FitchLearning

Unsupervised Machine Learning
Part 1

Dr. Claus Huber, CEFA, CFA, FRM
Head of Quantitative Modelling & Analytics
Helvetia Insurance, Basel

May 2023

Agenda

- Principal Component Analysis
- Hierarchical Clustering
- K-Means Clustering
- Self-Organising Map

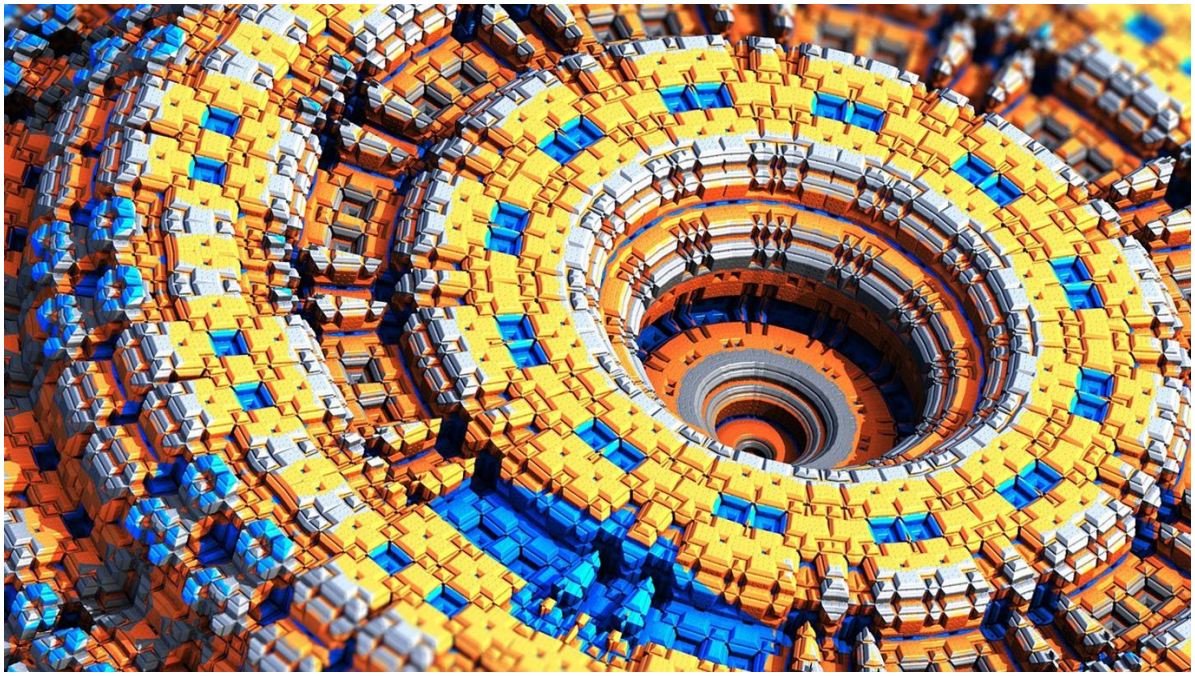
- Appendix: PCA, Eigenvectors and Eigenvalues

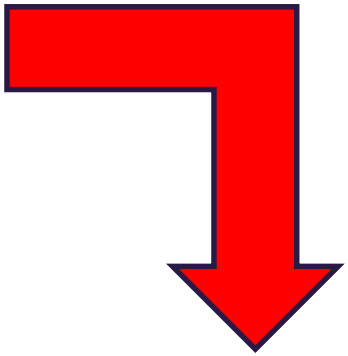
Why Unsupervised Learning?

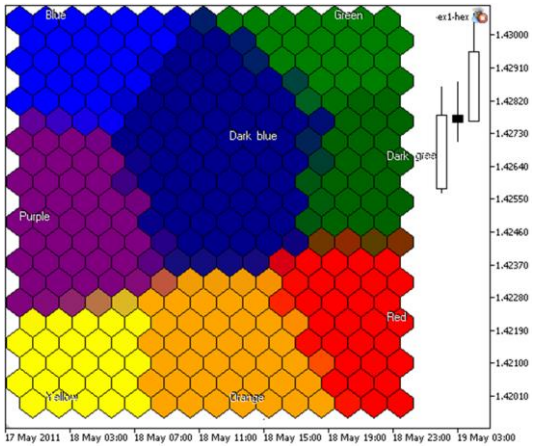
- Find structure in unknown, possibly high-dimensional data
- In a data set, find individual data points that belong together
  - For example, are there assets that exhibit similar risk behaviour?
  - Automated text analysis: are there documents that deal with the same topic?
- Understand structure of the data → assign labels to previously unlabelled data
- Build a possible point of departure for further analysis (for example, pre-processing for supervised learning)
- Reduce dimensionality of data → easier interpretation, reduce noise
- Visual Representation, e.g., in 2 or 3 dimensions

From High-Dimensional to Low-Dimensional

- 2D representation can be interpreted visually







Source: https://towardsdatascience.com/the-art-of-effective-visualization-of-multi-dimensional-data-6c7202990c57

Source: https://www.mql5.com/en/articles/283

Applications of Unsupervised Learning in Finance

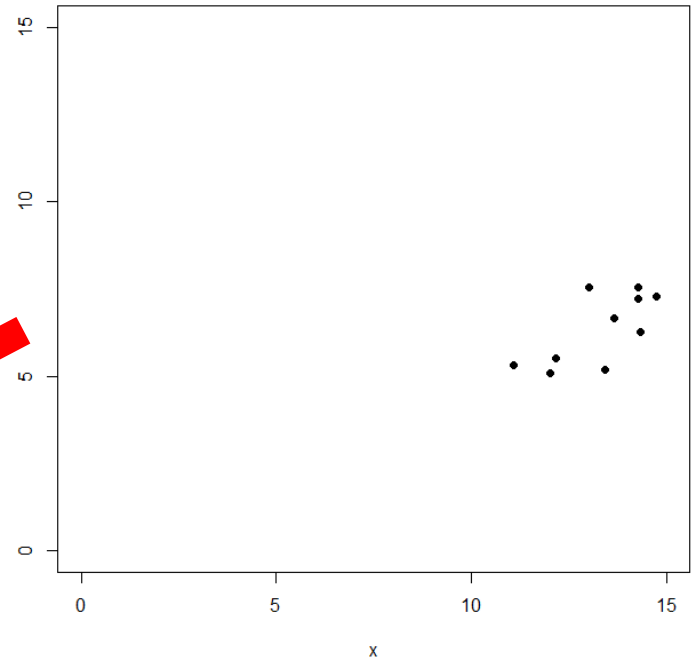
- Macro Forecasting:
  - Vectors containing economic variables define state of the economy or state of financial markets
- Portfolio Management
  - Asset Allocation: identify clusters of risk and diversify portfolio
  - Trading: relative value
- Natural Language Processing:
  - Identifying documents with similar content
- Return-based style classification of:
  - mutual funds, hedge funds: option-like returns
- Outlier / Fraud Detection
  - Vectors contain data on the last few transactions (e.g., $ amounts)
  - Large deviations are indicators of fraud

Principal Component Analysis

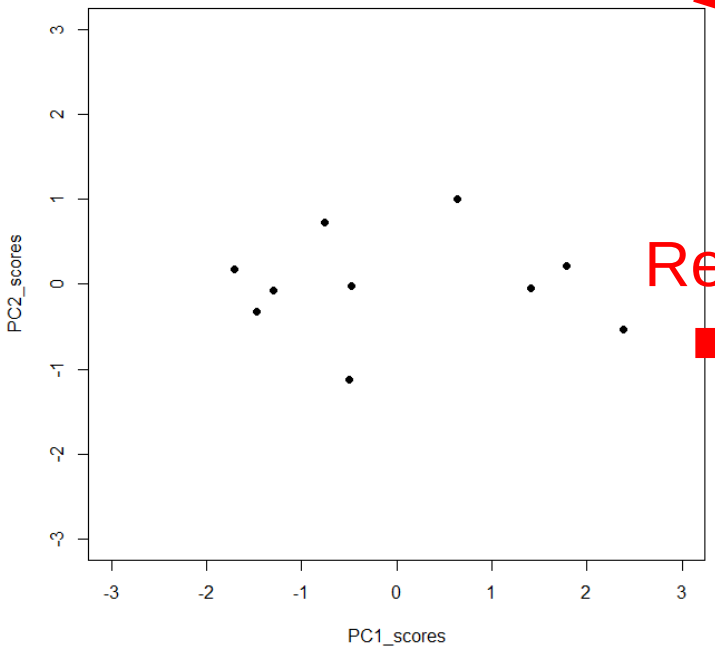
- PCA reduces dimensionality of the data
  - For example, PCA can reduce a variable system comprising 100 periods (rows) and 10 variables (columns or dimensions) from 100 x 10 to 100 x 3
- Principal components represent underlying structure in the data
  - Pre-process data as input for other ML methods
- Aggregate highly correlated variables to principal components
- PCA can be quickly performed with statistical packages (R, Python, Matlab, …)
- Applications (for example):
  - Image processing
  - Gene analysis
  - Relative value trading
  - Risk modelling

PCA: Idea

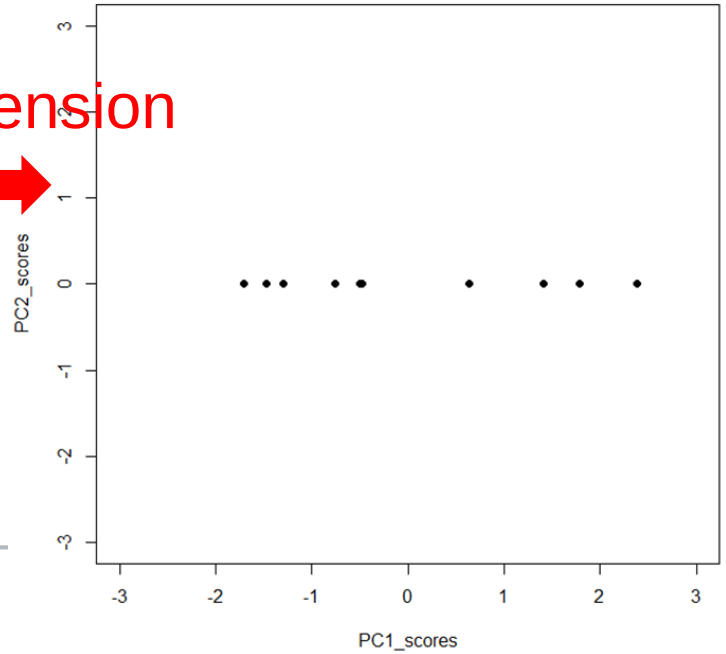
- 2 variables: 2 dimensions
- Highly correlated



Orthogonalise



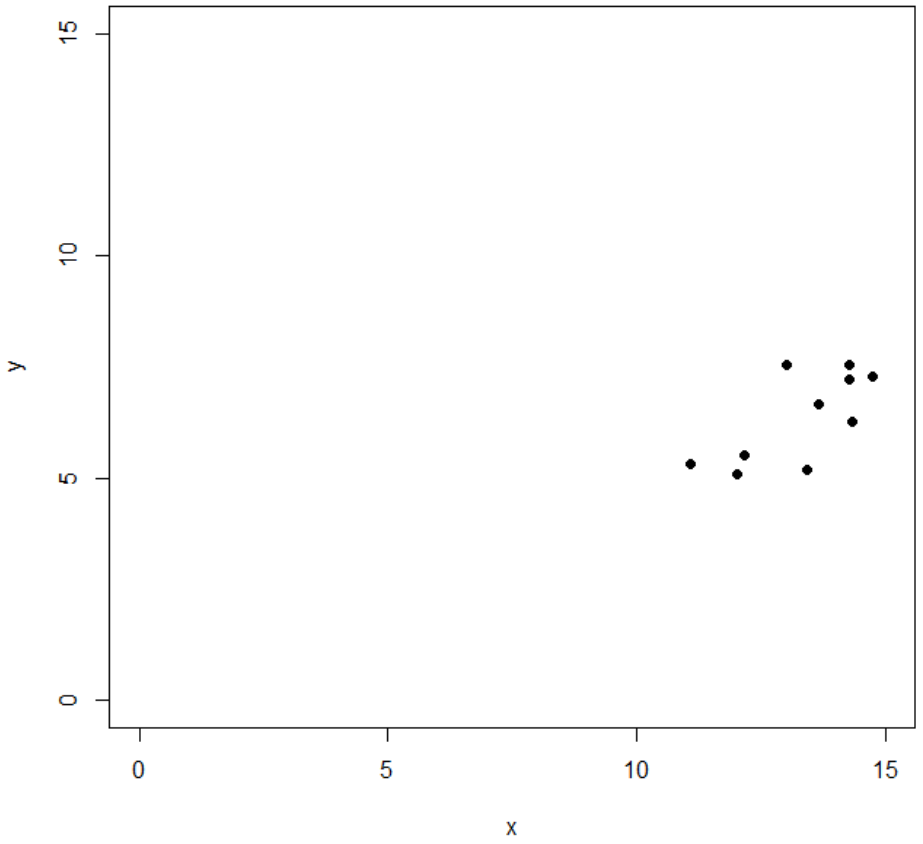
Reduce Dimension



PCA: Example

- 2 variables: 2 dimensions
- Highly correlated

	x	y
1	14.3	7.5
2	12.2	5.5
3	13.7	6.7
4	12.0	5.1
5	13.4	5.2
6	14.3	6.3
7	13.0	7.6
8	14.8	7.3
9	11.1	5.3
10	14.3	7.2



Cov	x	y
x	1.47	0.88
y	0.88	1.04

PCA: Example

1) Subtract the mean to centre the variables→

2) Extract Eigenvectors and Eigenvalues from covariance matrix (matrix algebra!!)

3) Multiply data by Eigenvector

	x_centred	
	x	y
1	0.99	1.13
2	1.11	-0.87
[illegible]	0.39	0.33
4	-1.31	-1.27
5	0.09	-1.17
6	0.99	-0.07
7	-0.31	1.23
8	1.49	0.93
9	-2.21	-1.07
10	0.99	0.83

	Eigenvalues	
	2.16	0.35
	Eigenvectors	
	PC1	PC2
x	-0.79	0.62
y	-0.62	-0.79

- 1st Eigenvector = 1st Principal Component

PCA: Eigenvectors

- The Eigenvectors describe how the variables of the system “fluctuate” together
  - Multiply the VCV of orig. data by Eigenvectors to get to transformed orthogonal data
- Eigenvectors are also called **Feature Vectors**
- The loadings (= elements of the Eigenvectors) can be understood as the weights for each original variable when calculating the principal components
  - Large loadings, i.e., absolute values of elements of the eigenvectors, have more relevance than elements with smaller absolute values
  - Interpret loadings like “beta factors” of the PCs: multiply the original data by those “beta factors” to get the uncorrelated factor scores
- The signs of the EV are arbitrary: an EV $\begin{pmatrix} -6 \\ 0 \\ 4 \end{pmatrix}$ has the same Eigenvalue as an EV $\begin{pmatrix} 6 \\ 0 \\ -4 \end{pmatrix}$ or an EV $c * \begin{pmatrix} 6 \\ 0 \\ -4 \end{pmatrix}$

Change signs of Eigenvectors to facilitate interpretation

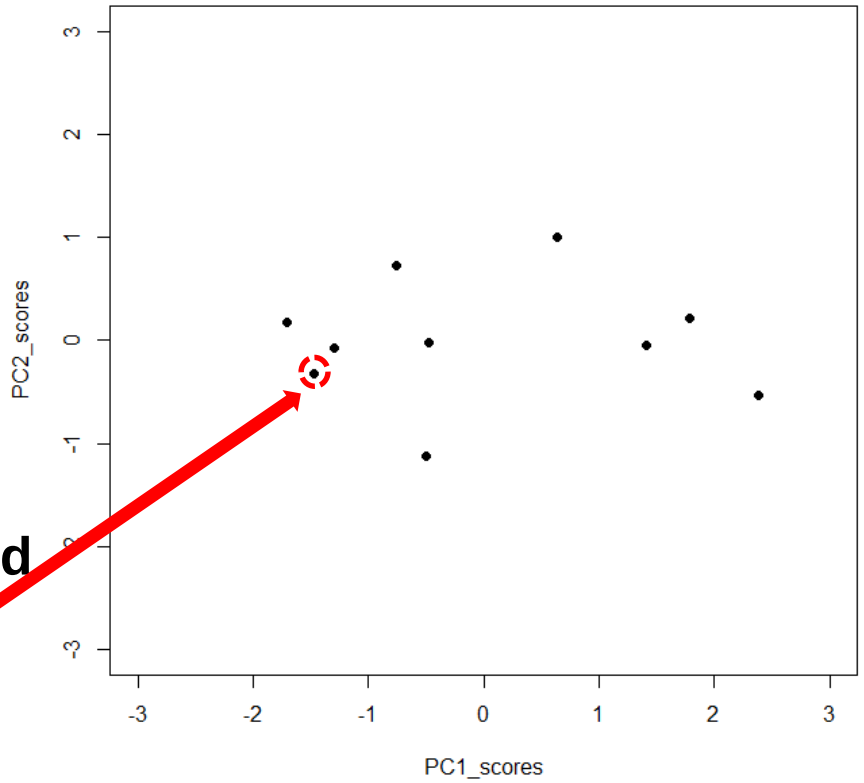
PCA: Transforming the Data to Factor Scores

- Use Eigenvectors to transform the data to Factor Scores
- 1st Factor Score of PC1:

  -0.79 * 0.99 + (-0.62)* 1.13 = -1.48
- 1st Factor Score of PC2:

  0.62 * 0.99 + (-0.79) * 1.13 = -0.28
- The Factor Scores are **uncorrelated**

	x_centred		Factor Scores	
	x	y	PC1	PC2
1	0.99	1.13	-1.48	-0.28
2	-1.11	-0.87	1.41	0.00
3	0.39	0.33	-0.51	-0.02
4	-1.31	-1.27	1.81	0.19
5	0.09	-1.17	0.65	0.98
6	0.99	-0.07	-0.74	0.67
7	-0.31	1.23	-0.51	-1.16
8	1.49	0.93	-1.75	0.19
9	-2.21	-1.07	2.40	-0.52
10	0.99	0.83	-1.29	-0.04



VCV of Factor Scores		
Cov	x	y
x	2.16	0.00
y	0.00	0.35

Factor Loadings

PCA: Reconstructing the Original Data

- We can use the factor scores to reconstruct the original variables:
- $\text{x\_scores} = \text{Eigenvectors}^T \times \text{x\_centred}$

[k x T] [k x N] [N x T]

T: # of observations
N: # of variables
k: # of retained PCs

- We can reconstruct the original variables:
- $\text{x\_centred'} = (\text{Eigenvectors}^T)^{-1} \times \text{x\_scores}$

[T x N] [N x k] [k x T]

- Given that $(\text{Eigenvectors}^T)^{-1} = \text{Eigenvectors}$:
- $\text{x\_centred'} = (\text{Eigenvectors} \times \text{x\_scores})^T$

[T x N] { [N x k] [k x T] }^{T}

PCA: Reconstructing the Original Data

- **With both PC1 & PC2:**
- x'(t)= PC1(x) * x_score(1,t) + PC2(x) * x_score(2, t)
- x'(t)= (-0.79) * (-1.48) + 0.62 * (-0.28) = 0.99
- y'(t)= (-0.62) * (-1.48) + (-0.79) * (-0.28) = 1.13
- We achieve exact replication, residuals are 0.

	Eigenvalues	
	2.16	0.35
	Eigenvectors	
	PC1	PC2
x	-0.79	0.62
y	-0.62	-0.79

- **With PC1 only:**
- x'(t)= (-0.79) * (-1.48) = 1.16
- y'(t)= (-0.62) * (-1.48) = 0.91
- Some information is lost, residuals are ≠ 0

PCA: Reconstructing the Original Data

- Using all PCs, residuals are 0
- If we use fewer PCs, residuals will be ≠ 0
- We can use the factor scores to reconstruct the original variables

	x_centred	
	x	y
1	0.99	1.13
2	-1.11	-0.87
3	0.39	0.33
4	-1.31	-1.27
5	0.09	-1.17
6	0.99	-0.07
7	-0.31	1.23
8	1.49	0.93
9	-2.21	-1.07
10	0.99	0.83

Factor Scores	
PC1	PC2
-1.48	-0.28
1.41	0.00
-0.51	-0.02
1.81	0.19
0.65	0.98
-0.74	0.67
-0.51	-1.16
-1.75	0.19
2.40	-0.52
-1.29	-0.04

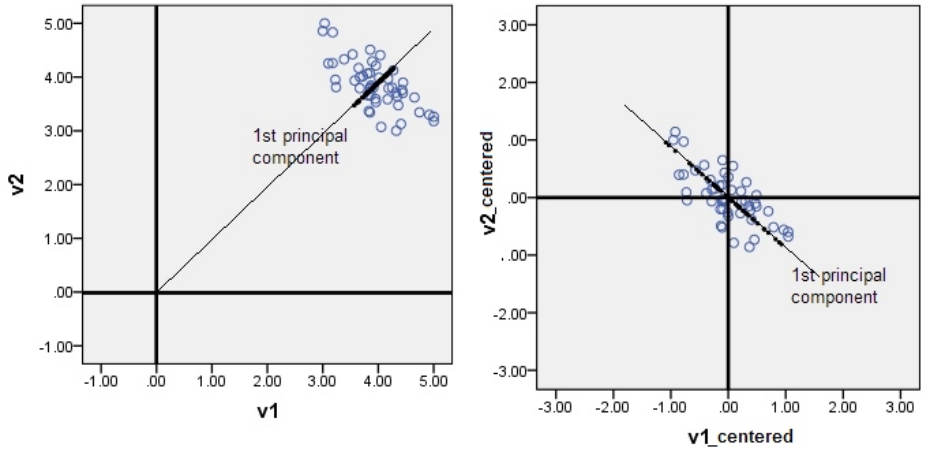
x_centred', reconstructed with PC1 & PC2			
		Residuals	
x'	y'	e(x)	e(y)
0.99	1.13	0.00	0.00
-1.11	-0.87	0.00	0.00
0.39	0.33	0.00	0.00
-1.31	-1.27	0.00	0.00
0.09	-1.17	0.00	0.00
0.99	-0.07	0.00	0.00
-0.31	1.23	0.00	0.00
1.49	0.93	0.00	0.00
-2.21	-1.07	0.00	0.00
0.99	0.83	0.00	0.00

x_centred', reconstructed with PC1 only			
		Residuals	
x'	y'	e(x)	e(y)
1.16	0.91	-0.17	0.22
-1.11	-0.87	0.00	0.00
0.40	0.31	-0.01	0.02
-1.43	-1.12	0.12	-0.15
-0.51	-0.40	0.60	-0.77
0.58	0.45	0.41	-0.52
0.40	0.32	-0.71	0.91
1.38	1.08	0.11	-0.15
-1.89	-1.48	-0.32	0.41
1.02	0.80	-0.03	0.03

- Factor scores are the original data in a rotated coordinate system

Why De-Meaning Before Applying PCA?

- If data is not de-meaned (left chart), PCA might not find the correct orientation of the PCs (Data in right chart is de-meaned)



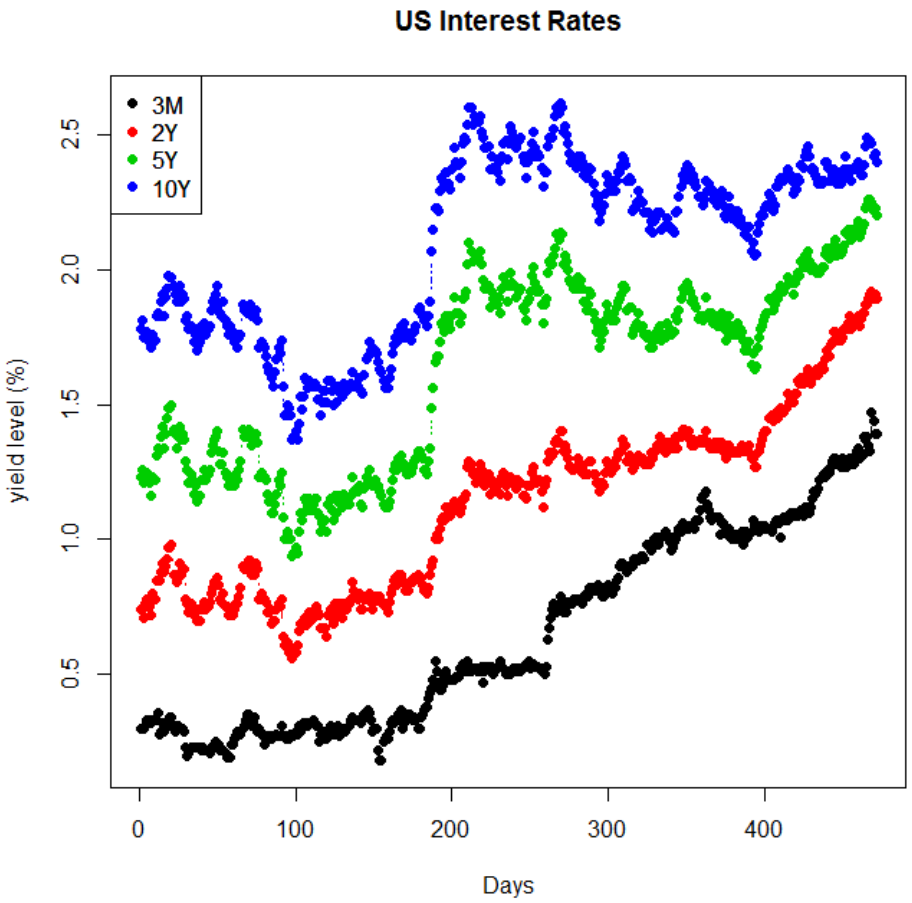
https://stats.stackexchange.com/questions/22329/how-does-centering-the-data-get-rid-of-the-intercept-in-regression-and-pca

https://stats.stackexchange.com/questions/189822/how-does-centering-make-a-difference-in-pca-for-svd-and-eigen-decomposition#189902

- If PCA is performed on VCV, de-meaning does not make a difference (usually the case with our applications)
  - Calculating covariance includes deducting the mean already: $\frac{1}{T-1}\sum_{t=1}^{T}(x_t - \bar{x})(y - \bar{y})$
- However, if PCA is performed on uncentred data via Singular Value Decomposition, results will be different for centred & uncentred data
- Try the 2 examples in R code “R PCA Demeaning.R”

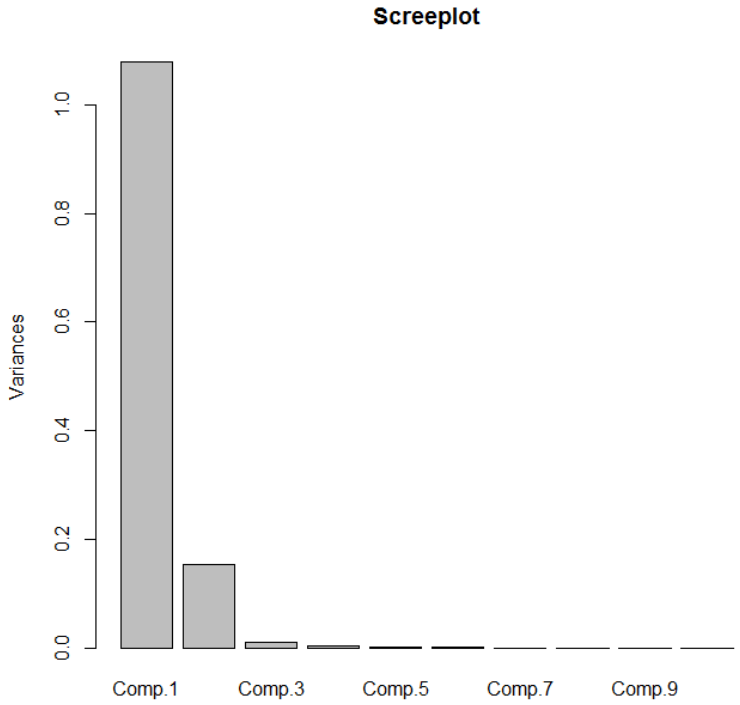
PCA: Example Interest Rates

- US interest rates (471 daily data points, 16 Feb 2016 – 29 Dec 2017, 471 days)
- Chart shows only 4 maturities
- Our data has 11 maturities: 1M, 3M, 6M, 12M, 2Y, 3Y, 5Y, 7Y, 10Y, 20Y, 30Y
  - Matrix 471 x 11
- Run PCA:
  - R: prcomp or princomp
  - Extract Eigenvectors and –values manually (exercise!)
  - See also Pelata et al. (2012), Redfern / MacLean (2014)



PCA: How many PCs to extract?

- Scree plot: plot Eigenvalues in descending order
- When do Eigenvalues level off? → This point determines the number of PCs (here: 2 or 3)
- Pro: simple graphical interpretation
- Con: subjective

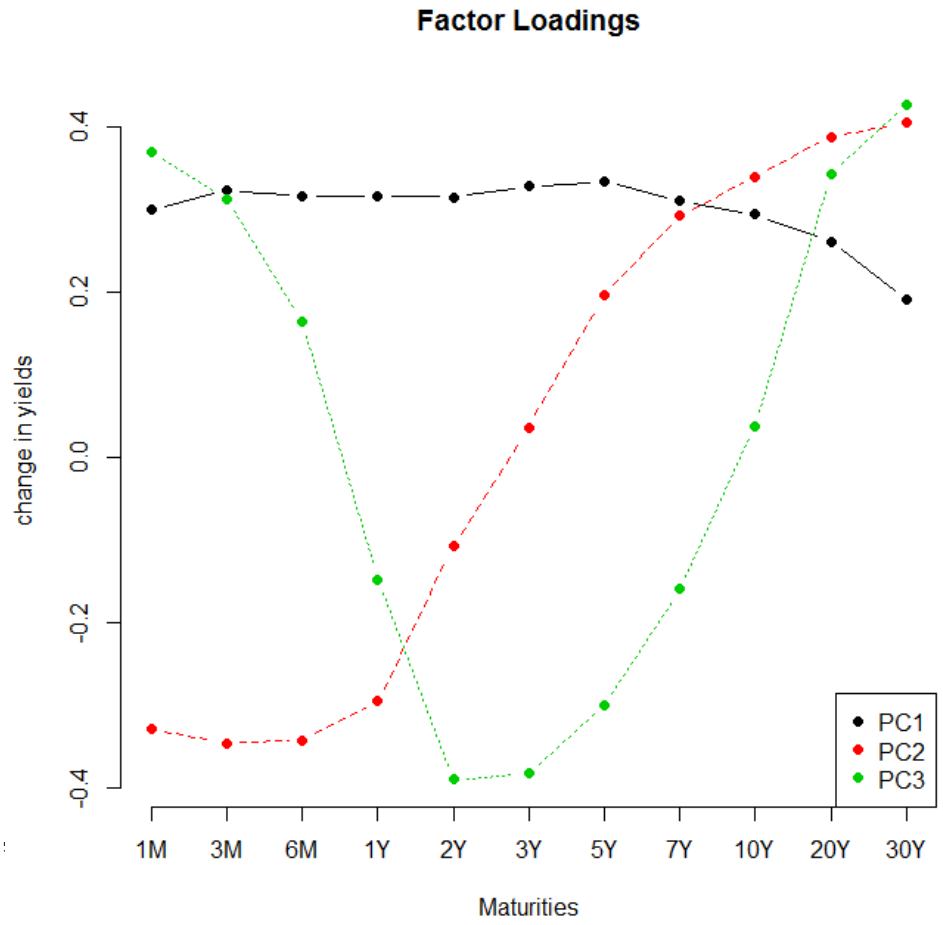


PCA: How many PCs to extract?

- **Alternatives**
- Kaiser Criterion: only retain PCs with Eigenvalues > 1 (here: PC1 only)
  - Rationale: scaled variables (mean 0, variance 1) exhibit on average a variance of 1 → only take those PCs that contribute above average to explanation of total variance
- Variance Explained Criterion: keep as many PCs as required to explain X% of variance (e.g., 90%)
  - PC1 only explains 87% of variance, PC1 to PC2 explain 99%, PC1 – PC3 explain 100%

PCA: Loadings

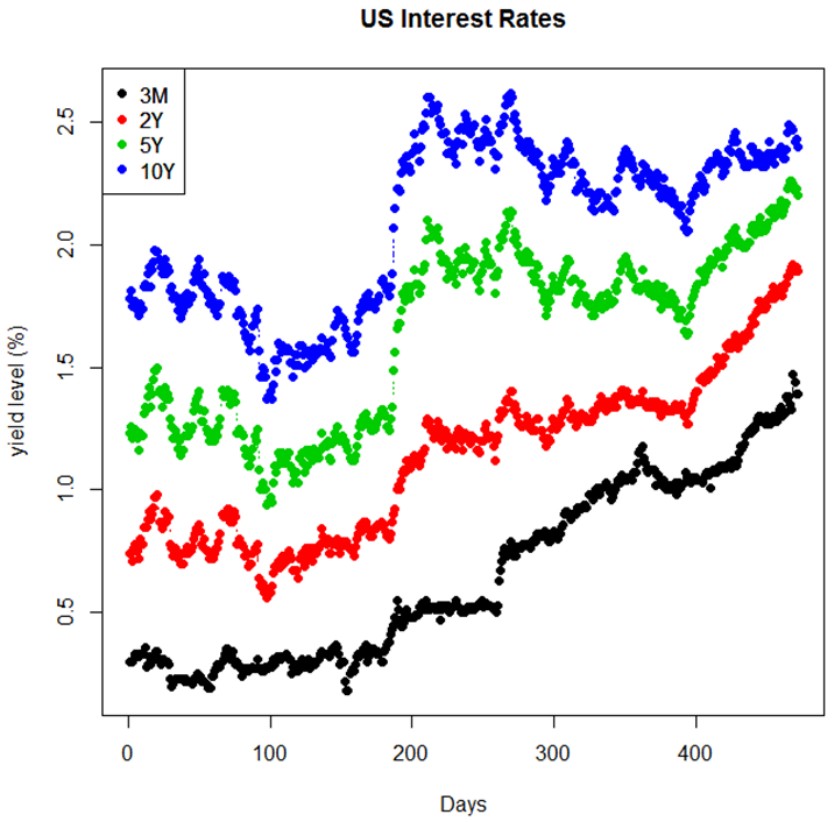
- We use the first 3 Eigenvectors
- Each Eigenvector has 11 elements
- Chart shows Eigenvectors' loadings (i.e., elements of the Eigenvectors)
  - PC1: If IR↑ then PC1↑ and v.v. → Duration
  - PC2: short maturities: if IR↑ then PC2↓, long maturities: if IR↑ then PC2↑ → Twist (slope of yield curve)
  - PC3: short & long maturities: if IR↑ then PC3↑, mid-term mat.: if IR↑ then PC3↓ → Curvature



Large loadings, i.e., absolute values of elements of the eigenvectors, have more relevance than elements with smaller absolute values

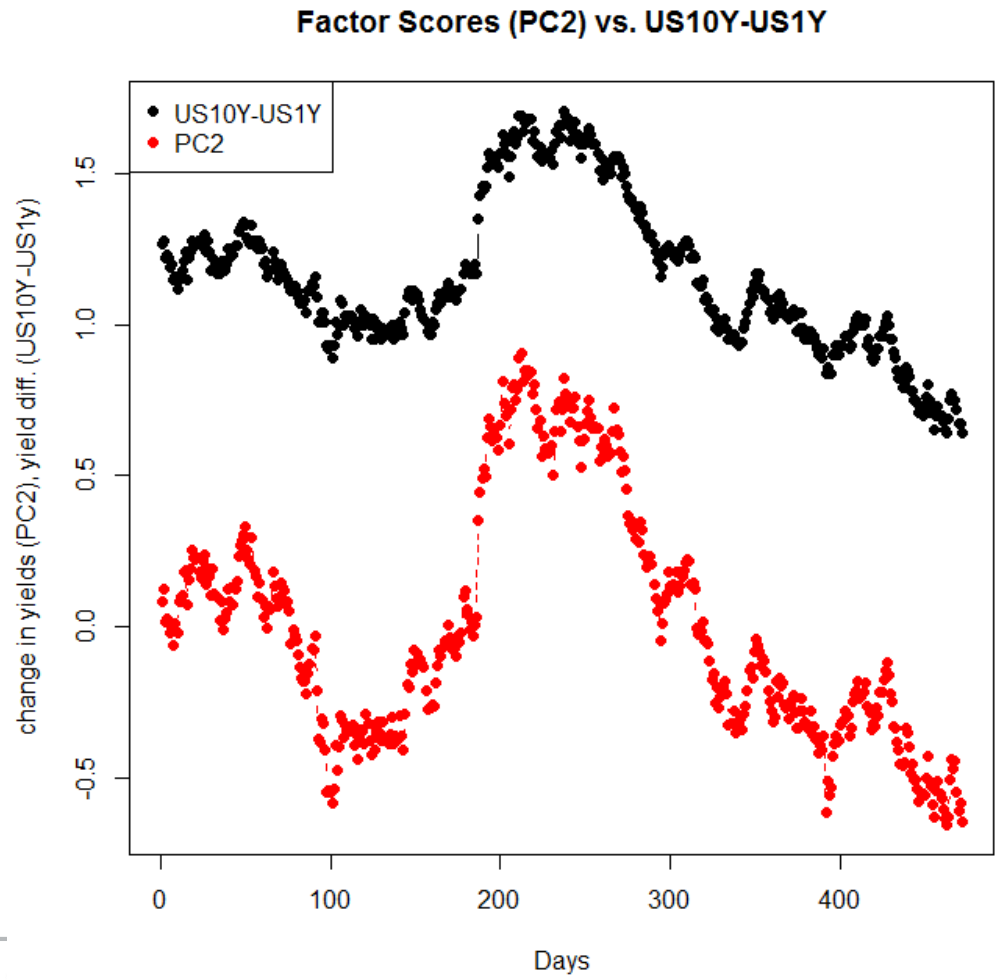
PCA: Factor Scores Over Time

- Orig. data 471 x 11: PCA transforms to 471 x 3 (left chart)
- IR increased (PC1), slope first steepened, then flattened (PC2), curvature virtually unchanged (PC3)



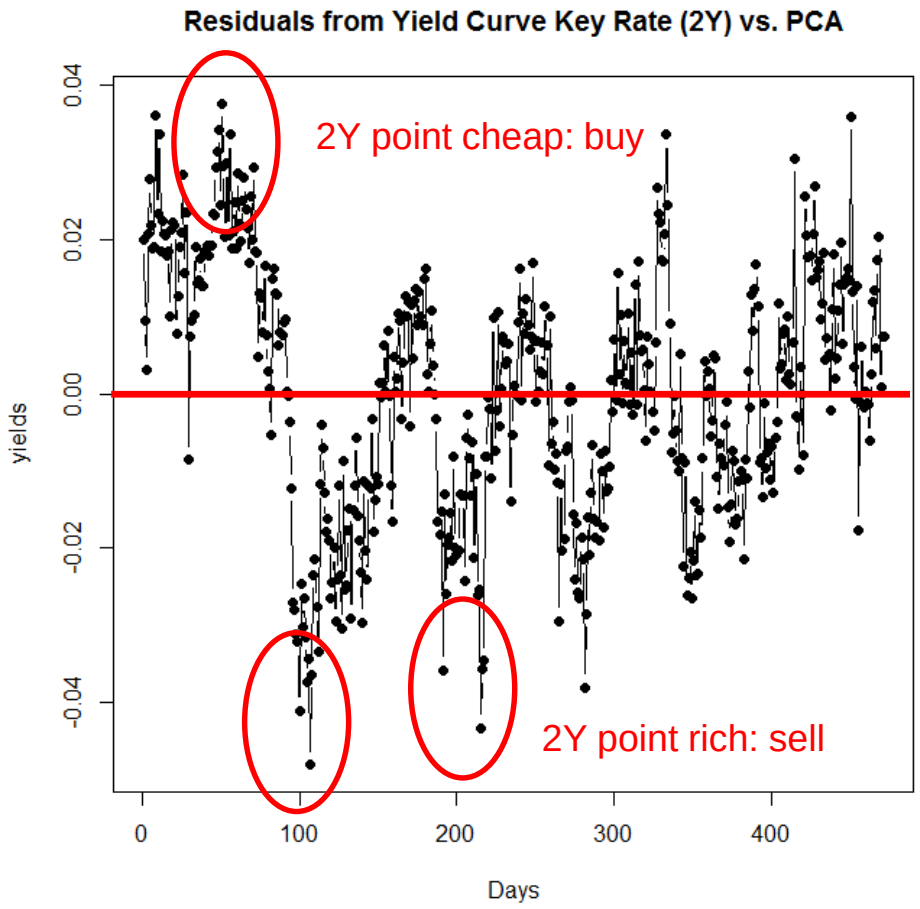
PCA: Factor Scores Over Time

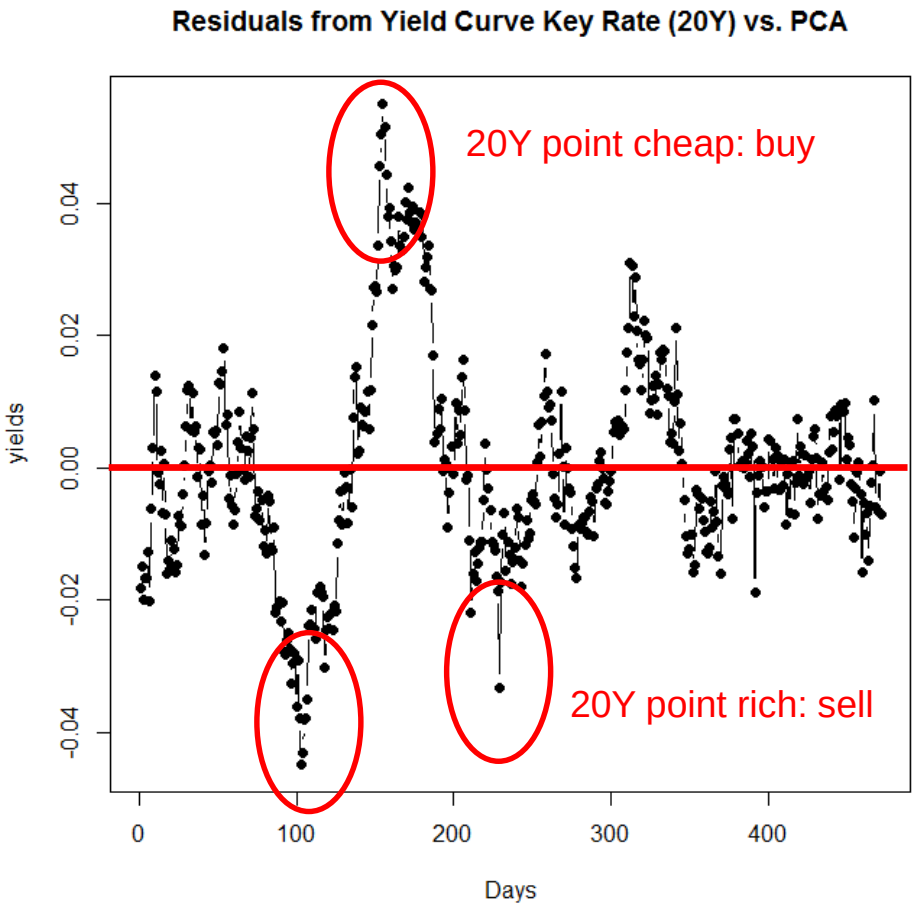
- Slope of the Yield Curve: compare PC2 with yield differential US10Y – US1Y



PCA: Analysis of Residuals

- Entry points for trades?
- Residual > 0: 20 Y Key Rate yields higher than implied by PCA → initiate long position, e.g., short 10Y, long 20Y, short 30Y (duration-weighted)





PCA: Further Applications

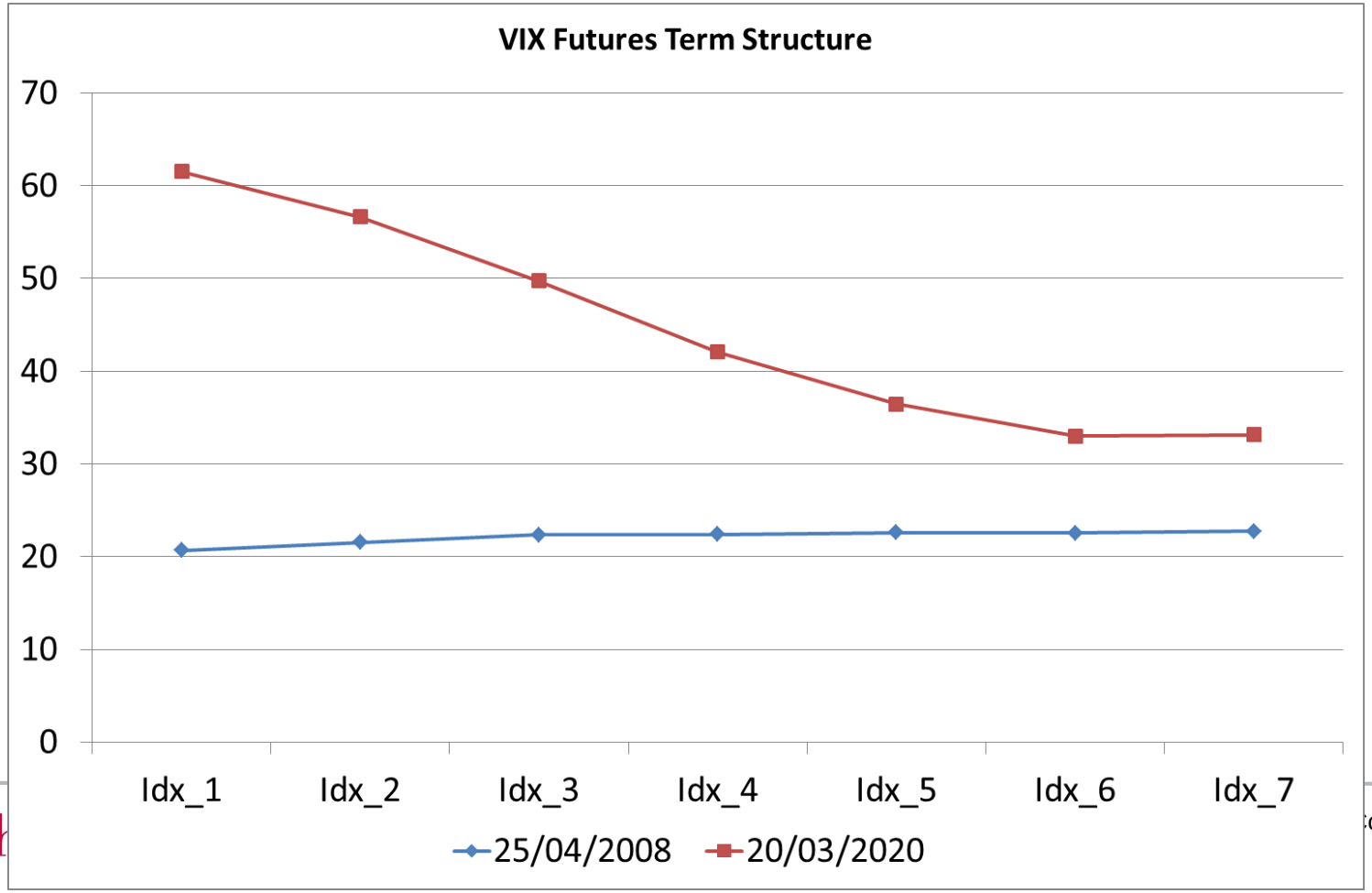
- Factor Scores are “lean” representatives of the original variable system → less noise
  - Use Factor Scores to forecast: for example, apply AR-models
  - Macro-econometric forecasting: rather than using several dozen variables, only use 6 factors (e.g., Artis et al. (2005))
- Value-at-Risk: use PCs to model risk factors
- Use Loadings to calculate hedge ratios to immunise portfolios against:

a) parallel shifts only

b) parallel shifts AND changes in slope

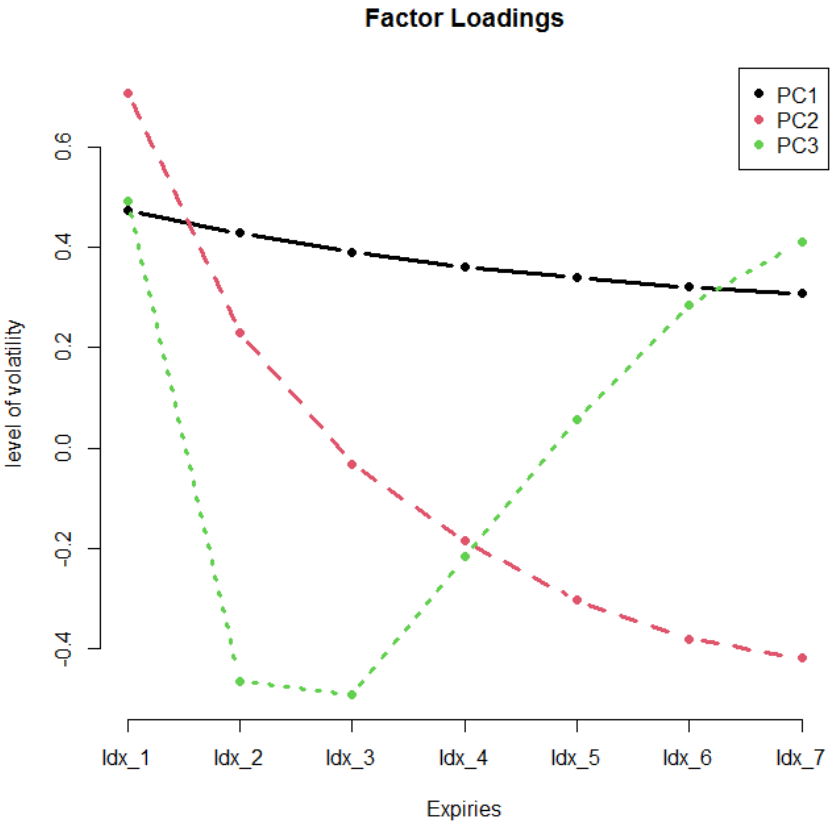
PCA Applied to the VIX Future Term Structure

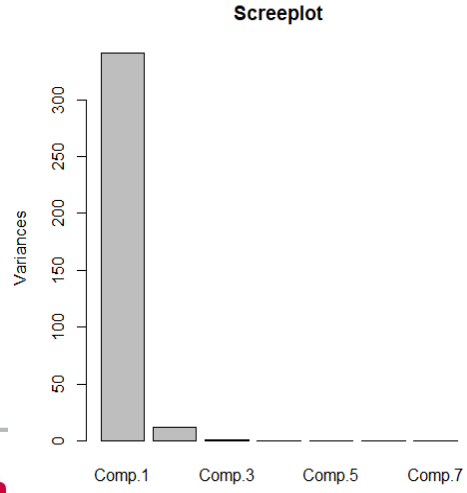
- Inputs: first 7 expiries, weekly volatility levels from April 2008 to June 2020: data set has 636 x 7 data points



PCA Applied to the VIX Future Term Structure

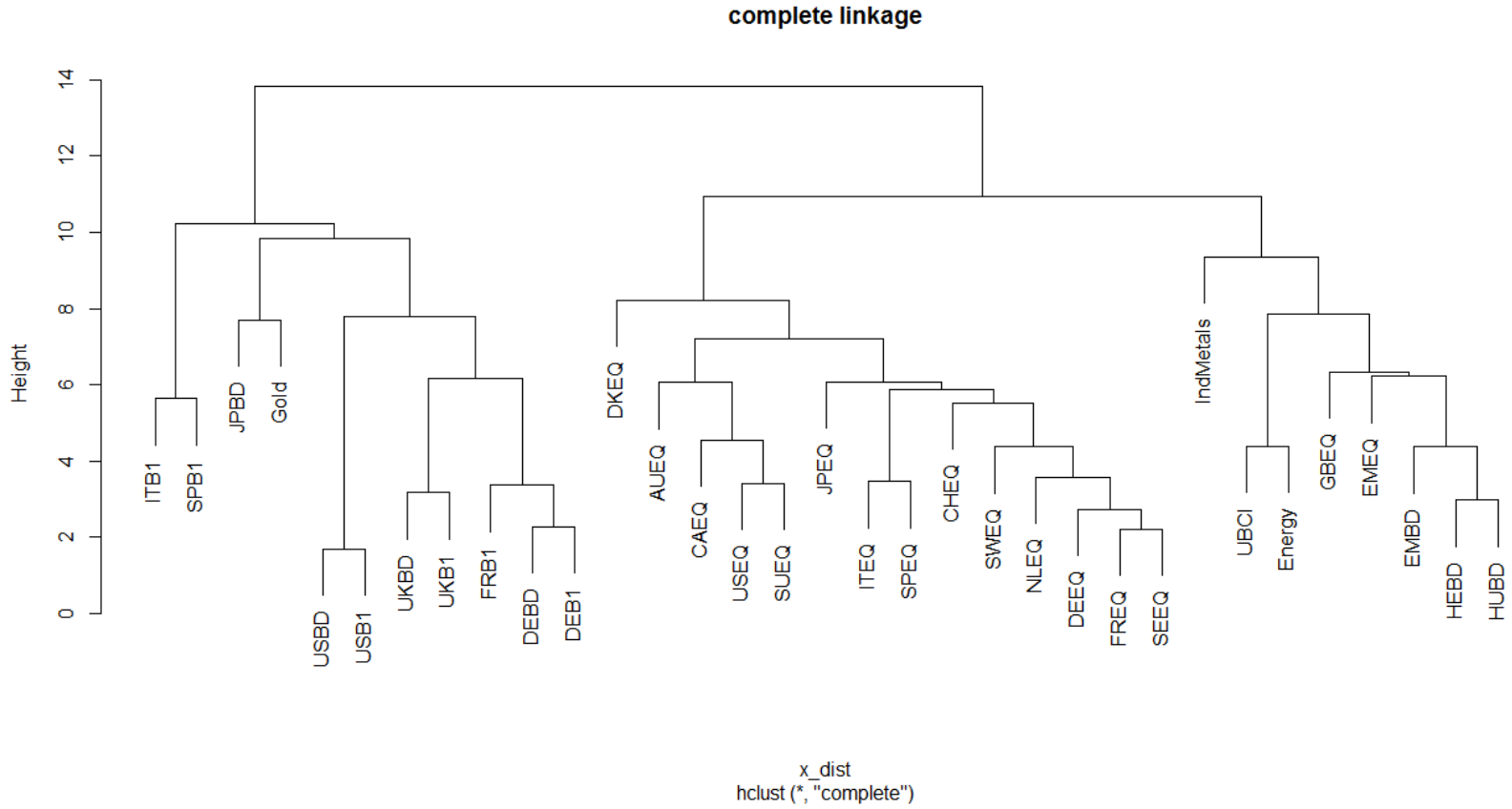
- The first 2 PCs explain almost 100% of the variance
- PC2 ***upward*** shaped
- PCs similar to interest rates: level, slope, curvature
- Similar picture as with other term structures (interest rates, commodities): max. 3 PCs





Hierarchical Clustering: Asset Allocation

- Time series analysis: 33 financial variables, monthly Apr 2015 to Mar 2020 (60 rows x 33 columns)
- Which assets belong together, which ones are outliers (interesting for diversification!)
- **Dendrogram:**



Hierarchical Clustering

- Agglomerative: start with individual data points to merge until only 1 large cluster remains (bottom-up)
  - Most frequently used
- Divisive: (top-down)

- How do we define proximity between **clusters**?
- Distance between 2 **data points** can be expressed by Sum of Squared Errors, Euclidean Distance

Distance Measures: Sum of Squared Errors (SSE) and Euclidean Distance (ED)

$$\text{SSE} = \sum_{t=1}^{T} \left(x_{i,t} - x_{j,t} \right)^2$$

- $x_{i,t}$: characteristics of the samples, e.g., $ amount of transaction at time t (T is number of observation points)
- The lower the SSE, the more similar are 2 samples

$$\text{ED} = \sqrt{\sum_{t=1}^{T} \left(x_{i,t} - x_{j,t} \right)^2}$$

- SSE squares the deviations and hence weighs differences more heavily than ED
- SSE has well-defined derivatives: useful for optimisation

Hierarchical Agglomerative Clustering: Process

- Process of Agglomerative Clustering:

1. Set each data item to be a single cluster
2. Calculate distance measure
3. Merge the two closest clusters
4. Update the distance matrix
5. Go to 3 until only 1 cluster remains

- Expensive in terms of computing time
- No random seed required, results are deterministic and hence reproducible: multiple runs give the same result

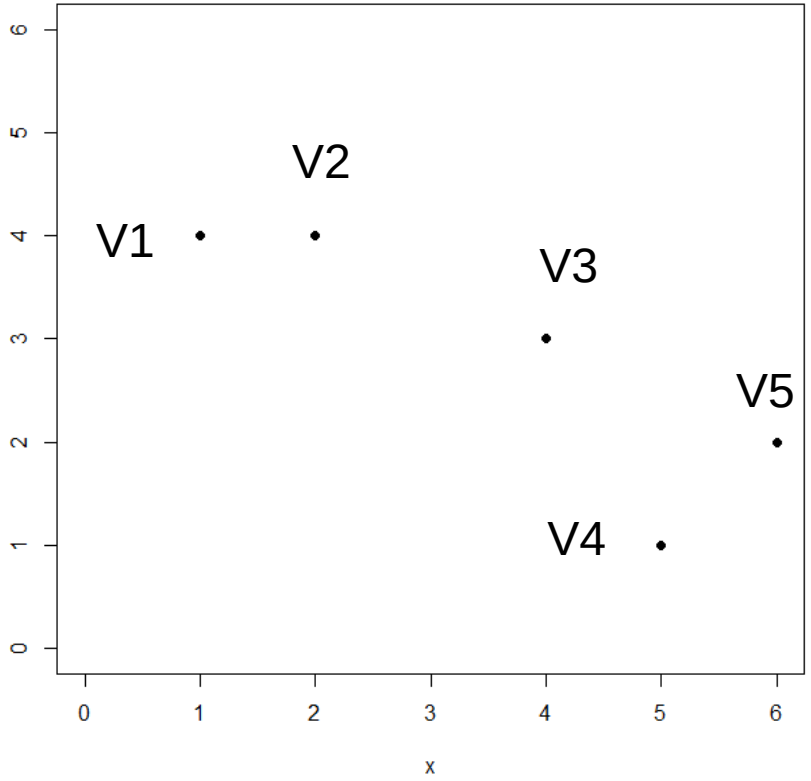
Hierarchical Agglomerative Clustering: Example & Distance Matrix

- 5 data points in 2-dimensional space
- Distance Matrix shows **Euclidean Distances**
- For example, (V1, V2):

$$= \sqrt{(1-2)^2 + (4-4)^2} = 1.0$$

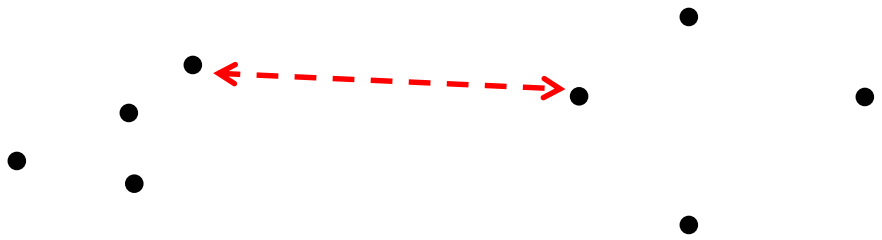
	V1	V2	V3	V4	V5
X	1	2	4	5	6
Y	4	4	3	1	2

Distances	V1	V2	V3	V4	V5
V1	0				
V2	1.00	0			
V3	3.16	2.24	0		
V4	5.00	4.24	2.24	0	
V5	5.39	4.47	2.24	1.41	0



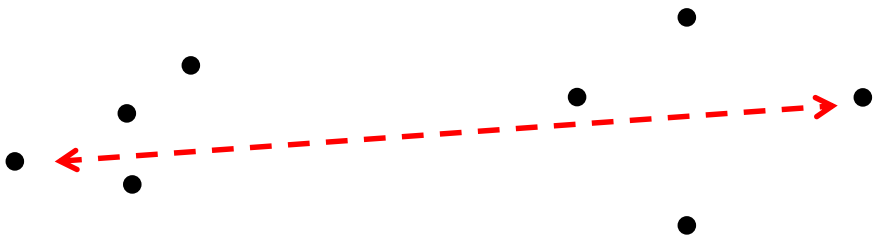
Hierarchical Agglomerative Clustering

- After the 2 most similar variables were identified, we need to update the Distance Matrix
- 3 methods are common:
  a) Single Link or MIN,
  b) Complete Linkage or MAX,
  c) Group Average Distance or Group Average Linkage
- Single Link or MIN: the smallest distance between 2 data points assigned to different clusters



Hierarchical Agglomerative Clustering

- Complete Linkage or MAX: distance of two clusters is based on the two most distant points in the different clusters
- Determined by all pairs of points in the two clusters



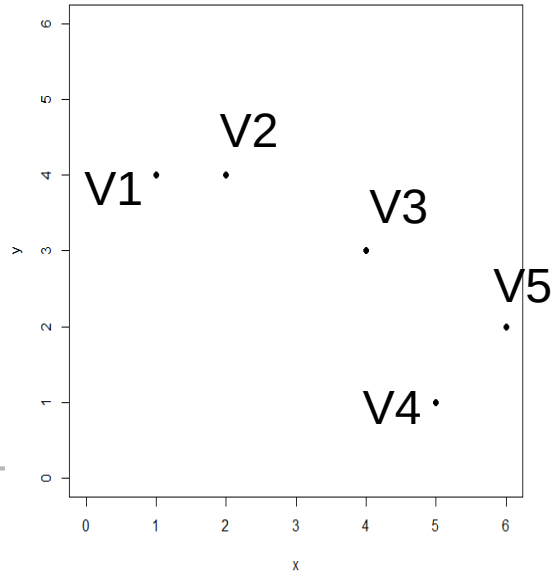
Example: Single Linkage

	V1	V2	V3	V4	V5
X	1	2	4	5	6
Y	4	4	3	1	2

Distances	V1	V2	V3	V4	V5
V1	0				
V2	1.00	0			
V3	3.16	2.24	0		
V4	5.00	4.24	2.24	0	
V5	5.39	4.47	2.24	1.41	0

- First step identical for Single and Complete Linkage: merge V1 and V2
- 2nd step, Single Linkage: find min of each (V1 and V2) and other variables
- Search for min in the highlighted cells, row by row:
- V3: min(3.16, 2.24) = 2.24
- V4: min(5.00, 4.24) = 4.24
- V5: min(5.39, 4.47) = 4.47

Distances	V1	V2	V3	V4	V5
V1	0				
V2	1.00	0			
V3	3.16	2.24	0		
V4	5.00	4.24	2.24	0	
V5	5.39	4.47	2.24	1.41	0

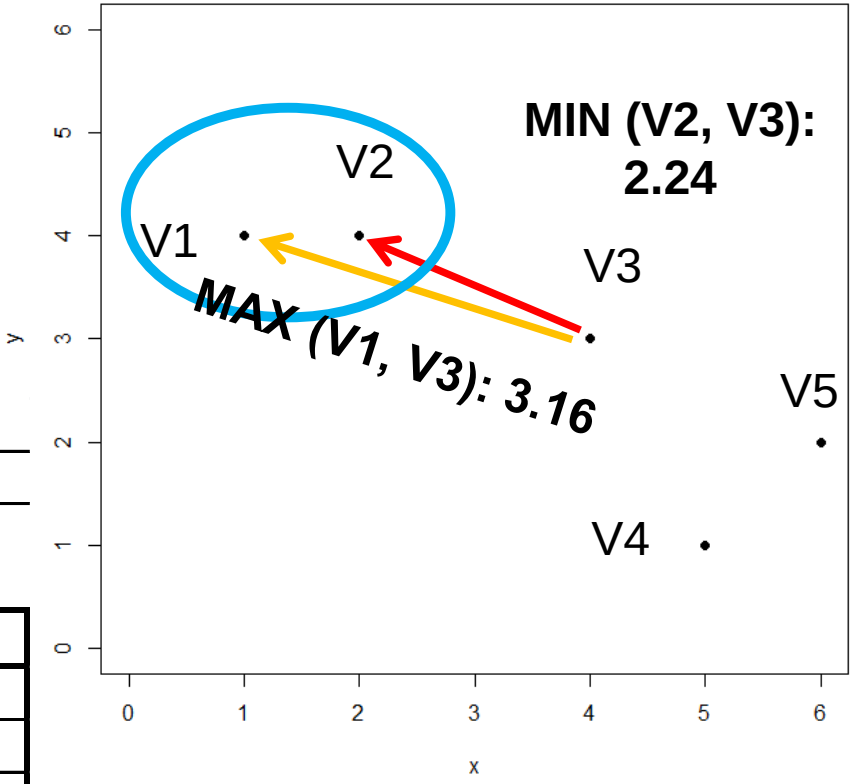


Hierarchical Agglomerative Clustering: Example & Distance Matrix

- Single Linkage (MIN, red arrow):
  - (V1, V2) to V3: min(3.16, 2.24) = 2.24
- Complete Linkage (MAX, orange arrow):
  - (V1, V2) to V3: max(3.16, 2.24) = 3.16

	V1	V2	V3	V4	V5
X	1	2	4	5	6
Y	4	4	3	1	2

Distances	V1	V2	V3	V4	V5
V1	0				
V2	1.00	0			
V3	3.16	2.24	0		
V4	5.00	4.24	2.24	0	
V5	5.39	4.47	2.24	1.41	0



Example: Single Linkage

- Distance matrix is updated with the cluster (V1, V2)
- Next merge V4 and V5

DM updated

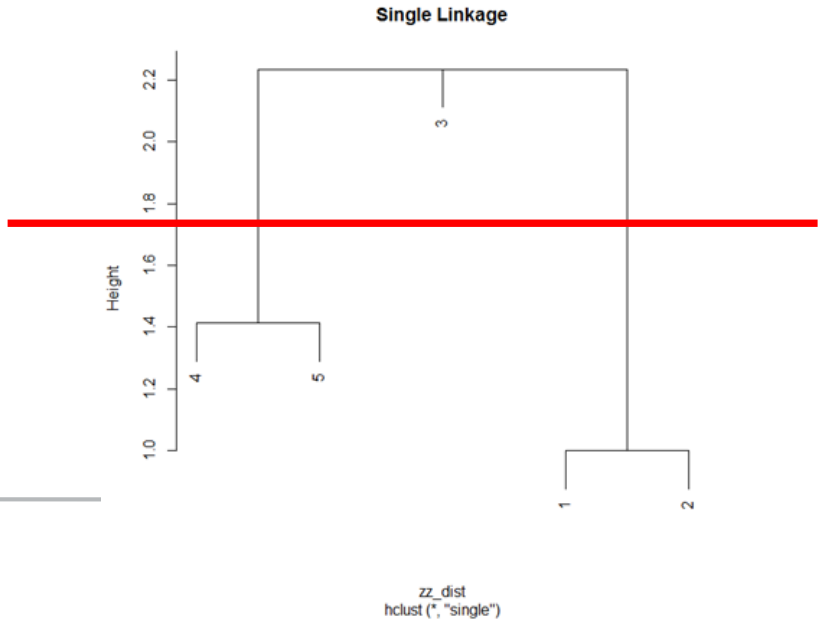
Distances	V1, V2	V3	V4	V5
V1, V2	0			
V3	2.24	0		
V4	4.24	2.24	0	
V5	4.47	2.24	1.41	0

Distances	V1, V2	V3	V4, V5
V1, V2	0		
V3	2.24	0	
V4, V5	4.24	2.24	0

- (V1, V2), V3: min(2.24) = 2.24
- (V1, V2), (V4, V5): min(2.24, 4.24) = 2.24

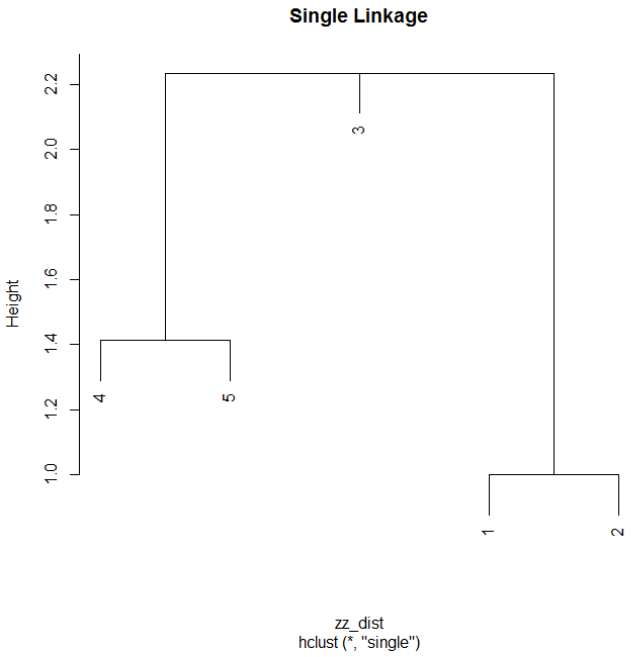
→ V3 has lowest distance to both (V1, V2) and (V4, V5)

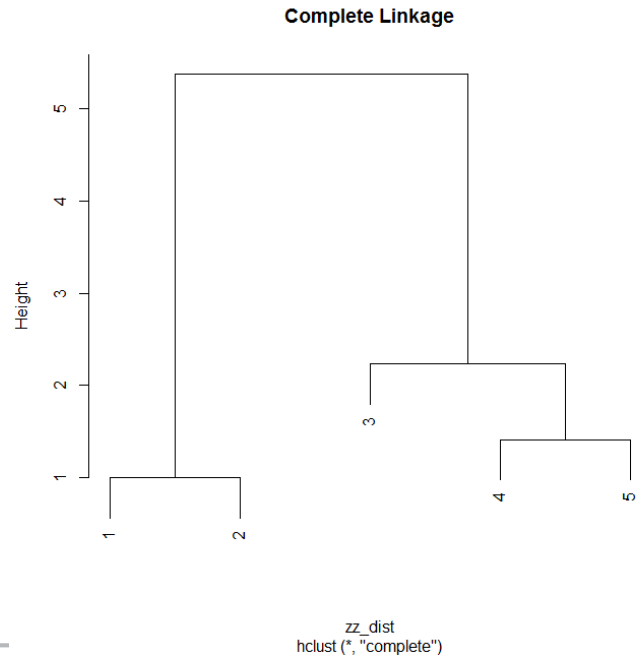
→ V3 merges (V1, V2) and (V4, V5)



Hierarchical Agglomerative Clustering

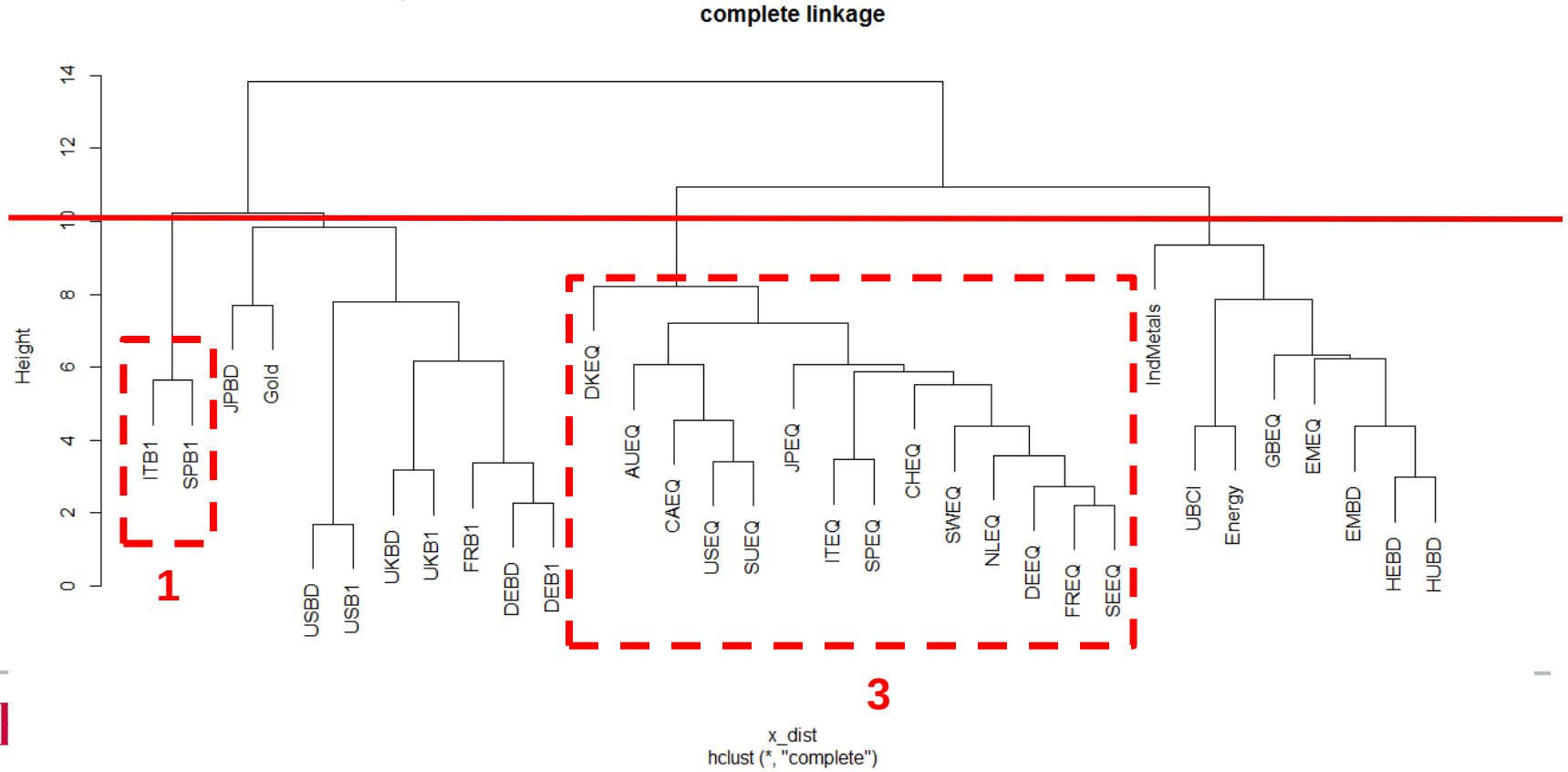
- Single combines (1, 2) first, then (4, 5). Last step 3 is to merge (1, 2) and (4, 5)
- Complete also starts with (1, 2) and (4, 5), but then merges 3 with (4, 5)





Hierarchical Clustering: Asset Allocation

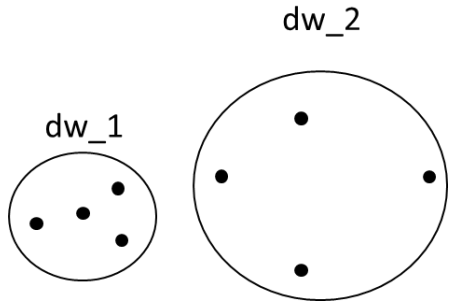
- Time series analysis: 33 financial variables, monthly Apr 2015 to Mar 2020 (60 rows x 33 columns)
- Shape of the dendrogram used for assigning asset weights (Raffinot (2018)): no forecasts required
- Each cluster gets equal weight: 4 clusters → 25% weight each
- Example Cluster 1 (Italian and Spanish government bonds): 2 assets
  → 25% / 2 = 12.5% weight each



Hierarchical Clustering: Optimal Number of Clusters

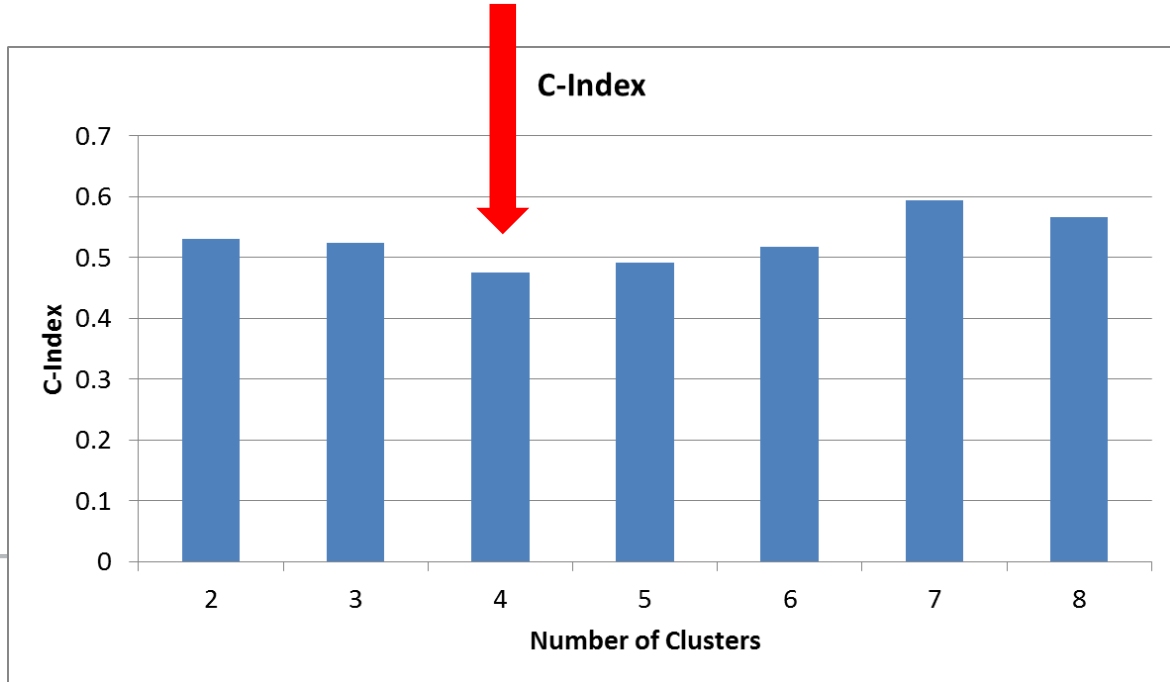
- There are many criteria to determine the optimal number of clusters (Milligan / Cooper (1985))
  - Criteria sometimes difficult to calculate → R NbClust helps
- For example: C-Index (Hubert / Levin (1976)):
  - [dw - min(dw)] / [max(dw) - min(dw)]
  - dw is the sum of the within cluster distances
  - max(dw) ≠ min(dw)
  - 0 ≤ C-index ≤ 1
- Calculate C-Index for each hierarchy level
- The minimum C-Index determines the optimal number of clusters

dw_1 < dw_2



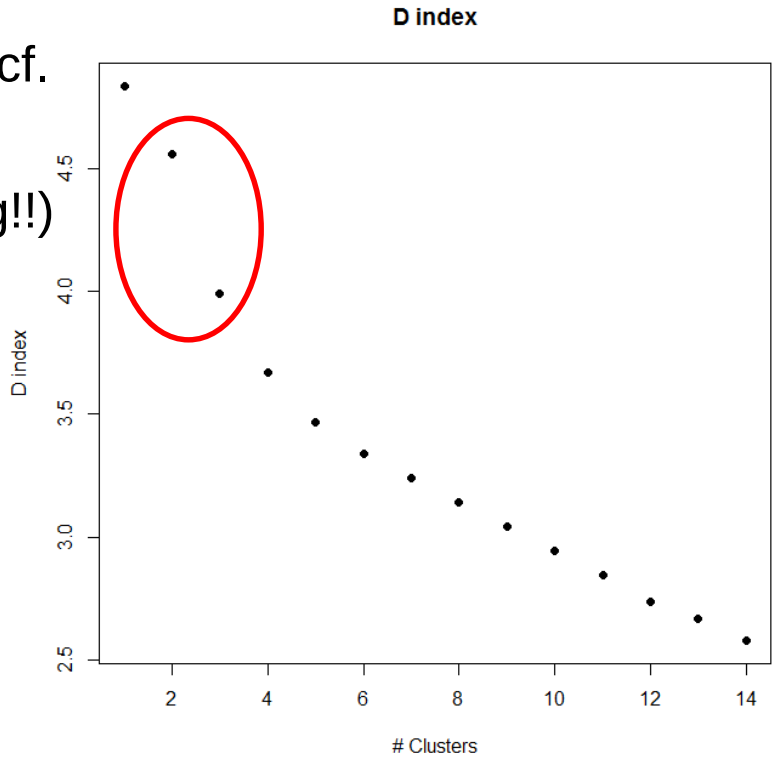
Example

For our Asset Allocation Example:



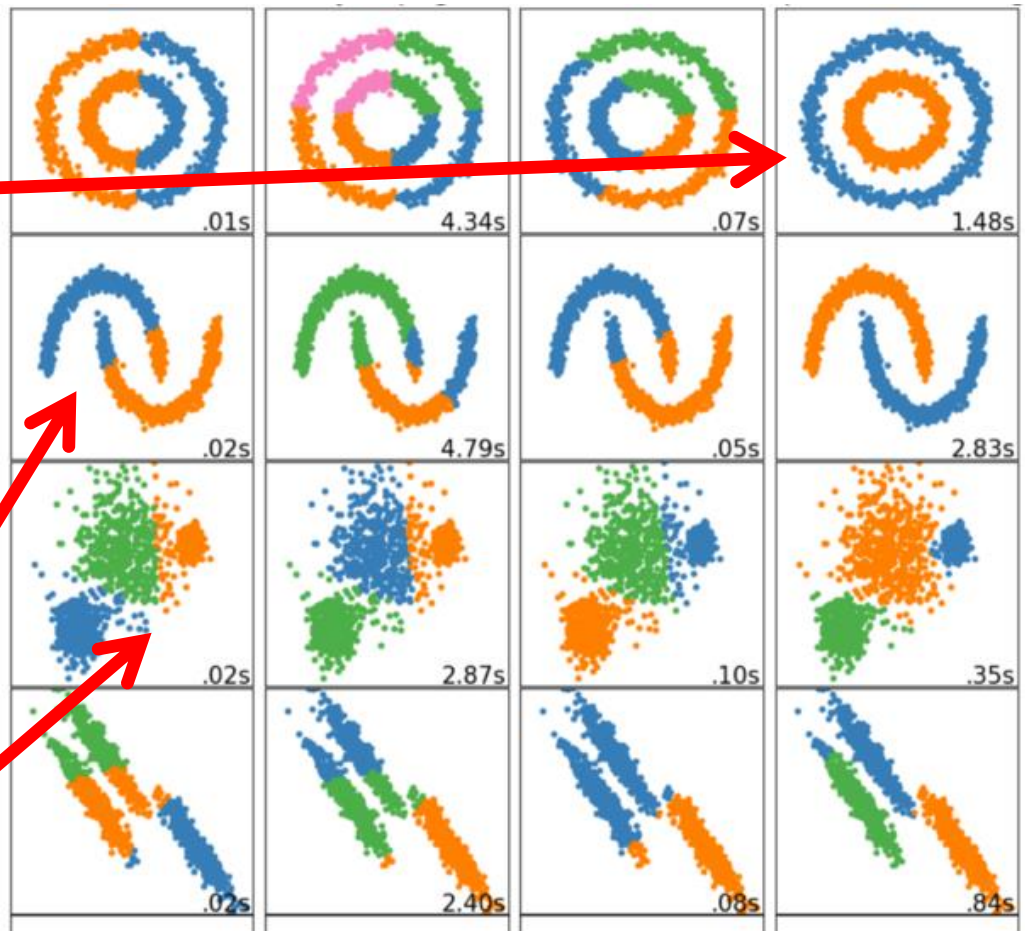
Hierarchical Clustering: Optimal Number of Clusters

- By graphical inspection (subjective)
- R function NbClust: calculate 30 indices (cf. Milligan / Cooper (1985)) and take the number most indices choose (data mining!!)
- Example D-index: choose # clusters with largest gap → 3
- There is no clear analytical answer: sometimes a clustering may be mathematically good, but economically useless
- If in doubt, choose the model that allows for the clearest economic interpretation



Hierarchical Clustering: Features

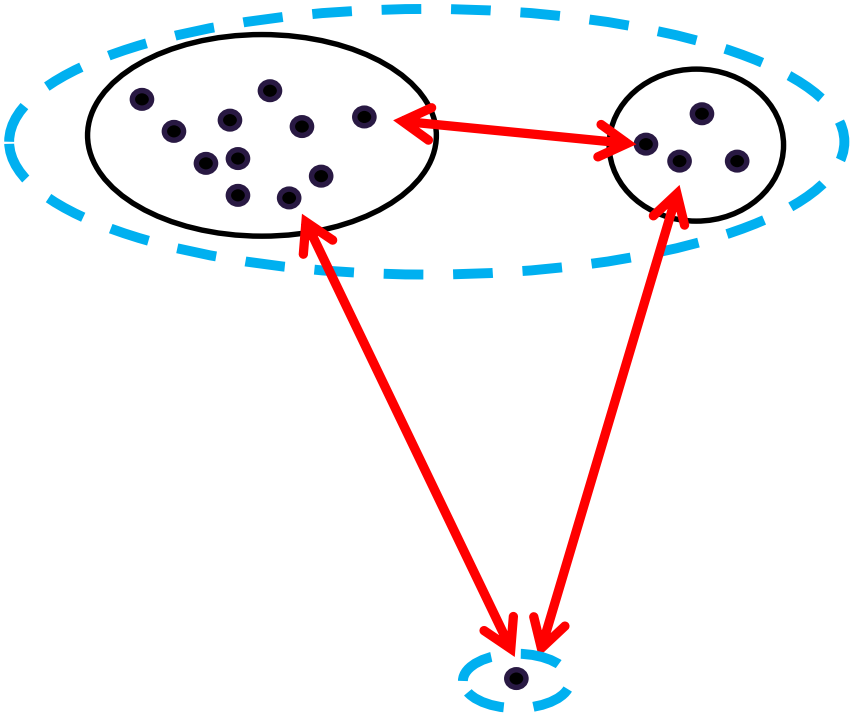
- Features of Single Linkage / MIN:
  - Pro: can handle non-elliptical shapes
  - Con: Sensitive to noise and outliers
- Features of Complete Linkage / MAX:
  - Pro: less sensitive to noise and outliers
  - Con: Biased towards globular clusters, tends to break large clusters



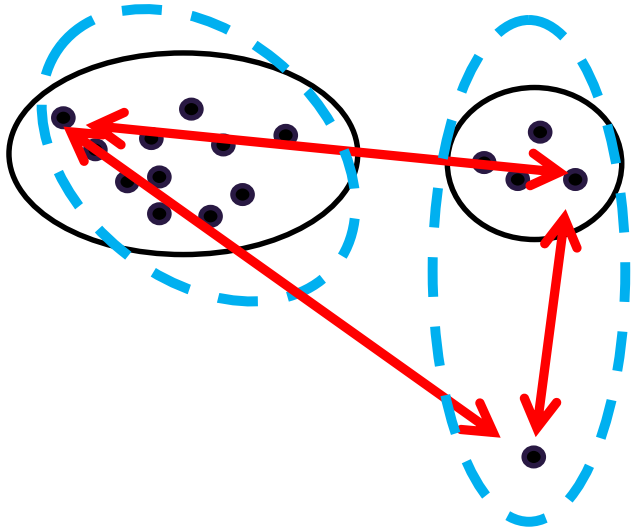
Source: https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68

Hierarchical Clustering: Features

- MIN is sensitive to outliers

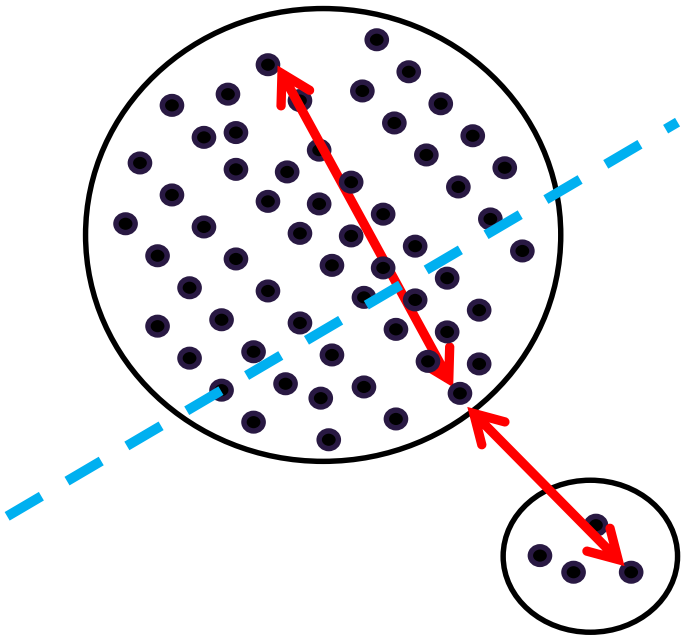


- MAX less sensitive



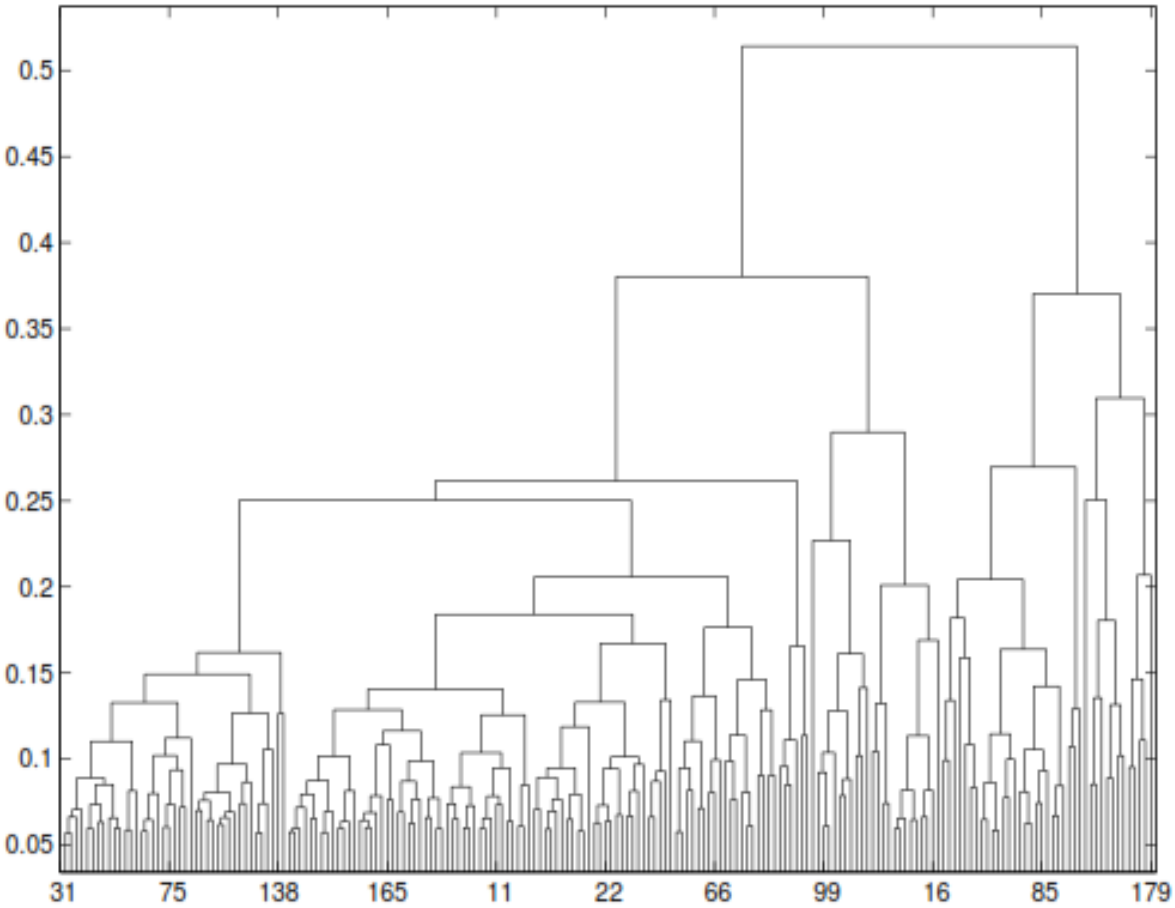
Hierarchical Clustering: Features

- MAX tends to break large clusters



Hierarchical Clustering: Index Tracking

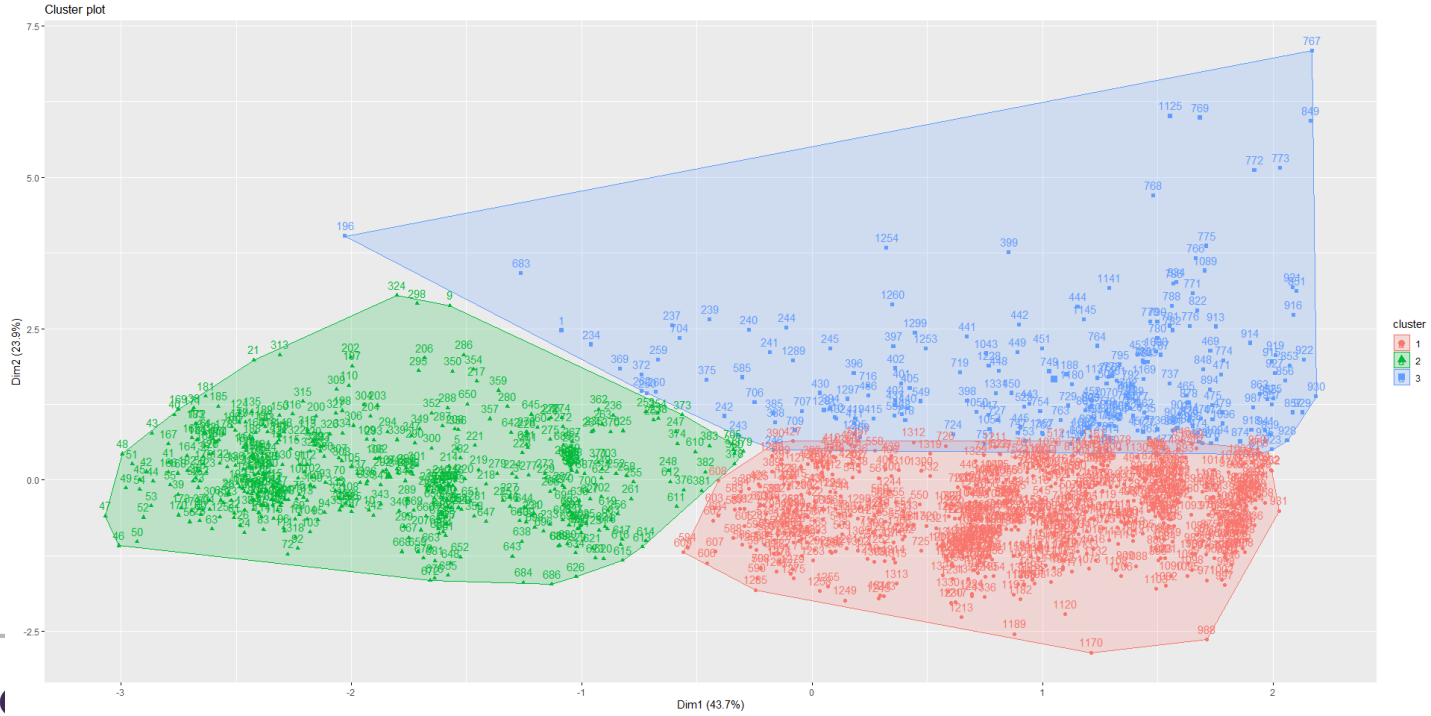
- Dose / Cincotti (2005)):
  - Track index (SPX) with as few stocks as possible
  - Identify clusters among the index constituents
  - Pick 1 stock from each cluster
  - Minimise tracking error of stock portfolio to index
  - 50 stocks used to track SPX



(a)

k-Means Clustering

- Are there similar market environments / phases?
- Find k clusters (=market phases) in the data
- 5 variables: 10Y, 10Y-2Y, VIX, dVIX, cor(SPX, US10Y)
- Weekly data from Feb94 to Feb20 (time series with 1359 X 5 data points)



k-Means Clustering

- Simple methodology
- The method focuses on centroids: each cluster defines 1 prototype
- Each data point is assigned to one cluster only
- Find prototypes that represent the data: find centroids that minimise the intra-class variance (SSE)
- Number of clusters k has to be provided by user

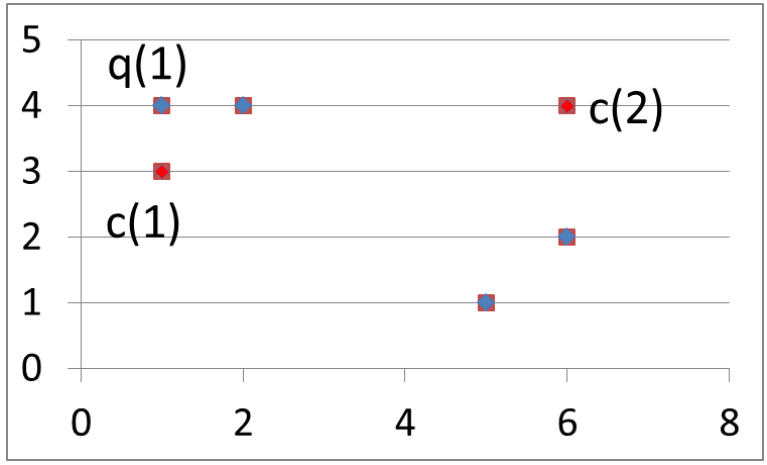
k-Means Clustering

- Inputs: specify number of clusters k and objects that we need to be clustered
- Result: k clusters that minimise an error criterion, e.g., SSE from data points to their assigned clusters
- Process:

1. Define k points as initial centroids c(j)
2. Calculate distance metric for each data point x(i) to each centroid c(j)
3. Assign each object to one cluster based on distance to centroids
4. Update centroids and calculate clustering error
5. Repeat from 2 until clustering error does not change much anymore

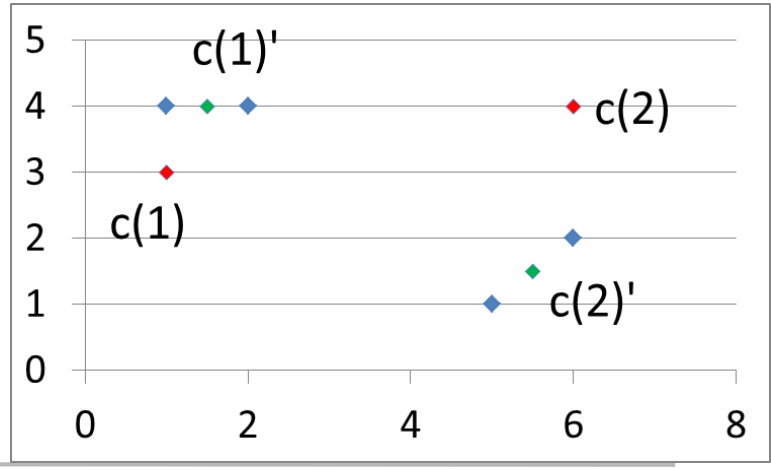
- Distance typically measured by Euclidean Distance, others possible

k-Means: Example



- 4 data points (blue)
- Initial centroids: c(1) and c(2) (red)
- ED of q(1) to c(1) $= \sqrt{(1-1)^2 + (4-3)^2} = 1.0$
- ED of q(1) to c(2) $= \sqrt{(1-6)^2 + (4-4)^2} = 5.0$

		q(1)	q(2)	q(3)	q(4)	c(1)	c(2)
Original data	X	1	2	5	6	1	6
	Y	4	4	1	2	3	4
Calculate ED to c(i)	q(i) to c(1)	1.0	1.4	4.5	5.1	NA	NA
	q(i) to c(2)	5.0	4.0	3.2	2.0	NA	NA
Determine cluster	q(i) to cluster	c(1)	c(1)	c(2)	c(2)	NA	NA
New centroids	X	NA	NA	NA	NA	1.5	5.5
	Y	NA	NA	NA	NA	4	1.5



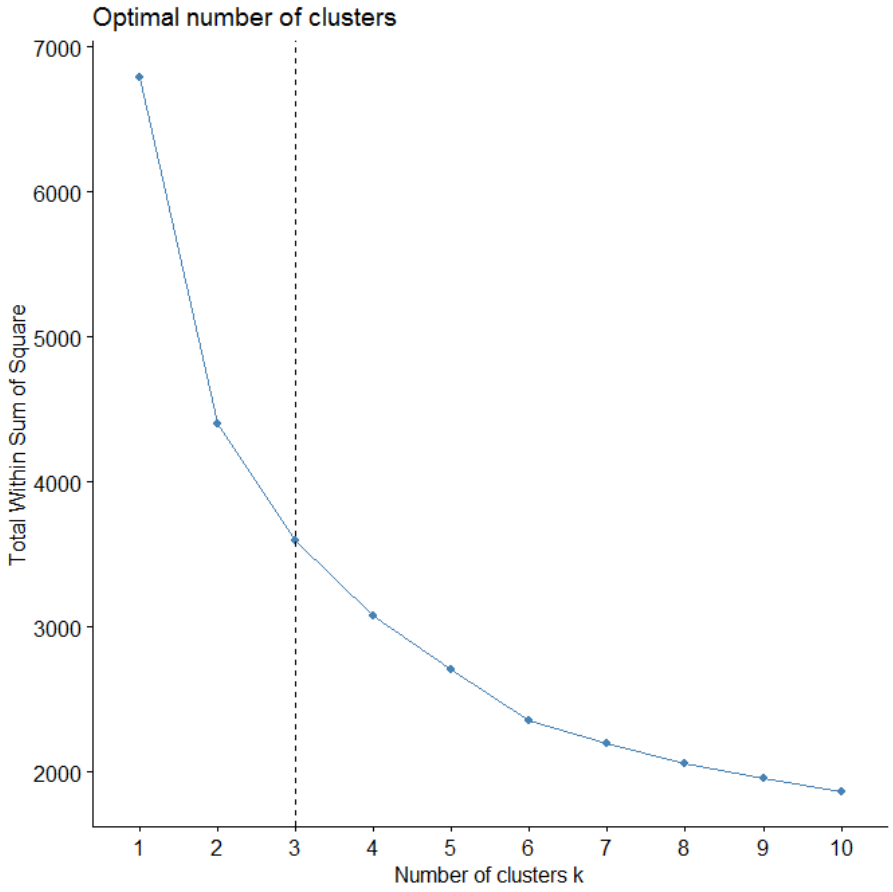
k-Means: Initialisation

- Results of k-Means are highly dependent on initialisation of the centroids
- Ideally, initial centroids are placed as close as possible to the actual ones

1. Random initialisation of centroids
   - Run k-Means several times and choose the model with the smallest error
   - There is no guarantee that this finds the optimal model
2. Run Hierarchical Clustering to set initial centroids (HC is deterministic)
3. Many others

k-Means: How to select the optimal number of clusters (k)?

- Total Within Sum of Squares (TWSS)
  - Cut tree at the point where adding further clusters does not lead to a significant reduction of the TWSS anymore
- Gap Statistic
- Silhouette



k-Means: Difficulties

- k-means has problems when clusters are of differing:
  - Sizes
  - Densities
  - Non-globular shapes

- Outliers

k-Means: Difficulties

- Difficulties of k-Means (Source: Tan et al. (2018, Ch. 8, p. 510ff.)
- Original data has 3 clusters
- k-Means does recognise 3 clusters, but assigns data differently

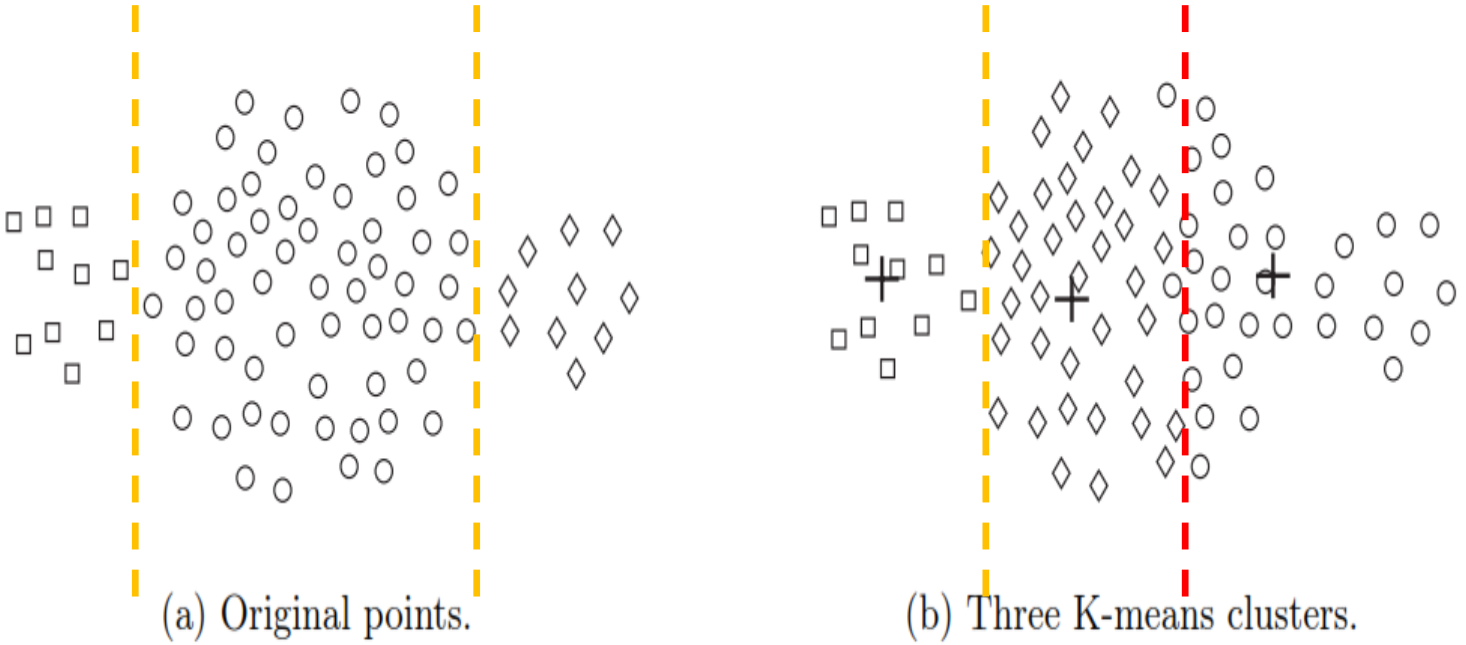
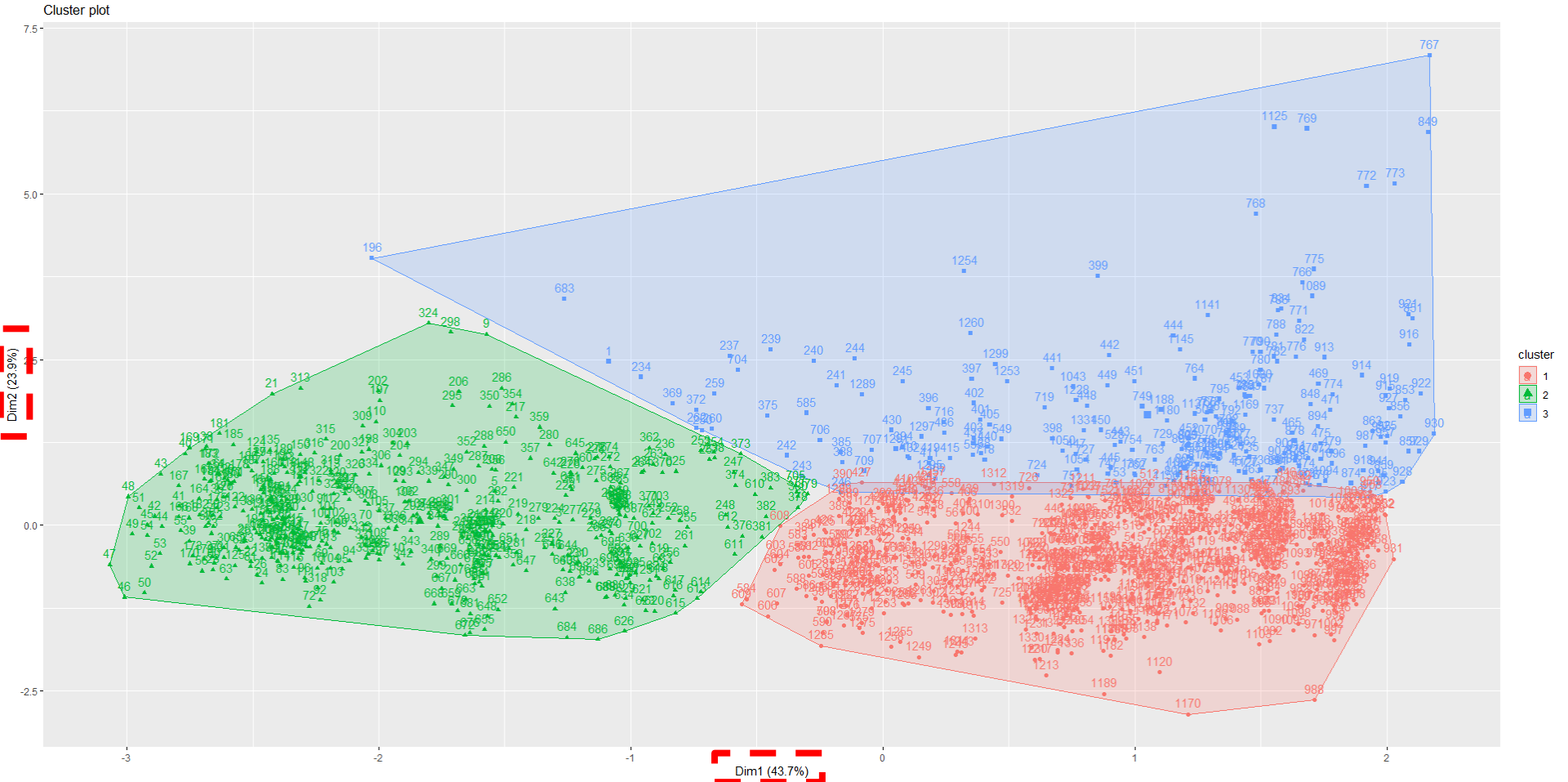


Figure 8.9. K-means with clusters of different size.

Graphical Representation

- Our data are higher than 2-dimensional → graphical representation?
- Run k-Means on original data: find k clusters
  - Input to PCA: original data → PCA calculates PCs and scores. Each score is assigned to 1 of the k clusters
- Display **factor scores** of the first 2 PCs in 2-dimensional chart



Interpreting Clusters and Market Phases, Use for Tactical Asset Allocation

- For example, if there are 3 clusters, each cluster corresponds to 1 allocation:

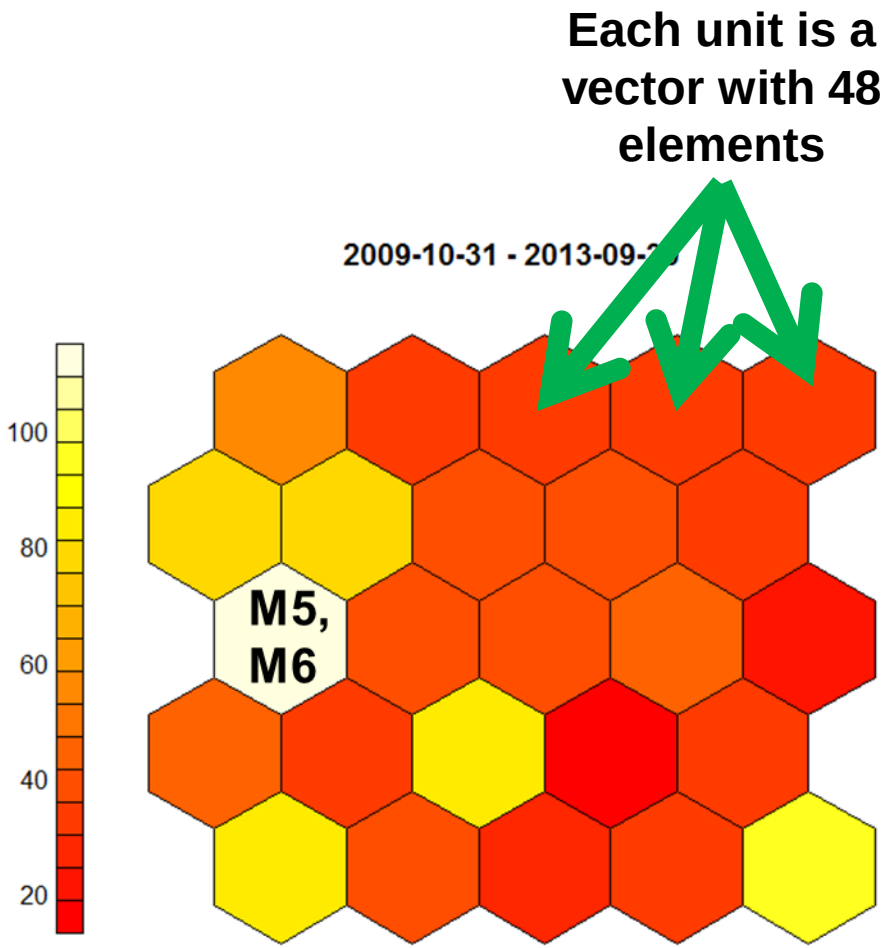
cluster	1	2	3
equities	100%	50%	0%
bonds	0%	50%	100%

- In a similar fashion, for example, use for market phase 1 a trend-following model and for phase 2 a mean-reversion model. Phase 3: blend of both

- Include proximity to clusters in TAA decision:
  - If current phase is close to the transition area of 2 or more clusters, we are uncertain about the market phase → allocate 50/50
  - If current phase is far away from transition area, we are sure about the market phase → allocate 100/0 or 0/100
  - Decision about 100/0 or 0/100 depends on the conditional return distribution of the assets in the clusters

Self-Organising Map (SOM), Kohonen (1982, 2001)

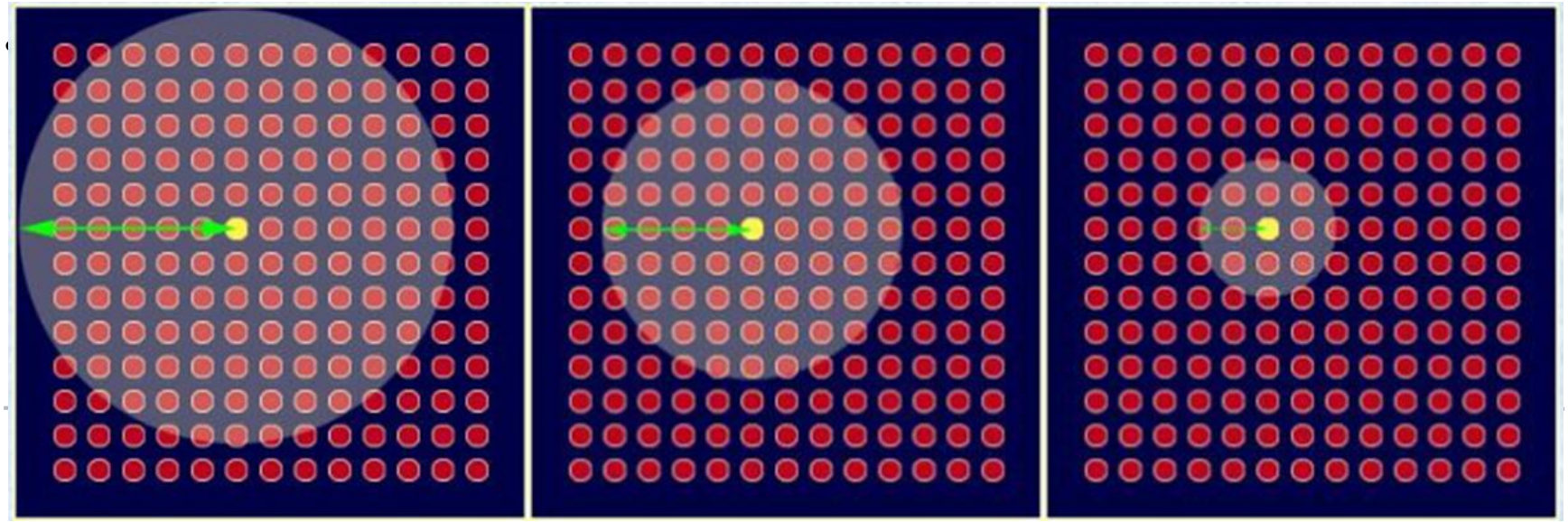
- Self-Organising Map (SOM): project objects on a map
- Similar objects are being projected closely together
- Example: Hedge Fund managers
- Database with 1000 HF, 48 monthly returns
- Managers with similar return profiles appear on the same unit: Managers M5 and M6 on the same unit



How Do SOM “Learn”? (1)

- Learning = creating “good” representatives, i.e., the units of the SOM are calibrated such that they represent a subset of the sample

1. Random initialisation, also PCA-based
2. Samples are presented to the SOM
3. Identify the unit most similar to the current sample or sample subset (Best Matching Unit or BMU)
4. Update units to become more similar to the sample (see next slide)



How Do SOM "Learn"? (2)

- How many and which neighbouring units are updated depends on the neighbourhood function $\theta(t)$ → this is one of the reasons how non-linearity can be reflected in the SOM
- The learning rate α determines the size of the weight adjustment, the neighbourhood function $\theta(t)$ the radius around the BMU:

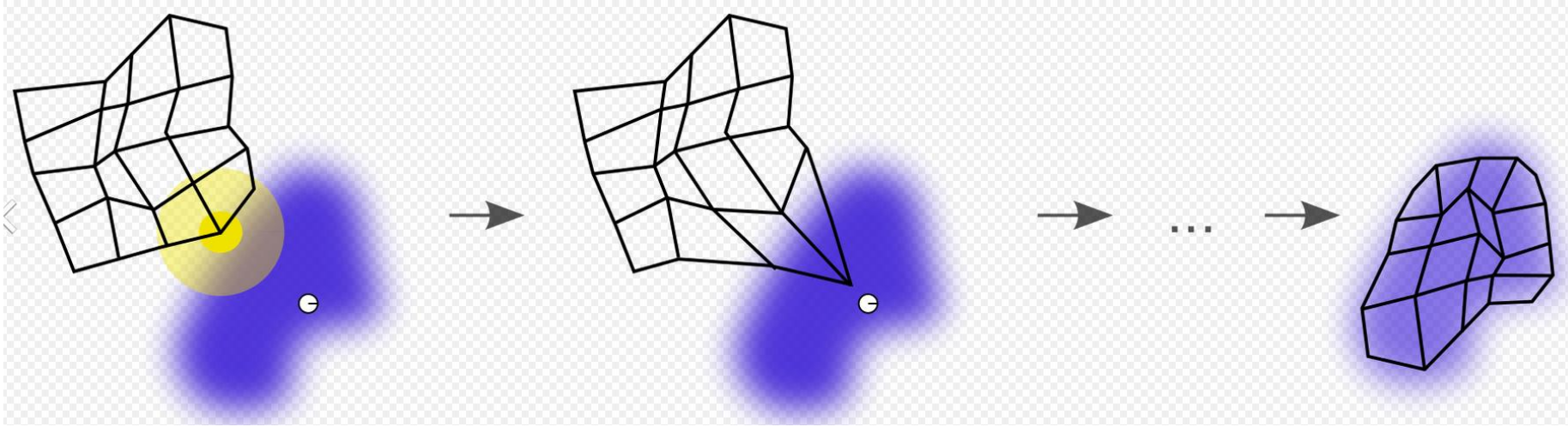
$$w_{ij}(t+1) = w_{ij}(t) + \theta(t) \cdot \alpha(t) \cdot \left(x_i(t) - w_{ij}(t)\right)$$

  - $x_i(t)$: characteristics of the samples, e.g., returns of manager i at learning cycle t
  - For example, α = 0.05 means that centroids' weights get adjusted by 5% of the difference between sample and existing weight
- $\theta(t)$ starts with a large value to include many units in the weight adjustment process and ends with only 1 unit (= BMU) being adjusted

5. Loop back to step 2 until map error does not change anymore
   - If only the BMU would be updated, SOM would yield identical results as k-Means

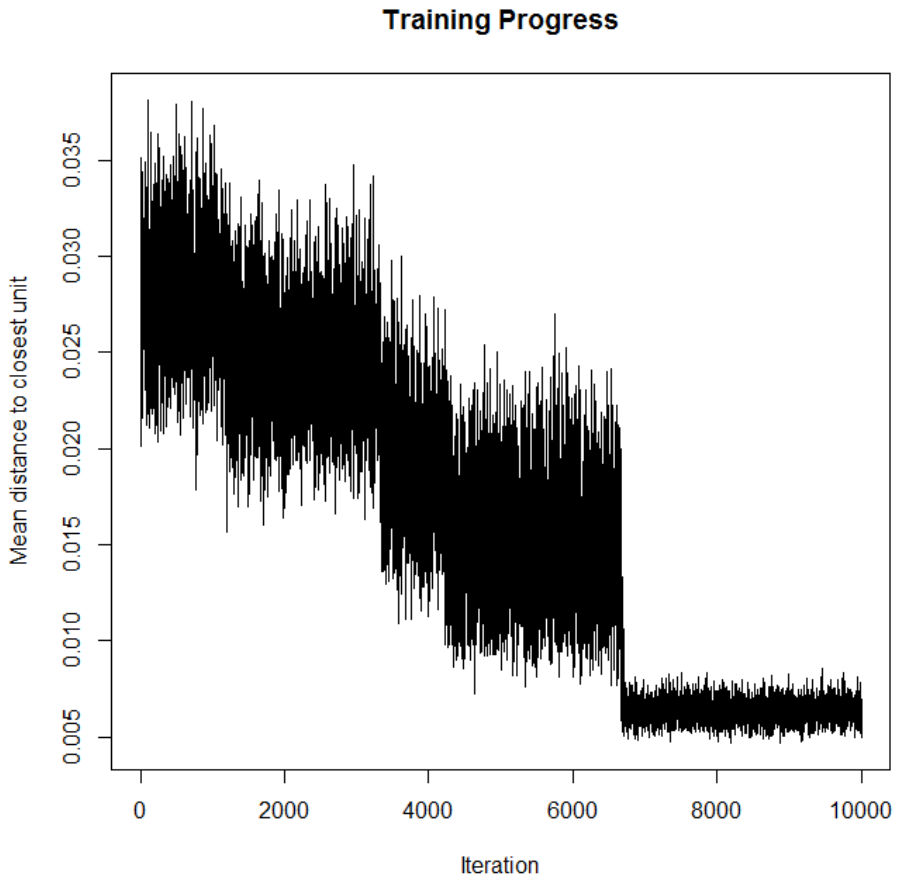
How SOM Wrap Around the Data Distribution

- Source: By Mcld - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=10373592
- Blue cloud is the distribution of the training data (in our example: the hedge fund database)
- Small white disc is the current data point drawn from that distribution (i.e., one hedge fund manager)
- At first (left) the 5 x 5 SOM units are arbitrarily positioned in the data space
- The unit (highlighted in yellow) which is nearest to the training data point is selected
- It is moved towards the training data point, as (to a lesser extent) are its neighbours on the grid. After many iterations the grid tends to approximate the data distribution (right)



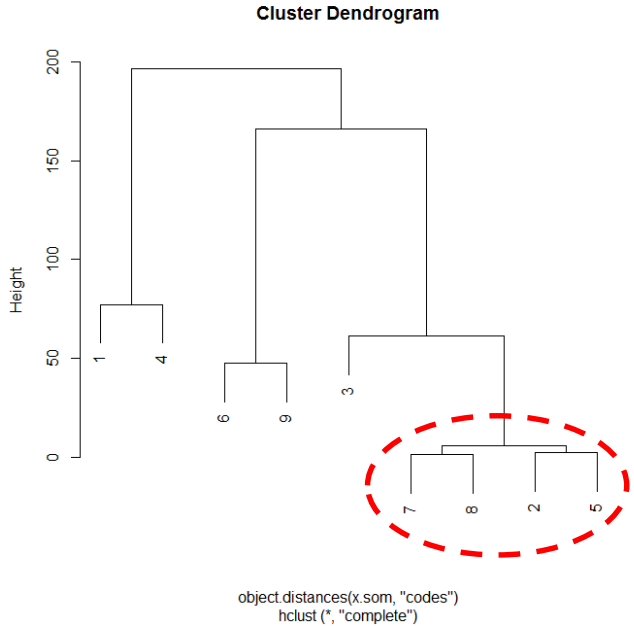
SOM: Training

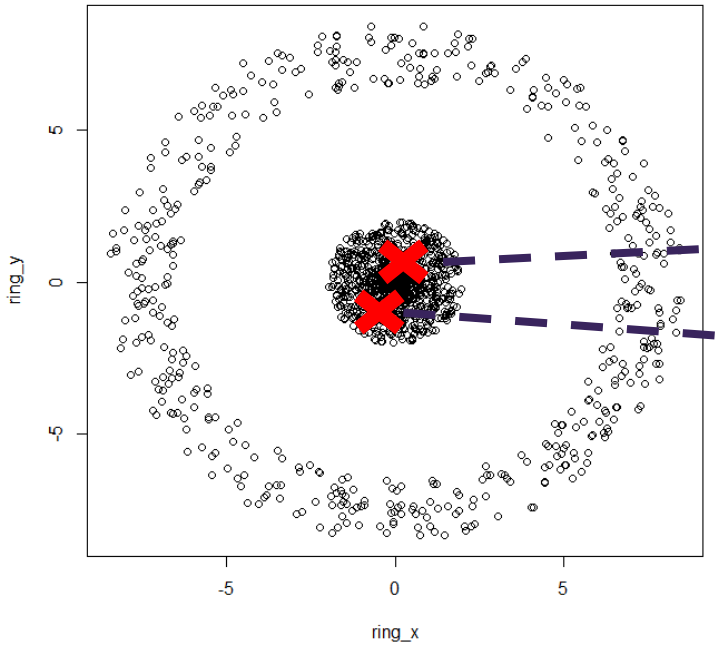
- Training finishes when error does not decline further

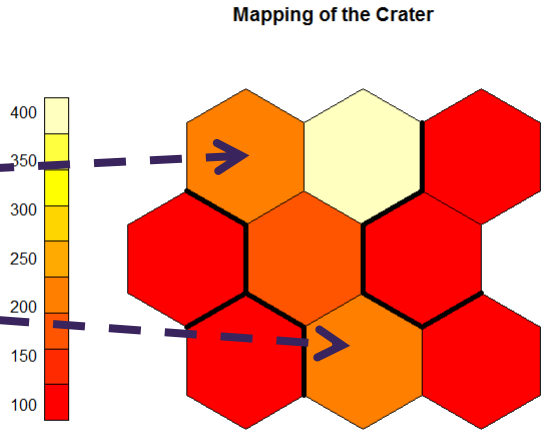


Non-Linearity in SOM: Crater

- Crater core: 1000 data points, crater ring: 500 (2 dimensions: x, y)
- SOM maps crater core onto units 2, 5, 7, 8
- Crater ring is mapped onto other units
- For this example, the R code is “R Crater Image CQF.R” in the supplementary material
- If 2 data points in the core get mapped onto distant units at the beginning of the training, the neighbourhood function takes care that both units get updated and hence move closer together (i.e., the centroids’ weights get adjusted in the same direction)

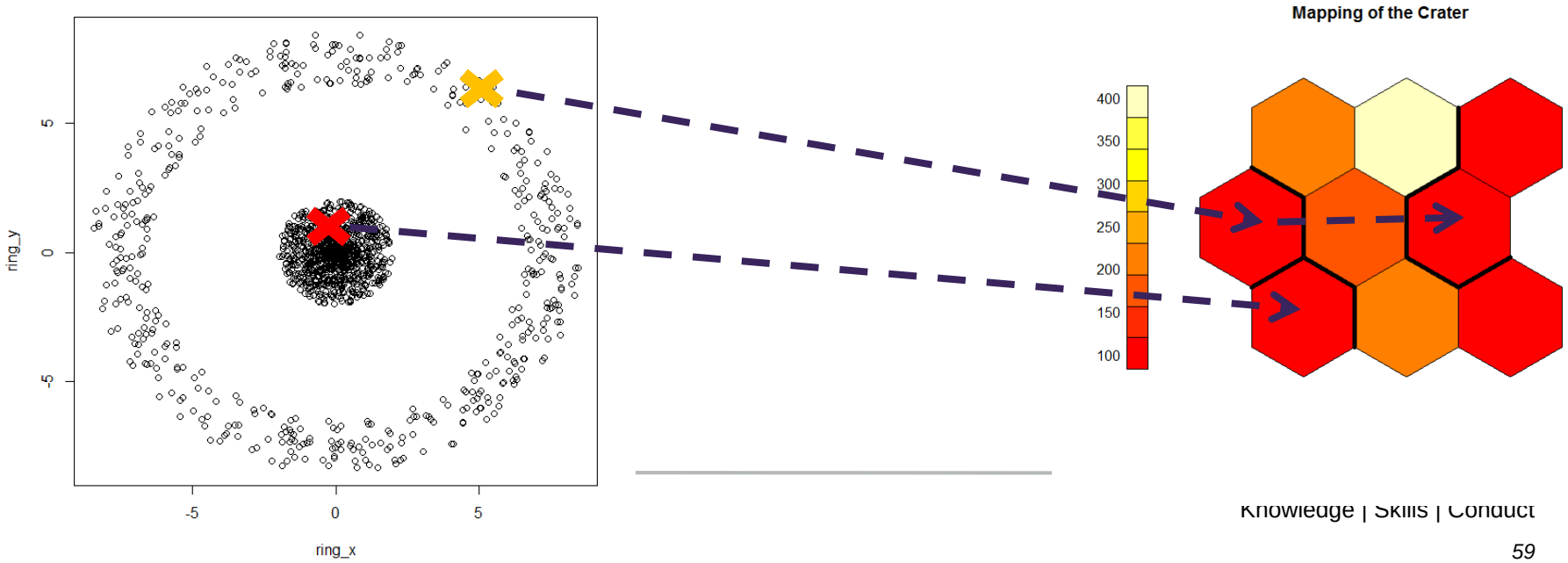






Non-Linearity in SOM: Crater

- If 2 data points that lie in ring and core are mapped onto near-by units at the beginning of the training, the neighbourhood function involves both in updating → units move away from each other
  - When the 2 data points are presented to the units in the early phase of learning, units 4 and 6 (amongst others) are adjusted to resemble more centroid 1 (→ neighbourhood function)
  - In a later phase of training, the neighbourhood function helps to determine that data point 1 belongs to unit 4 and hence is updated (and also possibly unit 1), but unit 6 is out of reach of the neighbourhood function and hence does not get updated
  - When data point 2 is presented in the later phase of training, it resembles unit 6 more than unit 4 and hence only unit 6 is updated: the BMU for the **orange unit** is no longer on unit 4, but on unit 6



Assessing the Quality of a SOM: Quantisation Error

- Quantisation Error: **on average** over the dataset, pairs of neighbouring **data** will be projected onto the same or neighbouring **units**
- Calculate Distance between each data point and its BMU:
  - $\mathrm{QE} = \frac{1}{N}\sqrt{\sum_{j=1}^{N}(c_j - x_j)^2}$

N: # of data points

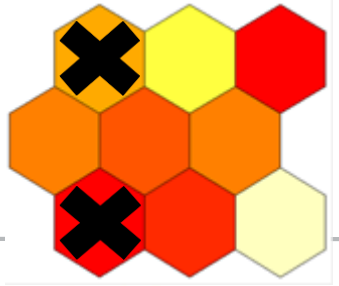
c_j: BMU (vector of size T)

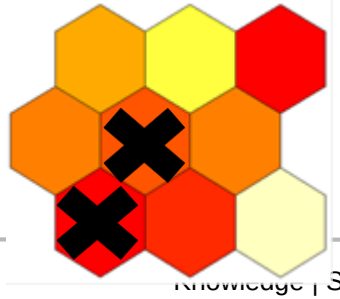
x_j: data point j (vector of size T)

- Use QE to compare 2 SOMs: the SOM with the lowest QE fits the data more accurately **on average**
- QE only measures relationship between units and items mapped to those units, **NOT** how units preserve neighbourhood of the data

Assessing the Quality of a SOM: Topographic Error

- Topology Preservation: **neighbourhood** relationships of the data should be reflected by the SOM
- Topographic Error:
  - Evaluate the distance of the **BMU and the 2nd BMU for each data point**
  - $\mathrm{TE} = \frac{1}{N}\sum_{j=1}^{N} u(x_j)$ N: # of data points $0 \leq \mathrm{TE} \leq 1$
- Explanation of $u(x_j)$ by example:
  - Manager 1 mapped onto unit 1 (bottom left)
  - 2nd BMU is unit 7 → not adjacent to unit 1 → $u(x_1) = 1$
  - Manager 2 mapped onto unit 1 (bottom left)
  - 2nd BMU is unit 5 → adjacent to unit 1 → $u(x_2) = 0$

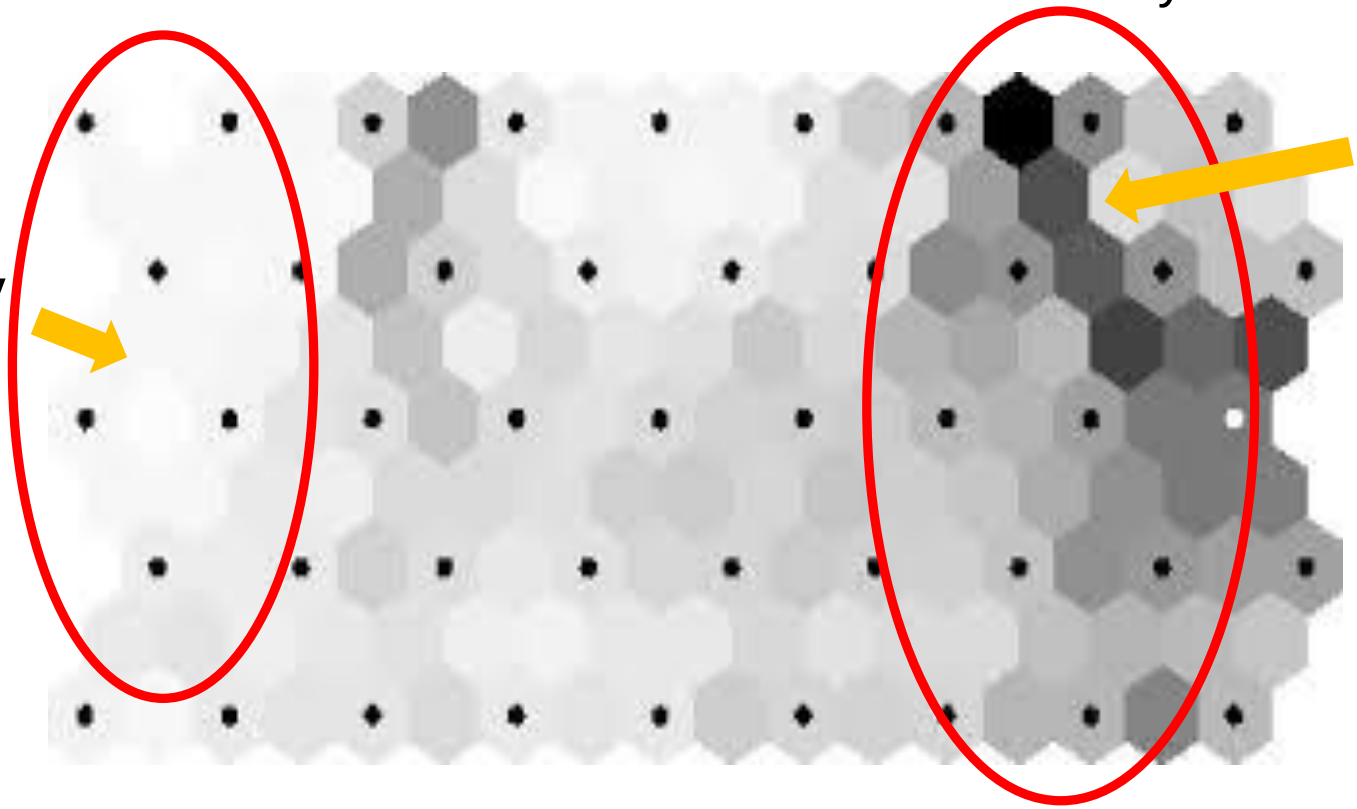




SOM: U-Matrix (=Unified Distance Matrix)

- In the chart below the units have black dots and the distances are marked in grey
- The dark elements show dissimilarities between nearby units

Small distances between units: they form one cluster



Large distances between units: separate cluster

- Chart taken from: https://users.ics.aalto.fi/jhollmen/dippa/node24.html

SOM: Features

- SOM reduce dimensionality of data (here: from 48 dimensions to 2)
- Units have neighbourhood relationships to other units: SOM preserves neighbourhood of data → Topology preservation
- SOM learning starts global (a large part of the SOM's units are involved in the learning process) and ends local (only the BMU learns)
- SOM is a **local** function approximator…
- … as opposed to **global** function approximators like ANN
  - With ANN, function approximations are widely distributed in the network. Difficult to interpret results
- Local models can be more easily interpreted than global models (it is hard to interpret ANN)
- SOM can deal with non-linear & noisy data

SOM: Missing Data and Forecasting

- SOM can handle missing data → they are replaced by their local centroids (likely less interfering than using the mean, a global measure)

- This feature can be used for forecasting:
  - Train SOM with variables in (N + 1) vectors (N regressors, 1 regressand):

	Economic Variables (Predictors)			
	V_1	V_2	…	V_N
2013-12-31				
2014-01-31				
2014-02-28				
⋮	⋮	⋮	⋮	⋮
2017-12-31				

	Equity Returns
	Predicted
2014-01-31	
2014-02-28	
2014-03-31	
2018-01-31	

  - To forecast, use predictors as of time t and leave the value for the predicted variable NA → SOM will replace the NA with the centroid of the assigned cluster

2018-03-31	x1	x2	x3	x4

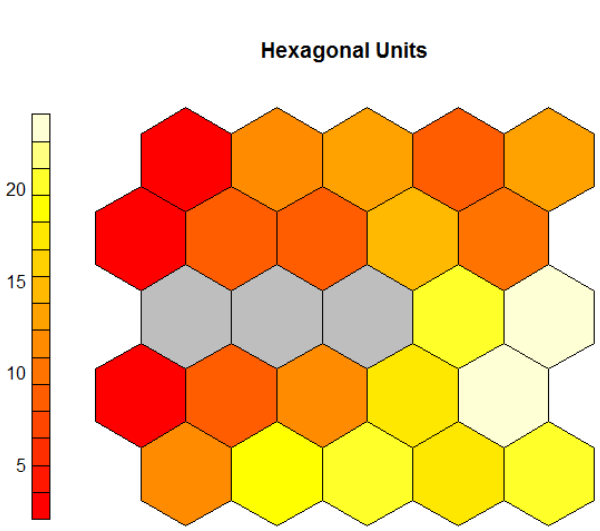
2018-04-30	NA

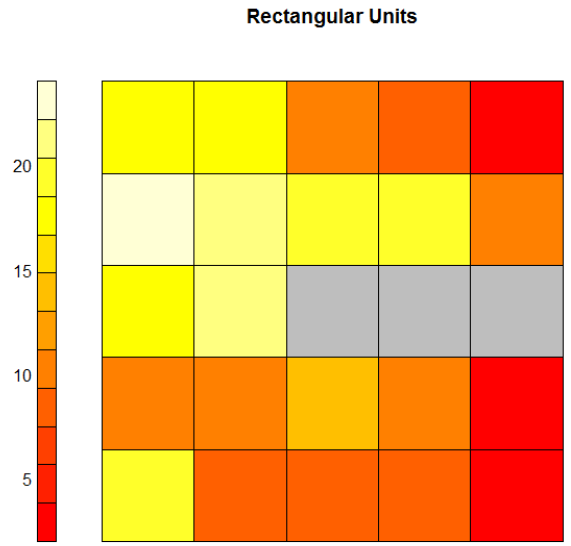
SOM: Decisions to be Made

- Architecture
  - Shape of the SOM
  - Shape of the units (hexagonal vs. rectangular)
  - Number of units
- Learning Parameters
  - Shape of the neighbourhood function (e.g., bubble or Gaussian)
  - Learning rate α
- Interpretation of the SOM
  - Number of clusters
  - Are clusters meaningful economically?

SOM: Shape of the Units

- Hexagonal units have 6 neighbours each, rectangular units 4
- Hexagonal units can capture the data distribution more smoothly





Number of Units (=Codebook Vectors)

- Rule-of-thumb: ca. 50 items (here: managers) per unit on average sufficient (Kohonen (2013))
- Shape of the SOM:
  - 1 to 3 dimensions: typically 2-dimensional (visual representation)
  - Choose symmetric grid, assign 50 items to each unit (for 1000 items: 20 units → 4 x 4 or 5 x 5)
  - Rectangular grid based on the length of the first 2 PCs (Kohonen, 2013). Example: Eigenvalue 1 explains 40% of total variance, Eigenvalue 2: 20% → horizontal to vertical = 2:1 (for our 1000 managers: ca. 20 units: 7 horizontal, 3 vertical)
- Some centroids might not be needed → no items mapped on those empty units

SOM: Initialisation

- Completely random initialisation: SOM initially does not reflect any structure
  - Random initialisation preferred for non-linear data (Akinduko et al. (2016))
- PCA initialisation: SOM initially reflects global structure
- The PCA-initiated weights are synthetic factor scores → they reflect the variable system's global (correlation) structure
  - Result: SOM's initial state is globally ordered. SOM can now work to create local ordering
  - PCA initialisation can lead to faster convergence
- Other methods, like k-Means, can also be used to initialise SOM
- 2 main sources of varying results (deBodt et al. (2002)):

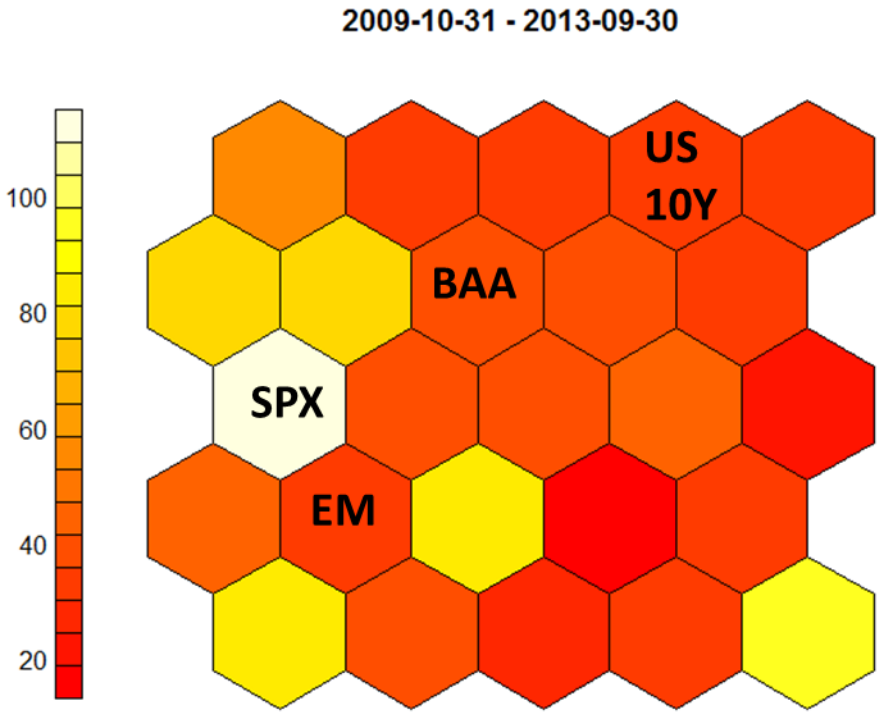
1. Initialisation of the units influences their final positions
2. Sampling of inputs used for learning

SOM: Applications

- SOM can be applied to many problems to identify similarities in data, e.g., fraud detection, benchmarking
- Text classification (Kohonen (2013)):
  - For classification / categorisation of large document collections (several thousand documents), e.g., Encyclopedia Britannica or patent abstracts
  - In each text, count number of words
  - Generate word histograms: for each document, count number of words
  - Feed to SOM
  - Result: clusters on SOM exhibit texts with similar word count or similar content
- SOM for risk analysis: If assets of an existing portfolio come all from the same part of the SOM, there is little diversification to expect in times of crisis
- Crisis prediction (Sarlin, P. (2014))

SOM: Visual Risk Analysis

- SOM can be used to identify sensitivity to benchmarks
  - SPX: S&P500 Equity Index
  - BAA: Moody's Baa Spread
  - EM: Emerging Market Equities
  - US10Y
- Managers in the bottom right corner independent of BMs



Summary

- Unsupervised ML methods useful for:
  - Dimension reduction
  - Noise reduction
  - Understanding the structure of the data
- PCA: simple & useful tool for Machine Learning → master it!!
- Always prefer a model with clear interpretation over a model with a better quantitative fit
- ML tools are powerful, but require caution and experience
- Visual interpretation – excellent to show clients
- Practice! – powerful tools are available free (R, Python)

Appendix: PCA, Eigenvectors and Eigenvalues

- Eigenvectors and Eigenvalues play a central role in PCA
- This appendix tries to shed some light on the rationale of PCA in connection with Eigenvectors and Eigenvalues
- Starting point is a matrix A (in the context of PCA this would be the VCV):
  - When multiplied by an Eigenvector u we get the same result as if A would be multiplied by a scalar λ, its Eigenvalue:
- $Au = \lambda u$ or rewritten: $(A - \lambda I)u = 0$
- See also complementary R code “R Eigenvectors.R”

Eigenvectors and Eigenvalues

- For covariance matrix …

Cov	x	y
x	1.47	0.88
y	0.88	1.04

- … we get the Eigenvectors and corresponding Eigenvalues:

	Eigenvalues	
	2.16	0.35
	Eigenvectors	
	PC1	PC2
x	-0.79	0.62
y	-0.62	-0.79

Eigenvalues

- The variance of our variable system is 1.47 + 1.04 = 2.51
  - Sum of the diagonal of our VCV
- The sum of the 2 Eigenvalues is 2.16 + 0.35 = 2.51
  - This is identical to the sum of the diagonal

Eigenvalues

- Verify $Au_1 = \lambda u_1$:
- Au_1 = 1.47 * (-0.79) + 0.88 * -0.62 = -1.70
  0.88 * (-0.79) + 1.04 * -0.62 = -1.33
  = **(-1.70 -1.33)**
- Now with Eigenvalues:
- $\lambda_1 u_1 = 2.16$ * (-0.79 -0.62)
  = **(-1.70 -1.33)**
- $\lambda_2 u_2 = 0.35$ * (0.62 -0.79)
  = **(0.22 -0.28)**

Eigenvectors: Orthogonality

- When 2 vectors are orthogonal to each other, their dot product is 0
  - Geometrically, this means that the 2 Eigenvectors are at a right angle
- Our 2 Eigenvectors are supposed to be orthogonal to each other, hence:
- $\begin{pmatrix}-0.79\\-0.62\end{pmatrix} \cdot \begin{pmatrix}+0.62\\-0.79\end{pmatrix}$ = -0.79 * -0.62 + 0.62 * -0.79 = 0 → in fact our Eigenvectors are orthogonal

Eigenvectors: Orthogonality

- The **green** dots represent the **VCV** of the original data:

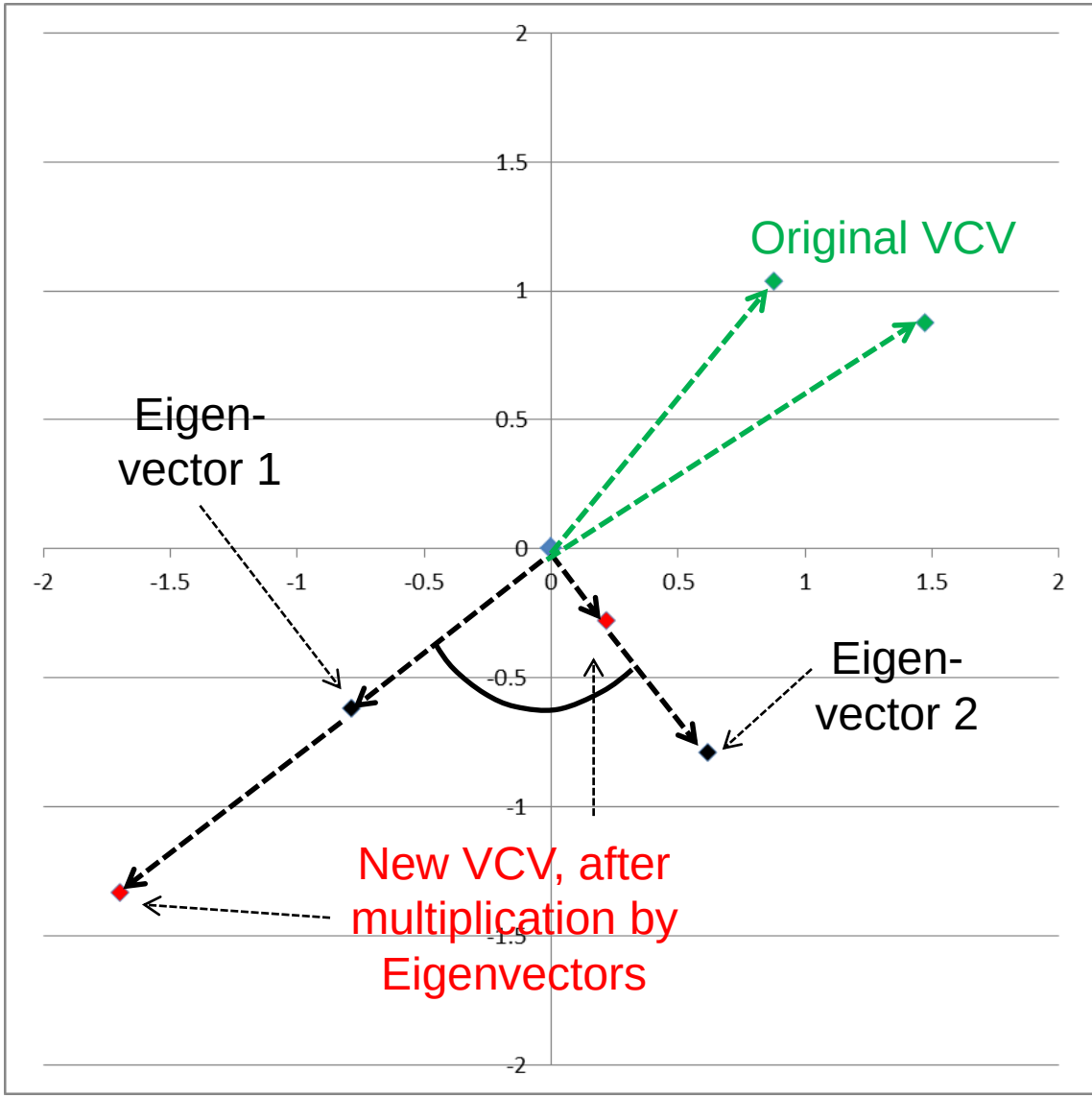
$$\begin{pmatrix} 1.47 & 0.88 \\ 0.88 & 1.04 \end{pmatrix}$$

- The **black** dots are the 2 Eigenvectors:

$$\begin{pmatrix} -0.79 \\ -0.62 \end{pmatrix}, \begin{pmatrix} +0.62 \\ -0.79 \end{pmatrix}$$

  - Both Eigenvectors have length = 1

- The **red** dots represent the new VCV after multiplication by the 2 Eigenvectors



Eigenvectors: Orthogonality

- Multiplication of A (= VCV) by the matrix with the Eigenvectors linearly transforms A
- The transformed matrix, i.e., the new VCV $\begin{pmatrix} -1.70 & 0.22 \\ -1.33 & -0.28 \end{pmatrix}$, reflects the same structure as the original VCV
  - Are the new VCV's components (= column vectors) orthogonal to each other?:
  - $\begin{pmatrix} -1.70 \\ -1.33 \end{pmatrix} \cdot \begin{pmatrix} +0.22 \\ -0.28 \end{pmatrix}$ = -1.70 * 0.22 + -1.33 * -0.28 = 0 → the 2 column vectors are orthogonal!
- Hence the new VCV reflects the same structure as the original VCV, but the new VCV's components (= column vectors) are orthogonal to each other

Why does the sum of all Eigenvalues explain the variance of the variable system?

- When we compare the lengths of the 2 coordinates of the new VCV, we find that the multiplication by Eigenvector 1 had a larger impact than by Eigenvector 2
- Compare the lengths of the 2 transformed coordinates, i.e., the column vectors in the transformed matrix $\begin{pmatrix} -1.70 & 0.22 \\ -1.33 & -0.28 \end{pmatrix}$:
- $\sqrt{(-1.70)^2+(-1.33)^2} = 2.16 \rightarrow$ identical with Eigenvalue 1
- $\sqrt{(+0.22)^2+(-0.28)^2} = 0.35 \rightarrow$ identical with Eigenvalue 2
- The length of the transformed vectors reflects the amount of information the transformed vectors convey

PCA, Eigenvectors and Eigenvalues

- Eigenvectors and Eigenvalues can be applied to square matrices: eigen-decomposition
- The eigen-decomposition of a matrix represents their structure
- PCA is the result of an eigen-decomposition of a covariance matrix (VCV) and provides the least square representation of the original data
- For a detailed discussion of eigenvectors, eigenvalues and PCA see Abdi (2007)

PCA, Eigenvectors and Eigenvalues

1) Mathematical Representation

- Matrix * vector = scalar * vector
- Orthogonalisation: find a transformation that transforms the original data points so that they preserve the structure of the original data, but remove correlation
- When we apply PCA, A is the VCV of our data and we want to find the Eigenvectors EV and Eigenvalues lambda

2) Economic meaning

- EV is a vector that is scaled up by a transformation, but does not change fundamentally by the transformation
- The larger the eigenvalue the more important the corresponding EV
  - Eigenvalue gives the variance explained by the corresponding Eigenvector

PCA, Eigenvectors and Eigenvalues

- Eigenvalues are the factors by which the vectors of a system get scaled
- An eigenvalue of 2 means that the length of the EV has doubled
- Length of (3 2) = sqrt(3^2 + 2^2) = sqrt(13) = 3.61
- (3 2) * 2 = (6 4) => length of (6 4) = sqrt(6^2 + 4^2) = sqrt(52) = 7.21
- An eigenvalue of 0.5 means that the length of the EV has halved
- Eigenvalue of 1 → EV remains the same, but there could be a rotation around the origin, e.g., rotating a cube in 3D space around its origin
- The rotation can introduce orthogonality
- Typically the Eigenvectors are normalised, i.e., scaled to a length of 1

References

- Abdi, H. (2007). The eigen-decomposition: Eigenvalues and eigenvectors. Encyclopedia of measurement and statistics, 304-308.
- Akinduko, A. A., Mirkes, E. M., & Gorban, A. N. (2016). SOM: Stochastic initialization versus principal components. Information Sciences, 364, 213-221.
- Artis, M. J., Banerjee, A., & Marcellino, M. (2005). Factor forecasts for the UK. Journal of forecasting, 24(4), 279-298.
- Barreto G.A. (2007) Time Series Prediction with the Self-Organizing Map: A Review. In: Hammer B., Hitzler P. (eds) Perspectives of Neural-Symbolic Integration. Studies in Computational Intelligence, vol 77. Springer, Berlin, Heidelberg.
- Boivin, J., & Ng, S. (2006). Are more data always better for factor analysis?. Journal of Econometrics, 132(1), 169-194.
- De Bodt, E., Cottrell, M., & Verleysen, M. (2002). Statistical tools to assess the reliability of self-organizing maps. Neural Networks, 15(8-9), 967-978.
- Dose, C., & Cincotti, S. (2005). Clustering of financial time series with application to index and enhanced index tracking portfolio. Physica A: Statistical Mechanics and its Applications, 355(1), 145-151.
- Huber, C. (2019). Machine Learning for Hedge Fund Selection. Wilmott Magazine, 2019(100), 74-81.
- Huber, C. (2019). R Tutorial on Machine Learning: How to Visualize Option-Like Hedge Fund Returns for Risk Analysis. Wilmott Magazine, 2019(99), 36-41.
- Huber, C. (2019): Data for the above R tutorial, http://rodexrisk.com/?page_id=431.

References

- Lebart L., Morineau A., Piron M. (2000). Statistique exploratoire multidimensionnelle. Dunod, Paris, France. ISBN 2100053515.
- Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. Biological cybernetics, 43(1), 59-69.
- Kohonen, T. 2001. Self-organizing maps, ser. Information Sciences. Berlin: Springer, 30.
- Kohonen, T. (2013). Essentials of the self-organizing map. Neural networks, 37, 52-65.
- Lendasse, A., François, D., Wertz, V., & Verleysen, M. (2005). Vector quantization: a weighted version for time-series forecasting. Future Generation Computer Systems, 21(7), 1056-1067.
- Merényi, E., Tasdemir, K., & Zhang, L. (2009). Learning highly structured manifolds: harnessing the power of SOMs. In Similarity-based clustering (pp. 138-168). Springer, Berlin, Heidelberg.
- Milligan, G. W., & Cooper, M. C. (1985). An examination of procedures for determining the number of clusters in a data set. Psychometrika, 50(2), 159-179.
- Pelata, M., Giannopoulos, P., Haworth, H. (2012): PCA Unleashed, Credit Suisse Interest Rate Strategy.
- Raffinot, T. (2018). The hierarchical equal risk contribution portfolio. Available at SSRN 3237540.
- Redfern, D., MacLean, D. (2014): Principal Component Analysis for Yield Curve Modelling, Moody's Analytics.
- Ritter, H.G., & Kohonen, T. (1989). Self-organizing semantic maps. Biological Cybernetics, 61, 241-254.

References

- Sarlin, P. (2014). Mapping financial stability. Springer.
- Yin, H. (2008). Learning nonlinear principal manifolds by self-organising maps. In Principal manifolds for data visualization and dimension reduction (pp. 68-95). Springer, Berlin, Heidelberg.