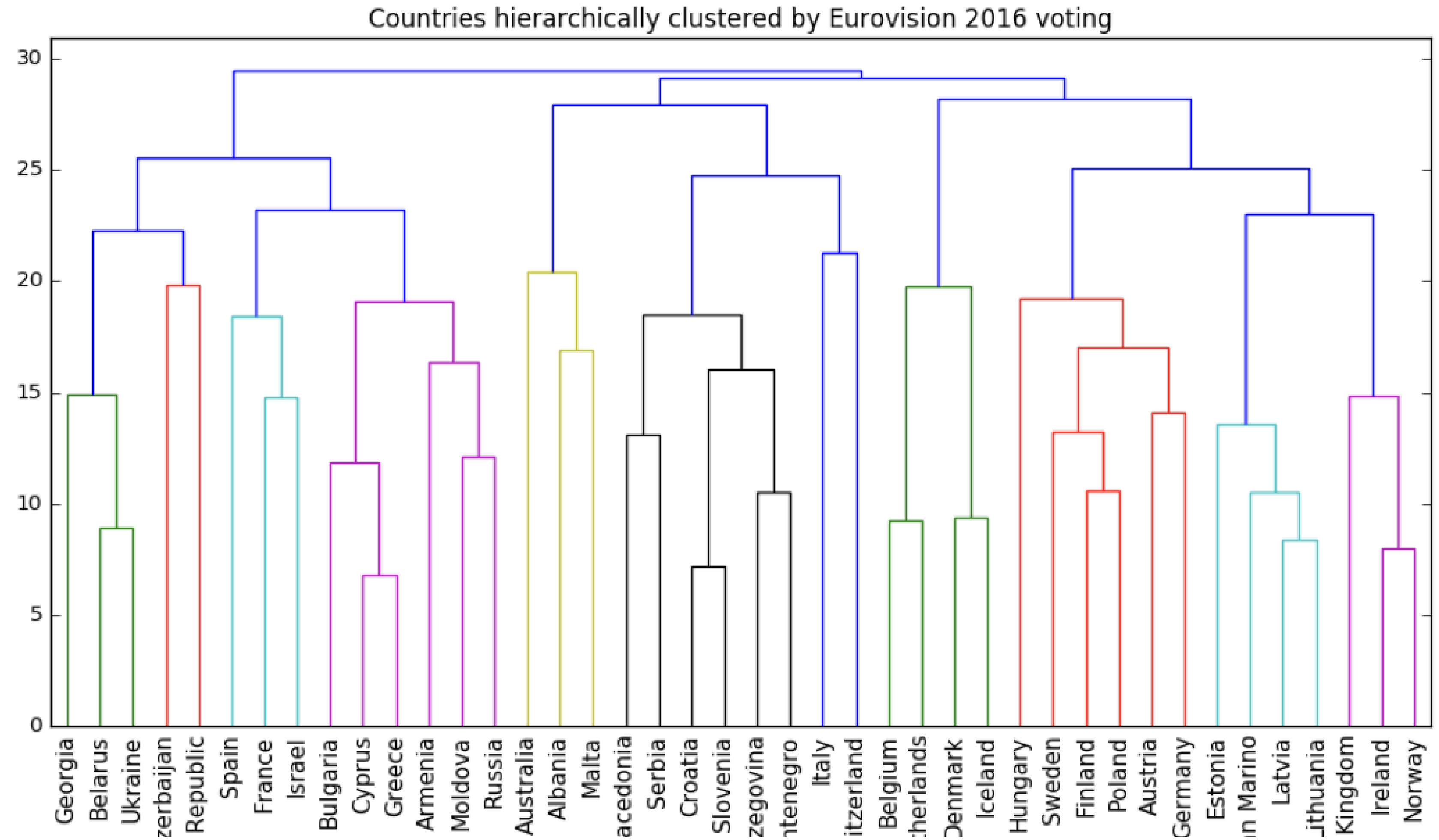
Eurovision scoring dataset

- Countries gave scores to songs performed at the Eurovision 2016
- 2D array of scores
- Rows are countries, columns are songs



[1] http://www.eurovision.tv/page/results

Hierarchical clustering of voting countries

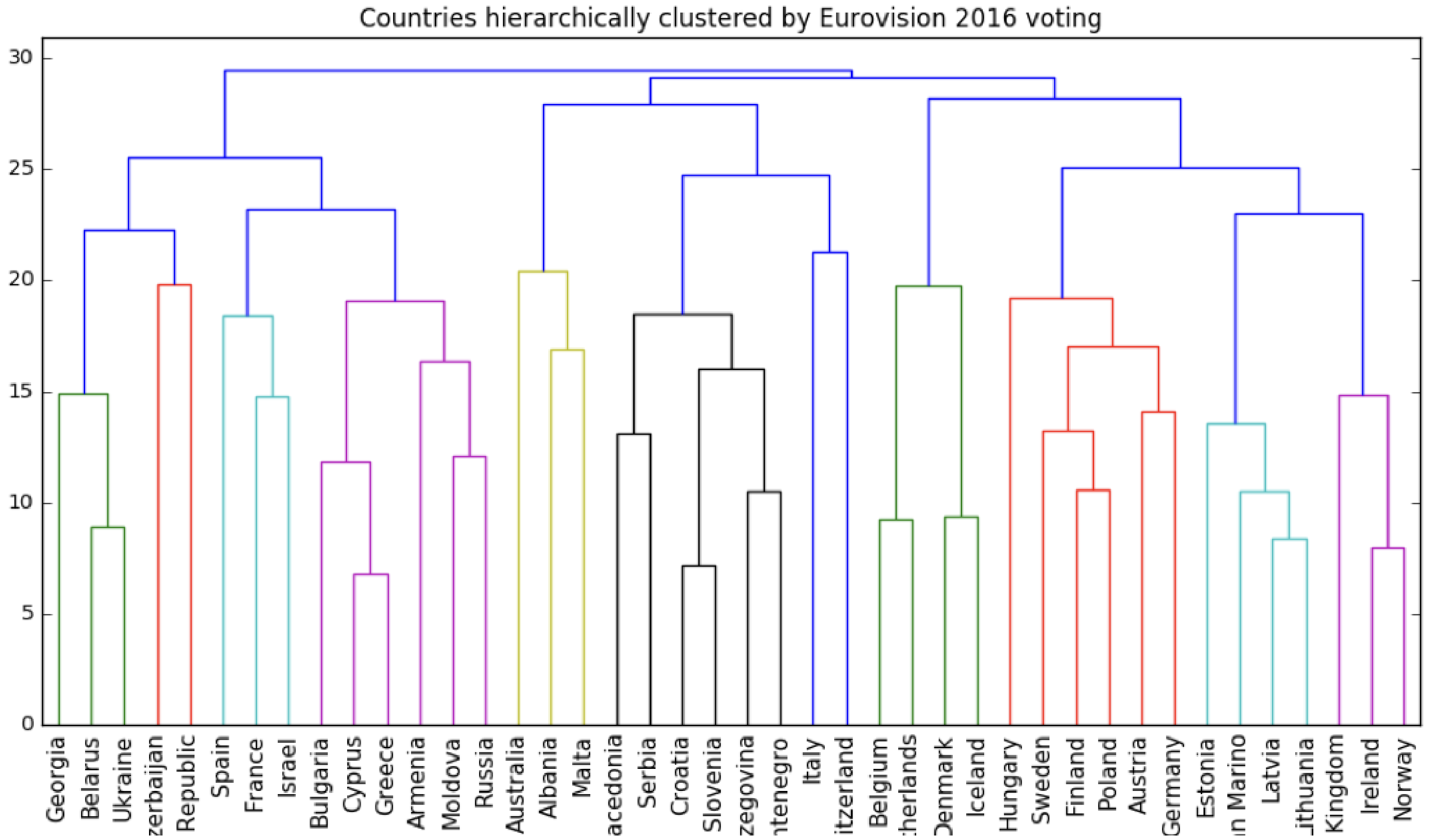


Hierarchical clustering

- Every country begins in a separate cluster
- At each step, the two closest clusters are merged
- Continue until all countries in a single cluster
- This is "agglomerative" hierarchical clustering

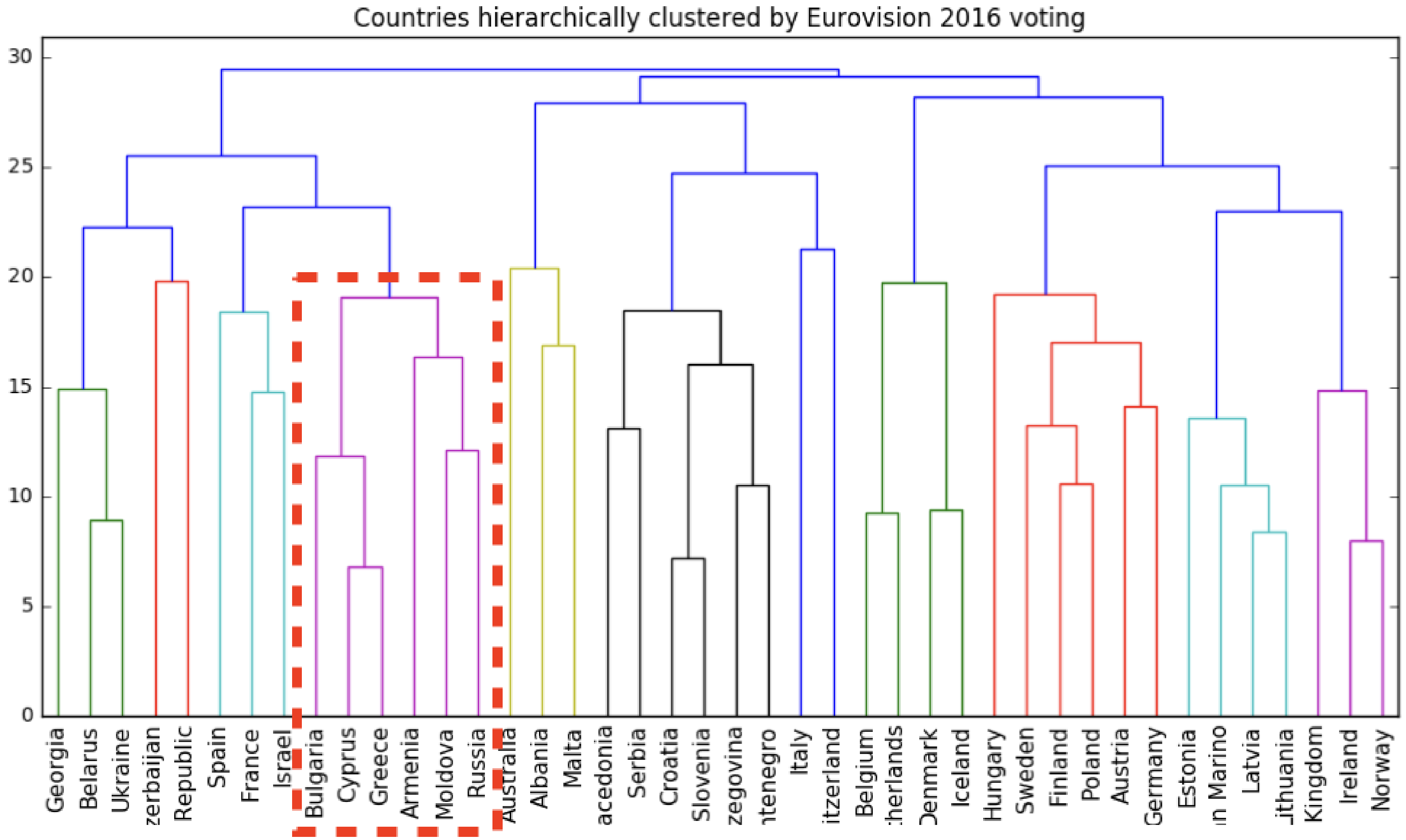
The dendrogram of a hierarchical clustering

- Read from the bottom up
- Vertical lines represent clusters

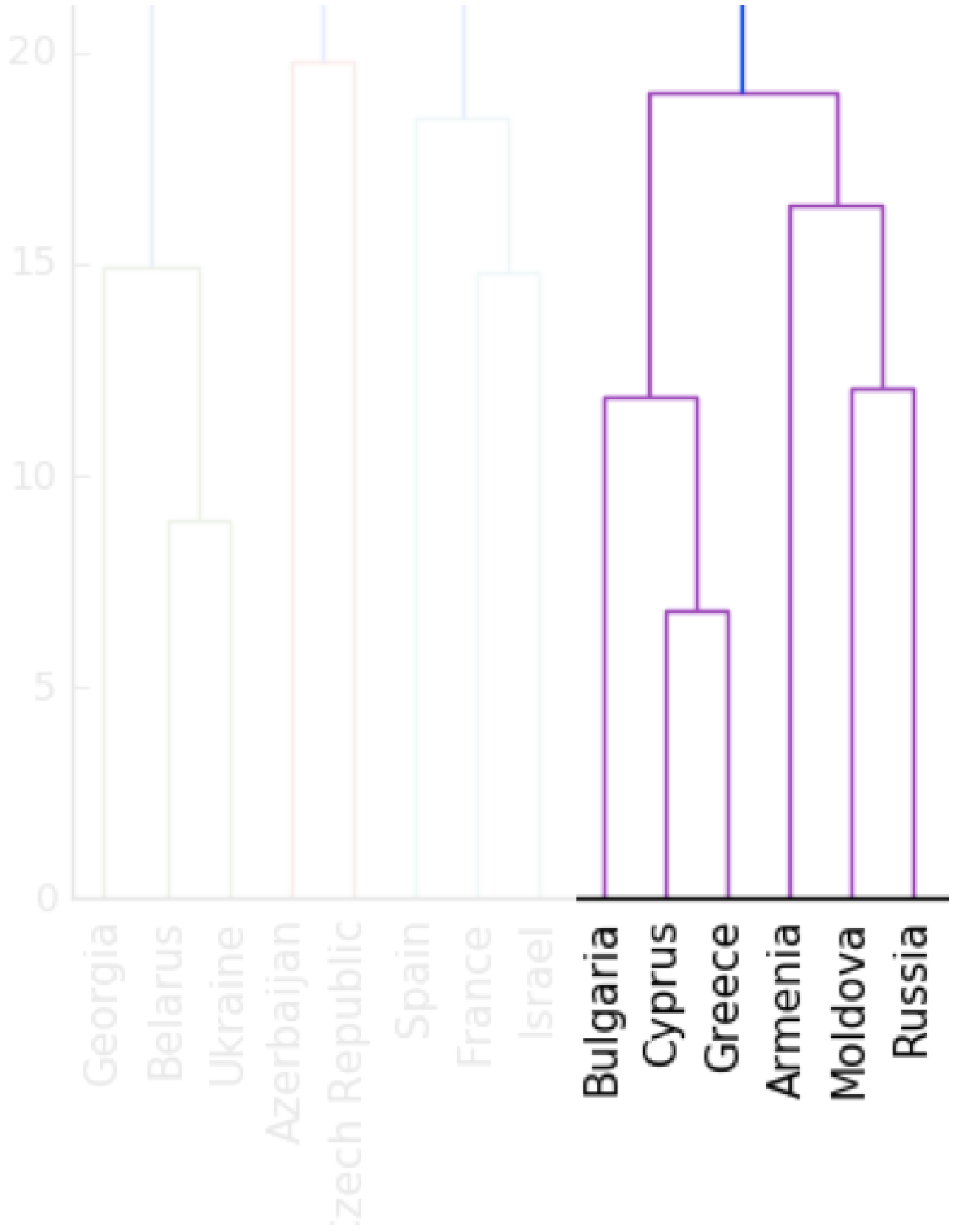


The dendrogram of a hierarchical clustering

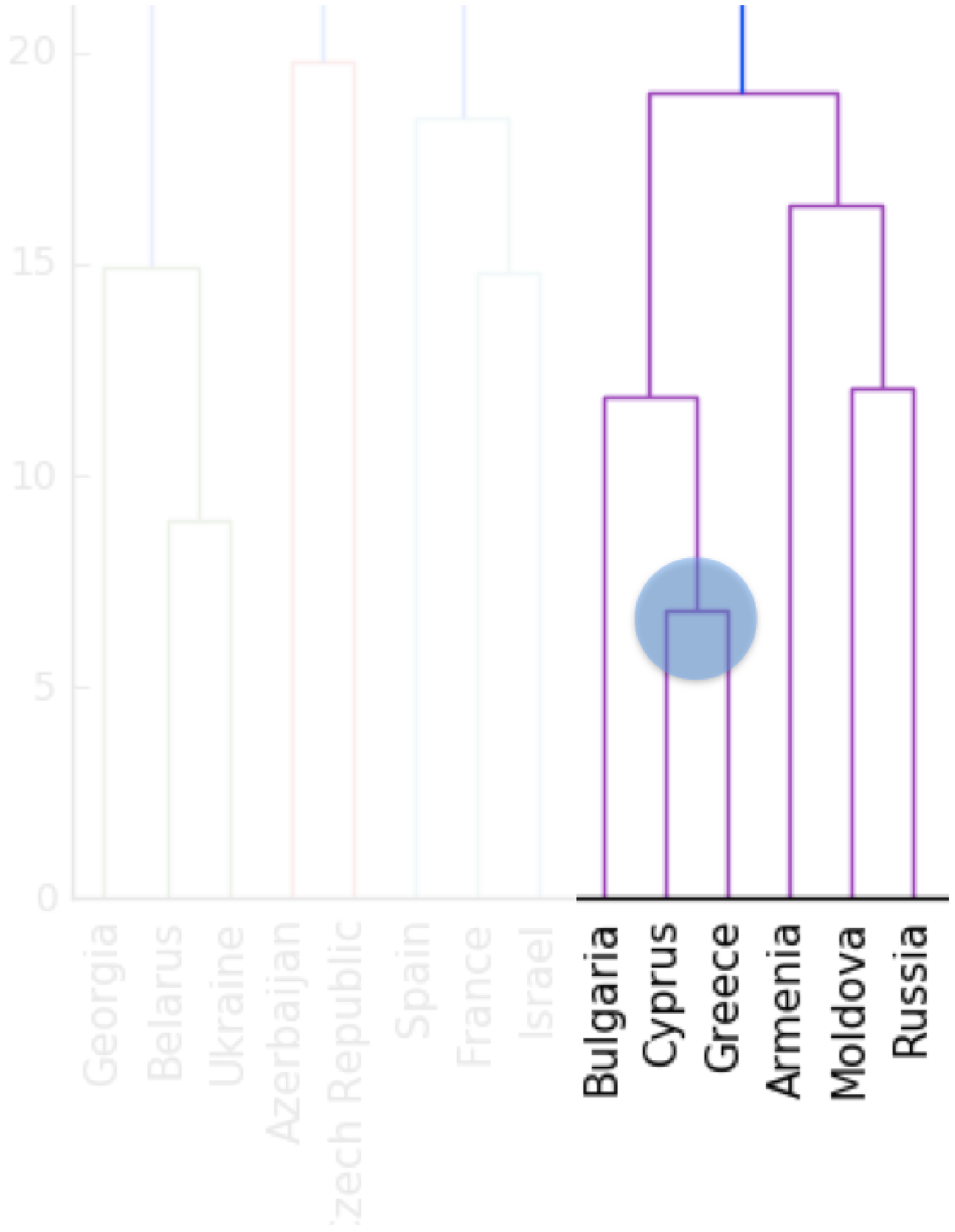
- Read from the bottom up
- Vertical lines represent clusters



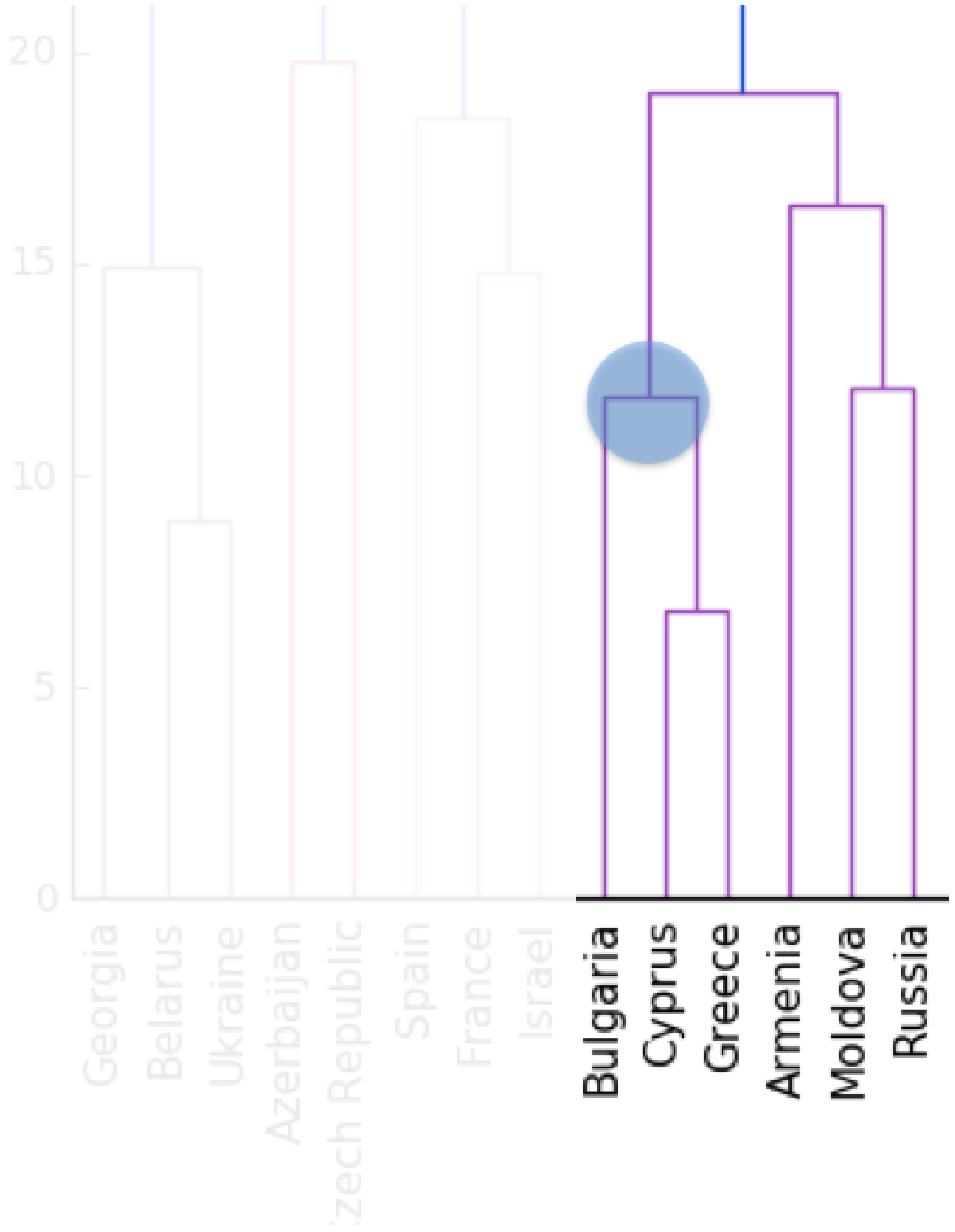
Dendrograms, step-by-step



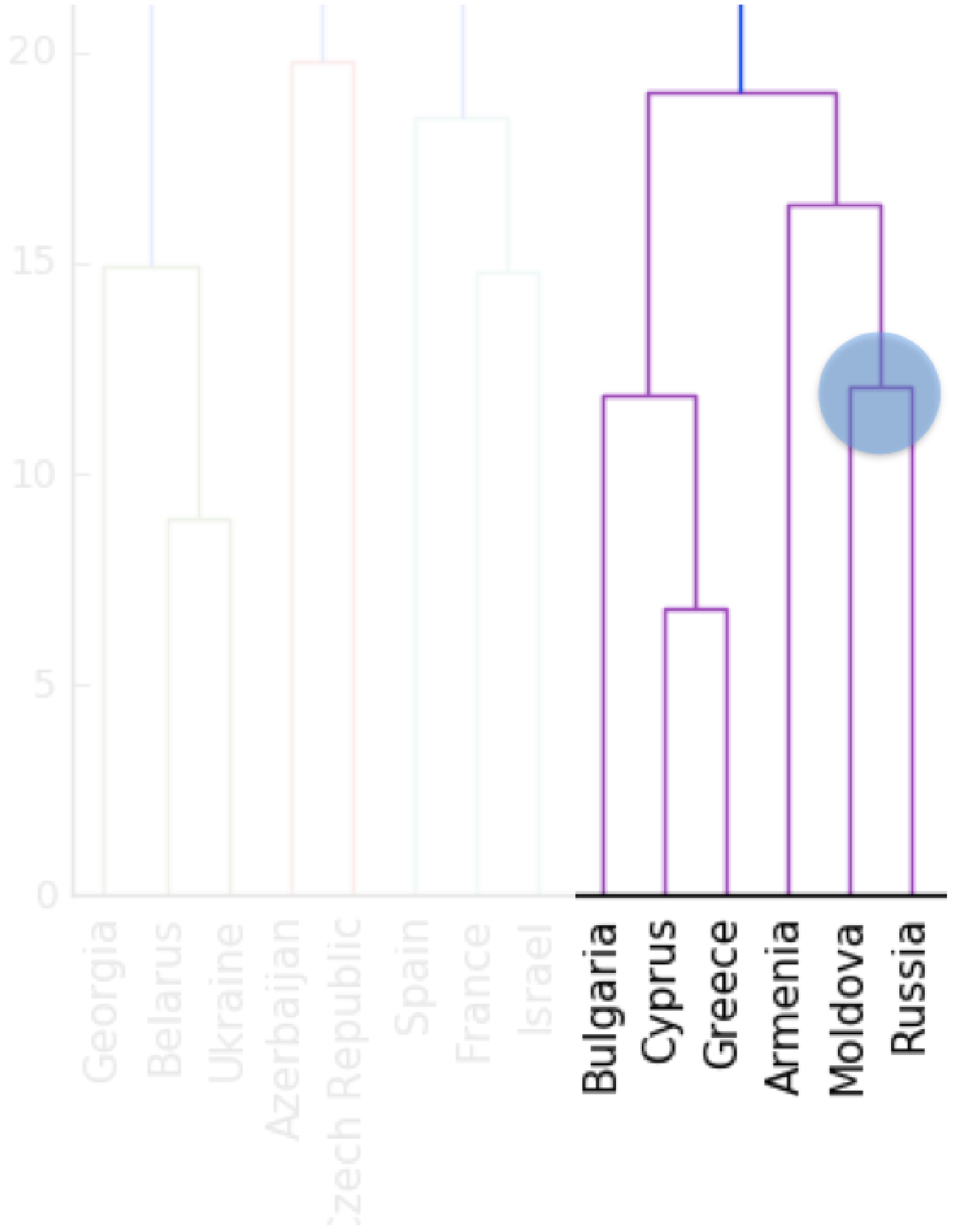
Dendrograms, step-by-step



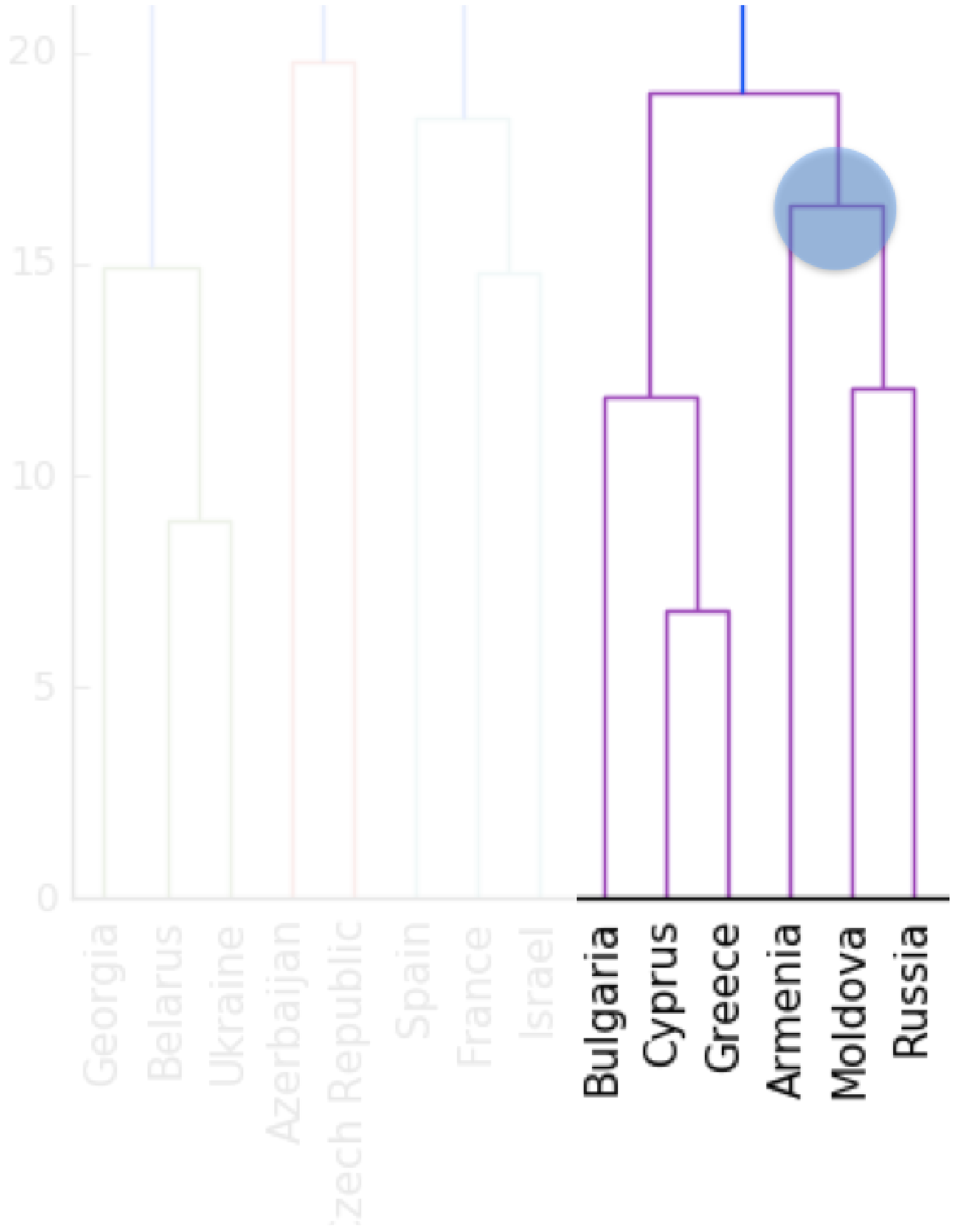
Dendrograms, step-by-step



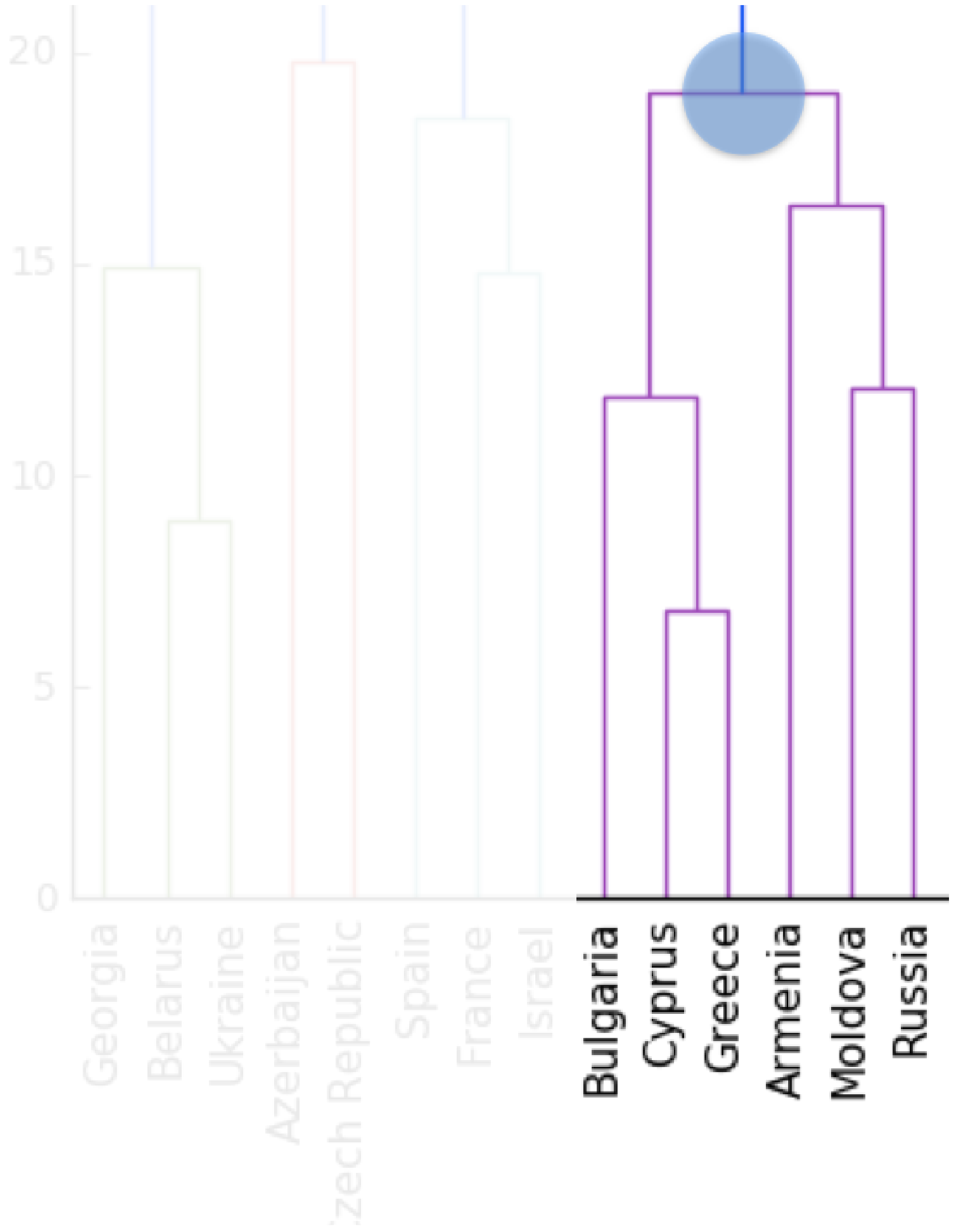
Dendrograms, step-by-step



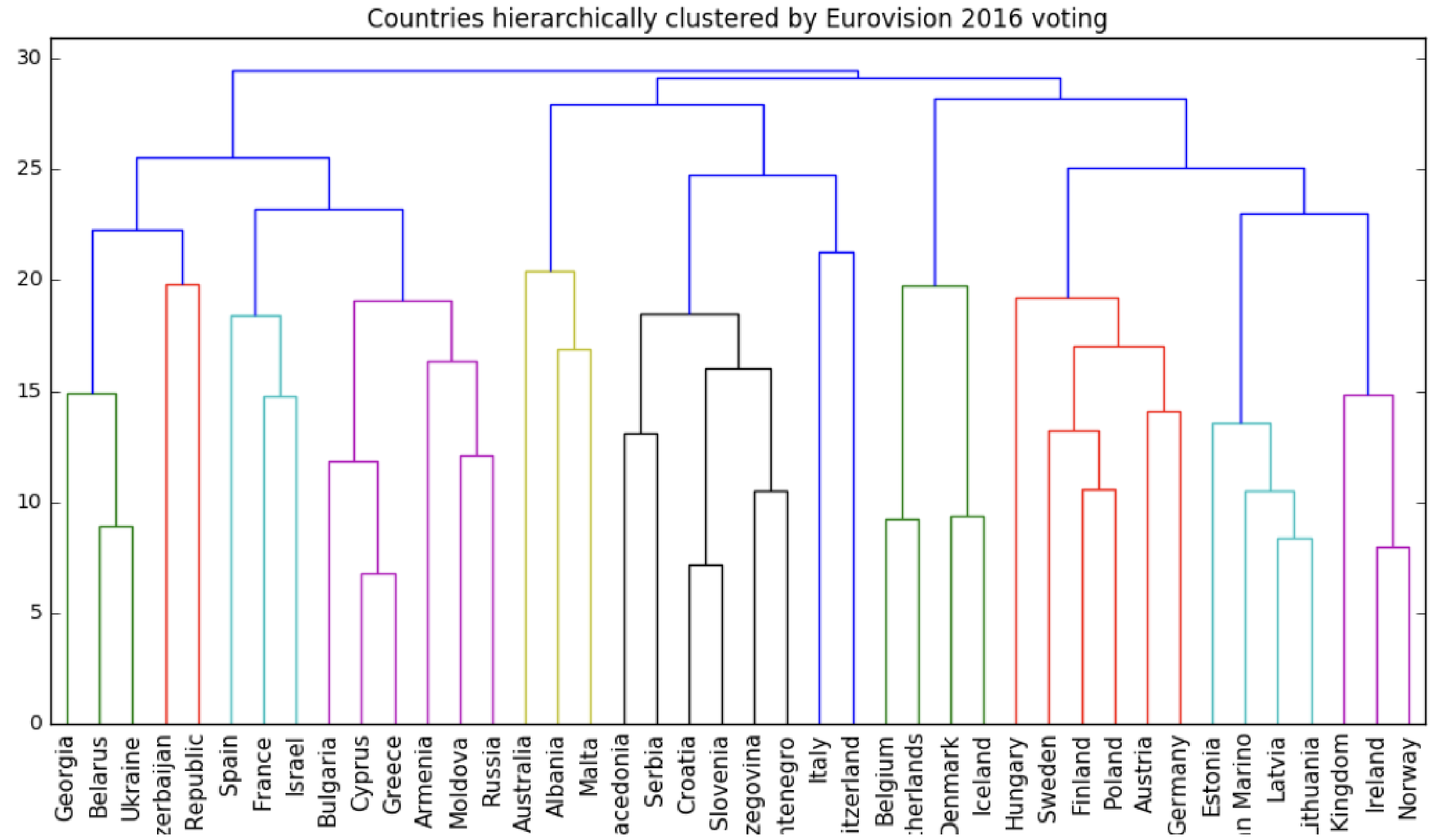
Dendrograms, step-by-step



Dendrograms, step-by-step



Dendrograms, step-by-step



Hierarchical clustering with SciPy

- Given `samples` (the array of scores), and `country_names`

```
import matplotlib.pyplot as plt
from scipy.cluster.hierarchy import linkage, dendrogram
mergings = linkage(samples, method='complete')
dendrogram(mergings,
           labels=country_names,
           leaf_rotation=90,
           leaf_font_size=6)
plt.show()
```

Let's practice!

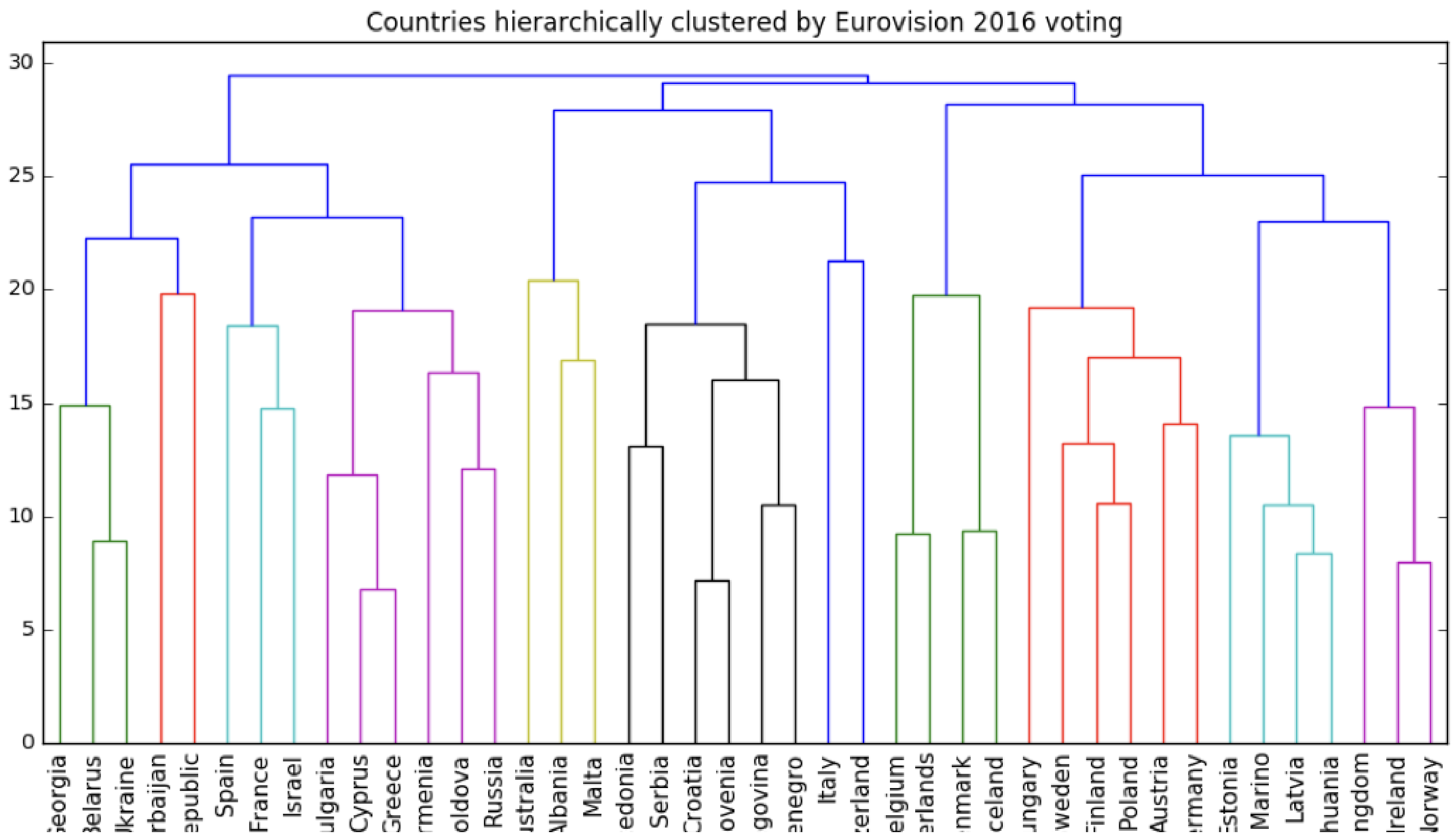
Cluster labels in hierarchical clustering

UNSUPERVISED LEARNING IN PYTHON

Benjamin Wilson
Director of Research at lateral.io

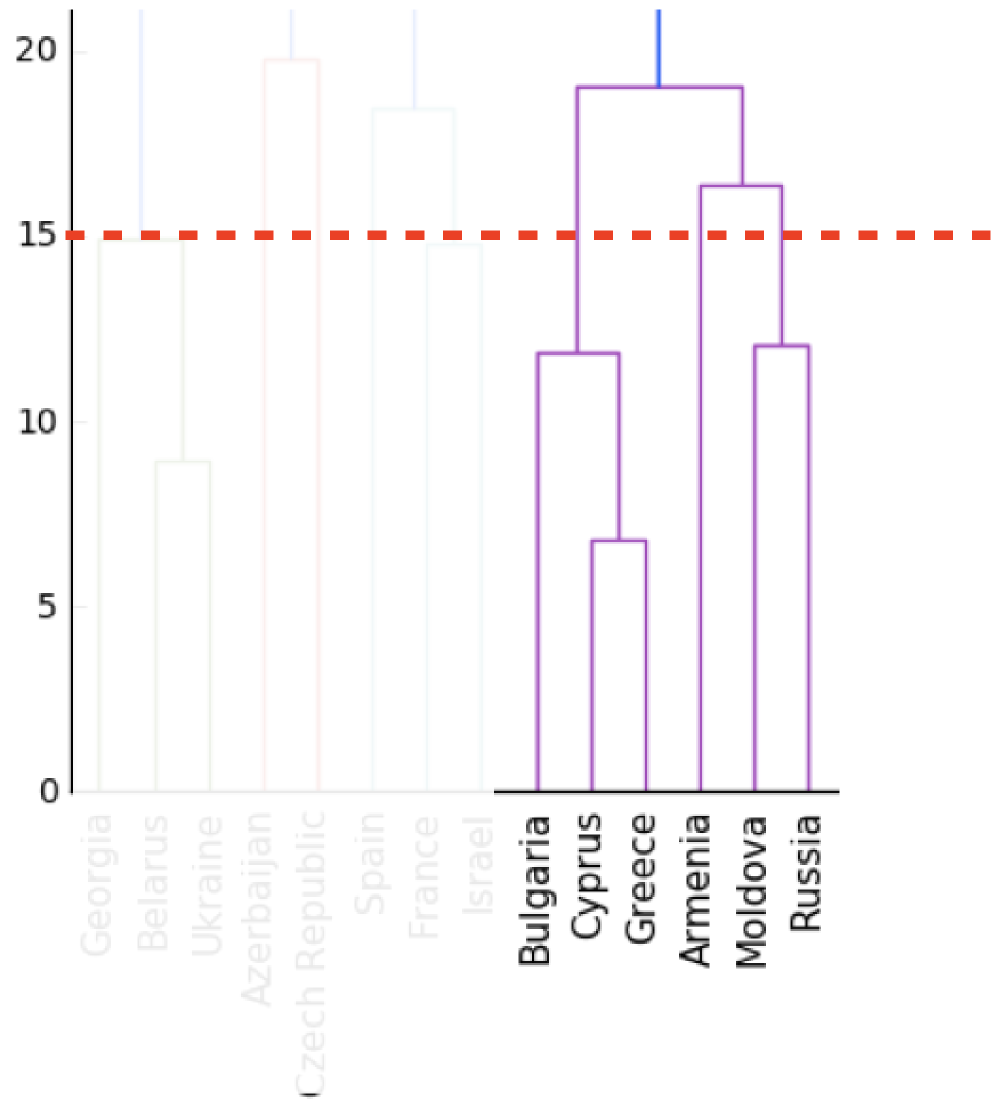
Cluster labels in hierarchical clustering

- Not only a visualization tool!
- Cluster labels at any intermediate stage can be recovered
- For use in e.g. cross-tabulations



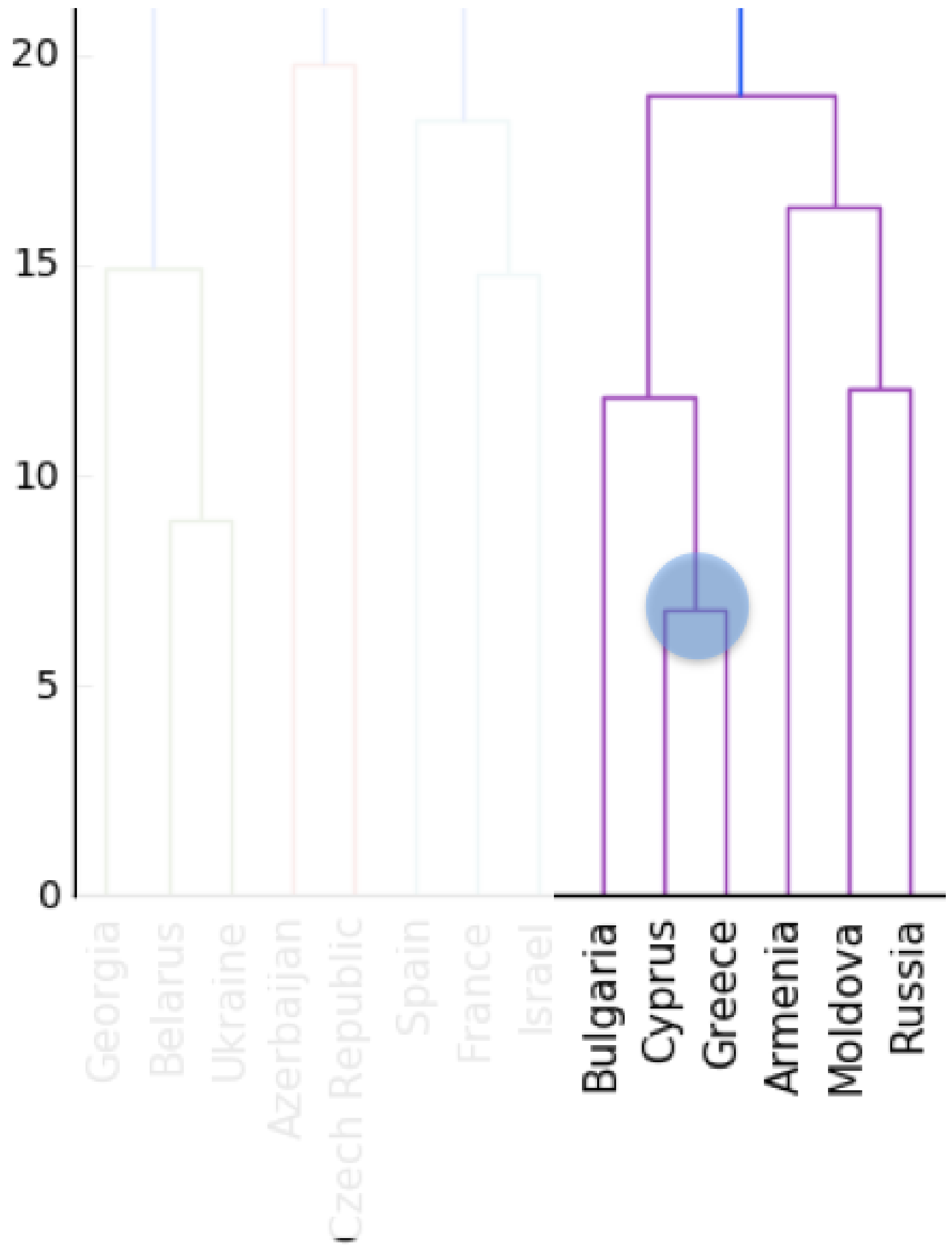
Intermediate clusterings & height on dendrogram

- E.g. at height 15:
  - Bulgaria, Cyprus, Greece are one cluster
  - Russia and Moldova are another
  - Armenia in a cluster on its own



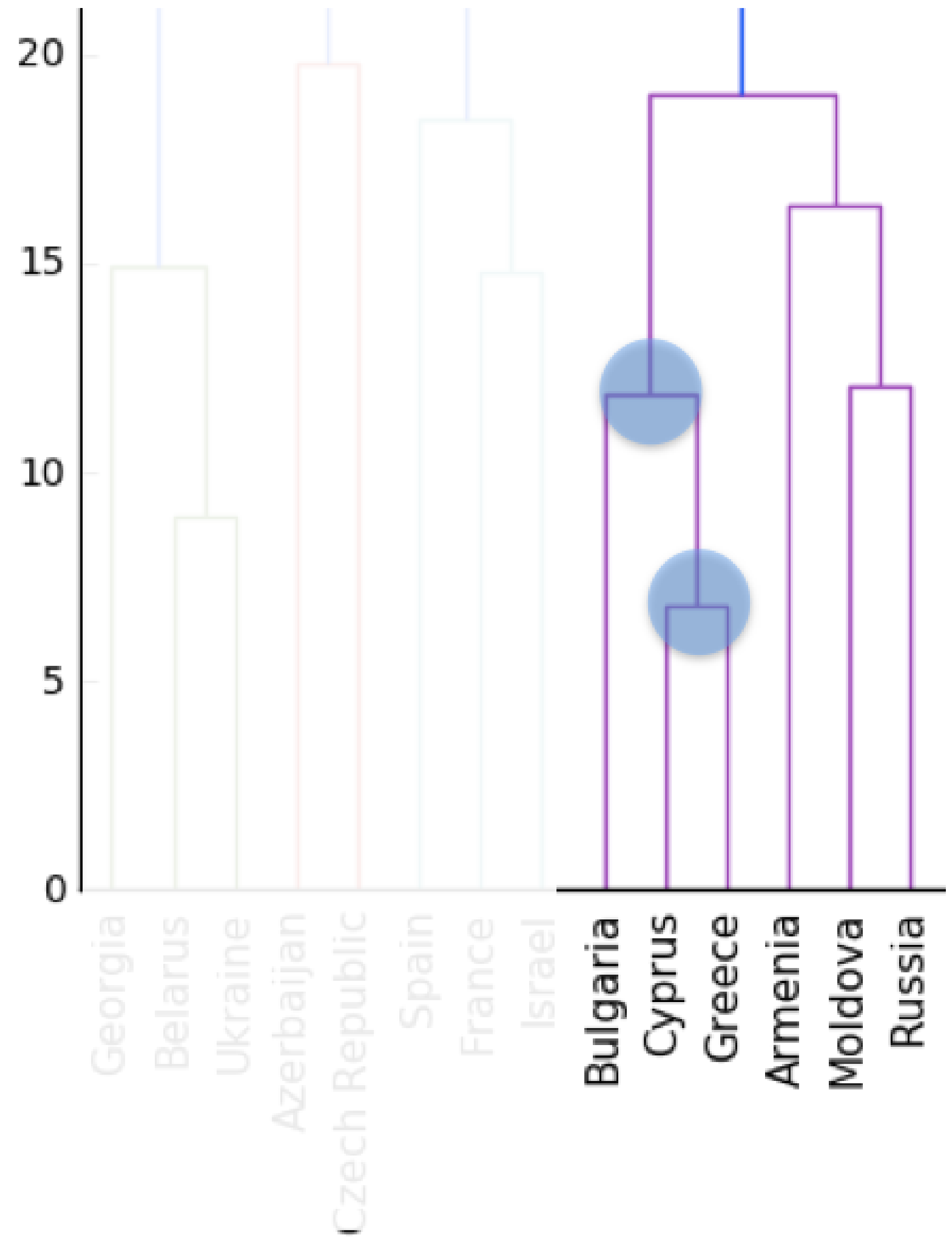
Dendrograms show cluster distances

- Height on dendrogram = distance between merging clusters
- E.g. clusters with only Cyprus and Greece had distance approx. 6



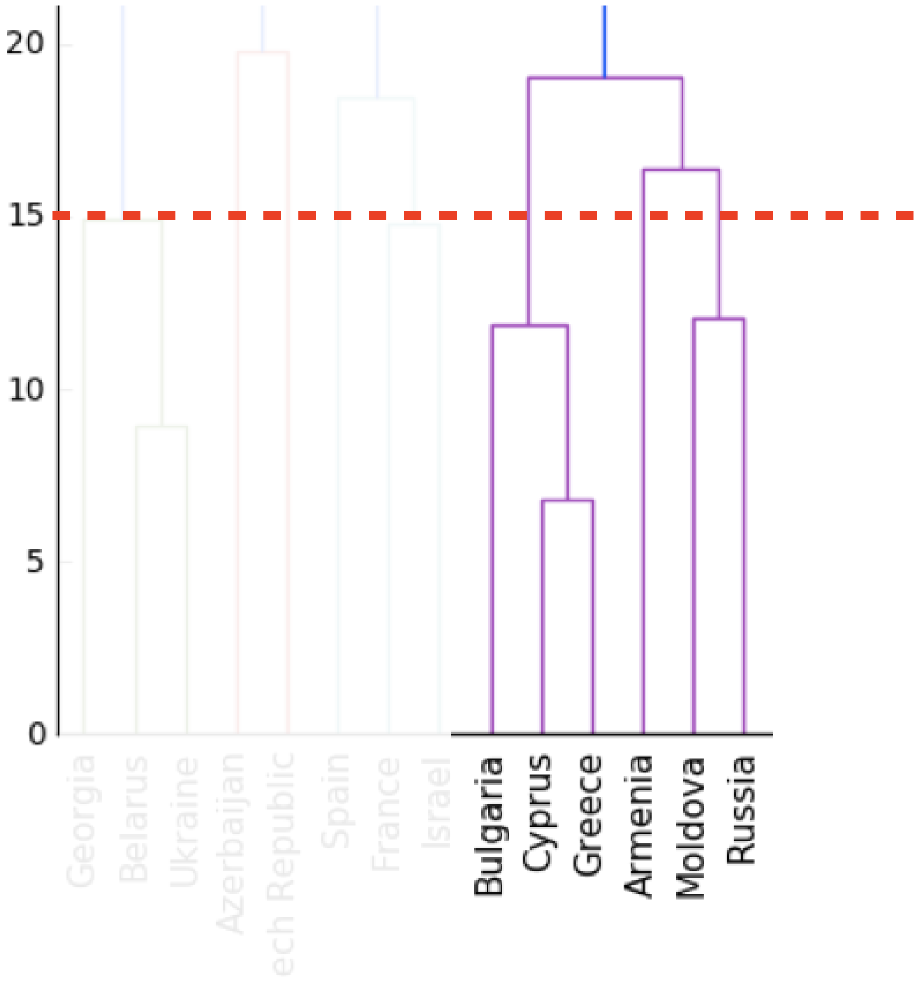
Dendrograms show cluster distances

- Height on dendrogram = distance between merging clusters
- E.g. clusters with only Cyprus and Greece had distance approx. 6
- This new cluster distance approx. 12 from cluster with only Bulgaria



Intermediate clusterings & height on dendrogram

- Height on dendrogram specifies max. distance between merging clusters
- Don't merge clusters further apart than this (e.g. 15)

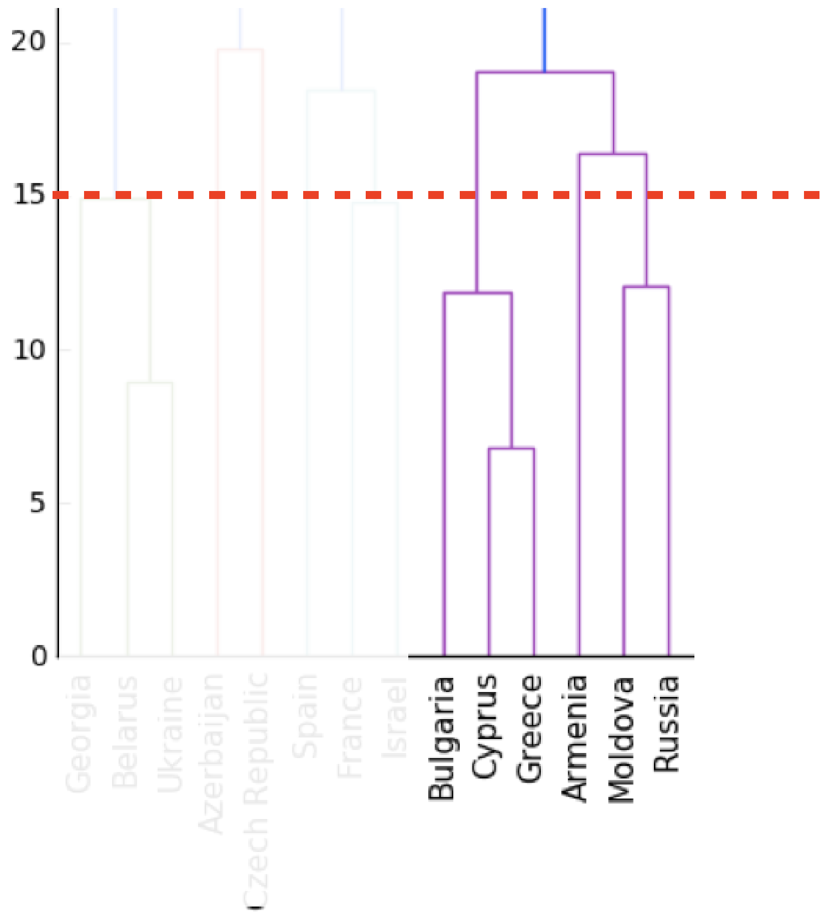


Distance between clusters

- Defined by a "linkage method"
- In "complete" linkage: distance between clusters is max. distance between their samples
- Specified via method parameter, e.g. linkage(samples, method="complete")
- Different linkage method, different hierarchical clustering!

Extracting cluster labels

- Use the `fcluster()` function
- Returns a NumPy array of cluster labels



Extracting cluster labels using fcluster

```
from scipy.cluster.hierarchy import linkage
mergings = linkage(samples, method='complete')
from scipy.cluster.hierarchy import fcluster
labels = fcluster(mergings, 15, criterion='distance')
print(labels)
```

```
[ 9  8 11 20  2  1 17 14 ... ]
```

Aligning cluster labels with country names

Given a list of strings `country_names`:

```
import pandas as pd
pairs = pd.DataFrame({'labels': labels, 'countries': country_names}
print(pairs.sort_values('labels'))
```

```
          countries  labels
5           Belarus       1
40          Ukraine       1
...
36            Spain       5
8          Bulgaria       6
19           Greece       6
10           Cyprus       6
28          Moldova       7
...
```

Let's practice!

UNSUPERVISED LEARNING IN PYTHON

t-SNE for 2-dimensional maps

UNSUPERVISED LEARNING IN PYTHON

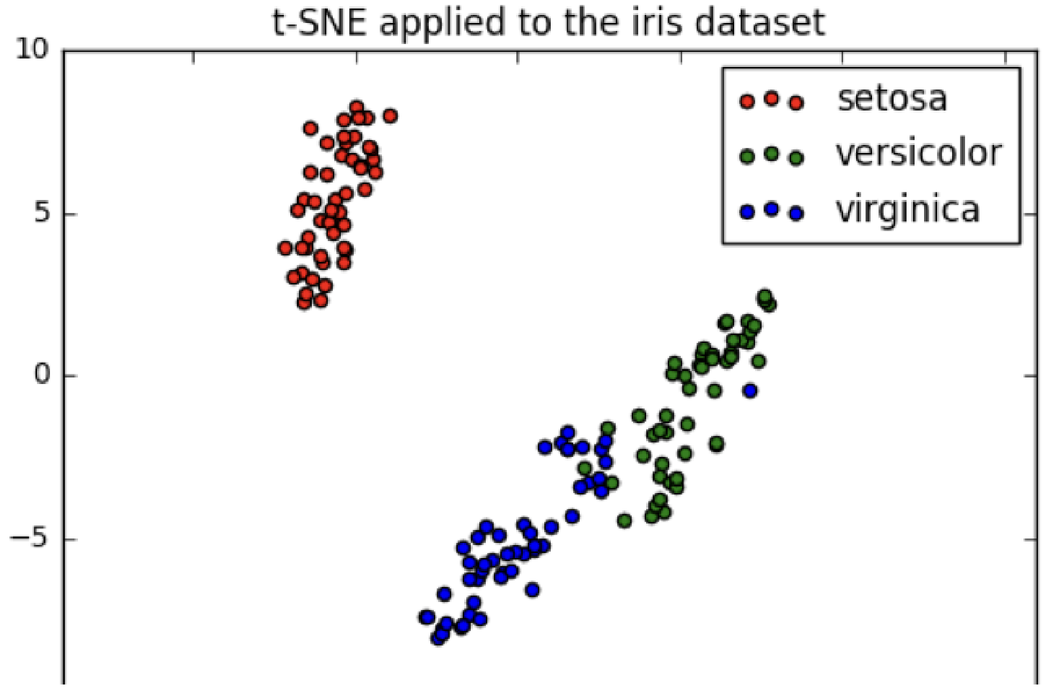
Benjamin Wilson
Director of Research at lateral.io

t-SNE for 2-dimensional maps

- t-SNE = "t-distributed stochastic neighbor embedding"
- Maps samples to 2D space (or 3D)
- Map approximately preserves nearness of samples
- Great for inspecting datasets

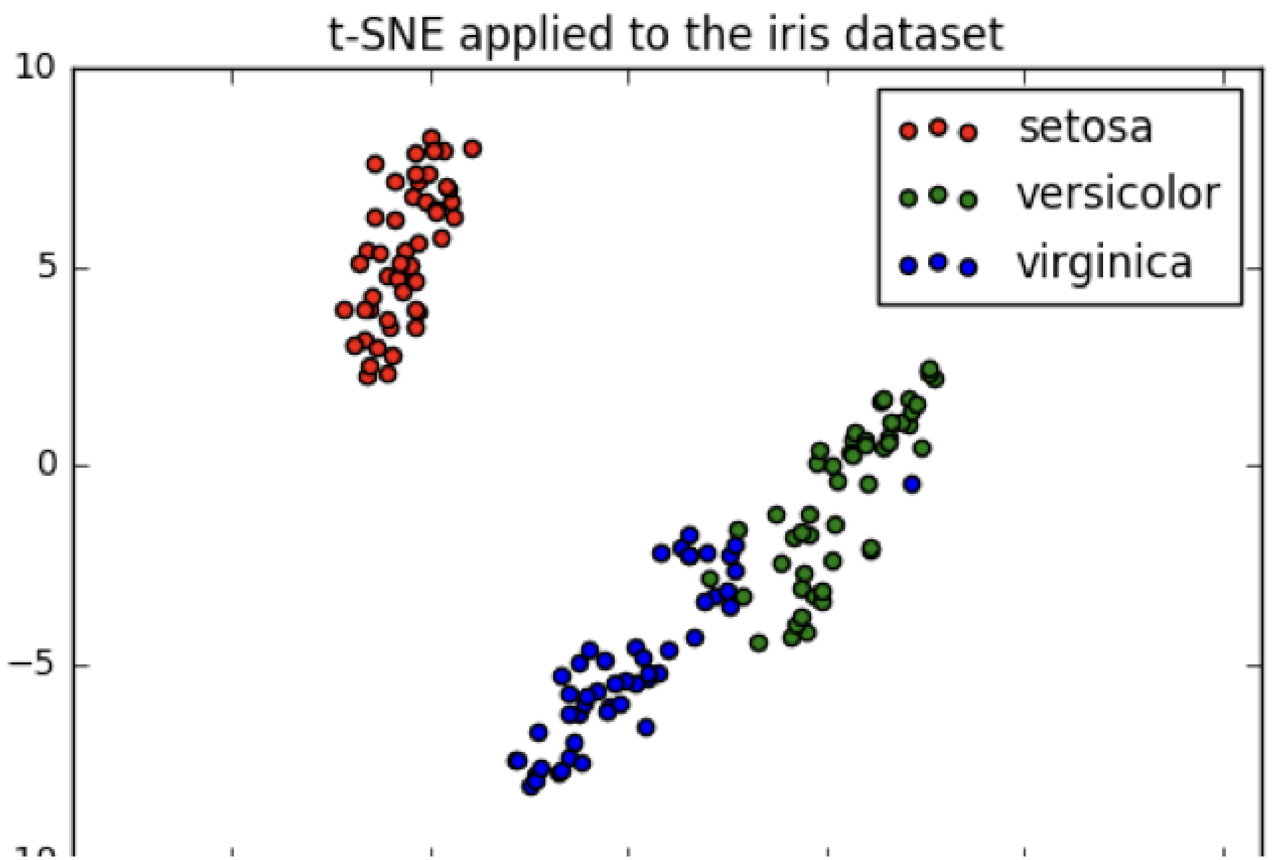
t-SNE on the iris dataset

- Iris dataset has 4 measurements, so samples are 4-dimensional
- t-SNE maps samples to 2D space
- t-SNE didn't know that there were different species
- ... yet kept the species mostly separate



Interpreting t-SNE scatter plots

- "versicolor" and "virginica" harder to distinguish from one another
- Consistent with k-means inertia plot: could argue for 2 clusters, or for 3



t-SNE in sklearn

- 2D NumPy array `samples`

```
print(samples)
```

```
[[ 5.   3.3  1.4  0.2]
 [ 5.   3.5  1.3  0.3]
 [ 4.9  2.4  3.3  1. ]
 [ 6.3  2.8  5.1  1.5]
 ...
 [ 4.9  3.1  1.5  0.1]]
```

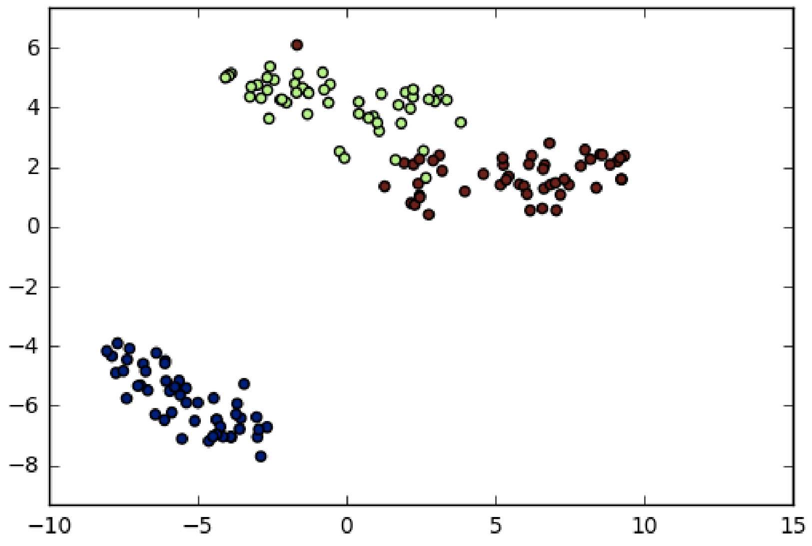
- List `species` giving species of labels as number (0, 1, or 2)

```
print(species)
```

```
[0, 0, 1, 2, ..., 0]
```

t-SNE in sklearn

```
import matplotlib.pyplot as plt
from sklearn.manifold import TSNE
model = TSNE(learning_rate=100)
transformed = model.fit_transform(samples)
xs = transformed[:,0]
ys = transformed[:,1]
plt.scatter(xs, ys, c=species)
plt.show()
```



t-SNE has only fit_transform()

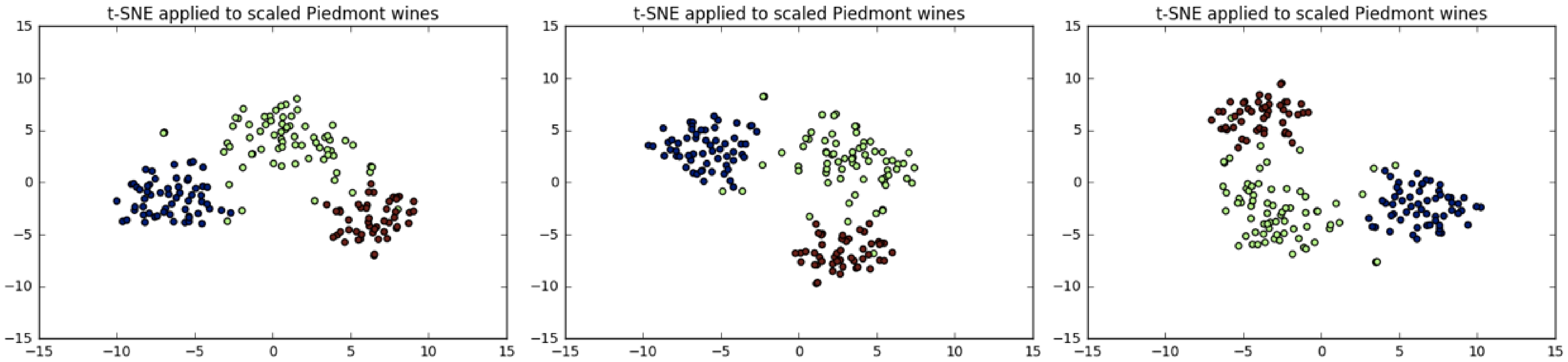
- Has a `fit_transform()` method
- Simultaneously fits the model and transforms the data
- Has no separate `fit()` or `transform()` methods
- Can't extend the map to include new data samples
- Must start over each time!

t-SNE learning rate

- Choose learning rate for the dataset
- Wrong choice: points bunch together
- Try values between 50 and 200

Different every time

- t-SNE features are different every time
- Piedmont wines, 3 runs, 3 different scatter plots!
- ... however: The wine varieties (=colors) have same position relative to one another



Let's practice!

UNSUPERVISED LEARNING IN PYTHON