Reducing dimensionality

Principal components

R.W. Oldford

University of Waterloo

Reducing dimensions

Recall how orthogonal projections work.

Given $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_k]$, an orthogonal projection matrix $\mathbf{P}$ is easily constructed as

$$\mathbf{P} = \mathbf{V}\left(\mathbf{V}^{\mathsf{T}}\mathbf{V}\right)^{-1}\mathbf{V}^{\mathsf{T}}.$$

And, if the column vectors of $\mathbf{V}$ form an **orthonormal basis** for $\mathcal{S}$, then

$$\mathbf{P} = \mathbf{V}\mathbf{V}^{\mathsf{T}}.$$

So far, we have only considered the choice where the $\mathbf{v}_i$ s are unit vectors in the direction of the original data axes $\mathbf{e}_i$.

Projections onto these directions simply return the scatterplots on pairs of the original variates.

Are there other directions which would do as well (or possibly better)?

Imagine that we have n points $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^p$ centred so that $\sum_{i=1}^{n} \mathbf{x}_i = \mathbf{0}$ and we denote by $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^{\mathsf{T}}$ the $n \times p$ real matrix whose ith row is $\mathbf{x}_i^{\mathsf{T}}$.

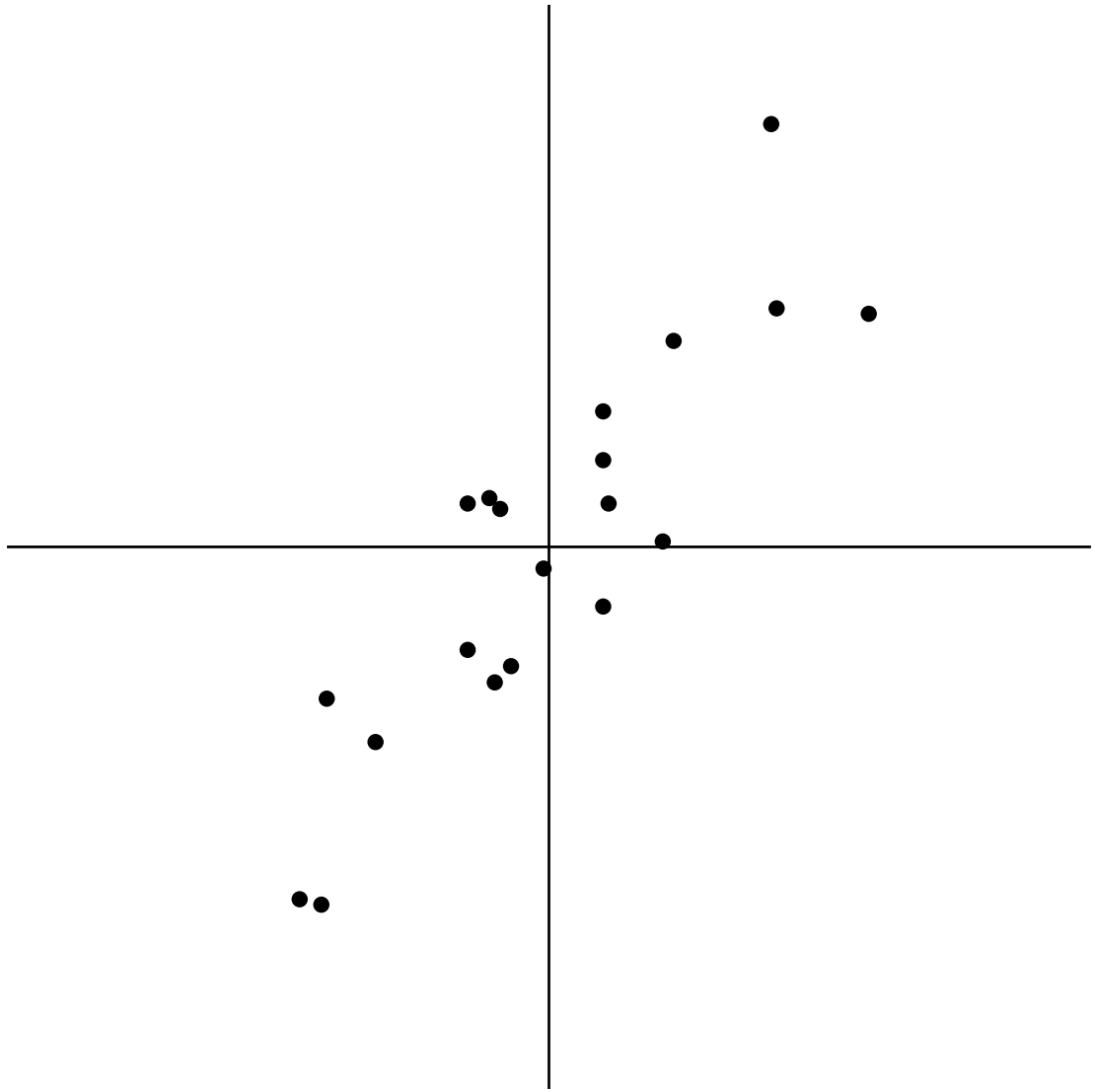
Being centred, we can now ask whether the data truly lie in a lin ear subspace of $\mathbb{R}^p$. And, if they do, can we find that subspace?

Alternatively, do they lie **nearly** in a linear subspace and could we find it?

University of Waterloo

Reducing dimensions - Finding the principal axes

For example suppose $n = 20$ and $p = 2$ so that a point cloud might look like:

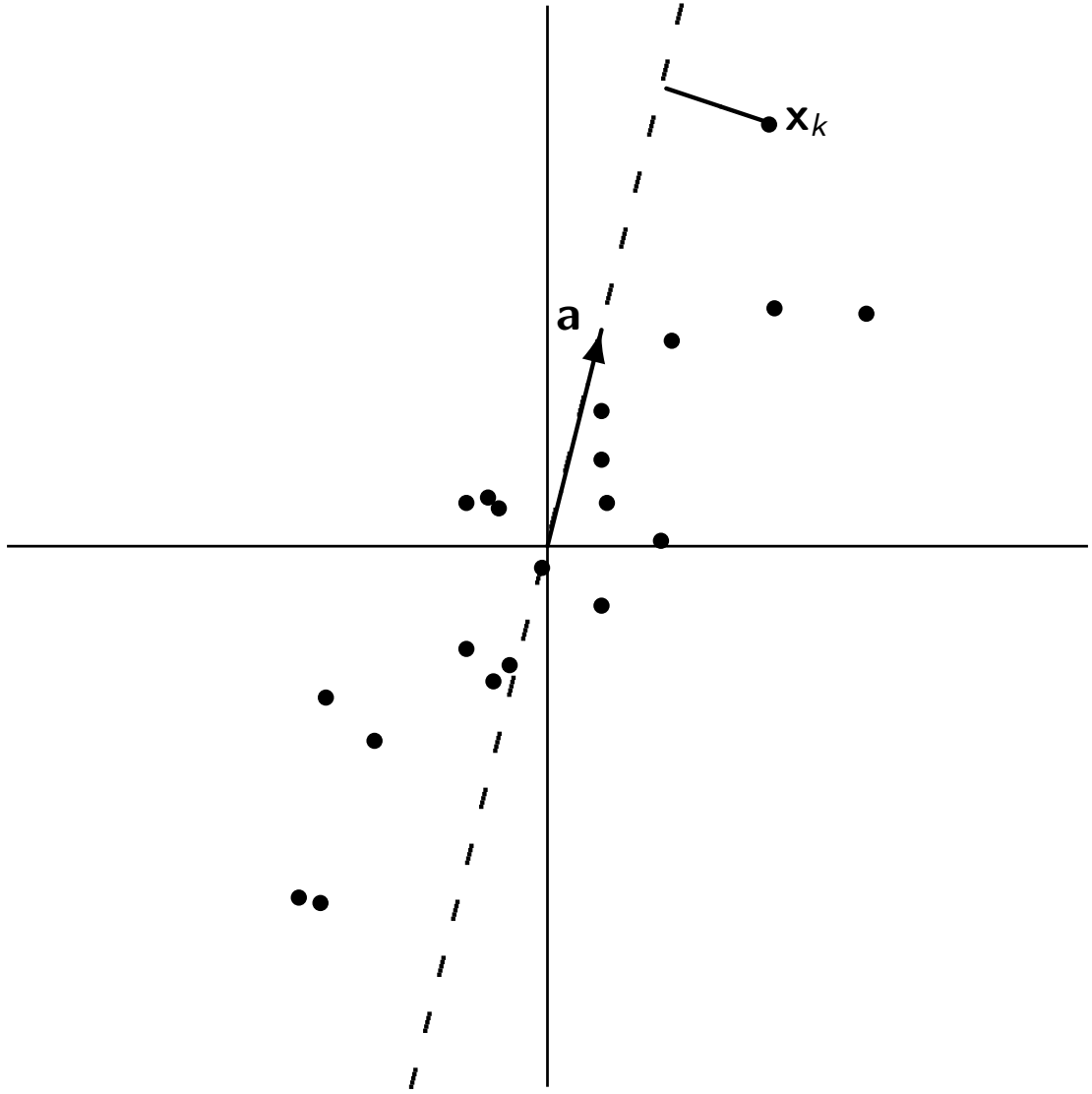


The points $\mathbf{x}_1, \ldots, \mathbf{x}_n$ lie in the plane, but do not occupy all of it. They appear nearly to lie in a one-dimensional subspace of $\mathbb{R}^2$

University of Waterloo

Reducing dimensions - Finding the principal axes

We can think about orthogonally projecting the points (or equivalently, vectors) $\mathbf{x}_1, \ldots, \mathbf{x}_n$ onto any direction vector $\mathbf{a} \in \mathbb{R}^p$ (i.e. $\|\mathbf{a}\| = 1, \quad \mathbf{a} \in \mathbb{R}^p$).



University of Waterloo

Reducing dimensions - Finding the principal axes

The orthogonal projection of the point $\mathbf{x}_k$ onto $\mathbf{a}$ (i.e. onto the $span\{\mathbf{a}\}$) is

$$(\mathbf{a}\mathbf{a}^\mathsf{T})\mathbf{x}_k = \mathbf{a} \times w_k$$

a vector in the direction of $\mathbf{a} \times \text{sign}(w_k)$ of length $|w_k|$ with $w_k = \mathbf{a}^\mathsf{T}\mathbf{x}_k$.

Note that the squared length of this projection is

$$w_k{}^2 = ||\mathbf{a}\mathbf{a}^\mathsf{T}\mathbf{x}_k||^2 = \mathbf{x}_k{}^\mathsf{T}\mathbf{a}\mathbf{a}^\mathsf{T}\mathbf{x}_k = \mathbf{a}^\mathsf{T}\mathbf{x}_k\mathbf{x}_k{}^\mathsf{T}\mathbf{a}.$$

Since every point can be projected onto the direction vector $\mathbf{a}$, we might ask what vector would maximize (or minimize) the sum of the squared lengths of the projections?

That is, onto which direction would the projections have the largest average squared length?

Because the points are already centred about $\mathbf{0}$, this is the same as asking which direction are the original data points most (or least) spread out (or variable)?

University of Waterloo

Reducing dimensions - Finding the principal axes

Mathematically we want to find the direction vector $\mathbf{a}$, which maximizes (minimizes) the sum $\sum_{k=1}^{n} w_k^2$.

This sum can in turn be expressed in terms of the original points in the point cloud as:

$$\sum_{k=1}^{n} {w_k}^2 = \sum_{k=1}^{n} {\mathbf{x}_k}^{\mathsf{T}} \mathbf{a}\mathbf{a}^{\mathsf{T}} \mathbf{x}_k$$

$$= \sum_{k=1}^{n} \mathbf{a}^{\mathsf{T}} \mathbf{x}_k {\mathbf{x}_k}^{\mathsf{T}} \mathbf{a}$$

$$= \mathbf{a}^{\mathsf{T}} \left(\sum_{k=1}^{n} \mathbf{x}_k {\mathbf{x}_k}^{\mathsf{T}} \right) \mathbf{a}$$

$$= \mathbf{a}^{\mathsf{T}} \left([\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n] \begin{bmatrix} {\mathbf{x}_1}^{\mathsf{T}} \\ {\mathbf{x}_2}^{\mathsf{T}} \\ \vdots \\ {\mathbf{x}_n}^{\mathsf{T}} \end{bmatrix} \right) \mathbf{a}$$

$$= \mathbf{a}^{\mathsf{T}} (\mathbf{X}^{\mathsf{T}} \mathbf{X}) \mathbf{a}$$

where $\mathbf{X}^{\mathsf{T}} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]$ is the $p \times n$ matrix of data vectors.

University of Waterloo

Reducing dimensions - Finding the principal axes

The maximization (minimization) problem can now be expressed as follows.

Find $\mathbf{a} \in \mathbb{R}^p$ which maximizes (minimizes)

$$\mathbf{a}^\mathsf{T}(\mathbf{X}^\mathsf{T}\mathbf{X})\mathbf{a}$$

subject to the constraint that $\mathbf{a}^\mathsf{T}\mathbf{a} = 1$.

We can write this as an unconstrained optimization by introducing a *Lagrange multiplier* λ. The problem then becomes to find $\lambda \in \mathbb{R}$ and $\mathbf{a} \in \mathbb{R}^p$ which maximizes (minimizes)

$$\mathbf{a}^\mathsf{T}(\mathbf{X}^\mathsf{T}\mathbf{X})\mathbf{a} + \lambda(1 - \mathbf{a}^\mathsf{T}\mathbf{a})$$

which we now simply differentiate with respect to $\mathbf{a}$, set to zero, solve, etc.

Note that the objective function to be maximized (minimized) is a **quadratic form** in $\mathbf{a}$. More generally, a quadratic form in $\mathbf{z} \in \mathbb{R}^p$ can always be written as

$$Q(\mathbf{z}) = \mathbf{z}^\mathsf{T}\mathbf{A}\mathbf{z} + \mathbf{b}^\mathsf{T}\mathbf{z} + c$$

where $\mathbf{A} \in \mathbb{R}^{p \times p}$, $\mathbf{b} \in \mathbb{R}^p$, and $c \in \mathbb{R}$ are all constants (wlog $\mathbf{A} = \mathbf{A}^\mathsf{T}$).

University of Waterloo

Reducing dimensions - Finding the principal axes

Differentiating the quadratic form

$$Q(\mathbf{z}) = \mathbf{z}^\mathsf{T}\mathbf{A}\mathbf{z} + \mathbf{b}^\mathsf{T}\mathbf{z} + c$$

with respect to the vector $\mathbf{z}$ is

$$\frac{\partial}{\partial \mathbf{z}} Q(\mathbf{z}) = 2\mathbf{A}\mathbf{z} + \mathbf{b}.$$

For our problem, we have the variable vector $\mathbf{z} = \mathbf{a}$ and constants $c = \lambda$, $\mathbf{b} = \mathbf{0}$, and $\mathbf{A} = \mathbf{X}^\mathsf{T}\mathbf{X} - \lambda\mathbf{I}_p$. Differentiating with respect to $\mathbf{a}$ and setting the result to $\mathbf{0}$ gives the set of equations:

$$2(\mathbf{X}^\mathsf{T}\mathbf{X} - \lambda\mathbf{I}_p)\mathbf{a} = \mathbf{0}$$

or

$$(\mathbf{X}^\mathsf{T}\mathbf{X})\mathbf{a} = \lambda\mathbf{a}.$$

Differentiating with respect to λ, setting to zero and solving yields $\mathbf{a}^\mathsf{T}\mathbf{a} = 1$.

Which should look familiar . . . ?

University of Waterloo

Reducing dimensions - Finding the principal axes

The solution to the systems of equations

$$(\mathbf{X}^T\mathbf{X})\mathbf{a} = \lambda\mathbf{a} \quad \text{and} \quad \mathbf{a}^T\mathbf{a} = 1$$

are the *eigen-vector* $\mathbf{a}$ and its corresponding *eigen-value* λ of the real symmetric matrix $\mathbf{X}^T\mathbf{X}$. The quadratic form we are maximizing (minimizing) is

$$\mathbf{a}^T(\mathbf{X}^T\mathbf{X})\mathbf{a} = \lambda\mathbf{a}^T\mathbf{a} = \lambda.$$

To maximize (minimize) this quadratic form, we choose the eigen-vector corresponding to the largest (smallest) eigen-value.

Denote the solution to this problem as $\mathbf{v}_1$ (or $\mathbf{v}_p$ for the minimization problem).

The solution $\mathbf{v}_1$ (or $\mathbf{v}_p$) will be the eigen-vector of $(\mathbf{X}^T\mathbf{X})$ which corresponds to its largest (or smallest) eigen-value λ_1 (or λ_p).

Putting all eigen-vectors into an orthogonal matrix $\mathbf{V} = [\mathbf{v}_1, \cdots, \mathbf{v}_p]$ ordered by the eigen-values $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p$ we have

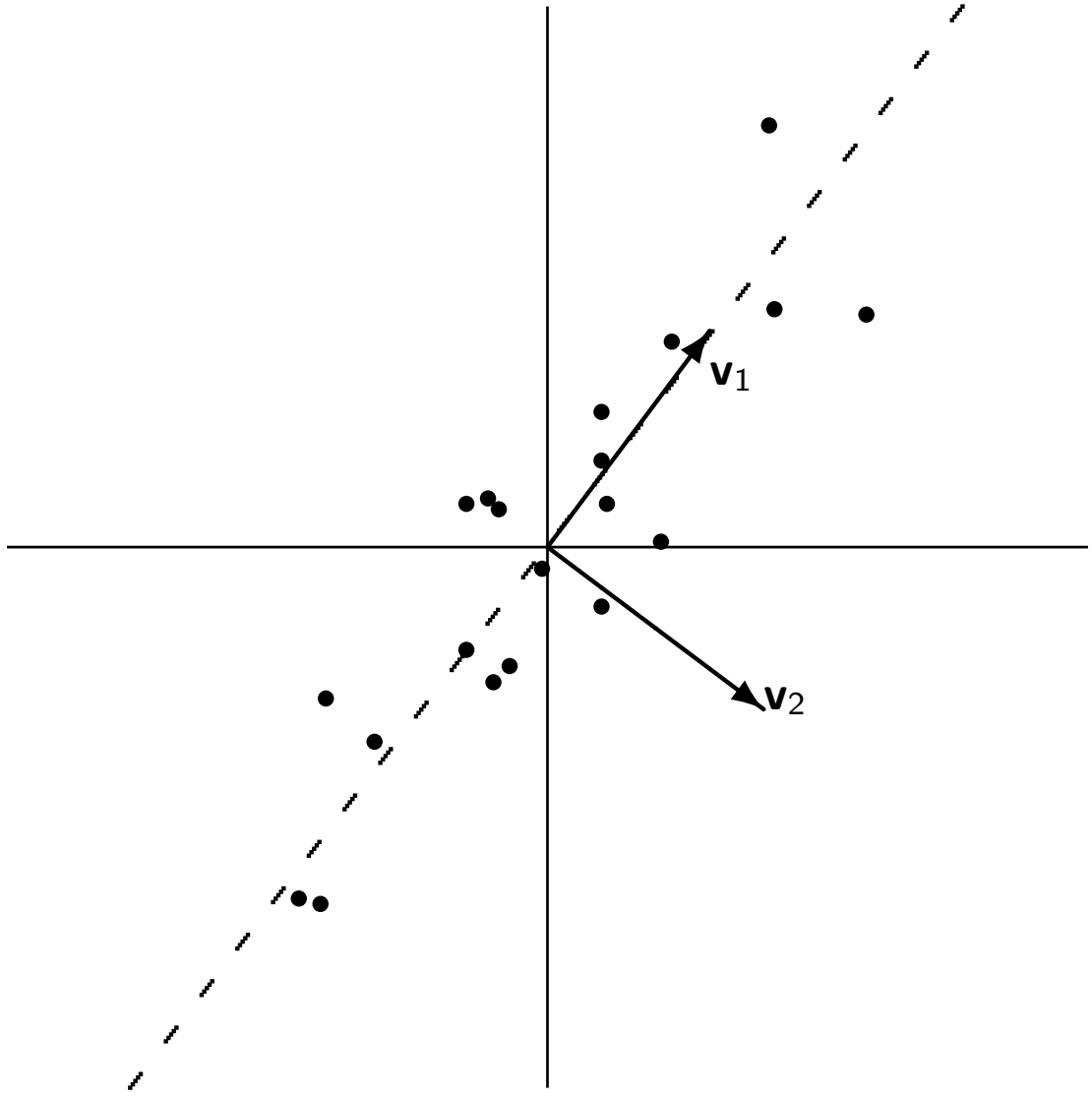
$$(\mathbf{X}^T\mathbf{X}) = \mathbf{V}\mathbf{D}_\lambda\mathbf{V}^T$$

is the **eigen-decomposition** of $\mathbf{X}^T\mathbf{X}$ with $\mathbf{D}_\lambda = diag(\lambda_1, \ldots, \lambda_p)$.

University of Waterloo

Reducing dimensions - Finding the principal axes

The figure below shows $\mathbf{v}_1$ (and $\mathbf{v}_2$) for this data.



University of Waterloo

Reducing dimensions - Finding the principal axes

Consider a change of variables $\mathbf{y} = \mathbf{V}^T\mathbf{x}$ (with $\mathbf{V}^T\mathbf{V} = \mathbf{I}_p = \mathbf{V}\mathbf{V}^T$) or equivalently $\mathbf{V}\mathbf{y} = \mathbf{x}$. The data in the original coordinate system has

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} = \mathbf{I}_p \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} = [\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_p] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix}$$

and, expressed in the new coordinate system, is expressed as

$$\mathbf{x} = \mathbf{V}\mathbf{y} = [\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_p] \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_p \end{bmatrix} \qquad (= \mathbf{V}\mathbf{V}^T\mathbf{x} = \mathbf{x})$$

The values $y_1, \ldots, y_p$ are coordinates on new axes $\mathbf{v}_1, \ldots \mathbf{v}_p$. The axes $\mathbf{v}_1, \ldots, \mathbf{v}_p$ are called the **principal axes** and the values or variables $y_1, \ldots, y_p$ the **principal coordinates** or **principal components**.

- Sometimes both axes and coordinates/variables are called the principal components. (. . . sigh)
- Values of variable y_1 are most spread out, . . . , the values of variable y_p are least spread out.
- y_i and y_j are uncorrelated for $i \neq j$ (the $\mathbf{y}$ points are now aligned along their principal axes).
- Alternatively, holding the axes fixed, matrix $\mathbf{V}^T$ rotates each $\mathbf{x}$ into position $\mathbf{y} = \mathbf{V}^T\mathbf{x}$.

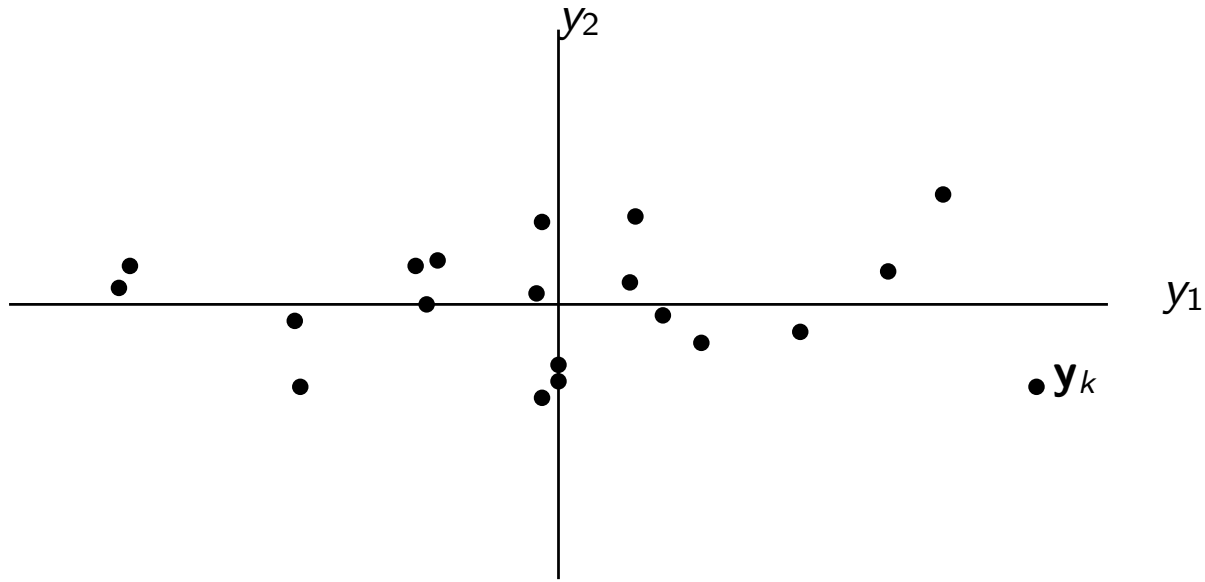
University of Waterloo

Reducing dimensions - Finding the principal axes

For our example, the transform $\mathbf{y} = \mathbf{V}^T\mathbf{x}$ yields

$$y_1 = \mathbf{v}_1{}^T\mathbf{x} = c_1x_1 + c_2x_2 + \cdots + c_px_p$$

as a weighted linear combination of the original x variables (using entries of $\mathbf{v}_1$ as weights). The following figure shows the points as they appear in the new co-ordinate system (e.g. $\mathbf{y}_k = \mathbf{V}^T\mathbf{x}_k$).



Note that the transformation rotates the points into position and (in this example) reflects them through one (or more) of the **principal axes**.

University of Waterloo

Reducing dimensions - Finding the principal axes

For our example, consider only the `Species == "versicolor"` of the `iris` data and the first three variates. In three dimensions, the plot looks like

```
library(loon)
data <- l_scale3D(iris[iris$Species == "versicolor", 1:4])
p3D <- l_plot3D(data[,1:3], showGuides = TRUE)
plot(p3D)
```

Which can now be rotated by hand to get to the principal components.

University of Waterloo

Reducing dimensions

What do the eigen-values represent?

The eigen equation is ($\forall\ j = 1, \ldots, p$):

$$
\begin{aligned}
(\mathbf{X}^\mathsf{T}\mathbf{X})\mathbf{v}_j = \lambda_j \mathbf{v}_j \quad &\Longrightarrow \quad \mathbf{v}_j{}^\mathsf{T}(\mathbf{X}^\mathsf{T}\mathbf{X})\mathbf{v}_j \;=\; \lambda_j \mathbf{v}_j{}^\mathsf{T}\mathbf{v}_j \\
&\Longrightarrow \quad (\mathbf{X}\mathbf{v}_j)^\mathsf{T}(\mathbf{X}\mathbf{v}_j) \;=\; \lambda_j \times 1 \\
&\Longrightarrow \quad \mathbf{z}^\mathsf{T}\mathbf{z} \;=\; \lambda_j \\
&\Longrightarrow \quad \textstyle\sum_{i=1}^{n} z_i^2 \;=\; \lambda_j \geq 0
\end{aligned}
$$

- $z_i = \mathbf{v}_j{}^\mathsf{T}\mathbf{x}_i$ is the jth coordinate of the data point $\mathbf{x}_i$ projected onto the principal axis $\mathbf{v}_j$.
- $\sum_{i=1}^{n} \mathbf{x}_i = \mathbf{0} \Longrightarrow \sum_{i=1}^{n} z_i = 0$ which means $\lambda_j = \sum_{i=1}^{n} z_i^2$ is proportional to the sample variance of the data when projected onto the jth principal axis.

If $\lambda_j = 0$, then

- the sample variance of the data along the jth principal axis is zero,
- the projection of every point onto the direction $\mathbf{v}_j$ is identically 0, and
- because the λs are non-negative and ordered,

$$\lambda_j = 0 \Longrightarrow \lambda_{j+1} = \cdots = \lambda_p = 0.$$

University of Waterloo

Reducing dimensions

What do the eigen-values tell us?

Geometrically then,

the data lie in a space orthogonal to $\mathbf{v}_j$.

In fact,

the data lie in a space orthogonal to $\text{span}\{\mathbf{v}_j, \mathbf{v}_{j+1}, \ldots, \mathbf{v}_p\}$.

If so, then the coordinates z_k for $k \geq j$ might be thrown away without any loss in the geometry of the point configuration!

That is, **suppose that** there were some integer d with $1 < d < p$ such that

- $\lambda_1 \geq \cdots \geq \lambda_d > 0$, and $\lambda_j = 0$ for $j > d$,

it would follow that

- the points $\mathbf{x}_1, \ldots, \mathbf{x}_n$ lie in a d-dimensional subspace of $\mathbb{R}^p$, and
- that space is defined by the principal axes $\mathbf{v}_1, \ldots, \mathbf{v}_d$.

That is $\mathbf{x}_i \in \text{span}\{\mathbf{v}_1, \ldots, \mathbf{v}_d\} \subset \mathbb{R}^p$ for all $i = 1, \ldots, n$, so we could replace each $\mathbf{x}_i \in \mathbb{R}^p$ by the d-dimensional vector of the first d coordinates of $\mathbf{y} = \mathbf{V}^\mathsf{T}\mathbf{x}$!

Question: What if we only have $\lambda_j \approx 0$ for $j > d$?
Answer: The points $\mathbf{x}_1, \ldots, \mathbf{x}_n$ **nearly** lie in a d-dimensional subspace.

Perhaps we can still reduce consideration to d dimensions but now with **some** loss of information about the geometry of the points.

- How much loss? Depends on the values of $\lambda_{d+1} \geq \cdots \geq \lambda_p$.

University of Waterloo

Reducing dimensions

The singular value decomposition

Let $\mathbf{Y}^\mathsf{T} = [\mathbf{y}_1, \ldots, \mathbf{y}_n]$ be the $p \times n$ matrix of the points $\mathbf{y}_1, \ldots, \mathbf{y}_n$ in the coordinate system of the principal axes.

Note that these coordinates in the new space are simply given by

$$\mathbf{Y} = \mathbf{X}[\mathbf{v}_1, \ldots, \mathbf{v}_p] = \mathbf{XV}.$$

The ith column of $\mathbf{Y}$ is the ith **principal component**.

When we want to reduce the dimensionality to **only those defined by the first d principal components**, then we could right-multiply $\mathbf{X}$ by the $p \times d$ matrix $[\mathbf{v}_1, \ldots, \mathbf{v}_d]$ or equivalently simply select the first d columns of $\mathbf{Y}$.

There is a decomposition of a real rectangular $n \times p$ matrix $\mathbf{X}$ which is particularly handy called the **singular value decomposition**:

$$\mathbf{X} = \mathbf{U}\mathbf{D}_\sigma\mathbf{V}^\mathsf{T}$$

where $\mathbf{U}$ is an $n \times p$ matrix with the property that $\mathbf{U}^\mathsf{T}\mathbf{U} = \mathbf{I}_p$, $\mathbf{V}$ is a $p \times p$ matrix with $\mathbf{V}^\mathsf{T}\mathbf{V} = \mathbf{V}\mathbf{V}^\mathsf{T} = \mathbf{I}_p$ and $\mathbf{D}_\sigma = diag(\sigma_1, \ldots, \sigma_p)$ with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$.

- The scalars σ_i are called the **singular values** of $\mathbf{X}$, the columns of $\mathbf{U}$ the **left singular vectors** and the columns of $\mathbf{V}$ the **right singular vectors**.

University of Waterloo

The singular value decomposition connection to the eigen-decomposition

Letting $\mathbf{X} = \mathbf{U}\mathbf{D}_\sigma\mathbf{V}^\mathsf{T}$ be the singular value decomposition of $\mathbf{X}$, then

$$\begin{aligned}
\mathbf{X}^\mathsf{T}\mathbf{X} &= (\mathbf{U}\mathbf{D}_\sigma\mathbf{V}^\mathsf{T})^\mathsf{T}\mathbf{U}\mathbf{D}_\sigma\mathbf{V}^\mathsf{T} \\
&= \mathbf{V}\mathbf{D}_\sigma\mathbf{U}^\mathsf{T}\mathbf{U}\mathbf{D}_\sigma\mathbf{V}^\mathsf{T} \\
&= \mathbf{V}\mathbf{D}_\sigma\mathbf{D}_\sigma\mathbf{V}^\mathsf{T} \\
&= \mathbf{V}\mathbf{D}_\sigma^2\mathbf{V}^\mathsf{T}.
\end{aligned}$$

Which is just the eigen-decomposition of $\mathbf{X}^\mathsf{T}\mathbf{X}$ with $\mathbf{D}_\sigma^2 = \mathbf{D}_\lambda$. In particular note that $\sigma_i = \sqrt{\lambda_i}$; unlike λ, σ is on the same scale as the data values.

So, a quick way to get the principal components is to find the singular value decomposition of the (centred) matrix $\mathbf{X}$ and then

$$\mathbf{Y} = \mathbf{X}\mathbf{V} = \mathbf{U}\mathbf{D}_\sigma$$

The singular value decomposition (svd) in R

For example, in R

```
# Principal components
#   ... via a ...
#  Singular value decomposition (svd)
#              T
#   X = U D V
library(loon)
data <- oliveAcids
# Scale as well as centre since we are looking at scatterplots
data <- scale(data, center=TRUE, scale=TRUE)
svd_data <- svd(data)
u <- svd_data$u
d <- svd_data$d
v <- svd_data$v
```

Reducing dimensions

The singular value decomposition (svd) in R

```
#
# V has orthonormal columns
#
round(t(v) %*%  v, 10)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [1,]    1    0    0    0    0    0    0    0
## [2,]    0    1    0    0    0    0    0    0
## [3,]    0    0    1    0    0    0    0    0
## [4,]    0    0    0    1    0    0    0    0
## [5,]    0    0    0    0    1    0    0    0
## [6,]    0    0    0    0    0    1    0    0
## [7,]    0    0    0    0    0    0    1    0
## [8,]    0    0    0    0    0    0    0    1
```

```
# and, being square, so are its rows
round(v %*%  t(v), 10)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [1,]    1    0    0    0    0    0    0    0
## [2,]    0    1    0    0    0    0    0    0
## [3,]    0    0    1    0    0    0    0    0
## [4,]    0    0    0    1    0    0    0    0
## [5,]    0    0    0    0    1    0    0    0
## [6,]    0    0    0    0    0    1    0    0
## [7,]    0    0    0    0    0    0    1    0
## [8,]    0    0    0    0    0    0    0    1
```

University of Waterloo

The singular value decomposition (svd) in R

```
#
# Similarly u has orthonormal columns
# (but not rows)

round(t(u) %*% u, 10)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [1,]    1    0    0    0    0    0    0    0
## [2,]    0    1    0    0    0    0    0    0
## [3,]    0    0    1    0    0    0    0    0
## [4,]    0    0    0    1    0    0    0    0
## [5,]    0    0    0    0    1    0    0    0
## [6,]    0    0    0    0    0    1    0    0
## [7,]    0    0    0    0    0    0    1    0
## [8,]    0    0    0    0    0    0    0    1
```

```
# compare: u %*% t(u)  ... n by n non-orthogonal
#
```

Reducing dimensions

The singular value decomposition (svd) in R

```
# d contains the singular values of X
# Here they are in a diagonal matrix
round(diag(d), 0)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [1,]   46    0    0    0    0    0    0    0
## [2,]    0   32    0    0    0    0    0    0
## [3,]    0    0   24    0    0    0    0    0
## [4,]    0    0    0   21    0    0    0    0
## [5,]    0    0    0    0   14    0    0    0
## [6,]    0    0    0    0    0   12    0    0
## [7,]    0    0    0    0    0    0    8    0
## [8,]    0    0    0    0    0    0    0    1
```

Clearly, some of these are much larger than others. Perhaps we do not need all of these dimensions to explore the data?

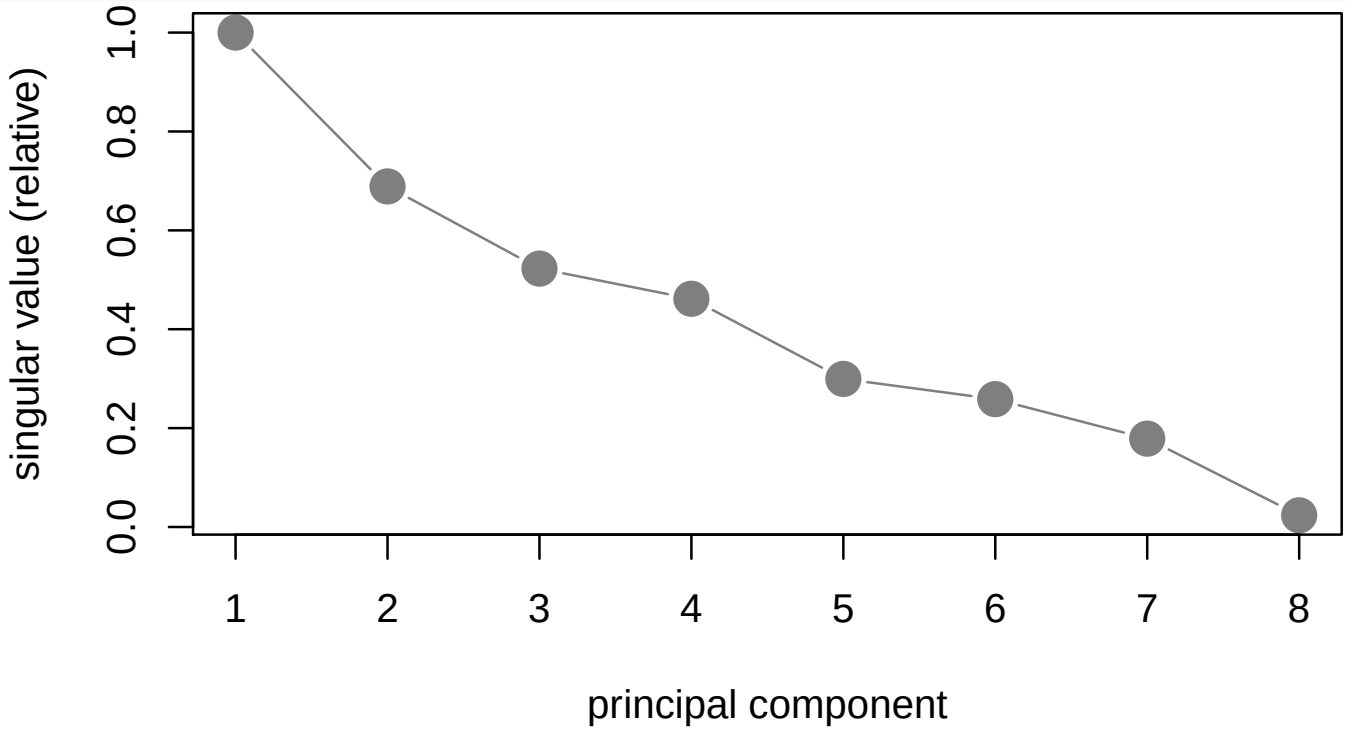
To get a sense of the relative size we look at a so-called **scree plot**.

University of Waterloo

The effective dimensionality of the data

Scree plots: determining the effective dimension of the subspace.

```
# The following is a "Scree plot"
plot(d/max(d), # contributions to the effective dimension
     type="b", xlab="principal component", ylab="singular value (relative)",
     col="grey50", pch=16, cex=2)
```



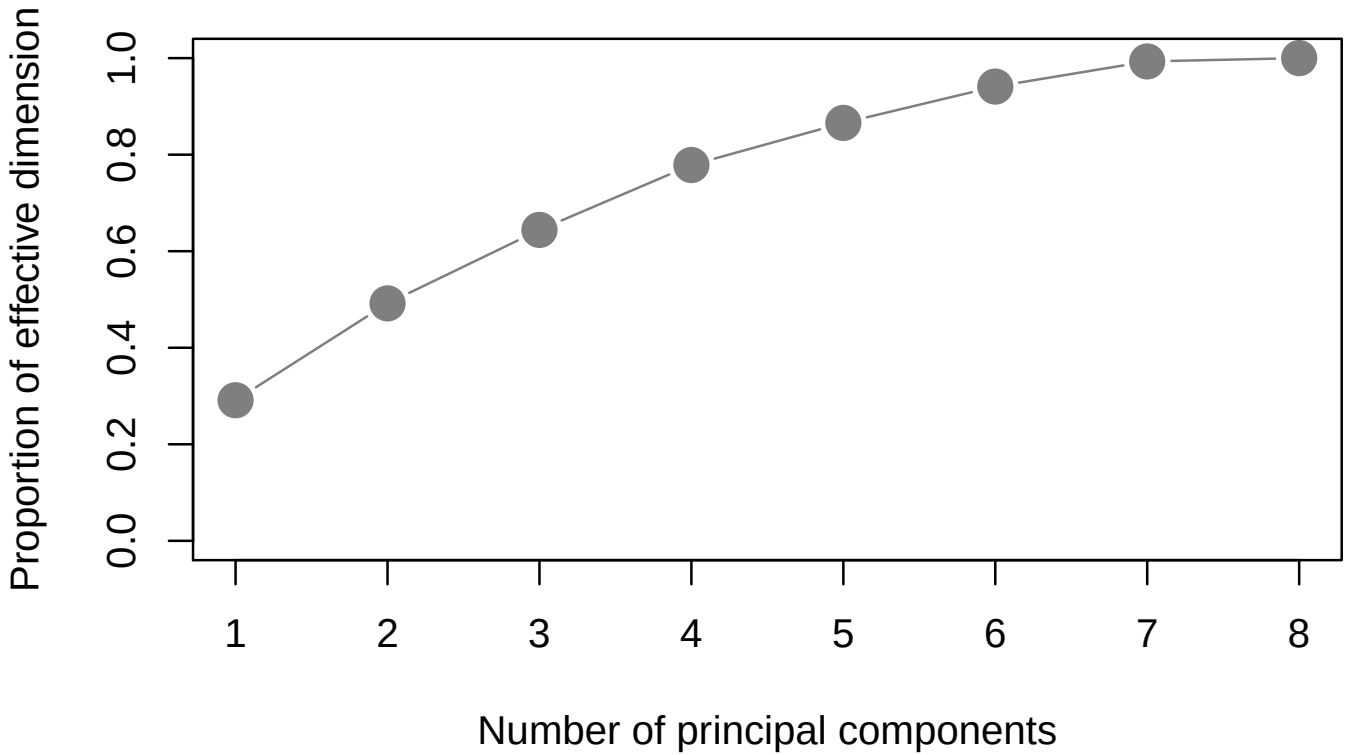
Perhaps 4 dimensions are enough? 5?
Ideally there would be a large noticeable drop at some point.

Reducing dimensions

Now, $\frac{\sum \sigma_i}{\sigma_{max}}$ is a measure of the effective (fractional) dimensionality of the data. This suggests that we might consider looking at the ratio $\sum_{i=1}^{d} \sigma_i / \sum_{j=1}^{p} \sigma_j$ as a measure of the proportion of the dimensionality retained by the first d components.

This suggests an alternative

```
# The following is
plot(cumsum(d)/sum(d), # proportion of effective dimension
     type="b", col="grey50", xlab="Number of principal components",
     ylab="Proportion of effective dimension", ylim=0:1, pch=16, cex=2)
```



University of Waterloo

Reducing dimensions - a more complex example

In the `frey` dataset to be found in the `loon.data` package, we have 1,965 images of Brendan Frey's face. Here are 4 examples:




University of Waterloo

Reducing dimensions - a more complex example

- Each pixel has a grey value in [0,1].
- Each image is a 28×20 pixel image.

So each image is a point in 560 dimensions.

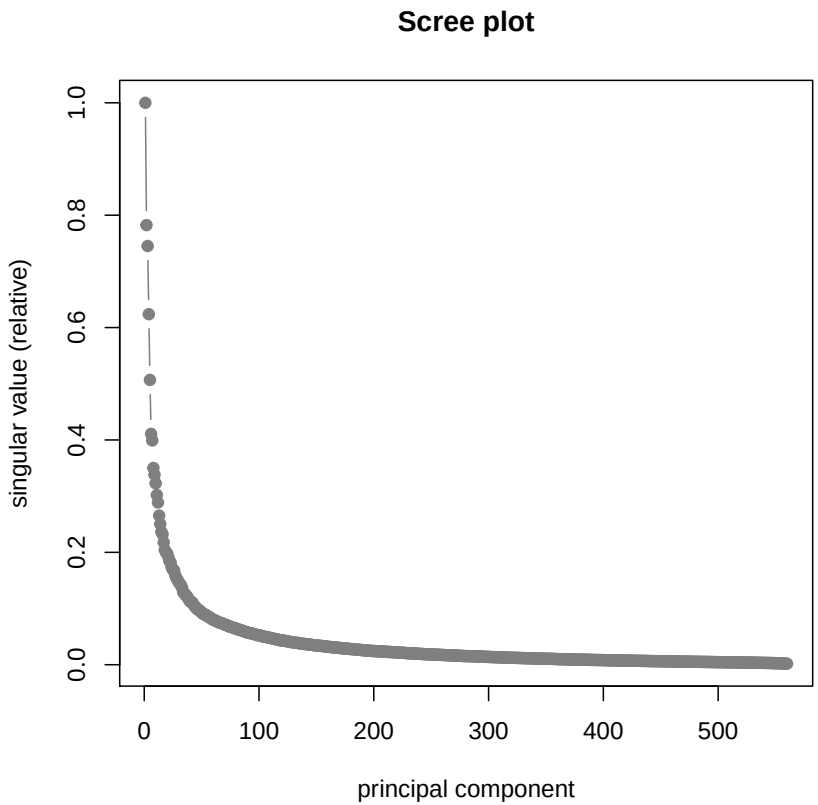
University of Waterloo

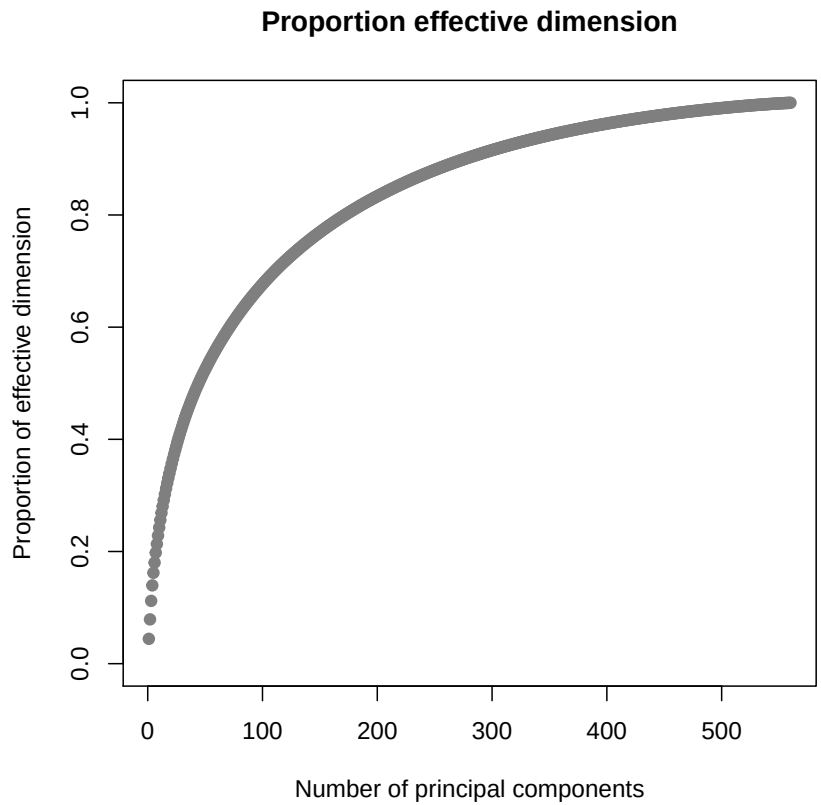
Reducing dimensions - a more complex example

```
# Principal components
#   ... via a ...
#  Singular value decomposition (svd)
#              T
#   X = U D V
library(loon)
library(loon.data)
data(frey)
data <- t(frey)
# Scale as well as centre since we are looking at scatterplots
data <- scale(data, center=TRUE, scale=FALSE)
svd_data <- svd(data)
u <- svd_data$u
d <- svd_data$d
v <- svd_data$v
```

University of Waterloo

Reducing dimensions - a more complex example





Doesn't look like we need all 560 dimensions, but how many should we choose? 100? Let's look at 20.

University of Waterloo

Reducing dimensions - a more complex example

```
# Here are the images
frey.imgs <- l_image_import_array(frey, 28,20,
                                  img_in_row = FALSE, rotate = 90)
l_imageviewer(frey.imgs)

reduced <- data %*% v[, 1:20]

nav1 <- l_navgraph(reduced, linkingGroup="frey")
gl1 <- l_glyph_add_image(nav1$plot, images=frey.imgs,
                         label="frey faces")

scags2d <- scagnostics2d(reduced)
nav2 <- l_ng_plots(measures=scags2d, linkingGroup="frey")
gl2 <- l_glyph_add_image(nav2$plot, images=frey.imgs,
                         label="frey faces")
```

University of Waterloo

Reducing dimensions - a more complex example

Some remarks:

- the dimension reduction made *no use* of the fact that the data:
  - were faces (i.e. no facial feature extracted)
  - images (i.e. spatial position not really used)
- principal components does not care which position in the 560-dimensional vector that each pixel location occupies
  - i.e. could have randomly assigned the image pixel array locations to the locations in the vector
- still got some useful positioning even in only the first two principal components
- there is no reason to reduce the dimensionality to *only* 2 or 3 dimensions
  - can reduce to 20 or even more dimensions if that is what the scree plot suggests
  - can reduce to any number provided that the determination of interesting projections is computationally feasible
- principal components is looking for low-dimensional **linear** subspaces (or linear **manifolds**)

University of Waterloo