Dimensionality Reduction

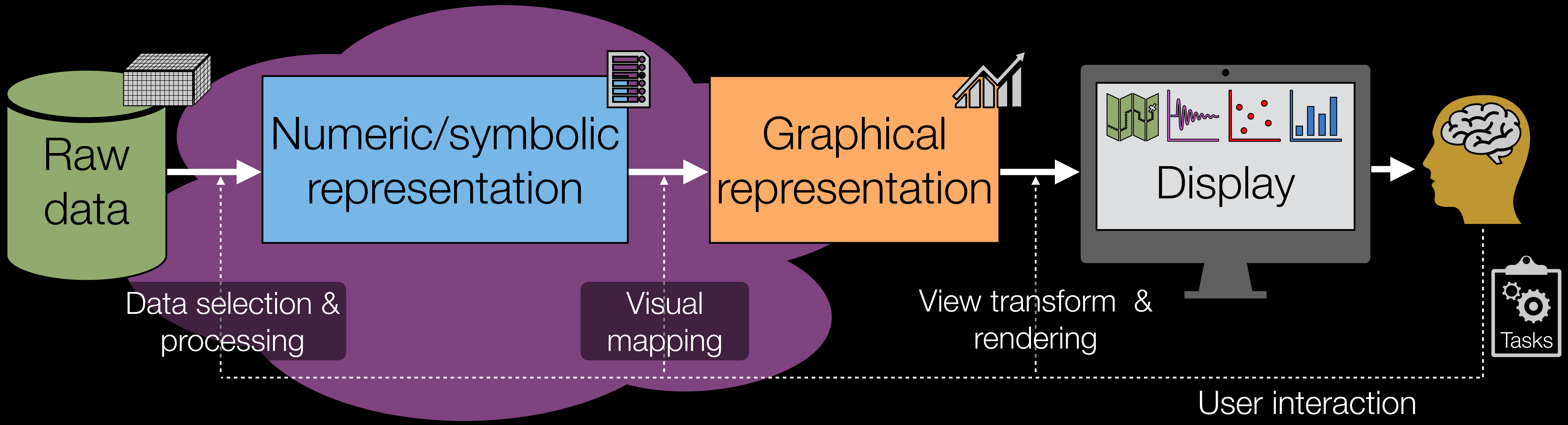
Prof. Dr. Renato Pajarola
Visualization and MultiMedia Lab
Department of Informatics
University of Zürich

Copyrights

Overview

1. Dimensionality reduction
2. Principal component analysis
3. Multidimensional scaling
4. Self-organizing maps

Where are we in the Visualization Pipeline?

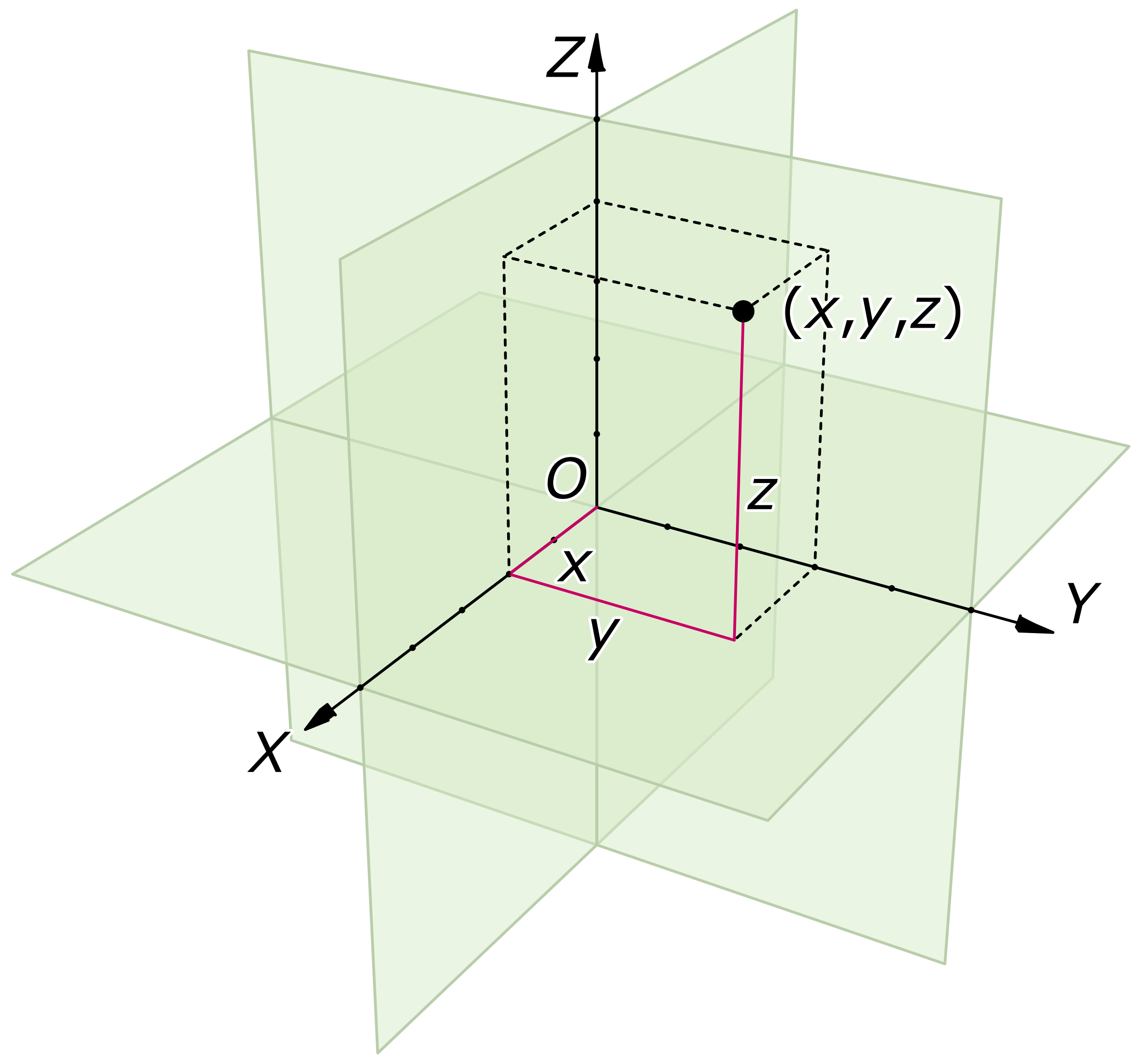


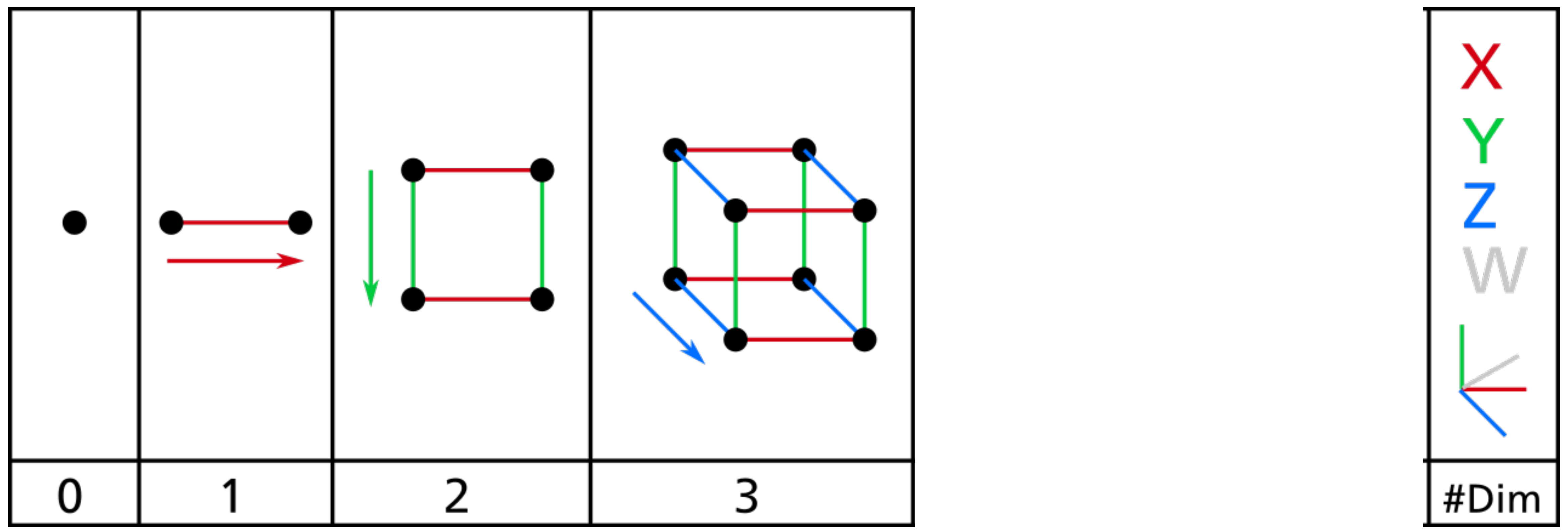
Raw data
Numeric/symbolic representation
Graphical representation
Display
Data selection & processing
Visual mapping
View transform & rendering
Tasks
User interaction

- Data preprocessing and transformation
  - Selection of information and mapping to fundamental computer data types
  - Data cleaning, interpolation, sampling, filtering, aggregation, partitioning
- Mapping for visualization
  - Specific visual representation (geometry, color)
  - Embedding in Euclidean 2D/3D space
- Rendering transformations
  - Final image synthesis by 2D imaging and 3D graphics technology
  - Interactive data and view selection

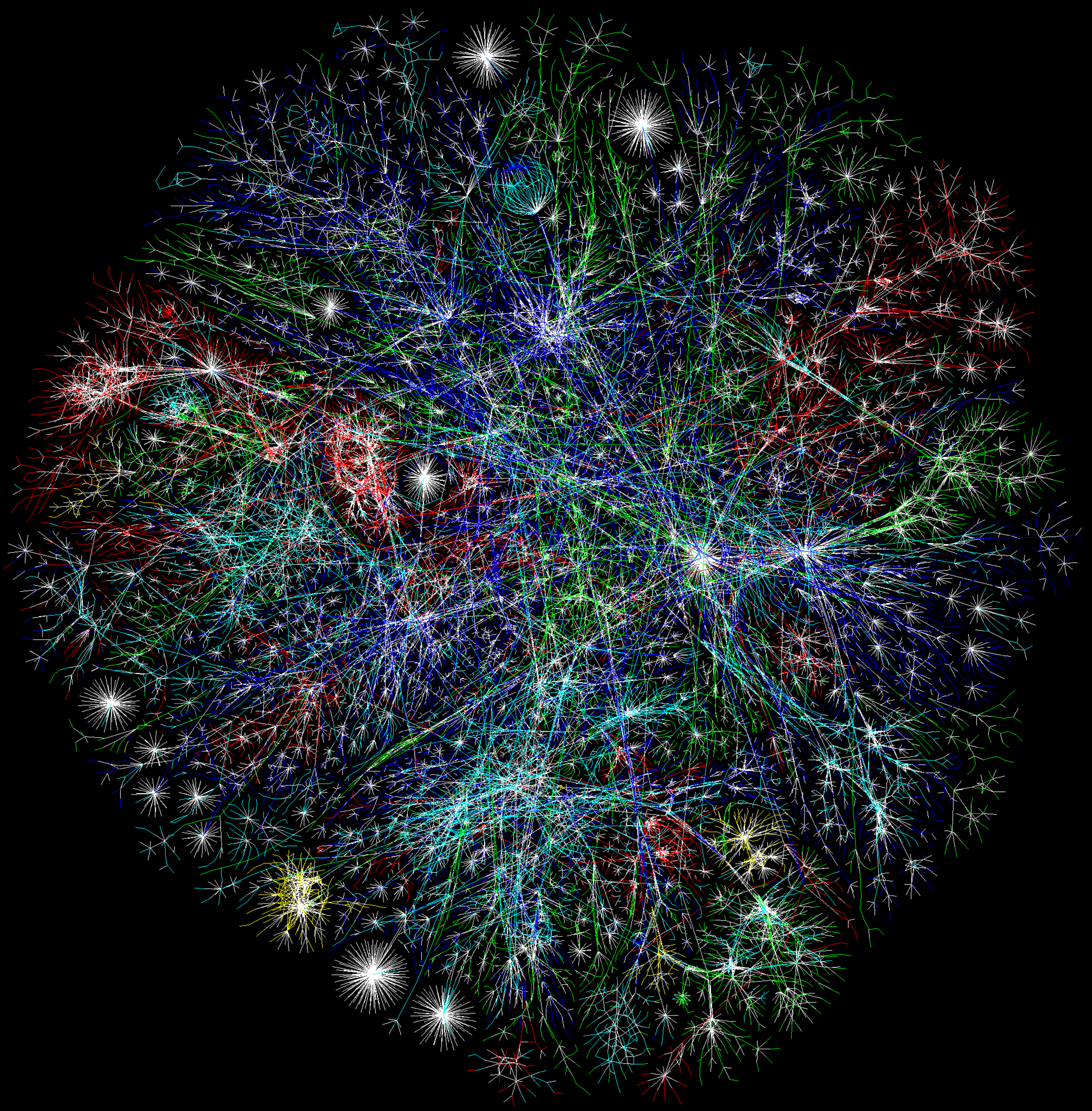
Dimensionality

- Dimension of space in which some data object is embedded
- Points, lines or volumes in Euclidean 3D space
  - 0D, 1D, 2D or 3D dimensional in their extend!
- Geometric objects have 3D coordinates
- Data objects have k attributes or variables
  - Data vector with k entries
  - Dimensionality of data determined by k
- Data set typically consists of a list of $\boldsymbol{n}$ data records $\boldsymbol{r}_i$ each with k variables v_j





Dimensionality Reduction



There are situations where the dimensionality exceeds the capabilities of the visualization technique.

The goal of dimensionality reduction is to reduce the number of dimensions, data attributes while preserving as much information as possible.

Dimensionality Reduction

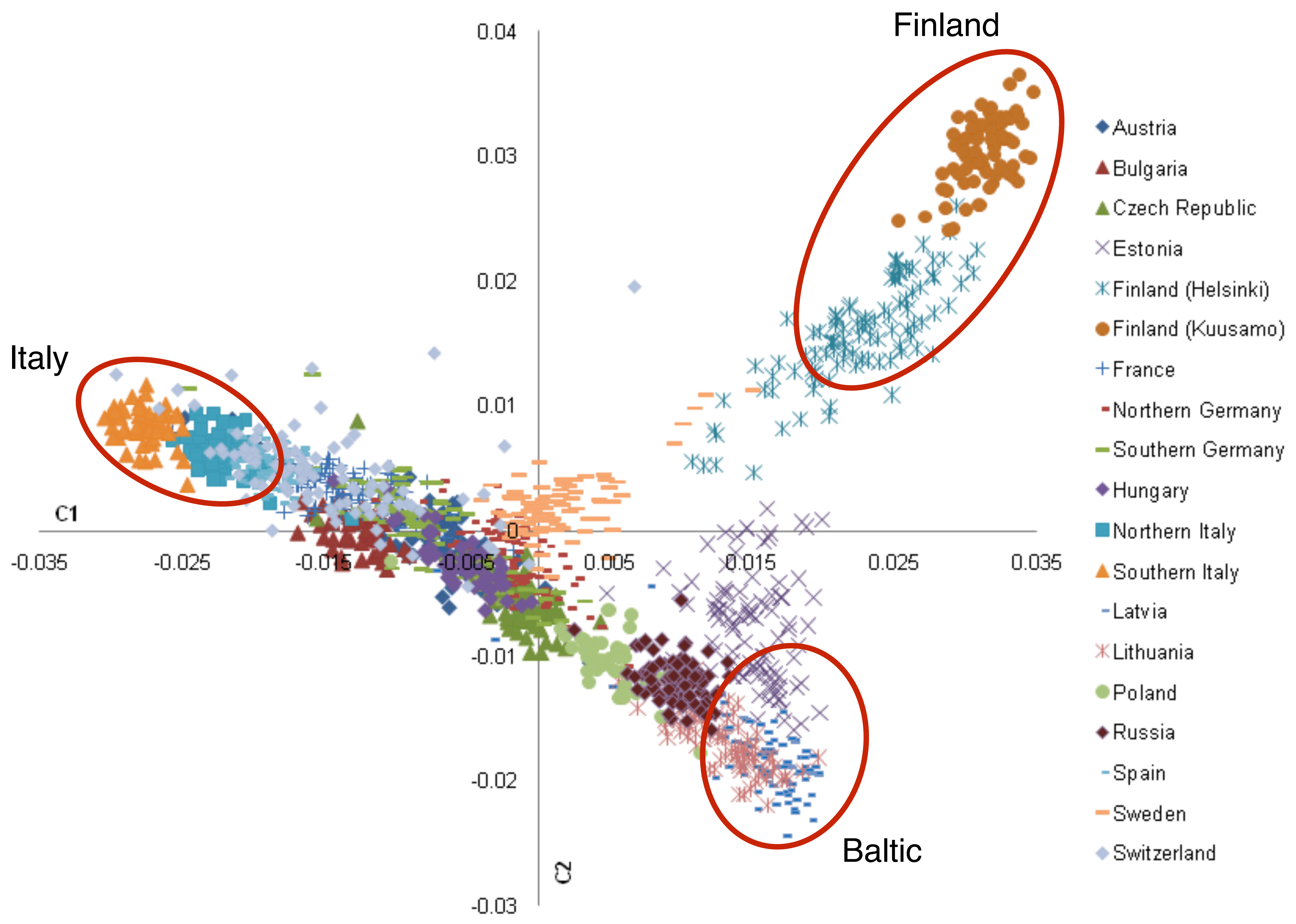
- Many datasets have a dimensionality, number of attribute value fields per data point, that exceeds the traditional plotting and visualization domain
  - 1D charts and curve plots, 2D height-plots and maps, 3D visualizations
- Dimensionality reduction techniques must preserve relative relations between data records
  - E.g. clusters, patterns, outliers
- Computational dimension reduction methods depend on starting configurations and algorithm parameters, mostly not unique
  - Principal component analysis (PCA)
  - Multidimensional scaling (MDS)
  - Kohonen self-organizing maps (SOMs)
- Mapping into 2D or 3D very important step for data visualization

Principal Component Analysis (PCA)

An orthogonal linear transformation that transforms a set of possibly correlated variables into a set of linearly uncorrelated variables called *principal components*.

The projection of the data onto the first coordinate or principal component captures the largest possible variance in the data.

The second greatest variance on the second coordinate, and so on.

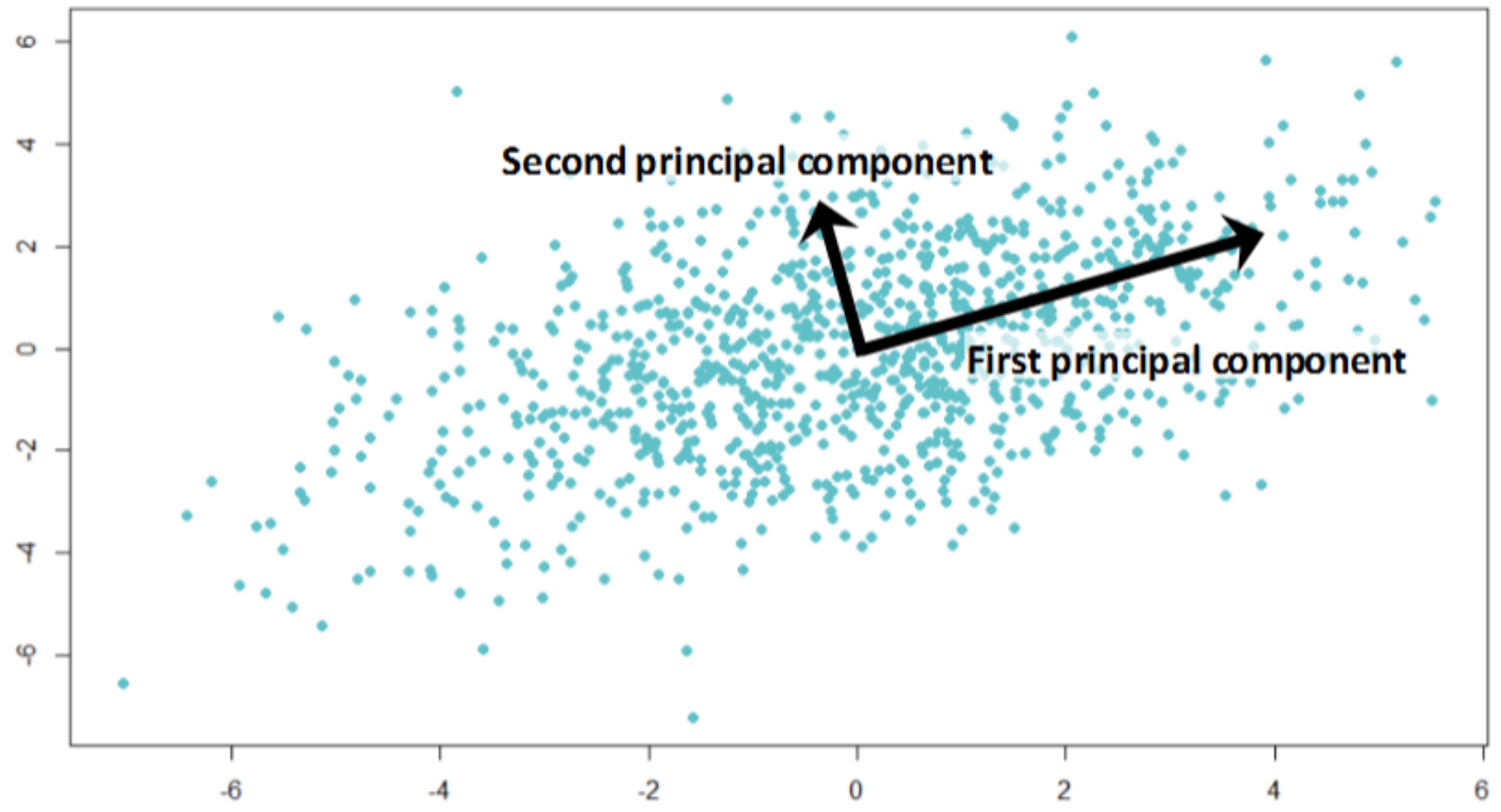


http://archhades.blogspot.com/2011/06/northern-italians-are-biologically.html

PCA of single nucleotide polymorphisms (SNPs) genotyped with the Illumina Infinium platform

PCA by Eigendecomposition

- PCA is mathematically defined as the eigendecomposition of the data's covariance matrix
  - The principal components being the eigenvectors
- Basic steps
  1. Subtract mean from each one of the dimensions to center the data around the origin
  2. Build covariance matrix
  3. Get normalized and ordered eigenvectors
  4. Project data onto new reduced dimensions

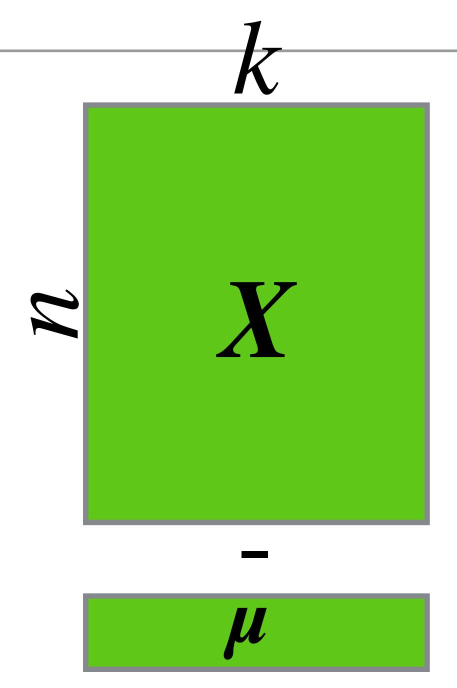


https://www.analyticsvidhya.com/blog/2016/03/pca-practical-guide-principal-component-analysis-python/

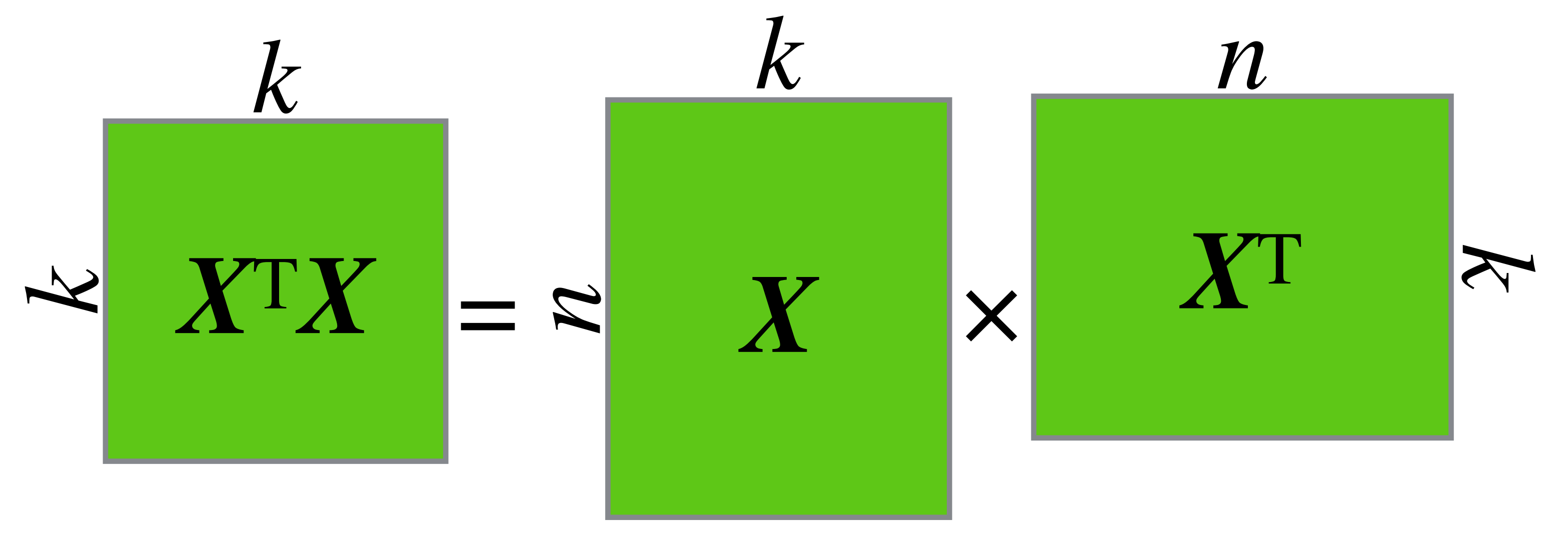
PCA Basic Steps

1. Subtract column-wise mean from data
   - Data given by n x k matrix X of n data records (rows) with k values each (columns)
   - Mean vector given by k-dimensional vector $\boldsymbol{\mu}$ as average over columns
   - Column-wise zero mean meatrix $X \rightarrow X - \boldsymbol{\mu}$ (per row)

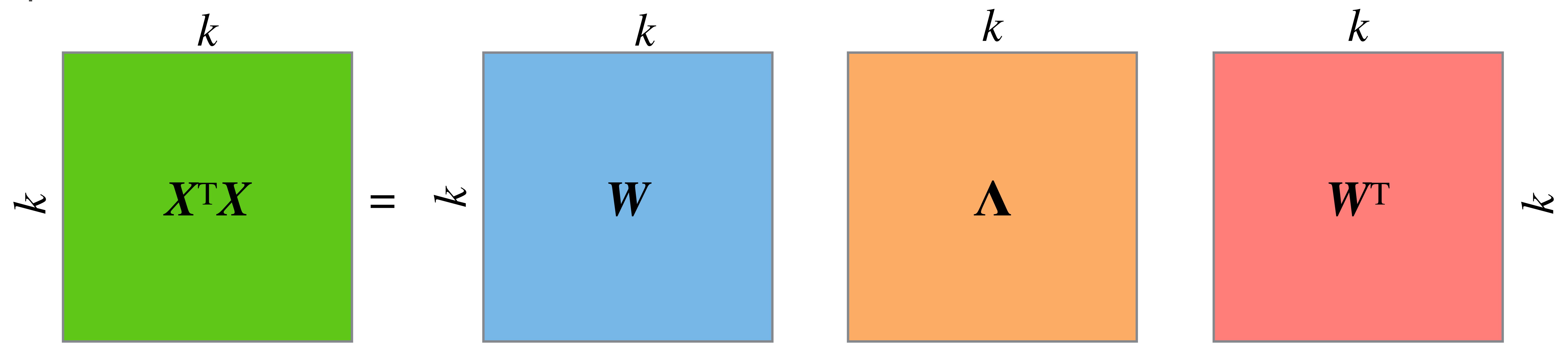
     - mean shifted to zero



2. Build covariance matrix
   - Multiplication with transpose gives k x k covariance matrix X^TX

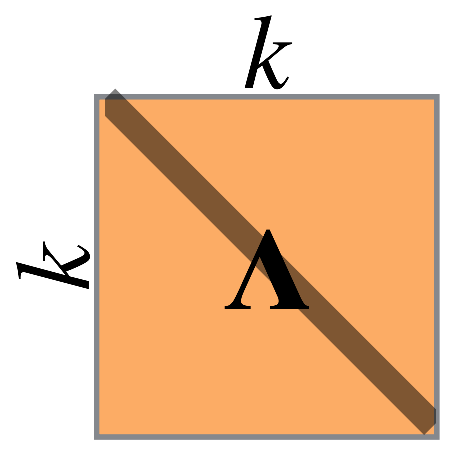


3. Build eigendecomposition
   - Compute eigendecomposition $X^TX = W\ \Lambda\ W^T$

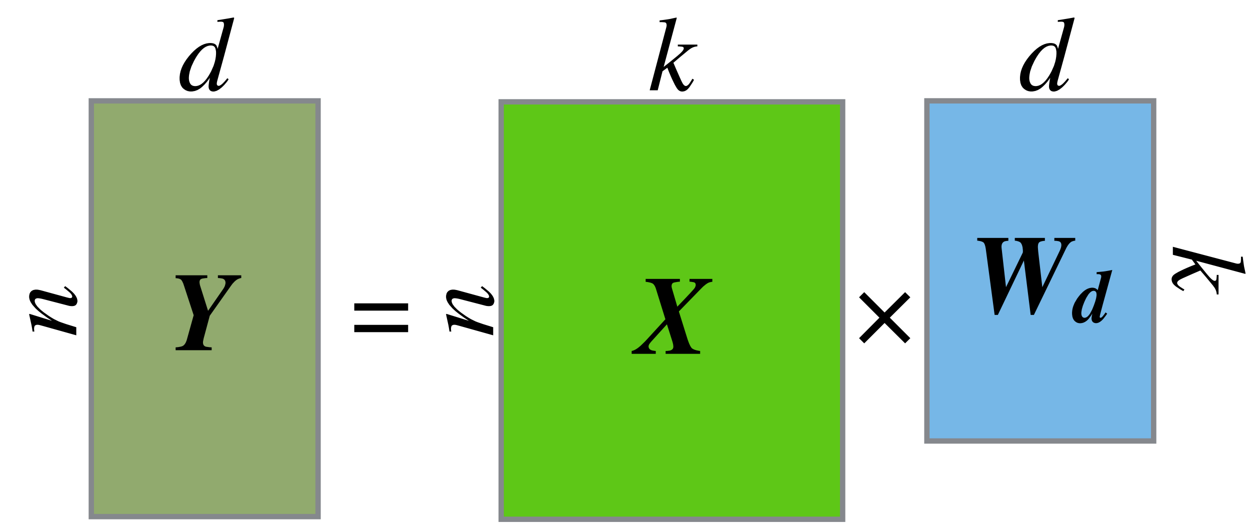


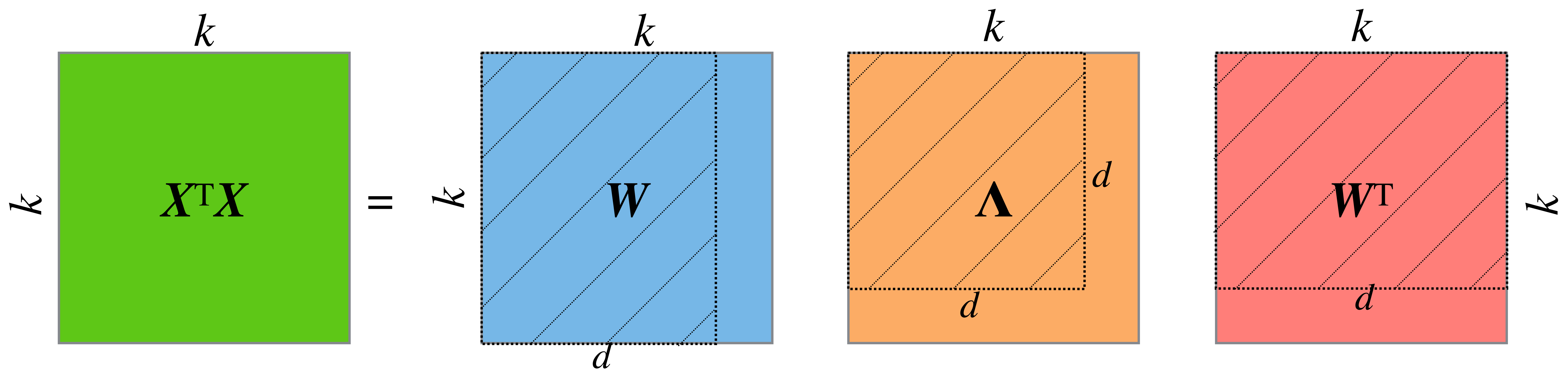
PCA Basic Steps

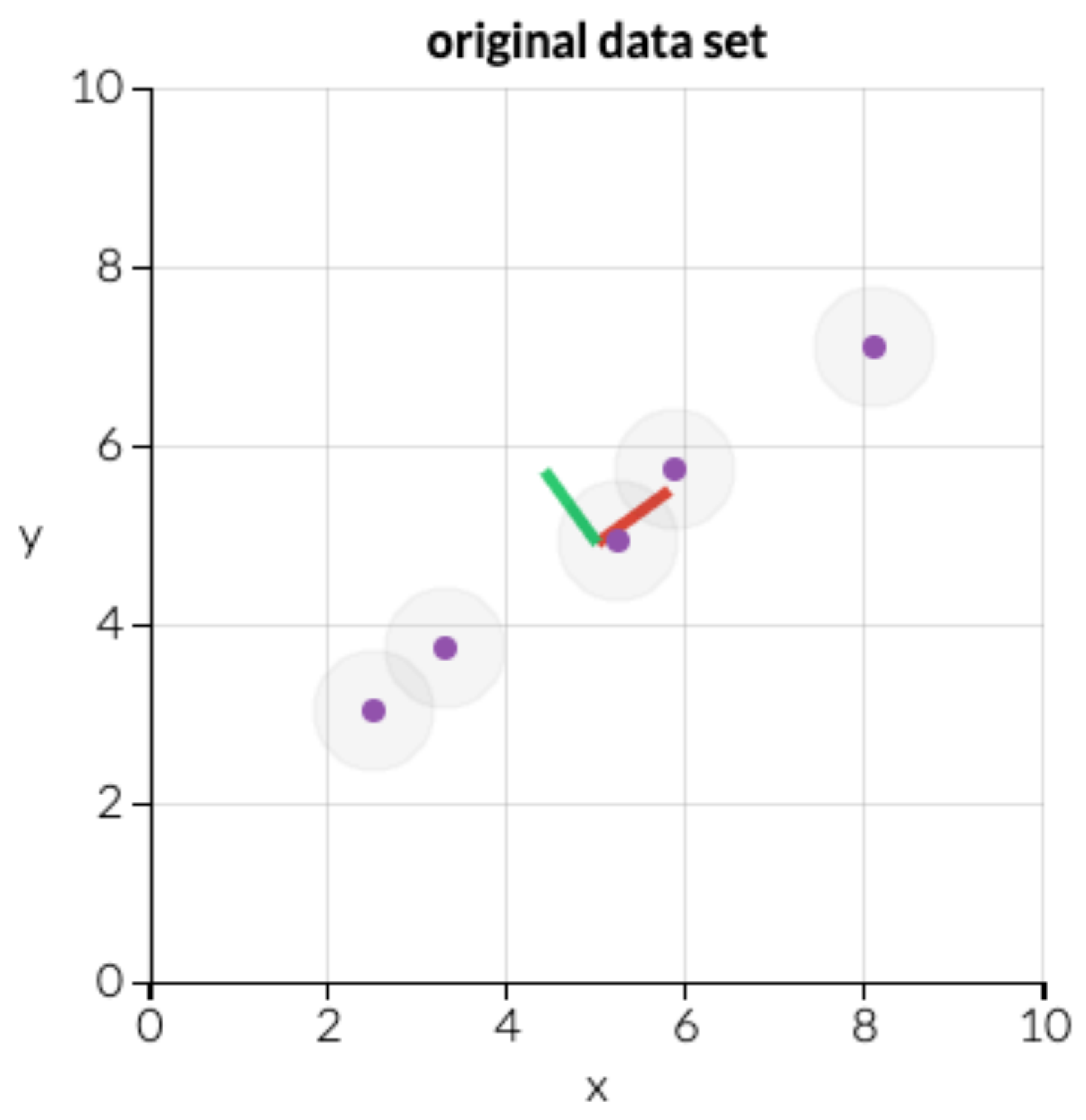
3. Get ordered eigenvectors
   - Select d eigenvalues from diagonal matrix $\boldsymbol{\Lambda}$ of eigendecomposition $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X} = \boldsymbol{W}\ \boldsymbol{\Lambda}\ \boldsymbol{W}^{\mathrm{T}}$
   - Columns of $\boldsymbol{W}$ denote the eigenvectors of $\boldsymbol{X}$

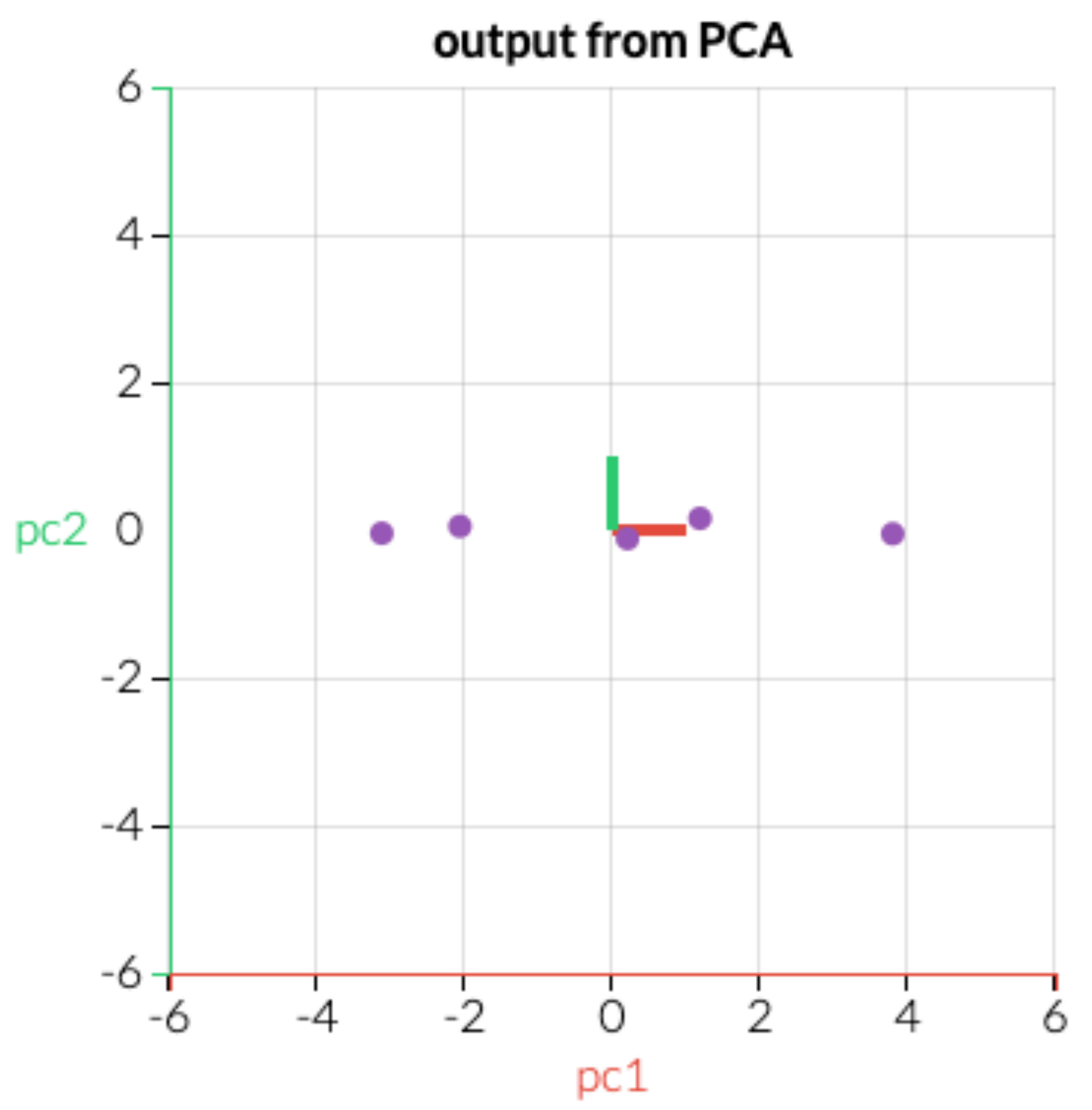


4. Project data onto new reduced dimensions
   - Use $\boldsymbol{W}_d$ using only the first d columns (eigenvectors) of $\boldsymbol{W}$
   - Project data $\boldsymbol{X}$ into reduced d-dimensional space by $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{W}_d$



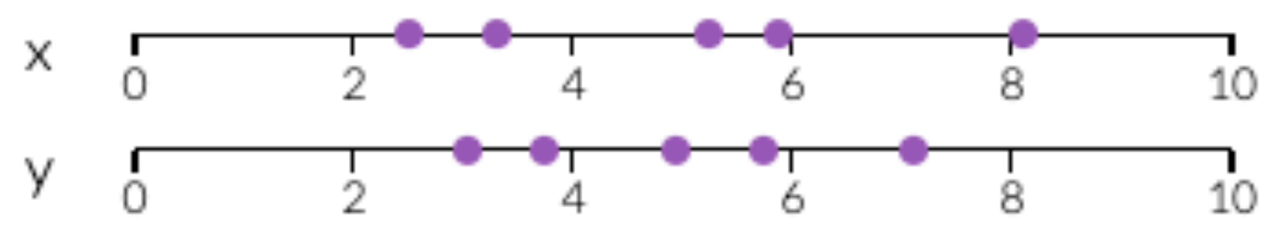


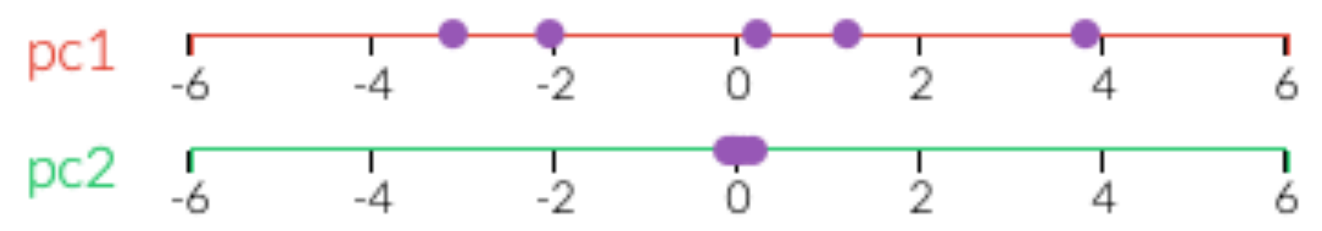




PCA is useful for eliminating dimensions. Below, we've plotted the data along a pair of lines: one composed of the x-values and another of the y-values.

If we're going to only see the data along one dimension, though, it might be better to make that dimension the principal component with most variation. We don't lose much by dropping PC2 since it contributes the least to the variation in the data set.

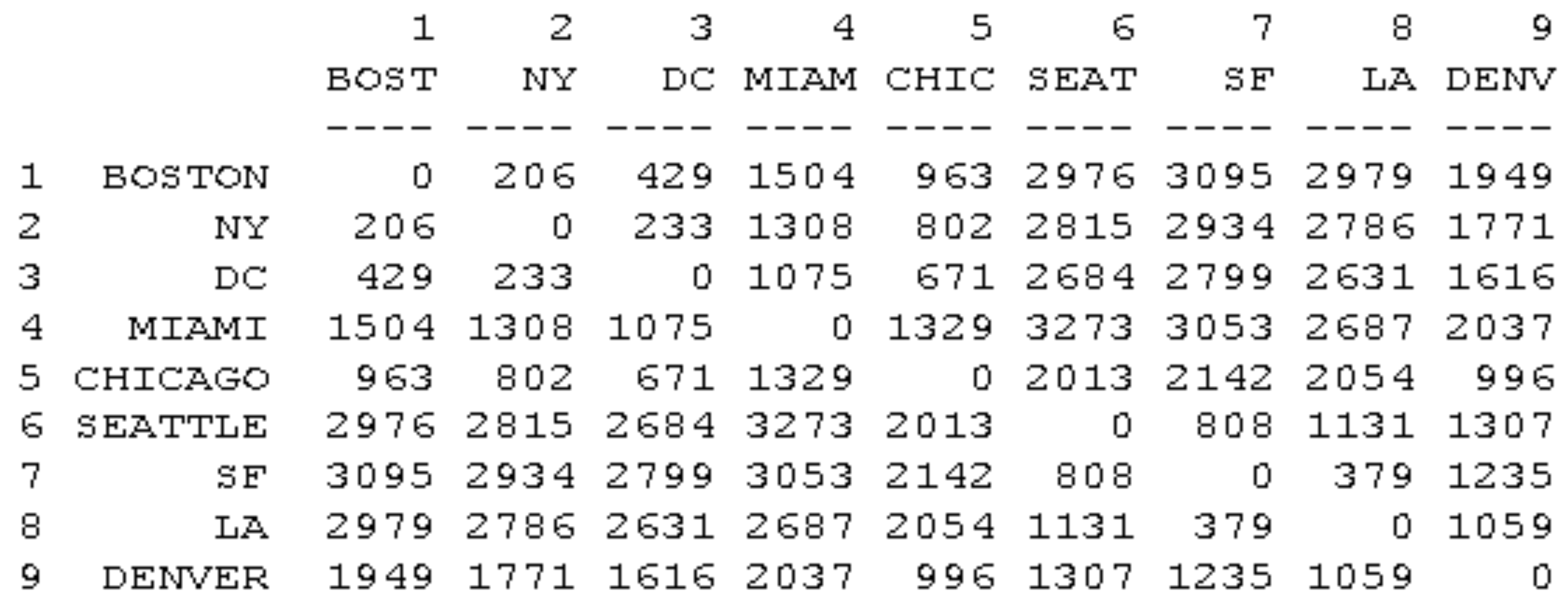




http://setosa.io/ev/principal-component-analysis/

Multidimensional Scaling

- Multidimensional scaling (MDS) maps high-dimensional data to lower dimensions
  - Starts with points mapped arbitrarily in the 2D plane, or 3D space
  - Compare matrix of Euclidean 2D/3D pairwise distances $\delta_{i,j}$ with the input distance matrix $D_{i,j}$ → stress function
    - e.g. mean-squared error
  - Iteratively adjust coordinates, move the 2D/3D points, to minimize stress
    - gradient descent, conjugate gradient, simulated annealing etc.

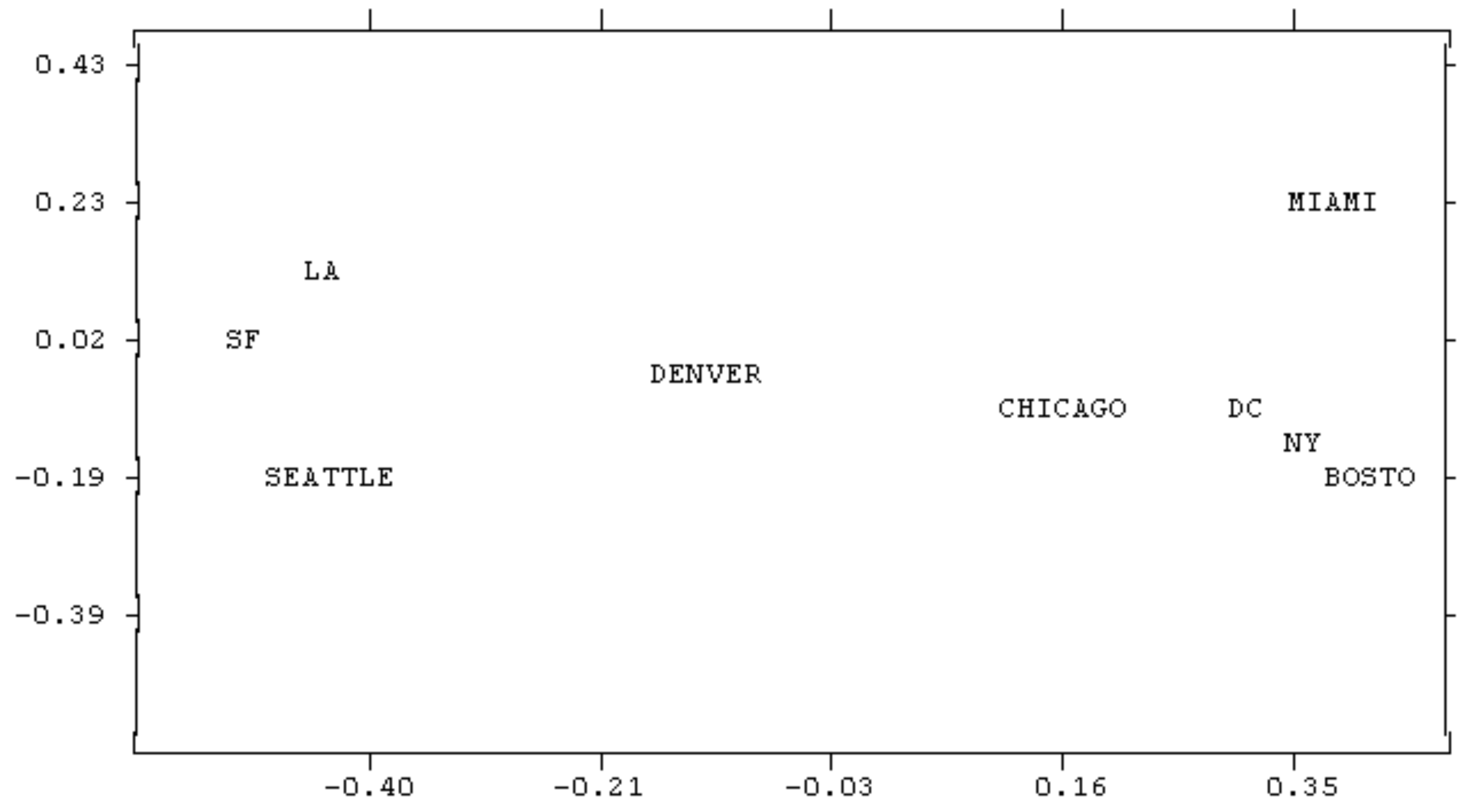


```
                1    2    3    4    5    6    7    8    9
             BOST   NY   DC MIAM CHIC SEAT   SF   LA DENV
             ---- ---- ---- ---- ---- ---- ---- ---- ----
1    BOSTON     0  206  429 1504  963 2976 3095 2979 1949
2        NY   206    0  233 1308  802 2815 2934 2786 1771
3        DC   429  233    0 1075  671 2684 2799 2631 1616
4     MIAMI  1504 1308 1075    0 1329 3273 3053 2687 2037
5   CHICAGO   963  802  671 1329    0 2013 2142 2054  996
6   SEATTLE  2976 2815 2684 3273 2013    0  808 1131 1307
7        SF  3095 2934 2799 3053 2142  808    0  379 1235
8        LA  2979 2786 2631 2687 2054 1131  379    0 1059
9    DENVER  1949 1771 1616 2037  996 1307 1235 1059    0
```

Stephen P. Borgatti, http://www.analytictech.com/borgatti/mds.htm, 1997.

Multidimensional Scaling

- Multidimensional scaling (MDS) maps high-dimensional data to lower dimensions
  - Starts with points mapped arbitrarily in the 2D plane, or 3D space
  - Compare matrix of Euclidean 2D/3D pairwise distances $\delta_{i,j}$ with the input distance matrix $D_{i,j}$ → stress function
    - e.g. mean-squared error
  - Iteratively adjust coordinates, move the 2D/3D points, to minimize stress
    - gradient descent, conjugate gradient, simulated annealing etc.



Stephen P. Borgatti, http://www.analytictech.com/borgatti/mds.htm, 1997.

MDS Algorithm

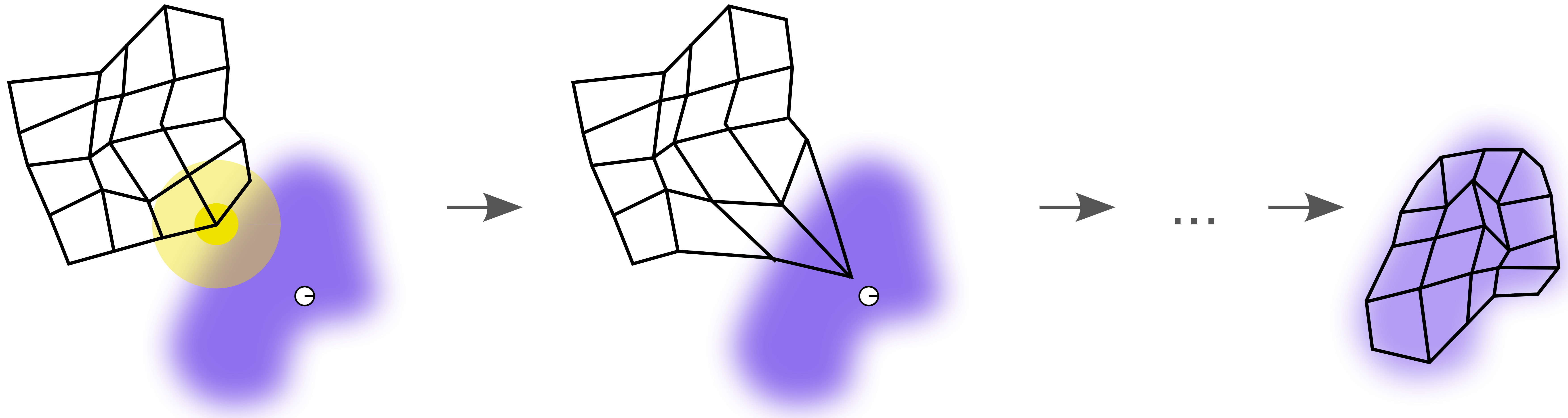
- Basic structure of method:
  1. Given n data records in k dimensions, create n x n distance matrix $\boldsymbol{D_s}$
  2. Initialize (random or from PCA) locations for projected points in $d \leq 3$ dimensions, stored in a n x d matrix $\boldsymbol{L}$
  3. Compute n x n (distance) matrix $\boldsymbol{L_s}$ containing similarities of point pairs in $\boldsymbol{L}$
  4. Compute stress $\boldsymbol{S}$, measuring the difference between $\boldsymbol{D_s}$ and $\boldsymbol{L_s}$
  5. If $\boldsymbol{S}$ is sufficiently small or has not changed, terminate
  6. Update positions of $\boldsymbol{L}$ in a direction to best reduce the stress $\boldsymbol{S}$
  7. Return to Step 3

Self-Organizing Maps (Kohonen Maps)

- Self-organizing maps (SOMs) reduce the dimensionality of data through self-organizing artificial neural networks (ANNs)
    - unsupervised learning technique
    - reveal similarities in input data
  - Start with randomly initialized map of neurons with weight vectors $\boldsymbol{w}$
    - neurons in some regular grid, with weight vectors of the same dimension k as input data points
  - Training causes the nodes of the neural net to respond similarly to certain input patterns
    - iteratively adjust initial weight vectors of neurons to the training data
- An input data point $\boldsymbol{x}$ is eventually placed on the map by finding the neuron with the closest weight vector $\boldsymbol{w}$
  - http://davis.wpi.edu/~matt/courses/soms/applet.html

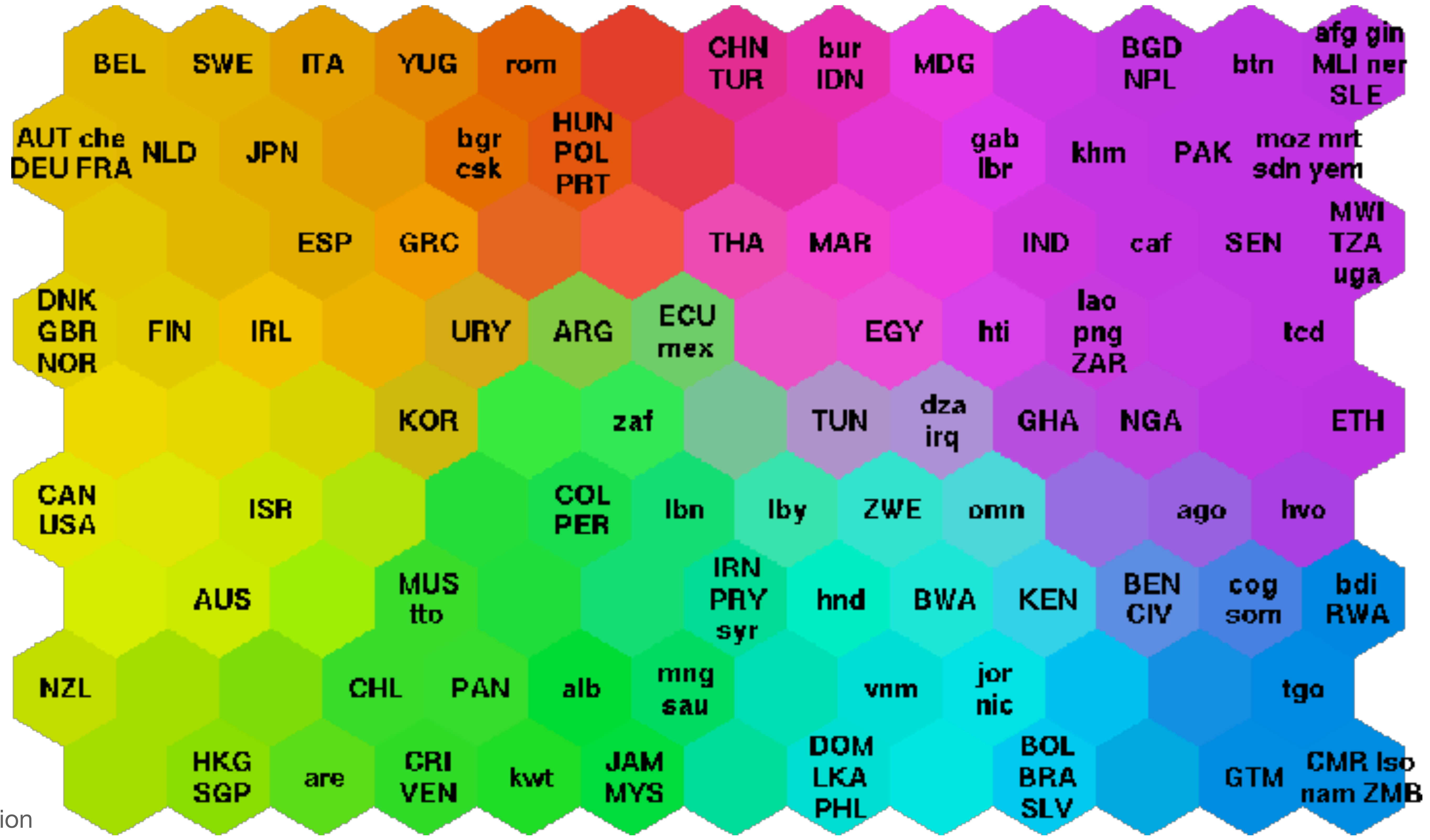
SOM Training

- Iterates over all input data samples $\boldsymbol{x}_i \in \boldsymbol{X}$ with attribute vectors $\boldsymbol{x}_i$ to train the neuron weights $\boldsymbol{w}_j$
  - Find closest neuron v and scale the weights $\boldsymbol{w}_j$ of itself and all its neighbors to become more similar to $\boldsymbol{x}_i$
    - $\boldsymbol{w}_j = \boldsymbol{w}_j + \Theta(j, v, t) \bullet \alpha(t) \bullet (\boldsymbol{x}_i - \boldsymbol{w}_j)$
    - scale factor $\alpha(t)$ decreases over time
    - distance weight function $\Theta(j, v, t)$, (e.g. Gaussian) becomes more narrow over time



SOM Example

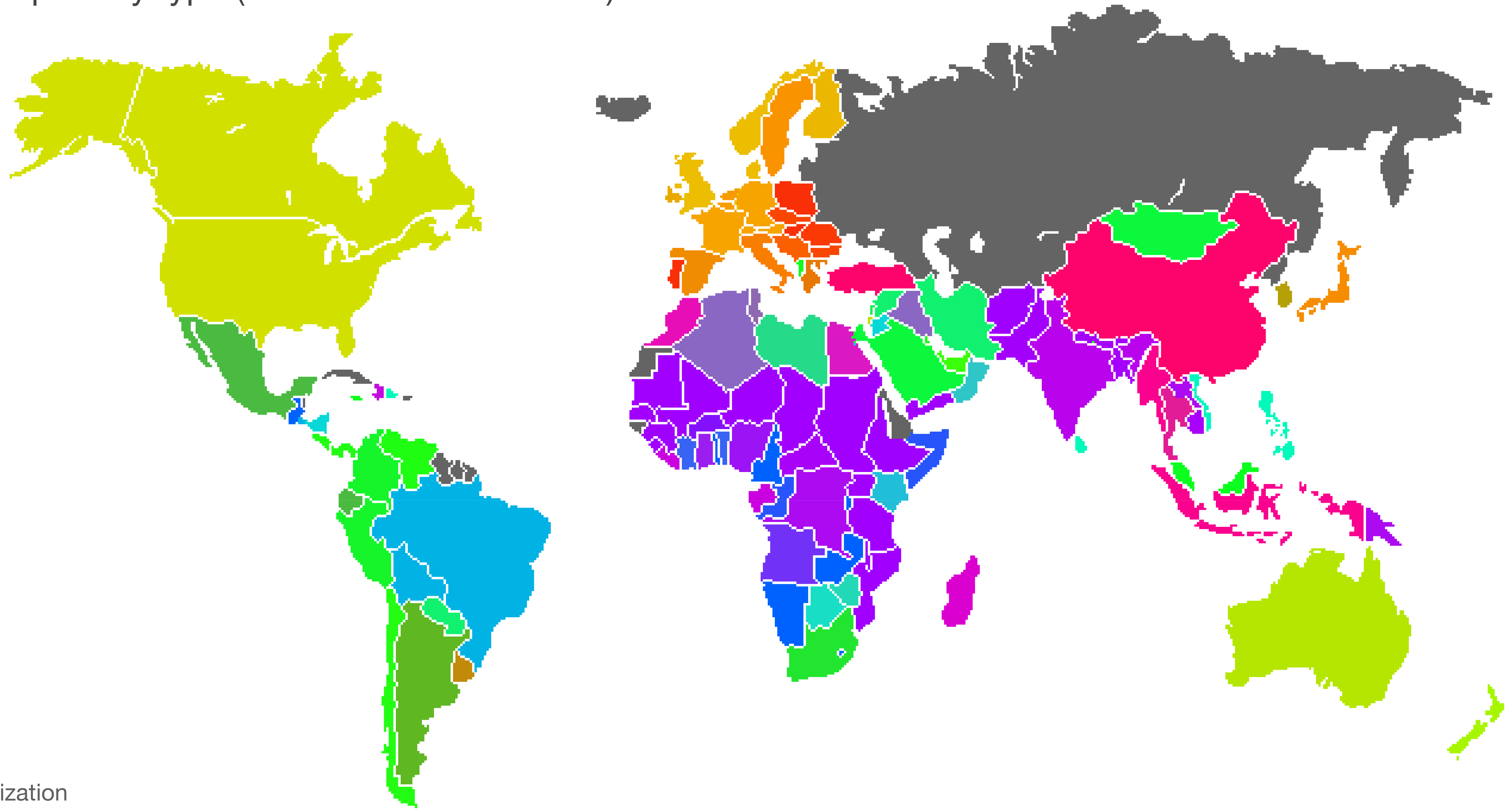
- 39-dimensional data vector describing quality-of-life factors
  - Color generated to smoothly transition between final map locations
    - indicates poverty type (set of influential factors) and relates countries
- Final positions Y in low-dimensional map space are the closest nodes of input data vectors X



Neural Networks Research Centre Helsinki University of Technology, http://www.cis.hut.fi/research/som-research/worldmap.html.

SOM Example

- 39-dimensional data vector describing quality-of-life factors
  - Color generated to smoothly transition between final map locations
    - indicates poverty type (set of influential factors) and relates countries
- Final positions Y in low-dimensional map space are the closest nodes of input data vectors X



Neural Networks Research Centre Helsinki University of Technology, http://www.cis.hut.fi/research/som-research/worldmap.html.

Recap

- **Dimensionality reduction:** dimensionality and data dimensions, reduction of dimensionality
- **Principal component analysis**: linear transformation and projection into lower dimensional space, orthogonal principal component axis corresponding to directions of strongest data variance, deterministic method with unique solution
- **Multidimensional scaling**: quantifies differences between distances in original k-dimensional data space and lower-dimensional mapping as cost function, solution by iterative numerical optimization, wide range of (differentiable) cost functions supported
- **Self-organizing maps**: learned mapping into lower dimensional space, unsupervised learning using simple artificial neural network

- Text book [2] *Mathematical Principles for Scientific Computing and Visualization* Chapter(s): 6.6, 6.9