The SVD

Helwig Hauser *et al.*,
UiB Dept. of Informatics

Looking Back & Forth

Last time:

- vector space and basis
- coordinates wrt. a basis
- changing the basis
- projection

Remember:

- changing the basis of mappings, etc.

Today:

- SVD (singular value decomposition)
- interpreting the SVD
- examples
- low-rank approximation

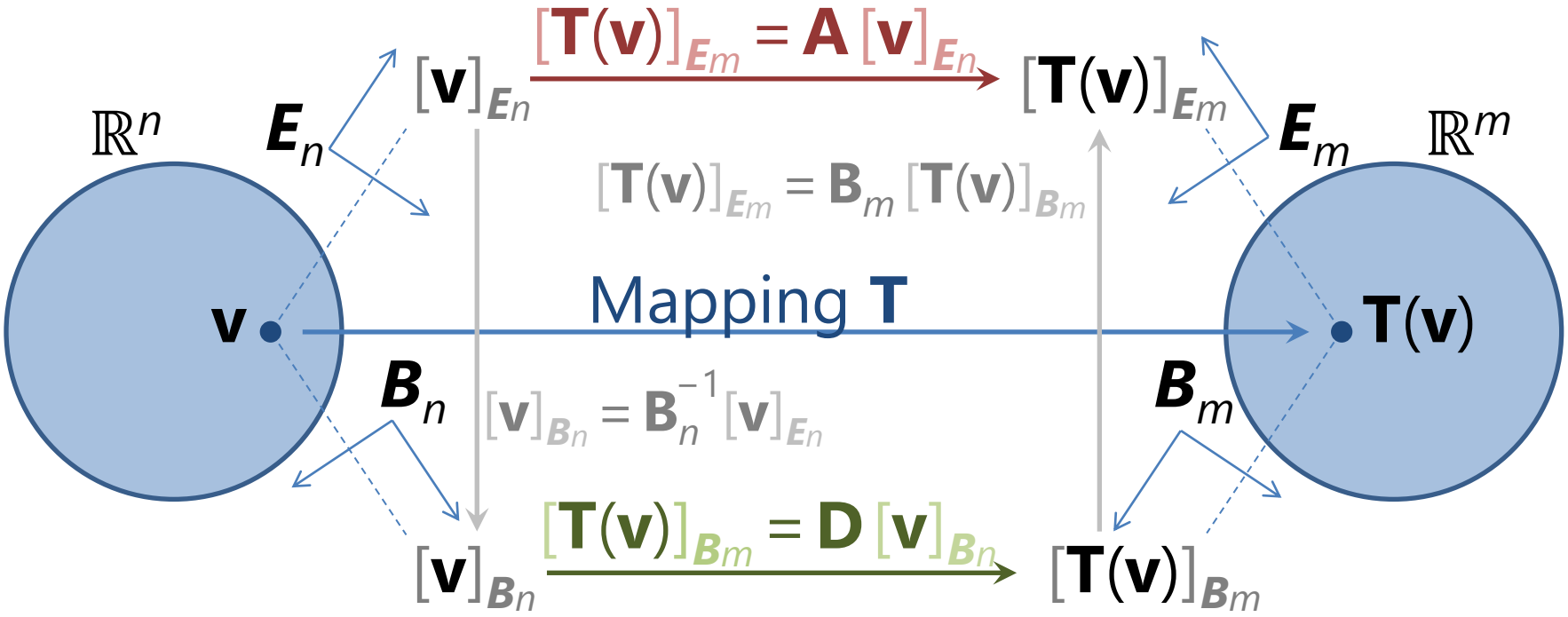
Repetition: Mappings, different bases

T maps from $\mathbb{R}^n$ to $\mathbb{R}^m$ … $\mathbf{T}(\mathbf{v}) = \mathbf{A}\,\mathbf{v}$ in std. coords. $\boldsymbol{E}_n$, $\boldsymbol{E}_m$

… $\boldsymbol{B}_n$: alternative basis of $\mathbb{R}^n$, ergo $[\mathbf{v}]_{\boldsymbol{E}_n} = \mathbf{B}_n\,[\mathbf{v}]_{\boldsymbol{B}_n}$

$\boldsymbol{B}_m$: alternative basis of $\mathbb{R}^m$, ergo $[\mathbf{v}]_{\boldsymbol{E}_m} = \mathbf{B}_m\,[\mathbf{v}]_{\boldsymbol{B}_m}$

$\Rightarrow [\mathbf{T}(\mathbf{v})]_{\boldsymbol{E}_m} = \mathbf{B}_m\,\mathbf{D}\,\mathbf{B}_n^{-1}\,[\mathbf{v}]_{\boldsymbol{E}_n}$



Changing a Mapping's Basis (and back)

Assume we have a linear mapping $\mathcal{M}$ from $\mathbb{R}^n$ to $\mathbb{R}^m$, i.e., $\mathcal{M}: \mathbb{R}^n \to \mathbb{R}^m$ and $\mathcal{M}(\mathbf{v}) = \mathbf{A}\,\mathbf{v}$ (in standard coordinates).

Realizing $\mathcal{M}(\mathbf{v})$, based on considering $\mathbf{v}$ in $\mathcal{B}$-coordinates and the result, $\mathcal{M}(\mathbf{v})$, in $\mathcal{G}$-coords., $\mathcal{M}(\,(c_j)_{\mathcal{B}}\,)_{\mathcal{G}}$ amounts to the left-multiplication of $(c_j)_{\mathcal{B}}$ with $\mathbf{D} = \mathbf{G}^{-1}\mathbf{A}\mathbf{B}$.

$$\mathcal{M}(\mathbf{v})_{\mathcal{G}} = \mathbf{G}^{-1}\,\mathcal{M}(\mathbf{v})_{\mathcal{E}} = \mathbf{G}^{-1}\,\mathbf{A}\,\mathbf{B}\,(c_j)_{\mathcal{B}} = \mathbf{D}\,(c_j)_{\mathcal{B}}$$

This means that we can then write

$$\mathcal{M}(\,(v_i)_{\mathcal{E}}\,)_{\mathcal{E}} = \mathbf{A}\,(v_i)_{\mathcal{E}} = \mathbf{G}\,\mathbf{D}\,\mathbf{B}^{-1}\,(v_i)_{\mathcal{E}}$$

SVD – Introduction

Singular value decomposition (SVD):

- decomposes a matrix into 3 constitutive components (matrices):
  $\mathbf{A} = \mathbf{U}\,\mathbf{\Sigma}\,\mathbf{V}^*$
- useful for many purposes:
  - understanding data
  - finding the pseudo-inverse of **A**
  - solving homogeneous equation systems $\mathbf{A}\,\mathbf{x} = 0$
  - finding the range, null space, and rank of **A**
  - low-rank approximation of **A**
  - least-squares minimization
  - etc.

more about these matrices in a moment

Singular Value Decomposition

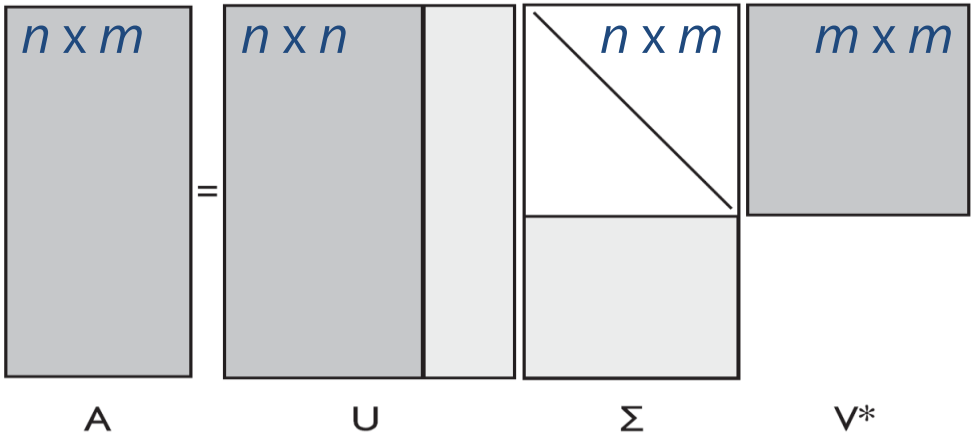
Components:

- Matrix $\mathbf{A} \in \mathbb{C}^{n \times m}$ ($n \geq m$, rank $r \leq m$) decomposes into $\mathbf{U}\,\mathbf{\Sigma}\,\mathbf{V}^*$ with
  - the singular values $\{\sigma_j\}$, constituting the diagonal matrix $\mathbf{\Sigma}$, $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r \geq \sigma_{r+1} = \ldots = \sigma_m = 0$
  - two unitary change-of-basis matrices $\mathbf{U}$ and $\mathbf{V}$
    - $\mathbf{U} = (\mathbf{u}_1\ \mathbf{u}_2\ \ldots\ \mathbf{u}_n) \in \mathbb{R}^{n \times n}$ represents a basis of the destination space
    - $\mathbf{\Sigma} \in \mathbb{R}^{n \times m}$ hosts the (sorted) singular values along the main diagonal
    - $\mathbf{V} = (\mathbf{v}_1\ \mathbf{v}_2\ \ldots\ \mathbf{v}_m) \in \mathbb{R}^{m \times m}$ represents a basis of the domain

Theorem:

- Every matrix $\mathbf{A} \in \mathbb{C}^{n \times m}$ has a singular value decomposition (SVD).
- Furthermore, the singular values $\{\sigma_j\}$ are uniquely determined, and, if $\mathbf{A}$ is square and the σ_j are distinct, the singular vectors $\{\mathbf{u}_j\}$ and $\{\mathbf{v}_j\}$ are uniquely determined up to complex signs (complex scalar factors of absolute value 1).

SVD – Illustration (case $n \geq m$)



this is also the decomposition that MatLab gives via call

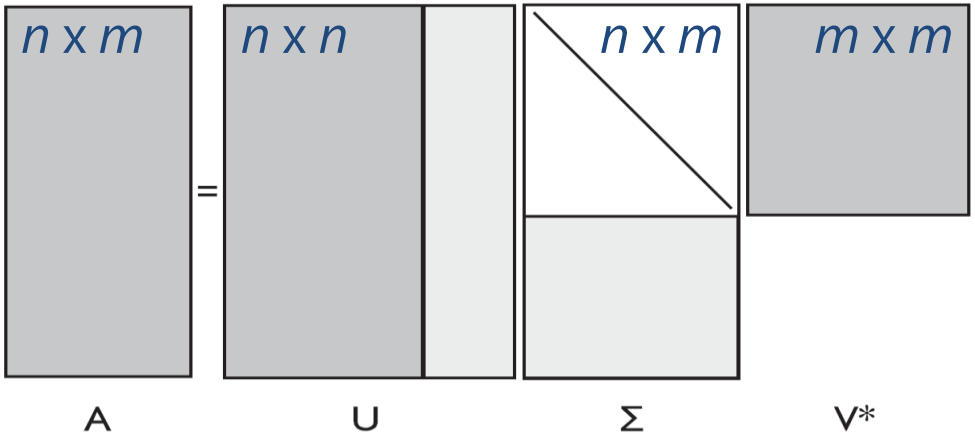
[**U**, **S**, **V**] = svd(**A**)

Notice the "silent" rows & columns in U and Σ:

$$\mathbf{A} = \mathbf{U}\,\mathbf{\Sigma}\,\mathbf{V}^*$$

$$= \begin{pmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \dots & \mathbf{u}_m & \underbrace{\mathbf{u}_{m+1} & \dots & \mathbf{u}_n}_{\text{silent columns}} \end{pmatrix} \begin{pmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_m \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix} \left.\begin{matrix} \\ \\ \\ \\ \\ \\ \\ \end{matrix}\right\} \begin{matrix} n-m \\ \text{silent} \\ \text{rows} \end{matrix} \begin{pmatrix} \mathbf{v}_1^\top \\ \mathbf{v}_2^\top \\ \vdots \\ \mathbf{v}_m^\top \end{pmatrix}$$

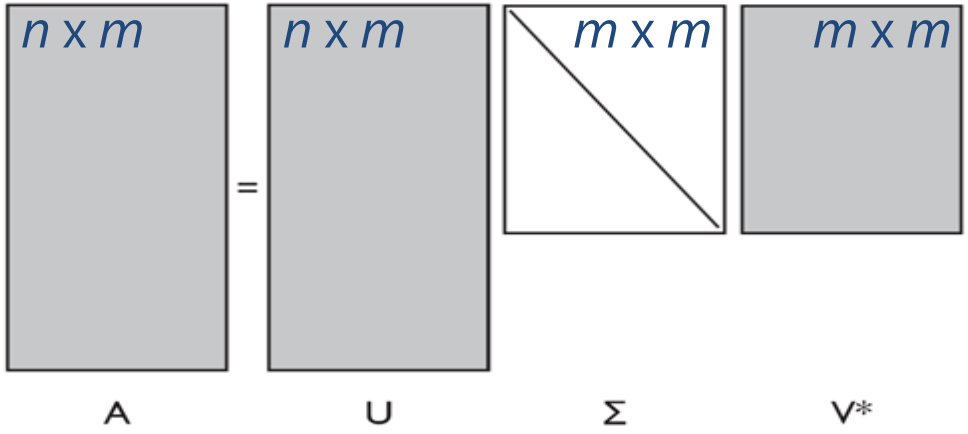
SVD – Illustration (case $n \geq m$)



this is also the decomposition that MatLab gives via call

[**U**, **S**, **V**] = svd(**A**)

Reduced SVD (without "silent" rows & columns):



SVD – Facts (1a)

If matrix A is of rank *r*, then the first *r* σ_j **are non-zero.**

Given rank(**A**)=*r*,

- range(**A**) = ($\mathbf{u}_1$, $\mathbf{u}_2$, …, $\mathbf{u}_r$)
- null(**A**) = ($\mathbf{v}_{r+1}$, $\mathbf{v}_{r+2}$, …, $\mathbf{v}_m$)

in MATLAB: „`range(A)`“ and „`null(A)`“

A's norm is given by the SVD:

$$||\mathbf{A}||_2 = \sigma_1 \quad \text{and} \quad ||\mathbf{A}||_F = \sqrt{\sum \sigma_k^2}$$

F … Frobenius norm

If A is square, then

$$|\det(\mathbf{A})| = \prod \sigma_k$$

Based on $\mathbf{A} = \sum \sigma_j \mathbf{u}_j \mathbf{v}_j^*$ **, A's best rank-*k* approximation is**

- with $||\mathbf{A} - \mathbf{A}_k||_2 = \sigma_{k+1}$

and $||\mathbf{A} - \mathbf{A}_k||_F = \sqrt{\sum_{j=k+1}^{r} \sigma_j^2}$

$$\mathbf{A}_k = \sum_{j=1}^{k} \sigma_j \mathbf{u}_j \mathbf{v}_j^*$$

[added later…]

Understanding Matrices

A matrix A $\in \mathbb{R}^{n \times m}$ (*n* rows, *m* columns)

- can be seen as an operation
  - **b** = **A x** (**A** acting upon vector **x**, resulting in vector **b**)
  - **Q** = **A P** (**A** acting upon matrix **P**, resulting in matrix **Q**)
- can also be thought of as operation, even if there is no argument
  - **A** = **A I** (**A** acting upon identity **I**, resulting in matrix **A**)

Interpretation of A as an operation:

- all that **A** does, is (a combination of)
  - some **rotation** (can include the flipping of axes)
  - some **scaling** (stretching/compression)
- that observation leads to interesting and useful decompositions like the SVD (and PCA, ...)

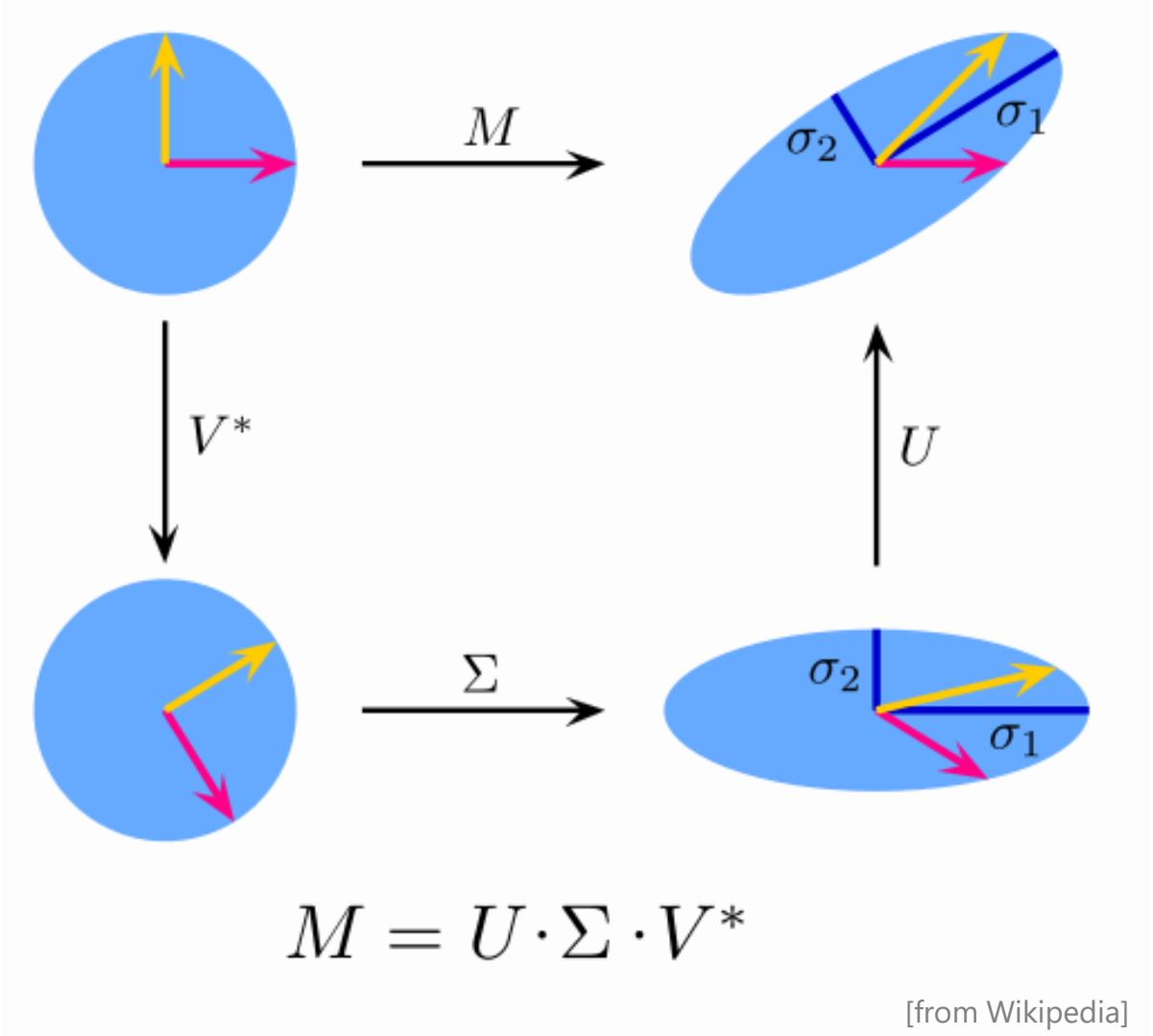
Understanding Matrices

A matrix A $\in \mathbb{R}^{n \times m}$ (*n* rows, *m* columns)

- can be seen as an
  - **b** = **A x** (**A** acti
  - **Q** = **A P** (**A** act
- can also be thoug
  - **A** = **A I** (**A** acti

Interpretation of A

- all that **A** does, is
  - some **rotation**
  - some **scaling** (s
- that observation le
  like the SVD (and



[from Wikipedia]

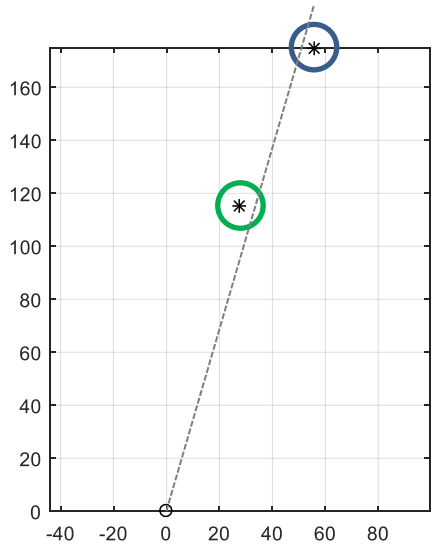
SVD, another Illustration

Given that A = U Σ V* (and **A** = **A I**),

- the SVD can be seen as a sequence of three operations on **I**:
  1. change of basis (from standard-coordinates to **V**), by left-multiplication with **V***
  2. axis-aligned scaling/stretching by left-multiplication with **Σ** (after this operation, the result is considered in **U**-coordinates!)
  3. change of basis (from **U**-coordinates to std.-coordinates), by left-multiplication with **U**

Example:

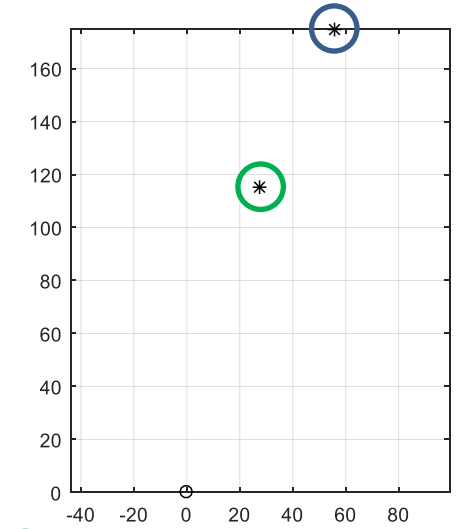
- assume **A** = $\begin{bmatrix} 55.5000 & 27.6600 \\ 175.0000 & 115.0000 \end{bmatrix}$
- almost on a line through the origin!
- i.e., the distance from the origin explains both variables simultaneously (almost 100%)



SVD, another Illustration (cont'd)

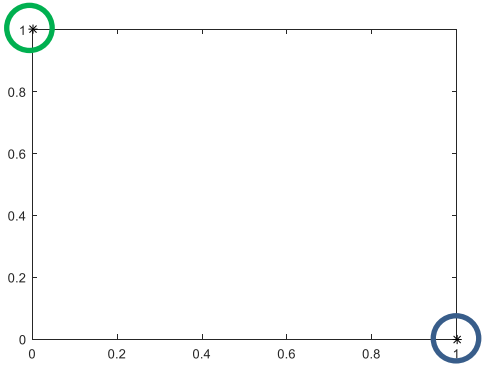
Given $\mathbf{A} = \begin{bmatrix} 55.5000 & 27.6600 \\ 175.0000 & 115.0000 \end{bmatrix}$,

- we can think of **A** as being constructed of two vectors $\mathbf{a}_1$ and $\mathbf{a}_2$



In terms of $\mathbf{A} = \mathbf{A}\,\mathbf{I}$ **and** $\mathbf{A} = \mathbf{U}\,\mathbf{\Sigma}\,\mathbf{V}^*$,

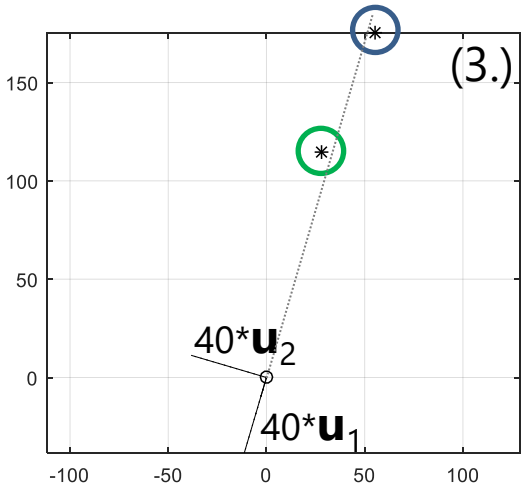
- we first compute the SVD with MatLab: [**U**, **Σ**, **V**] = svd(**A**):
  - $\mathbf{U} = \begin{bmatrix} -0.2824 & -0.9593 \\ -0.9593 & 0.2824 \end{bmatrix}$, $\mathbf{\Sigma} = \begin{bmatrix} 218.2783 & 0 \\ 0 & 7.0644 \end{bmatrix}$, $\mathbf{V} = \begin{bmatrix} -0.8409 & -0.5412 \\ -0.5412 & 0.8409 \end{bmatrix}$
- we can think of **A** being constructed as
  - starting from axis-aligned unit vectors $\mathbf{e}_1$ & $\mathbf{e}_2$ (in standard-coordinates)
  - and then "take them through operations" **V***, **Σ**, and **U**, because of $\mathbf{A} = \mathbf{A}\,\mathbf{I} = \mathbf{U}\,\mathbf{\Sigma}\,\mathbf{V}^*\,\mathbf{I}$

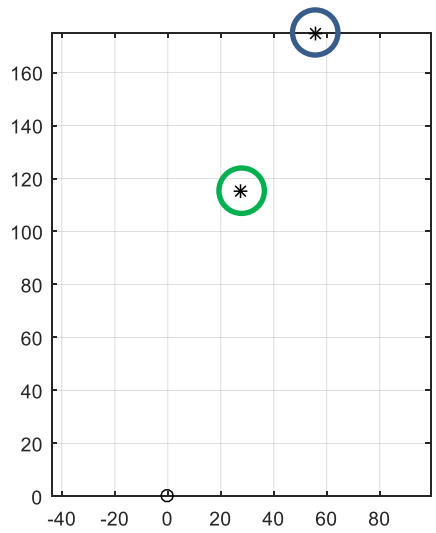


SVD, another Illustration (cont'd)

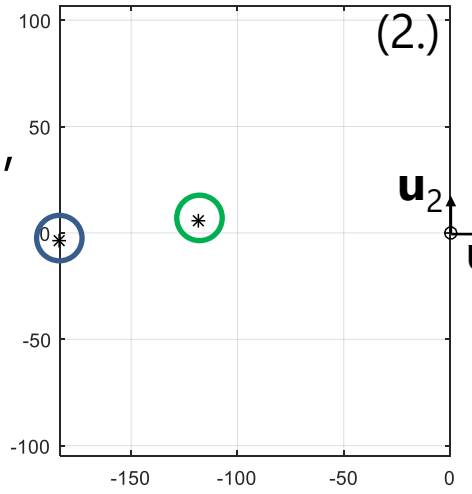
Given A = $\begin{bmatrix} 55.5000 & 27.6600 \\ 175.0000 & 115.0000 \end{bmatrix}$

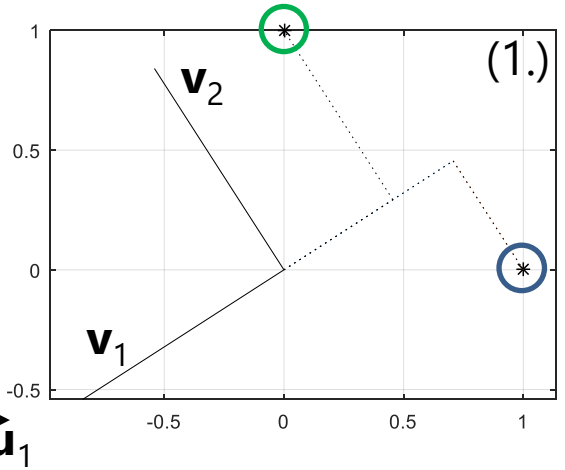
- and its decomposition
  - $\mathbf{U} = \begin{bmatrix} -0.2824 & -0.9593 \\ -0.9593 & 0.2824 \end{bmatrix}$, $\mathbf{\Sigma} = \begin{bmatrix} 218 \end{bmatrix}$
    $\mathbf{V} = \begin{bmatrix} -0.8409 & -0.5412 \\ -0.5412 & 0.8409 \end{bmatrix}$
- we do the following, starting (in standard-coordinates)





1. consider $\mathbf{e}_1$ and $\mathbf{e}_2$ in **V**-coordinates:
   $[\mathbf{e}_1]_\mathbf{V} = \mathbf{V}^* \mathbf{e}_1 = \begin{bmatrix} -0.8409 \\ -0.5412 \end{bmatrix}$, $[\mathbf{e}_2]_\mathbf{V} = \begin{bmatrix} -0.5412 \\ 0.8409 \end{bmatrix}$
2. scale them by **Σ**
   (result in **U**-coordinates!):
   $[\mathbf{a}_1]_\mathbf{U} = \mathbf{\Sigma}[\mathbf{e}_1]_\mathbf{V} = \begin{bmatrix} -183.5501 \\ -3.8232 \end{bmatrix}$,
   $[\mathbf{a}_2]_\mathbf{U} = \begin{bmatrix} -118.1304 \\ 5.9404 \end{bmatrix}$
3. transform them back into std.-coordinates





Interpreting U, Σ, and V

Given $\mathbf{A} \in \mathbb{R}^{n \times m}$ (m vectors $\mathbf{a}_i$, each n-dimensional, $m > n$)

- then, based on $\mathbf{A} = \mathbf{A}\,\mathbf{I} = \mathbf{U}\,\mathbf{\Sigma}\,\mathbf{V}^*\,\mathbf{I}$, the following interpretation is possible
  - for every vector $\mathbf{a}_i$, we start with $\mathbf{e}_i \in \mathrm{R}^m$ ($\mathbf{I} \in \mathrm{R}^{m \times m}$)
  - we then consider the $\mathbf{e}_i$ in **V**-coordinates: this takes care that the $\mathbf{e}_i$ get aligned to each other such that their structure in the first n dimensions conforms with the structure of **A** (not scaled, not rotated yet)
    - ex. 1: if all $\mathbf{a}_i$ are on a line in $\mathbb{R}^n$, then all $[\mathbf{e}_i]_{\mathbf{V}}$ will be on a line (1st $\mathbb{R}^n$), too
    - ex. 2: if all $\mathbf{a}_i$ are in a plane in $\mathbb{R}^n$, then all $[\mathbf{e}_i]_{\mathbf{V}}$ will be in a plane (1st $\mathbb{R}^n$), too
  - **Σ** then scales the vectors (per axis) to the scale of **A** (the first component will be scaled most, etc.)
  - using **U** to get back to std.-coordinates (**A**'s coordinates), the vectors get rotated (and flipped) into **A**'s place

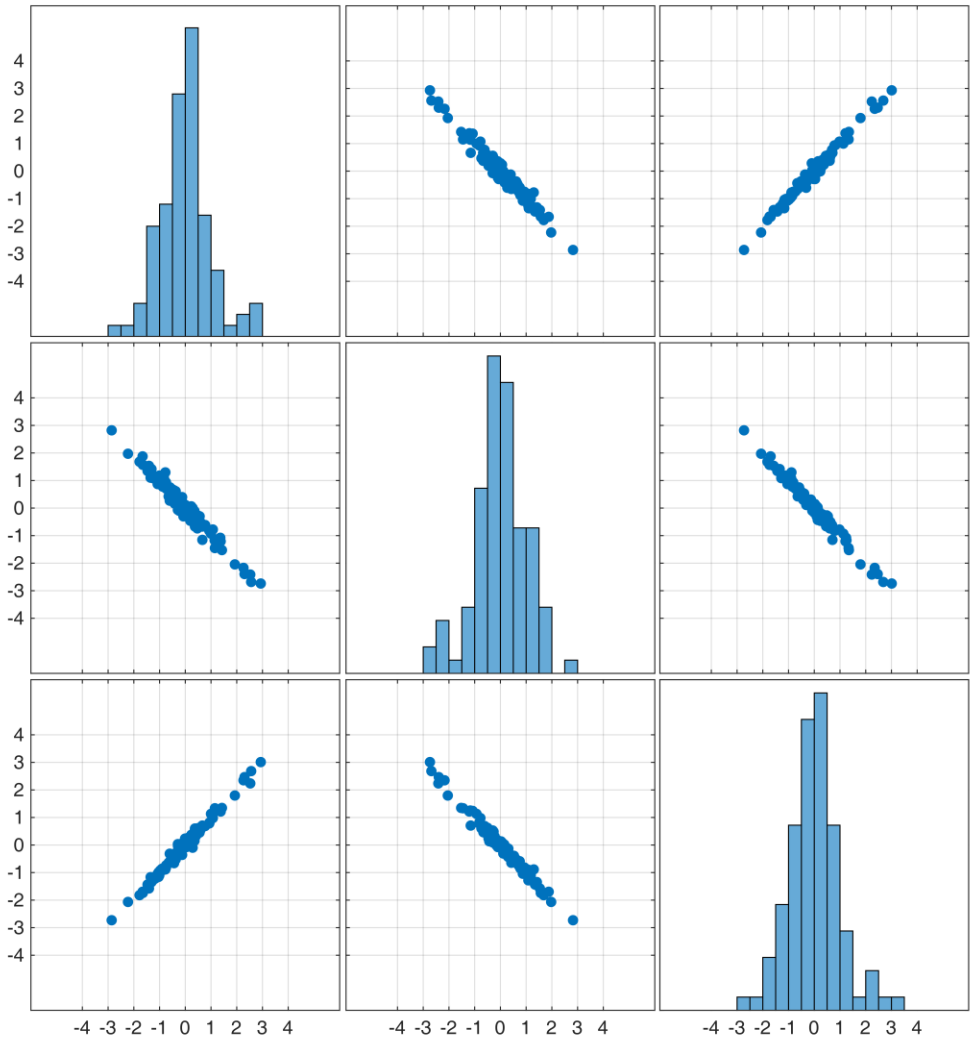
[see also the MATLAB example 2c_C]

SVD – Synthetic Example 1

[MATLAB example 2c_C]

A noisy line in 3D…

… quite obvious



SVD – Synthetic Example 1

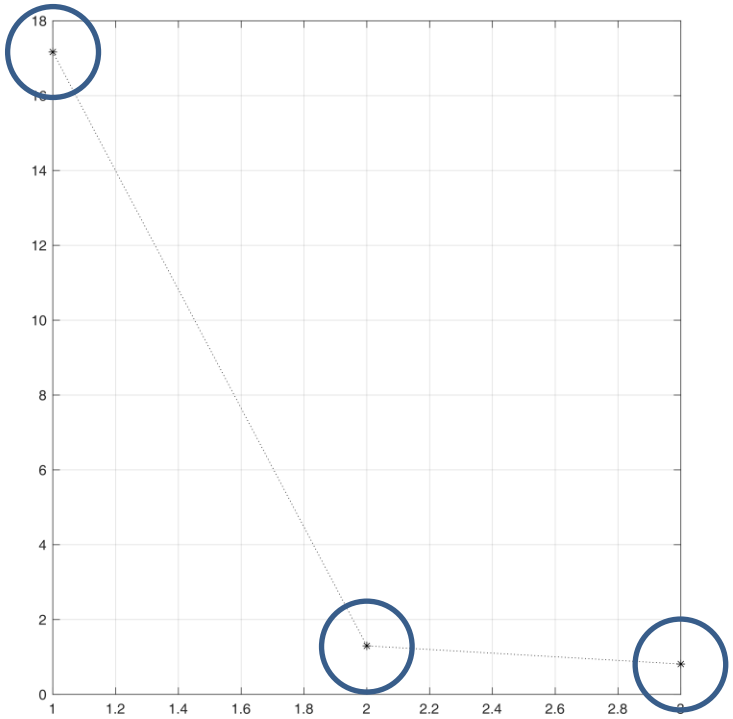
[MATLAB example 2c_C]

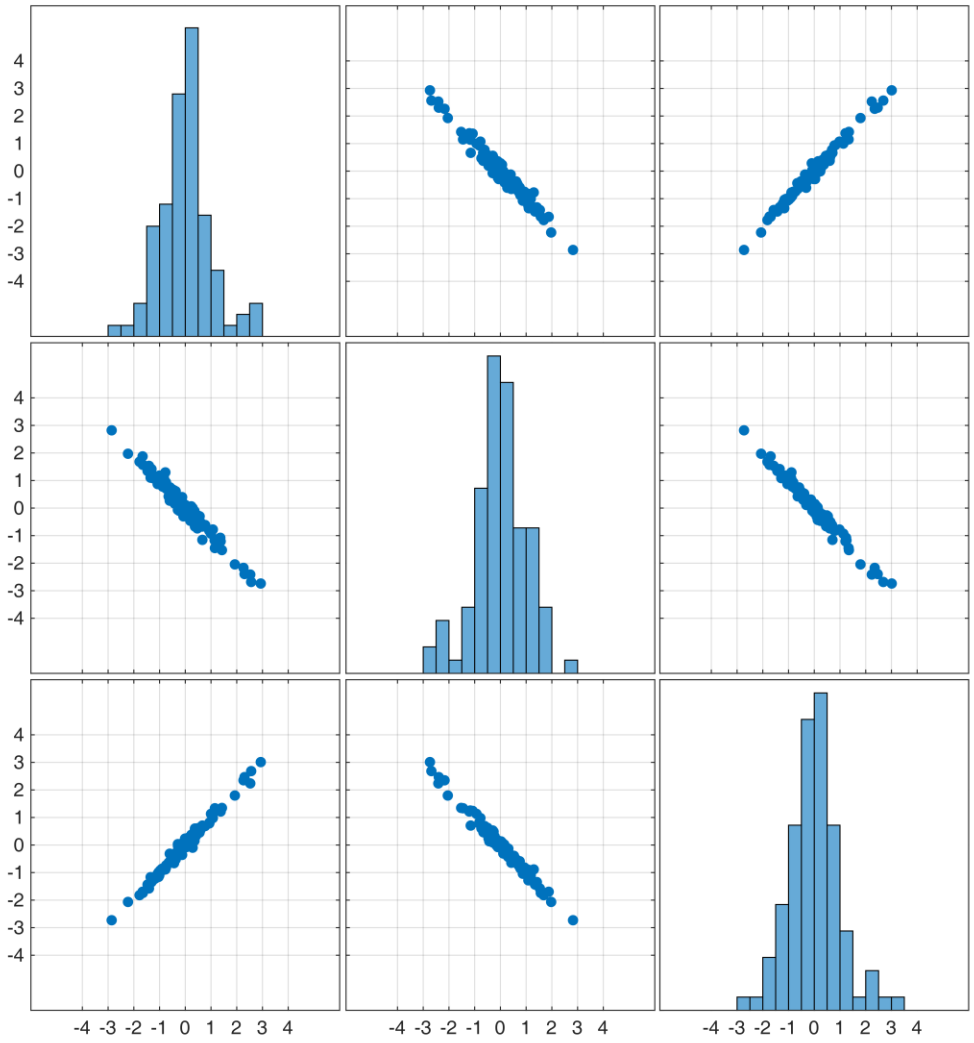
A noisy line in 3D…

… quite obvious

The singular values…

… equally clear!





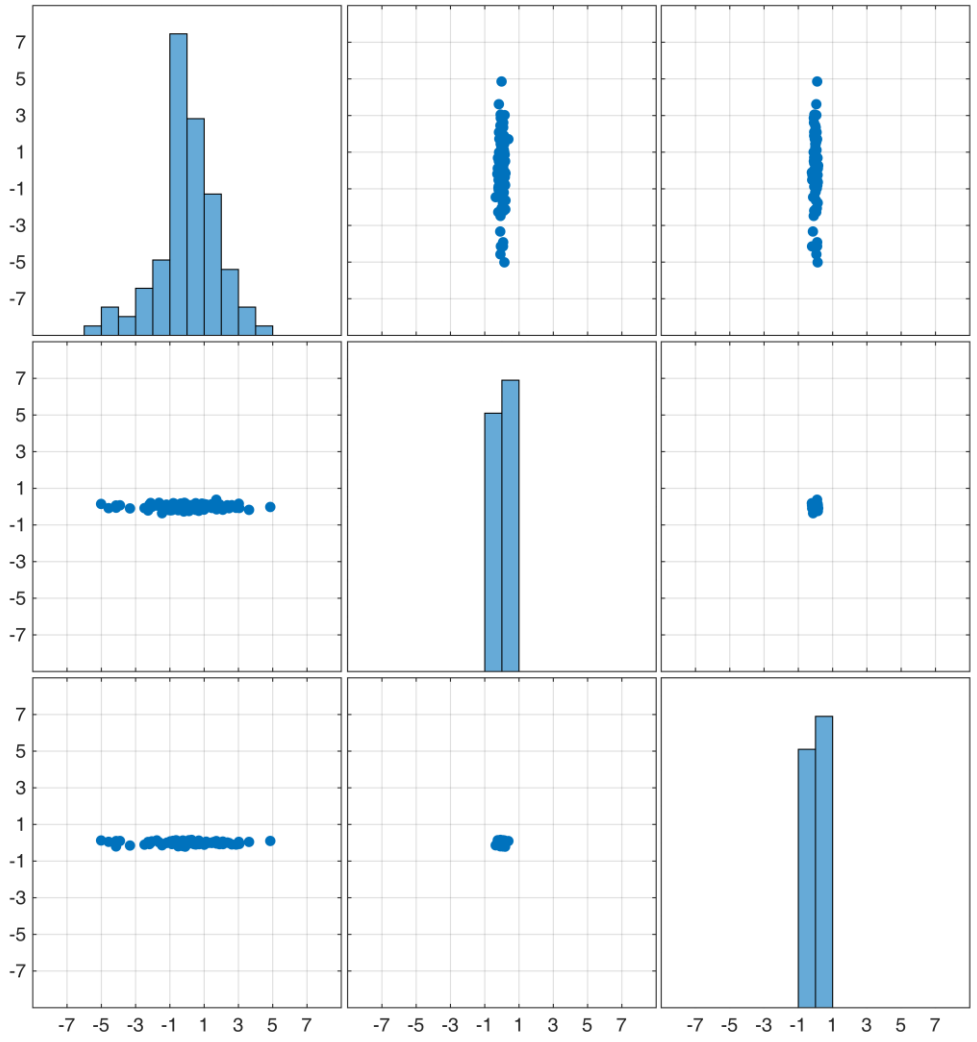
SVD – Synthetic Example 1

[MATLAB example 2c_C]

A noisy line in 3D…

… in **V** (after the SVD)

Even clearer! :–)

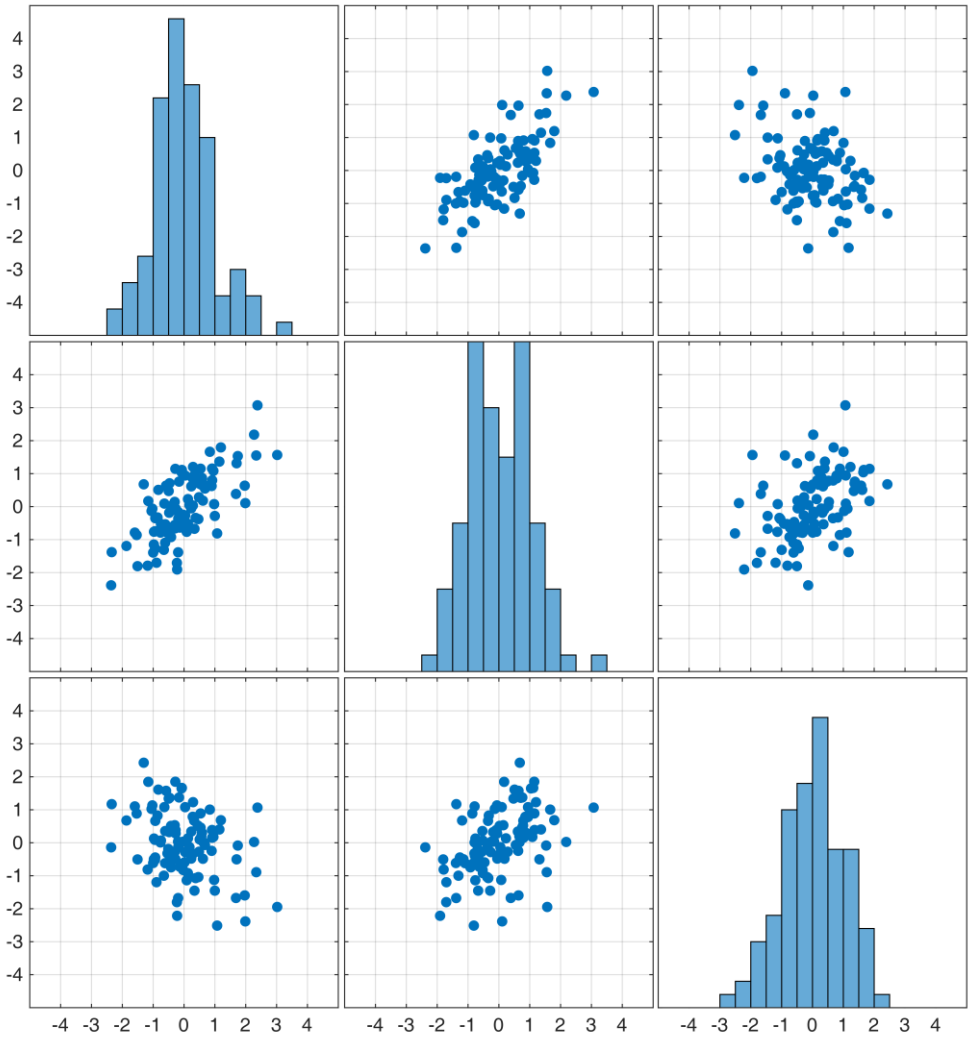


SVD – Synthetic Example 2

[MATLAB example 2c_C]

A noisy plane in 3D…

… not at all obvious!



SVD – Synthetic Example 2

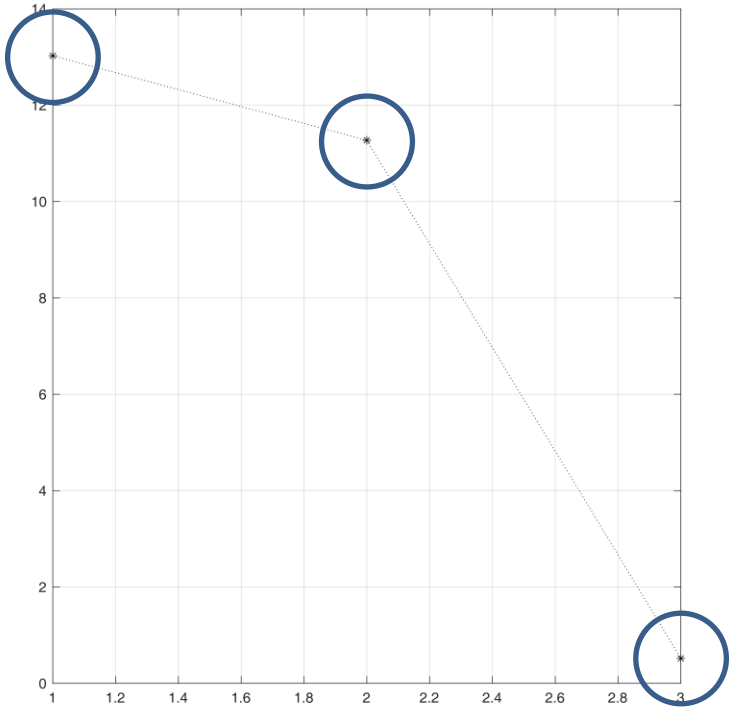
[MATLAB example 2c_C]

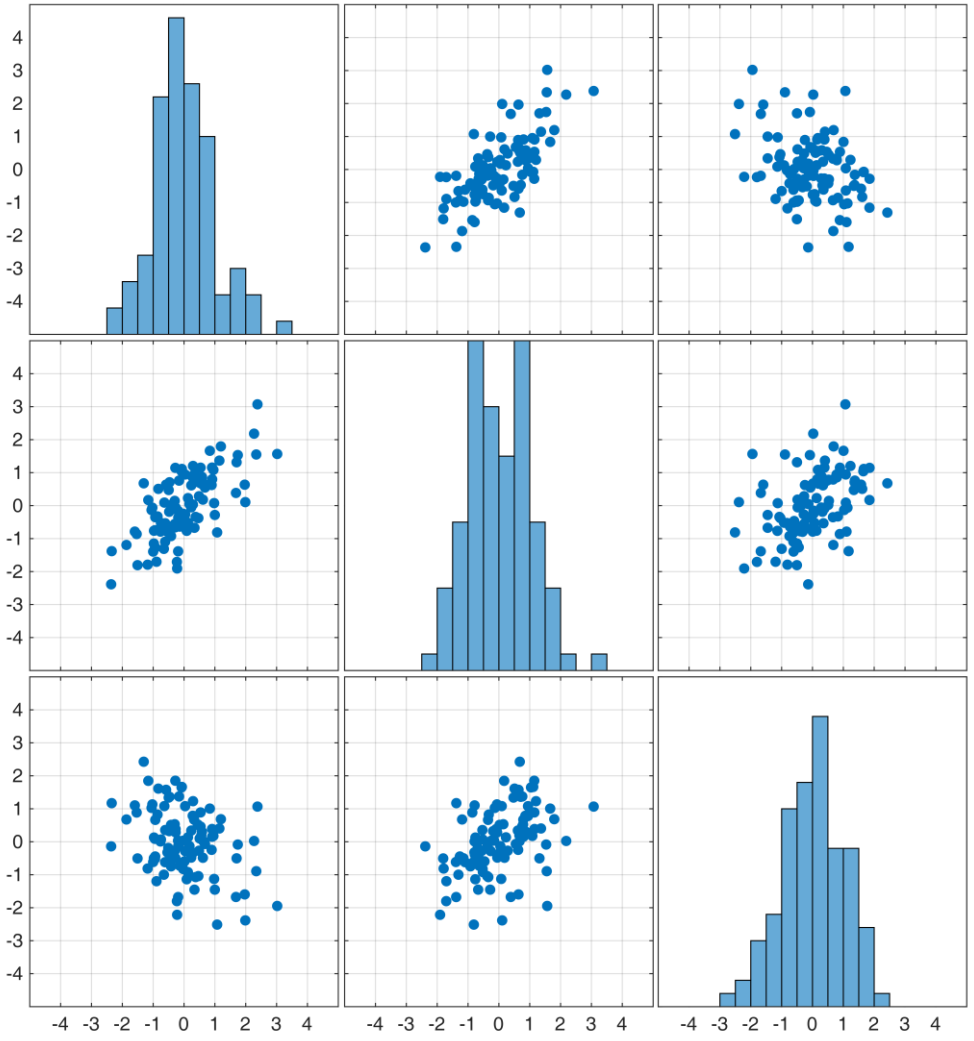
A noisy plane in 3D…

… not at all obvious!

The singular values…

… very clear!





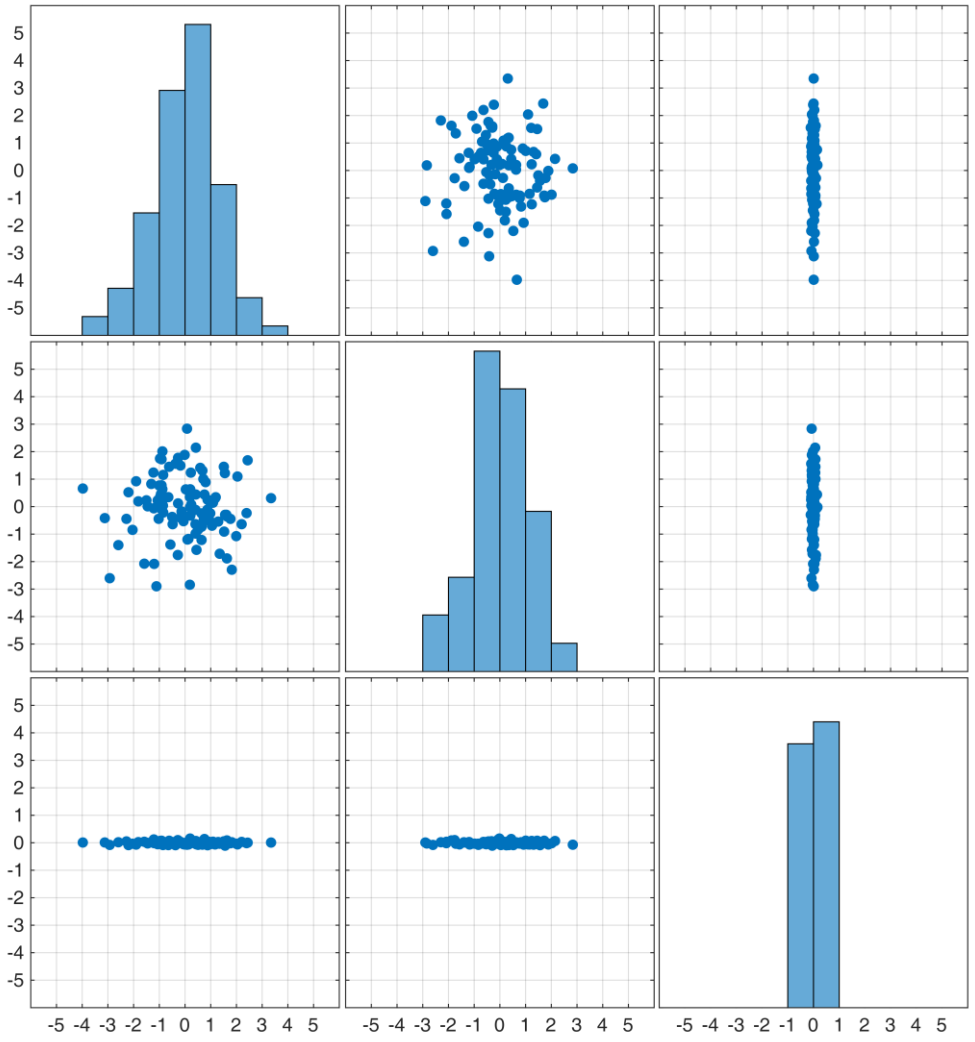
SVD – Synthetic Example 2

[MATLAB example 2c_C]

A noisy plane in 3D…

… in **V** (after the SVD)

Very clear! :–)



SVD – Example (1)

Assume, the following matrix A describes…

… the linear mapping **T** from $\mathbb{R}^2$ to $\mathbb{R}^2$: $\mathbf{A} = \begin{pmatrix} -0.4 & 1.2 \\ 1.2 & -1.1 \end{pmatrix}$

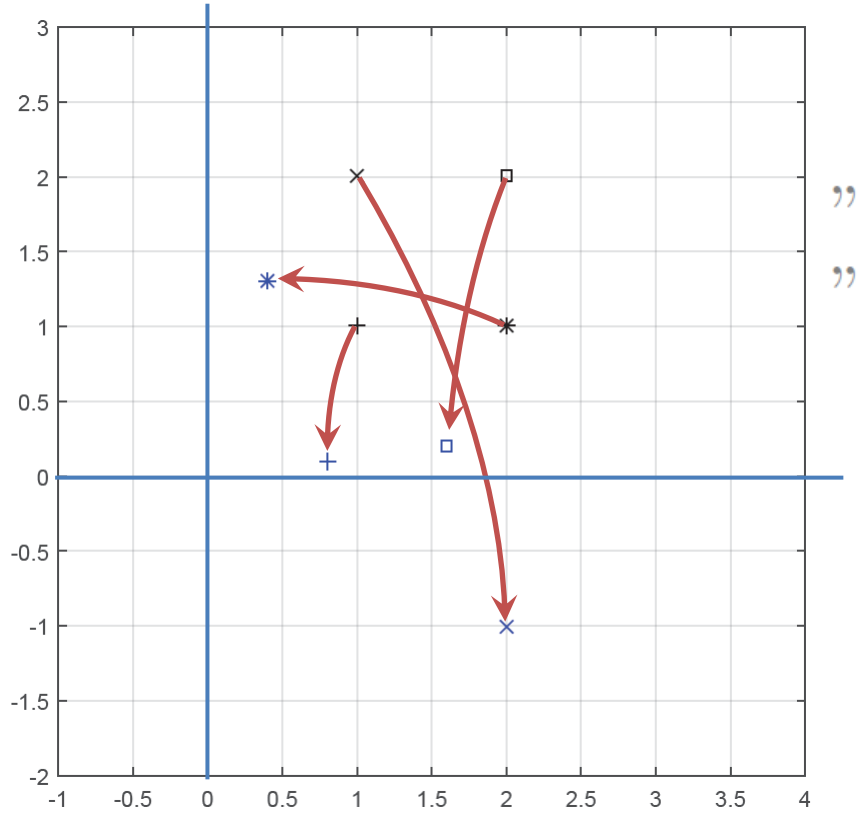
What does A do?

SVD – Example (2)

Assume, the following matrix A describes…

… the linear mapping **T** from $\mathbb{R}^2$ to $\mathbb{R}^2$: $\mathbf{A} = \begin{pmatrix} -0.4 & 1.2 \\ 1.2 & -1.1 \end{pmatrix}$

What does A do?



Can the SVD help?

„`A = [-0.4, 1.2; 1.2, -1.1]`“

„`[U, S, V] = svd(A)`“

leads to

$$\mathbf{U} = \begin{pmatrix} -0.6 & 0.8 \\ 0.8 & 0.6 \end{pmatrix}$$

$$\mathbf{\Sigma} = \begin{pmatrix} 2.0 & 0.0 \\ 0.0 & 0.5 \end{pmatrix}$$

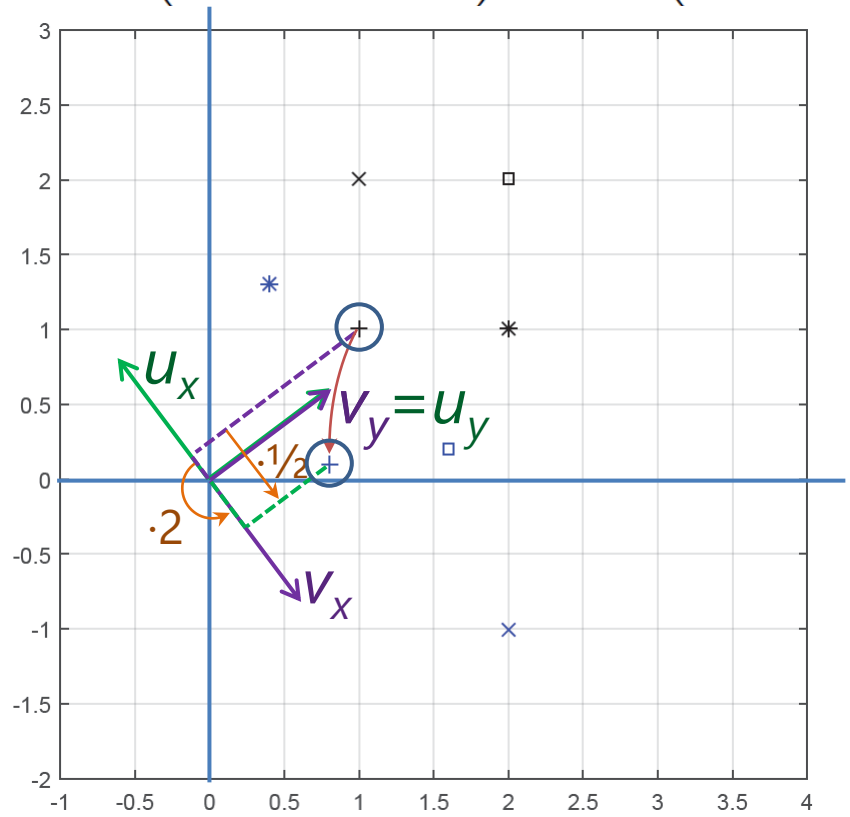
$$\mathbf{V} = \begin{pmatrix} 0.6 & 0.8 \\ -0.8 & 0.6 \end{pmatrix}$$

SVD – Example (3)

Assume, the following matrix A describes…

… the linear mapping **T** from $\mathbb{R}^2$ to $\mathbb{R}^2$: $\mathbf{A} = \begin{pmatrix} -0.4 & 1.2 \\ 1.2 & -1.1 \end{pmatrix}$

$$\mathbf{U} = \begin{pmatrix} -0.6 & 0.8 \\ 0.8 & 0.6 \end{pmatrix} \quad \mathbf{\Sigma} = \begin{pmatrix} 2.0 & 0.0 \\ 0.0 & 0.5 \end{pmatrix} \quad \mathbf{V} = \begin{pmatrix} 0.6 & 0.8 \\ -0.8 & 0.6 \end{pmatrix}$$



A … via V*, Σ, and U:

$$\mathbf{p}_+ = \begin{pmatrix} 1 \\ 1 \end{pmatrix}_{\mathcal{E}} = \begin{pmatrix} -0.2 \\ 1.4 \end{pmatrix}_{V}$$

$$\mathbf{T}(\mathbf{p}_+) = \begin{pmatrix} -0.4 \\ 0.7 \end{pmatrix}_{U} = \begin{pmatrix} 0.8 \\ 0.1 \end{pmatrix}_{\mathcal{E}}$$

SVD-steps:

1. represent in terms of **V**
2. axis-aligned scaling (x*2, y/2)
3. back to standard-coords. from **U**

result wrt. **U**!

Data Matrix A

Assume, we collected some data, for example:

- per subject (person): age, height, weight, etc.
- we can arrange the data into a data matrix:
  - either items into rows (and measures into columns)
  - or the other way around

Here (and often):

- data matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$ (usually $n \geq m$) represents
  - m measures (attributes, dimensions, etc.) – m columns
  - n data items (samples, data lines, etc.) – n rows
- **A** is, accordingly, composed of m-dimensional row-vectors!
- rows or columns? ... one has to think!

SVD($\mathbf{A}^T$) – now transposed

Given a data matrix $\mathbf{A}^T$, i.e.,

- with the data rows (instead of data columns)
  - usually, $\mathbf{A}^T$ has many more columns (items) than rows (dims.)
- also here the SVD of $\mathbf{A}^T$ is useful,
  we compute $\mathbf{A}^T = \mathbf{U}\,\mathbf{\Sigma}\,\mathbf{V}^*$, also!
- similarities:
  - **A** and $\mathbf{A}^T$ have the same singular values
- differences:
  - the matrices **U** and **V** of both SVD(**A**) and SVD($\mathbf{A}^T$)
    are each other's "inverses" (up to possible changes of the sign)
    - the **V** of SVD(**A**) is "almost" the inverse of **U** of SVD($\mathbf{A}^T$)
    - the **V** of SVD($\mathbf{A}^T$) is "almost" the inverse of **U** of SVD(**A**)
  - the silent rows & columns are in different places
    (silent columns in $\mathbf{\Sigma}$ and silent rows in $\mathbf{V}^*$)

Example: Hiking around Bergen etc.

[see also the MATLAB example 2c_A]

Data $D \in \mathbb{R}^{331\times 11}$:

- 331 hikes, 11 measures: **date**, **lon.** & **lat.**, **distance**, **daylight**, **length**, **hm up**, **effort-ind.**, **duration** , **speed**, **highest top**
- preprocessing: some normalization (to relate the measures)

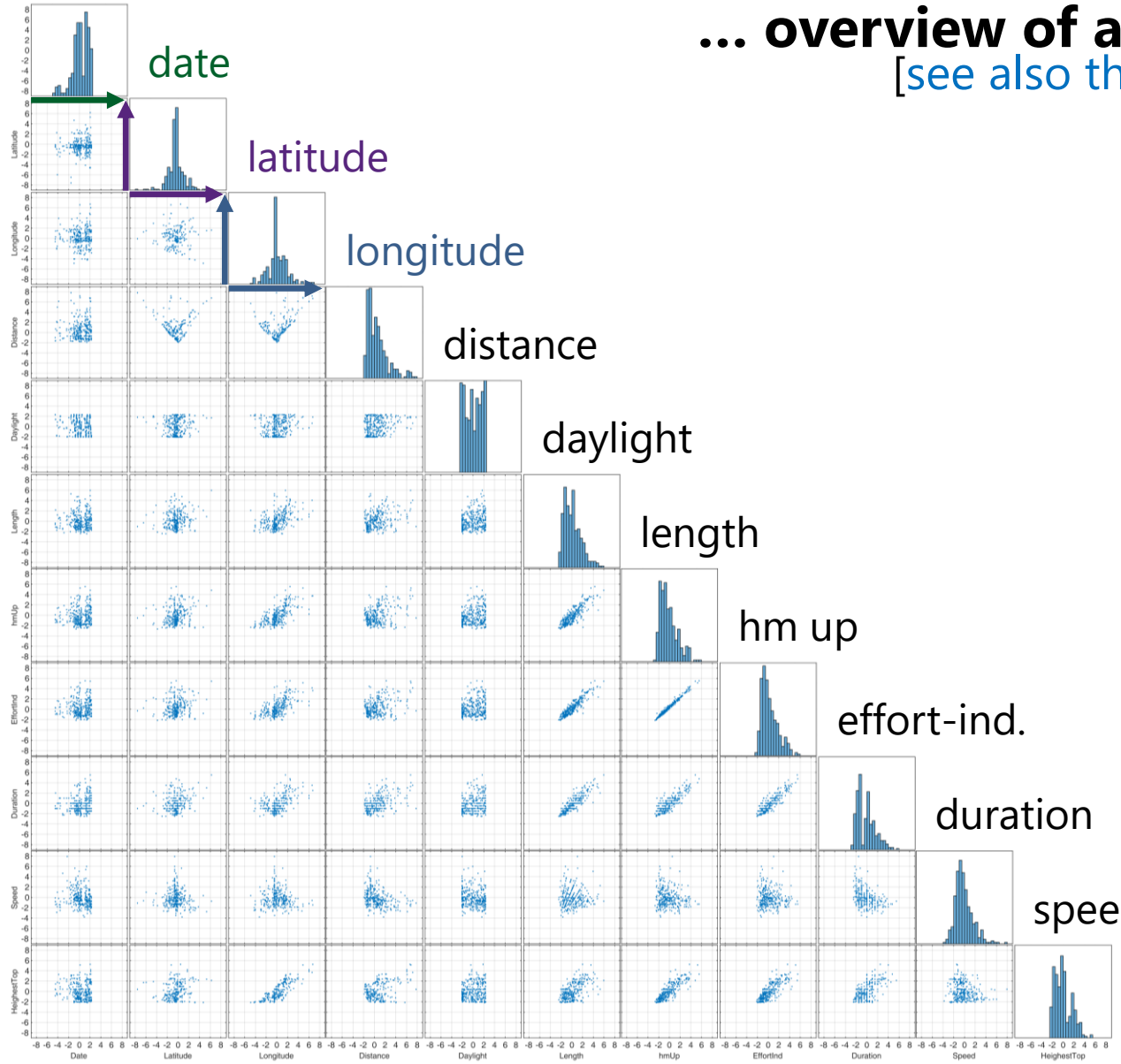
min.:	-4.51	-7.59	-4.88	-1.95	-2.09	-2.44	-2.68	-2.14	-2.50	-3.80	-2.17
q05:	-2.99	-2.37	-2.89	-1.46	-2.07	-1.78	-2.07	-1.52	-2.00	-2.40	-2.01
-2:	mid-2012	Bergen-10'	Bergen-15'	0km	6h	2.5km	150hm	0.06	1h	2km/h	50m
-1:	mid-2014	Bergen-5'	Bergen-7½'	10km	9h	5km	325hm	0.18	2h	2.5km/h	225m
0:	mid-2016	Bergen	Bergen	20km	12h	7.5km	500hm	0.30	3h	3km/h	400m
1:	mid-2018	Bergen+5'	Bergen+7½'	30km	15h	10km	675hm	0.42	4h	3.5km/h	575m
2:	mid-2020	Bergen+10'	Bergen+15'	40km	18h	12.5km	850hm	0.54	5h	4km/h	750m
4:	mid-2024	Bergen+20'	Bergen+30'	60km	24h	17.5km	1200hm	0.78	7h	5km/h	11050m
q95:	2.15	2.17	2.82	4.38	2.28	3.02	3.16	3.42	3.00	3.20	2.68
max.:	2.27	6.23	6.71	7.79	2.36	5.96	5.55	5.53	5.50	7.93	5.26

ID	Date	Latitude	Longitude	Distance	Daylight	Length	hmUp	EffortInd	Duration	Speed	HeighestTop
42	-4.51	-0.42	-0.62	-1.02	2.33	-0.64	-0.79	-0.38	-1.00	-0.10	-0.82
43	-4.51	-0.08	-0.10	-1.66	2.30	-0.40	0.08	0.32	-0.50	-0.80	0.44
44	-4.45	-0.19	0.31	-1.63	1.43	2.16	-0.91	0.26	2.00	-0.84	1.45
45	-4.45	-0.58	1.44	0.69	1.37	-1.16	-1.42	-0.98	-1.00	-1.40	0.36
47	-4.24	-0.94	-1.95	0.37	-1.86	-1.12	-1.19	-0.80	-1.00	-1.30	-0.34
48	-4.23	2.29	-2.07	2.14	-1.73	-1.76	-0.95	-0.79	-1.00	-2.90	-0.43
49	-4.14	-2.06	-1.54	1.95	0.17	-1.32	-1.41	-1.00	-1.00	-1.80	-0.66
50	-4.11	-5.96	1.50	4.96	0.86	1.20	1.34	1.66	0.00	1.00	1.77
53	-3.87	-1.53	3.94	6.11	-0.28	0.48	1.47	1.56	0.00	-0.20	3.29
54	-3.74	-0.90	0.94	-0.17	-1.88	0.68	-0.18	0.41	1.00	-1.40	1.62
57	-3.65	-2.43	0.71	0.97	-0.08	-1.36	-1.33	-0.96	-1.00	-1.90	-0.48
58	-3.59	1.17	0.04	-0.69	1.21	-0.92	-0.57	-0.29	0.00	-2.53	0.49
59	-3.59	0.42	-0.16	-1.51	1.36	0.64	0.61	0.97	1.00	-1.45	0.90
60	-3.56	1.29	2.74	2.41	1.91	0.24	0.03	0.45	1.00	-1.95	2.41
61	-3.55	-1.60	0.30	-0.18	2.09	-0.40	-1.23	-0.64	-0.50	-0.80	-0.50
64	-3.11	-0.36	-0.13	-1.24	0.74	-1.56	-0.83	-0.65	-2.00	1.20	0.44
65	-3.11	-0.42	-0.42	-1.15	0.92	-1.40	-1.97	-1.44	-2.25	4.67	-1.60
70	-2.87	-0.35	-0.64	-1.13	-0.39	-2.24	-1.90	-1.61	-2.25	-0.93	-1.19
71	-2.86	-1.25	1.81	2.07	-0.45	1.60	1.86	2.14	1.00	-0.25	2.53
73	-2.38	-0.14	1.55	0.38	-0.20	0.12	0.10	0.47	0.00	-0.80	1.95
74	-2.37	-0.59	1.46	0.71	-0.38	0.88	1.10	1.39	1.00	-1.15	2.33
75	-2.26	-0.63	-0.46	-0.72	-2.07	0.32	-1.14	-0.39	-1.20	3.22	-1.06
76	-2.25	-0.63	-0.46	-0.72	-2.05	1.04	0.54	1.03	-1.55	7.93	0.44
77	-2.24	-0.63	-0.46	-0.72	-1.94	-0.64	-1.41	-0.83	-1.50	1.87	-1.43
82	-1.96	0.25	1.25	0.52	1.75	0.80	0.94	1.26	1.00	-1.25	1.10
86	-1.83	-1.82	1.79	1.73	-1.20	-0.64	-0.18	0.06	-1.50	1.87	0.72
87	-1.79	-0.68	-1.63	0.02	-1.88	-0.68	-1.22	-0.70	-1.00	-0.20	-0.40
95	-1.68	-0.18	0.28	-1.65	-0.76	-0.32	0.43	0.60	-1.50	2.93	1.39
96	-1.68	-0.41	-0.13	-1.23	-0.73	-2.04	-1.12	-0.98	-2.00	-1.20	0.44
97	-1.68	-0.02	-0.78	-0.80	-0.70	-0.76	-0.65	-0.30	-1.50	1.47	-0.02

we can't see much from "just" the data...

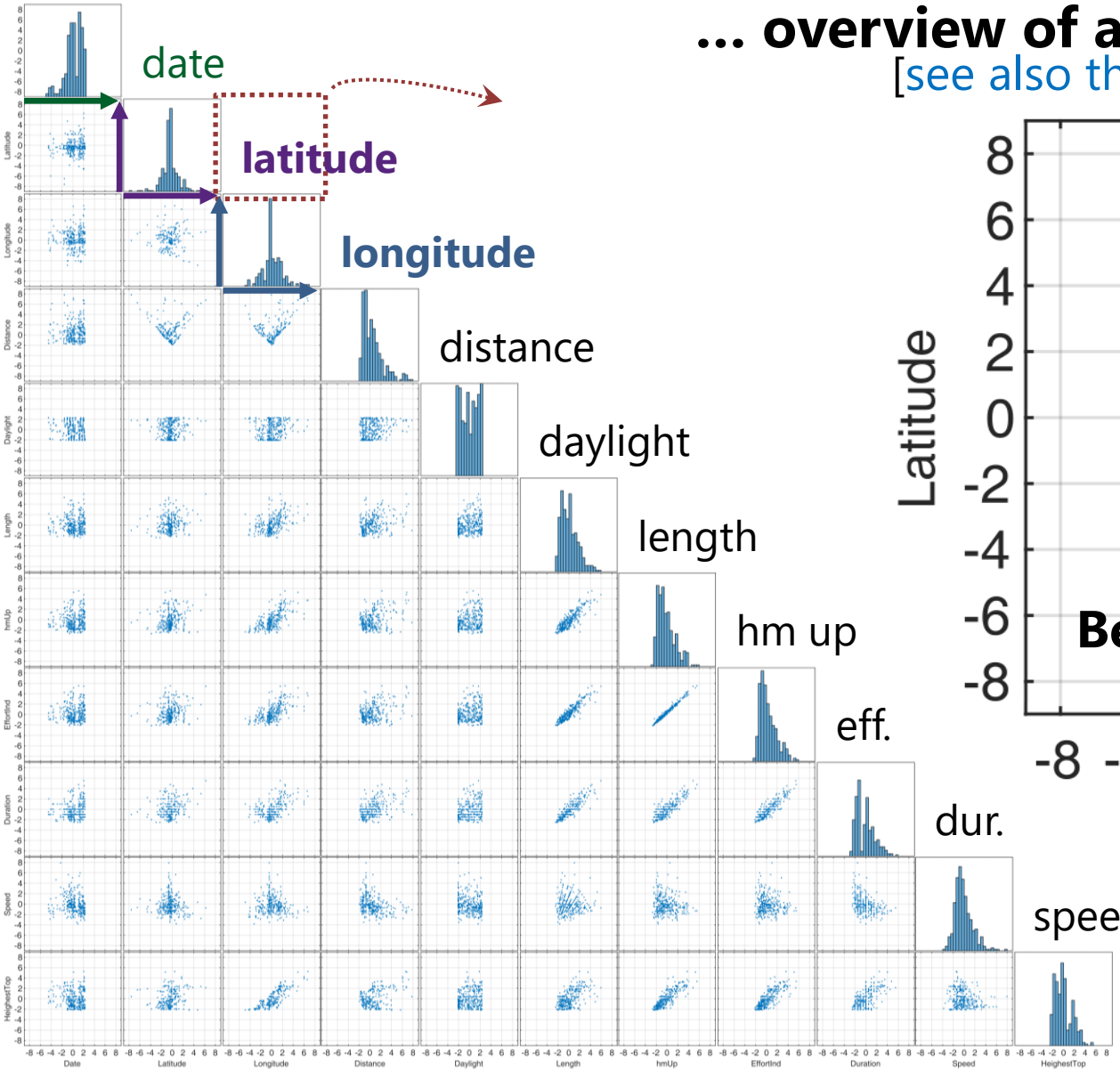
Example: Hiking around Bergen etc.

… overview of all bivariate relations
[see also the MATLAB example 2c_A]



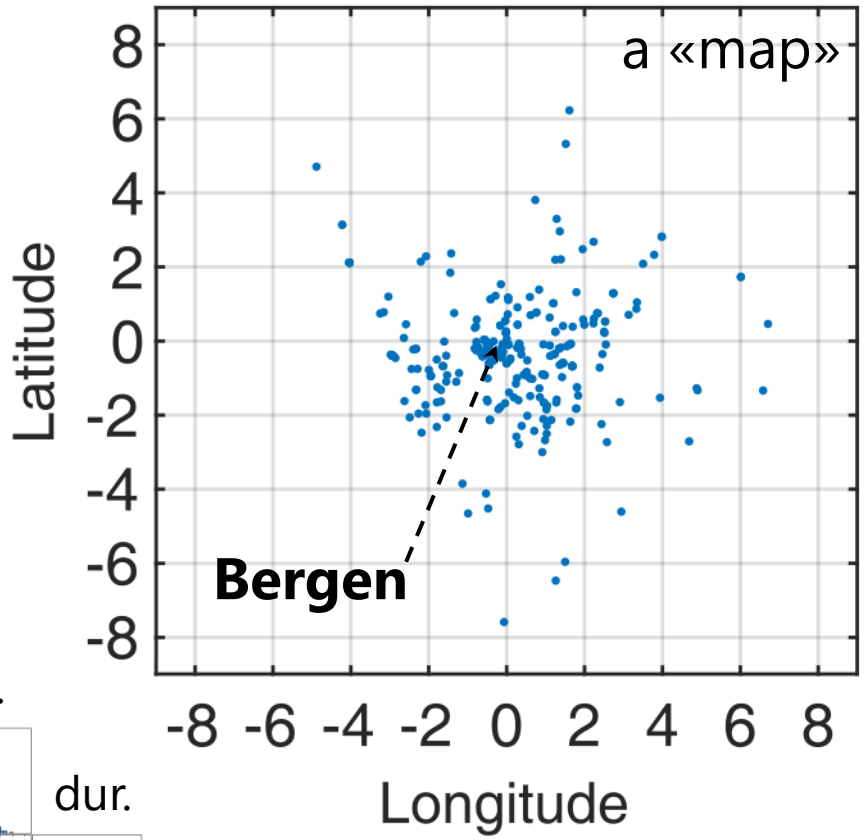
any insight?

Example: Hiking around Bergen etc.

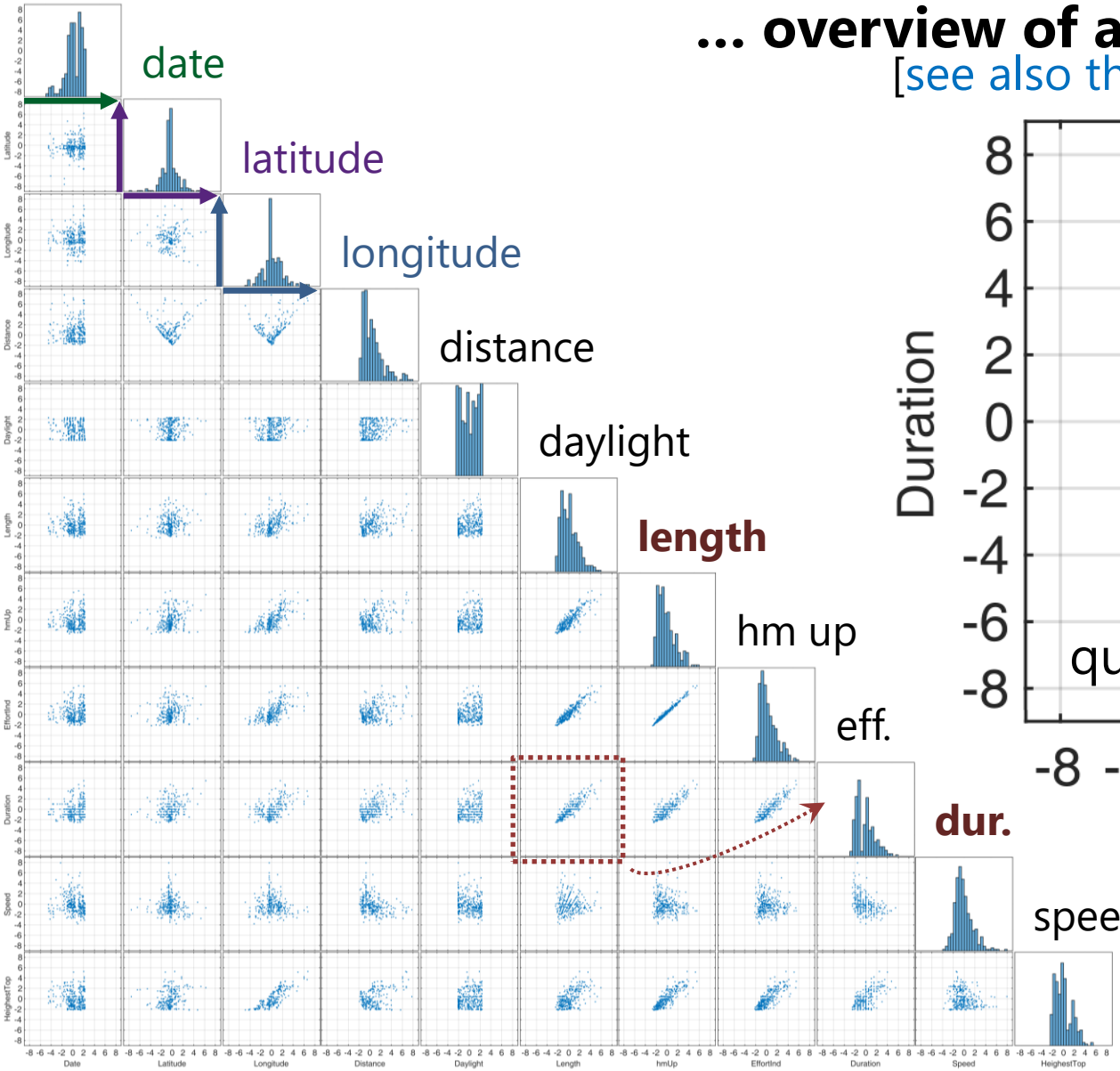


… overview of all bivariate relations

[see also the MATLAB example 2c_A]

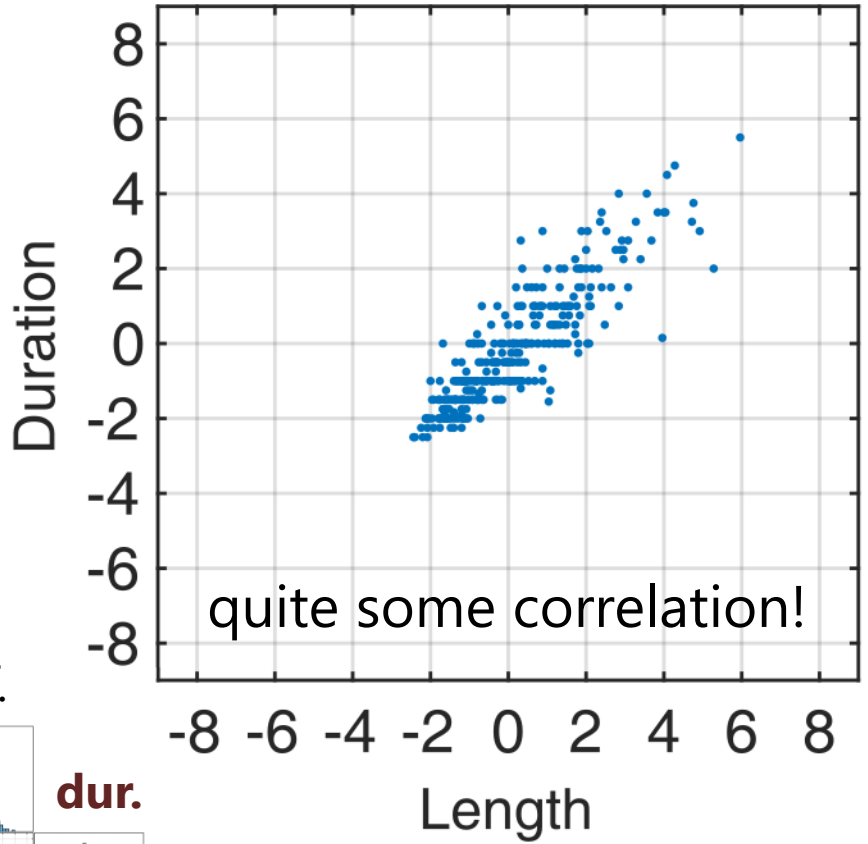


Example: Hiking around Bergen etc.



… overview of all bivariate relations

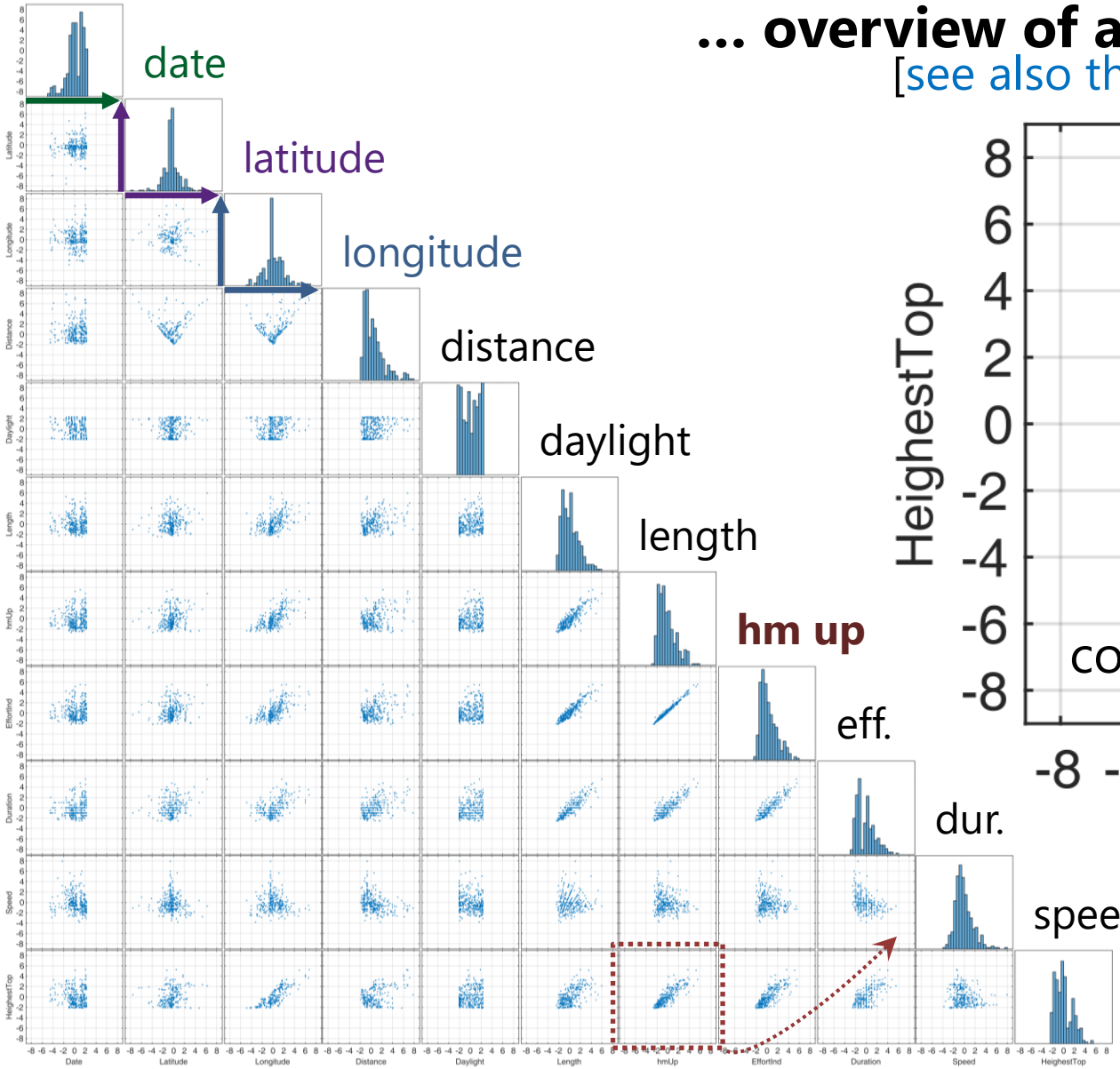
[see also the MATLAB example 2c_A]

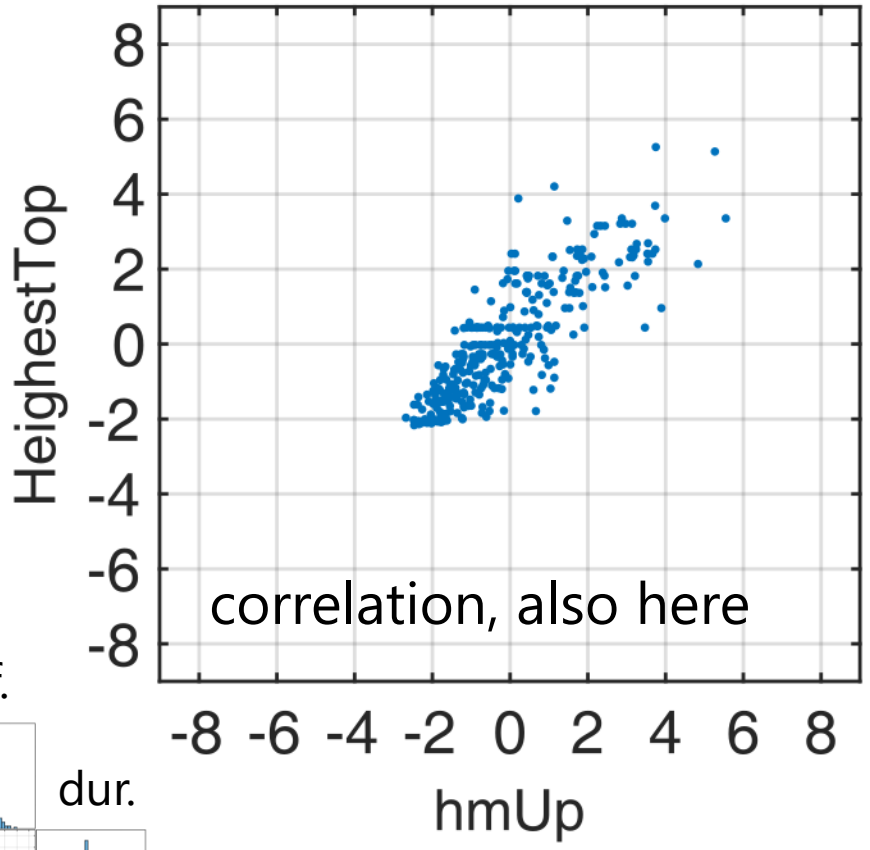


Example: Hiking around Bergen etc.

… overview of all bivariate relations

[see also the MATLAB example 2c_A]

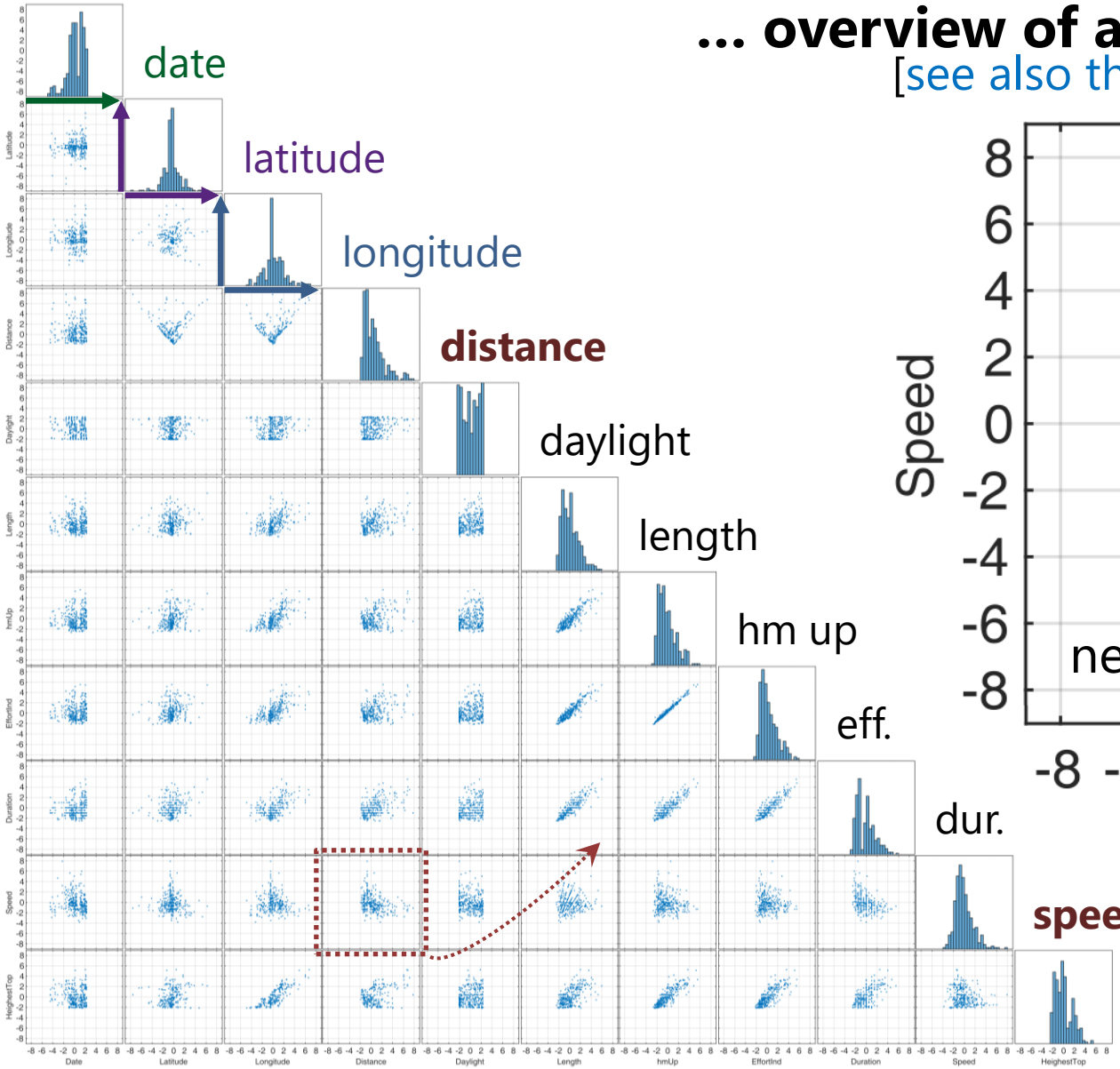


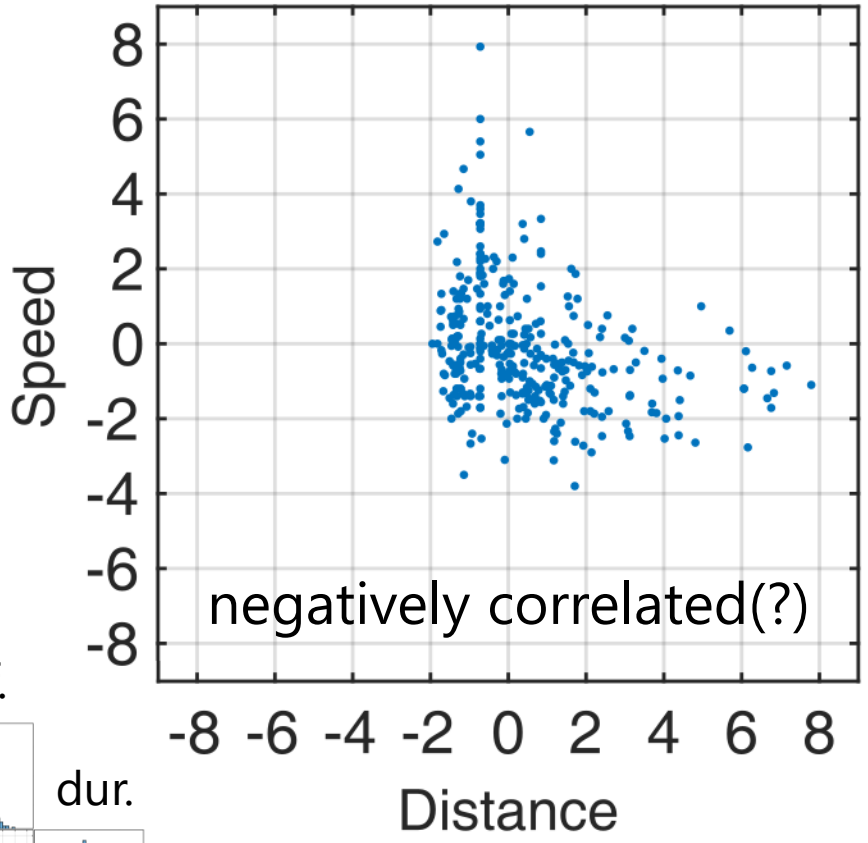


Example: Hiking around Bergen etc.

… overview of all bivariate relations

[see also the MATLAB example 2c_A]

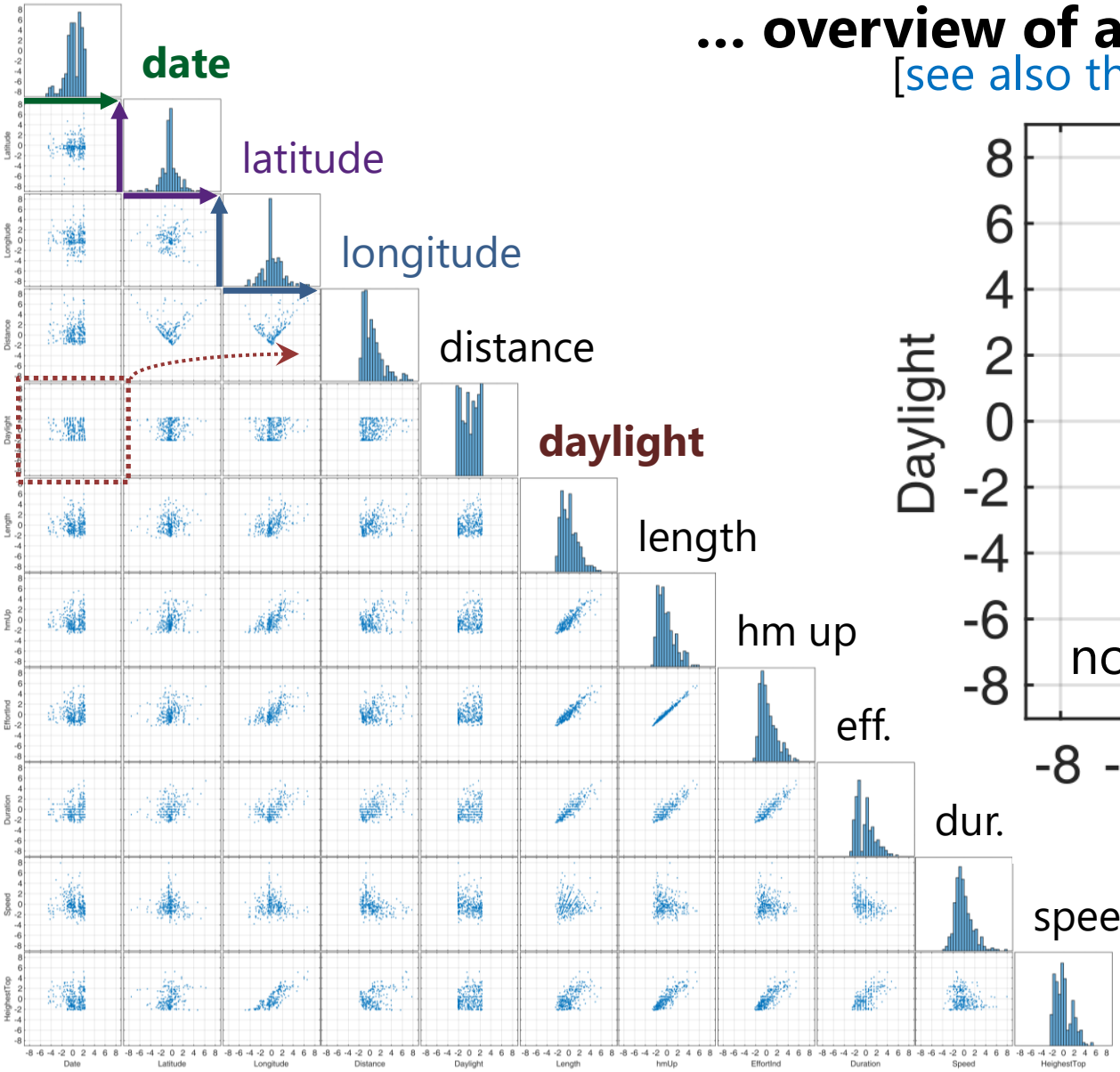


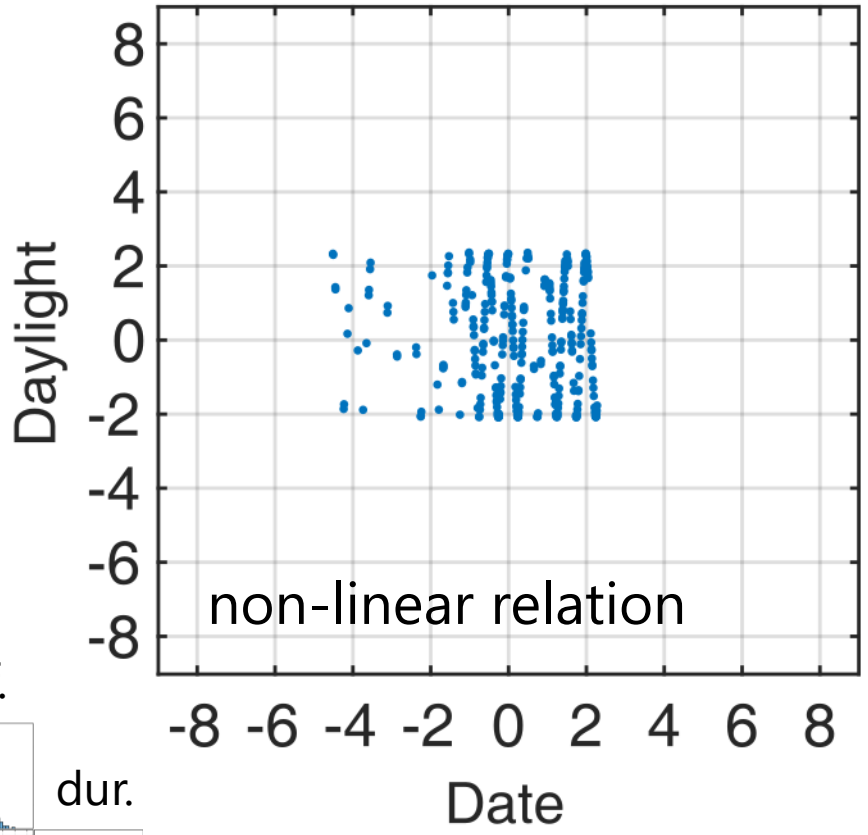


Example: Hiking around Bergen etc.

… overview of all bivariate relations

[see also the MATLAB example 2c_A]



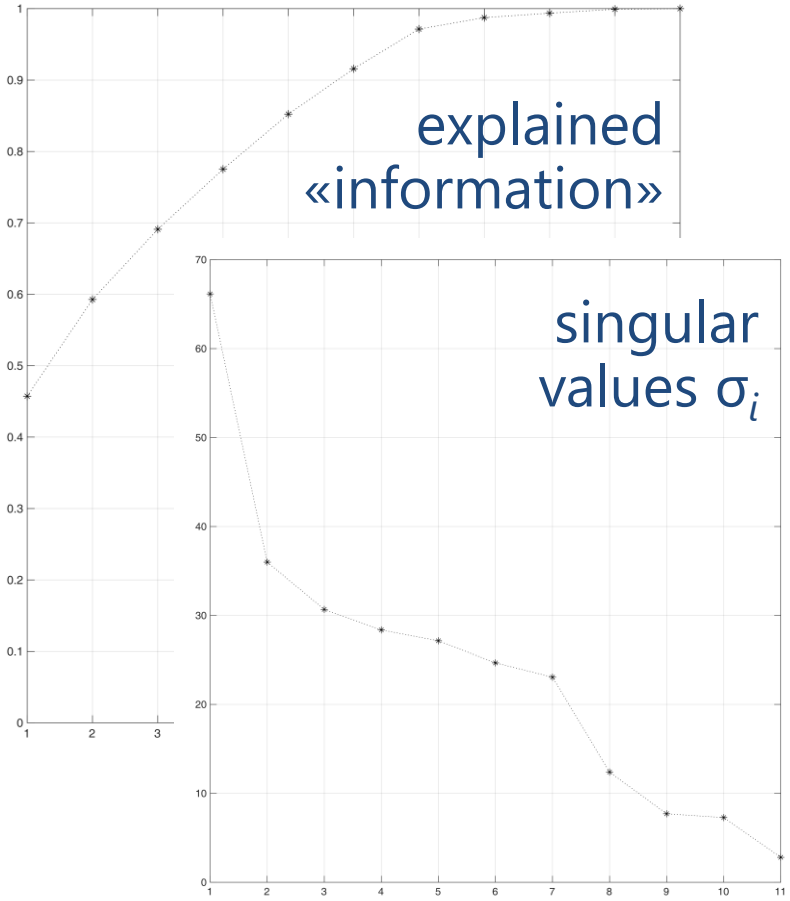


Example: Hiking around Bergen etc.

[see also the MATLAB example 2c_A]

Let's use the SVD:

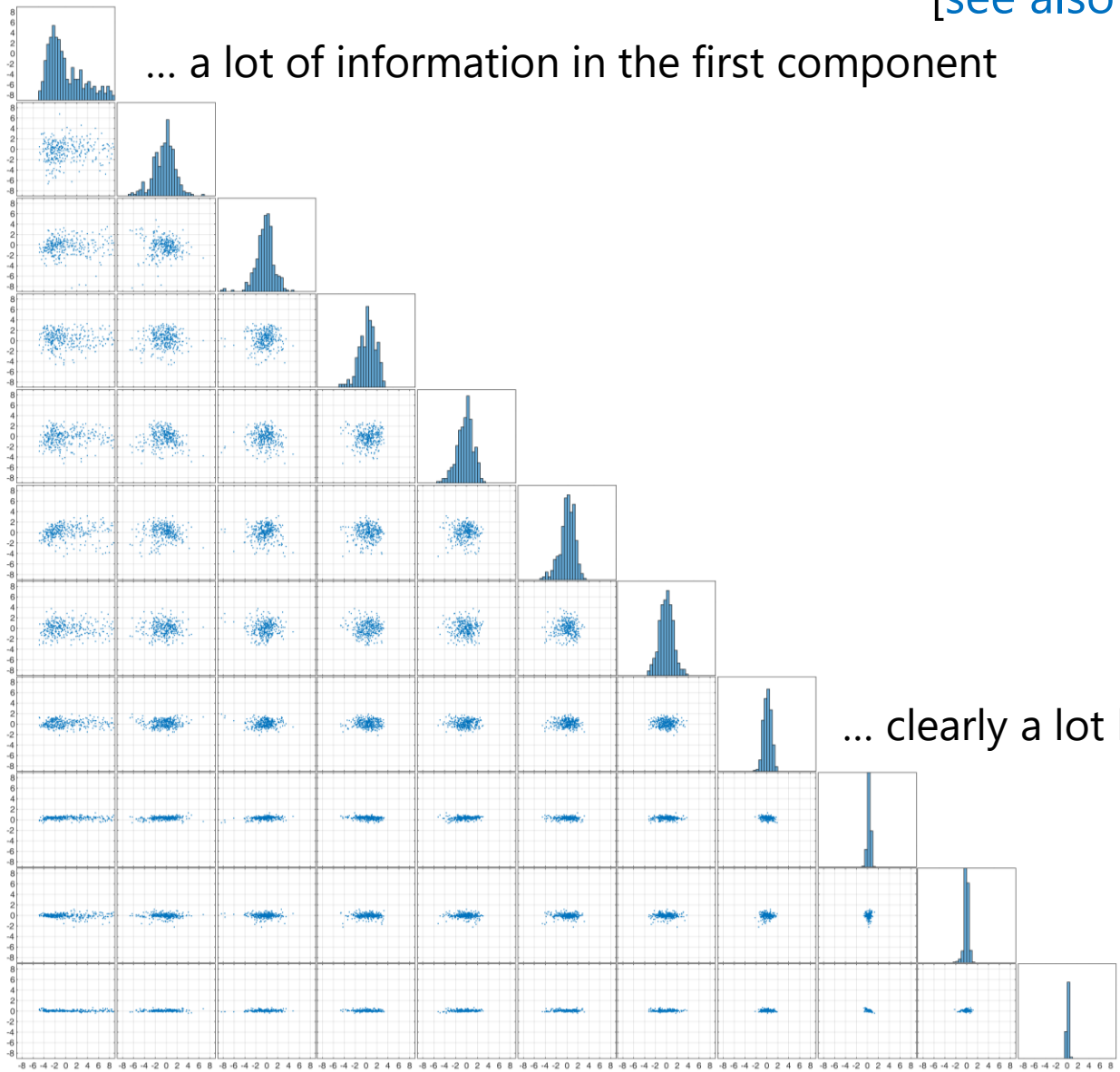
- [**U**, **Σ**, **V**] = svd(**D**) leads to $\mathbf{U} \in \mathbb{R}^{331\times331}$, $\mathbf{\Sigma} \in \mathbb{R}^{331\times11}$, $\mathbf{V} \in \mathbb{R}^{11\times11}$
- since **D** hosts row vectors, we "construct" **D** from the left: **D** = **I U Σ V***
- first, we look at the singular values using plot(diag(**Σ**), ...)
  - σ_1 about twice as large as σ_2
  - about 45% "energy" in c_1 (σ_1^2), ~60% in c_1+c_2 (σ_1^2 +σ_2^2), >90% in c_1...c_6
  - how many dimensions to represent **D** well?



Example: Hiking around Bergen etc.

[see also the MATLAB example 2c_A]

D in **V**-coordinates



… a lot of information in the first component

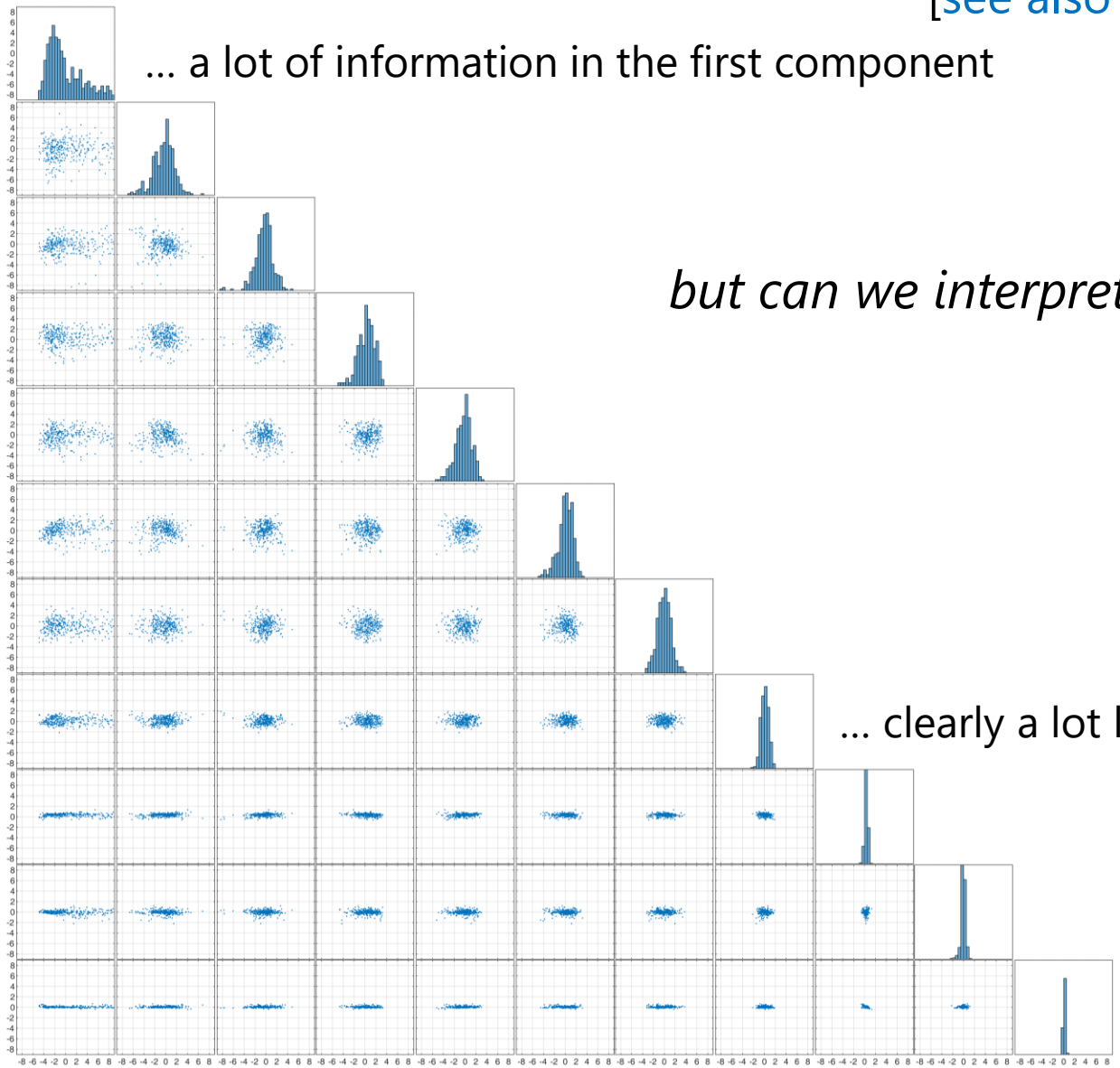
… clearly a lot less information

… very little information in the last component

Example: Hiking around Bergen etc.

[see also the MATLAB example 2c_A]

D in **V**-coordinates



… a lot of information in the first component

but can we interpret the new axes?

… clearly a lot less information

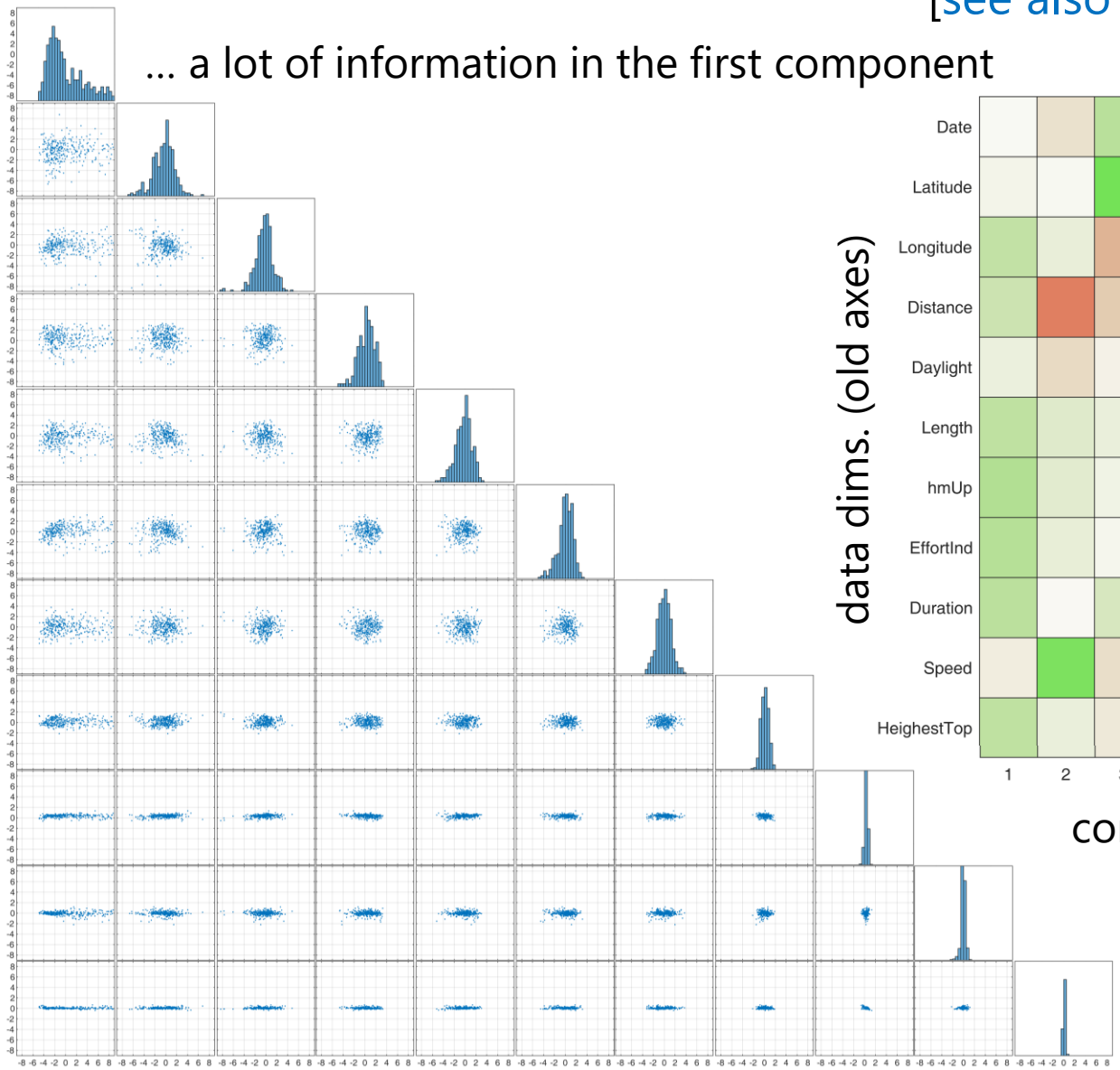
… very little information in the last component

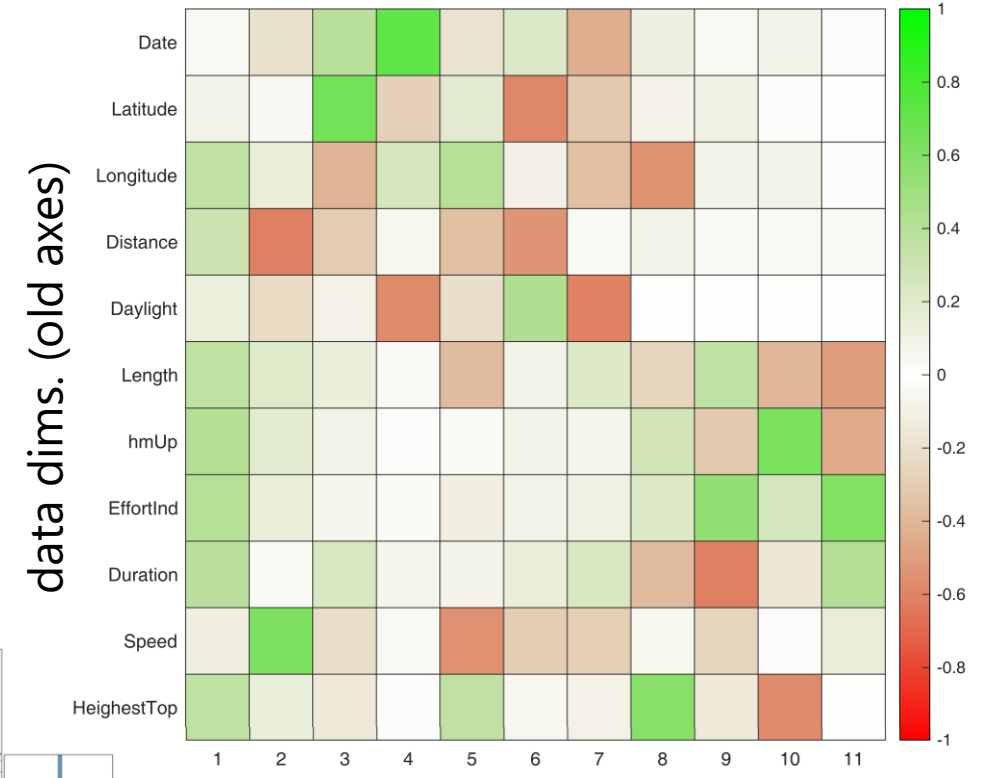
Example: Hiking around Bergen etc.

[see also the MATLAB example 2c_A]

… a lot of information in the first component

we look at **V**!

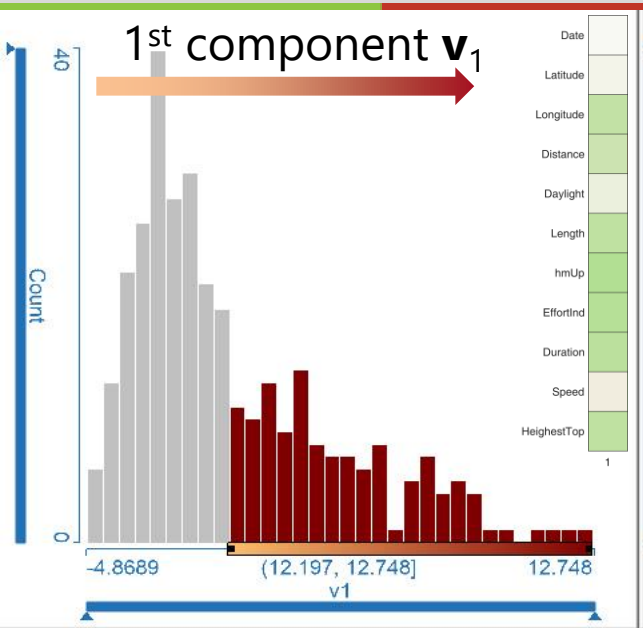


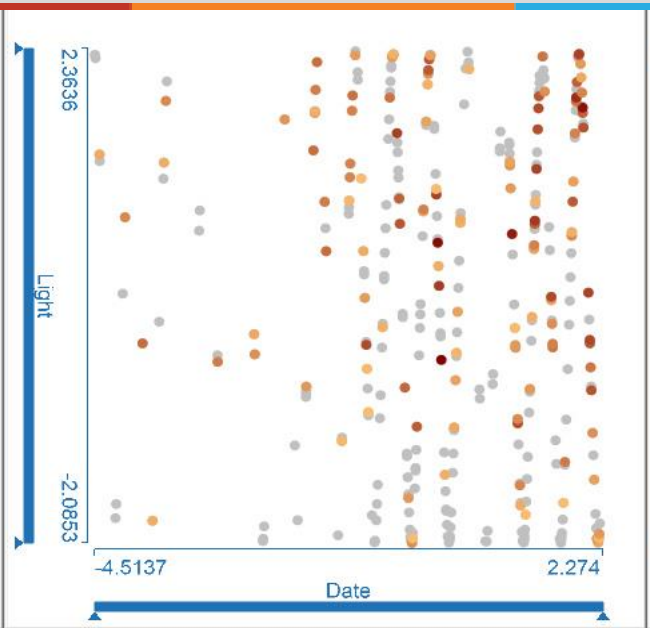


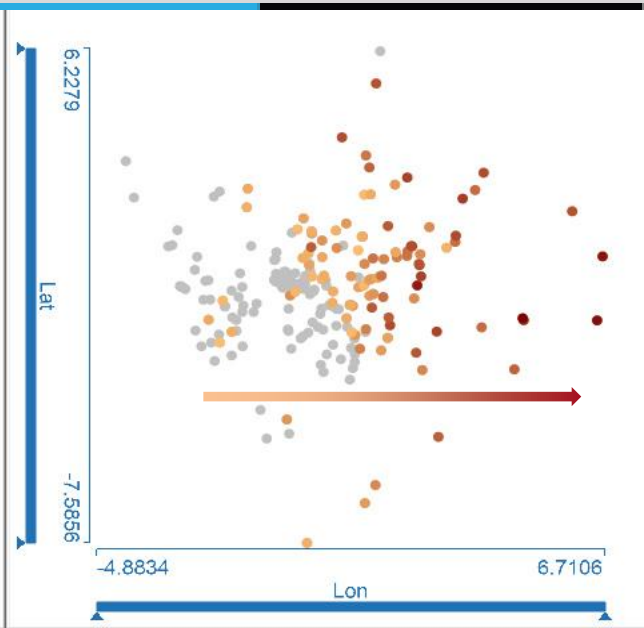
contributions:

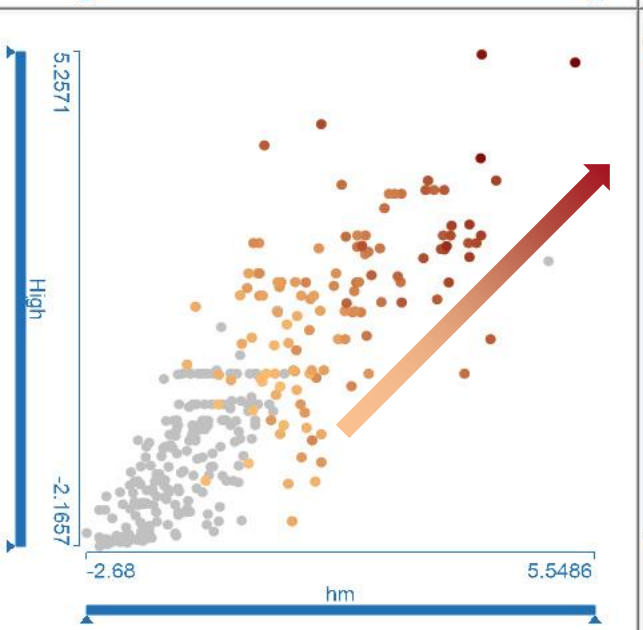
- **green**: positively
- **red**: negatively
- white: none

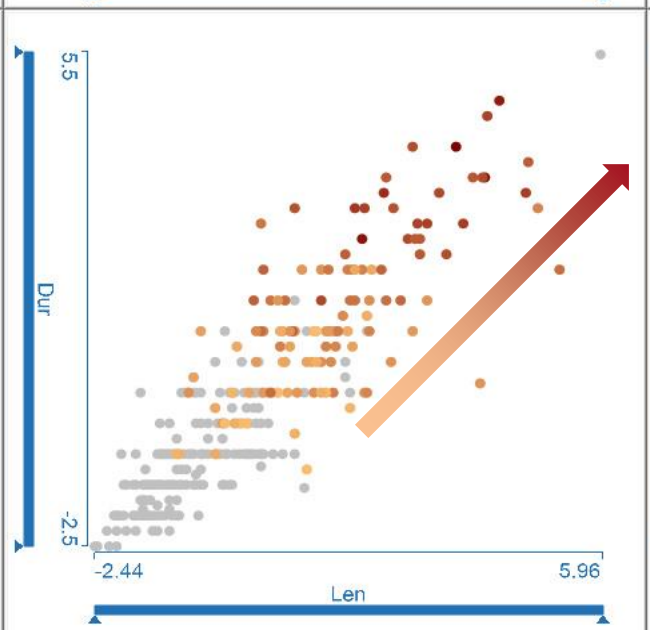
Example: Hiking around Bergen etc.

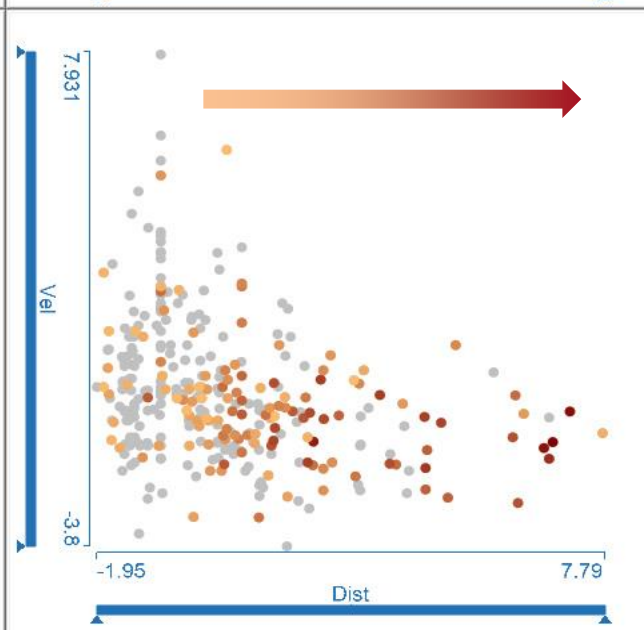




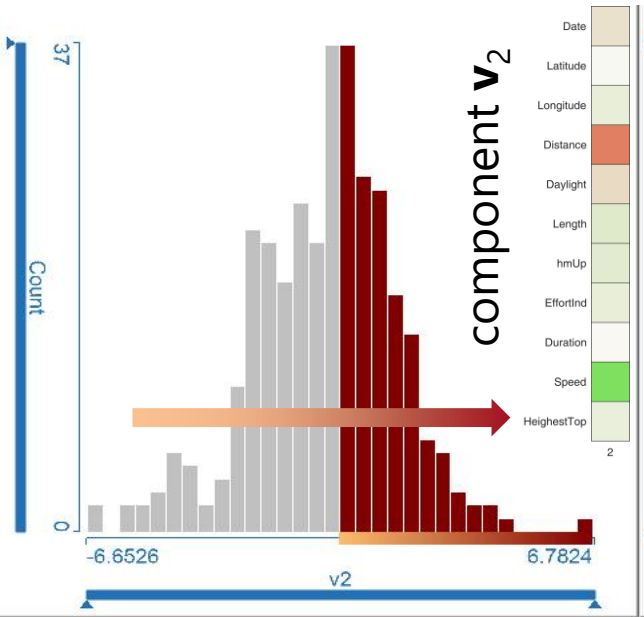


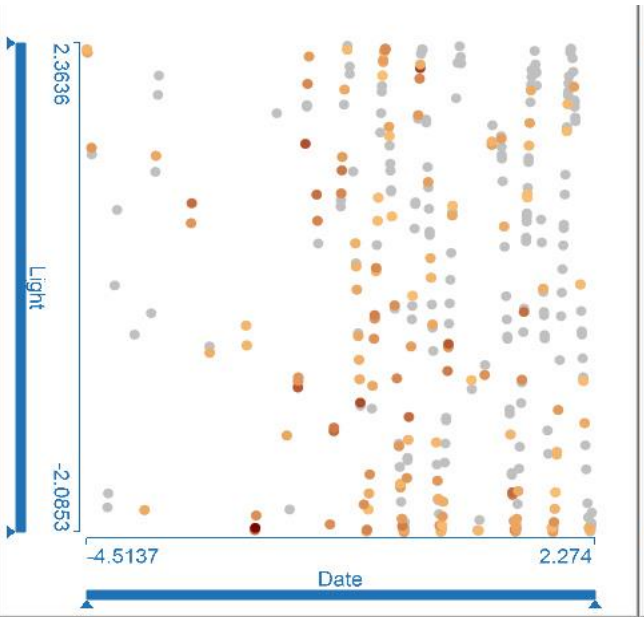


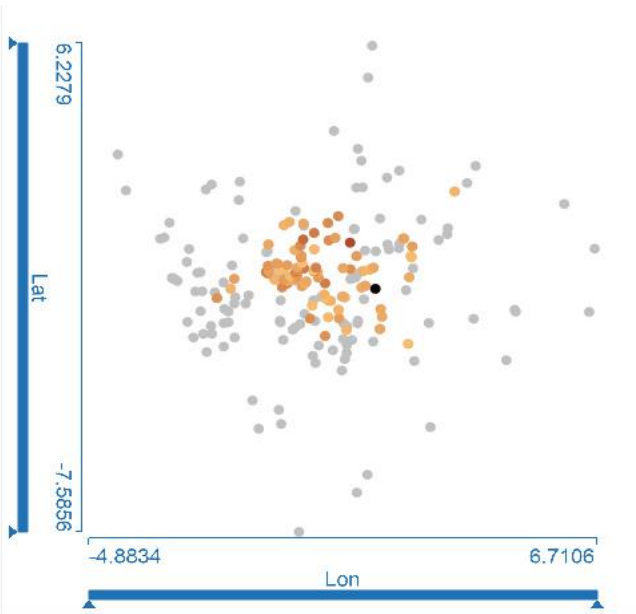


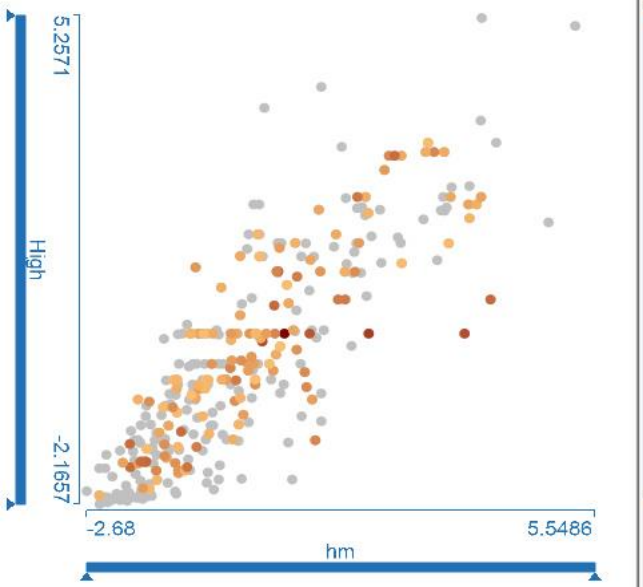


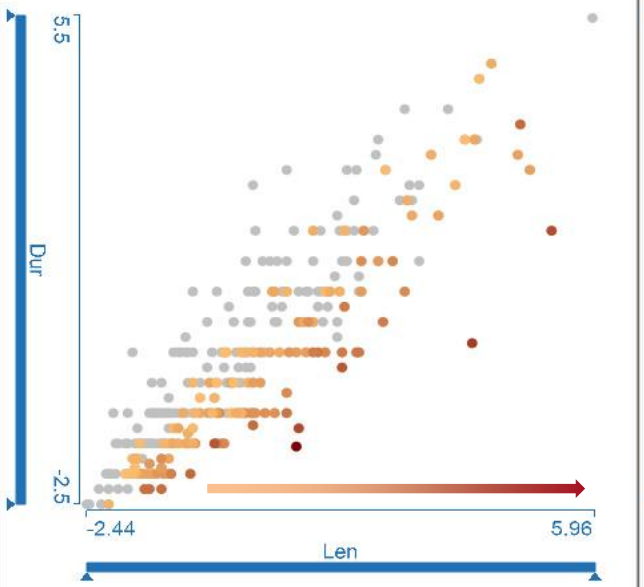
Example: Hiking around Bergen etc.

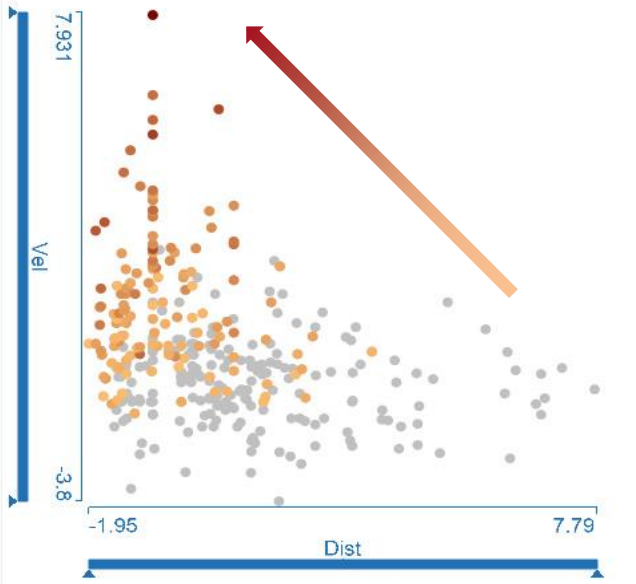




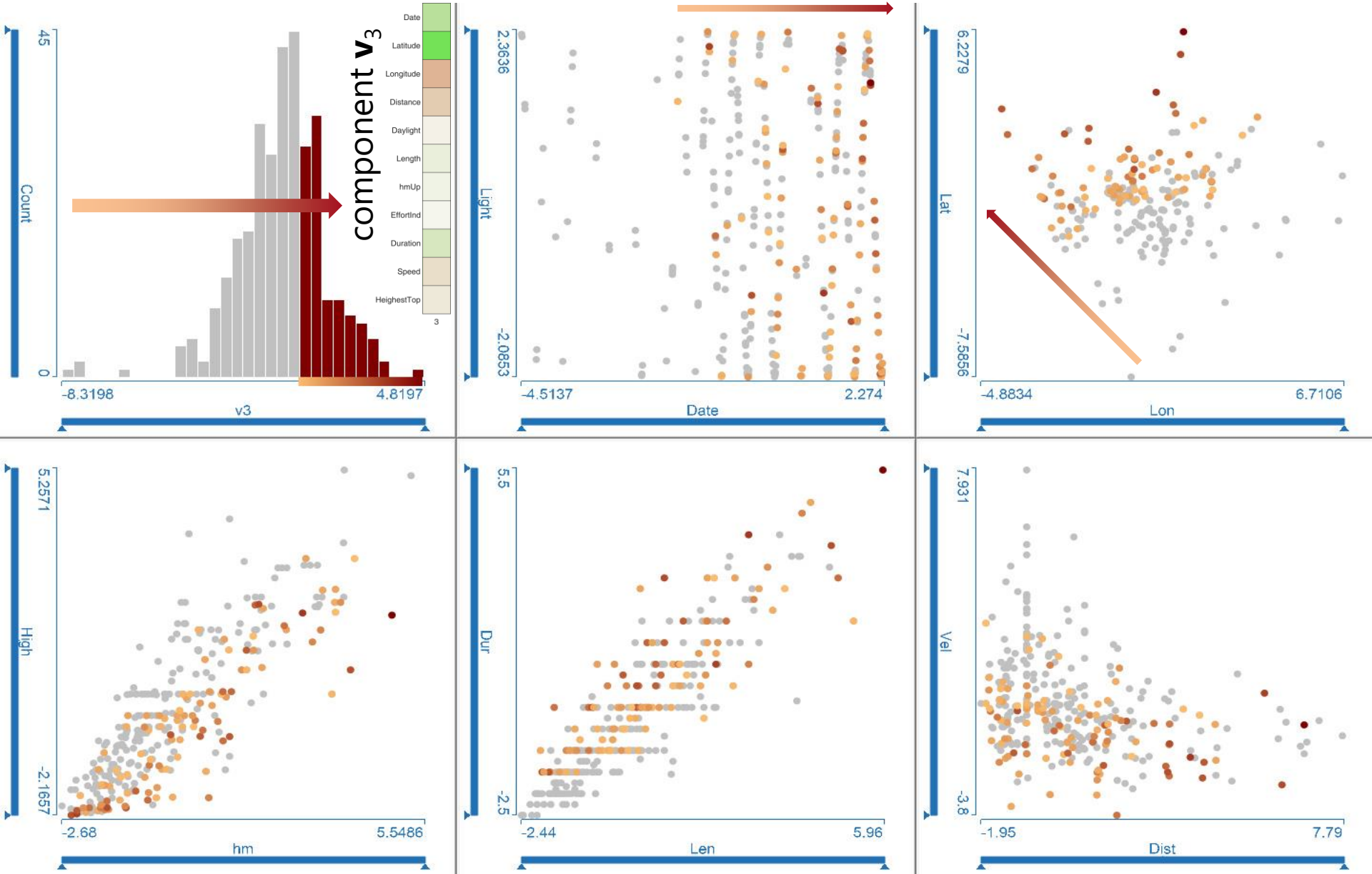




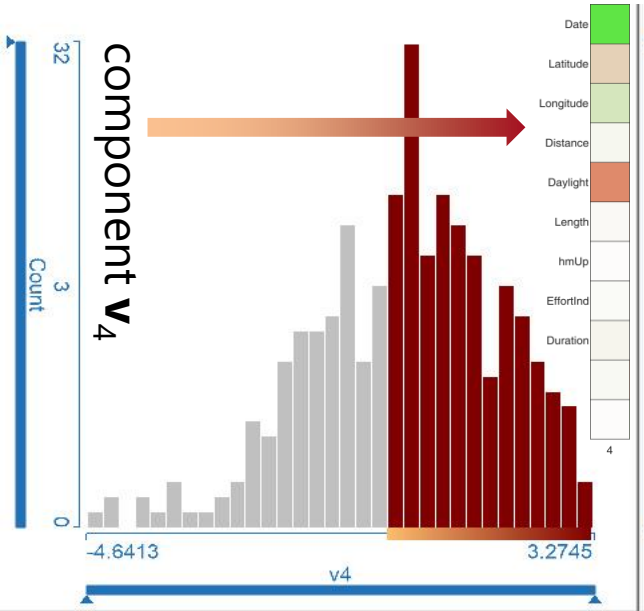


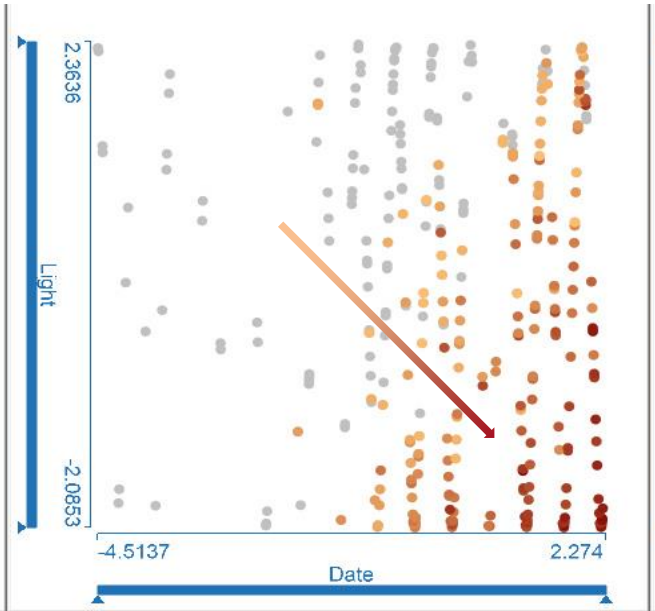


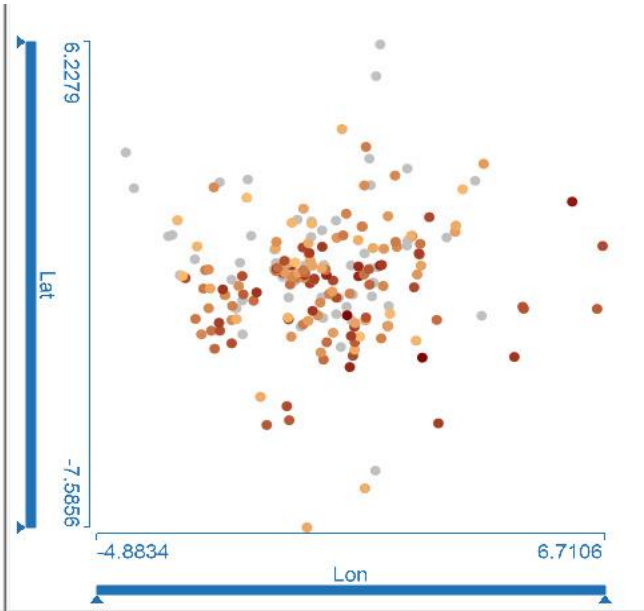
Example: Hiking around Bergen etc.

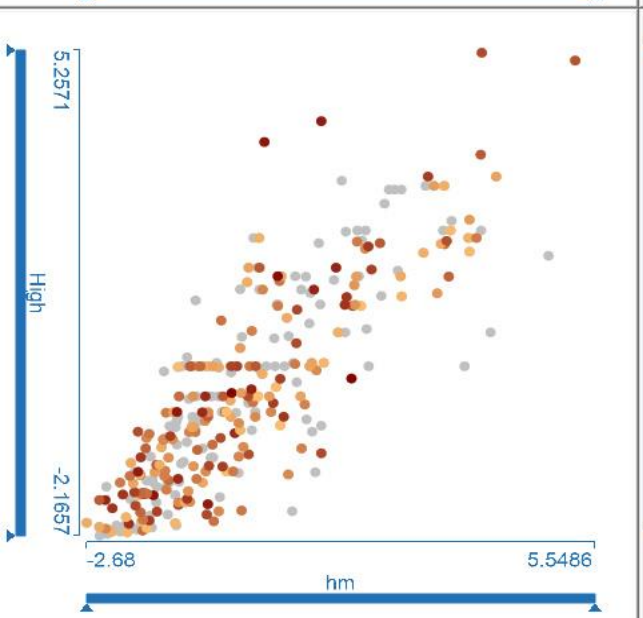


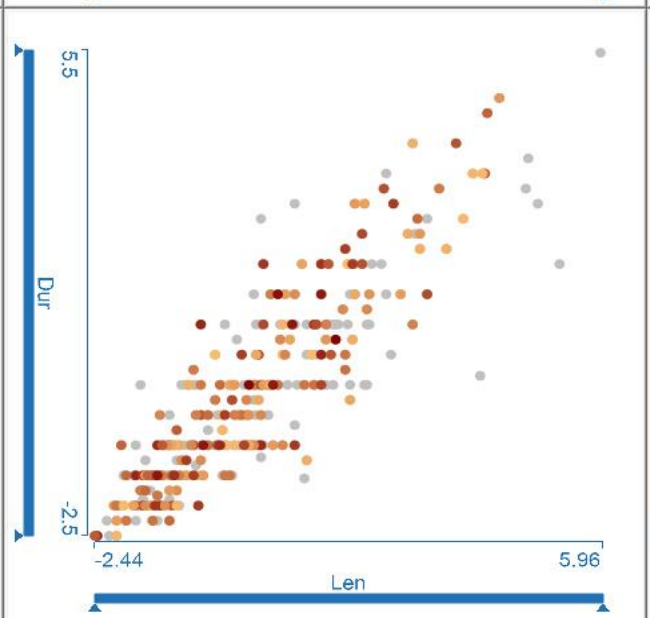
Example: Hiking around Bergen etc.

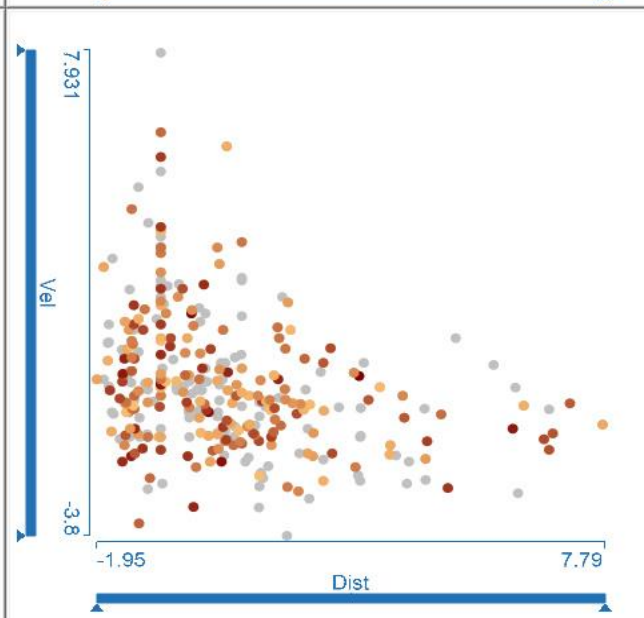




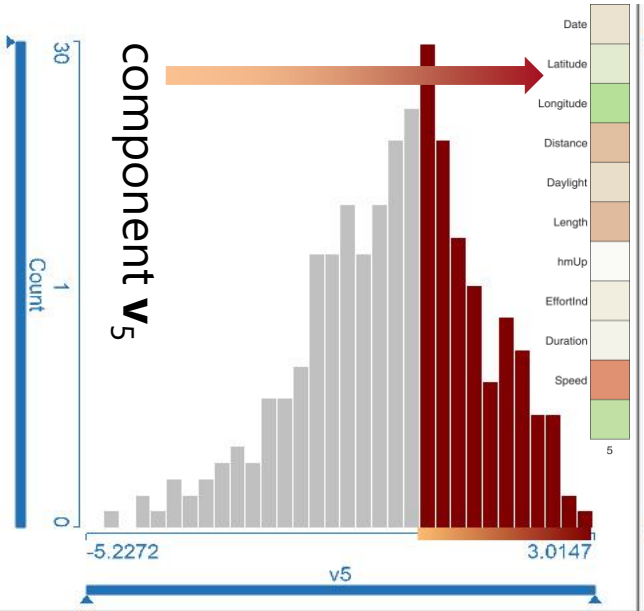


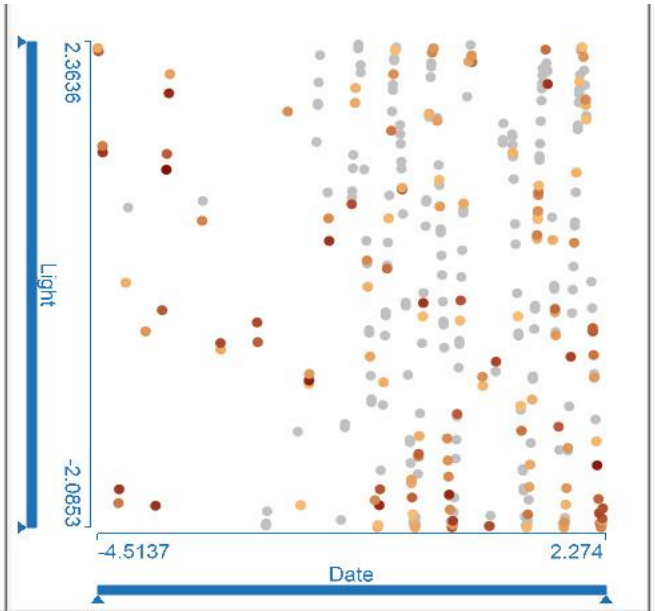


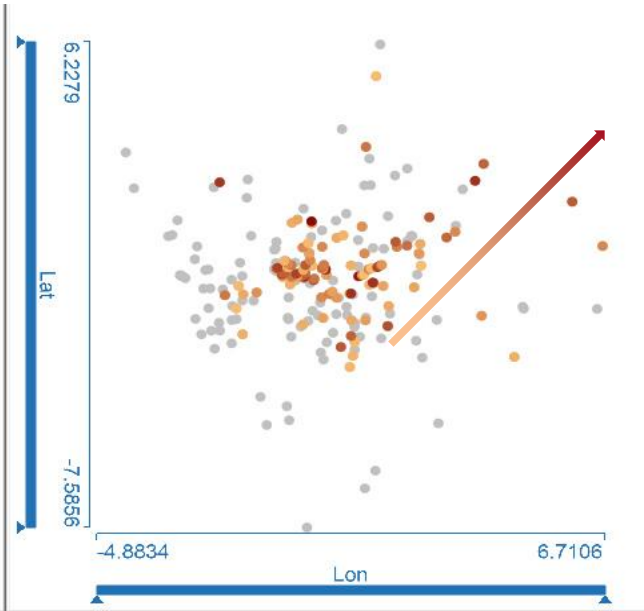


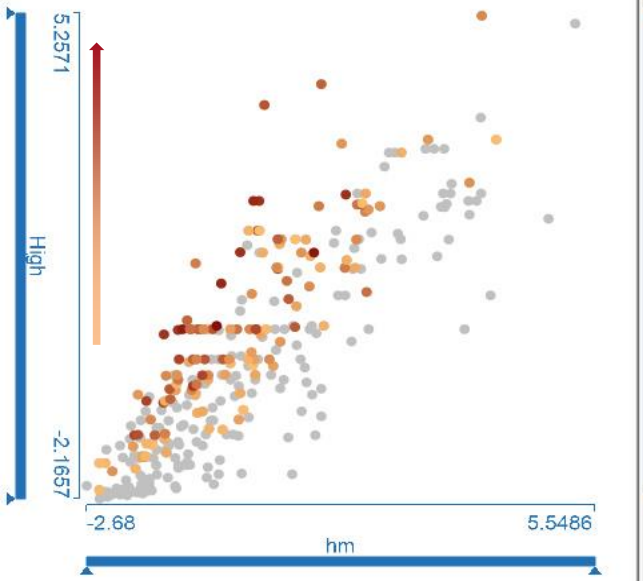


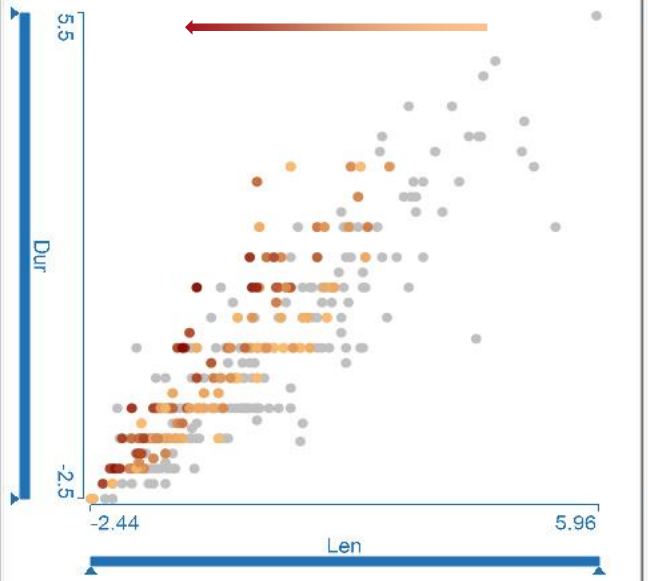
Example: Hiking around Bergen etc.

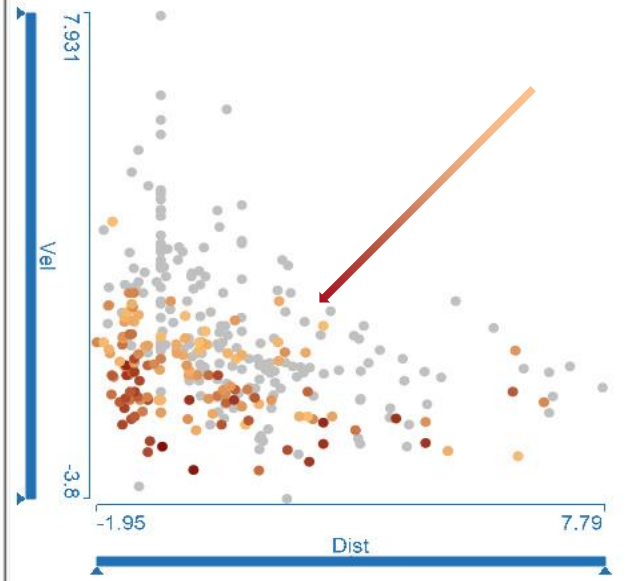




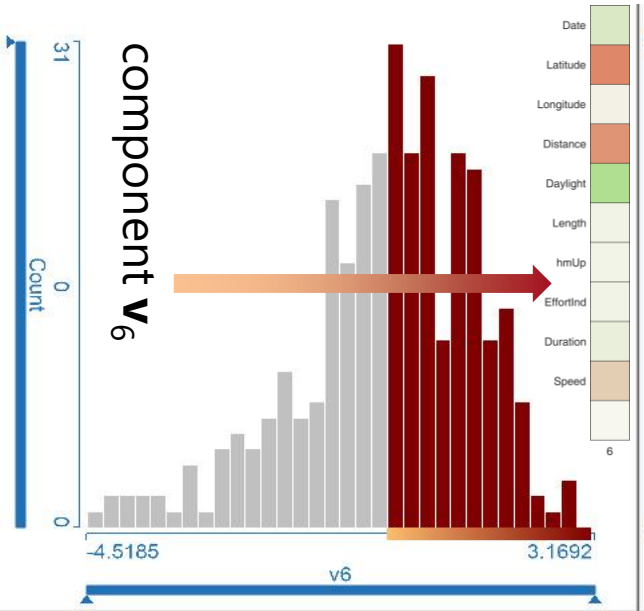


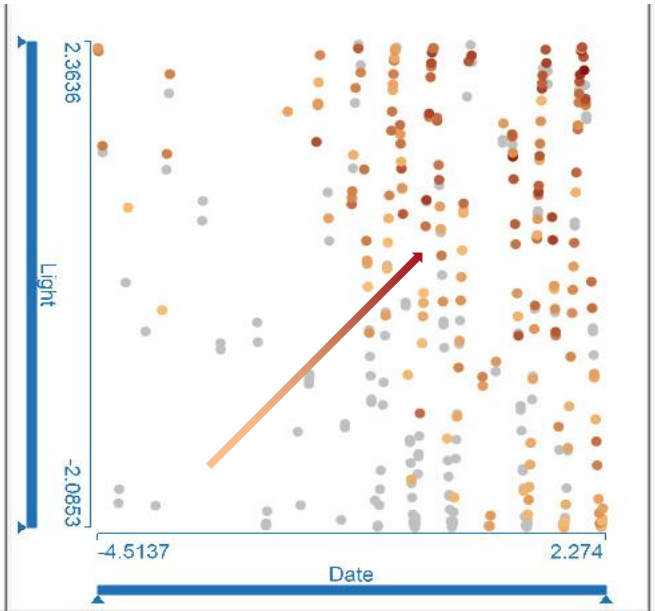


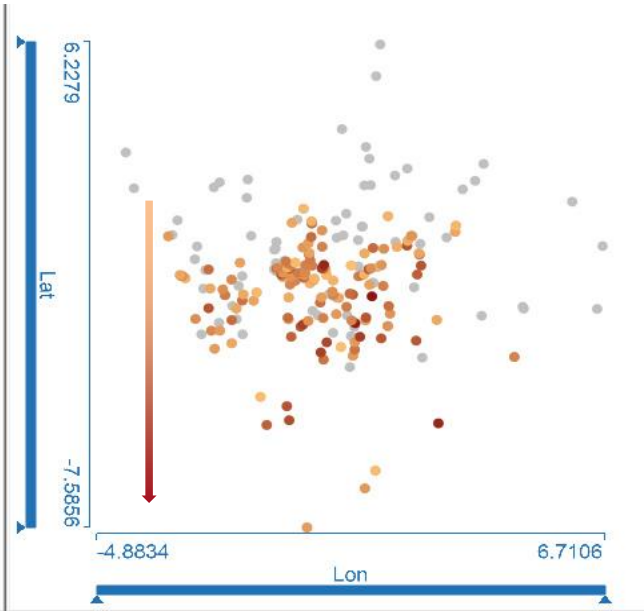


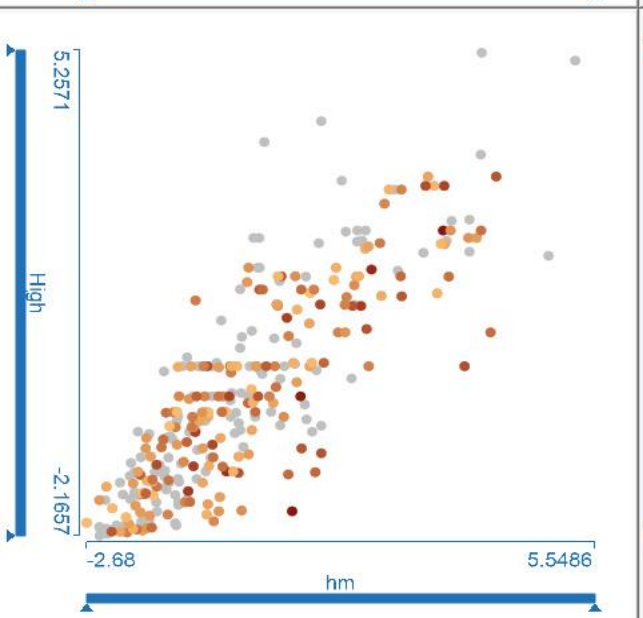


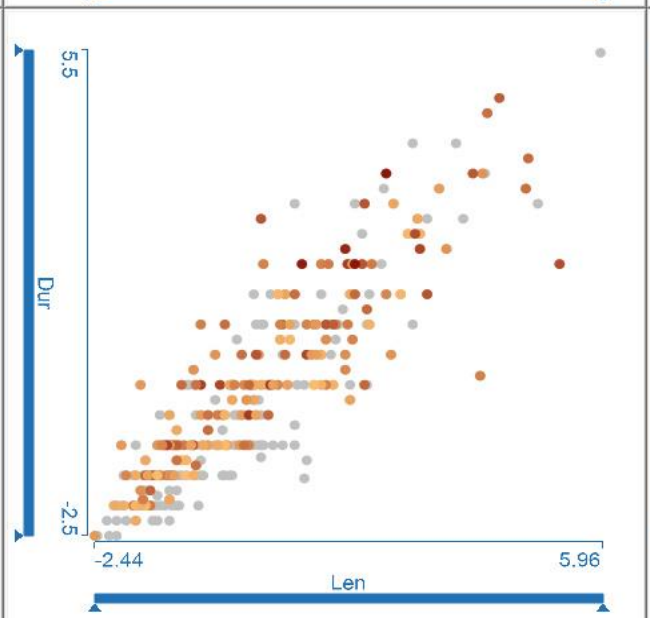
Example: Hiking around Bergen etc.

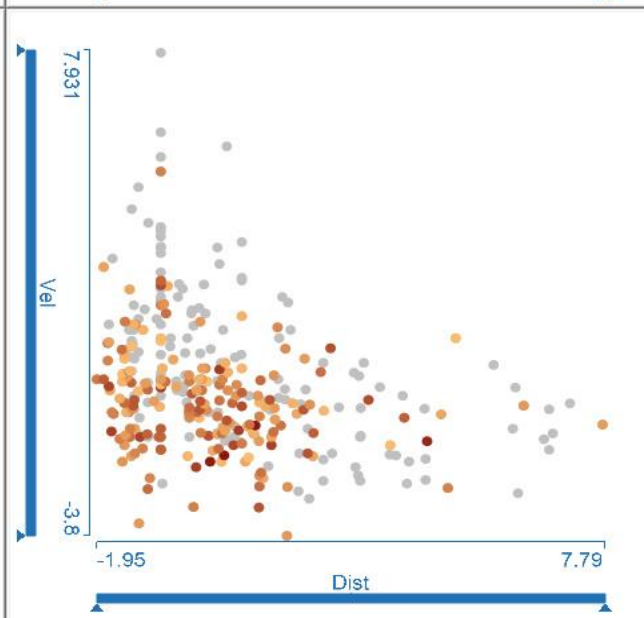












Example: Hiking around Bergen etc.

[see also the MATLAB example 2c_A]

What have we learned?

- the strongest information in these 331 hiking-data is a (combined) measure for the *extent of the activity* (considering length, hm, duration, etc.)
- secondly, the variation between "distant and slow" and "nearby and fast" seems to be informative
- a 1D, 4(±1)D, or 7D approximation seems to be appropriate – dependent on how much information should be recovered

Rewriting the SVD

Separating the σ in Σ
(into a sum of Σ_i with one σ per matrix),
we get:

– $\mathbf{A} = \mathbf{U}\,\mathbf{\Sigma}\,\mathbf{V}^*$

$$= \begin{pmatrix} \mathbf{u}_1 & \underbrace{\mathbf{u}_2 \ \dots}_{\text{zeroed!}} \end{pmatrix} \begin{pmatrix} \sigma_1 & 0 & \cdots \\ 0 & 0 & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} \begin{pmatrix} \mathbf{v}_1^\top \\ \mathbf{v}_2^\top \\ \vdots \end{pmatrix} \Big\} \text{zeroed!}$$

$$+ \begin{pmatrix} \underbrace{\mathbf{u}_1}_{\text{zeroed!}} & \mathbf{u}_2 & \underbrace{\mathbf{u}_3 \ \dots}_{\text{zeroed!}} \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 & \cdots \\ 0 & \sigma_2 & 0 & \cdots \\ 0 & 0 & 0 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \begin{pmatrix} \mathbf{v}_1^\top \\ \mathbf{v}_2^\top \\ \mathbf{v}_3^\top \\ \vdots \end{pmatrix} \begin{matrix} \Big\} \text{zeroed!} \\ \\ \Big\} \text{zeroed!} \end{matrix} + \dots$$

$$= \sigma_1 \mathbf{u}_1 \mathbf{v}_1^\top + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^\top + \dots$$

outer product

$$\mathbf{A} = \sum_{j=1}^{r} \sigma_j \mathbf{u}_j \mathbf{v}_j^*$$

SVD – Facts (1b)

If matrix A is of rank *r*, then *r* σ_j are non-zero.

Given rank(**A**)=*r*,

- range(**A**) = ($\mathbf{u}_1$, $\mathbf{u}_2$, …, $\mathbf{u}_r$)
- null(**A**) = ($\mathbf{v}_{r+1}$, $\mathbf{v}_{r+2}$, …, $\mathbf{v}_m$)

in MATLAB: „`range(A)`“ and „`null(A)`“

A's norm is given by the SVD:

$$||\mathbf{A}||_2 = \sigma_1 \quad \text{and} \quad ||\mathbf{A}||_F = \sqrt{\sum \sigma_k^2}$$

If A is square, then

$$|\det(\mathbf{A})| = \prod \sigma_k$$

Based on $\mathbf{A} = \sum \sigma_j \mathbf{u}_j \mathbf{v}_j^*$ **, A's best rank-*k* approximation is**

- with $||\mathbf{A} - \mathbf{A}_k||_2 = \sigma_{k+1}$

  and $||\mathbf{A} - \mathbf{A}_k||_F = \sqrt{\sum_{j=k+1}^{r} \sigma_j^2}$

$$\mathbf{A}_k = \sum_{j=1}^{k} \sigma_j \mathbf{u}_j \mathbf{v}_j^*$$

Rank-*k* Approximation

[MATLAB example 2c_B]

Given some (*n* x *d*)**–dimensional data D**
(*n* data items, *d* dimensions per item, rank $r \approx \min(n, d)$),

– we can approximate **D** with a lower-rank matrix $\mathbf{D}_k$ by only using the *k* first (largest) singular values

$$(\mathbf{u}_1\ \mathbf{u}_2\ \ldots\ \mathbf{u}_n) \begin{pmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_m \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix} \begin{pmatrix} \mathbf{v}_1^\top \\ \mathbf{v}_2^\top \\ \vdots \\ \mathbf{v}_m^\top \end{pmatrix}$$

Leaving $\mathbf{D}_k = \sum_{j=1}^{k} \sigma_j \mathbf{u}_j \mathbf{v}_j^*$

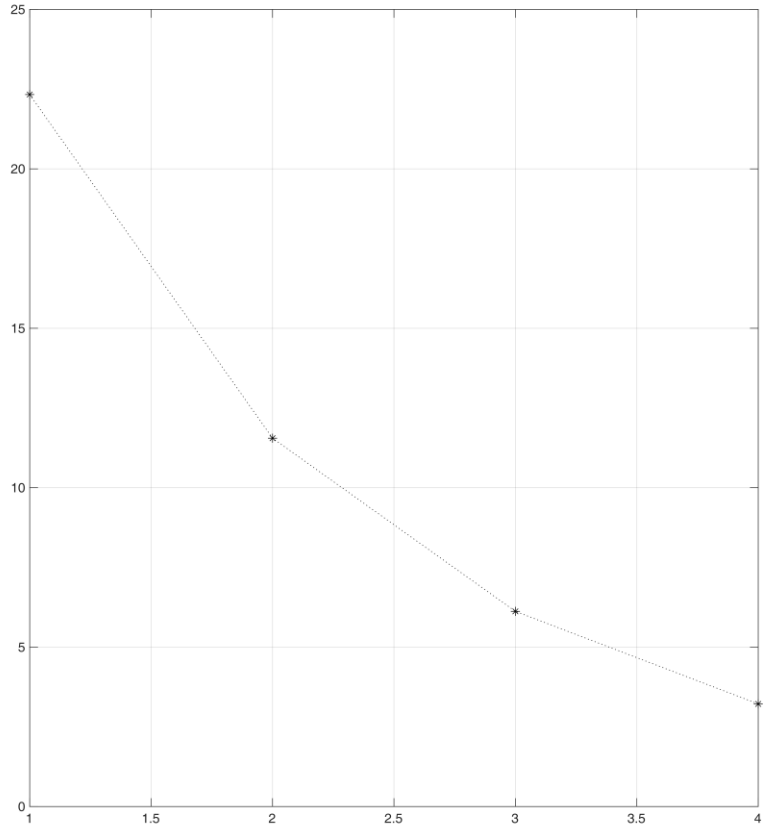
Low-rank Approximation in MATLAB

[MATLAB example 2c_B]

Given some (n x d)**–dimensional data D**

(n data items, d dimensions per item, rank $r \approx \min(n, d)$),

- we assume that we have set up the data matrix in **D**
- usually, we normalize (so that all dimensions relate to each other)
- we then compute the SVD and plot the singular values, for ex.:



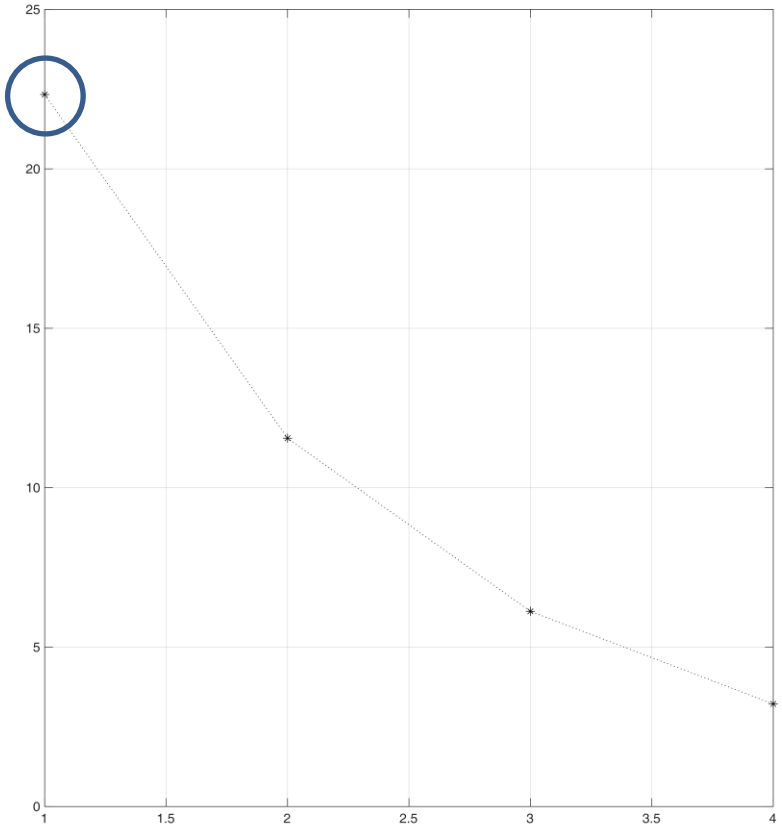
Low-rank Approximation in MATLAB

[MATLAB example 2c_B]

Given some (n x d)**–dimensional data D**

(n data items, d dimensions per item, rank $r \approx \min(n, d)$),

- we assume that we have set up the data matrix in **D**
- usually, we normalize (so that all dimensions relate to each other)
- we then compute the SVD and plot the singular values
- as σ_1 clearly dominates all other singular values, we start with a rank-1 approximation, for ex., by `D1 = U(:,1) * S(1,1) * V(:,1)';` in MatLab

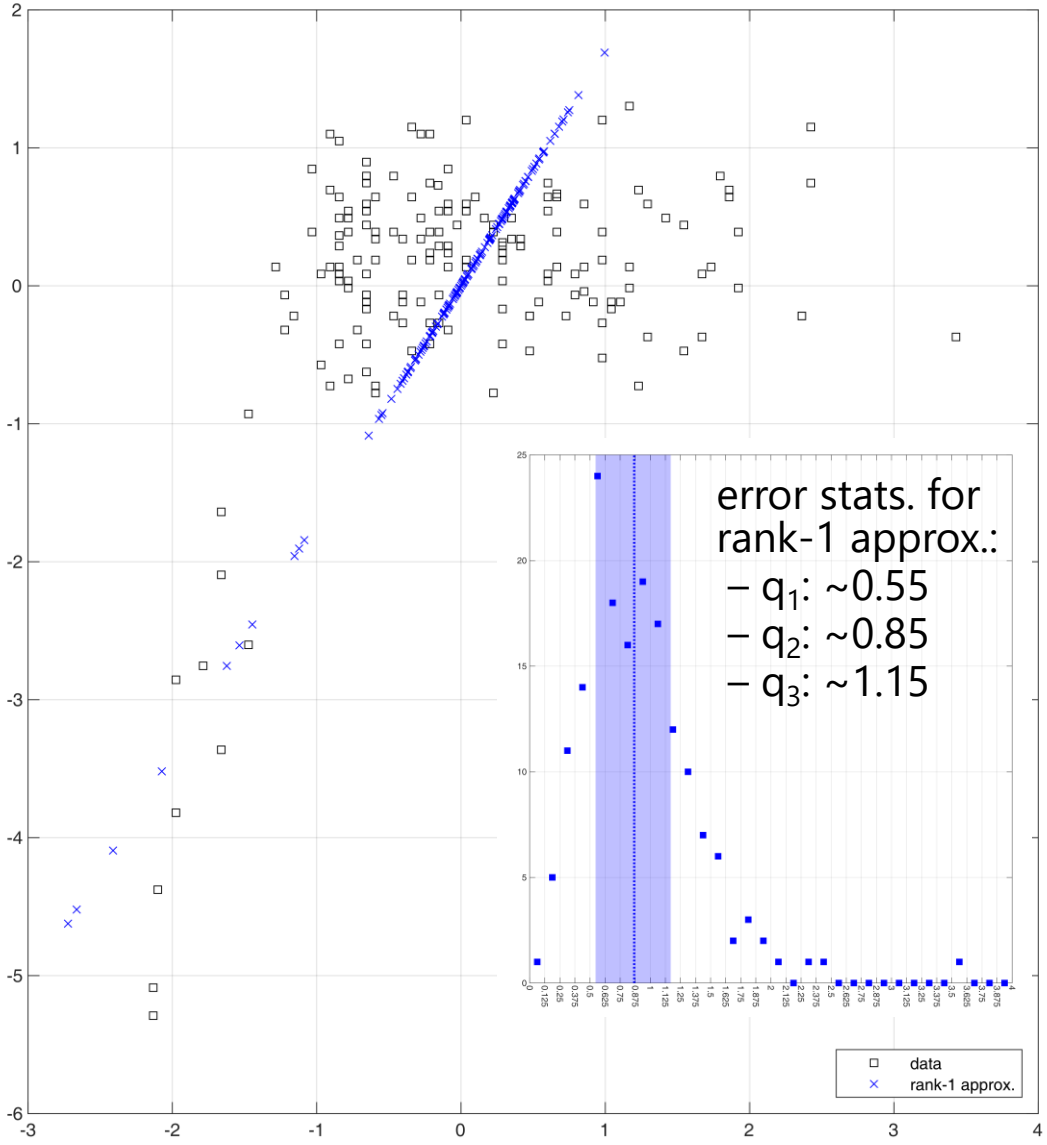


Low-rank Approximation in MATLAB

[MATLAB example 2c_B]

Given some (n x d)**–dimen**
(n data items, d dimension

- we assume that we have
- usually, we normalize (so
- we then compute the SV
  and plot the singular valu
- as σ_1 clearly dominates a
  singular values, we start
  rank-1 approximation, fo
  `D1 = U(:,1) * S(1,1) *`
  in MatLab
- of course, we wish to see
  how well $\mathbf{D}_1$
  approximates $\mathbf{D}$

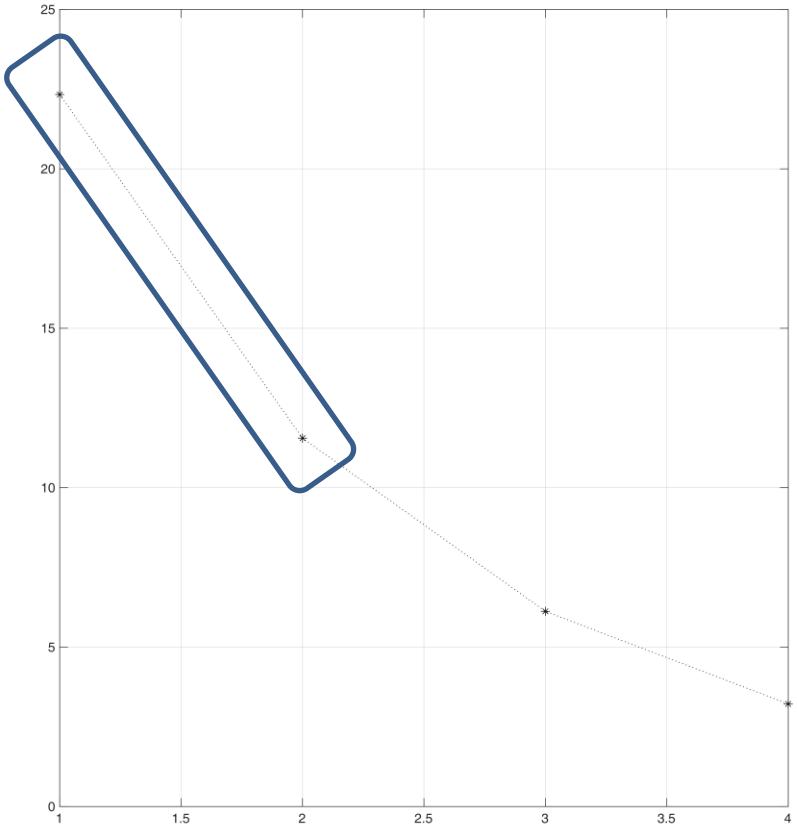


Low-rank Approximation in MATLAB

[MATLAB example 2c_B]

Given some (n x d)**–dimensional data D**
(n data items, d dimensions per item, rank $r \approx \min(n, d)$),

- we assume that we have set up the data matrix in **D**
- usually, we normalize (so that all dimensions relate to each other)
- we then compute the SVD and plot the singular values
- next, we follow up with a rank-2 approximation (with `D2 = D1 + U(:,2)*S(2,2)*V(:,2)';` in MatLab)

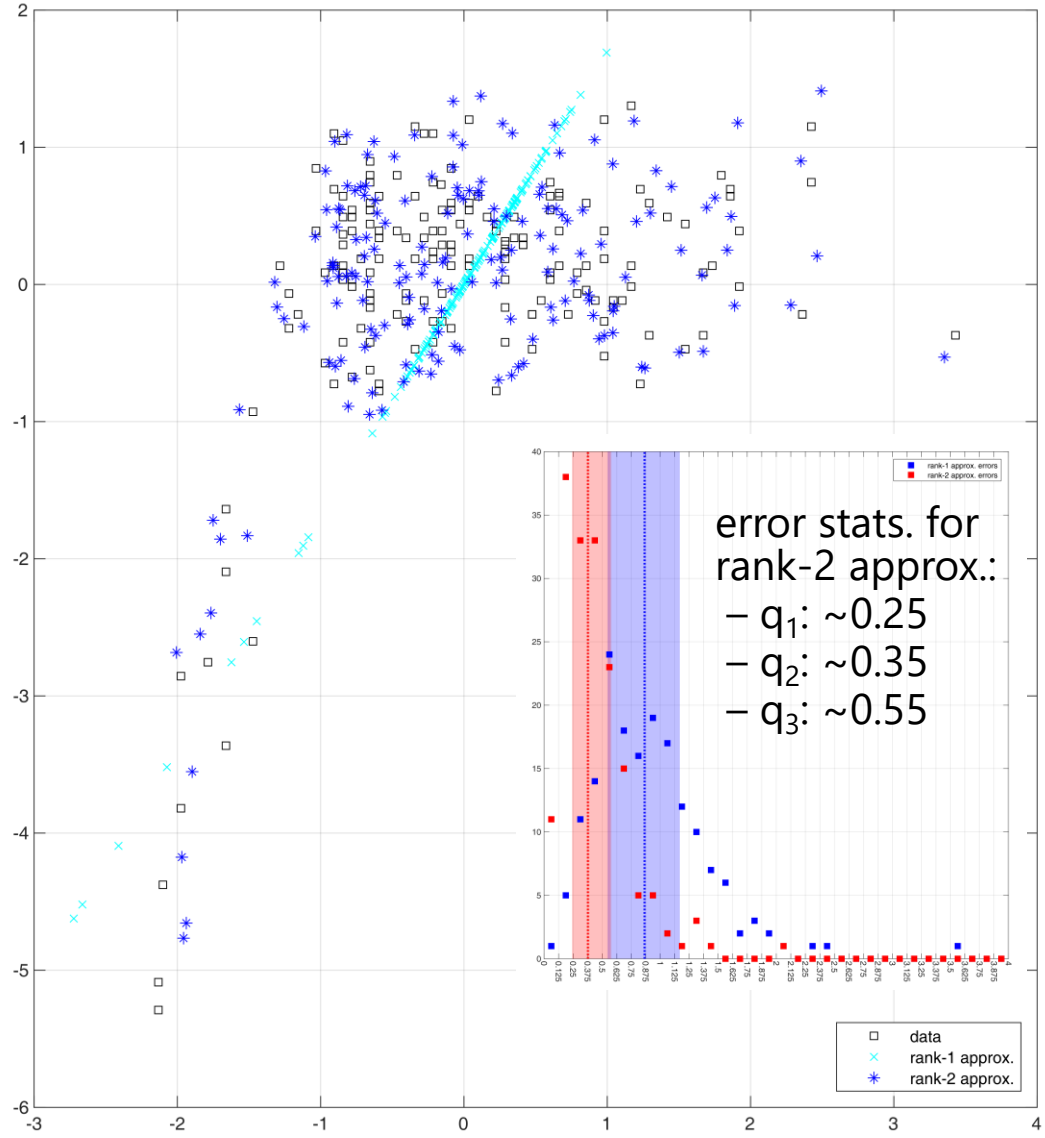


Low-rank Approximation in MATLAB

[MATLAB example 2c_B]

Given some (*n* x *d*)**–dimer**
(*n* data items, *d* dimensions

- we assume that we have s
- usually, we normalize (so
- we then compute the SVI and plot the singular valu
- next, we follow up with a rank-2 approximation (wi `D2 = D1 + U(:,2)*S(2,2)` in MatLab)
- again, we wish to see how well $\mathbf{D}_2$ approximates **D**
- much better!

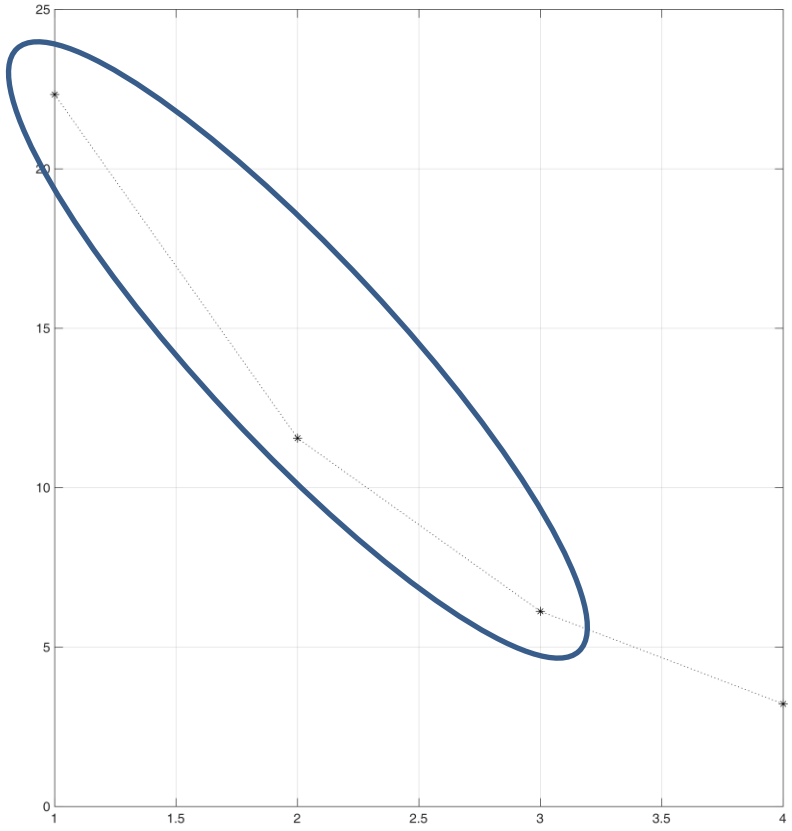


Low-rank Approximation in MATLAB

[MATLAB example 2c_B]

Given some (n x d)**–dimensional data D**
(n data items, d dimensions per item, rank $r \approx \min(n, d)$),

- we assume that we have set up the data matrix in **D**
- usually, we normalize (so that all dimensions relate to each other)
- we then compute the SVD and plot the singular values
- clearly, we also check a rank-3 approximation (with `D3=U(:,1:3)*S(1:3,1:3)*V(:,1:3)';` in MatLab)

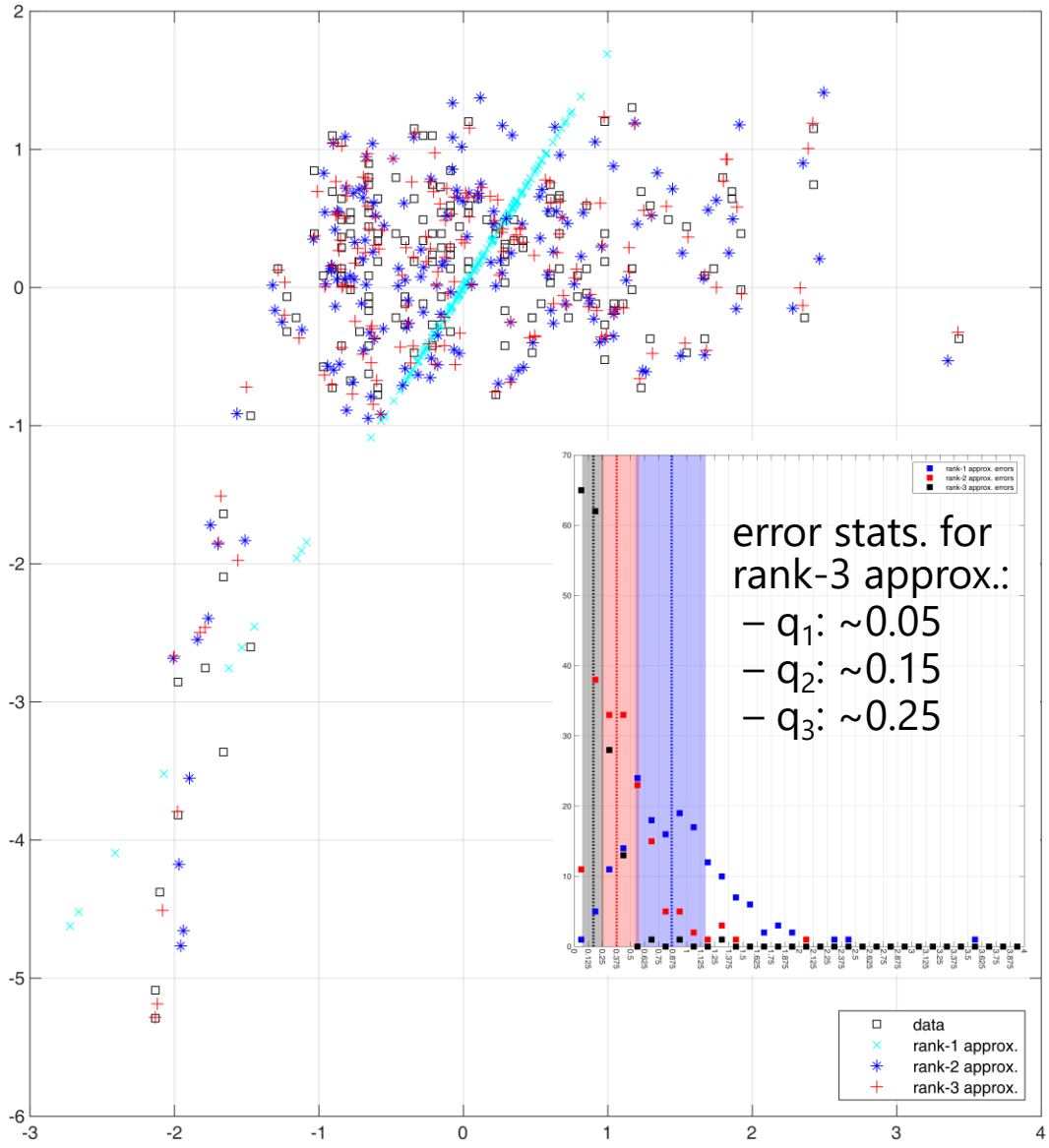


Low-rank Approximation in MATLAB

[MATLAB example 2c_B]

Given some (*n* x *d*)**–dimer**
(*n* data items, *d* dimension

- we assume that we have
- usually, we normalize (so
- we then compute the SV and plot the singular valu
- clearly, we also check a rank-3 approximation (w `D3=U(:,1:3)*S(1:3,1:3)` in MatLab)
- visualization!
- almost perfect!



Outlook

Next:

- eigenanalysis
- PCA

Reading and Related Material

In the book:

- chapter 2 (on linear systems), mostly section 2.1, and related parts
- chapter 15 (on SVD, etc.)

Course notes:

- section 4

An interactive online-book:

- http://ImmersiveMath.com/,
  chapters (1–4 &) 5 (on Gaussian elimination)
  and 6 (on The Matrix)

On Wikipedia:

- many good pages