CAP 5625: Computational Foundations for Artificial Intelligence

Unsupervised learning

Switching gears to unsupervised learning

Throughout the course our modeling has focused almost exclusively on supervised learning in which we have training data on a set of features with an associated response.

Unsupervised learning approaches do not have information about a response, and instead try to identify common (or differing) characteristics of the input data.

However, we will attempt to motivate the use of unsupervised learning to improve supervised learning tasks.

Our final statistical (machine) learning methods

In this lecture, we will introduce unsupervised learning methods for performing dimensionality reduction and clustering.

Notation

Input X: X is often multidimensional. Each dimension of X is referred to as a feature, predictor, or independent variable

Output Y: The response, or dependent variable

Categorization

Supervised learning vs. unsupervised learning

Is Y available in the training data?

Regression vs. classification

Is Y quantitative or qualitative?

Unsupervised approaches considered here

We will consider a set of commonly-used unsupervised learning approaches.

We will begin with **principal components analysis**, particularly with an application for data visualization and regression.

We will then move to **prototype clustering**.

Finally, we will introduce **hierarchical clustering**.

Dealing with correlated features

In the last couple lectures we have focused on regularization techniques that can deal with correlated features in different ways.

For example, ridge regression shrinks correlated features toward 0 together, while maintaining the presence of all p features in the fit model.

Lasso, on the other hand, shrinks correlated features toward 0 together, but after a certain threshold, will automatically set one of them to 0 for being redundant or uninformative, thereby performing simultaneous feature selection.

Here lasso may remove some features altogether, thereby ensuring the predictions are not based on those feature at all.

Sometimes it is useful to retain majority of features

Lasso is nice as it reduces the model to fewer features relative to ridge regression.

However, sometimes it may be useful to base prediction on all features, yet still ultimately having few parameters in the fit model.

For example, as we increase the number of features, the dimensionality of the parameter space increases, making the optimization problem difficult.

Yet it may be helpful to still use information on all features but reduce the ultimate dimension of the fit model (*i*.*e*., ultimately reduce the number of parameters needed to be estimated).

Sometimes it is useful to retain majority of features

Moreover, suppose that the true variable that is related to the response is not included as a feature, but many features measured are correlated with this variable.

For example, suppose a response Y is related to a true variable X_{temp} that measures temperature, yet we had measurements on 5 other features

X_1 = Number of snow days in schools

X_2 = Number of burst water pipes

X_3 = Number of individuals suffering from heat stroke

X_4 = Number of skidding accidents

X_5 = Amount of money spent on snow plowing

Sometimes it is useful to retain majority of features

X_1 = Number of snow days in schools

X_2 = Number of burst water pipes

X_3 = Number of individuals suffering from heat stroke

X_4 = Number of skidding accidents

X_5 = Amount of money spent on snow plowing

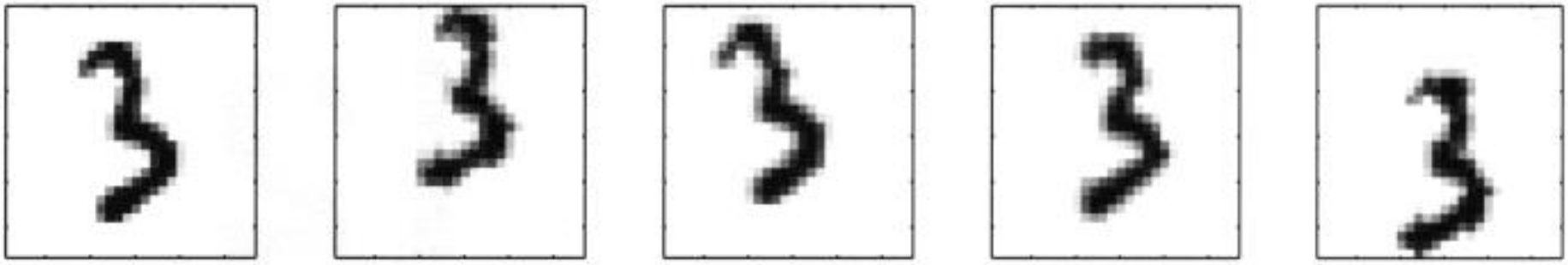
All 5 features are related to the underlying temperature, which is the true feature that we did not measure.

Therefore, it may be better if we had some feature X_{temp} that is a function of measured features $X_1, X_2, \ldots, X_5$. That is

$$X_{\text{temp}} = f(X_1, X_2, X_3, X_4, X_5)$$

Sometimes it is useful to retain majority of features

As another example, consider an observation of a handwritten digit.



Here, the number is displayed based on gray-scale pixel intensities in a 28×28 pixel grid, leading to $p = 28 \times 28 = 784$ features, with X_j taking values between 0 (white) and 1 (black) for $j = 1,2, \ldots, p$.

Individual pixels are not necessarily capturing the important information about the number, and it is really the path of the pen stroke that is important (in which it is expected that some pixels will be covered when other pixels have also been covered).

Bishop (2006) *Pattern Recognition and Machine Learning*. Springer.

Sometimes it is useful to retain majority of features

We are therefore really looking for other features that are functions of all the individual pixels that may represent a feature more like a pen stroke.

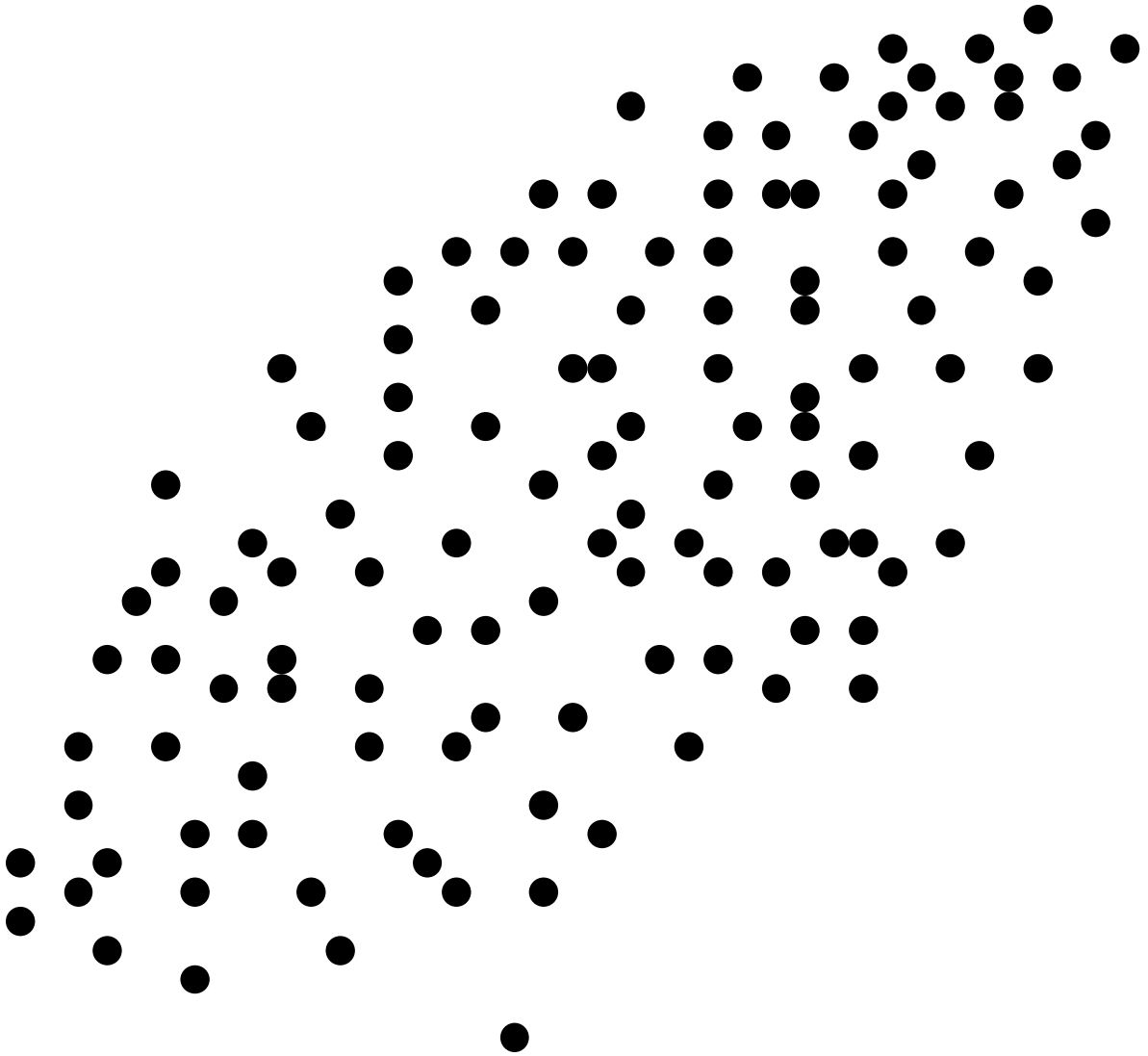
That is, it may be better if we had some feature X_{path} that is a function of measured features $X_1, X_2, \ldots, X_p$. That is

$$X_{\text{path}} = f(X_1, X_2, \ldots, X_p)$$

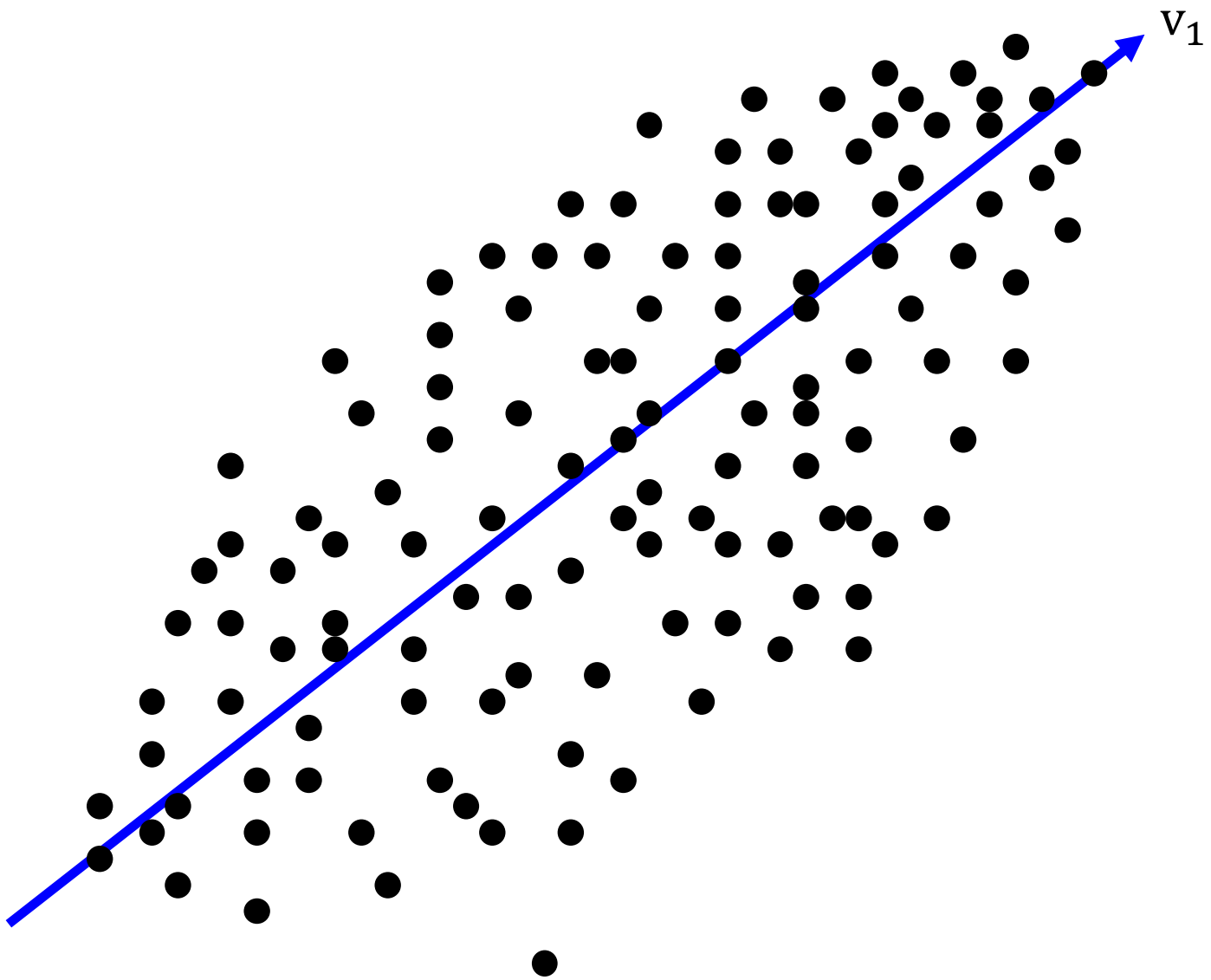
There exist methods that can create derived features from original features, and one class of techniques is known as dimensionality reduction methods, in which we project data points in a p-dimensional space into a much smaller space with dimension $M \ll p$.

Specifically, we will cover **principal components analysis** (**PCA**).

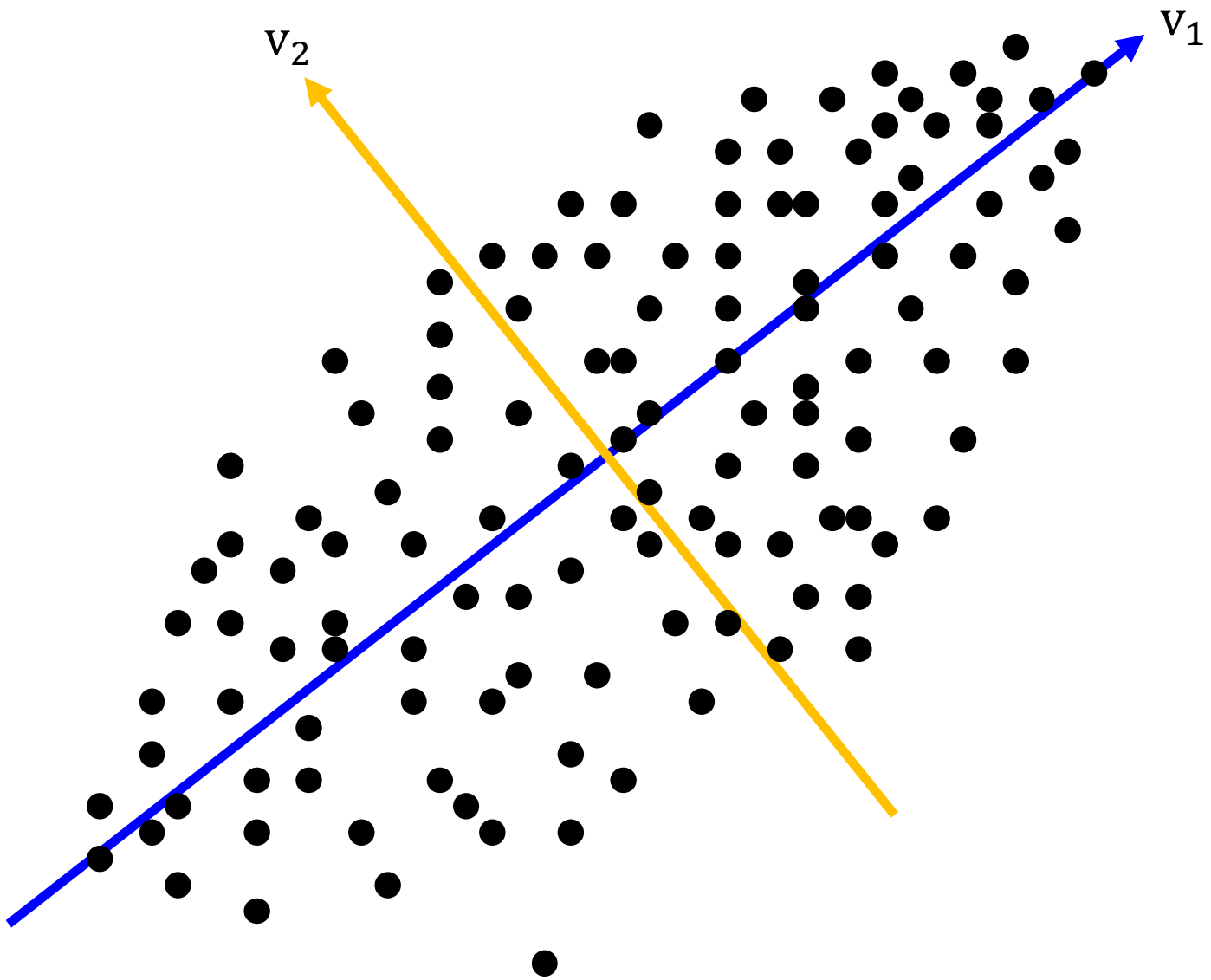
Visual depiction of projection of data x onto v_1 and v_2



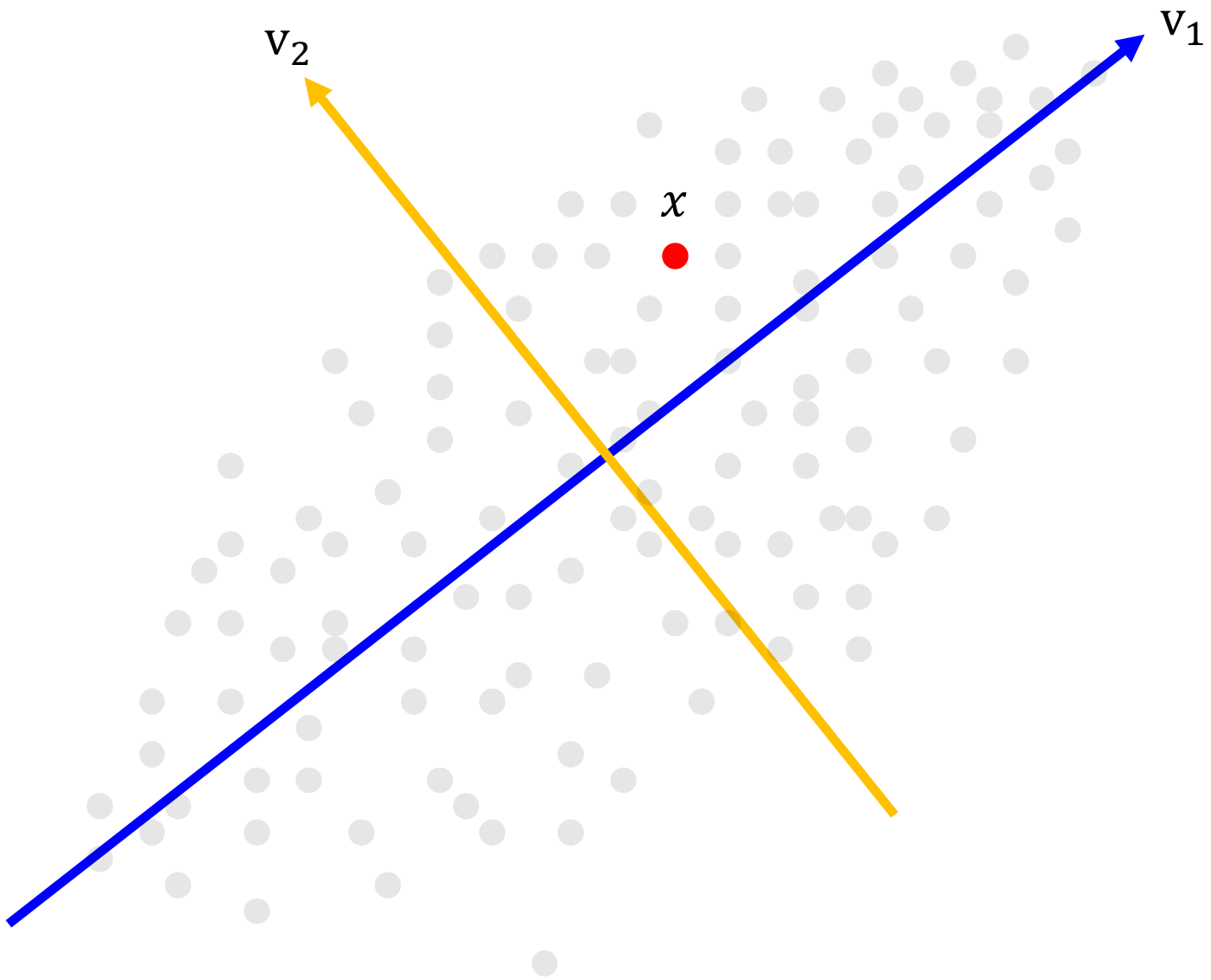
Visual depiction of projection of data x onto v_1 and v_2



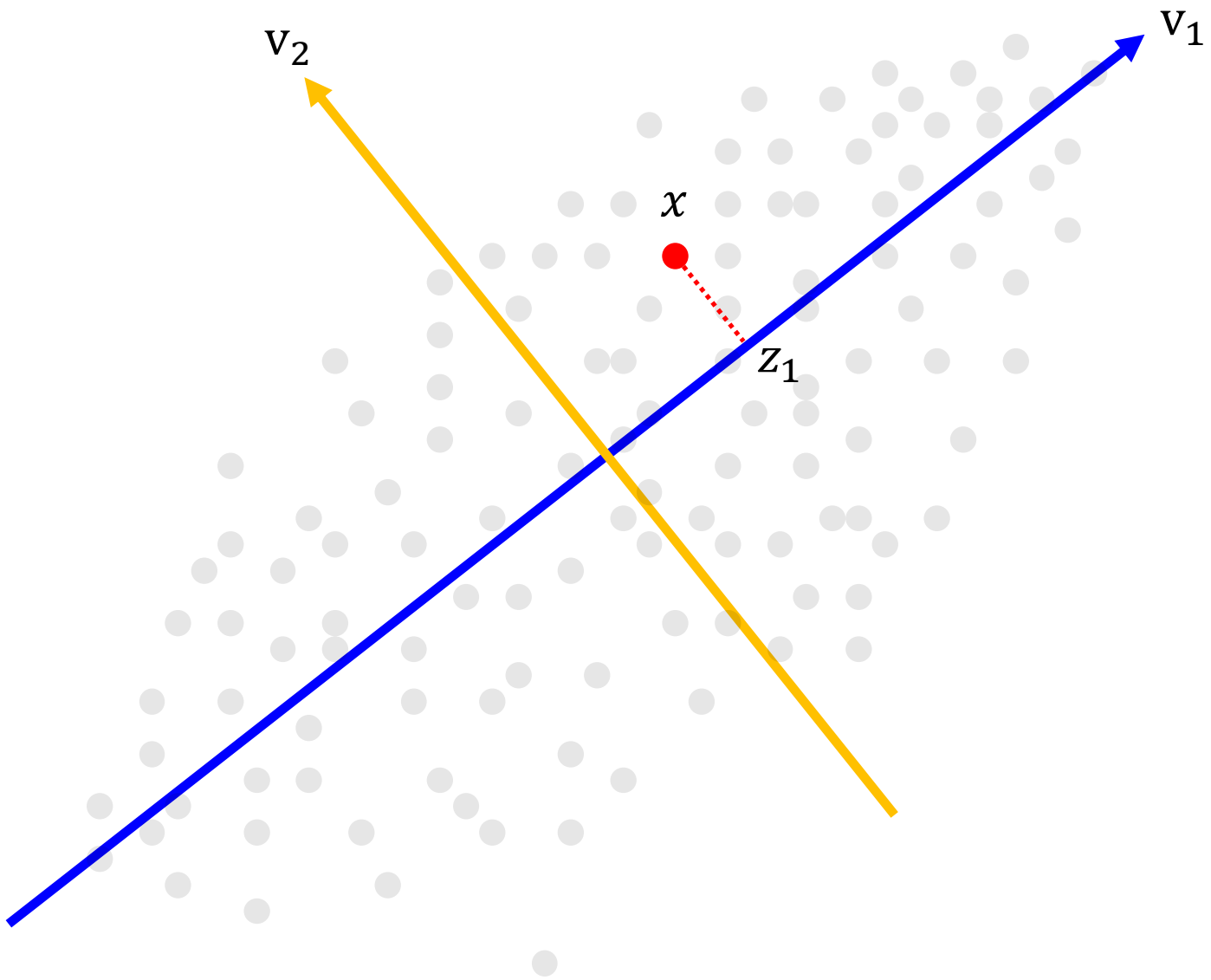
Visual depiction of projection of data x onto v_1 and v_2



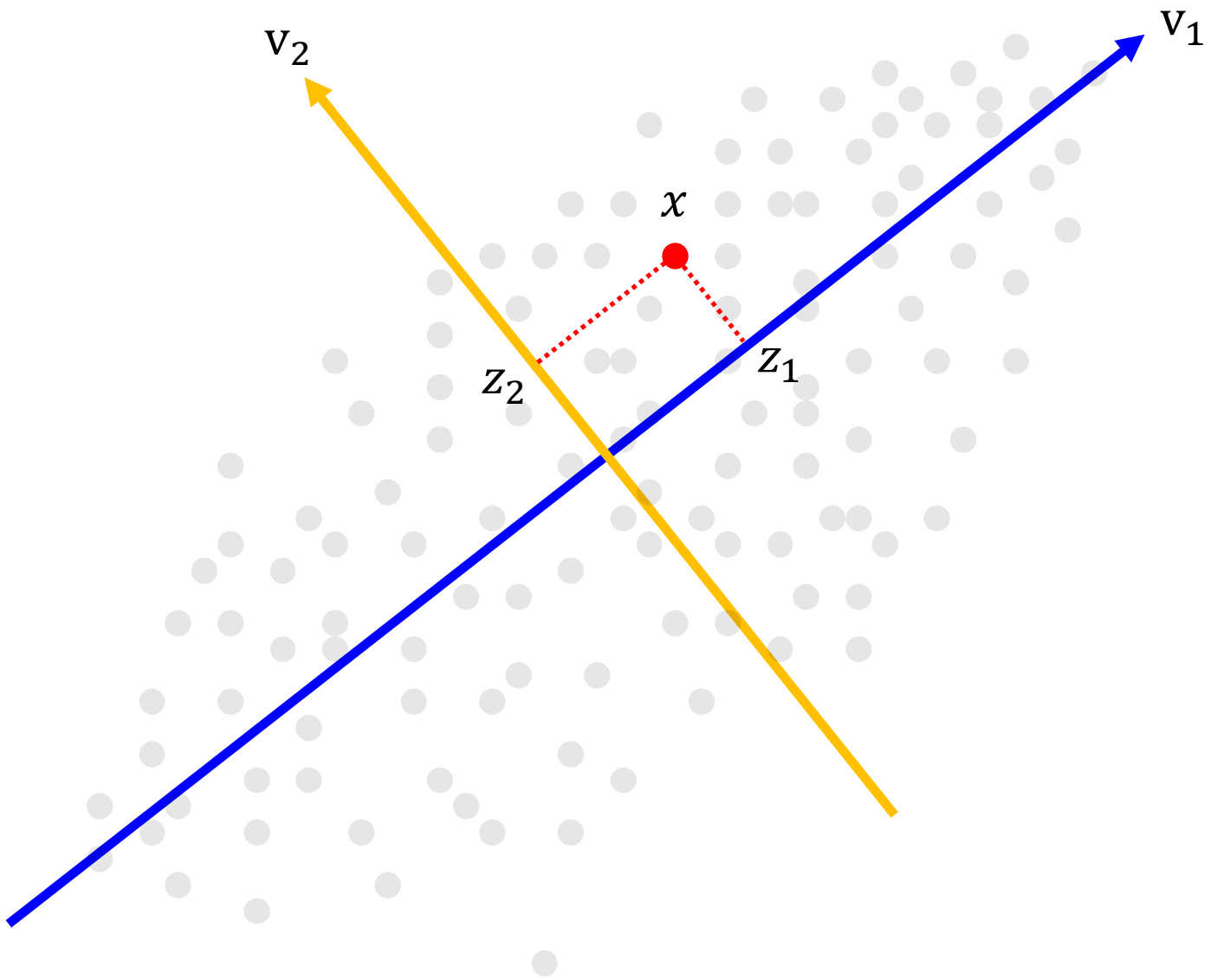
Visual depiction of projection of data x onto v_1 and v_2



Visual depiction of projection of data x onto v_1 and v_2



Visual depiction of projection of data x onto v_1 and v_2



Principal components analysis (dimension reduction)

Find p-dimensional principal components $\mathrm{v}_1, \mathrm{v}_2, \ldots, \mathrm{v}_p$ that are orthogonal (perpendicular) with constraint that they are of length one.

Principal component m takes the values

$$\mathrm{v}_m = [\mathrm{v}_{1m}, \mathrm{v}_{2m}, \ldots, \mathrm{v}_{pm}]$$

across its p dimensions.

Given these vectors, change coordinates of data points by projecting them onto principal component m, such that the datapoint

$$x = [x_1, x_2, \ldots, x_p]$$

has projected value

$$z_m = x_1\mathrm{v}_{1m} + x_2\mathrm{v}_{2m} + \cdots + x_p\mathrm{v}_{pm}$$

in principal component m, $m = 1, 2, \ldots, p$.

Description in vector notation

Find vectors (principal components) $\mathrm{v}_1, \mathrm{v}_2, \ldots, \mathrm{v}_p \in \mathbb{R}^p$ that are orthogonal (perpendicular) with constraint that they are of unit length.

That is, $\mathrm{v}_j^T \mathrm{v}_k = 0$ for $j \neq k$, and $\|\mathrm{v}_j\|_2^2 = \|\mathrm{v}_j\|_2 = 1$ for $j = 1,2, \ldots, p$.

Given these vectors, change coordinates of data points by projecting them onto principal component m, such that the datapoint

$$\mathrm{z}_m = \mathbf{X}\mathrm{v}_m$$

such that the datapoint $i, i = 1,2, \ldots, N$, has projected value

$$z_{im} = x_i^T \mathrm{v}_m = \sum_{j=1}^{p} x_{ij} v_{jm}$$

in principal component m.

Principal components analysis

Specifically, we want to choose the vectors $\mathrm{v}_1, \mathrm{v}_2, \ldots, \mathrm{v}_p$ such that the differences between the original points $x_i = [x_{i1}, x_{i2}, \ldots, x_{ip}]$ and the projected points $z_i = [z_{i1}, z_{i2}, \ldots, z_{ip}]$ are minimized.

Moreover, we want v_1 to lie in the direction of greatest variation in the dataset, with v_2 orthogonal and explaining the second greatest amount of variation in the data.

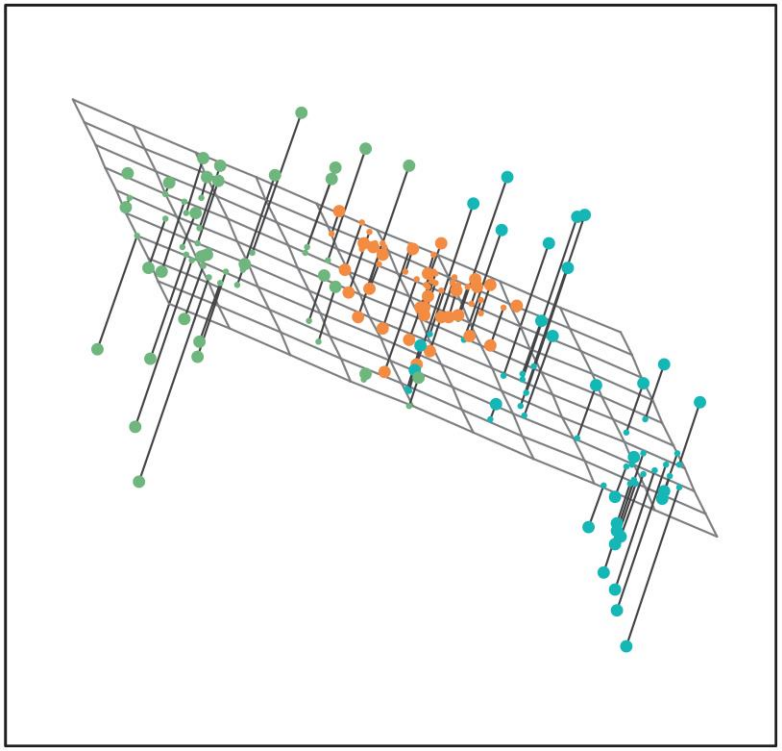
More generally, we want v_j to be the vector of unit length that describes the greatest amount of remaining variability in the dataset, conditioning that it is orthogonal to vectors $\mathrm{v}_1, \mathrm{v}_2, \ldots, \mathrm{v}_{j-1}$.

We will assume the data matrix is standardized (centered with features normalized by their standard deviation).

Illustration of PCA

$N = 90$ observations displayed in $p = 3$ features (dimensions), together with a plane spanned by the first 2 PCs that best fits the data.

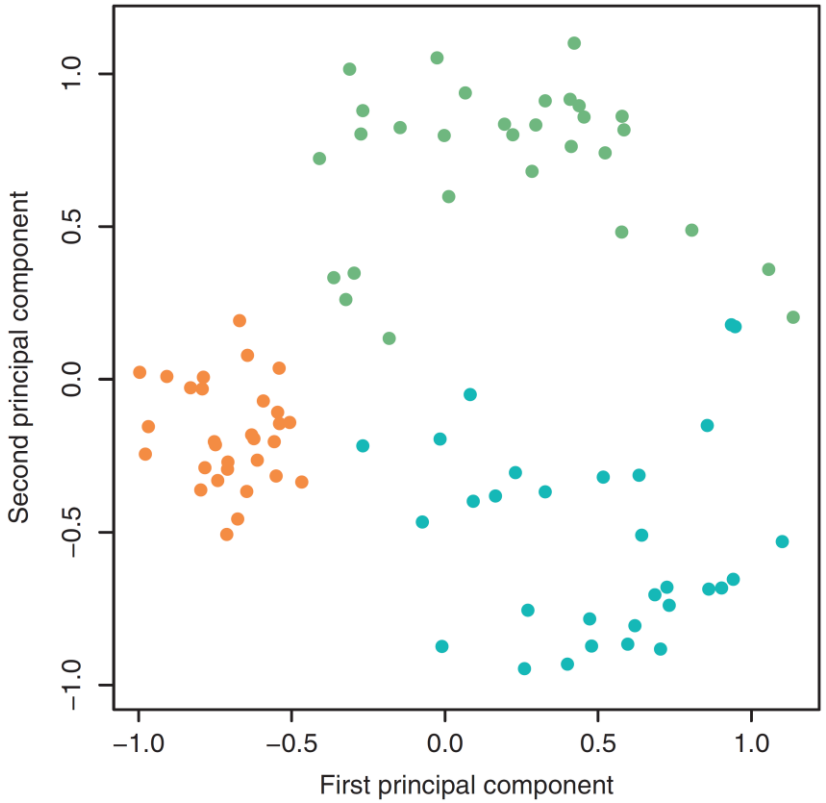
This plane minimizes sum of squared differences from the plane.



James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Illustration of PCA

Observations projected to the first 2 PCs, demonstrating how the plane spanned by the first 2 PCs preserves as much variability as possible, separating observations that seemingly derive from different groups (colors).

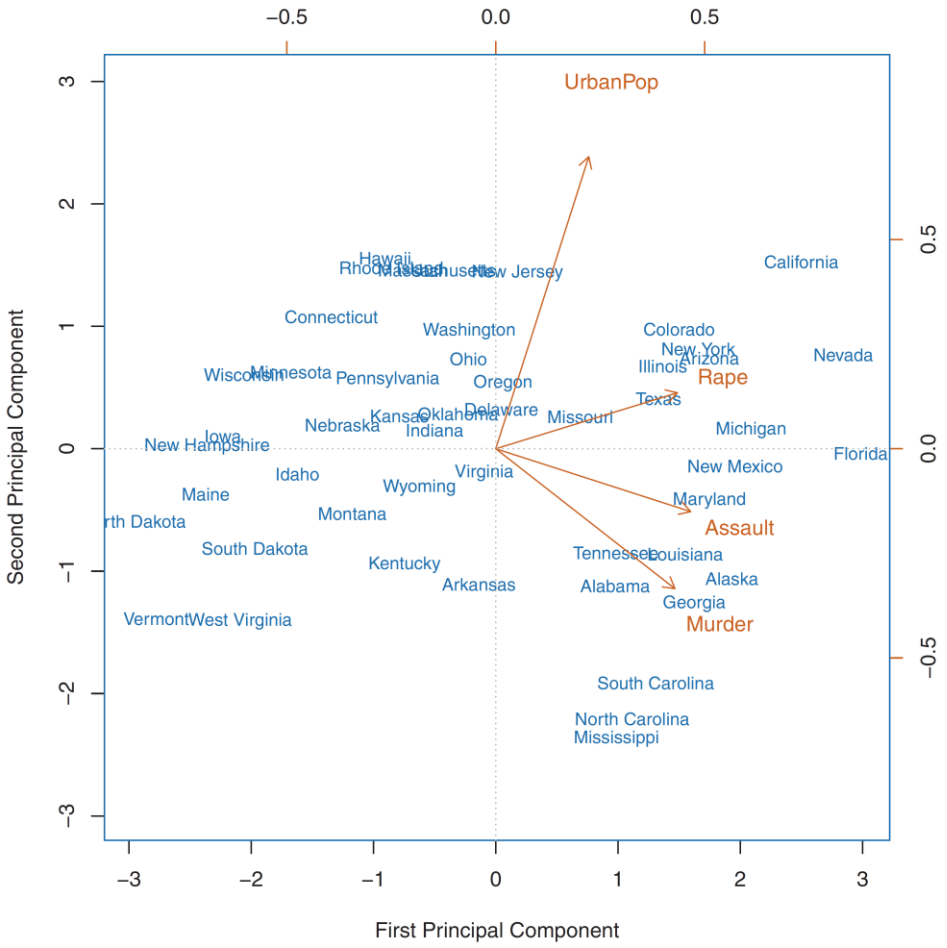


James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Illustration of a PC biplot

PC **biplot** depicts projected observations along the PCs defining the x- and y-axes, as well as the loadings of the features on these PCs.

	PC1	PC2
Murder	0.5358995	−0.4181809
Assault	0.5831836	−0.1879856
UrbanPop	0.2781909	0.8728062
Rape	0.5434321	0.1673186

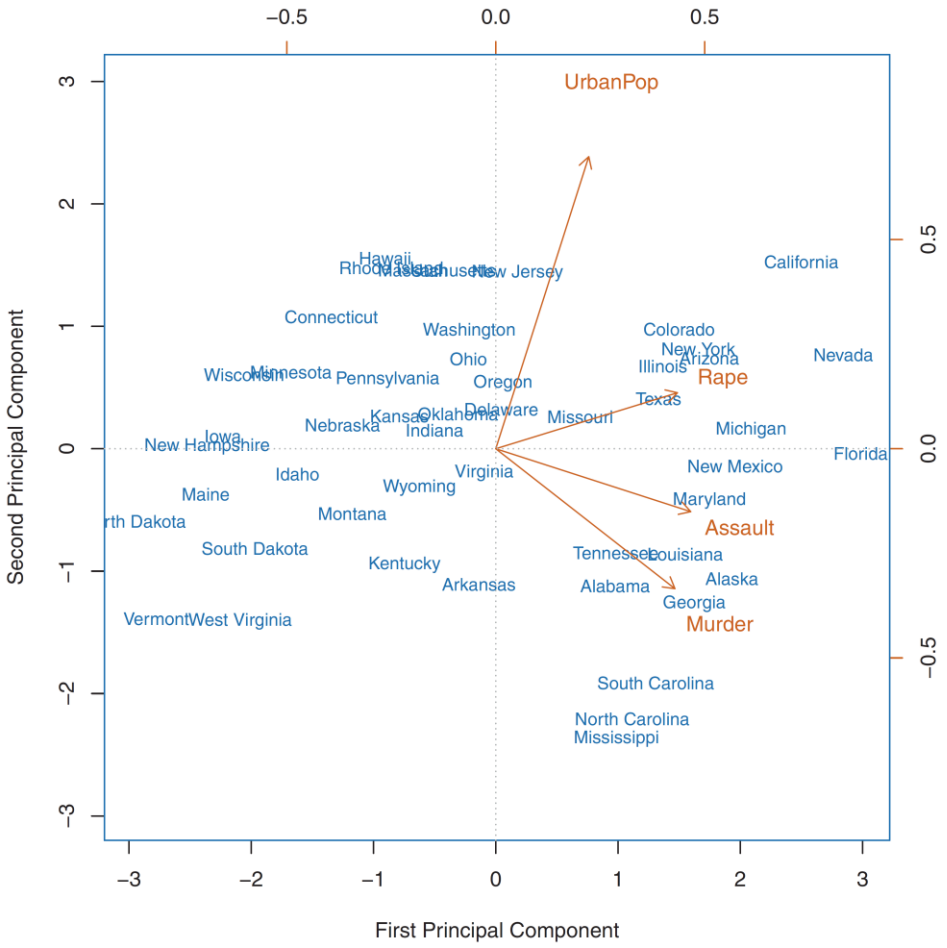


James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Illustration of a PC biplot

Data on $N = 50$ United States arrest records (each state) based on $p = 4$ features describing numbers of arrests for assaults, murders, and rapes, and percentage of population living in urban areas.

	PC1	PC2
Murder	0.5358995	−0.4181809
Assault	0.5831836	−0.1879856
UrbanPop	0.2781909	0.8728062
Rape	0.5434321	0.1673186

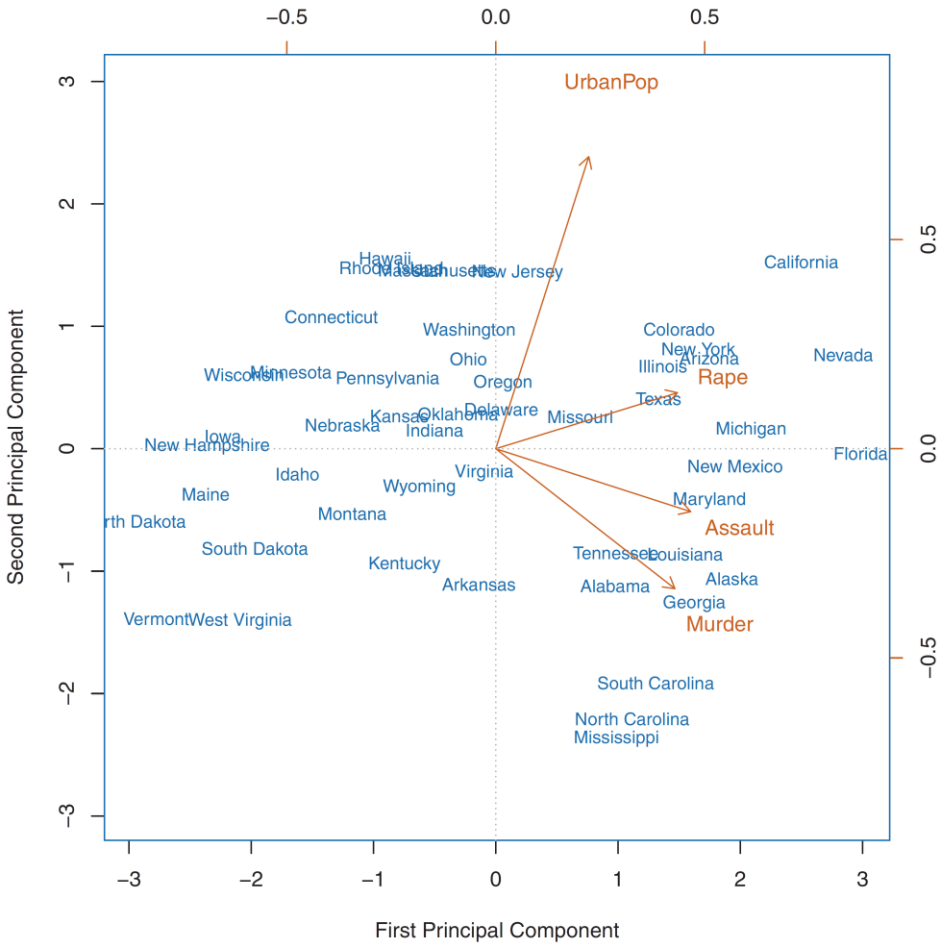


James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Illustration of a PC biplot

First loading places approximately equal weight on assault, murder, and rape, with substantially less weight on percentage of population in urban areas, indicating PC 1 corresponds roughly to measure of overall rates of serious crime.

	PC1	PC2
Murder	0.5358995	−0.4181809
Assault	0.5831836	−0.1879856
UrbanPop	0.2781909	0.8728062
Rape	0.5434321	0.1673186

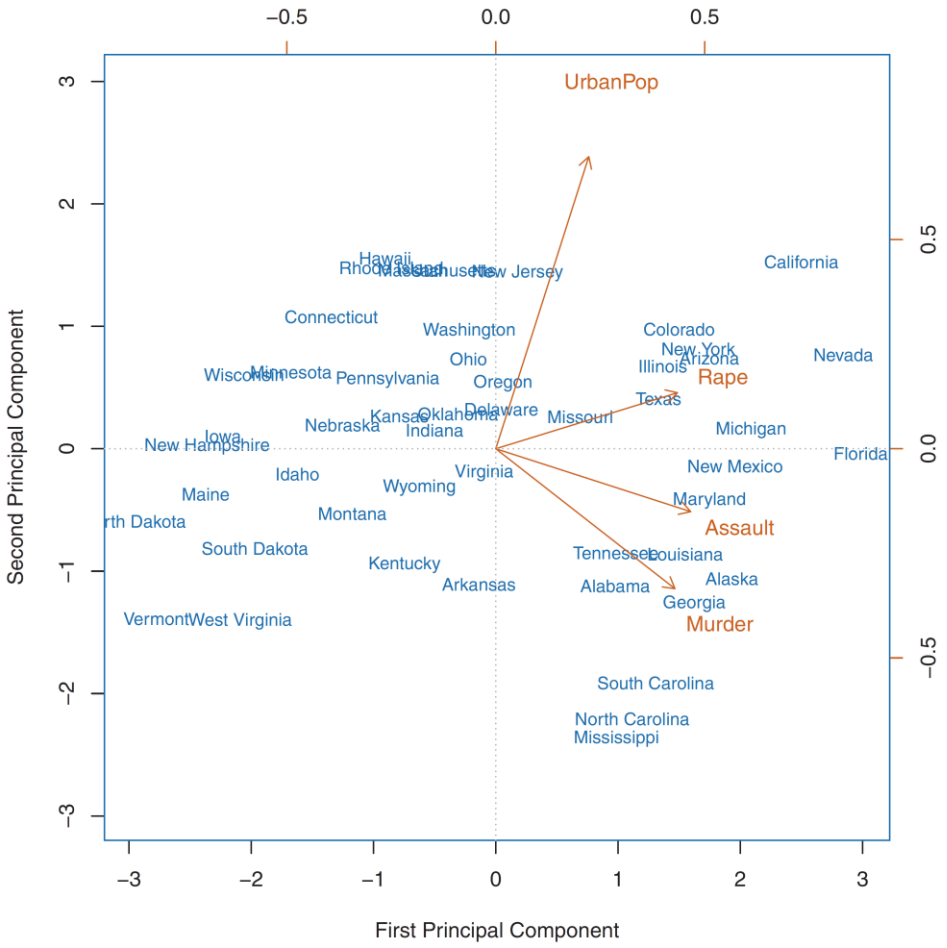


James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Illustration of a PC biplot

Second loading places most weight on percentage of population in urban areas, indicating PC 2 corresponds roughly to level of urbanization of the state.

	PC1	PC2
Murder	0.5358995	−0.4181809
Assault	0.5831836	−0.1879856
UrbanPop	0.2781909	0.8728062
Rape	0.5434321	0.1673186

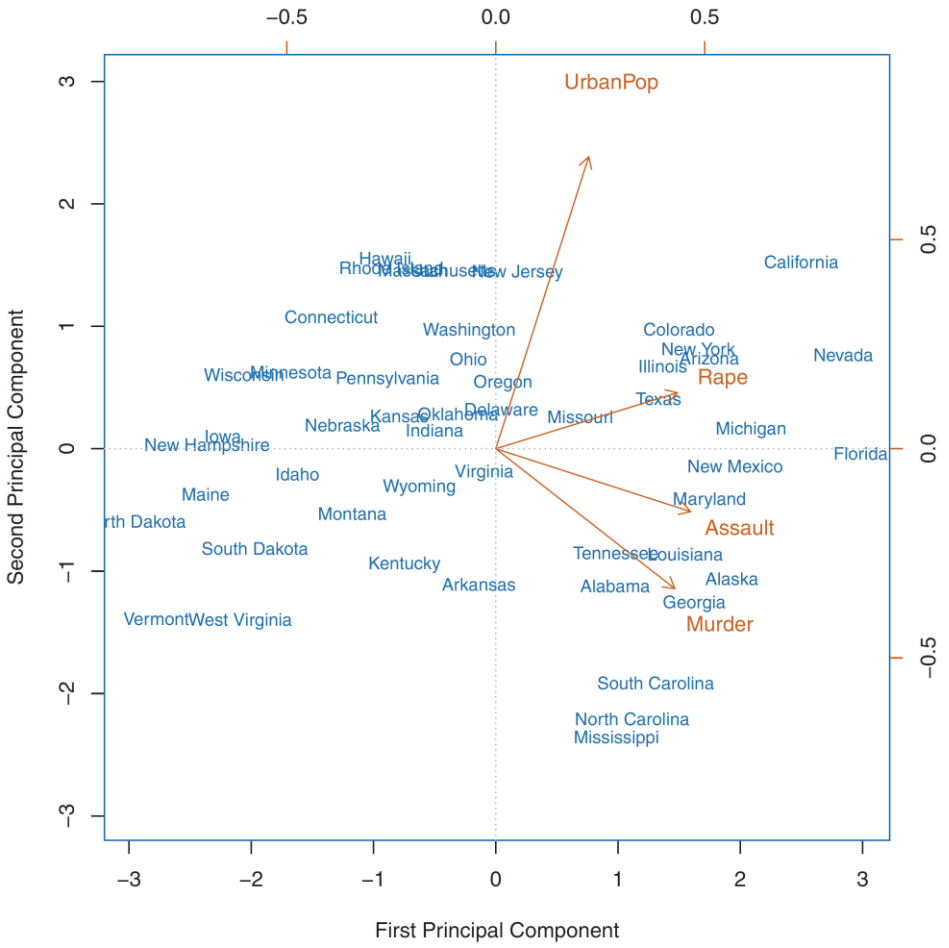


James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Illustration of a PC biplot

The crime-related features (murder, assault, rape) are located near each other, with the percentage of urban population distant from them, indicating crime-related variables are correlated with each other (states with high murder rates tend to have high assault and rape rates).

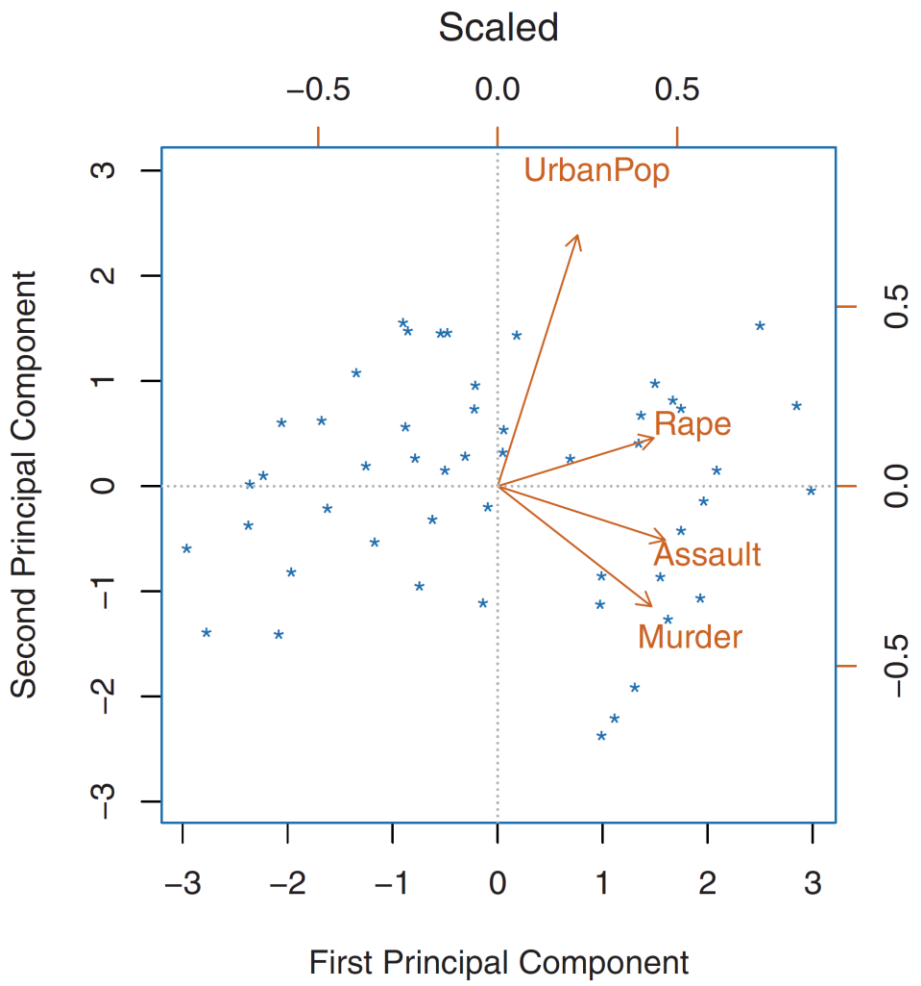
	PC1	PC2
Murder	0.5358995	−0.4181809
Assault	0.5831836	−0.1879856
UrbanPop	0.2781909	0.8728062
Rape	0.5434321	0.1673186

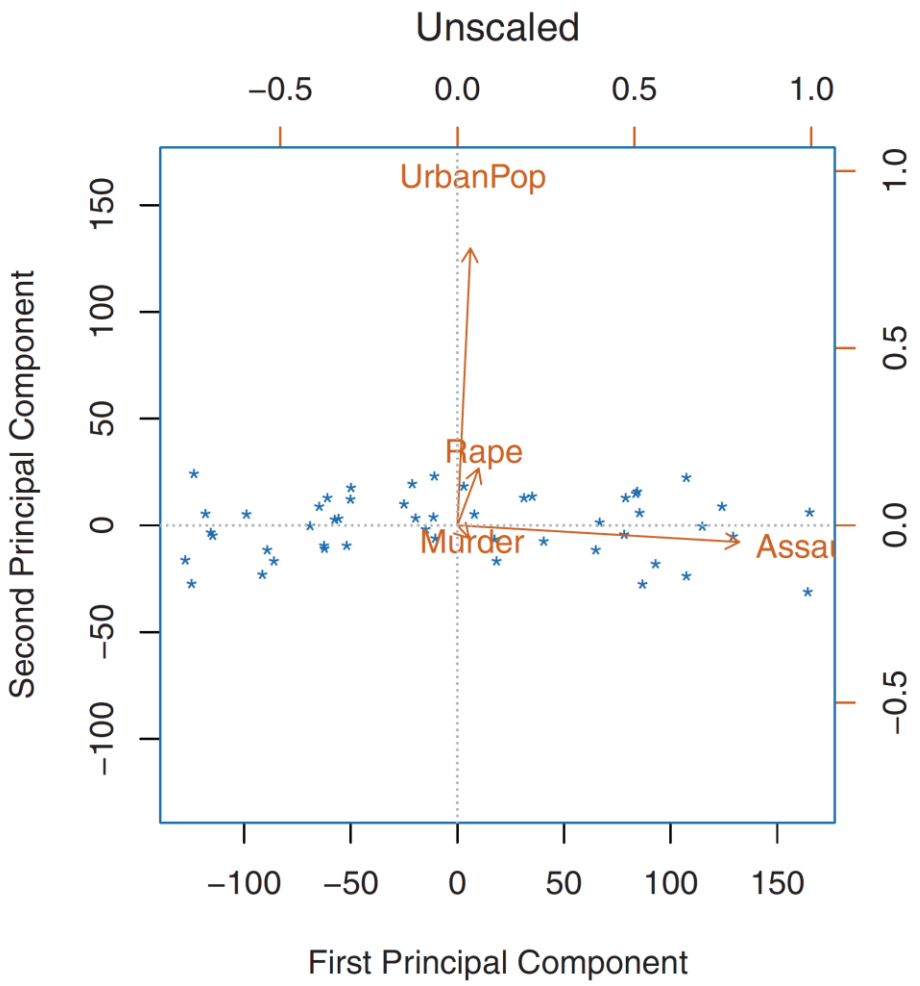


James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

It can be important to standardize features

Variance of assault is 6945.16, whereas the variances of murder, rape, and % urban are 18.97, 87.73, and 209.5, respectively.





James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

The sample covariance matrix

Given the data $\mathbf{X}$ are standardized, it is easy to compute the sample covariance (correlation) matrix $\mathbf{S} \in \mathbb{R}^{p \times p}$ across the p features.

The covariance is a measure of how much a feature varies from the mean with respect to another feature.

If one feature increases in value as another feature increases, then we would say these features have positive covariance.

In contrast, if a feature decreases in value as another feature decreases, then we would say these features have negative covariance.

The features are independent if they have 0 covariance.

Computing the covariance matrix

The element in the jth row and kth column of the sample covariance matrix is

$$\mathbf{S}_{jk} = Cov(\mathrm{x}_j, \mathrm{x}_k)$$

$$= \frac{1}{N}\sum_{i=1}^{N} x_{ij}x_{ik} - \left(\frac{1}{N}\sum_{i=1}^{N} x_{ij}\right)\left(\frac{1}{N}\sum_{i=1}^{N} x_{ik}\right)$$

$$= \frac{1}{N}\sum_{i=1}^{N} x_{ij}x_{ik}$$

$$= \frac{1}{N}\mathrm{x}_j^T \mathrm{x}_k$$

Computing the sample covariance matrix

Because $\mathbf{S}_{jk} = \frac{1}{N}\mathrm{x}_j^T\mathrm{x}_k$, we have that

$$\mathbf{S} = \frac{1}{N}\begin{bmatrix} \mathrm{x}_1^T\mathrm{x}_1 & \mathrm{x}_1^T\mathrm{x}_2 & \cdots & \mathrm{x}_1^T\mathrm{x}_p \\ \mathrm{x}_2^T\mathrm{x}_1 & \mathrm{x}_2^T\mathrm{x}_2 & \cdots & \mathrm{x}_2^T\mathrm{x}_p \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{x}_p^T\mathrm{x}_1 & \mathrm{x}_p^T\mathrm{x}_2 & \cdots & \mathrm{x}_p^T\mathrm{x}_p \end{bmatrix}$$

$$= \frac{1}{N}\begin{bmatrix} \mathrm{x}_1^T \\ \mathrm{x}_2^T \\ \vdots \\ \mathrm{x}_p^T \end{bmatrix}\begin{bmatrix} x_1 & \mathrm{x}_2 & \cdots & \mathrm{x}_p \end{bmatrix}$$

$$= \frac{1}{N}\mathbf{X}^T\mathbf{X}$$

Consider multiplying a vector v by the sample covariance matrix $\mathbf{S}$.

Example with 2 features

Consider the sample covariance matrix $\mathbf{S}$ and vector v

$$\mathbf{S} = \begin{bmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{bmatrix} \qquad v = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

Multiplying v by $\mathbf{S}$ one to five times, we respectively get

$$\mathbf{S}v = \begin{bmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1.2 \\ -0.2 \end{bmatrix}$$

$$\mathbf{S}(\mathbf{S}v) = \begin{bmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{bmatrix} \begin{bmatrix} -1.2 \\ -0.2 \end{bmatrix} = \begin{bmatrix} -2.5 \\ -1.0 \end{bmatrix}$$

$$\mathbf{S}(\mathbf{S}(\mathbf{S}v)) = \begin{bmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{bmatrix} \begin{bmatrix} -2.5 \\ -1.0 \end{bmatrix} = \begin{bmatrix} -6.0 \\ -2.7 \end{bmatrix}$$

$$\mathbf{S}\left(\mathbf{S}(\mathbf{S}(\mathbf{S}v))\right) = \begin{bmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{bmatrix} \begin{bmatrix} -6.0 \\ -2.7 \end{bmatrix} = \begin{bmatrix} -14.1 \\ -6.4 \end{bmatrix}$$

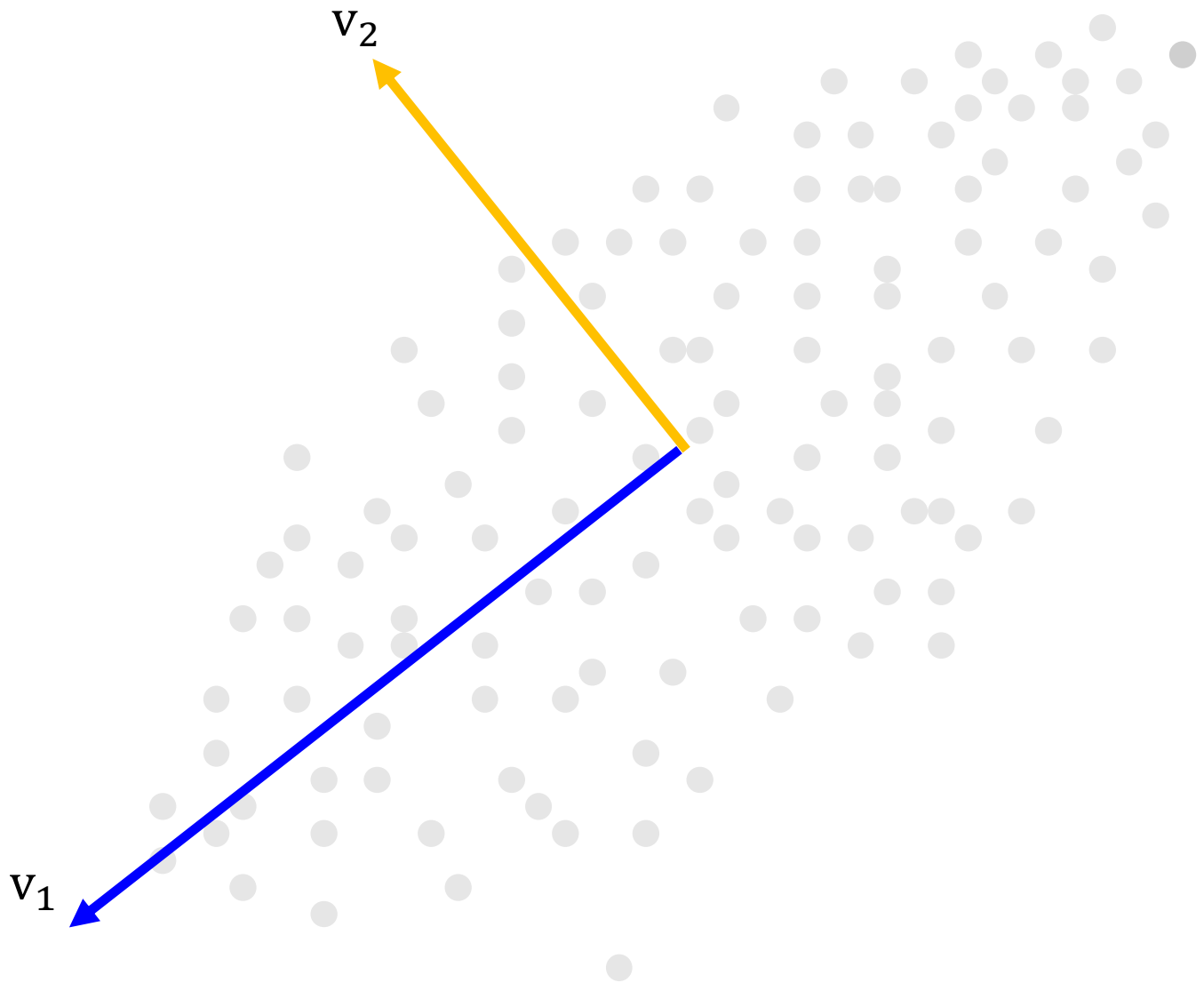
$$\mathbf{S}\left(\mathbf{S}\left(\mathbf{S}(\mathbf{S}v)\right)\right) = \begin{bmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{bmatrix} \begin{bmatrix} -14.1 \\ -6.4 \end{bmatrix} = \begin{bmatrix} -33.3 \\ -15.1 \end{bmatrix}$$

What are the slopes?

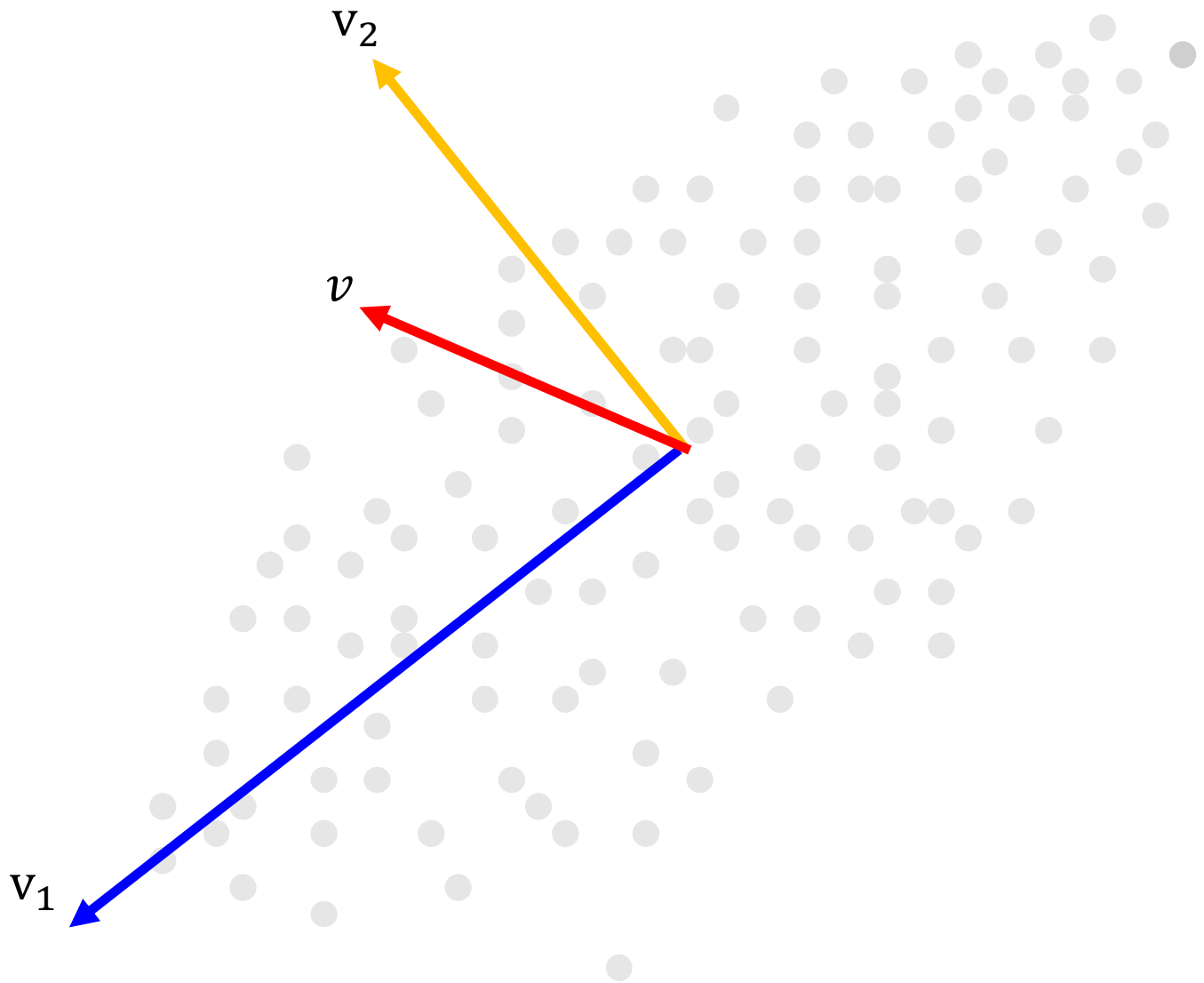
Vector	**Slope**
$v = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$	-1
$\mathbf{S}v = \begin{bmatrix} -1.2 \\ -0.2 \end{bmatrix}$	0.167
$\mathbf{S}(\mathbf{S}v) = \begin{bmatrix} -2.5 \\ -1.0 \end{bmatrix}$	0.400
$\mathbf{S}(\mathbf{S}(\mathbf{S}v)) = \begin{bmatrix} -6.0 \\ -2.7 \end{bmatrix}$	0.450
$\mathbf{S}\left(\mathbf{S}(\mathbf{S}(\mathbf{S}v))\right) = \begin{bmatrix} -14.1 \\ -6.4 \end{bmatrix}$	0.454
$\mathbf{S}\left(\mathbf{S}\left(\mathbf{S}(\mathbf{S}v)\right)\right) = \begin{bmatrix} -33.3 \\ -15.1 \end{bmatrix}$	0.454

Vector v begins to turn, until it does not turn anymore, and instead just changes in magnitude.

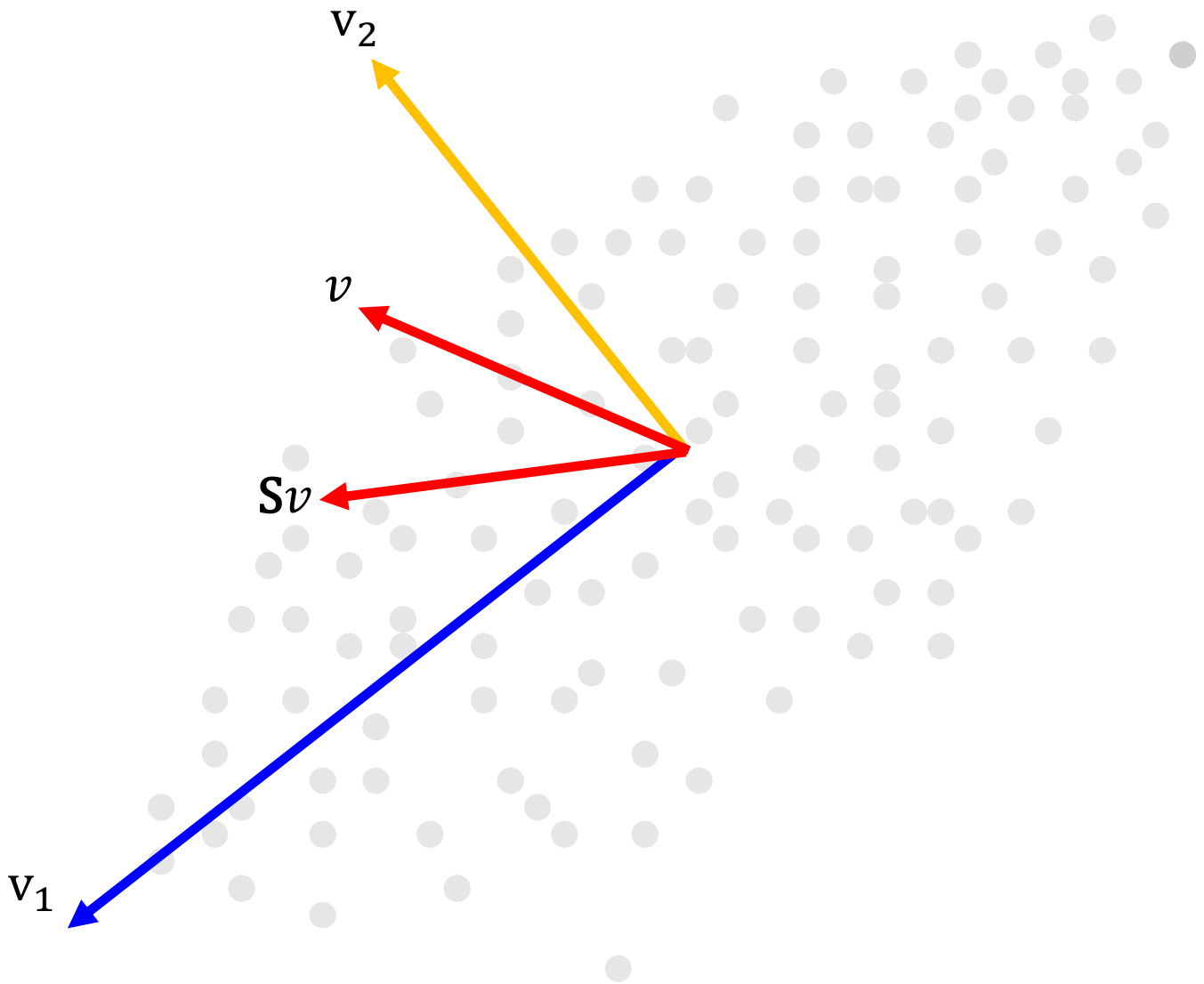
Multiplying vector v by covariance matrix $\mathbf{S}$ rotates it



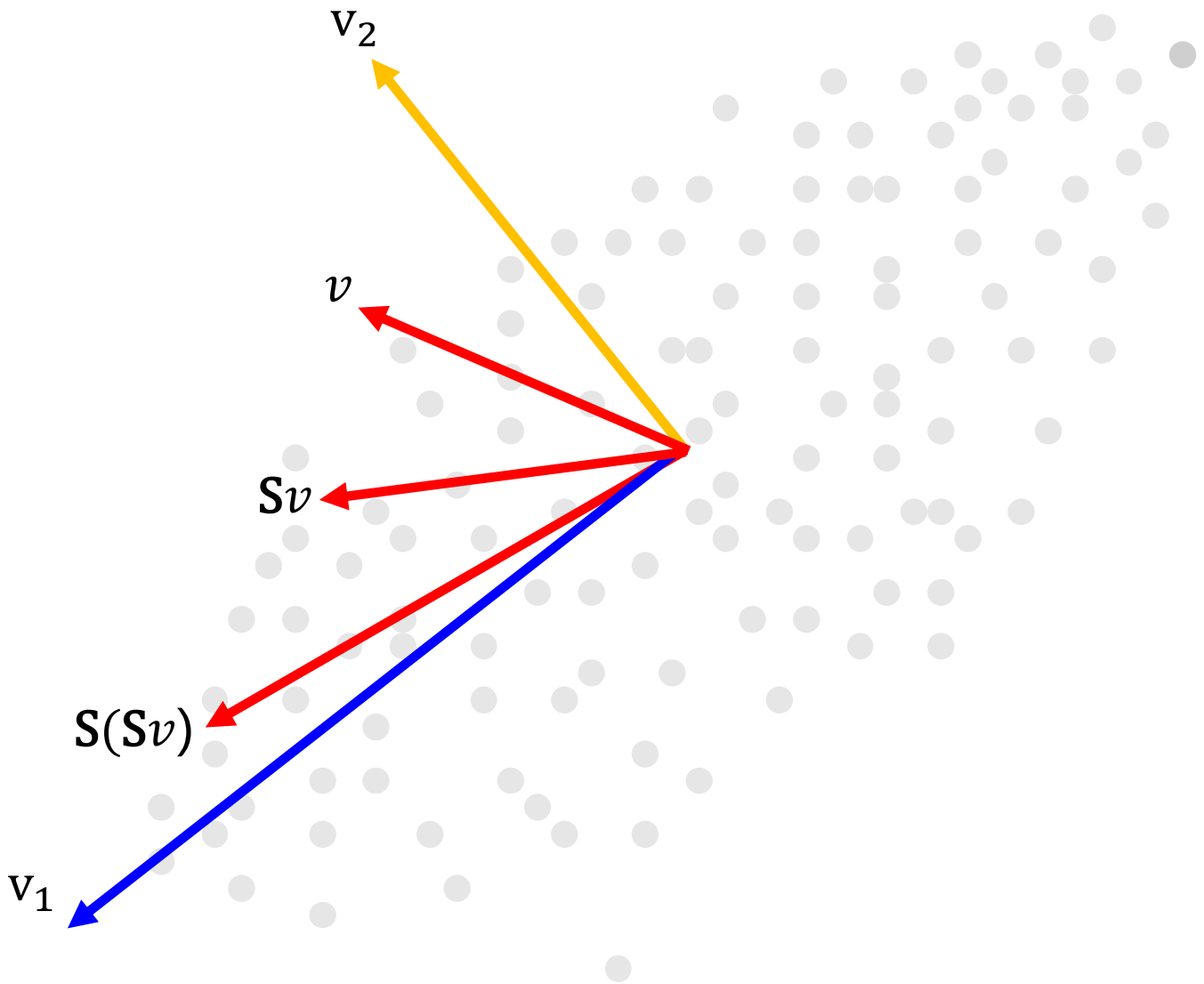
Multiplying vector v by covariance matrix $\mathbf{S}$ rotates it



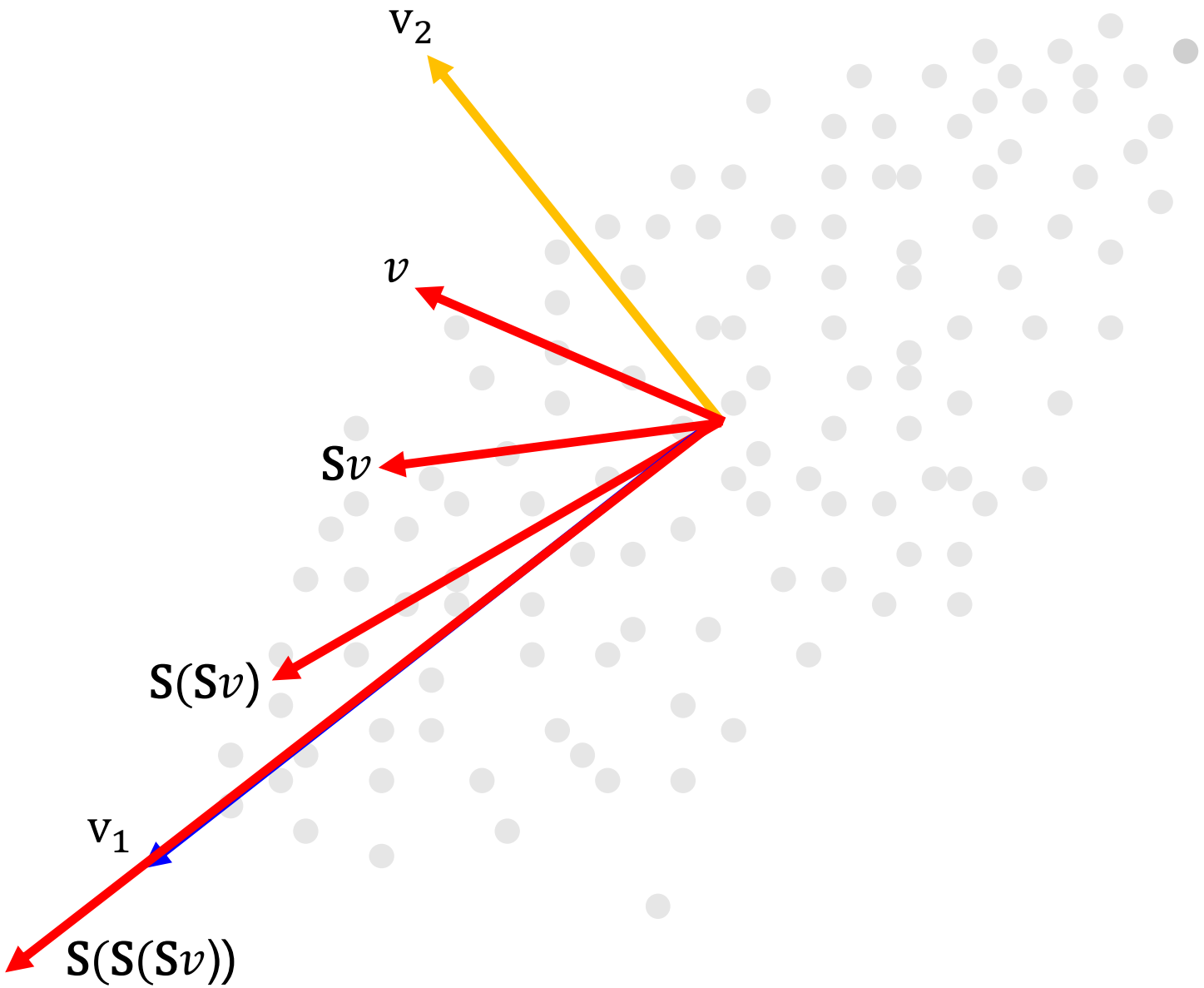
Multiplying vector v by covariance matrix $\mathbf{S}$ rotates it



Multiplying vector v by covariance matrix $\mathbf{S}$ rotates it



Multiplying vector v by covariance matrix $\mathbf{S}$ rotates it



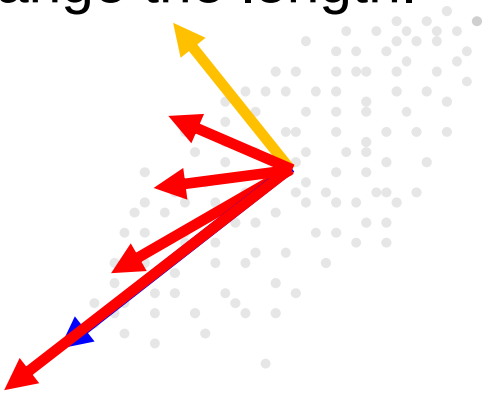
Multiplying vector v by covariance matrix $\mathbf{S}$ rotates it

Visually (here based on “intuition”), multiplying a vector v by covariance matrix $\mathbf{S}$ may turn it toward the direction of greatest variability in the data.

If we keep on multiplying by $\mathbf{S}$, then we get to a point where v does not rotate anymore, but instead just increases in magnitude.

Therefore, to find the direction of greatest variability in the data, we may want to identify a vector v such that multiplication by the covariance matrix $\mathbf{S}$ does not rotate it, but could change the length.

That is, we want $\mathbf{S}v = \lambda v$, where λ is a scalar that could change vector length, but not rotate it.



We seek eigenvectors and eigenvalues

The vector v that has this property is known as an **eigenvector**, and the scalar λ is known as its corresponding **eigenvalue**.

Therefore, principal components are eigenvectors.

The first principal component is the eigenvector that has the greatest eigenvalue.

The second principal component is the eigenvector that has the second greatest eigenvalue.

The jth principal component is the eigenvector that has the jth greatest eigenvalue.

Finding principal components (PCs)

1. Identify eigenvalues by solving the equation

$$\det\left(\mathbf{S} - \lambda \mathbf{I}_p\right) = 0$$

which will yield a set $\{\lambda_1, \lambda_2, \ldots, \lambda_p\}$ of p eigenvalues as solutions.

2. Given eigenvalue λ_j, discover eigenvector v_j by solving system

$$\mathbf{S} v_j = \lambda_j v_j$$

3. However, we want the eigenvectors to be unit length to avoid multiple solutions, which we can obtain as

$$v_j = \frac{v_j}{\left\| v_j \right\|_2}$$

4. Sort $\{v_1, v_2, \ldots, v_p\}$ such that $\lambda_j \geq \lambda_k$ for $j < k$.

Projecting the data onto top M principal components

Define the matrix $\mathbf{V} \in \mathbb{R}^{p \times p}$ of eigenvectors as

$$\mathbf{V} = [\mathrm{v}_1 \quad \mathrm{v}_2 \quad \cdots \quad \mathrm{v}_p]$$

and the elements of v_j are termed the loadings of PC j.

Suppose we wish to represent our dataset $\mathbf{X}$ using only M features instead of p.

Then define the matrix of the first M principal components as

$$\mathbf{V}_M = [\mathrm{v}_1 \quad \mathrm{v}_2 \quad \cdots \quad \mathrm{v}_M]$$

We can project the N p-dimensional datapoint into a smaller dimension $M \ll p$ by

$$\mathbf{Z} = \mathbf{X}\mathbf{V}_M$$

Projecting the data onto top M principal components

In more detail, we have that the matrix $Z \in \mathbb{R}^{N \times M}$ is

$$\mathbf{Z}_M = \begin{bmatrix} z_{11} & z_{12} & \cdots & z_{1M} \\ z_{21} & z_{22} & \cdots & z_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ z_{N1} & z_{N2} & \cdots & z_{NM} \end{bmatrix} = \begin{bmatrix} z_1^T \\ z_2^T \\ \vdots \\ z_N^T \end{bmatrix} = \begin{bmatrix} \mathrm{z}_1 & \mathrm{z}_2 & \cdots & \mathrm{z}_M \end{bmatrix}$$

the projection of the N observations onto top M principal components.

Because we define $\mathbf{Z}_M = \mathbf{X}\mathbf{V}_M$, we have for $m = 1,2, \ldots, M$,

$$\mathrm{z}_m = \mathbf{X}\mathrm{v}_m = \begin{bmatrix} x_1^T \mathrm{v}_m \\ x_2^T \mathrm{v}_m \\ \vdots \\ x_N^T \mathrm{v}_m \end{bmatrix}$$

Projecting the data onto top M principal components

The projection of observation i on the mth principal component is

$$z_{im} = x_i^T \mathrm{v}_m = \sum_{j=1}^{p} x_{ij} v_{jm}$$

Each observation i can be represented in M-dimensional space as

$$z_i = \begin{bmatrix} z_{i1} \\ z_{i2} \\ \vdots \\ z_{iM} \end{bmatrix} = \begin{bmatrix} x_i^T \mathrm{v}_1 \\ x_i^T \mathrm{v}_2 \\ \vdots \\ x_i^T \mathrm{v}_M \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{p} x_{ij} v_{j1} \\ \sum_{j=1}^{p} x_{ij} v_{j2} \\ \vdots \\ \sum_{j=1}^{p} x_{ij} v_{jM} \end{bmatrix}$$

How do we know this is dimension of greatest variation?

We visually (“intuitively”) discovered that eigenvectors may describe directions of greatest variability in the data.

However, we need to show that eigenvectors are indeed these directions.

Consider any vector v; we will prove that v is an eigenvector.

We will project points onto v, and then measure the variance of these projected points.

For $i = 1,2, \dots, N$, the projection of observation x_i^T onto v is

$$z_i = x_i^T v$$

The variance of the projected observations

The variance of the data $\mathrm{z} = [z_1, z_2, \ldots, z_N]^T$ of projected observations is

$$Var(\mathrm{z}) = \frac{1}{N}\sum_{i=1}^{N}(z_i - \bar{\mathrm{z}})^2$$

where $\bar{\mathrm{z}} = 0$ because

$$\bar{\mathrm{z}} = \frac{1}{N}\sum_{i=1}^{N} z_i = \frac{1}{N}\sum_{i=1}^{N} x_i^T v$$

$$= \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{p} x_{ij} v_j = \sum_{j=1}^{p}\left(\frac{1}{N}\sum_{i=1}^{N} x_{ij}\right) v_j$$

$$= \sum_{j=1}^{p} \bar{\mathrm{x}}_j v_j = 0$$

The variance of the projected observations

We therefore have

$$Var(\mathrm{z}) = \frac{1}{N}\sum_{i=1}^{N} z_i^2 = \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{j=1}^{p} x_{ij}v_j\right)^2$$

We want to choose v such that $Var(\mathrm{z})$ is maximal.

However, we cannot just simply take the gradient and set it to the 0 vector, because we also require that $\|v\|_2^2 = \|v\|_2 = 1$.

Therefore, this is a constrained optimization problem, which can be tackled using Lagrange multipliers (recall lecture on support vector machines).

Gradient of constrained optimization problem

We therefore have

$$\mathcal{L}(z,\lambda) = Var(\mathrm{z}) - \lambda(\|v\|_2^2) = \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{j=1}^{p} x_{ij}v_j\right)^2 - \lambda\left(\sum_{j=1}^{p} v_j^2 - 1\right)$$

where λ is the Lagrange multiplier.

Taking the partial derivative with respect to v_k, $k = 1,2,\ldots,p$, we have

$$\frac{\partial}{\partial v_k}\mathcal{L}(z,\lambda) = \frac{2}{N}\sum_{i=1}^{N} x_{ik}\left(\sum_{j=1}^{p} x_{ij}v_j\right) - 2\lambda v_k$$

$$= 2\sum_{j=1}^{p}\left(\frac{1}{N}\sum_{i=1}^{N} x_{ik}x_{ij}\right)v_j - 2\lambda v_k$$

Gradient of constrained optimization problem

Recall that

$$\mathbf{S}_{kj} = Cov(\mathrm{x}_k, \mathrm{x}_j) = \frac{1}{N}\sum_{i=1}^{N} x_{ik}x_{ij}$$

Plugging into the partial derivative gives

$$\frac{\partial}{\partial v_k}\mathcal{L}(z,\lambda) = 2\sum_{j=1}^{p}\mathbf{S}_{kj}v_j - 2\lambda v_k$$

Note that the term $\sum_{j=1}^{p}\mathbf{S}_{kj}v_j$ represents the kth row of the sample covariance matrix $\mathbf{S}$ multiplied by the vector v.

Gradient of constrained optimization problem

The gradient vector is therefore

$$\frac{\partial}{\partial v}\mathcal{L}(z,\lambda) = \begin{bmatrix} 2\sum_{j=1}^{p}\mathbf{S}_{1j}v_j - 2\lambda v_1 \\ 2\sum_{j=1}^{p}\mathbf{S}_{2j}v_j - 2\lambda v_2 \\ \vdots \\ 2\sum_{j=1}^{p}\mathbf{S}_{pj}v_j - 2\lambda v_p \end{bmatrix} = 2\begin{bmatrix} \sum_{j=1}^{p}\mathbf{S}_{1j}v_j \\ \sum_{j=1}^{p}\mathbf{S}_{2j}v_j \\ \vdots \\ \sum_{j=1}^{p}\mathbf{S}_{pj}v_j \end{bmatrix} - 2\lambda\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_p \end{bmatrix}$$

$$= 2\mathbf{S}v - 2\lambda v$$

Setting the gradient vector to the 0 vector, we find the system

$$\mathbf{S}v = \lambda v$$

which demonstrates that v is an eigenvector and that the Lagrange multiplier λ is the eigenvalue.

What is the variance along the eigenvector v?

We evaluate the variance in observations along eigenvector v, by projecting each observation onto v as $z_i = x_i^T v$, and then evaluating the variance across the projections as

$$Var(\mathrm{z}) = \frac{1}{N}\sum_{i=1}^{N} z_i^2 = \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{j=1}^{p} x_{ij} v_j\right)^2$$

$$= \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{j=1}^{p} x_{ij} v_j\right)\left(\sum_{k=1}^{p} x_{ik} v_k\right)$$

$$= \sum_{j=1}^{p}\sum_{k=1}^{p}\left(\frac{1}{N}\sum_{i=1}^{N} x_{ij} x_{ik}\right) v_j v_k$$

$$= \sum_{j=1}^{p}\sum_{k=1}^{p} \mathbf{S}_{jk} v_j v_k$$

What is the variance along the eigenvector v?

Reducing further gives

$$Var(\mathrm{z}) = \sum_{j=1}^{p}\sum_{k=1}^{p} \mathbf{S}_{jk} v_j v_k = \sum_{j=1}^{p}\left(\sum_{k=1}^{p} \mathbf{S}_{jk} v_k\right) v_j$$

Recall that $\sum_{k=1}^{p} \mathbf{S}_{jk} v_k = \lambda v_j$ and so

$$Var(\mathrm{z}) = \sum_{j=1}^{p} (\lambda v_j) v_j = \lambda \sum_{j=1}^{p} v_j^2 = \lambda \|v\|_2^2 = \lambda$$

Therefore, in the direction of eigenvector v, the variance of the projected points is the eigenvalue λ.

Principal components regression (PCR)

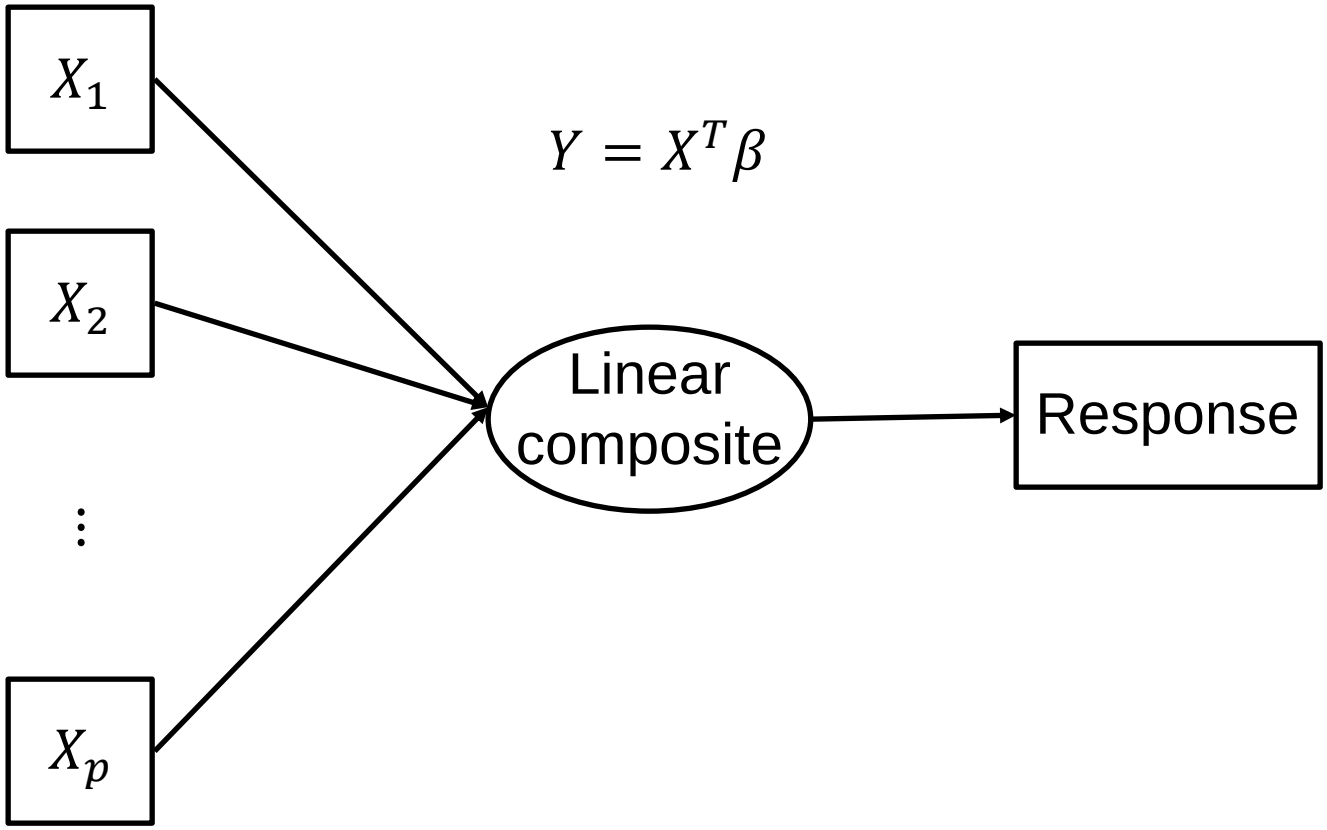
We have shown that principal components analysis can be a great tool for visualizing high-dimensional data in low dimension.

However, our original motivation for examining dimensionality reduction techniques was to reduce the number of features in our regression, by generating independent composite features that account for correlations of original features.

Our original training data was the set of tuples (x_i, y_i), $i = 1,2, \ldots, N$.

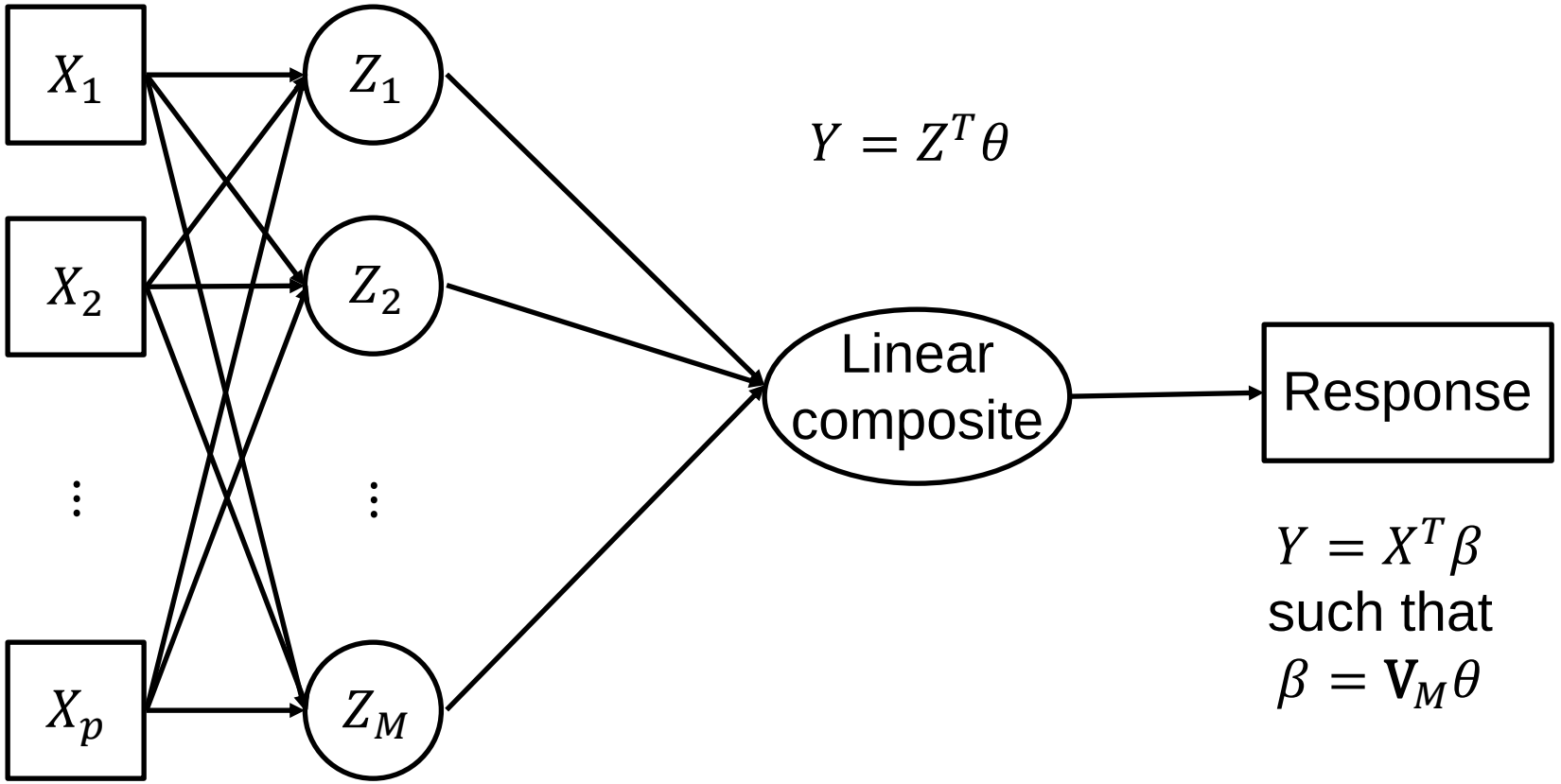
However, after performing PCA and projecting each input $x_i \in \mathbb{R}^p$ onto the top M PCs, we have a new training set (z_i, y_i) where $z_i \in \mathbb{R}^M$.

Visual illustration of least squares regression



Visual illustration of principal components regression

$$Z^T = X^T \mathbf{V}_M$$



$$Y = Z^T \theta$$

$$Y = X^T \beta$$

such that

$$\beta = \mathbf{V}_M \theta$$

Formalizing principal components regression (PCR)

To perform PCR, we identify the top M PCs of $\mathbf{X} \in \mathbb{R}^{N \times p}$ and sort them in $\mathbf{V}_M = [\mathrm{v}_1 \quad \mathrm{v}_2 \quad \cdots \quad \mathrm{v}_M]$ by eigenvalue, with $\mathbf{V}_M \in \mathbb{R}^{p \times M}$.

Project the original datapoints onto the top M PCs to form projected data matrix $\mathbf{Z}_M \in \mathbb{R}^{N \times M}$ as $\mathbf{Z}_M = \mathbf{X}\mathbf{V}_M$.

We will then assume response $\mathbf{y}$ can be predicted by

$$\mathbf{y} = \mathbf{Z}_M \theta$$

where $\theta \in \mathbb{R}^M$ is a vector of M model parameters $\theta^T = [\theta_1, \theta_2, \ldots, \theta_M]$.

Note that with $\beta = \mathbf{V}_M \theta$, we have

$$\mathbf{y} = \mathbf{Z}_M \theta = \mathbf{X}\mathbf{V}_M \theta = \mathbf{X}\beta$$

our original linear model.

Proportion of variance explained by top M PCs

For a particular principal component, denote the variance in the projected points along that component by λ.

Consider the set $\mathrm{v}_1, \mathrm{v}_2, \ldots, \mathrm{v}_p$ of PCs of the input $x_1, x_2, \ldots, x_N$, ordered by non-increasing variance explained, such that

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p$$

where λ_j, $j = 1, 2, \ldots, p$ is the amount of variation explained by principal component j.

The cumulative proportion of variation of the input explained (CPVE) by the first M PCs is

$$\mathrm{CPVE} = \frac{\lambda_1 + \lambda_2 + \cdots + \lambda_M}{\lambda_1 + \lambda_2 + \cdots + \lambda_p}$$

How many M principal components to choose?

It is often the case that the number M of principal components to include would be to choose the minimum number M such that CPVE explained is at least some threshold τ

$$\text{CPVE} \geq \tau$$

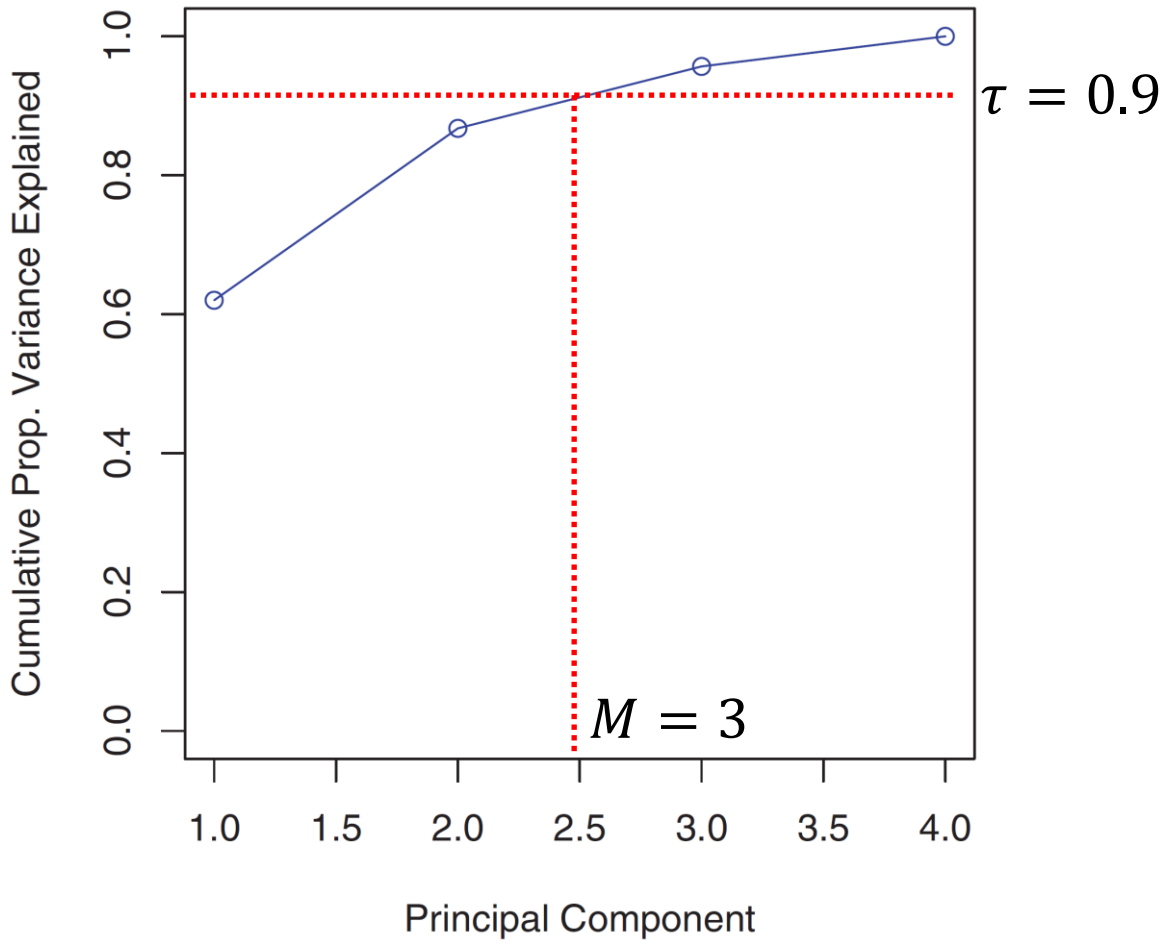
where τ is usually 0.90 or 0.95.

Alternatively, increasing the value M can be thought of as increasing model complexity, since $M = p$ is the same as performing least squares regression without constraints.

One can therefore use cross-validation approaches to identify the optimal value for M.

Choose based on cumulative variance explained

Data on $N = 50$ United States arrest records (each state) based on $p = 4$ features describing numbers of arrests, murders, and rapes, and percentage of population living in urban areas.

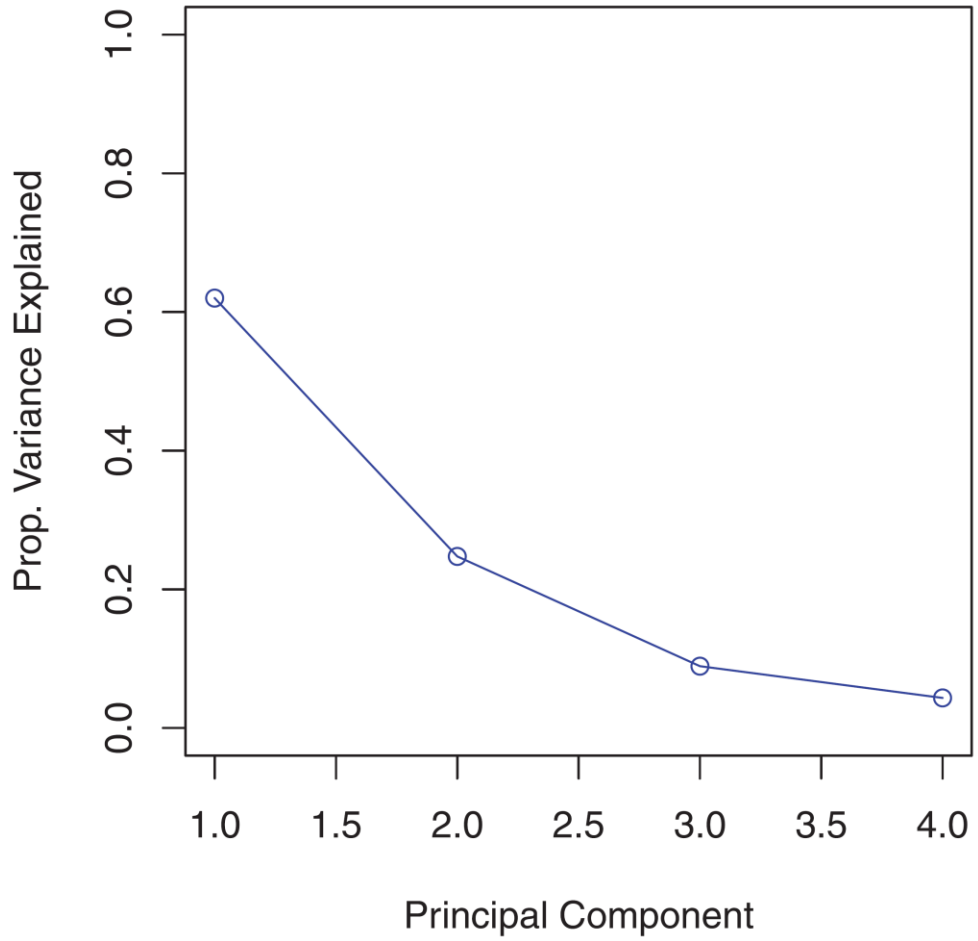


James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Choose based on scree plot

Scree plot gives the proportion of variance explained (PVE) by PC m as a function of m, and we choose the smallest number of PCs that are required to explain a sizeable amount of variability.

$$\frac{\lambda_m}{\lambda_1 + \cdots + \lambda_p}$$

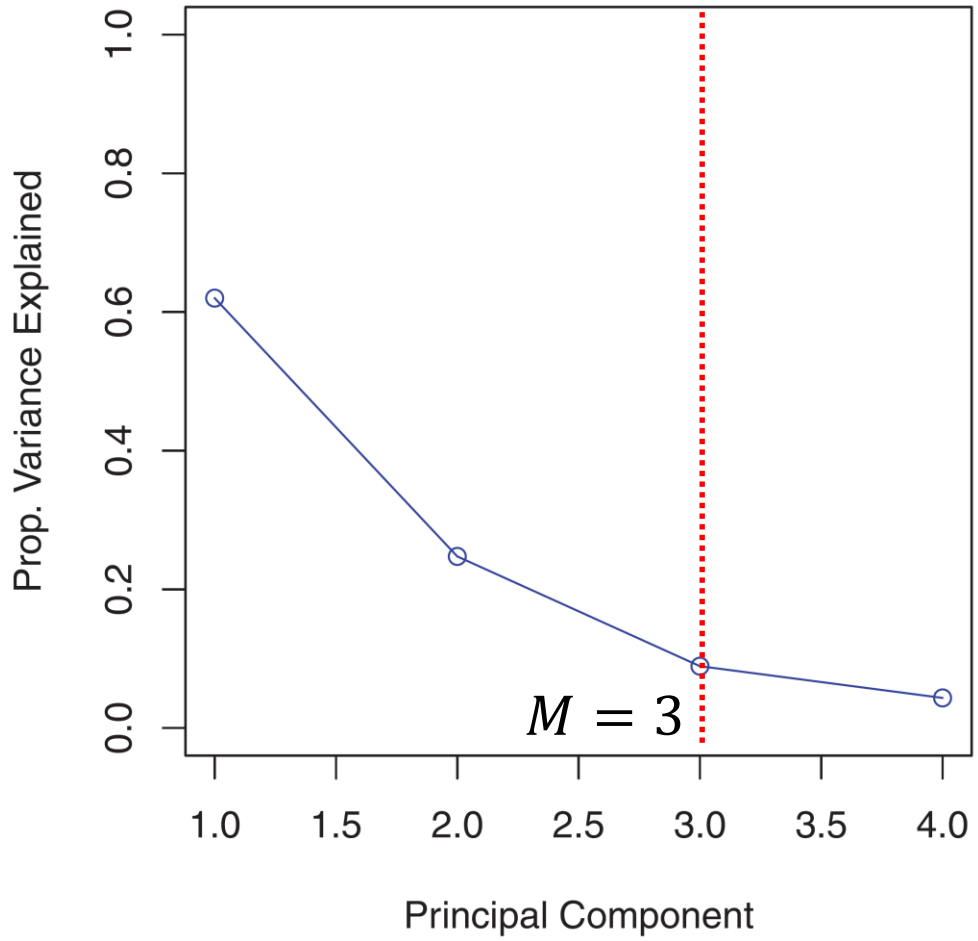


James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Choose based on scree plot

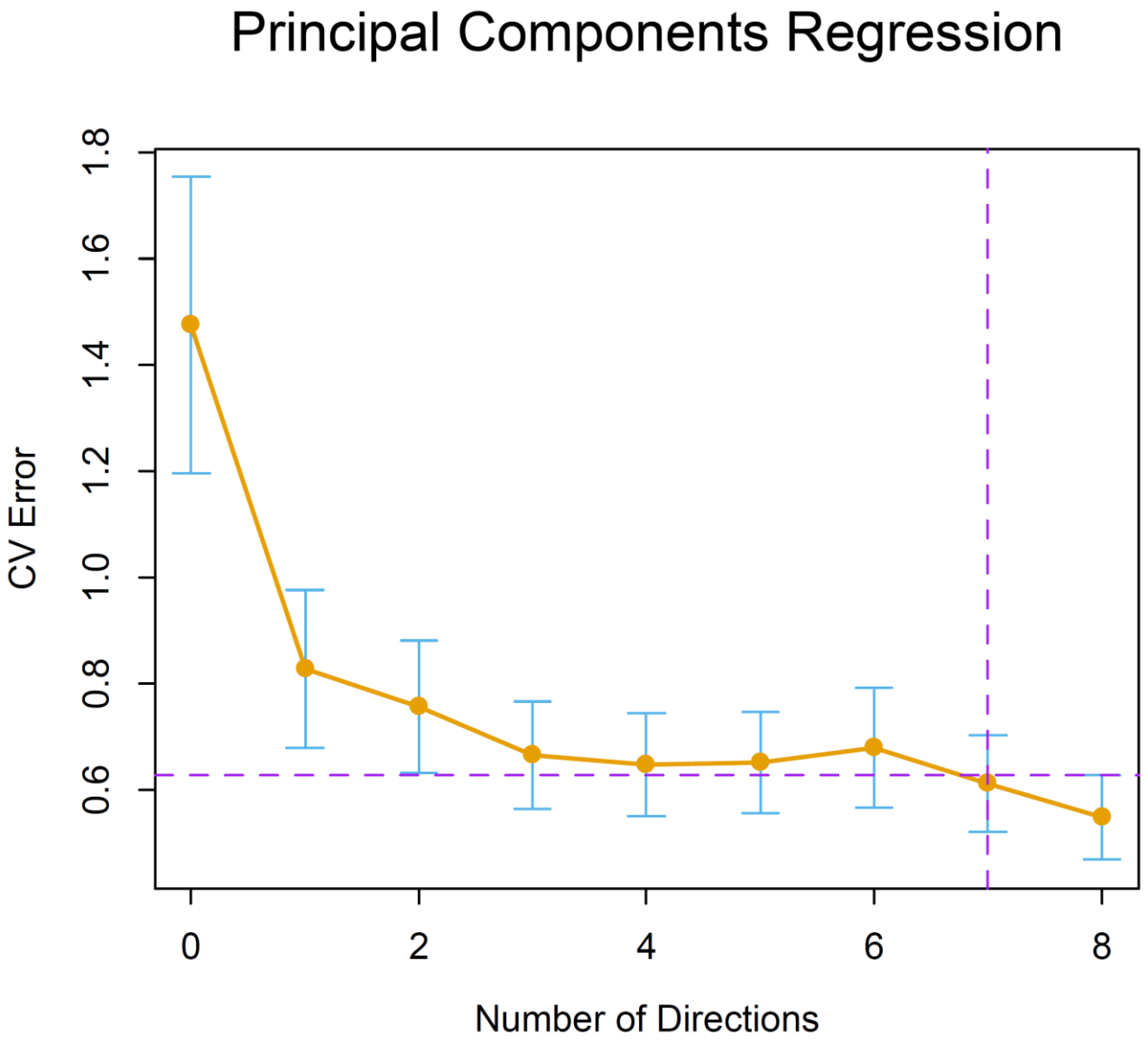
Want to eyeball where this drops off, by attempting to identify an elbow in the graph. We can see that after PC 3, the proportion of variability explained by the PCs is relatively small.

$$\frac{\lambda_m}{\lambda_1 + \cdots + \lambda_p}$$



James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Choose based on cross validation



Hastie *et al* (2009) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd Ed.

Comparison of different approaches

Inferred coefficients and test errors for different approaches on prostate cancer prediction dataset.

Term	LS	Best Subset	Ridge	Lasso	PCR
Intercept	2.465	2.477	2.452	2.468	2.497
lcavol	0.680	0.740	0.420	0.533	0.543
lweight	0.263	0.316	0.238	0.169	0.289
age	−0.141		−0.046		−0.152
lbph	0.210		0.162	0.002	0.214
svi	0.305		0.227	0.094	0.315
lcp	−0.288		0.000		−0.051
gleason	−0.021		0.040		0.232
pgg45	0.267		0.133		−0.056
Test Error	0.521	0.492	0.492	0.479	0.449

Adapted from Hastie *et al* (2009) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd Ed.

Comparison of different approaches

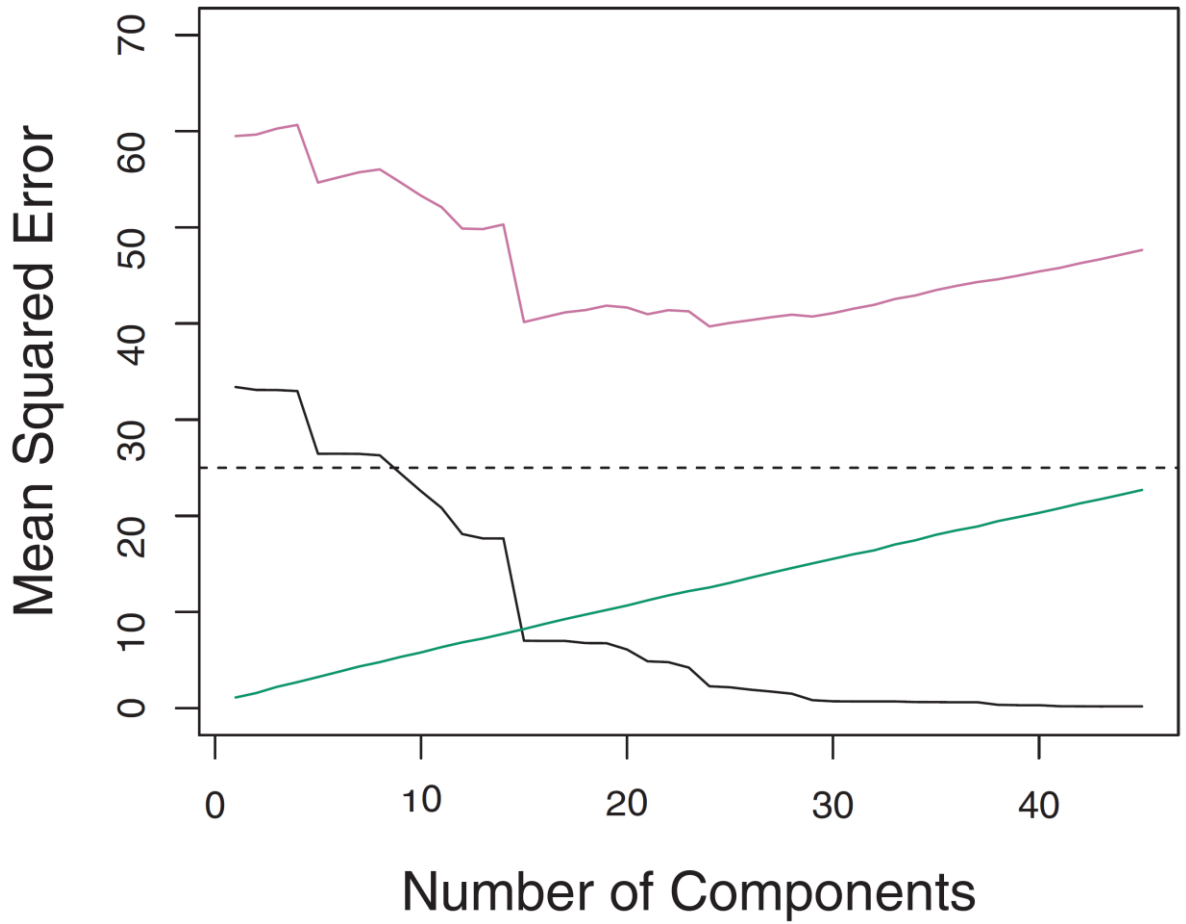
Because $\beta = \mathbf{V}_M \theta$, the parameter estimates ($\hat{\beta}_j$) for each feature j are non-zero because all features are used to construct PCs in $\mathbf{V}_M$, and this is the case even if $\hat{\theta}_m = 0$ for some $m = 1,2,\ldots,M$.

Term	LS	Best Subset	Ridge	Lasso	PCR
Intercept	2.465	2.477	2.452	2.468	2.497
lcavol	0.680	0.740	0.420	0.533	0.543
lweight	0.263	0.316	0.238	0.169	0.289
age	−0.141		−0.046		−0.152
lbph	0.210		0.162	0.002	0.214
svi	0.305		0.227	0.094	0.315
lcp	−0.288		0.000		−0.051
gleason	−0.021		0.040		0.232
pgg45	0.267		0.133		−0.056
Test Error	0.521	0.492	0.492	0.479	0.449

Adapted from Hastie *et al* (2009) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd Ed.

Number of PCs M makes bias-variance tradeoff

Optimal MSE (purple) occurs at an intermediate number of components M, due to a reduction in complexity by increasing squared bias (black) and decreasing variance (green) with smaller M.

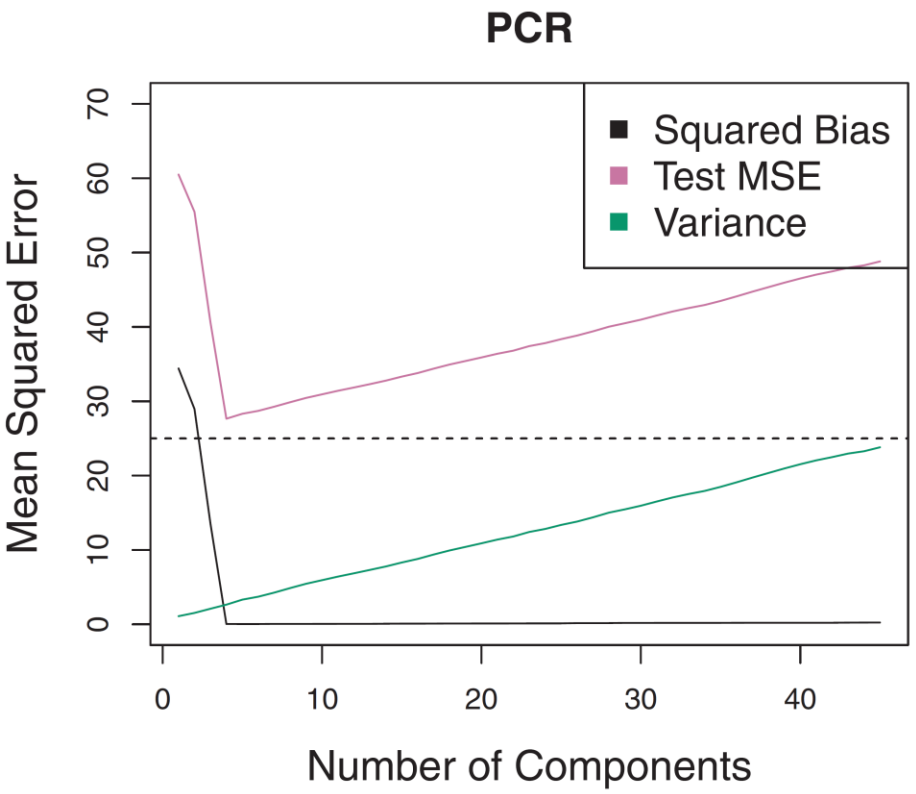


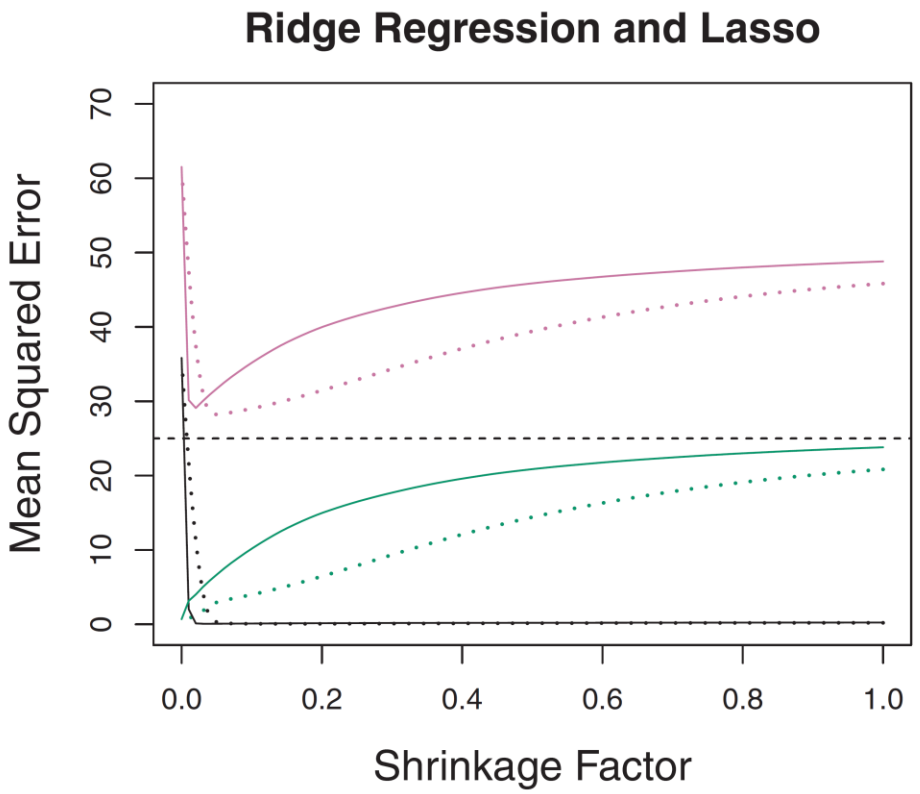
James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Number of PCs M makes bias-variance tradeoff

Dataset in which the first 5 PCs contain all the information about the response.

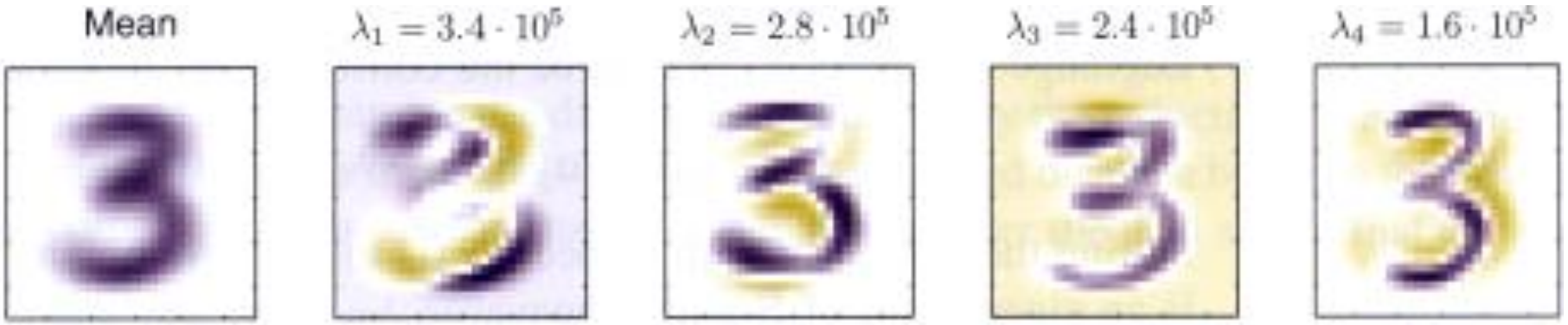
Optimal MSE occurs with exactly $M = 5$ PCs in PCR.





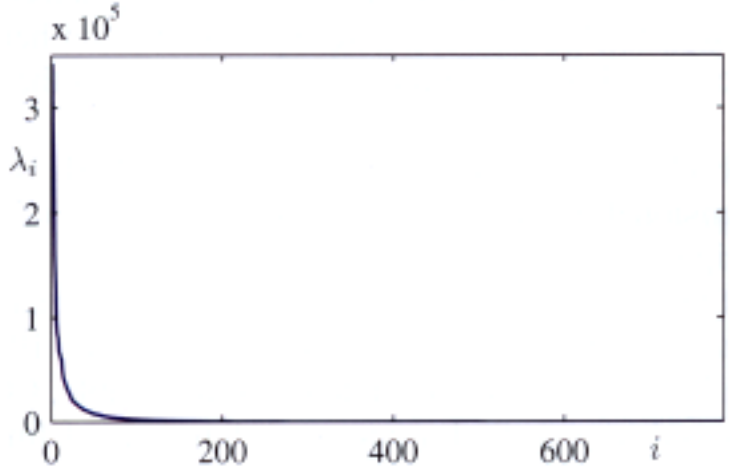
James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

PC discovering important composite features



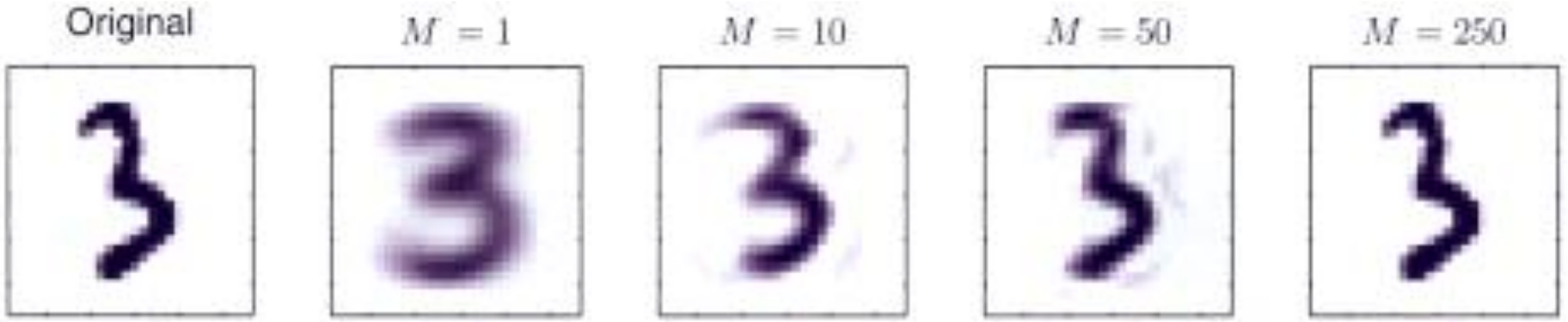
The first 4 PCs capture some of the major characteristics of the handwritten digit.

Certainly far fewer than the full set of features is needs to capture the majority of variation across example digits.



Bishop (2006) *Pattern Recognition and Machine Learning*. Springer.

PC discovering important composite features



As the number of PCs used increases to reconstruct a handwritten digit, the more accurate the reconstruction becomes.

By $M = 250$, the digit is virtually identical to the original, and at $M = 50$ the representation is very close.

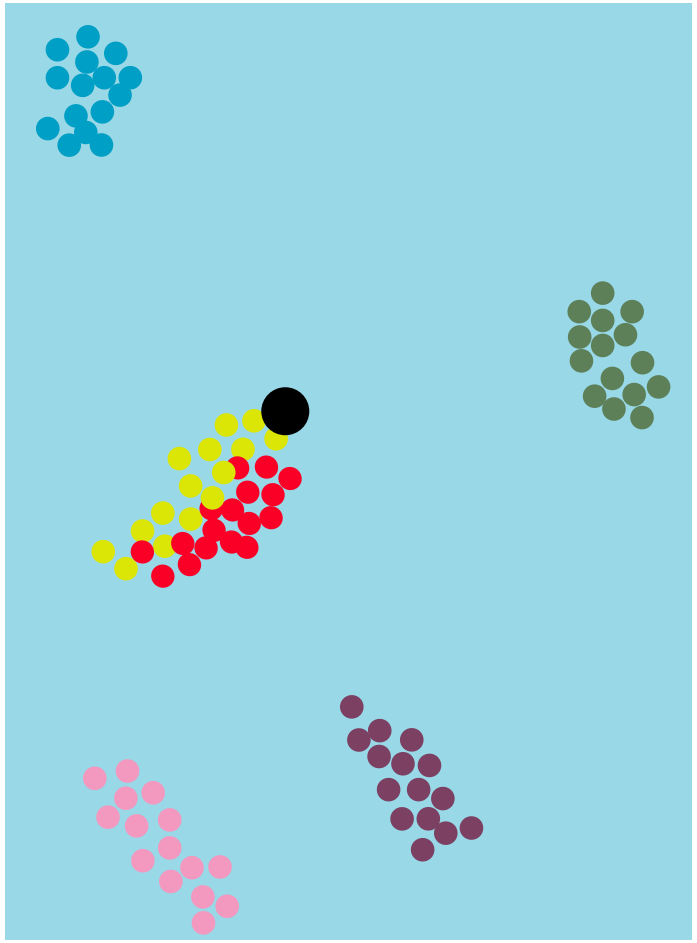
Bishop (2006) *Pattern Recognition and Machine Learning*. Springer.

Prototype clustering

Attempt to assign observations to a discrete number of *K* clusters

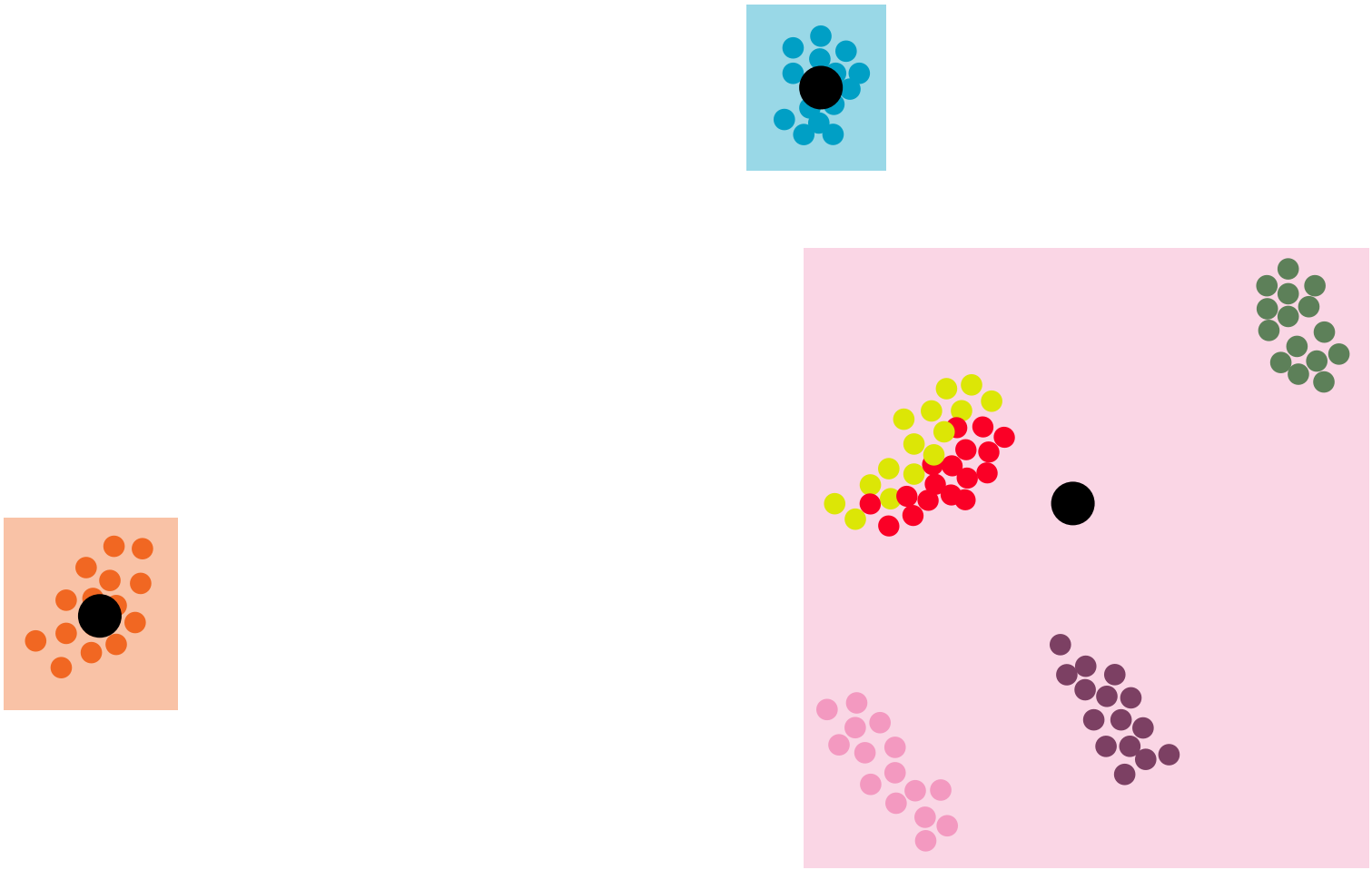
Prototype clustering

K = 2 clusters, with two **prototypes** (black circles) acting as representatives of the two clusters.



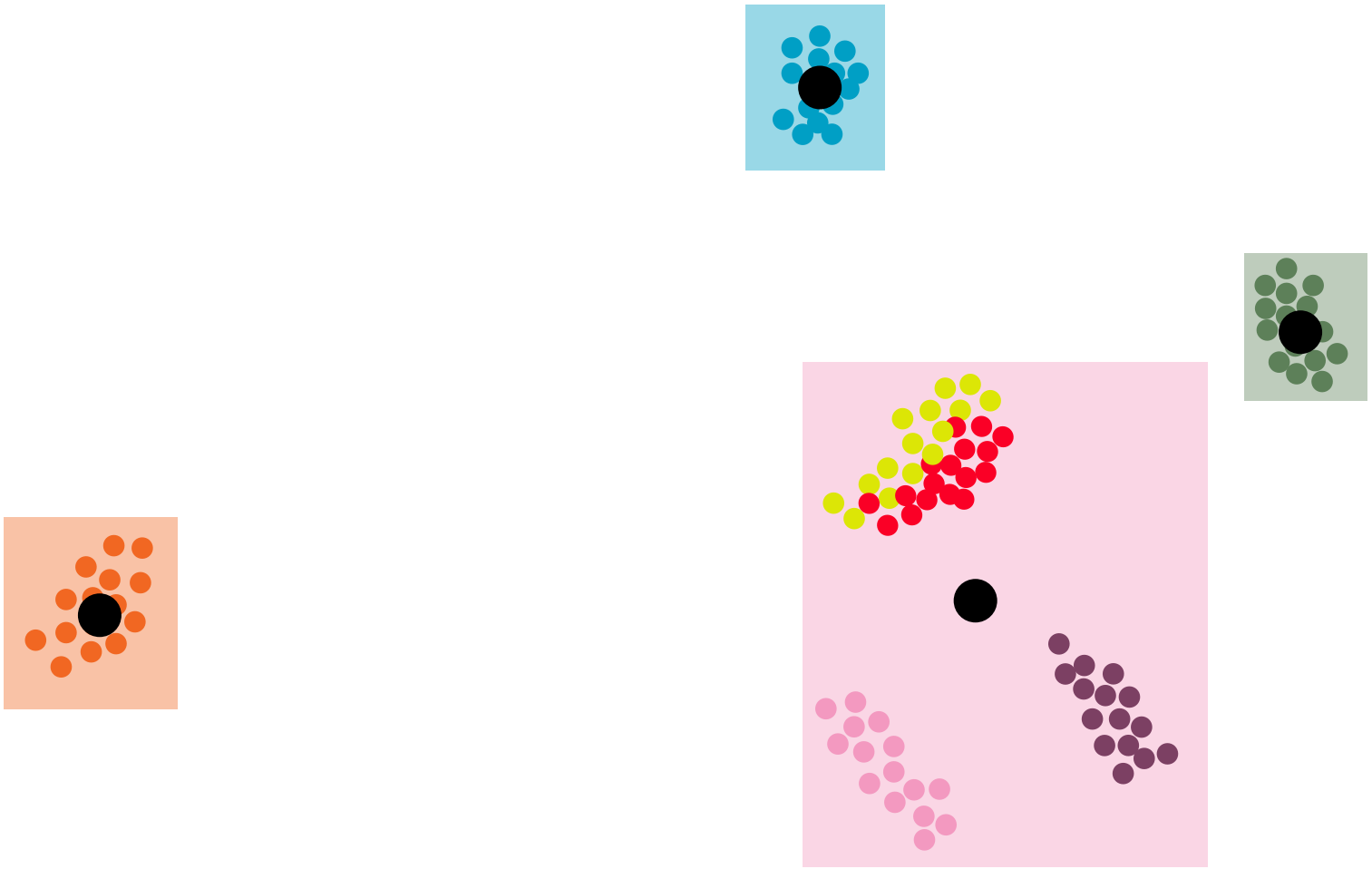
Prototype clustering

K = 3 clusters



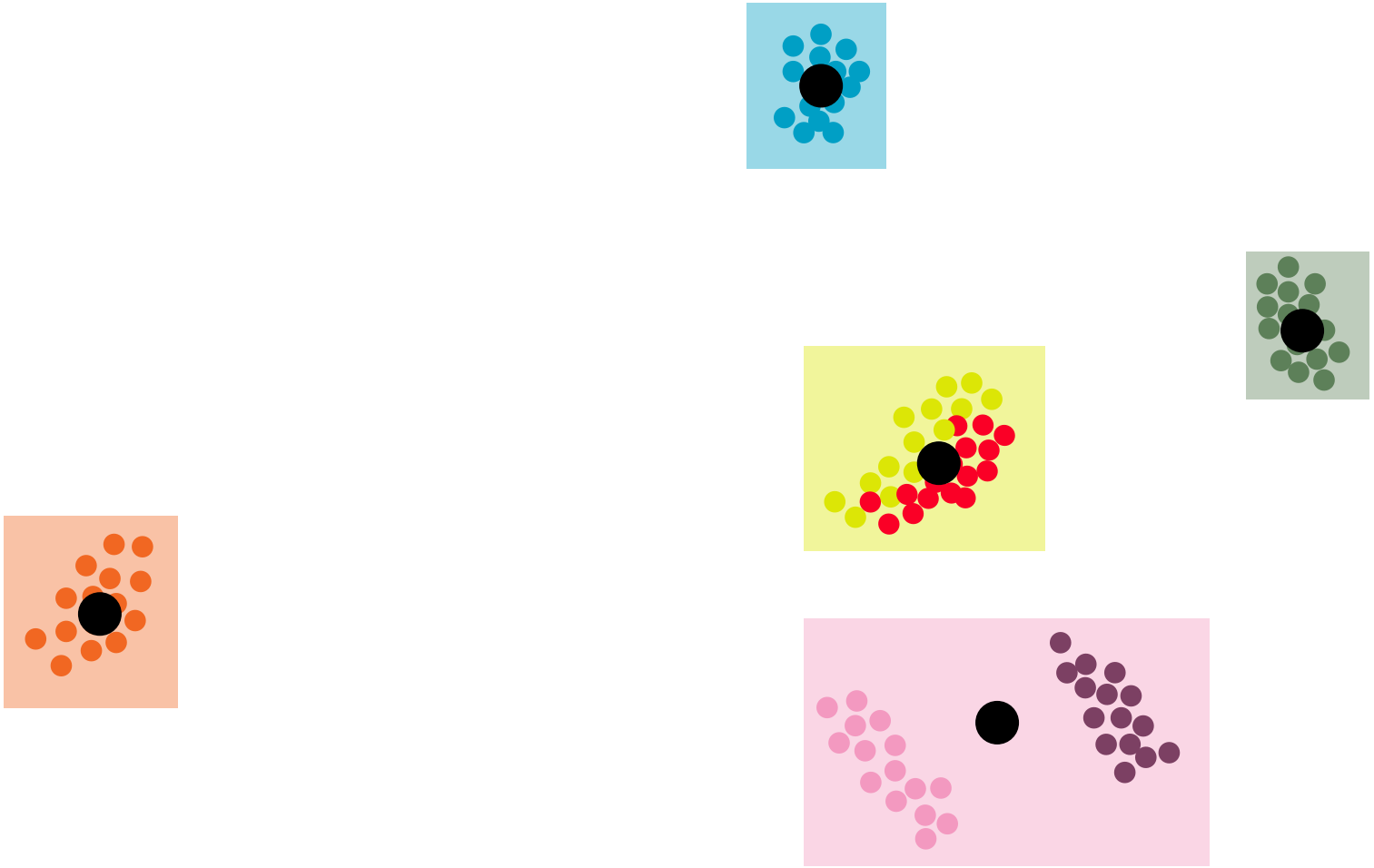
Prototype clustering

K = 4 clusters



Prototype clustering

K = 5 clusters



Prototype clustering

K = 6 clusters

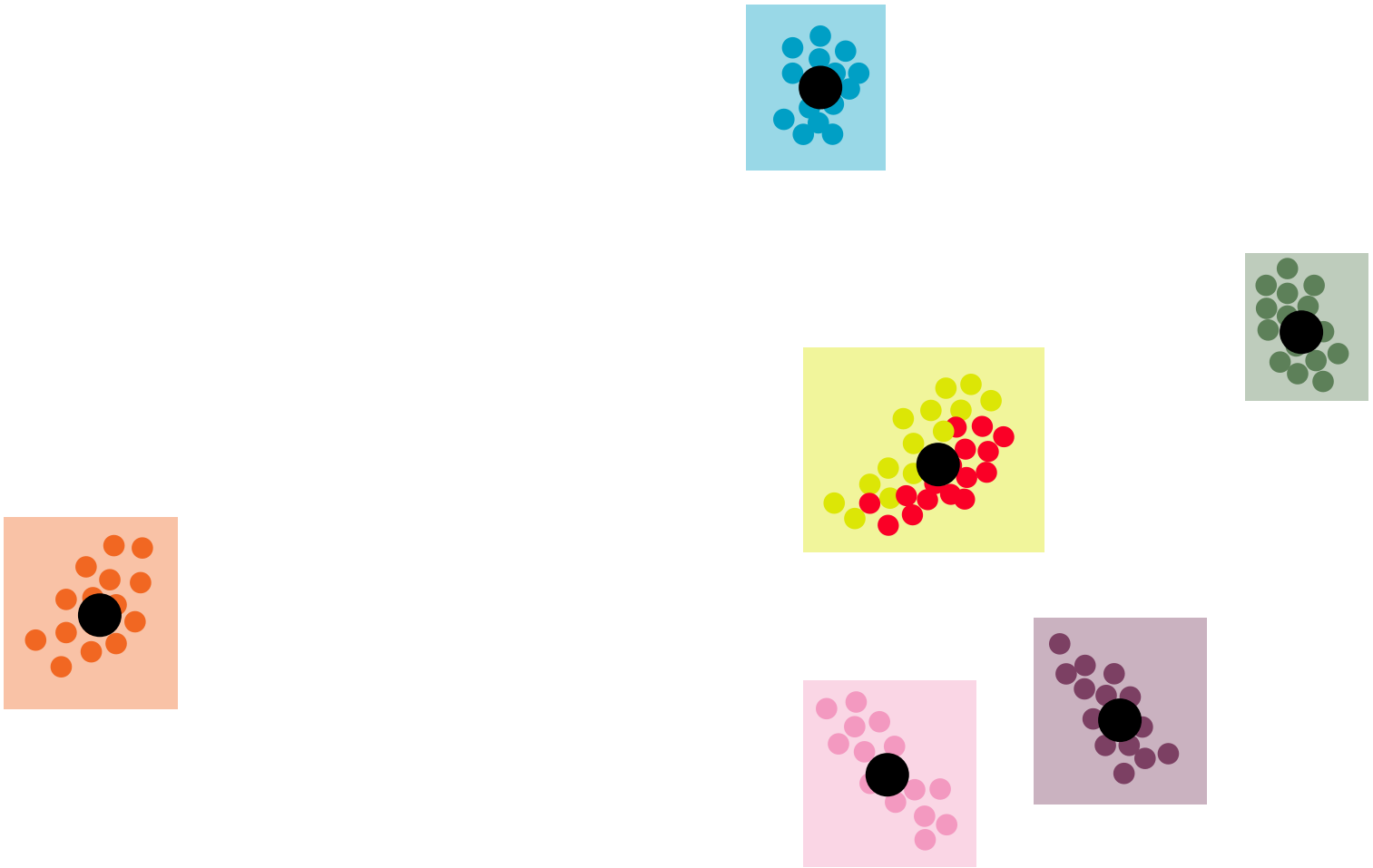
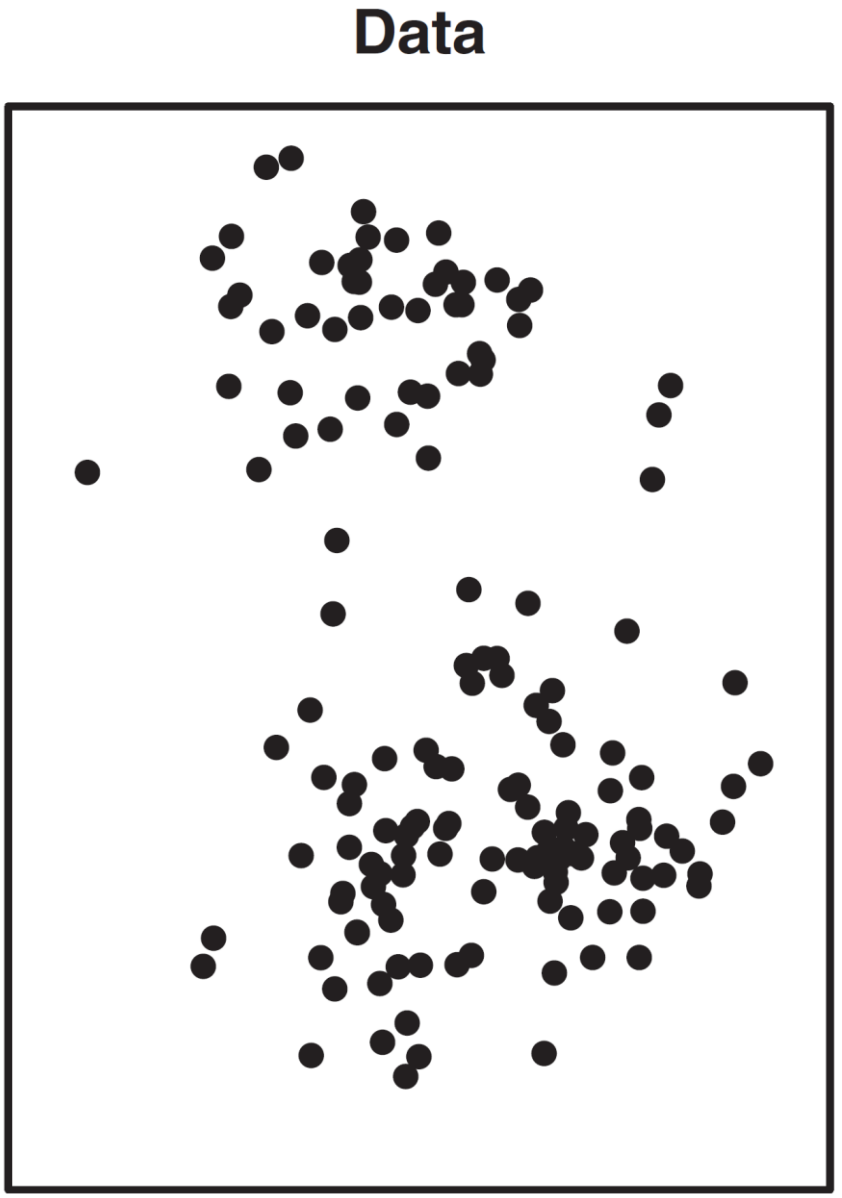


Illustration of K-means clustering with $K = 3$.

For illustration, we depict a set of observations in two-dimensional space.

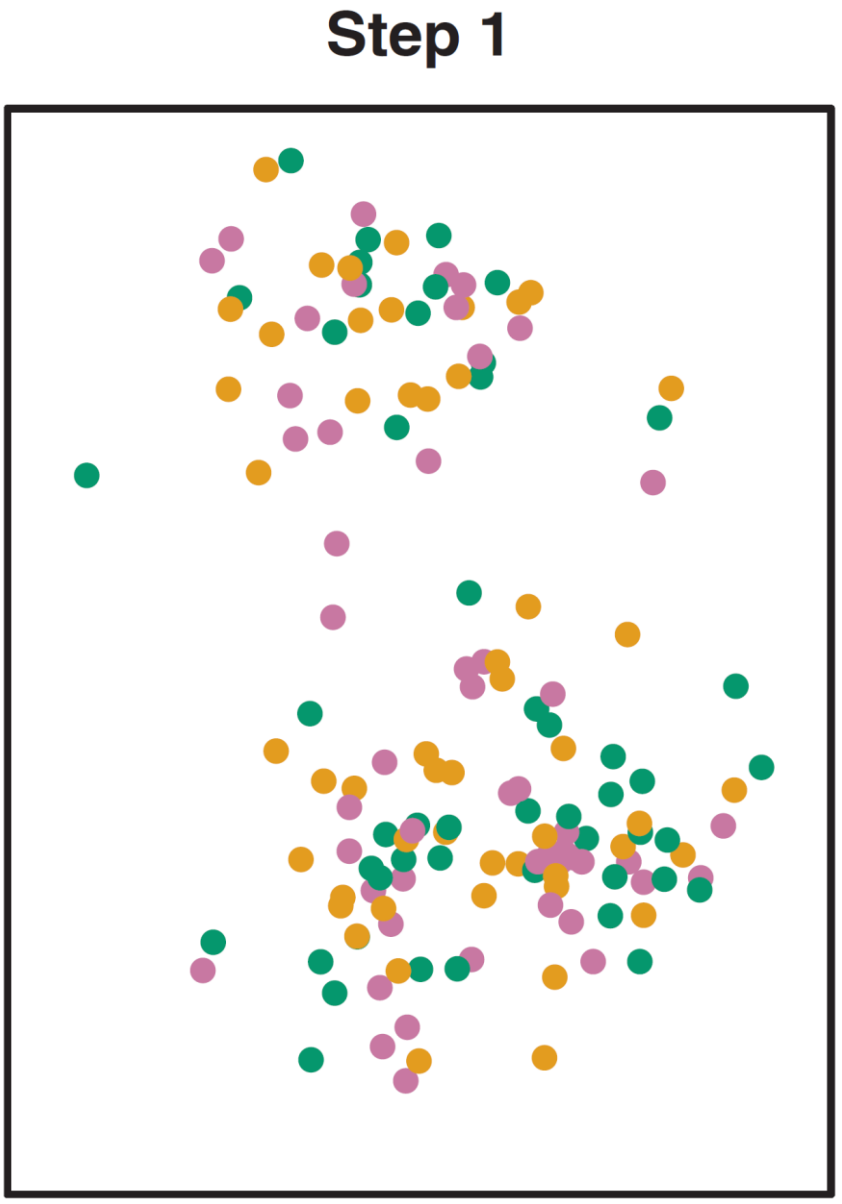
We will assume that there are $K = 3$ clusters, and will follow the K-means clustering algorithm to assign each observation to one of 3 clusters.



Adapted from James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Illustration of K-means clustering with $K = 3$.

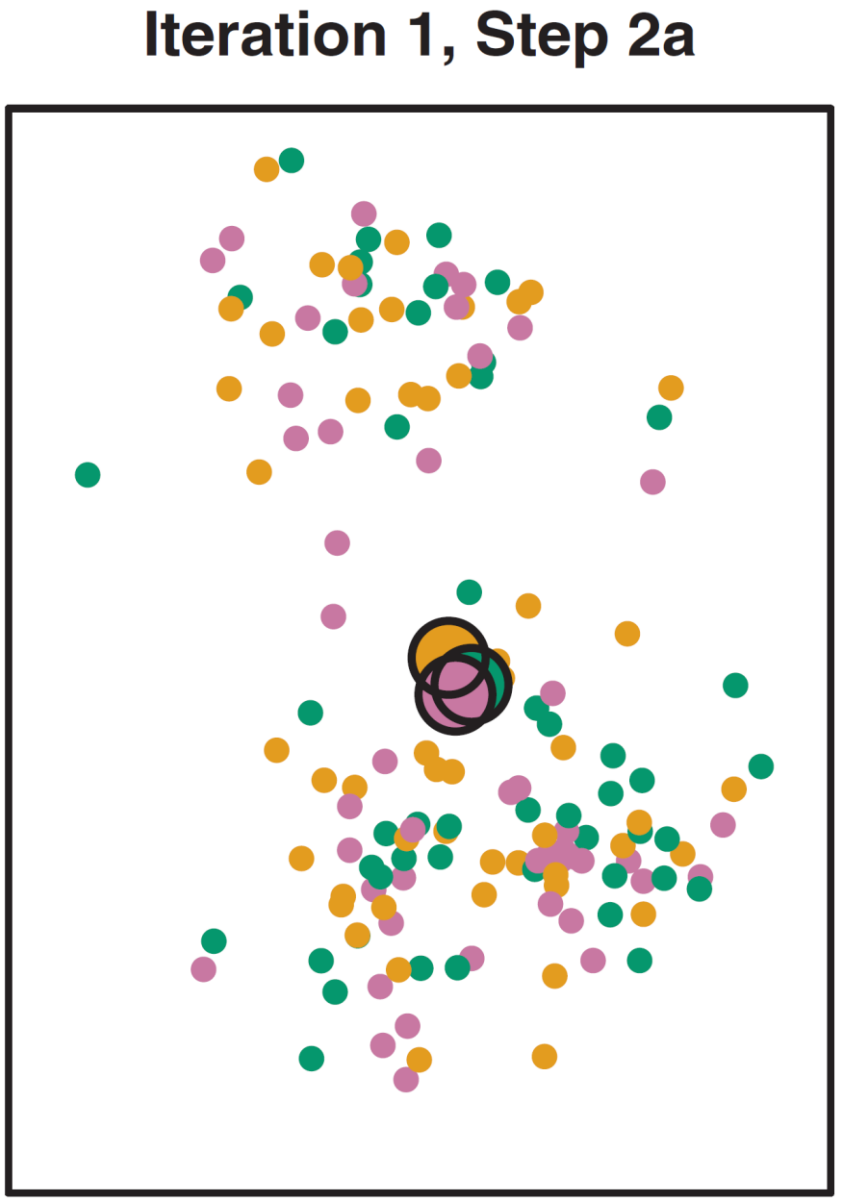
Randomly assign each observation to 1 of 3 clusters (orange, green, or purple), ensuring that at least 1 observation is in each cluster.



Adapted from James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Illustration of K-means clustering with $K = 3$.

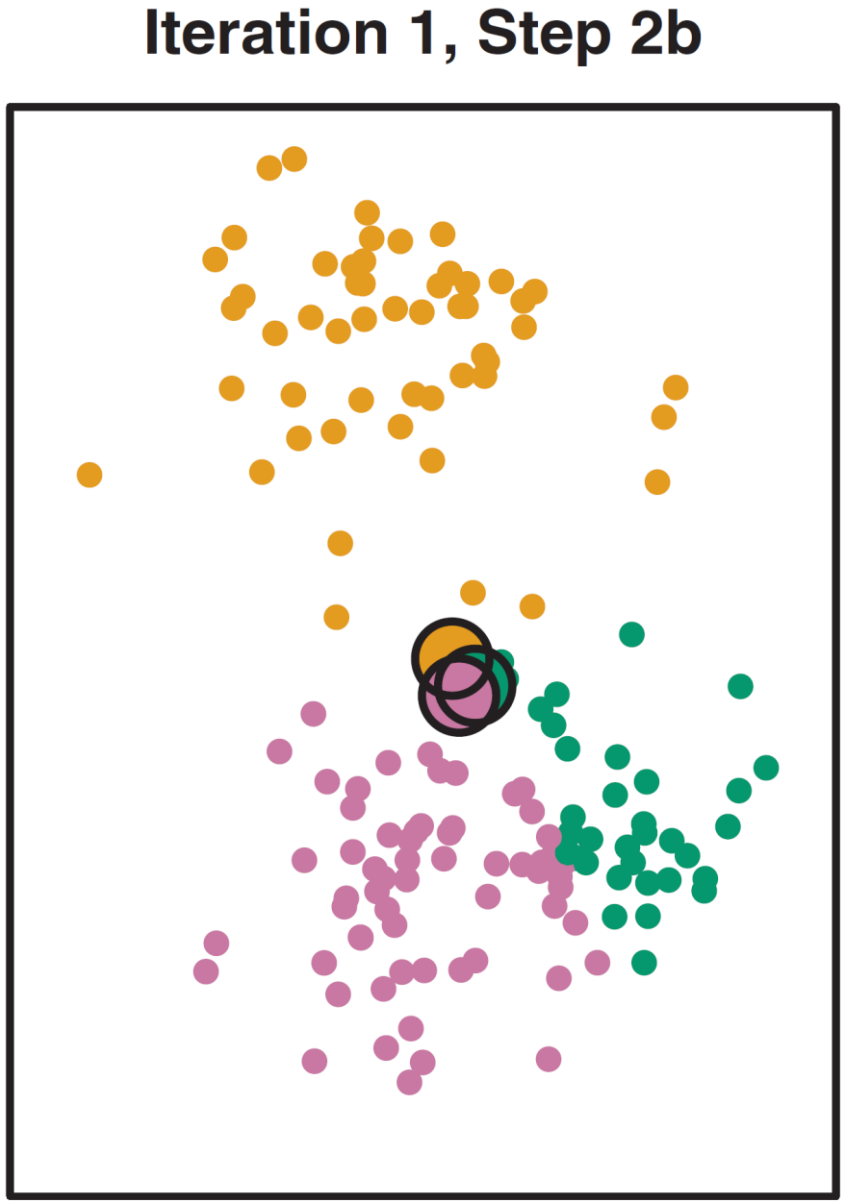
Compute the cluster centroids, which are the mean positions in p-dimensional space across the set of observations within each cluster.



Adapted from James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Illustration of K-means clustering with $K = 3$.

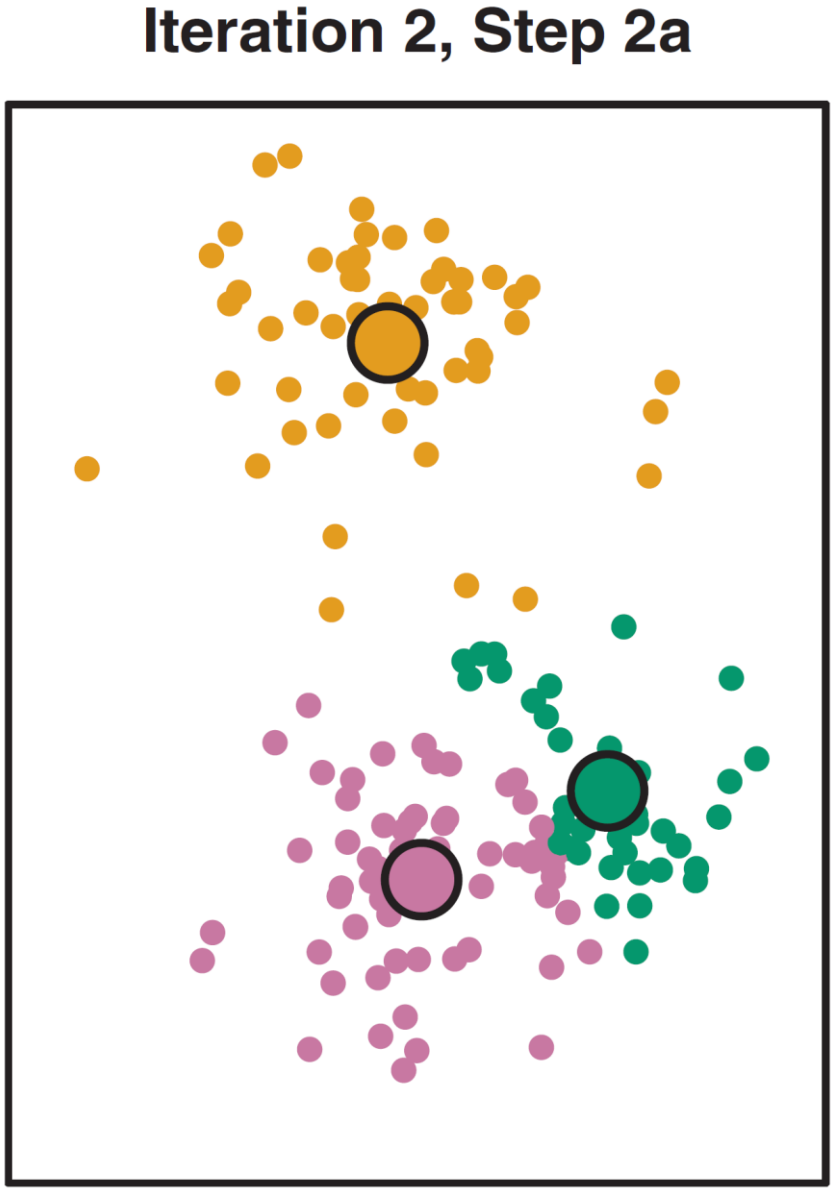
Update the cluster labels by assigning each observation to the cluster with the closest cluster centroid.



Adapted from James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Illustration of K-means clustering with $K = 3$.

Compute the cluster centroids, which are the mean positions in p-dimensional space across the set of observations within each cluster.

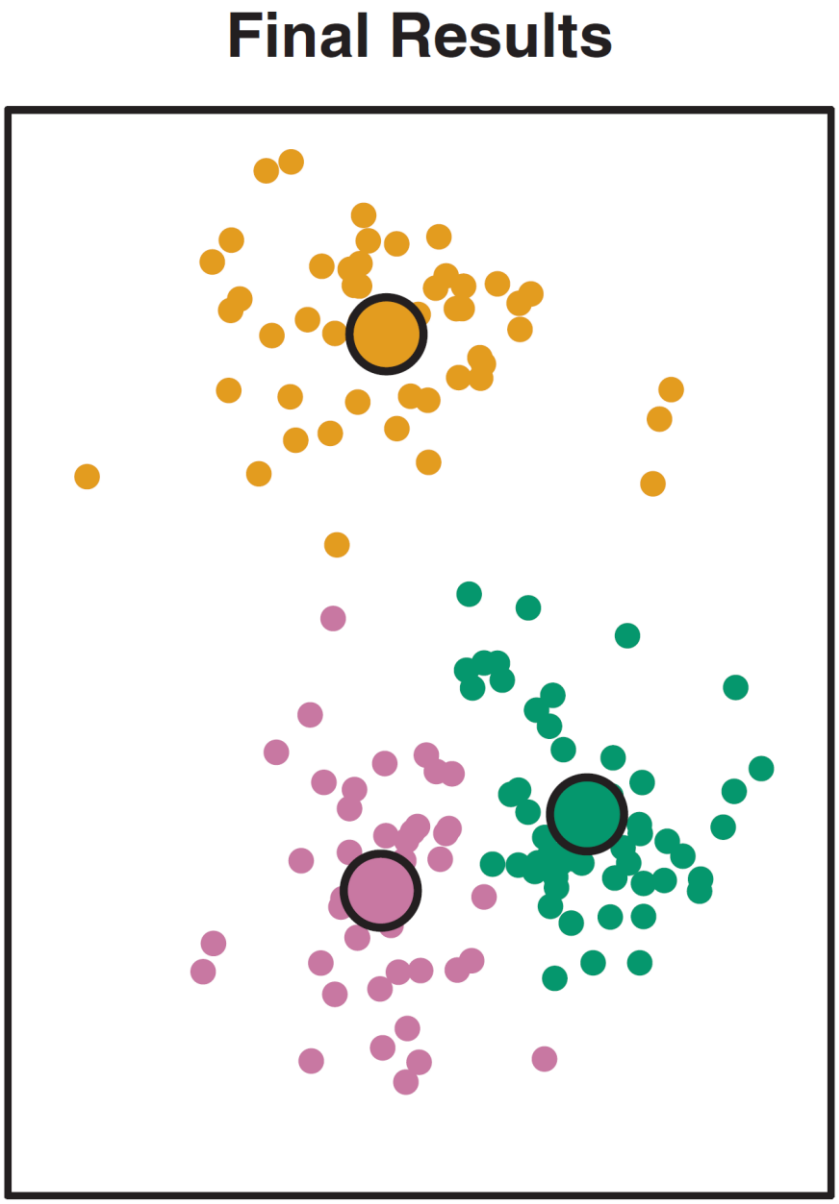


Adapted from James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Illustration of K-means clustering with $K = 3$.

Next, update the cluster labels by assigning each observation to the cluster with the closest cluster centroid.

Keep iterating until no observations change clusters.



Adapted from James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Algorithm for K-means clustering

Step 1: Assign a number (cluster label), from 1 to K, uniformly at random to each observation, with the constraint that each of the K labels is assigned to at least one observation.

Step 2: Iterate until the cluster assignments stop changing:

2a: For each of the K clusters, compute the cluster centroid, where the centroid for cluster k, $k = 1,2, \ldots, K$, is the mean p-dimensional feature vector across all the observations in cluster k.

2b: Assign each observation to the cluster whose centroid is closest based on the Euclidean distance.

K-means clustering his hard prototype clustering

K-means clustering is a prototype clustering approach for partitioning a set of observations into K distinct clusters.

Denote the set of observations in cluster k, $k = 1,2, \ldots, K$, by C_k.

Property 1: The set of clusters contain all observations, such that each of the N input observations will fall into one of the K clusters

Property 2: No observation belongs to more than one cluster.

Property 2 is known as **hard clustering**.

The goal of K-means clustering is to identify a set of “good” clusters, where a good cluster is one with smallest possible **within-cluster variation**.

A more formal description of K-means clustering

The goal of K-means clustering is to identify a set of "good" clusters, where a good cluster is one with smallest possible **within-cluster variation**, which we denote by $W(C_k)$ for cluster k and define as

$$W(C_k) = \frac{1}{|C_k|} \sum_{i \in C_k} \sum_{i' \in C_k} \sum_{j=1}^{p} \left(x_{ij} - x_{i'j}\right)^2$$

where $|C_k|$ is the cardinality of the set C_k, which is the number of observations in C_k.

That is, K-means clustering seeks to solve the problem

$$\underset{C_1, C_2, \ldots, C_K}{\arg\min} \sum_{k=1}^{K} W(C_k)$$

where we are attempting to find the set of K clusters $\{C_1, C_2, \ldots, C_K\}$ with the smallest total within-cluster variation.

A more formal description of K-means clustering

The prototype for cluster k, $k = 1,2,\ldots,K$, is the cluster centroid

$$m_k = \frac{1}{|C_k|} \sum_{i \in C_k} x_i$$

the mean p-dimensional feature vector for all the observations in C_k.

We can rewrite this cluster centroid as

$$m_k = \begin{bmatrix} m_{k1} \\ m_{k2} \\ \vdots \\ m_{kp} \end{bmatrix}$$

where the mean of feature j, $j = 1,2,\ldots,p$, in cluster k is

$$m_{kj} = \frac{1}{|C_k|} \sum_{i \in C_k} x_{ij}$$

A more formal description of K-means clustering

We rewrite the within-cluster variation as

$$W(C_k) = \frac{1}{|C_k|} \sum_{i \in C_k} \sum_{i' \in C_k} \sum_{j=1}^{p} \left(x_{ij} - x_{i'j}\right)^2$$

$$= \frac{1}{|C_k|} \sum_{i \in C_k} \sum_{i' \in C_k} \|x_i - x_{i'}\|_2^2$$

$$= 2 \sum_{i \in C_k} \|x_i - m_k\|_2^2$$

using the squared Euclidean distance between each observation, or between each observation and the cluster centroid.

K-means clustering algorithm only finds local optimum

The K-means clustering algorithm only finds local optima, and these optima are dependent on the initial cluster assignments, and therefore initial centroid locations.

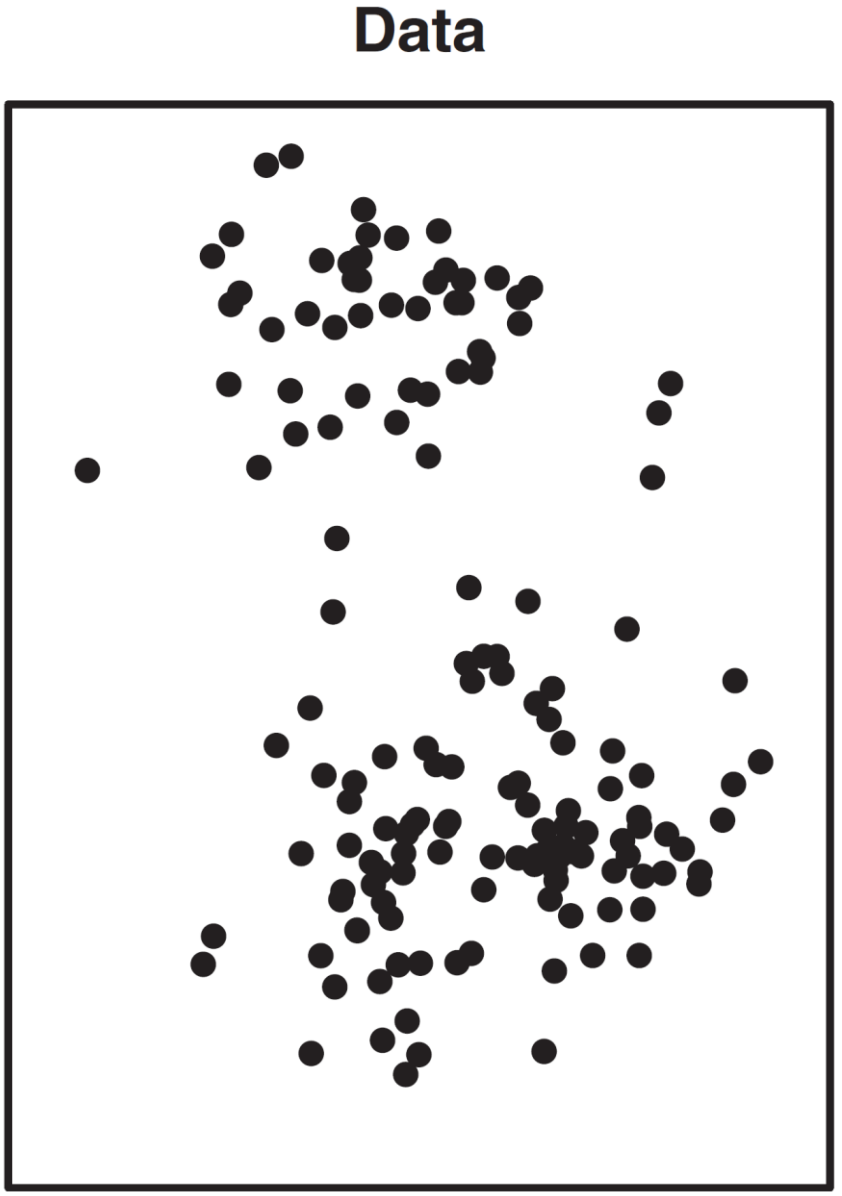
Therefore, the algorithm should be run many times with many random starting points for cluster labels.

The best cluster assignment will be the one that has the minimum total within-cluster variation, which is the quantity that we are trying to minimize in the first place.

Illustration of different optima with $K = 3$.

For illustration, we consider again a set of observations in two-dimensional space (right).

We will assume that there are $K = 3$ clusters, and employ random cluster assignments to examine how different local optima can be obtained.

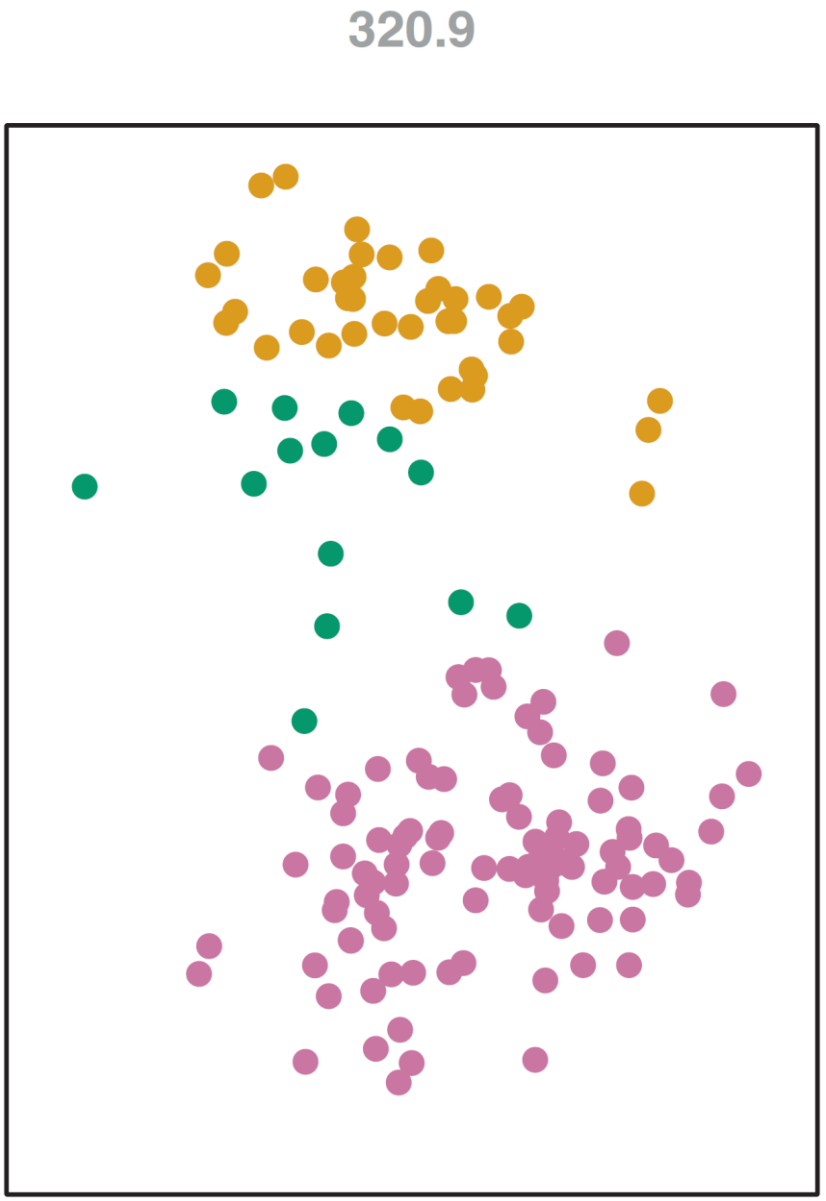


Adapted from James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Illustration of different optima with $K = 3$

The final cluster assignment to the right has total within-cluster variance of 320.9.

However, this may not be the set of clusters.

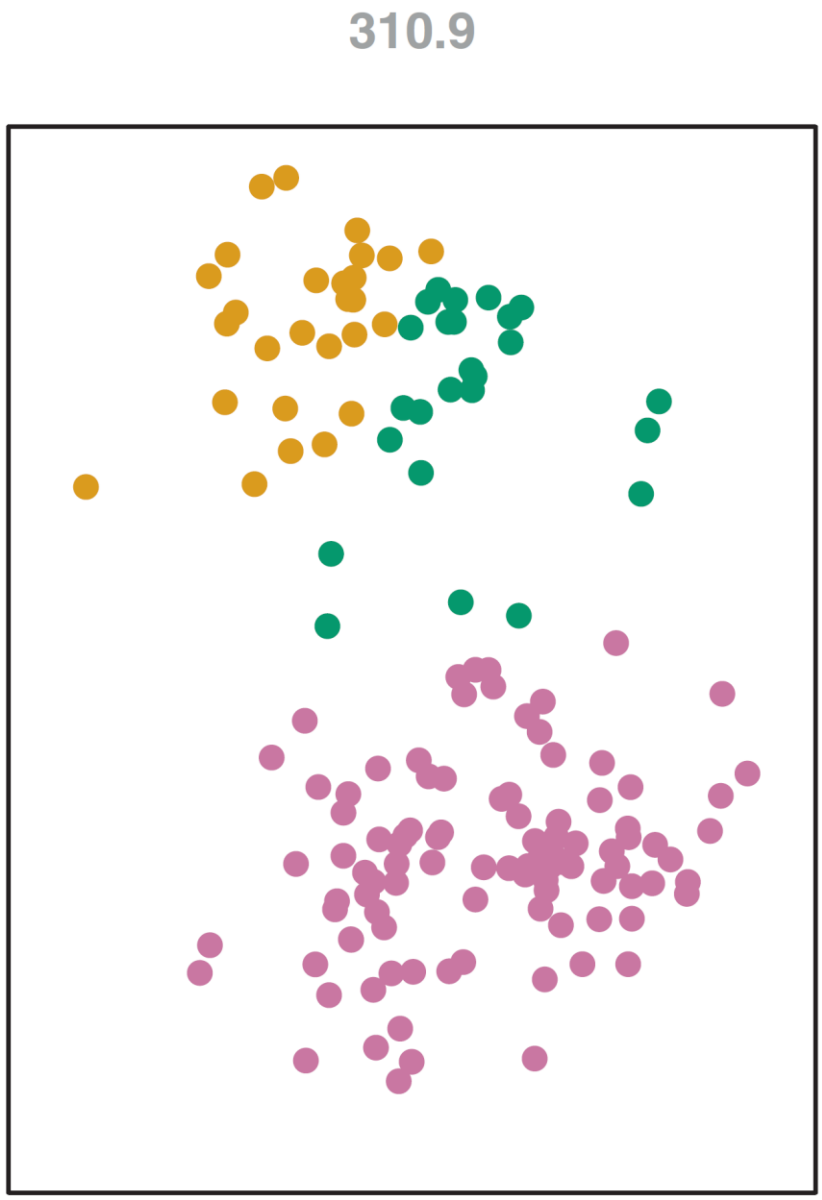


Adapted from James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Illustration of different optima with $K = 3$

With a different random start, the total within-cluster variance reduces to 310.9.

We would choose this clustering result over the other one, as it has smaller total within-cluster variation.

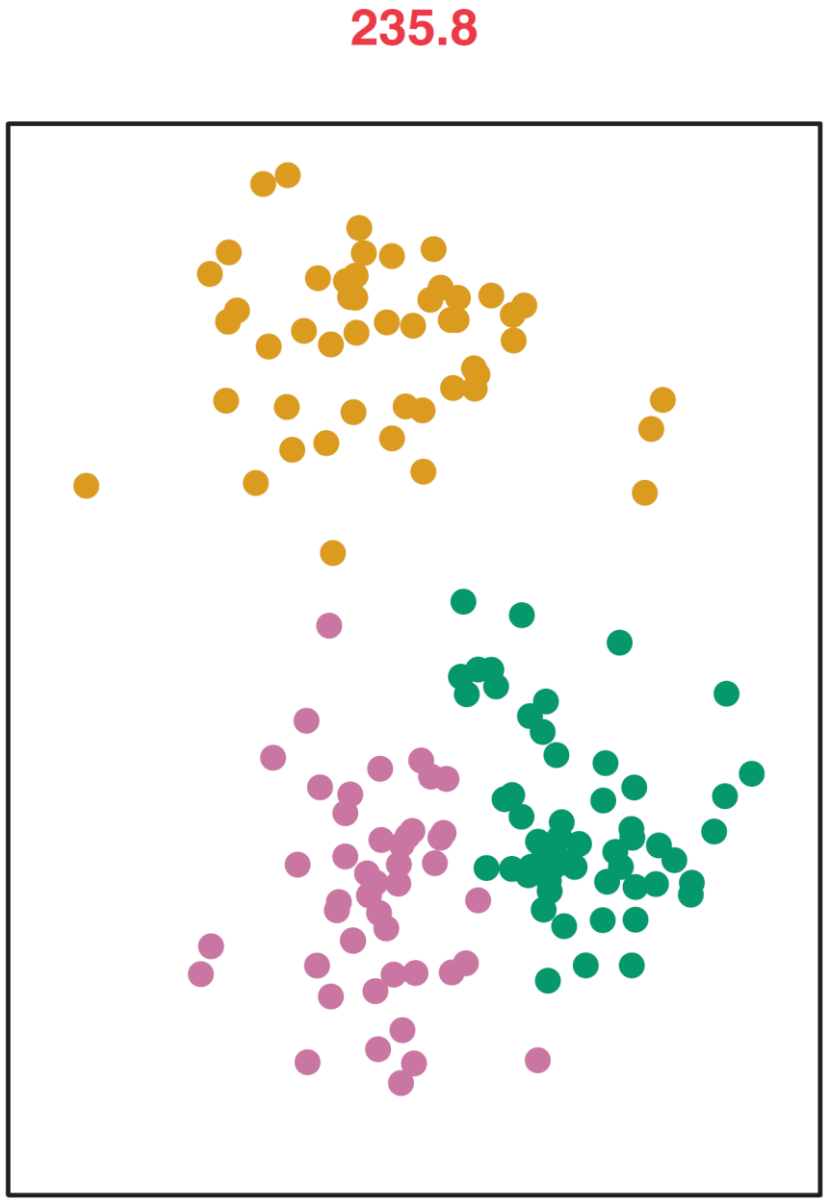


Adapted from James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Illustration of different optima with $K = 3$

Finally, another random start gives an even smaller total within-cluster variance of 235.8.

We would therefore choose this clustering result over the other two, as it has smallest total within-cluster variation.



Adapted from James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

K-means cluster labels are arbitrary

The cluster labels $1,2,\ldots,K$ are arbitrary.

That is, when running K-means clustering, it is possible to have an observation assigned to cluster k in one run and then assigned to cluster $\ell \neq k$ in another run.

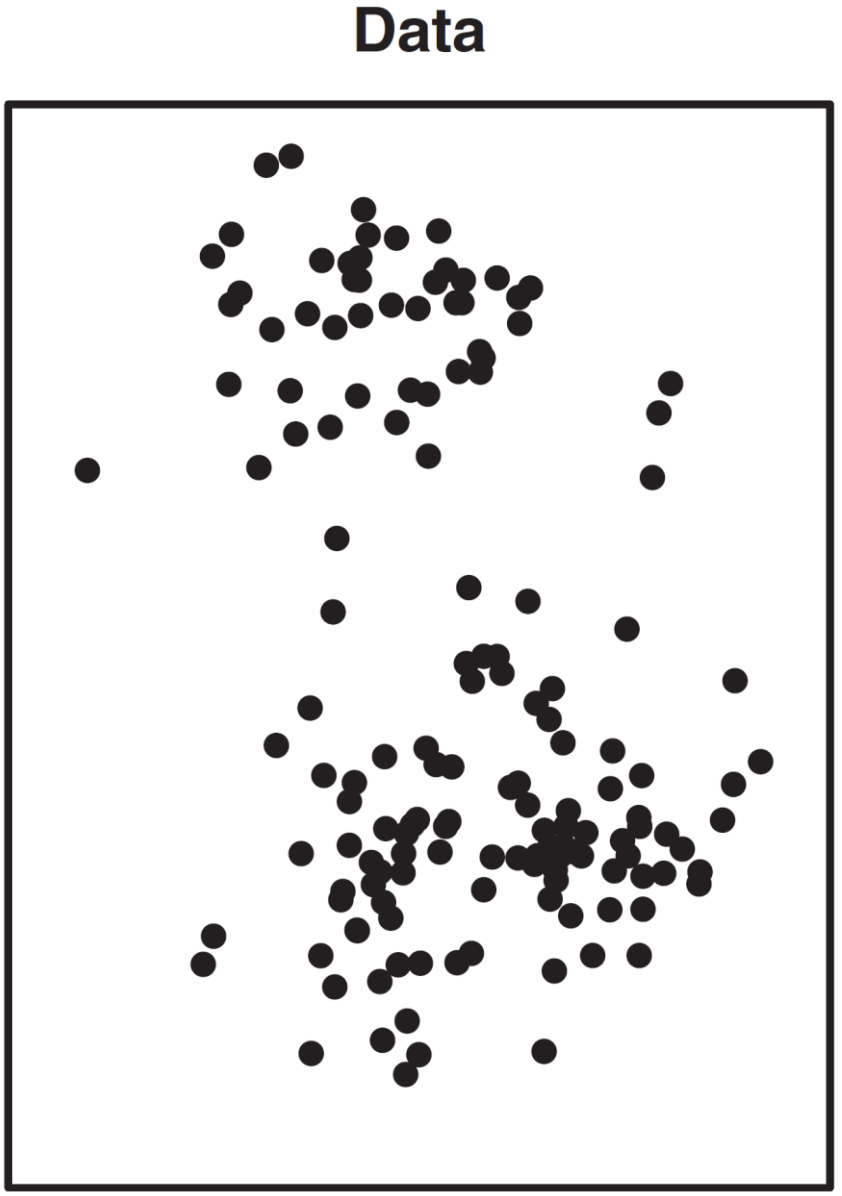
This is the case even if the total within-cluster variation is identical across runs, and also even when the partition of the set of observations into K non-overlapping sets is the same.

Therefore, meaning should not be placed on specific cluster labels, and instead we place meaning on the fact that a set of observations all fall within the same cluster, regardless of label.

Illustration of different cluster labels with $K = 3$

For illustration, we consider again a set of observations in two-dimensional space (right).

We will assume that there are $K = 3$ clusters, and employ random cluster assignments to examine how different cluster labels can be obtained, even with the same total within-cluster variation and the same partition of observations into K non-overlapping sets.

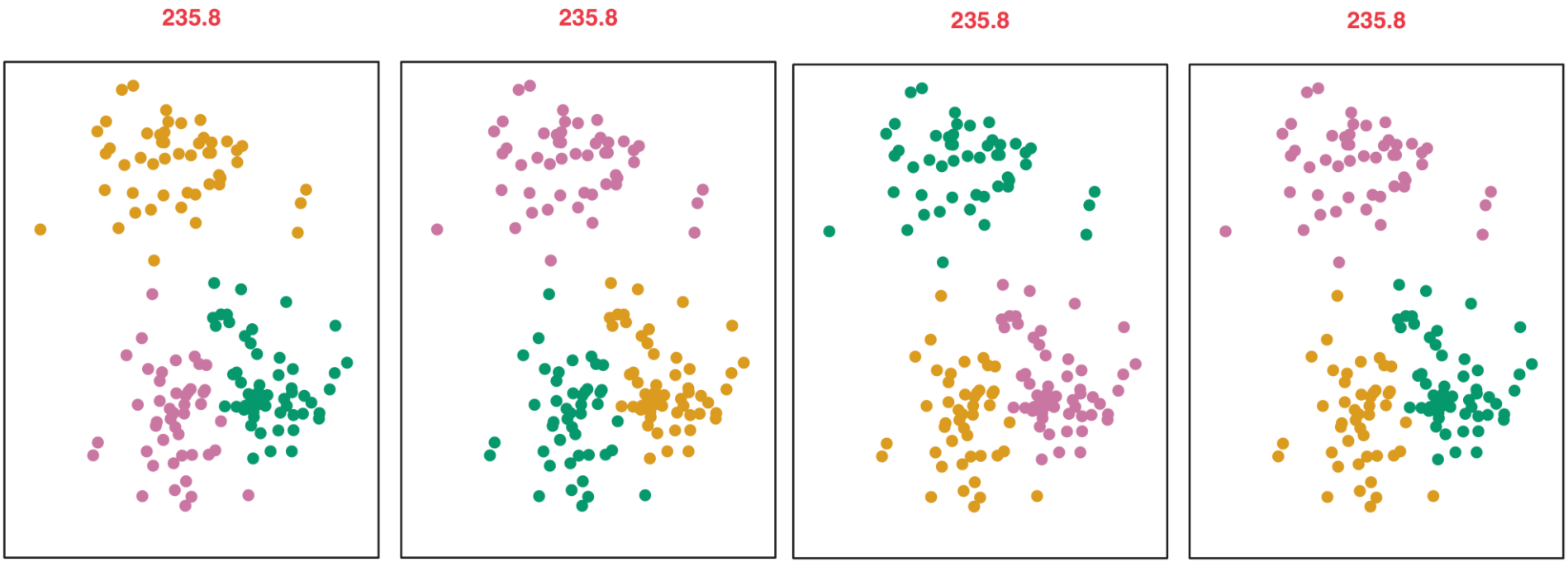


Adapted from James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Illustration of different cluster labels with $K = 3$

Below, cluster membership is identical across runs that had the same total within-cluster variance.

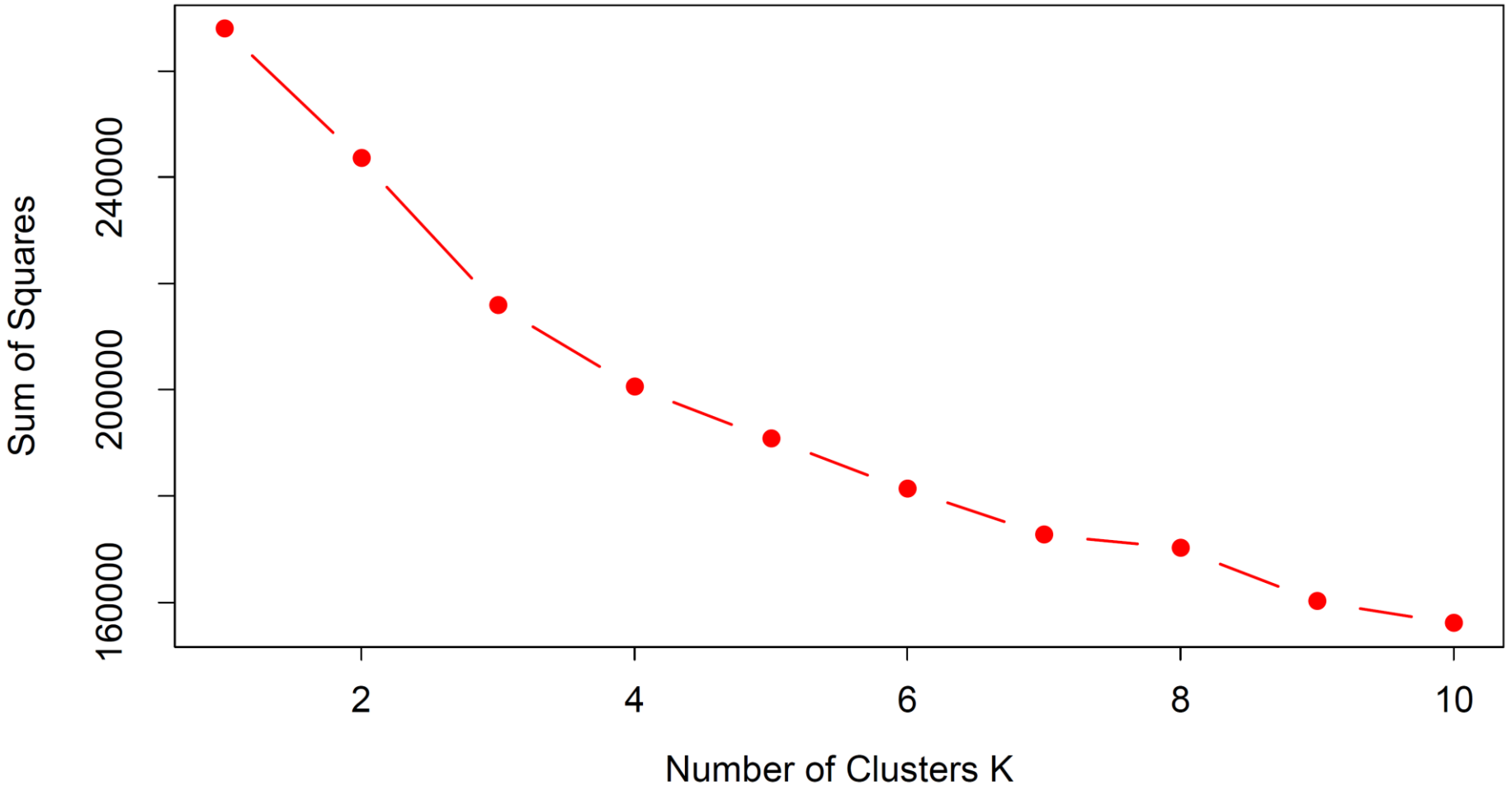
However, the cluster labels are varied across runs.



Adapted from James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Choosing the number of clusters using a scree plot

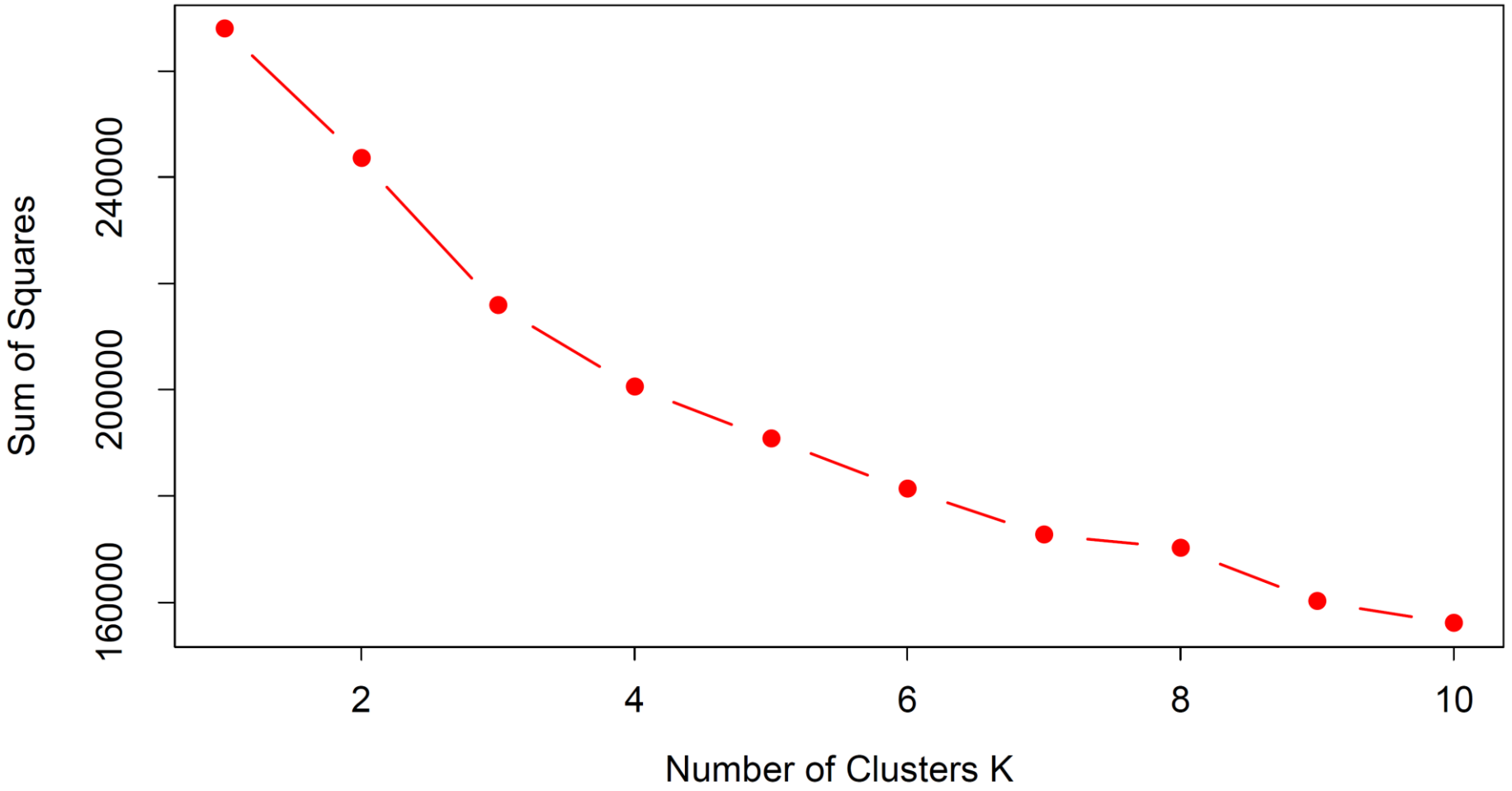
The number of clusters can be chosen with a scree plot, where we plot the number of clusters K on the x-axis, and the total within-cluster variance on the y-axis.



Hastie *et al* (2009) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd Ed.

Choosing the number of clusters using a scree plot

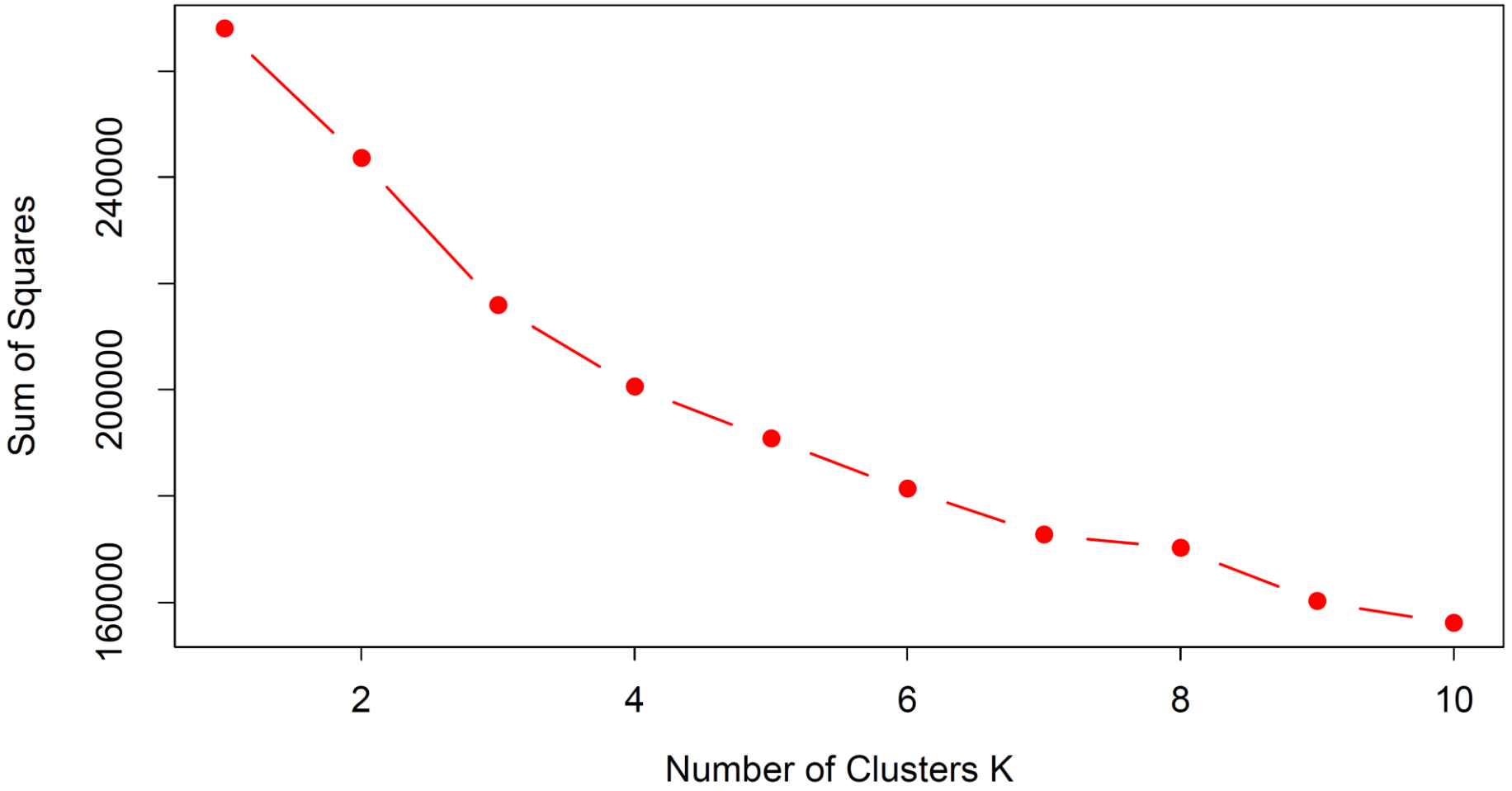
Due to the optimization criterion, it is expected that the total within-cluster variation will reduce with increasing K (consider the extreme example of $K = N$, for which the total within-cluster variance is 0).



Hastie *et al* (2009) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd Ed.

Choosing the number of clusters using a scree plot

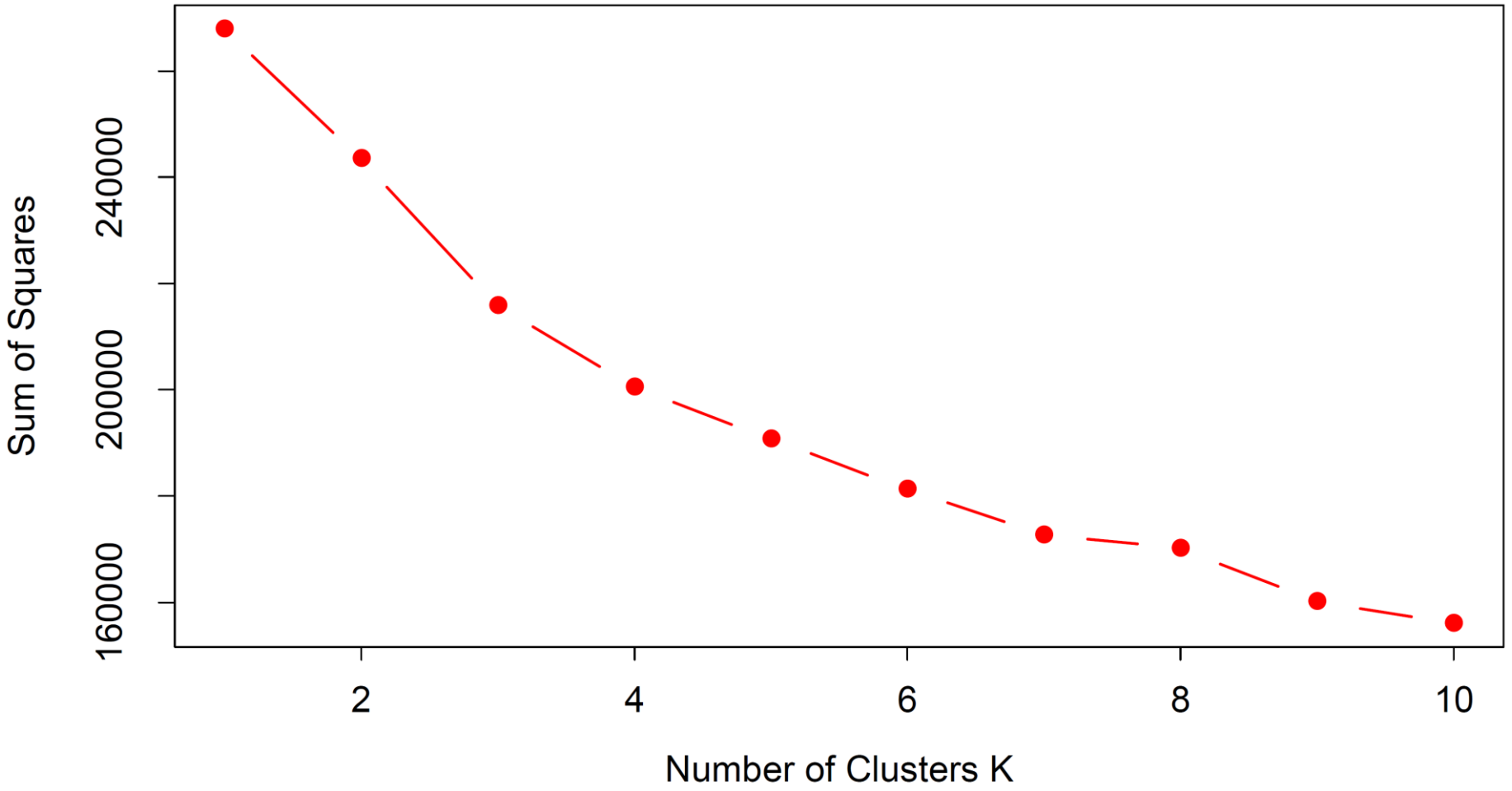
The idea is to find a kink (or elbow) within the curve, at which the total within-cluster variation begins to decay more slowly, which is not easy to observe in this plot.



Hastie *et al* (2009) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd Ed.

Choosing the number of clusters using a scree plot

Moreover, using a scree plot would yield erroneous results if the true number of clusters is $K = 1$, as you cannot see an elbow in the graph at $K = 1$.



Hastie *et al* (2009) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd Ed.

Hierarchical clustering

A disadvantage of prototype clustering (*e.g.*, K-means) approaches is that they require the number of clusters K to be pre-specified.

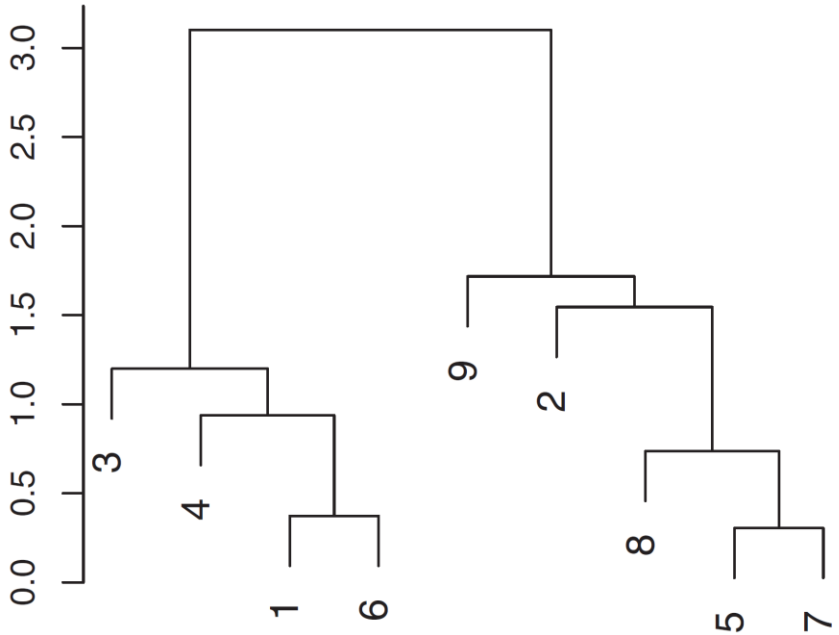
Hierarchical clustering is an alternative approach that does not commit to a particular choice of K, and instead explores all K from 1 to N, where N is the number of observations.

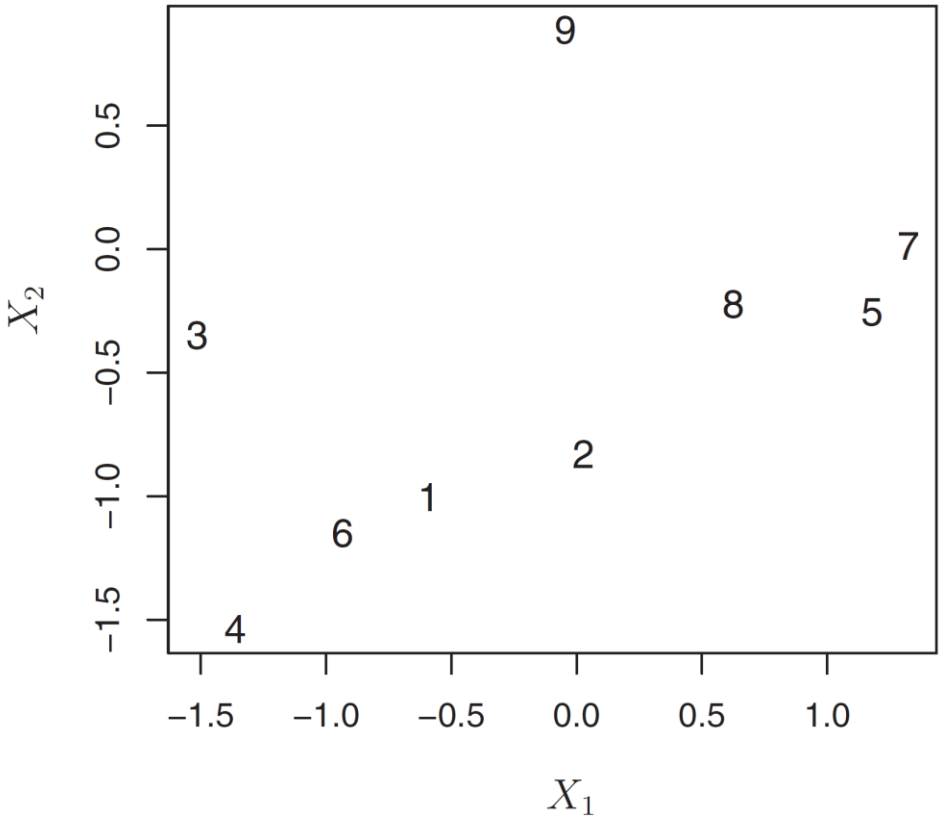
Moreover, results from hierarchical clustering are easy to visualize as they are represented by a tree known as a **dendrogram**.

We will focus on **bottom-up** (**agglomerative**) clustering algorithms, that build a tree (dendrogram) from the tips up to the root, rather than the **top-down** (**divisive**) clustering approach that we saw with decision trees, which built a tree from the root to the tips.

Illustration and interpretation of a dendrogram

Observations 5 and 7 are similar to each other, as are observations 1 and 6, as each of these pairs are the first observations to be fused.



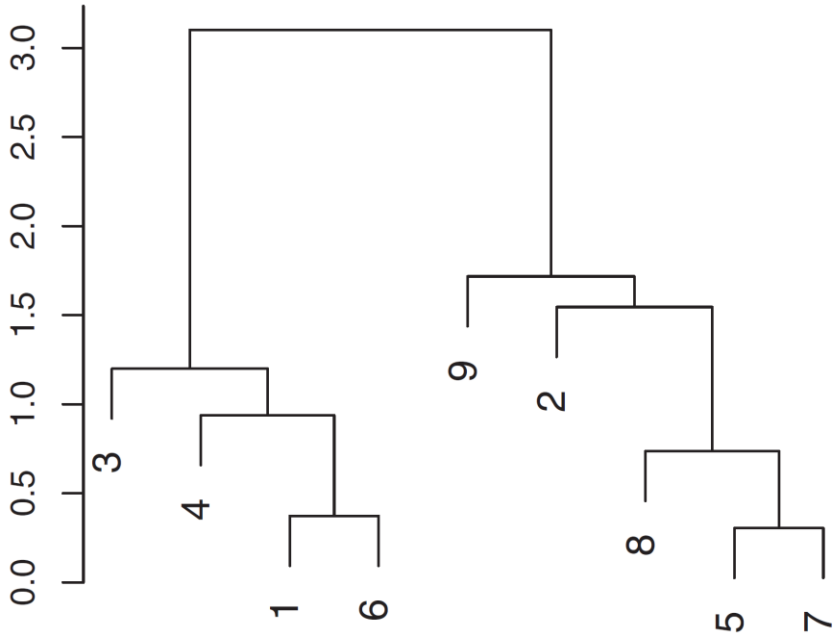


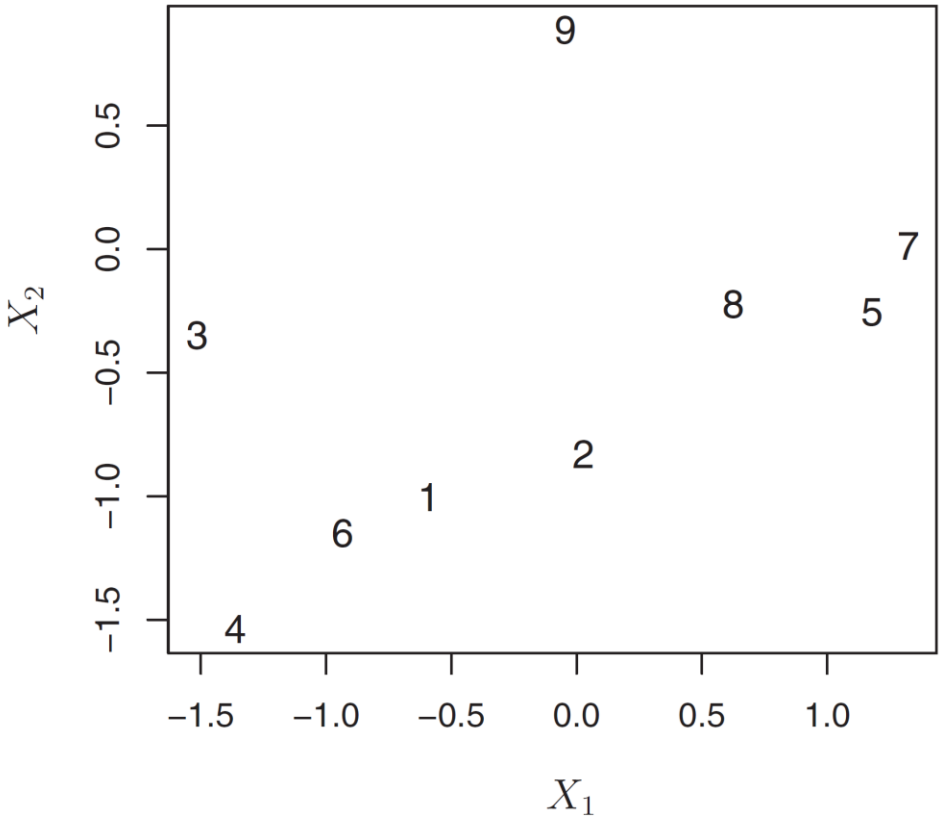
Adapted from James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Illustration and interpretation of a dendrogram

Observation 9 is no more similar to observation 2 than it is to observations 8, 5, and 7.

That is, only *vertical distance* (not horizontal distance) matters.



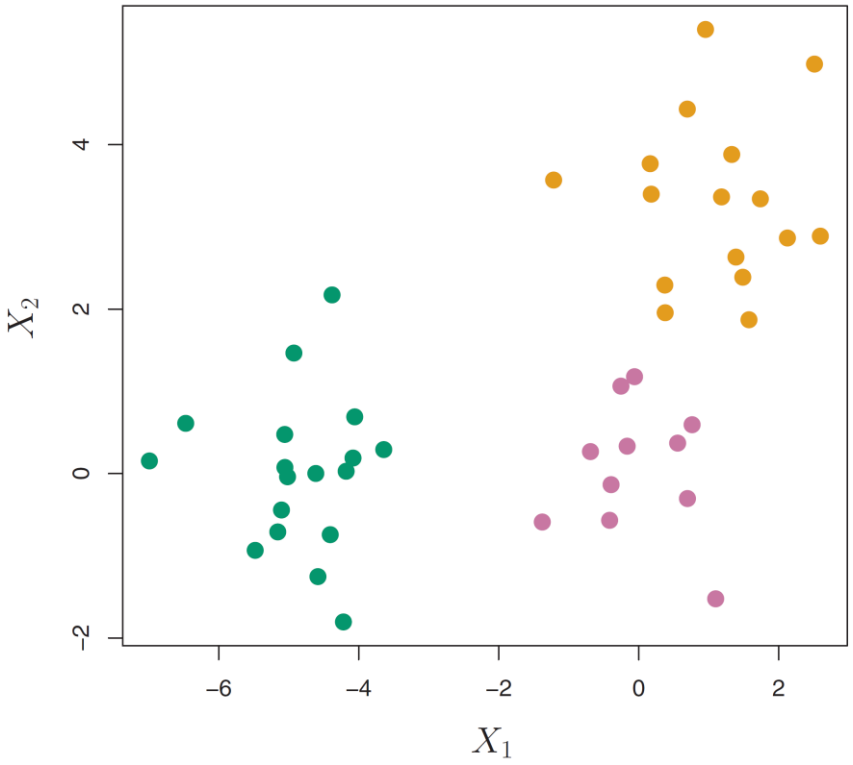


Adapted from James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Identifying clusters in a dendrogram

Consider the following set of $N = 45$ observations with $p = 2$ features that were generated based on a $K = 3$ class model, with the classes coded as orange, green, and purple.

Suppose, instead, that the observations were without class labels, and that we wanted to perform hierarchical clustering to learn the classes.

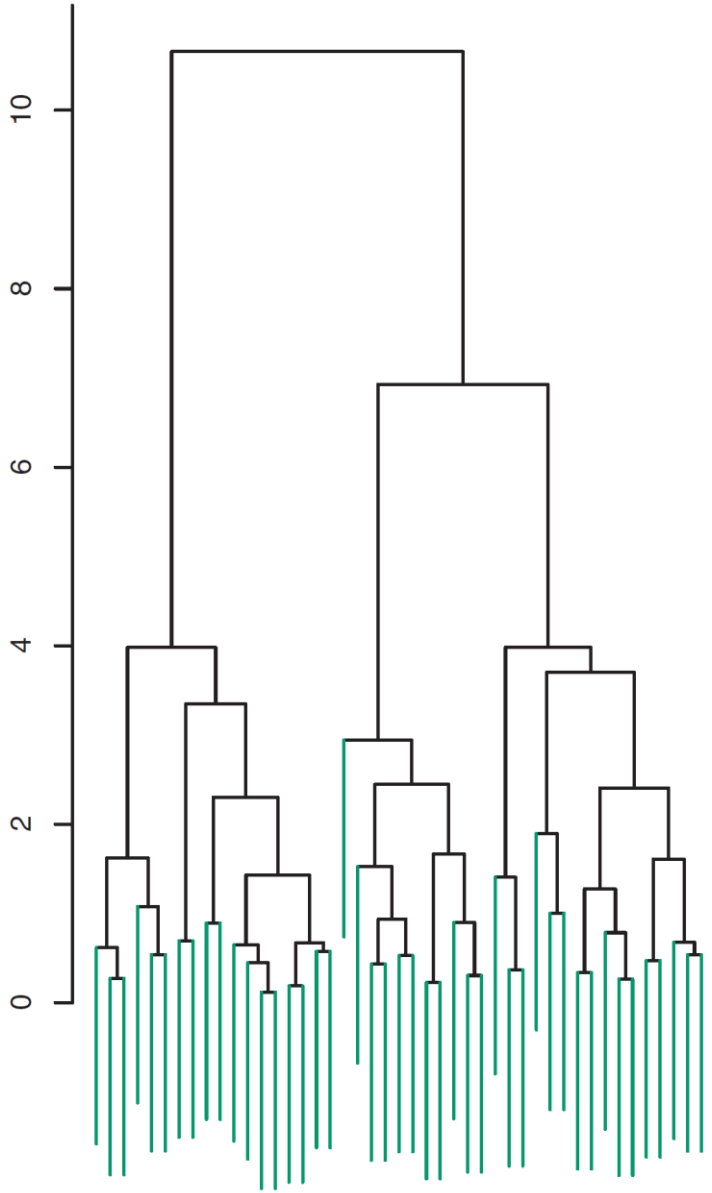


James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Identifying clusters in a dendrogram

Hierarchical clustering refers to the idea that clusters can be identified by making horizontal cuts along the dendrogram, and all observations descending from the same cut branch are in the a cluster.

If we cut above the root of the dendrogram, then we get $K = 1$ cluster corresponding to the color green, which includes the entire set of N observations.

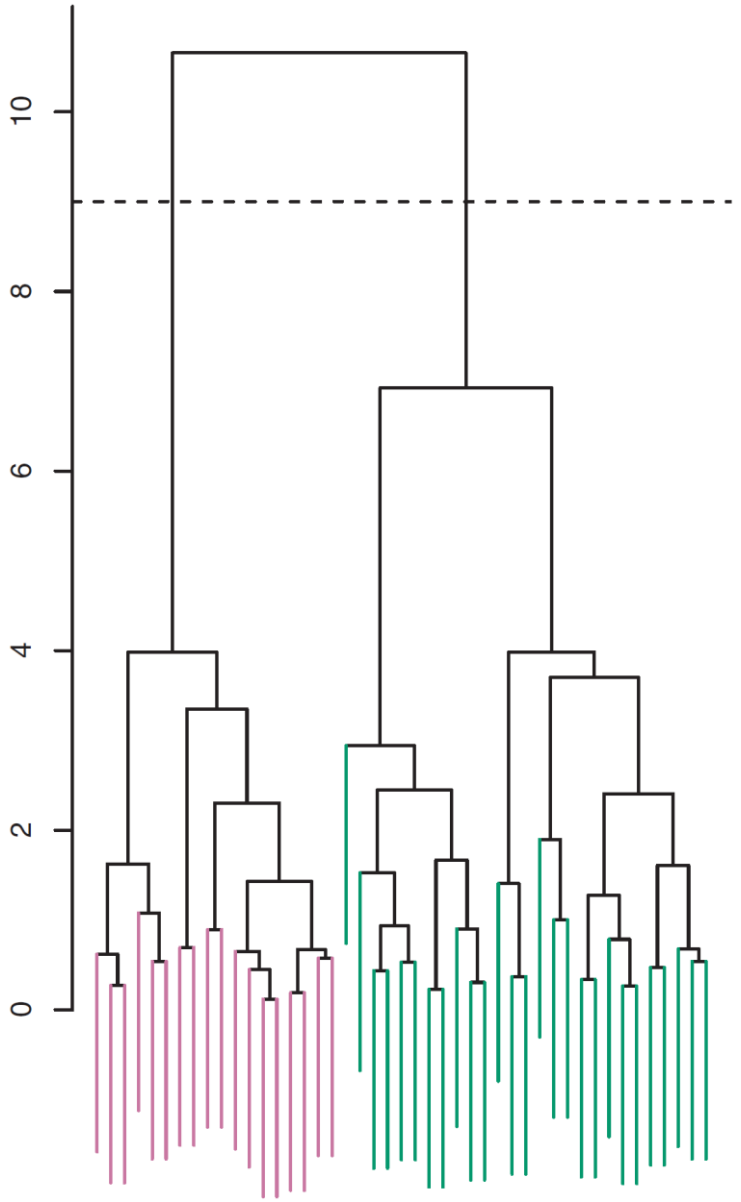


Adapted from James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Identifying clusters in a dendrogram

Cutting instead at a height of 9 yields $K = 2$ clusters.

One cluster corresponds to the purple observations, and the other cluster corresponds to the green observations.

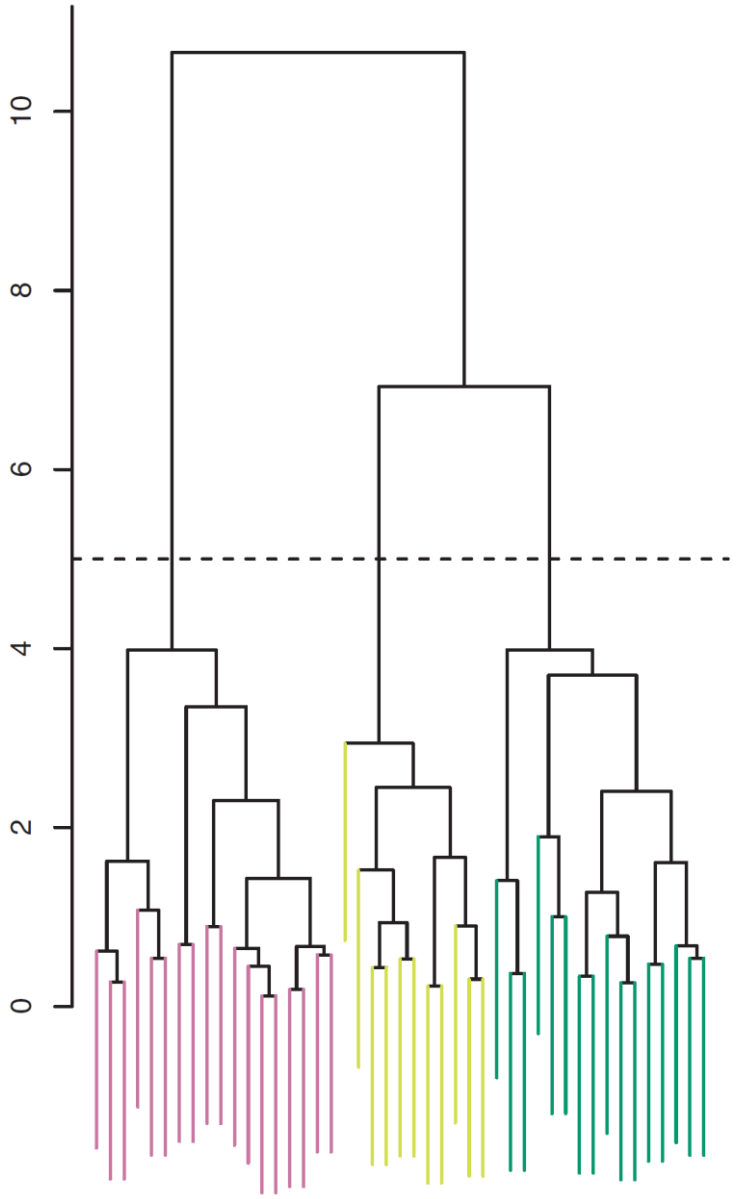


Adapted from James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Identifying clusters in a dendrogram

Cutting instead at a height of 5 yields $K = 3$ clusters.

One cluster corresponds to the purple observations, another cluster the orange observations, and the final cluster the green observations.



Adapted from James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Common dissimilarities used in hierarchical clustering

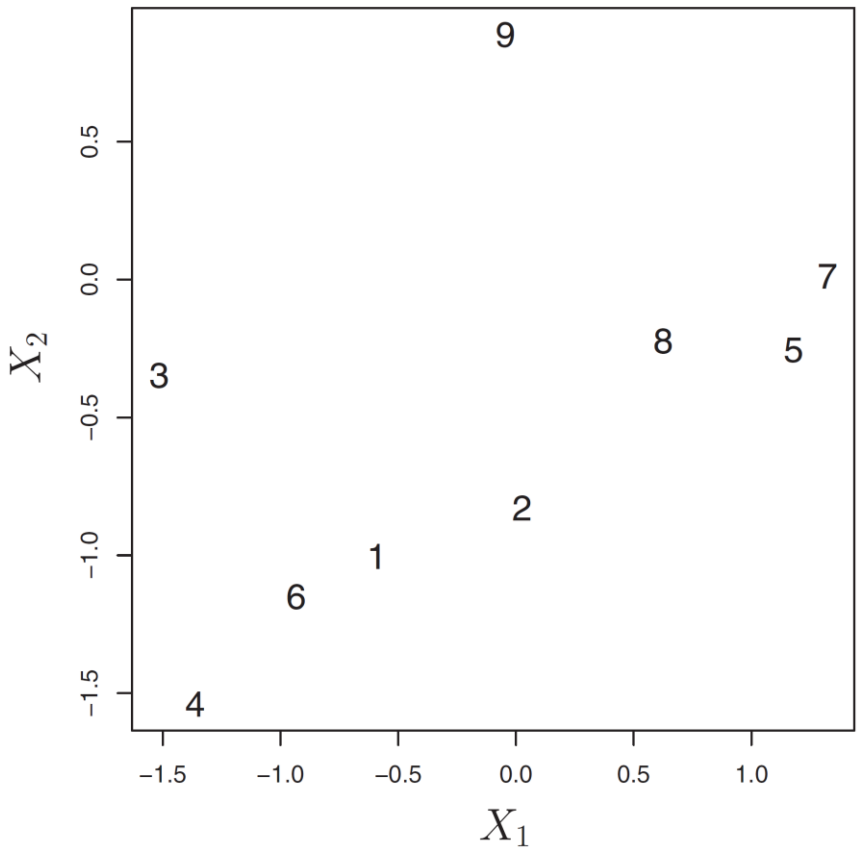
Linkage	*Description*
Complete	Maximal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *largest* of these dissimilarities.
Single	Minimal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *smallest* of these dissimilarities. Single linkage can result in extended, trailing clusters in which single observations are fused one-at-a-time.
Average	Mean intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *average* of these dissimilarities.
Centroid	Dissimilarity between the centroid for cluster A (a mean vector of length p) and the centroid for cluster B. Centroid linkage can result in undesirable *inversions*.

James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Illustration of complete linkage hierarchical clustering

We start out with 9 clusters and compute the pairwise Euclidean distance between each observation between a pair of clusters, and record the largest distance as the dissimilarity between those two clusters.

We then fuse the two clusters with this smallest dissimilarity.

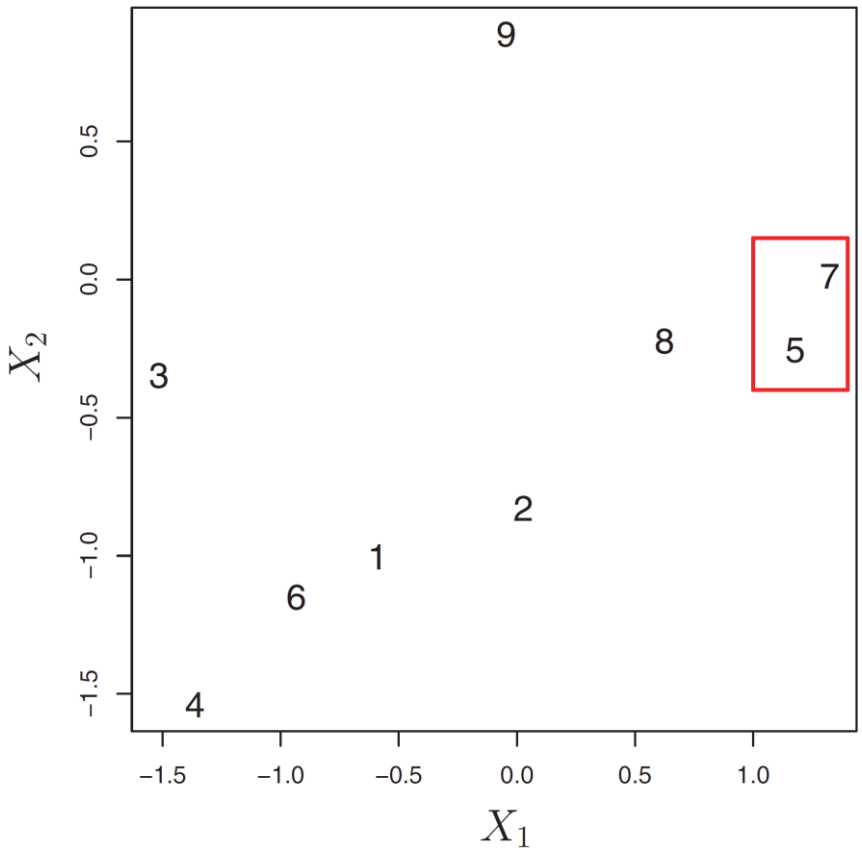


Adapted from James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Illustration of complete linkage hierarchical clustering

The two clusters (observations) with smallest maximum distance are observations 5 and 7.

These two observations are fused into a cluster (red), and we now have 8 clusters.



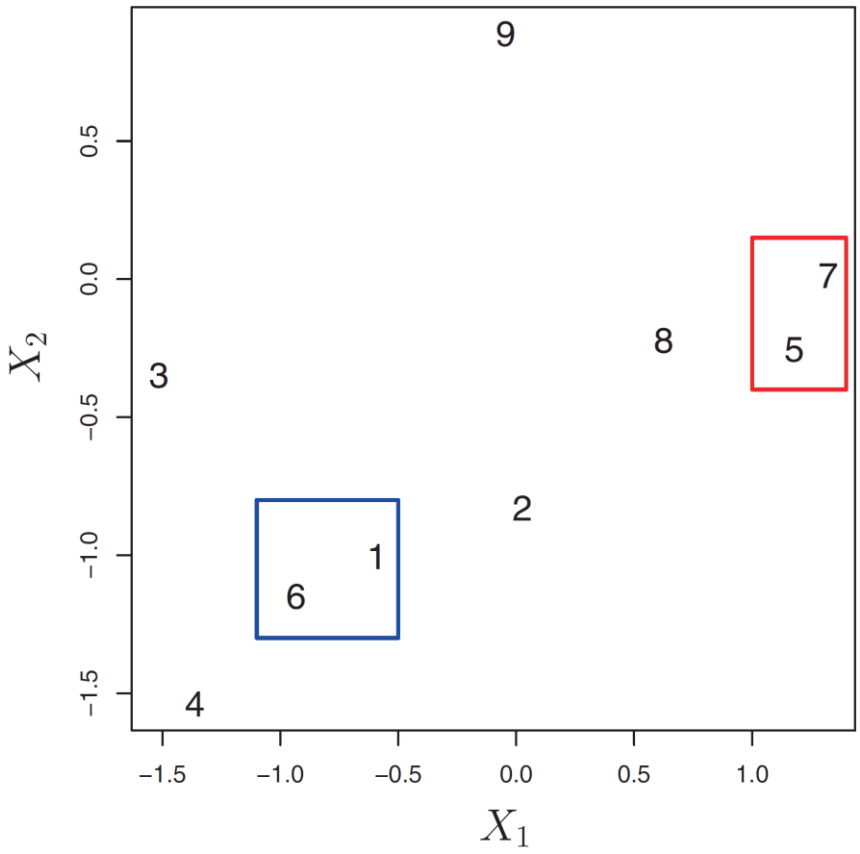
Adapted from James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Illustration of complete linkage hierarchical clustering

We next compute the maximum distance between each pair of the 8 clusters.

We find that the two clusters (observations) with smallest maximum distance are observations 1 and 6.

These two observations are fused into a cluster (blue), and we now have 7 clusters.



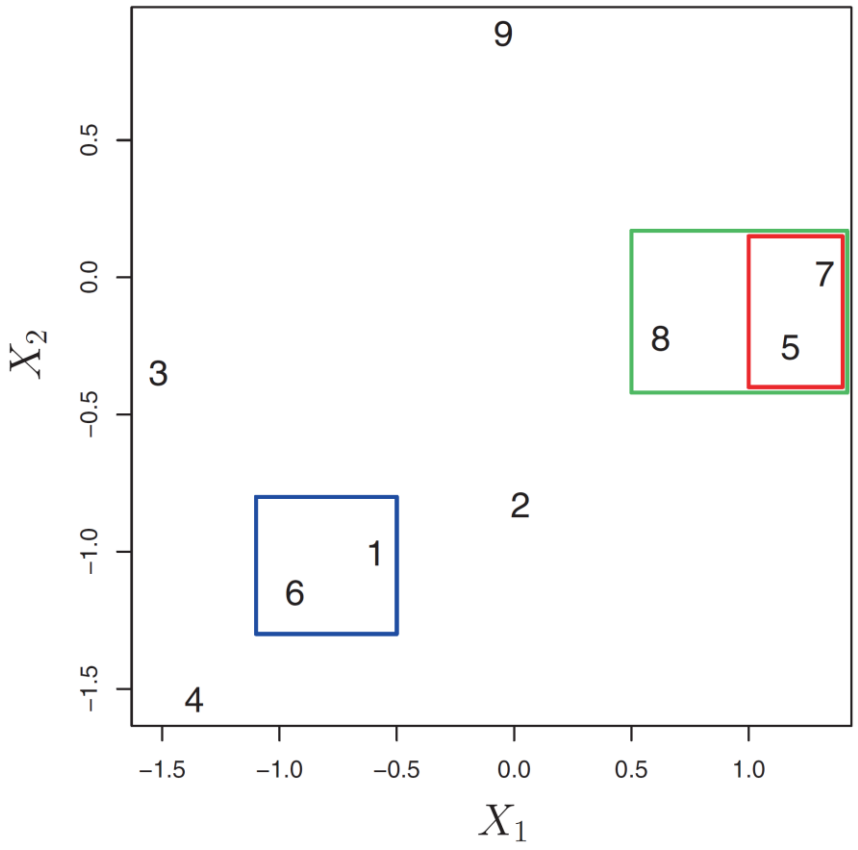
Adapted from James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Illustration of complete linkage hierarchical clustering

We next compute the maximum distance between each pair of the 7 clusters.

We find that the two clusters with smallest maximum distance are observation 8 with the red cluster (containing observations 5 and 7).

Observation 8 and the red cluster are fused into a cluster (green), containing observations 5, 7, and 8, and we now have 6 clusters.



Adapted from James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Algorithm for hierarchical clustering

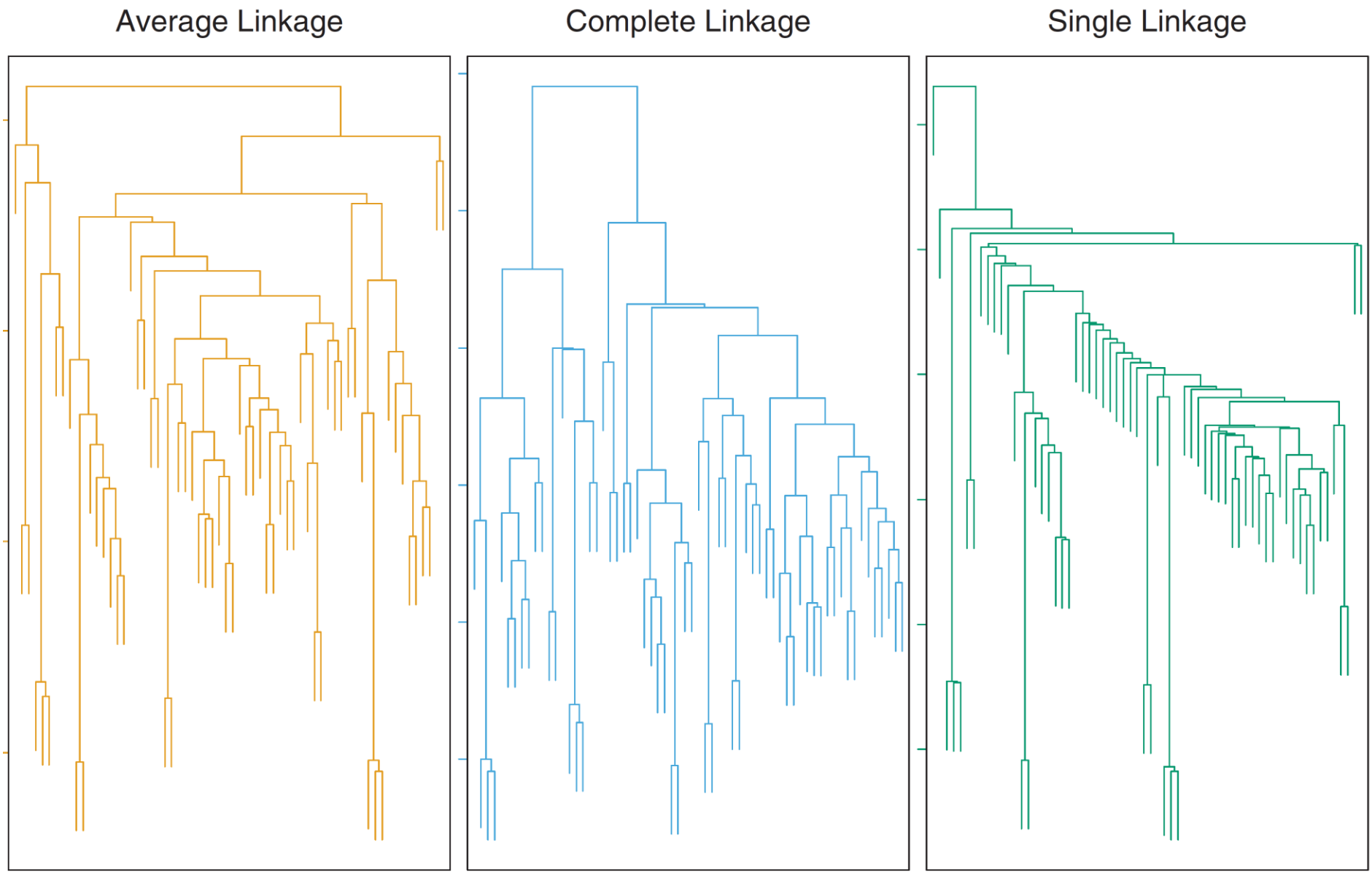
Step 1: Begin with N observations each treated as its own cluster, and a measure (*e.g.*, Euclidean distance) of dissimilarity between each of the $\binom{N}{2} = N(N-1)/2$ pairs of observations.

Step 2: For $i = N, N-1, \ldots, 2$:

2a: Examine all pairwise inter-cluster dissimilarities among the i clusters and identify the pair of clusters that are least dissimilar (most similar). Fuse these 2 clusters. The dissimilarity between 2 clusters indicates the height of the dendrogram at which the fusion should be placed.

2b: Compute the new pairwise inter-cluster dissimilarities among the remaining $i-1$ clusters.

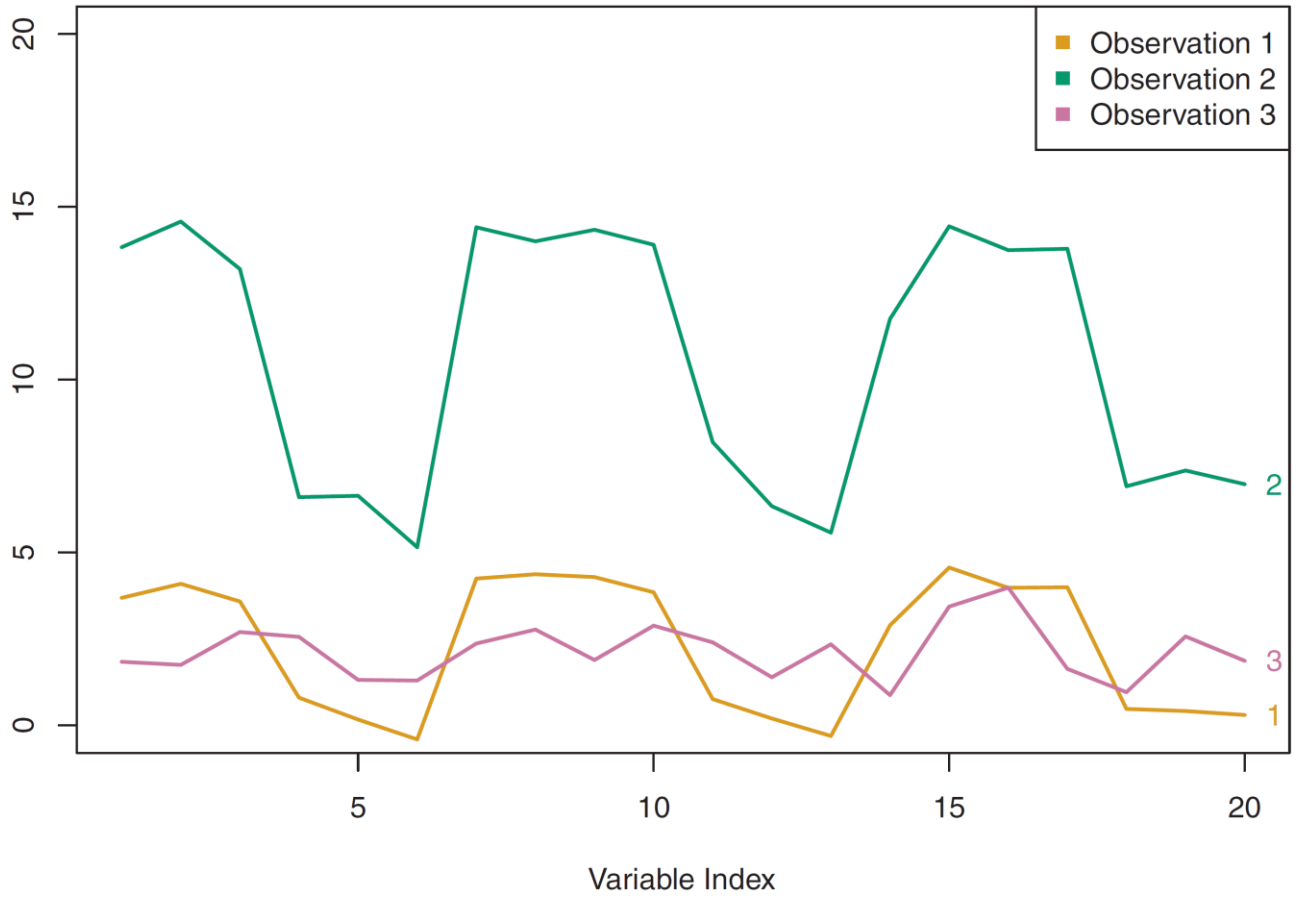
Single linkage returns unbalanced trees from “chaining”



James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Choosing dissimilarity measure

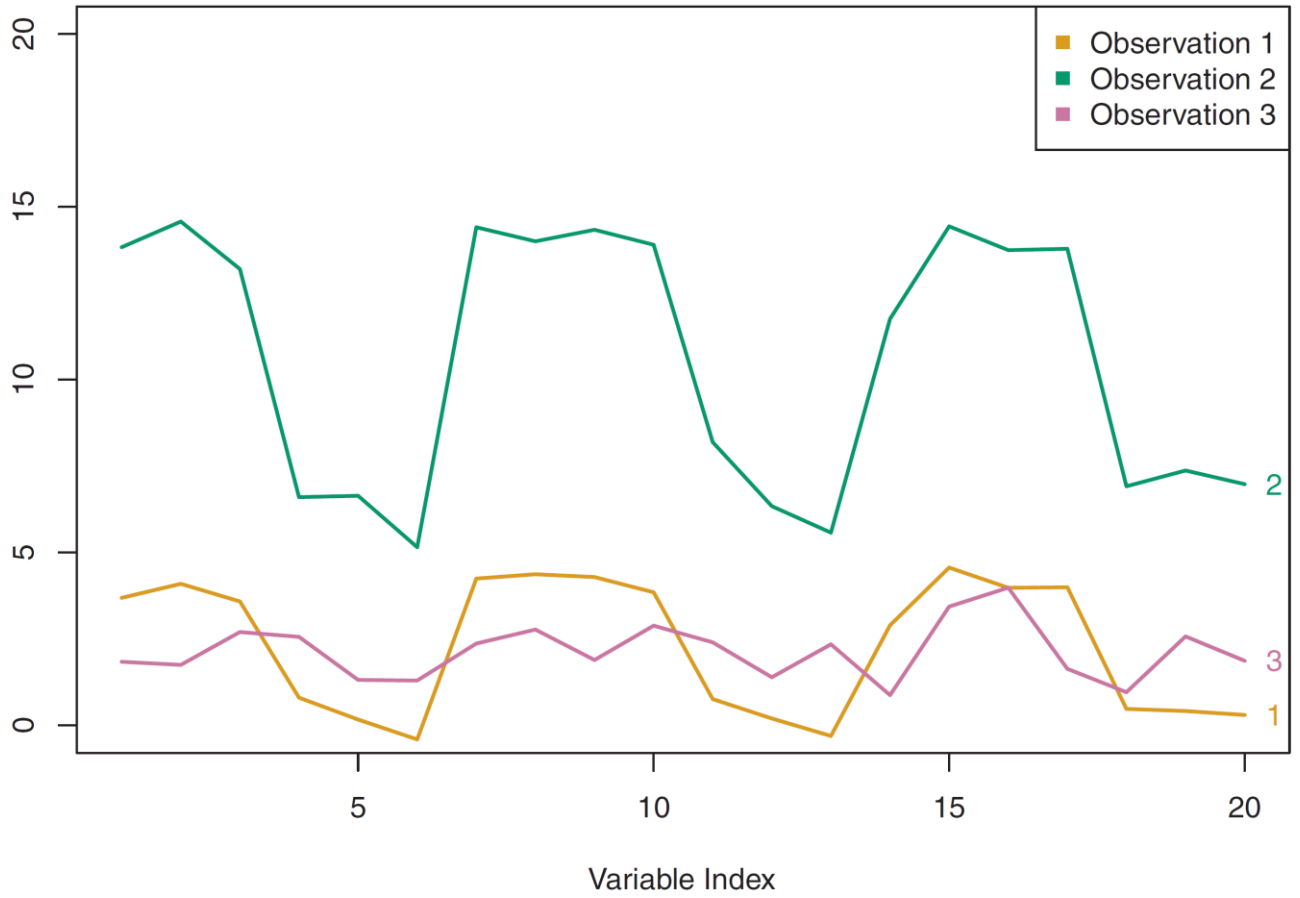
In addition to choosing a linkage (fusion) criterion, it is also important to decide on an appropriate dissimilarity measure between clusters or observations.



James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Choosing dissimilarity measure

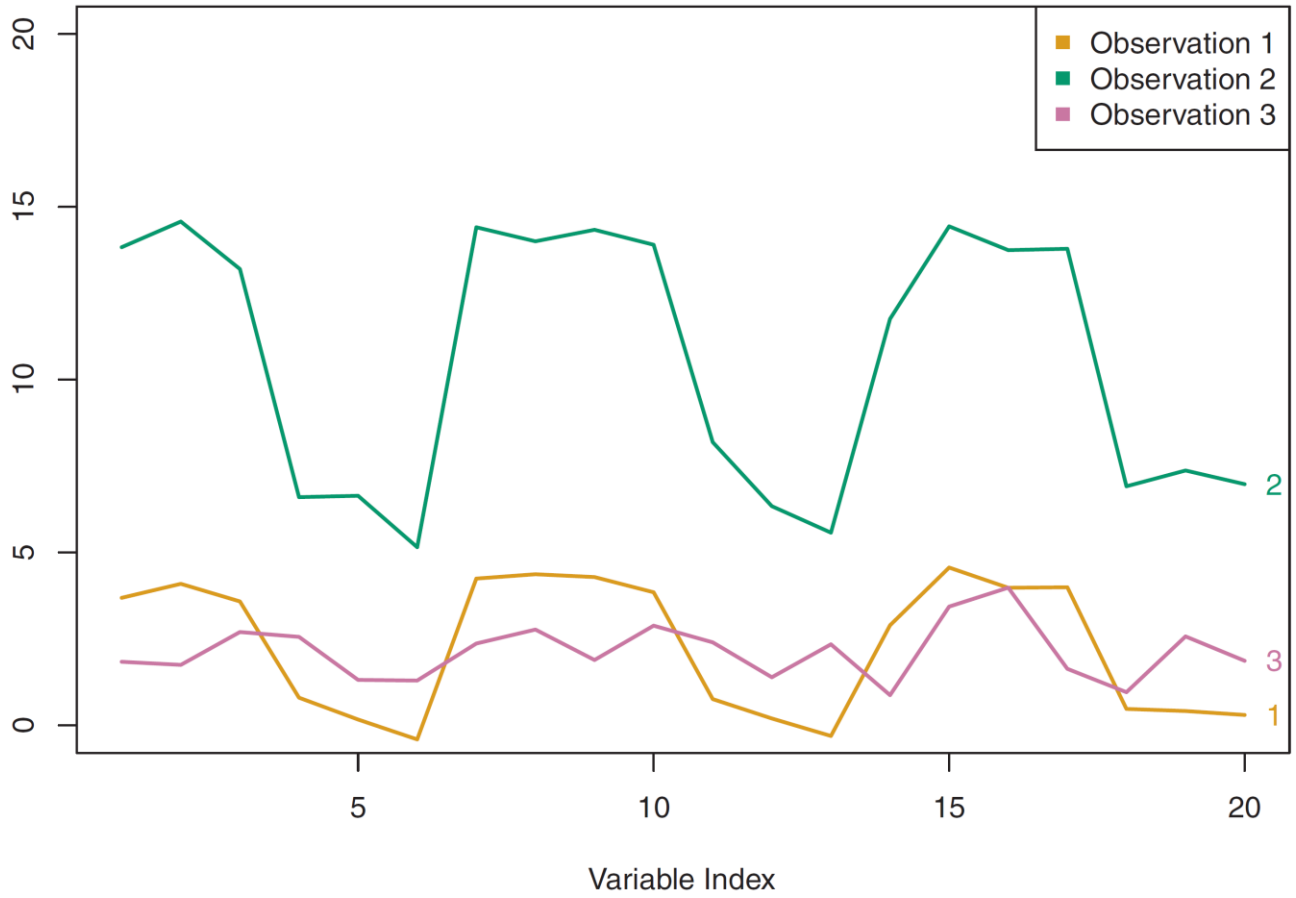
In the graph below, observations 1 and 3 have similar values (y-axis) for each of their features (x-axis), and would therefore have a relatively small Euclidean distance.



James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Choosing dissimilarity measure

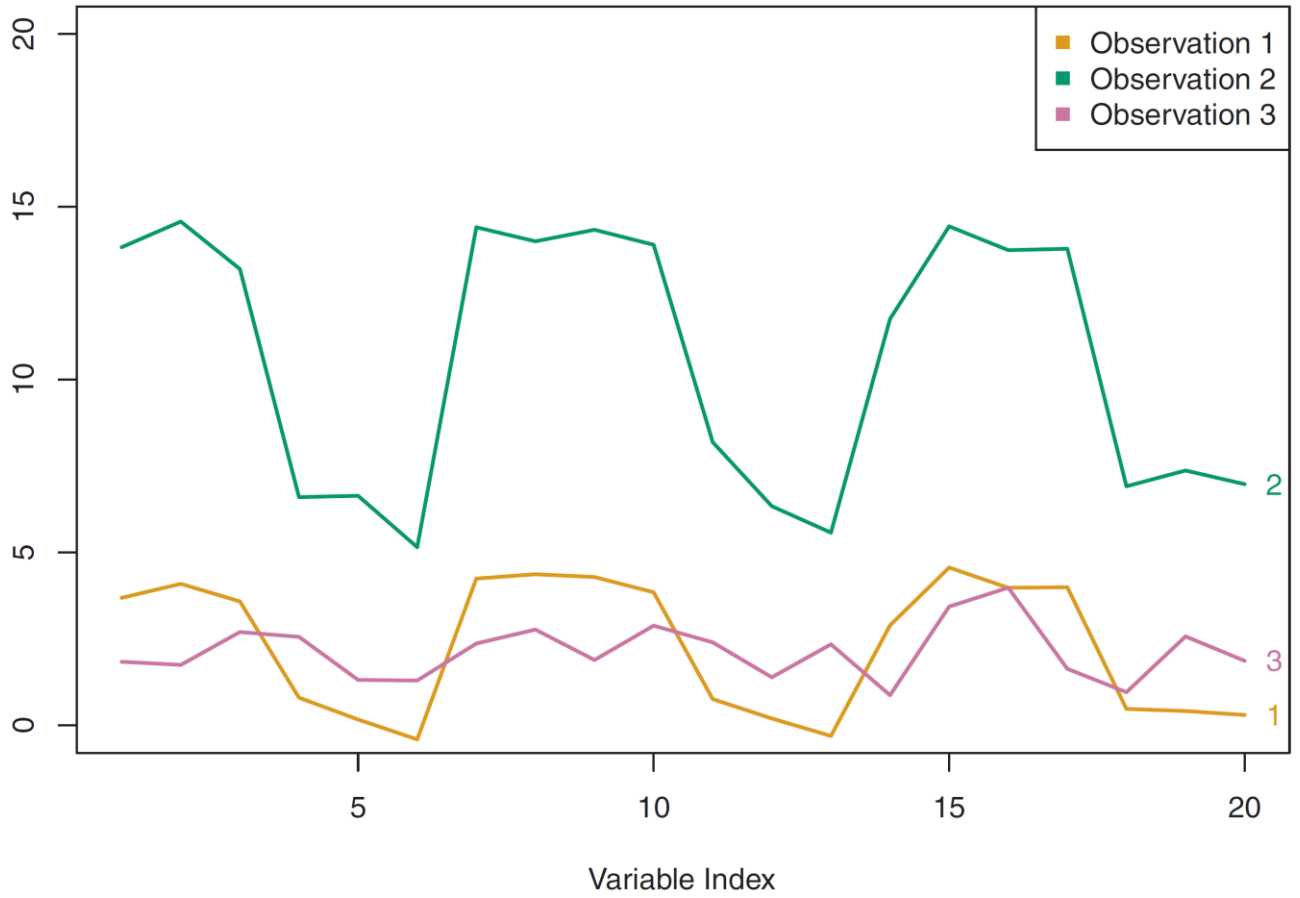
However, observations 1 and 3 are weakly correlated, and so would have a large distance if the dissimilarity was instead the correlation coefficient between the two observations.



James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Choosing dissimilarity measure

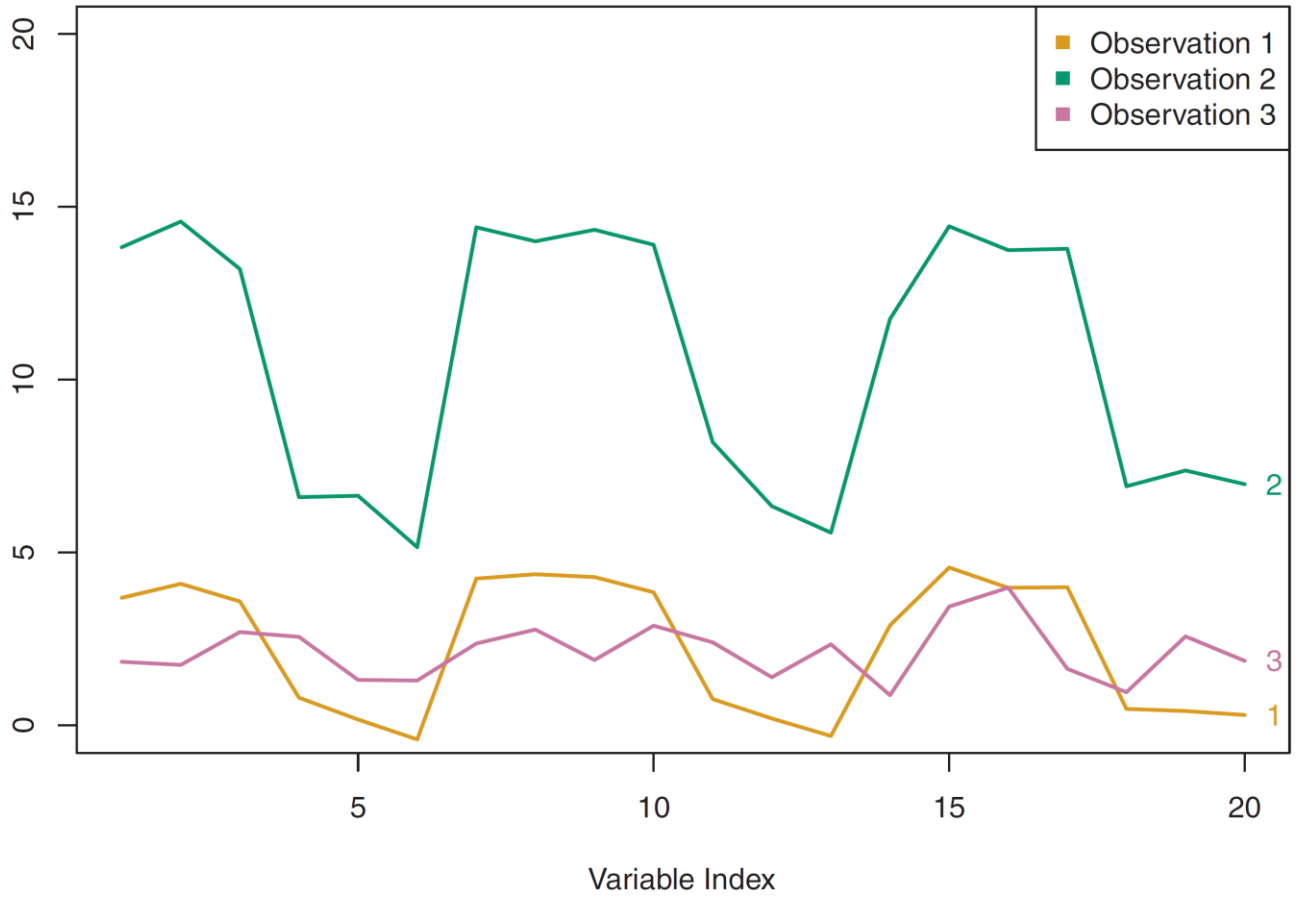
In contrast, observations 1 and 2 have large Euclidean distance, yet they have high positive correlation, and so their distance based on correlations would be low.



James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Choosing dissimilarity measure

Therefore, one is interested in overall trends and shapes, then a dissimilarity measure like the correlation would be more appropriate than the Euclidean distance measure.



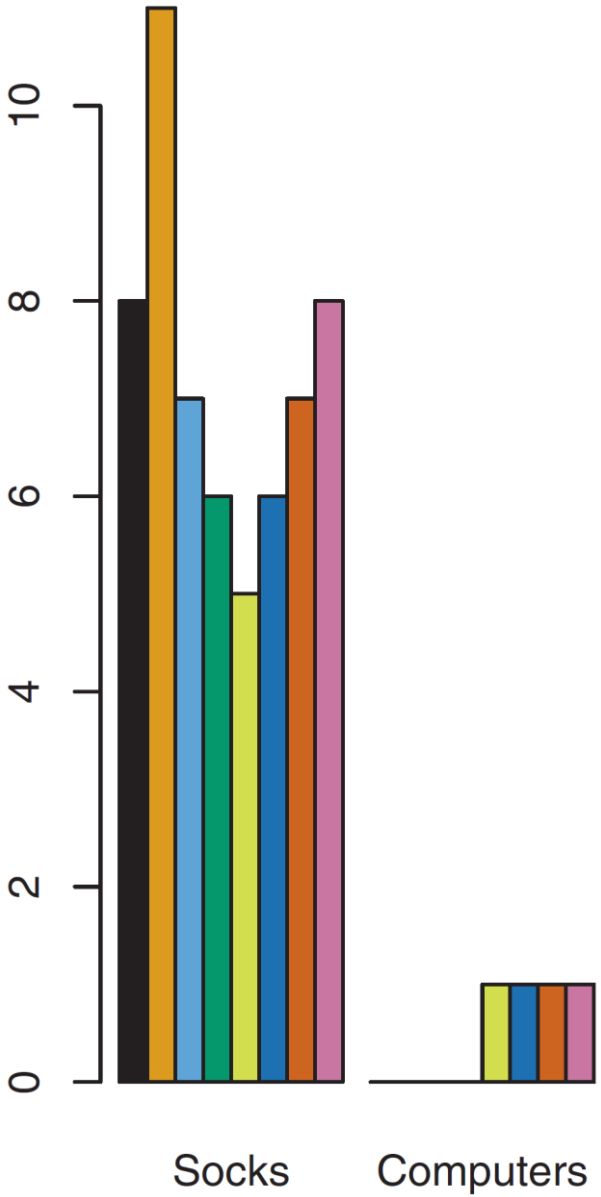
James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Should variables be scaled?

In addition to choosing the linkage criterion and the dissimilarity measure, it is also important to consider whether features should be scaled or not.

Suppose we are interested in identifying sets of shoppers who have similar purchasing behaviors so that we can target advertisements to them.

For illustration, we consider shoppers (colored bars) with two different items (socks and computers), and measure the number of socks and number of computers purchased.

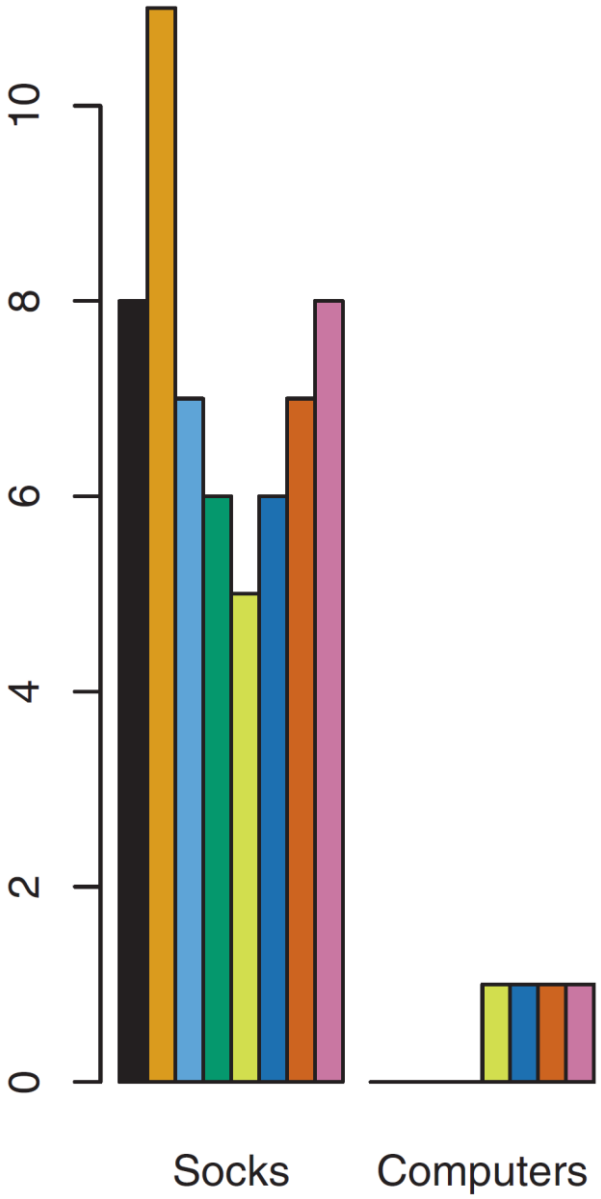


Adapted from James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Should variables be scaled?

By using Euclidean distances on this raw data, the number of socks purchased by individuals will drive the dissimilarity between individuals, with number of computers purchased having little effect.

This could be undesirable, as computers are more expensive (and we want shoppers to buy computers, not socks), and number of socks purchased by two shoppers may be less informative about their overall shopping preferences than a small difference in the number of purchased computers.

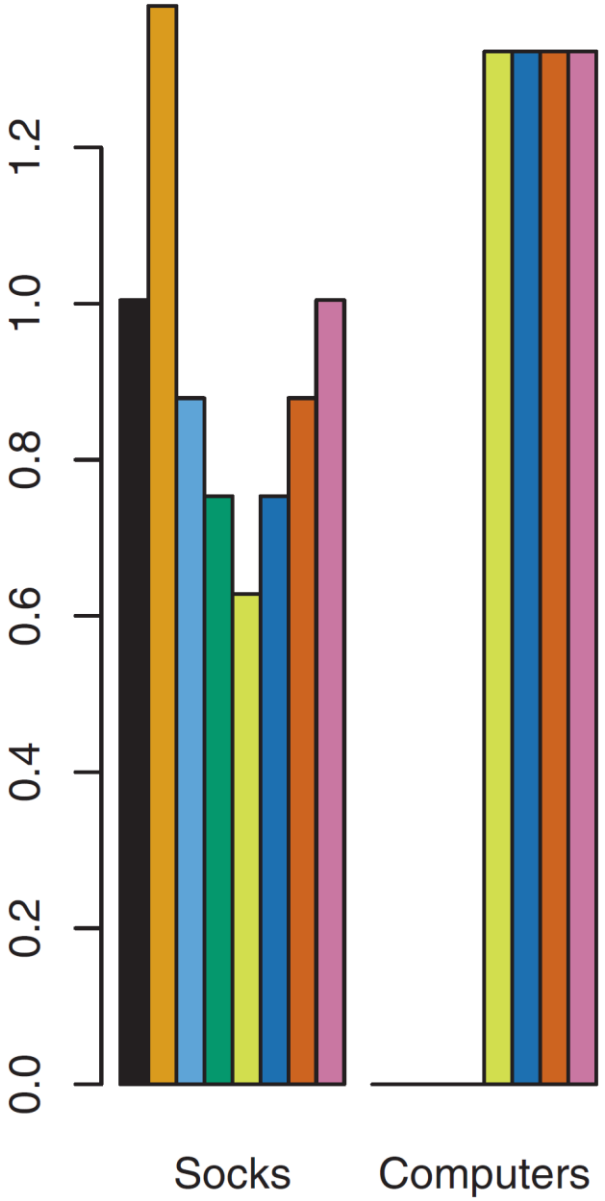


Adapted from James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Should variables be scaled?

Here we have the same data, but instead each variable is normalized by its standard deviation.

Now the number of computers purchased will have a greater effect on the inter-observation dissimilarities

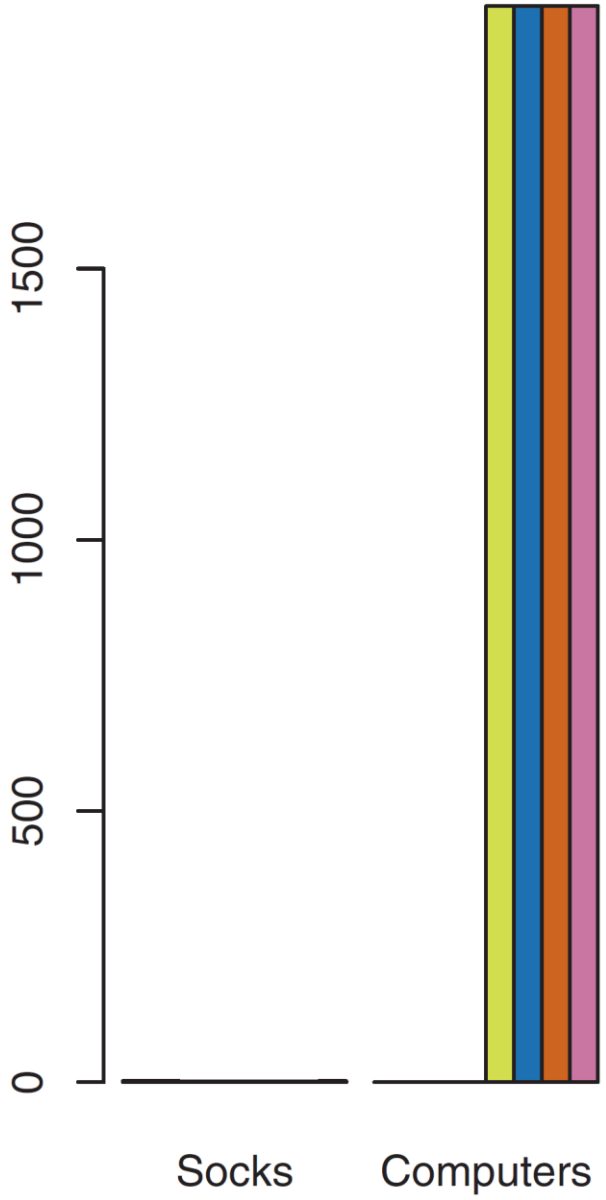


Adapted from James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Should variables be scaled?

Finally, we have the same data, but the y-axis represents the number of dollars spent by each shopper on socks and on computers.

Because computers are much more expensive than socks, now computers purchasing history will drive the inter-observation dissimilarities.



Adapted from James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.