# The PCA

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

# Looking Back & Forth

**Last time:**

- from changing bases to the SVD
- SVD facts and interpreting the SVD
- data matrices and the SVD
- rank-*k* approximation with the SVD

**Today:**

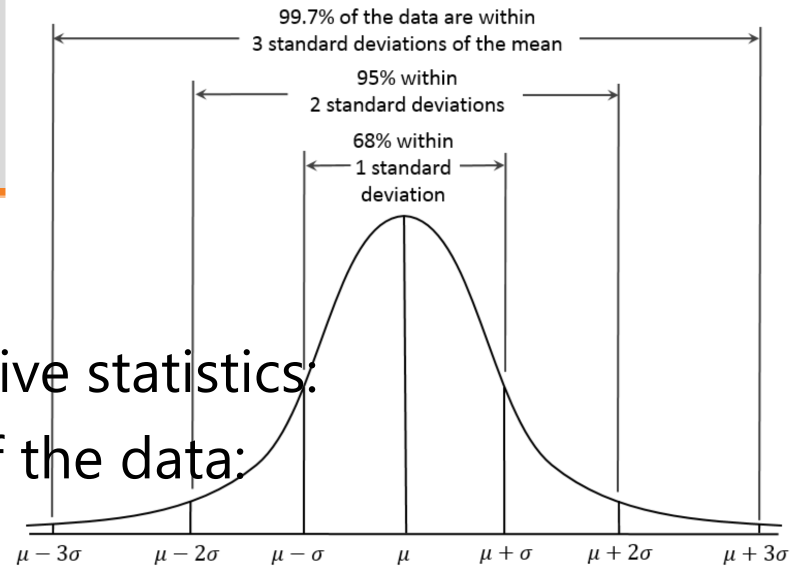
- variance, covariance, and correlation
- eigenanalysis
- PCA (principal component analysis)

# Introduction

**Principal Component Analysis (PCA)**:

- represents the ***data*** (also) ***in an alternative frame*** such that *most data variance is aligned with the new "x-axis"*
- PCA is ***based on an eigenanalysis of the covariance matrix*** and the corresponding *eigenvalues indicate, how much variance is explained by each axis*
- PCA can be used as a *basis for* ***dimension reduction***: if only those *k* axes are used that correspond to the largest variance in the data, this leads to a *k*-dimensional approximation
- PCA can be useful, *when dealing with noisy data*: much of the signal can be concentrated into the first components, possibly raising the SNR (after dimension reduction)

# Variance




## In 1D:

- 1st and 2nd order *moments* in descriptive statistics:
  - the *mean* $\mu$ estimates the *center* of the data:
  - the *standard deviation* $\sigma$ estimates the *variation / dispersion* of the data (68–95–99.7–rule):
  - the *variance* $\sigma^2$ is the *square of the standard deviation*:

$$\sigma = \sqrt{\frac{1}{n}\sum_i (x_i - \mu)^2}$$

$$\sigma^2 = \frac{1}{n}\sum_i (x_i - \mu)^2$$

## Rewriting variance:

- shift the data by $-\mu$ to center it
- computing the variance as $\mathbf{x}^\mathsf{T}\mathbf{x}/(n-1)$ (col. $\mathbf{x}$)
- `x = [65; 58; 74; 54; 72; 62; 72]`

!

```
mu = mean(x)
x_c = x-mu
```

```
x_size = size(x); n = x_size(1)
sigma_2 = (x_c.' * x_c) / (n-1)
```

note:
for an unbiased estimator,
divide by $n$-1

[MATLAB ex. 2d_A]

# Variance and Covariance

## In $n$D:

- the center is estimated per dimension (one may also rescale!)
- the variation is estimated (after centering):
  - within each dimension $\mathbf{x}$: $\sigma^2{}_{\mathbf{x}} = \mathbf{x}^{\mathsf{T}}\mathbf{x} / (n-1)$
  - between dims. $\mathbf{x}$ & $\mathbf{y}$ (covariance): $\sigma^2{}_{\mathbf{xy}} = \mathbf{x}^{\mathsf{T}}\mathbf{y} / (n-1)$

## All variances / covariances in $n$D:

- variances and covariances show up in the covariance matrix:
  - given data matrix $\mathbf{D}$ (items in cols.): $\mathbf{\Sigma}^2{}_{\mathbf{D}} = \mathbf{D}\,\mathbf{D}^{\mathsf{T}} / (n-1)$
  - transposed matrix $\mathbf{D}$ (items in rows): $\mathbf{\Sigma}^2{}_{\mathbf{D}} = \mathbf{D}^{\mathsf{T}}\mathbf{D} / (n-1)$

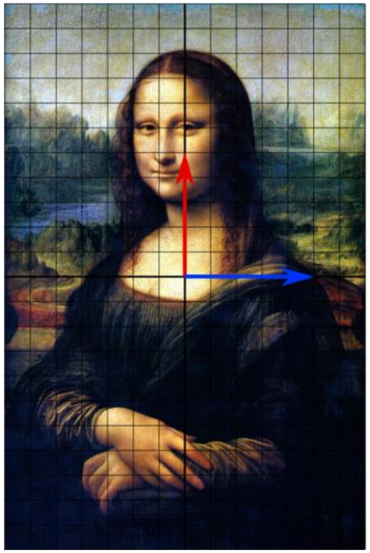
[see also MATLAB example 2d_B]

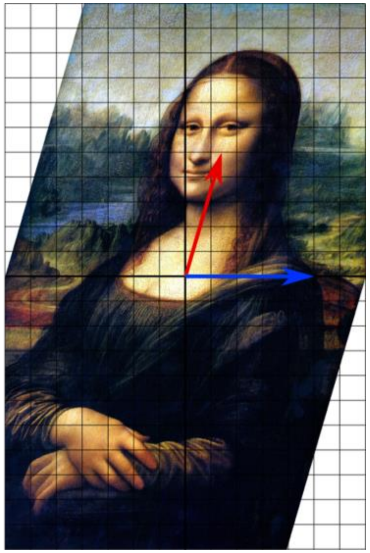
# Eigenanalysis

**Assume a square matrix A in $\mathbb{R}^{n \times n}$:**

- then an *eigenvalue problem* is written as $\mathbf{A}\,\mathbf{x} = \lambda\,\mathbf{x}$ with $\lambda$ being a scalar value, i.e., an *eigenvalue* of **A**
- to solve an eigenvalue problem for both $\mathbf{x}$ and $\lambda$ means to find—in addition to all ***eigenvalues*** $\lambda_i$—all corresponding vectors $\mathbf{x}_i$, i.e., the ***eigenvectors*** $\mathbf{x}_i$, for which the left-multiplication with **A** is the same as a scalar stretching/shrinking of **x** (by $\lambda$)
  (eigenvectors **x** do not change direction under **A**, only length, if at all)

[ex. from Wikipedia]





the blue vector is an eigenvector
of the depicted shear operation
(but not so the red vector)

# Eigenvalues

**Given a square, real-valued matrix A, then:**

- the eigenvalues can be positive, zero, or negative
- the eigenvalues can be complex (they appear in pairs for $\mathbf{A} \in \mathbb{R}^{nxn}$)
- the eigenvalues can appear multiple times (algebraic multiplicity $\mu$)
- all eigenvalues add up to the trace of **A**
- all eigenvalues multiply to the determinant of **A**
- **A** is invertible, *if and only if* all eigenvalues of **A** are non-zero
- **A** is invertible $\Rightarrow$ the eigenvalues of $\mathbf{A}^{-1}$ are $1/\lambda_i$
- if **A** is unitary, then the norm of all eigenvalues of **A** is 1
- if $\mathbf{A} = \mathbf{A}^*$ (**A** is self-adjoint), then all eigenvalues are real
  (in particular true for symmetric, real-valued matrices!)
- the eigenvalues of $\mathbf{A}^k$ are $\lambda_i{}^k$, if $k$ a positive integer
- if **A** is positive-/negative (semi-)definite,
  then all eigenvalues of **A** are positive/negative (incl. 0)

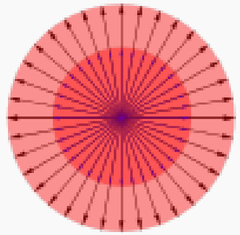
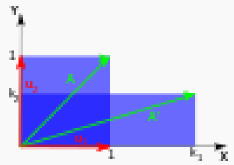
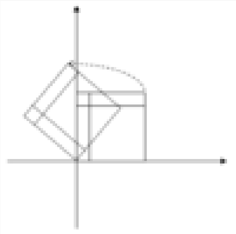
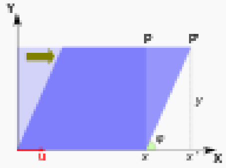
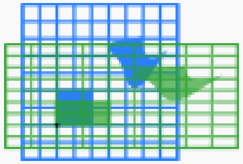
# Eigenvectors

**Each eigenvalue** $\lambda_i$ (algebraic multiplicity $\mu_i \geq 1$)

- corresponds to a subspace, i.e., the *eigenspace spanned by the corresponding eigenvector(s)*
- if **A** is symmetric,
  then the eigenvectors are orthogonal to each other
- any multiple of an eigenvector is also an eigenvector
- the eigenvectors can be complex (pairs of them)
- eigenspaces are $\gamma_i$-dimensional ($\gamma_i \geq 1$) – geometric multiplicity (the usual case is 1D)

in MATLAB: „`[V, L]=eig(A)`“ gives
the (normalized) eigenvectors (as columns of $\mathbf{V}$) and
the eigenvalues (as major diagonal of $\mathbf{L}$)

# Eigenvalues and Eigenvectors in 2D

| [from Wikipedia] | scaling | unequal scaling | rotation | horizontal shear | hyperbolic rotation |
|---|---|---|---|---|---|
| illustration |  |  |  |  |  |
| matrix | $\begin{bmatrix} k & 0 \\ 0 & k \end{bmatrix}$ | $\begin{bmatrix} k_1 & 0 \\ 0 & k_2 \end{bmatrix}$ | $\begin{bmatrix} c & -s \\ s & c \end{bmatrix}$ $c = \cos\theta$ $s = \sin\theta$ | $\begin{bmatrix} 1 & k \\ 0 & 1 \end{bmatrix}$ | $\begin{bmatrix} c & s \\ s & c \end{bmatrix}$ $c = \cosh\varphi$ $s = \sinh\varphi$ |
| characteristic polynomial | $(\lambda - k)^2$ | $(\lambda - k_1)(\lambda - k_2)$ | $\lambda^2 - 2c\lambda + 1$ | $(\lambda - 1)^2$ | $\lambda^2 - 2c\lambda + 1$ |
| eigenvalues $\lambda_i$ | $\lambda_1 = \lambda_2 = k$ | $\lambda_1 = k_1$ $\lambda_2 = k_2$ | $\lambda_1 = e^{\mathbf{i}\theta} = c + s\mathbf{i}$ $\lambda_2 = e^{-\mathbf{i}\theta} = c - s\mathbf{i}$ | $\lambda_1 = \lambda_2 = 1$ | $\lambda_1 = e^{\varphi}$ $\lambda_2 = e^{-\varphi},$ |
| algebraic multipl. $\mu_i = \mu(\lambda_i)$ | $\mu_1 = 2$ | $\mu_1 = 1$ $\mu_2 = 1$ | $\mu_1 = 1$ $\mu_2 = 1$ | $\mu_1 = 2$ | $\mu_1 = 1$ $\mu_2 = 1$ |
| geometric multipl. $\gamma_i = \gamma(\lambda_i)$ | $\gamma_1 = 2$ | $\gamma_1 = 1$ $\gamma_2 = 1$ | $\gamma_1 = 1$ $\gamma_2 = 1$ | $\gamma_1 = 1$ | $\gamma_1 = 1$ $\gamma_2 = 1$ |
| eigenvectors | All non-zero vectors | $u_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ $u_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ | $u_1 = \begin{bmatrix} 1 \\ -\mathbf{i} \end{bmatrix}$ $u_2 = \begin{bmatrix} 1 \\ +\mathbf{i} \end{bmatrix}$ | $u_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ | $u_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ $u_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$ |

# Writing it all-in-one: a Decomposition!

**Assuming** $n$ **solutions** $\mathbf{x}_i$ & $\lambda_i$ **to** $\mathbf{A}\,\mathbf{x} = \lambda\,\mathbf{x}$ (multiplicity OK)

– then

$$\left.\begin{aligned} \mathbf{A}\,\mathbf{x}_1 &= \lambda_1\,\mathbf{x}_1 \\ \mathbf{A}\,\mathbf{x}_2 &= \lambda_2\,\mathbf{x}_2 \\ &\vdots \\ \mathbf{A}\,\mathbf{x}_n &= \lambda_n\,\mathbf{x}_n \end{aligned}\right\} \Rightarrow \quad \mathbf{A}\,\mathbf{V} = \mathbf{V}\,\mathbf{\Lambda}$$

with $\mathbf{V} = (\mathbf{x}_1\ \mathbf{x}_2\ \ \ldots \mathbf{x}_n)$ and

$$\mathbf{\Lambda} = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{pmatrix}$$

$$\boxed{\Rightarrow \quad \mathbf{A} = \mathbf{V}\,\mathbf{\Lambda}\,\mathbf{V}^{-1}}$$

# A better Frame

## Eigenanalysis…

… leads to the eigenvalue/eigenvector decomposition $\mathbf{A} = \mathbf{V}\,\mathbf{\Lambda}\,\mathbf{V}^{-1}$
with $\mathbf{V}$ hosting all (normalized) eigenvectors
and $\mathbf{\Lambda}$ having all eigenvalues along the major diagonal

… if $\mathbf{A}$ is real-valued and symmetric, then $\mathbf{V}$ is unitary, i.e.,
an orthonormal basis of $\mathbb{R}^n$ (important for PCA)

## This suggests…

… that we can consider our $\mathbf{A}$ in terms of basis $\mathbf{V}$: $\mathbf{X} = \mathbf{V}^*\,\mathbf{A}$
($\mathbf{X}$ is then $\mathbf{A}$ in orthonormal $\mathbf{V}$-coordinates)

## This leads to:

[see the MATLAB example 2d_C]

– Given $\mathbf{A}$ in orthonormal $\mathbf{V}$-coords. (by $\mathbf{X} = \mathbf{V}^*\,\mathbf{A}$),
$\mathbf{A} = \mathbf{V}\,\mathbf{\Lambda}\,\mathbf{V}^*$ leads to $\underline{\mathbf{X} =}\ \mathbf{V}^*\,\mathbf{A} = \mathbf{V}^*\,\mathbf{V}\,\mathbf{\Lambda}\,\mathbf{V}^*\ \underline{= \mathbf{\Lambda}\,\mathbf{V}^*}$

– in particular $\mathbf{X}\,\mathbf{v} = \underline{[\mathbf{A}\,\mathbf{v}]_{\mathbf{V}} =}\ \mathbf{\Lambda}\,\mathbf{V}^*\,\mathbf{v}\ \underline{= \mathbf{\Lambda}\,[\mathbf{v}]_{\mathbf{V}}}$ (dim.-wise scaling!)

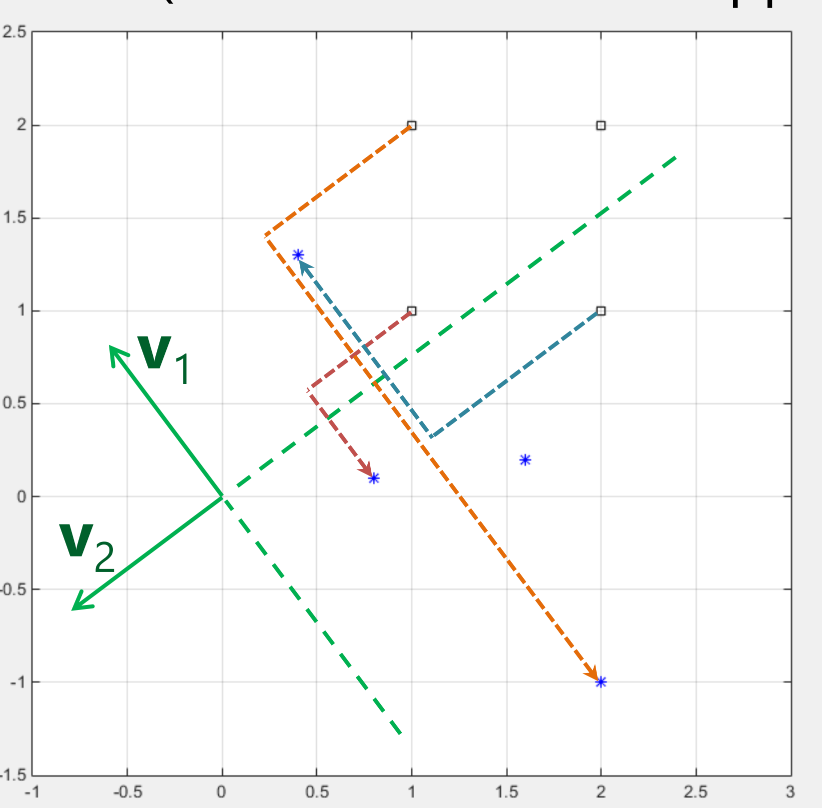
# Eigenanalysis – Interpretation

**Assuming a mapping A:** $\mathbb{R}^2 \to \mathbb{R}^2$ (again),

$$\mathbf{A} = \begin{pmatrix} -0.4 & 1.2 \\ 1.2 & -1.1 \end{pmatrix}$$

- and also considering four vectors $\begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 2 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \begin{pmatrix} 2 \\ 2 \end{pmatrix}$

  ('*' below after the mapping)




**Then the eigenanalysis of A gives**:
in MATLAB: „`[V, L]=eig(A)`"

- eigenvalues −2 and ½
- and eigenvectors $\overset{\mathbf{v}_1}{\begin{pmatrix} -0.6 \\ 0.8 \end{pmatrix}}$ & $\overset{\mathbf{v}_2}{\begin{pmatrix} -0.8 \\ -0.6 \end{pmatrix}}$

[see the MATLAB example 2d_D]

# SVD–Eigenanalysis Comparison

**Both are decomposition/diagonalization approaches,**

- i.e., both lead to a factorization of **A** into **P M Q** with **M** (singular values or eigenvalues) being diagonal and **P** & **Q** representing two change of basis steps

**But**

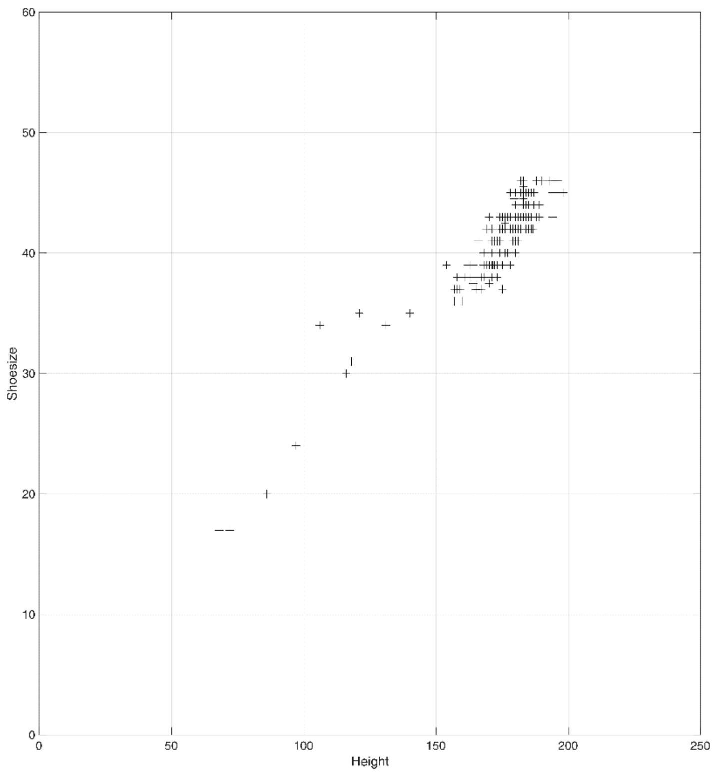
- the SVD exists always (+), even for rectangular matrices (+), while the eigendecomposition not necessarily exists (–)

  example: *nilpotent* matrix $\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$
- the SVD results in a real-valued solution (+), while the eigenvalues/eigenvectors may be complex-valued (–)
- the SVD uses two bases **U** & **V** (–), which always are unitary/orthonormal (+), the eigendecomp. uses one (+), **V**, that may not be orthogonal (–)

# PCA – Introduction

## Some data

- may be "intrinsically" $r$-dimensional,
  but represented $d$-dimensional with $d>r$
- example: if all data $\mathbf{d}_i \in \mathbb{R}^3$ are of form $\mathbf{d}_i = s_i\,\mathbf{v}$, $\mathbf{v} \in \mathbb{R}^3$, then
  their representation is 3D, while their intrinsic dimensionality is 1




**Let's look at some people's data**:

- we consider Height vs. Shoesize now
  (they look correlated, right?!)
- the covariance matrix looks like

$$\Sigma^2_{\mathrm{D}} = \begin{pmatrix} 388.7857 & \underline{82.9229} \\ \underline{82.9229} & 20.3942 \end{pmatrix}$$

with major off-diagonal values!

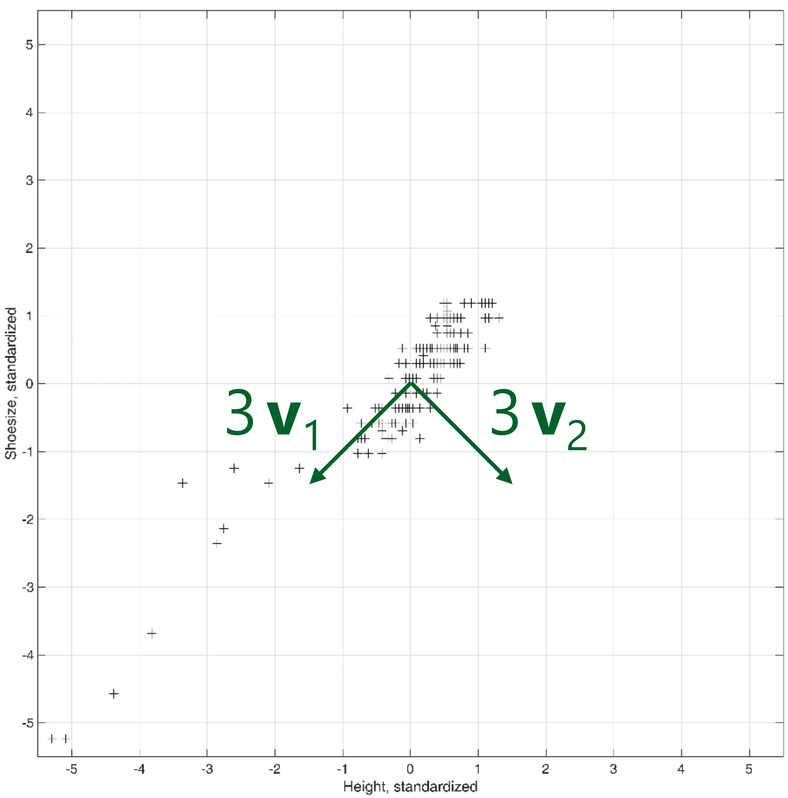
[see the MATLAB example 2d_E]

# PCA – Diagonalizing $\mathbf{\Sigma}^2$

**Next step** (after normalization/standardization)**:**

- transform such that the covariance matrix becomes diagonal!
  (since $\mathbf{\Sigma}^2$ is real-valued and symmetric, we use the eigendecomp.)

**This leads to the eigenvalues & eigenvectors of $\mathbf{\Sigma}^2$:**




- eigenvalues $\approx 1.93$ and $\approx 0.07$
  with the corresponding eigenvectors

$$\mathbf{V} = \begin{pmatrix} -0.7071 & 0.7071 \\ -0.7071 & -0.7071 \end{pmatrix}$$

  called *principal components* (PC)

- the vast difference in eigenvalues tells that this dataset is mostly 1D!

[see the MATLAB example 2d_E]

# PCA – Look at the Scores

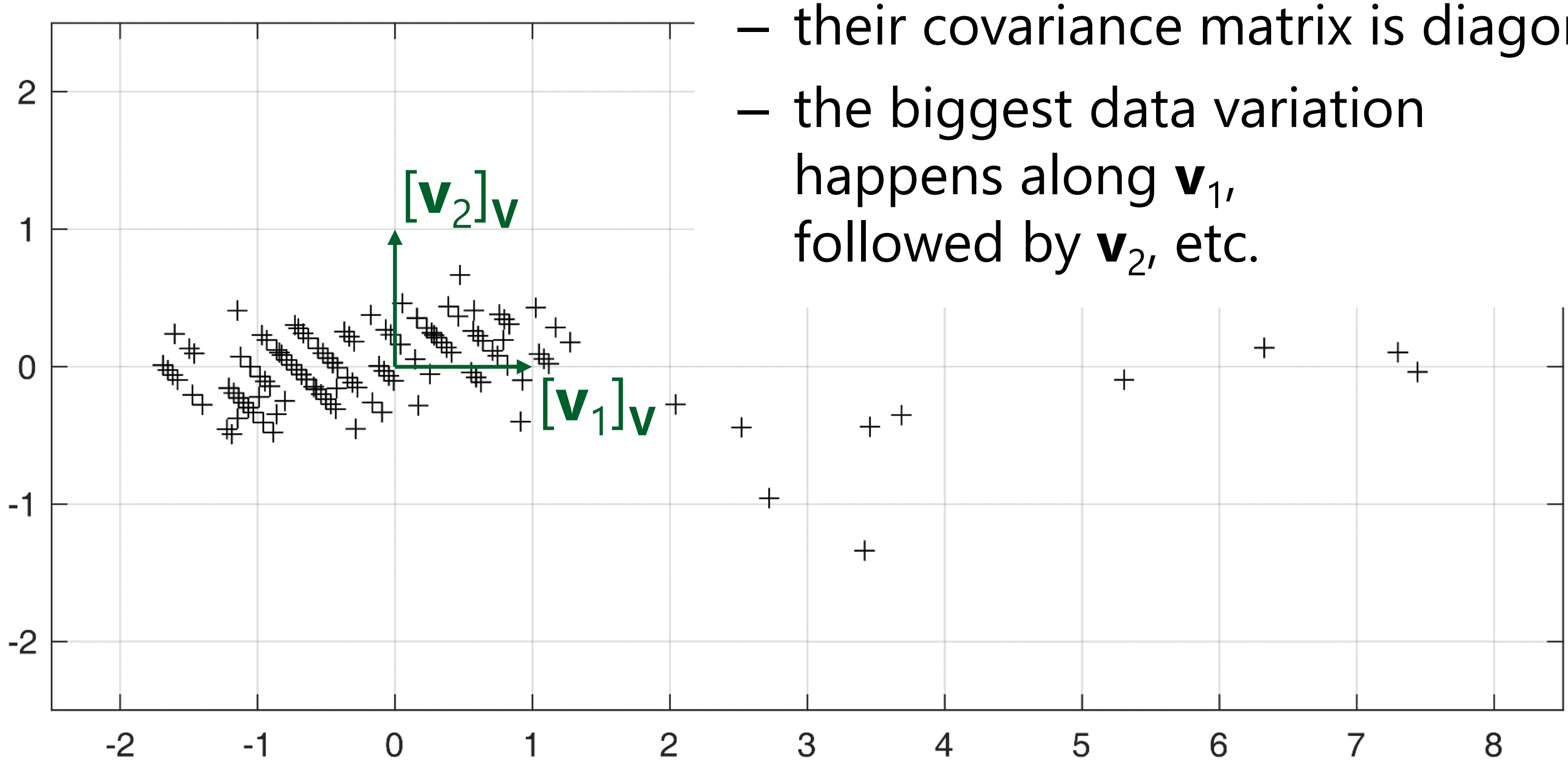
**Next step:**

– we can now look at the data in terms of $\mathbf{V}$, i.e., in the PC of $\mathbf{D}$ (the $[\mathbf{D}]_\mathbf{V}$ are called the *scores* in PCA)

**The scores are then uncorrelated**

– their covariance matrix is diagonal

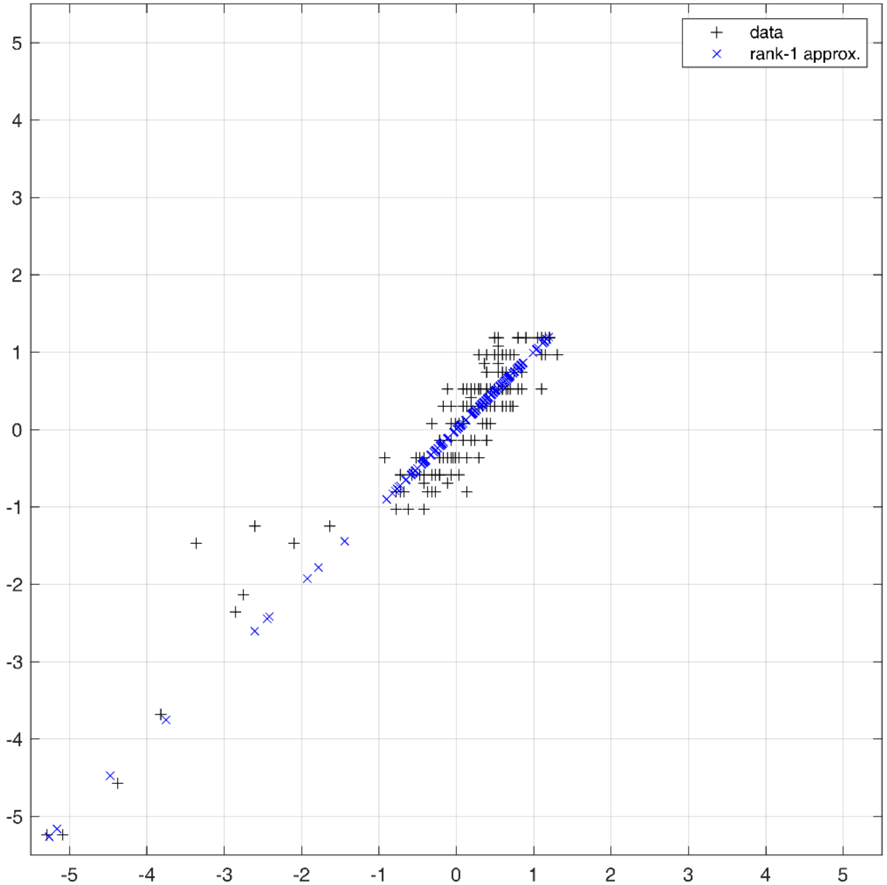
– the biggest data variation happens along $\mathbf{v}_1$, followed by $\mathbf{v}_2$, etc.



# PCA – Low-rank Approximation

## Similar to the SVD,

- we can compute a lower-($k$-)dimensional approximation of the (normalized) data **D** by "only" considering the largest eigenvalues




## Given scores $[\mathbf{D}]_{\mathbf{V}}$,

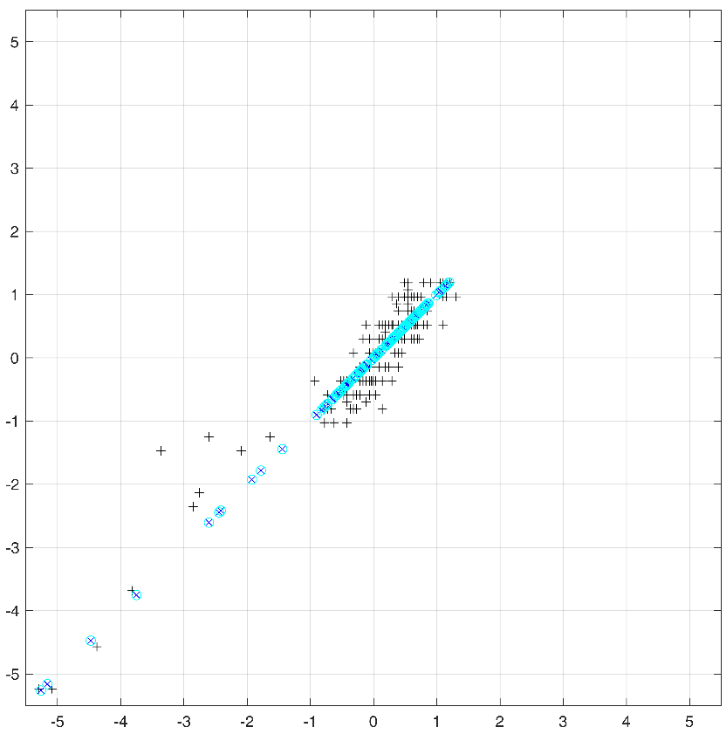
- we keep only the data rows corresponding to the largest $k$ eigenvalues
- assuming that we reordered $\mathbf{V}$ & $\mathbf{\Lambda}$ so that the eigenvalues in $\mathbf{\Lambda}$ became sorted in decreasing size
- we then compute $\mathbf{D}'_k$ by $\mathbf{D}'_k = (\mathbf{v}_1\ \mathbf{v}_2 \ldots \mathbf{v}_k)\ (\mathbf{d}_{\mathbf{V}1}{}^T\ \mathbf{d}_{\mathbf{V}2}{}^T \ldots \mathbf{d}_{\mathbf{V}k}{}^T)^T$

[see the MATLAB example 2d_E]

# PCA vs. SVD

**The lower-dimensional approximation with PCA overlaps with the rank-*k* approximation with SVD:**

- the eigenvalues from PCA, $\lambda_i$, are equal to $\sigma_i^2/(n-1)$
- the left-singular vectors **U** are the PC from PCA (up to sign change), if data rows are given (otherwise it's **V**)




**Relation:**

1. data **D**, *d* dims. x *n* items (in the cols.)
2. $\mathbf{\Sigma}^2 = \mathbf{D}\,\mathbf{D}^\mathrm{T} / (n-1)$ // covariance-matrix
3. $\mathbf{\Sigma}^2 = \mathbf{V}\,\mathbf{\Lambda}\,\mathbf{V}^*$ // eigendecomposition
4. $[\mathbf{D}]_\mathbf{V} = \mathbf{V}^*\,\mathbf{D}$ // scores
5. $\mathbf{D} = \mathbf{U}\,\mathbf{S}\,\mathbf{W}^*$ // SVD
6. $\mathbf{\Sigma}^2 = \mathbf{U}\,\mathbf{S}\,\mathbf{W}^*\,(\mathbf{W}\,\mathbf{S}^*\,\mathbf{U}^*) / (n-1)$ // 2. & 5.
   $= \underline{\mathbf{U}}\;\underline{(\mathbf{S}\,\mathbf{S}^* / (n-1))}\;\underline{\mathbf{U}^*} = \underline{\mathbf{V}}\,\underline{\mathbf{\Lambda}}\,\underline{\mathbf{V}^*}$ // 3.

# PCA – Investigate the Loadings (1)

**With PCA**

- we transform the data into a new Cartesian coordinate frame (the PCs are orthogonal to each other)
- so that the covariance between the scores is zero (we remove any redundancy from the data)

**Challenging** (as with the SVD)**:**

- considering the scores $[\mathbf{D}]_\mathbf{V}$ of data **D**, esp. after normalization, we usually lack a good interpretation of the new axes— they are linear combinations of the original data-axes, i.e., the "new data" $[\mathbf{D}]_\mathbf{V}$ is a mix (linear combination) of **D**
- one way to interpret the principal components, is to look at their *loadings*: to which degree play the original data-axes into them?

# PCA – Investigate the Loadings (2)

**The loadings**

- of PC $i$ are given by the components of eigenvector $\mathbf{v}_i$
- given data rows (items in cols.),
  - data item $j$ is given by column $j$ in $\mathbf{D}$: $(d_{\ldots,j})$
  - the scores of data item $j$ are then $(d_{\mathbf{V}\ldots,j}) = \mathbf{V}^* (d_{\ldots,j})$,
    i.e., $d_{\mathbf{V}1,j} = \mathbf{v}_1^{\mathsf{T}} (d_{\ldots,j})$, $d_{\mathbf{V}2,j} = \mathbf{v}_2^{\mathsf{T}} (d_{\ldots,j})$, aso.
- interpretation:
  - the “new” x-coordinate of some data item
    is a linear combination of all original attributes of this item,
    weighted by the entries of eigenvector #1,
    i.e.,
    a large absolute component #*k* in eigenvector #*g*
    tells about a major influence of data attribute #*k*
    onto the principal component #*g*

# PCA – Investigate the Loadings (3)

**Back to our example:**

– looking (again) at **V**, we see:

$$\mathbf{V} = \begin{pmatrix} -0.7071 & 0.7071 \\ -0.7071 & -0.7071 \end{pmatrix}$$

- the 1st principal component is a (negative) combination of Height and Shoesize, whereas
- the 2nd principal component expresses their disagreement (e.g., large feet, but short)

– taking also the vastly different eigenvalues (~1.93 vs. ~0.07) into account, we find that

- mostly, it's about the overall size (height & feet)
- minor differences exist (larger feet, while being a bit shorter)

# PCA – all of the data (1)

**Considering all of the people's data, i.e.,**

– the four data attributes Age, Height, Weight, and Shoesize

**Steps:** [see the MATLAB example 2d_F]

1. load the data
2. set up the data **D** (here data cols., again, i.e., items in the rows); compute also *n*, *d*, all per-dim. means **mu** & std.-dev. values **sigma**
3. normalize the data: **Dz** becomes the z-score of **D**
4. visualize to check

[…]

# PCA – all of the data (1)

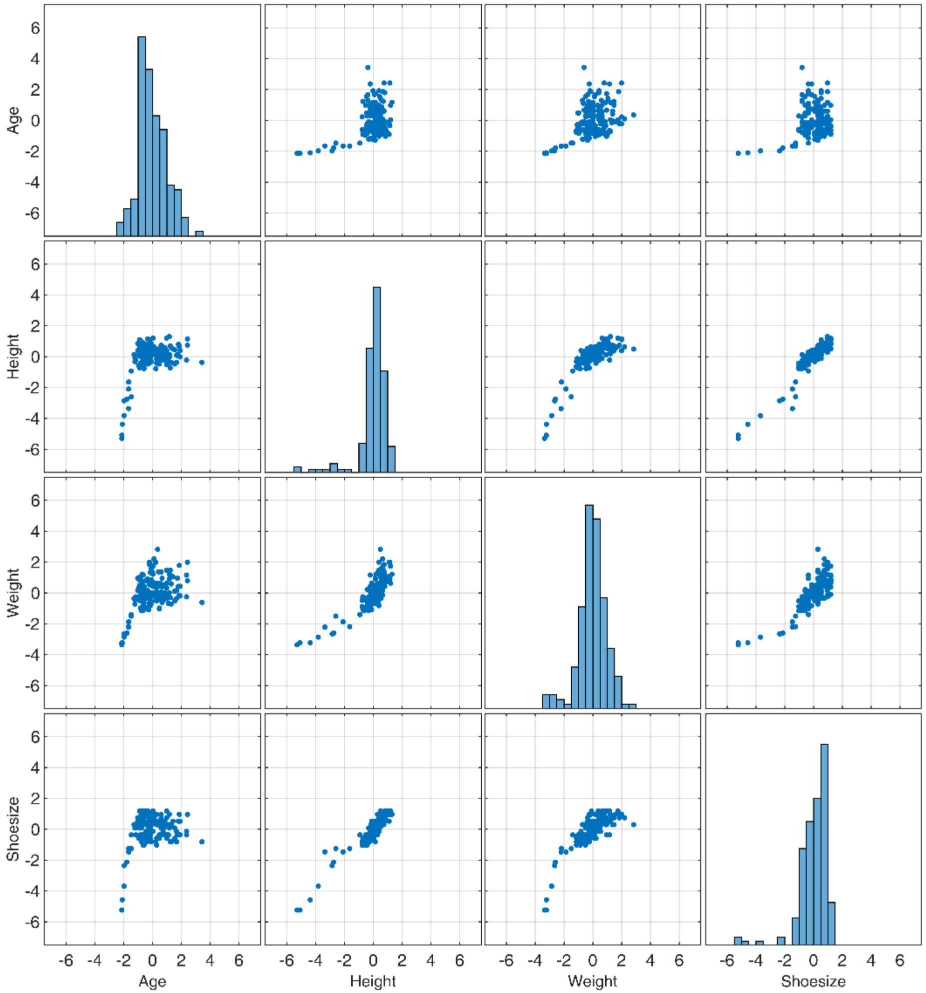
**Considering all of the people's data, i.e.,**

– the four data attributes Age, Height, Weight, and Shoesize

**Steps:**

1. load the data
2. set up the data **D** (here da
   compute also *n*, *d*, all per-
3. normalize the data: **Dz** b
4. visualize to check

[…]




[see the MATLAB example 2d_F]

# PCA – all of the data (2)

**Considering all of the people's data, i.e.,**

– the four data attributes Age, Height, Weight, and Shoesize

**Steps:**

[…]

5. compute the covariance matrix **S2z**
   and do the eigendecomposition into **Vz** and **Lz**
6. sort by the absolute eigenvalue value to get **LzS** and **VzS**
7. compute the scores **DzScores**
8. look at them

[…]

[see the MATLAB example 2d_F]

# PCA – all of the data (2)

**Considering all c**

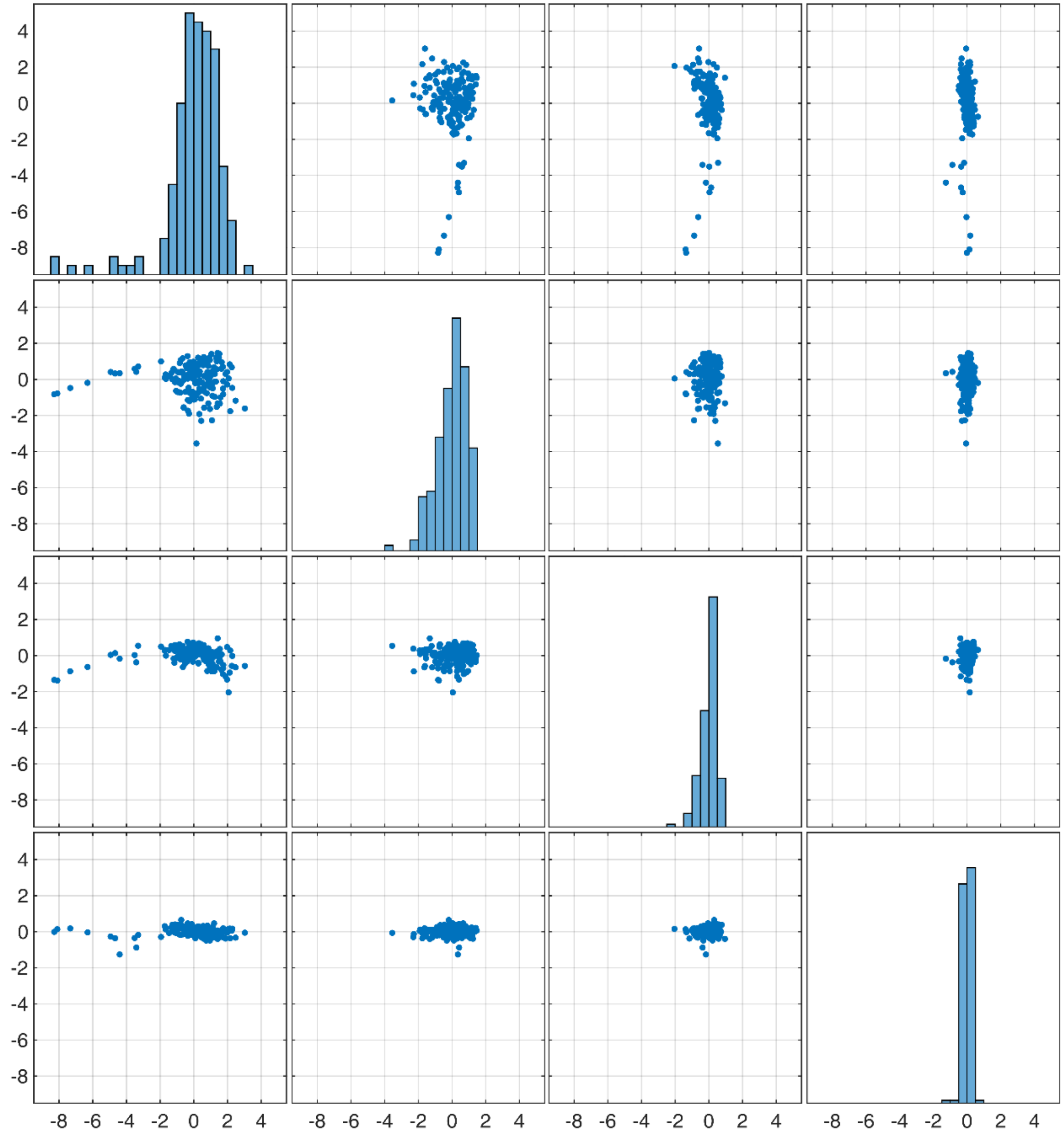
– the four data a

**Steps:**

[…]

5. compute the co
   and do the eige
6. sort by the abso
7. compute the sco
8. look at them

[…]




[see the MATLAB example 2d_F]

# PCA – all of the data (3)

**Considering all of the people's data, i.e.,**

- the four data attributes Age, Height, Weight, and Shoesize

**Steps:**

[…]

9. do a 2D-reconstruction **DzReco2**
10. do a comparative visualization
11. undo the normalization for the reconstruction, get **Dreco2**
12. visualize

[…]

[see the MATLAB example 2d_F]

# PCA – all of the data (3)

**Considering all**

– the four data

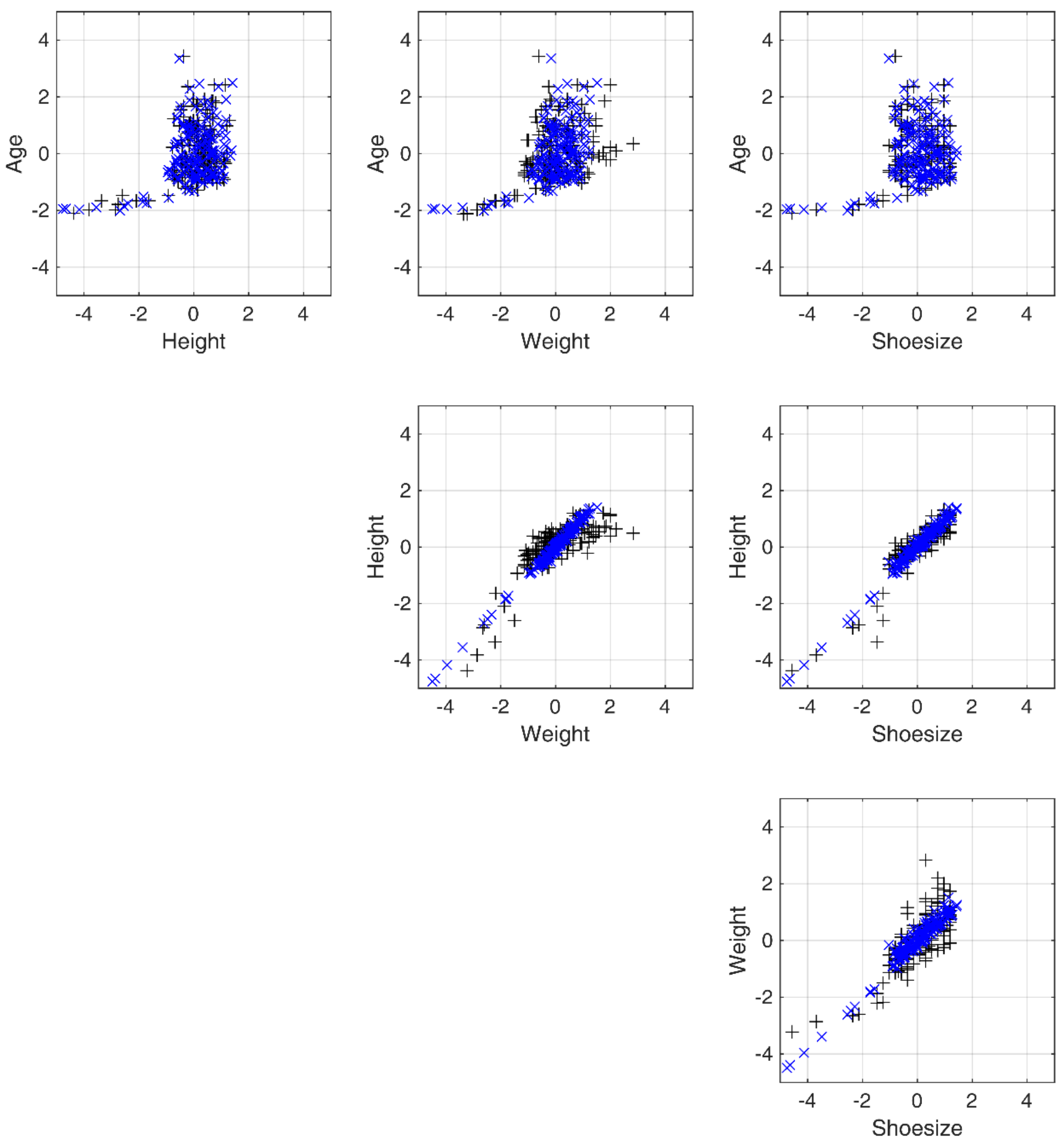
**Steps:**

[…]

9. do a 2D-reco
10. do a compar

11. undo the nor
12. visualize

[…]




[see the MATLAB example 2d_F]

# PCA – all of the data (3)

**Considering all of the people's data, i.e.,**

– the four data attributes Age, Height, Weight, and Shoesize

**Steps:**

[…]

9. do a 2D-reconstruction **DzReco2**
10. do a comparative visualization
11. undo the normalization for the reconstruction, get **Dreco2**
12. visualize

[…]

[see the MATLAB example 2d_F]

# PCA – all of the data (3)

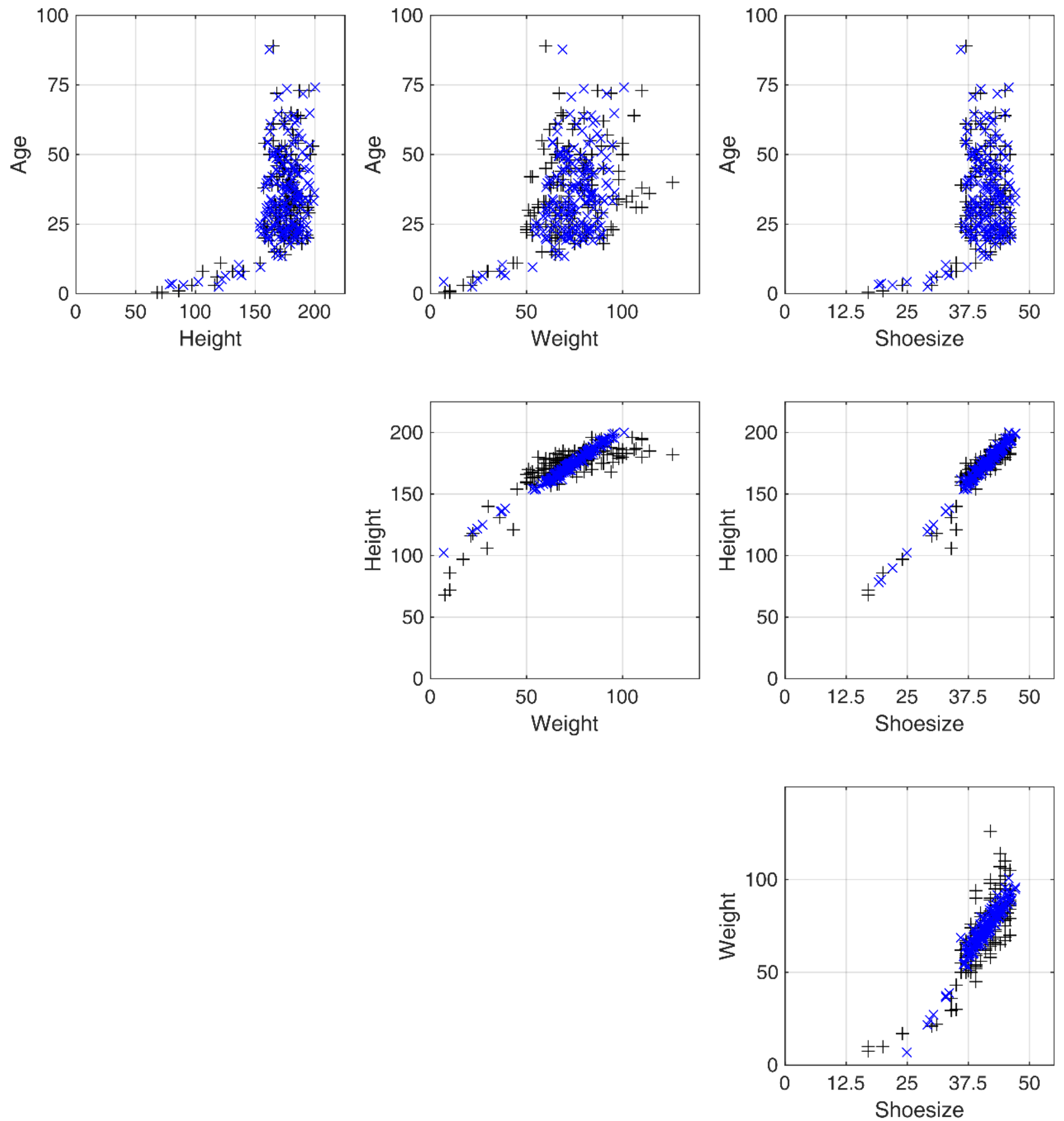
**Considering all**
- the four data

**Steps:**

[…]

9. do a 2D-recor
10. do a compara
11. undo the norr
12. visualize

[…]




[see the MATLAB example 2d_F

# PCA – all of the data (4)

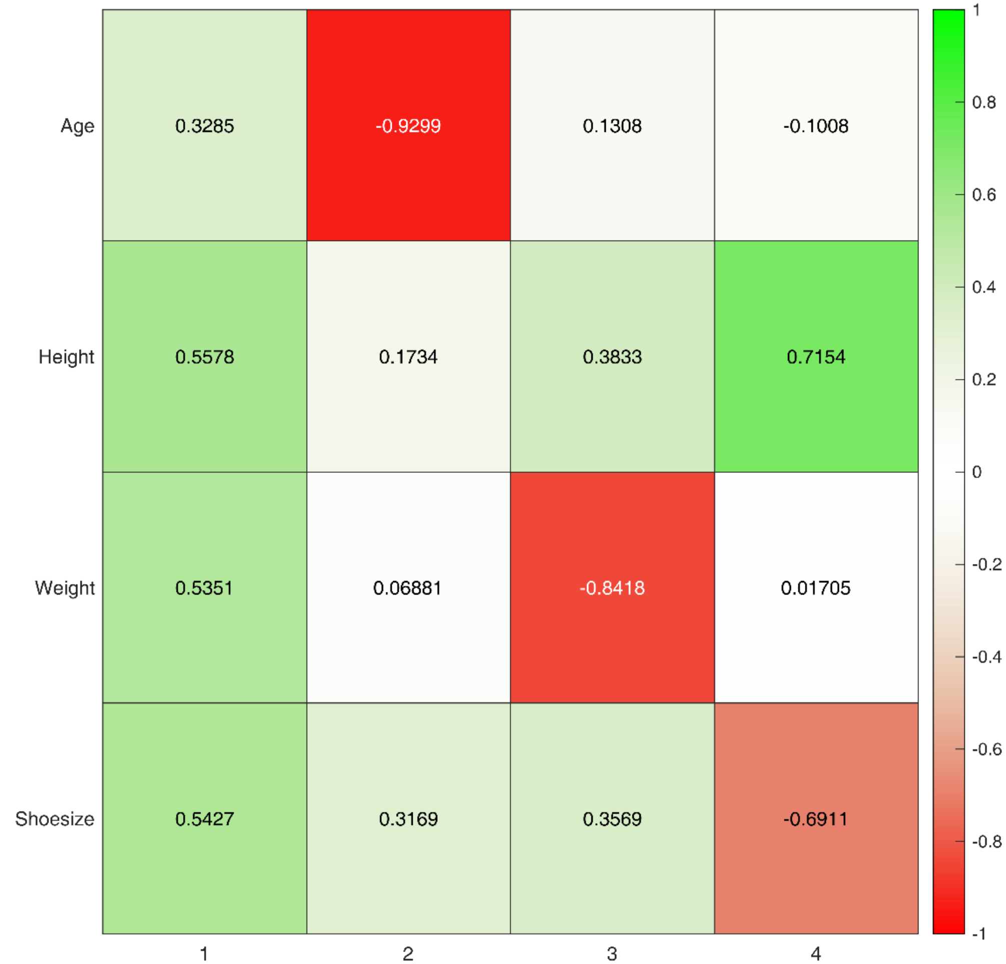
**Considering all of the people's data, i.e.,**

– the four data attributes Age, Height, Weight, and Shoesize

**Steps:**

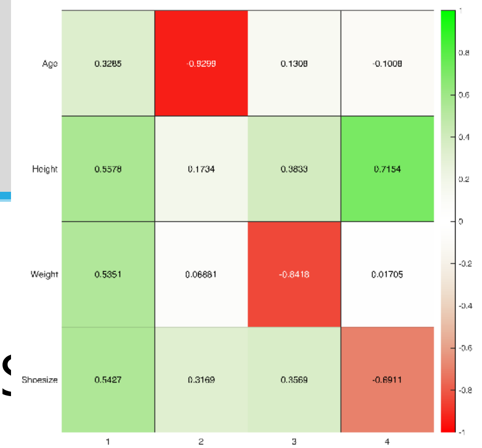
[…]

13. look at the loadings



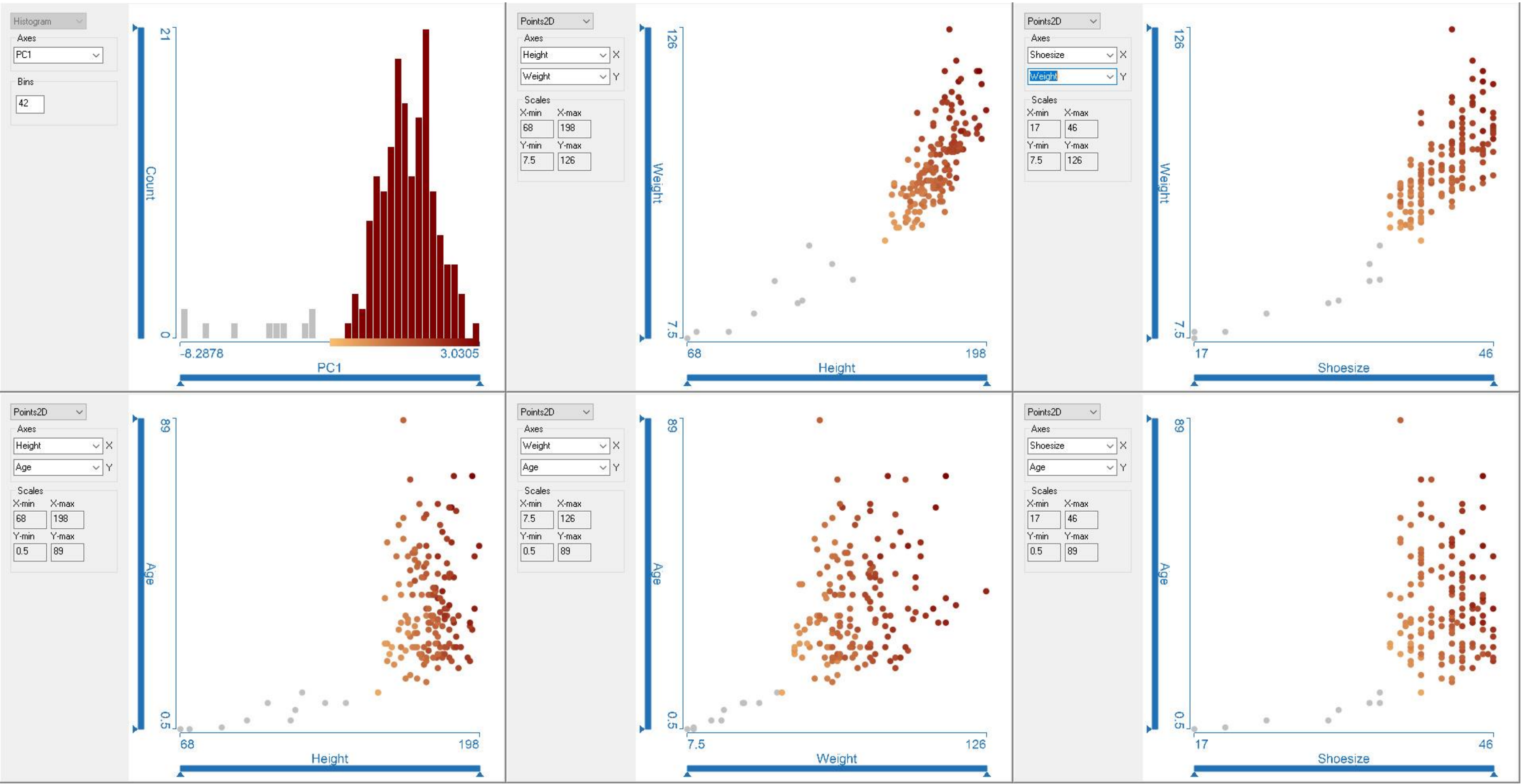

[see the MATLAB example 2d_F]

# PCA – all of the data (4)

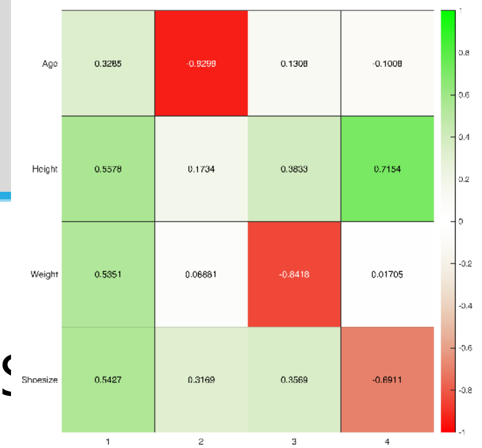



**Considering all of the people's data, i.e.,**

– the four data attributes Age, Height, Weight, and S

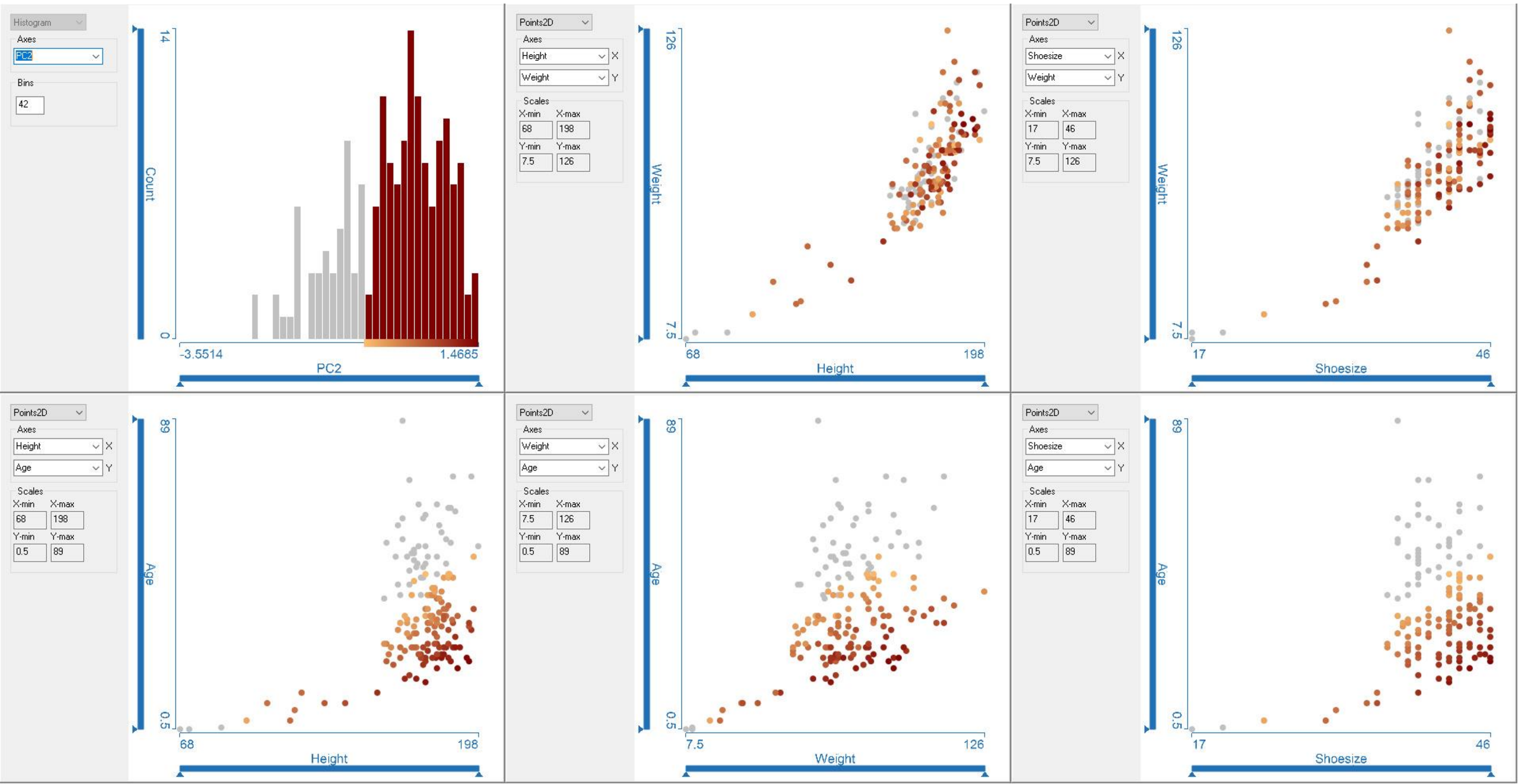


# PCA – all of the data (4)

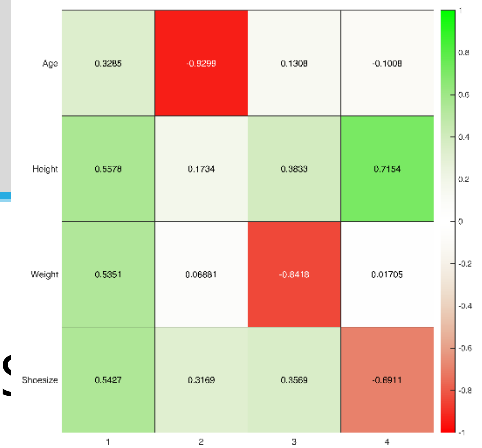



**Considering all of the people's data, i.e.,**

– the four data attributes Age, Height, Weight, and S

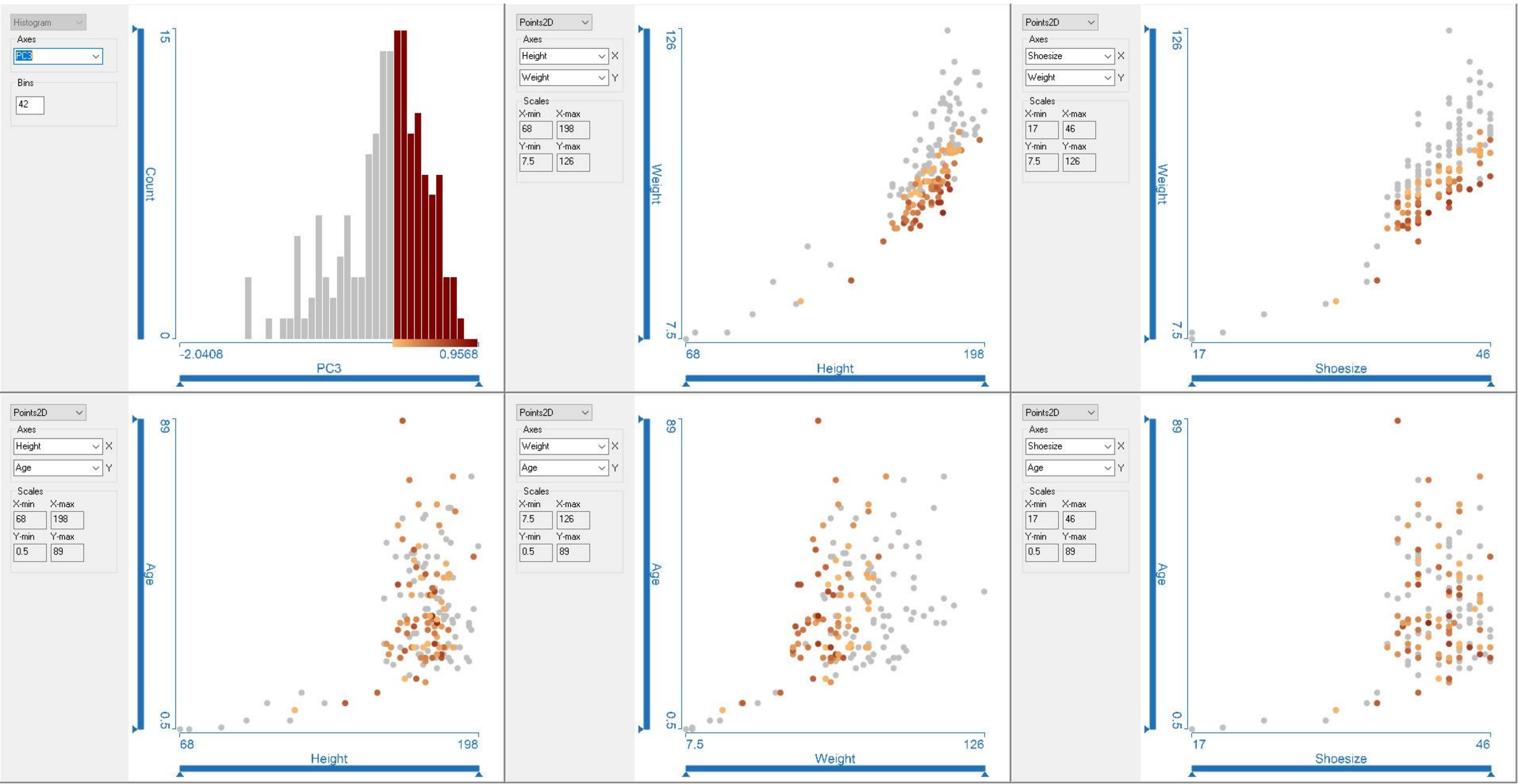


# PCA – all of the data (4)

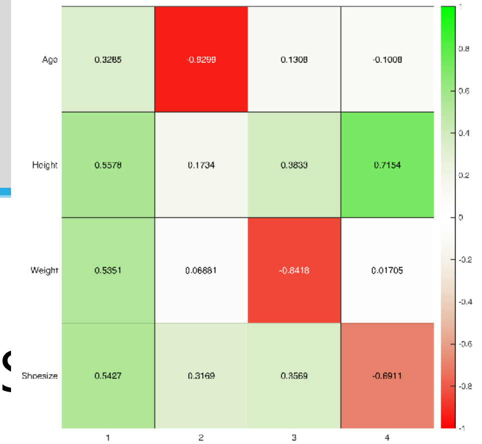



**Considering all of the people's data, i.e.,**

– the four data attributes Age, Height, Weight, and

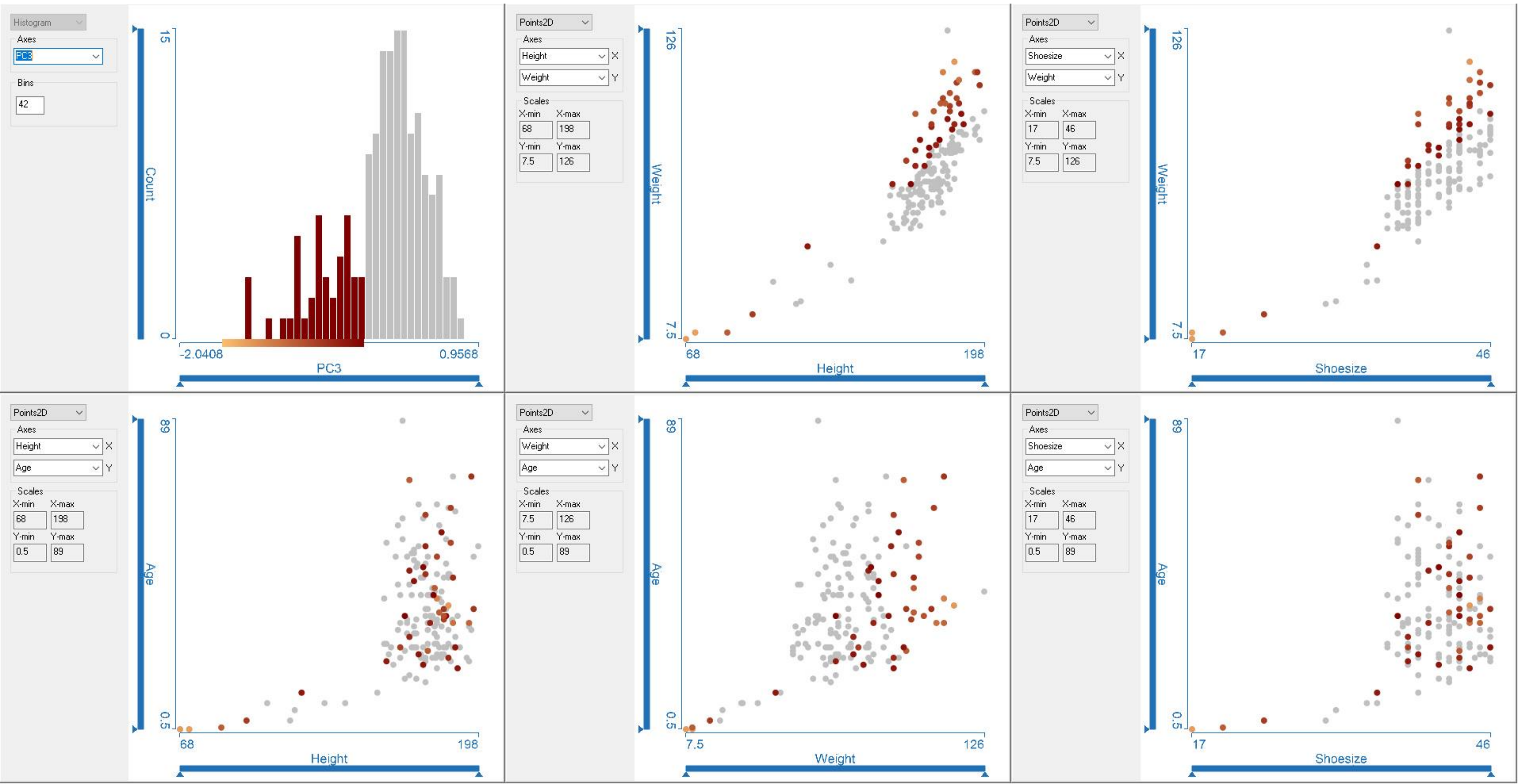


# PCA – all of the data (4)

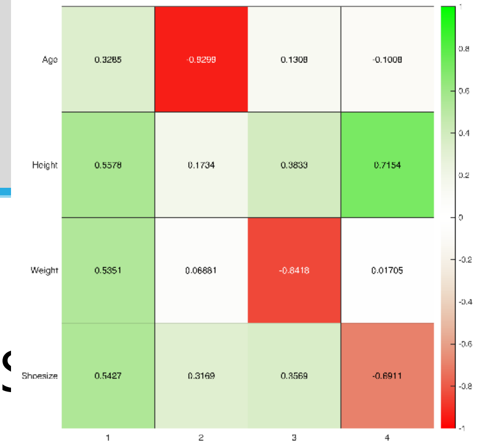



**Considering all of the people's data, i.e.,**

– the four data attributes Age, Height, Weight, and S

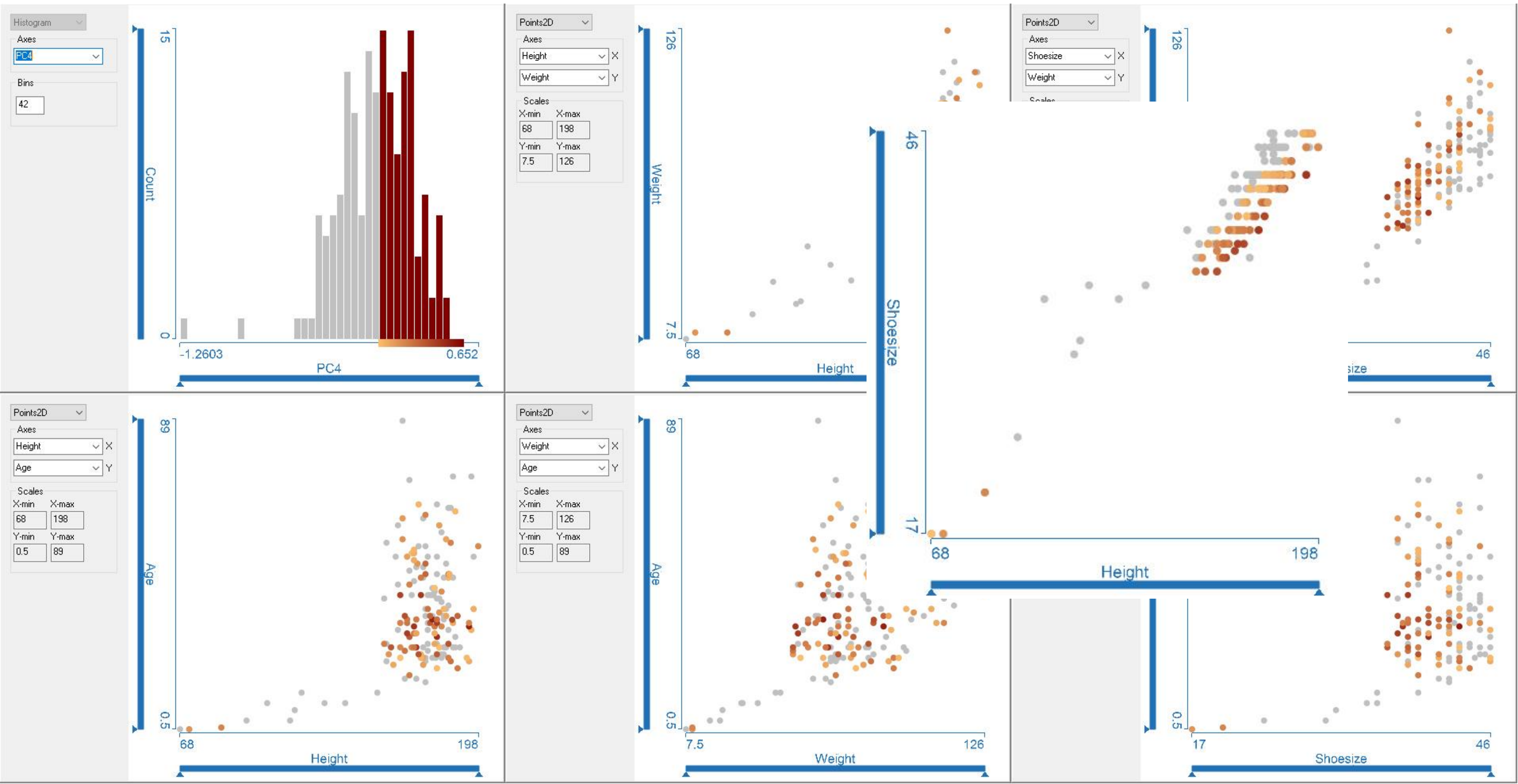


# PCA – all of the data (4)




**Considering all of the people's data, i.e.,**

– the four data attributes Age, Height, Weight, and S



# PCA – Remarks

**Principal Component Analysis (PCA)**

- can be seen as a ***best-possible linear technique for dimension reduction*** (best in the least sum of squared distances sense)
- is ***sensitive to the scaling of individual dimensions*** (therefore, often normalization is done dimension-wise first)
- this issue (PCA sensitive to dim.-scaling) implies that PCA is ***to a certain degree "arbitrary", when dimensions of different units are studied*** (that cannot be mapped to each other, like temperatures and costs)
- PCA ***makes only sense, when the data is centered***
- numerous extensions / alternative approaches exist, including non-linear techniques, sparse PCA, robust PCA, …

# How to Solve an Eigenproblem

**Focus up to here:**

- how to use an eigenanalysis

**Not touched** (yet)**:**

- how to solve an eigenproblem

**In short:**

- given $\mathbf{A}\,\mathbf{x} = \lambda\,\mathbf{x}$,
- we get to $(\mathbf{A}-\lambda\mathbf{I})\mathbf{x} = 0$
  - if $(\mathbf{A}-\lambda\mathbf{I})$ is non-singular, then only $\mathbf{x} = \mathbf{0}$ is a solution
  - thus, and in order to find non-degenerated solutions, we need to require that $(\mathbf{A}-\lambda\mathbf{I})$ is singular, i.e., $\det(\mathbf{A}-\lambda\mathbf{I}) = 0$
- $\det(\mathbf{A}-\lambda\mathbf{I}) = 0$ is called the characteristic equation of $\mathbf{A}$, leading to the eigenvalues $\lambda_i$
- having the $\lambda_i$, we can solve $(\mathbf{A}-\lambda\mathbf{I})\mathbf{x} = 0$ for the $\mathbf{x}_i$

# Outlook

**Next**:

- a new part of INF250 with
  - data fitting (somehow a continuation of this part!)
  - splines
  - iterative methods

# 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