Principal component analysis: theory and concepts

Introduction to Statistical Modelling

Prof. Joris Vankerschaver

Goal of dimensionality reduction

- Pre-processing
  - Remove collinear predictors (multicollinearity)
- Computational efficiency
  - Retain import features to speed up computational processing
- Visualization

Learning outcomes

At the end of this lecture, you should be able to:

1. Explain the ideas behind PCA
2. Do a PCA by hand given a covariance matrix
3. Do a PCA with R
4. Interpret and explain the PCA results
5. Build and explain a PCR model

References

- *Introduction to statistical modeling*. Chapter available on Ufora.
- *An Introduction to Statistical Learning*. Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani. Available for free online at https://www.statlearning.com/.
    - PCA: section 6.3

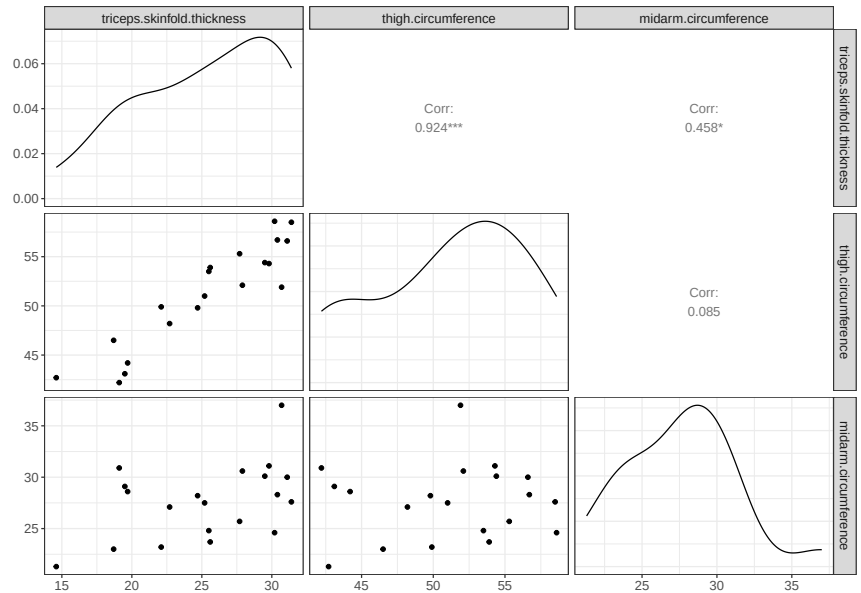
Reminder: multicollinearity

Bodyfat dataset: 20 observations, predict amount of body fat from three body measurements.

- Linear regression model:

$$\texttt{bodyfat} = 117.085 + 4.334 \cdot \texttt{triceps} - 2.857 \cdot \texttt{thigh} - 2.186 \cdot \texttt{midarm}$$

- **Do you see anything wrong with this?**



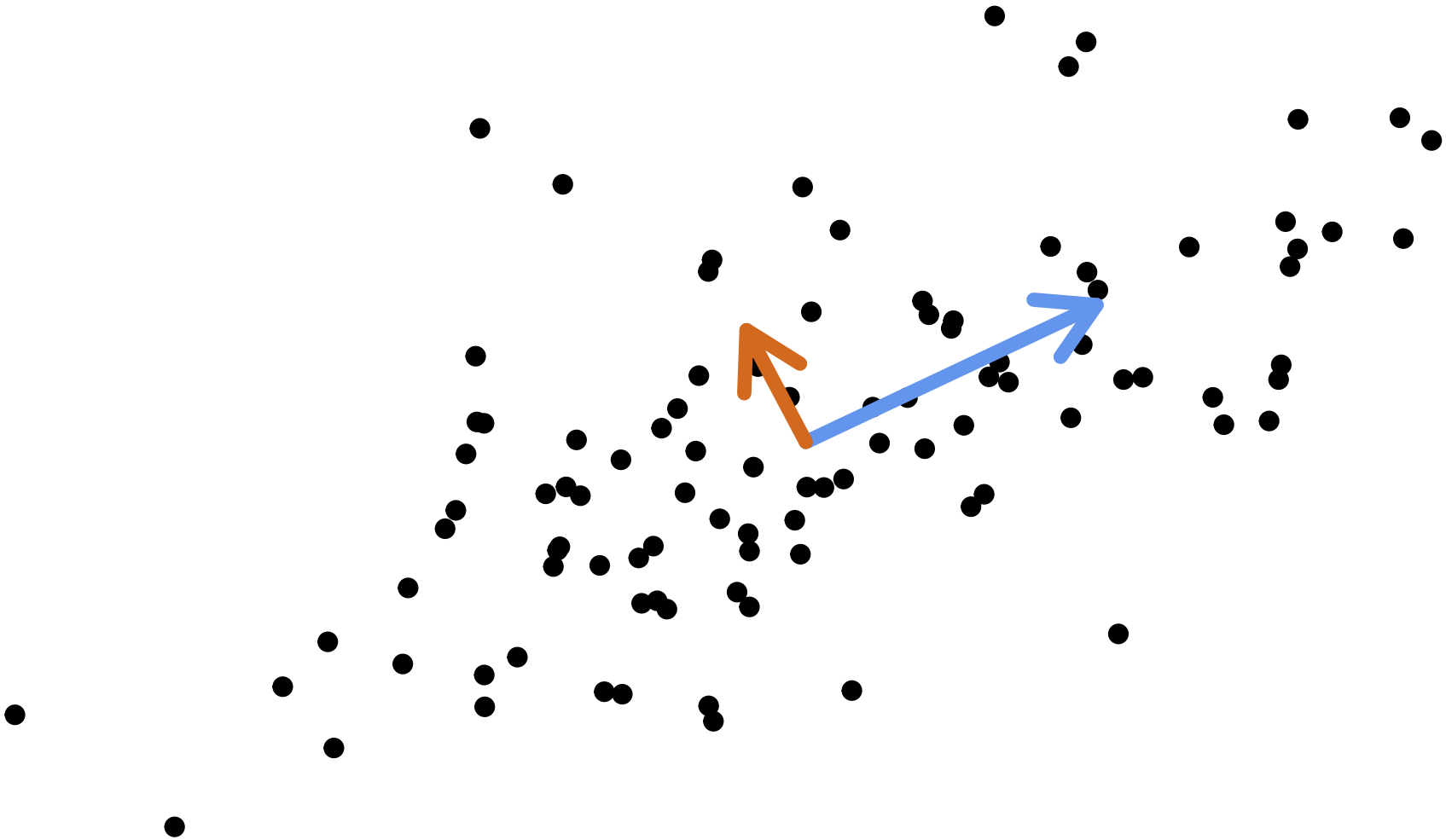
```
Call:
lm(formula = bodyfat ~ ., data = bodyfat)

Residuals:
    Min      1Q  Median      3Q     Max
-3.7263 -1.6111  0.3923  1.4656  4.1277

Coefficients:
                           Estimate Std. Error t value Pr(>|t|)
(Intercept)                 117.085     99.782   1.173    0.258
triceps.skinfold.thickness    4.334      3.016   1.437    0.170
thigh.circumference          -2.857      2.582  -1.106    0.285
midarm.circumference         -2.186      1.595  -1.370    0.190

Residual standard error: 2.48 on 16 degrees of freedom
Multiple R-squared:  0.8014,    Adjusted R-squared:  0.7641
F-statistic: 21.52 on 3 and 16 DF,  p-value: 7.343e-06
```

Principal component analysis

Directions of maximal variability

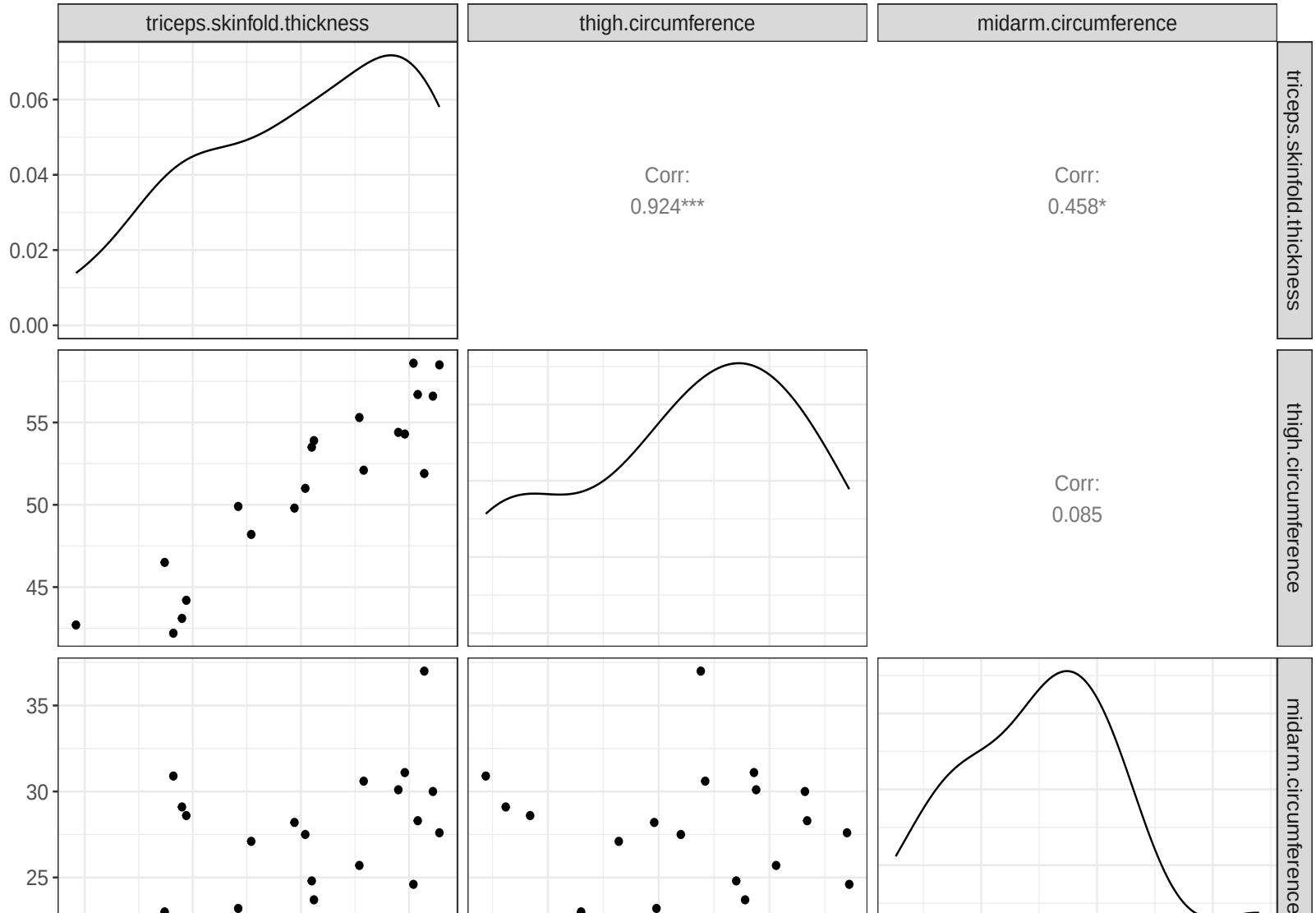
Intuitively:

- Find directions of maximal variability in the dataset
- Discard directions in which there is neglible variability



Directions of less variability

Since `triceps` and `thigh` are highly correlated, specifying both is superfluous. What do we lose if we throw away one of these variables?



Notation

Dataset:

- N observations $\mathbf{x}_k$, $k = 1, \ldots, N$
- Each observation is a (column) vector in $\mathbb{R}^D$

Data matrix:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_N^T \end{bmatrix} \in \mathbb{R}^{N \times D}$$

Columns of the data matrix:

- Referred to as features, predictors, independent variables
- Denoted by $\mathbf{X}_i$, $i = 1, \ldots, D$

Notation: example (body fat dataset)

- 20 observations with 3 features each
- Data matrix is 20×3 matrix
- Features:
  - $\mathbf{X}_1$: `triceps.skinfold.thickness`
  - $\mathbf{X}_2$: `thigh.circumference`
  - $\mathbf{X}_3$: `midarm.circumference`

The covariance matrix

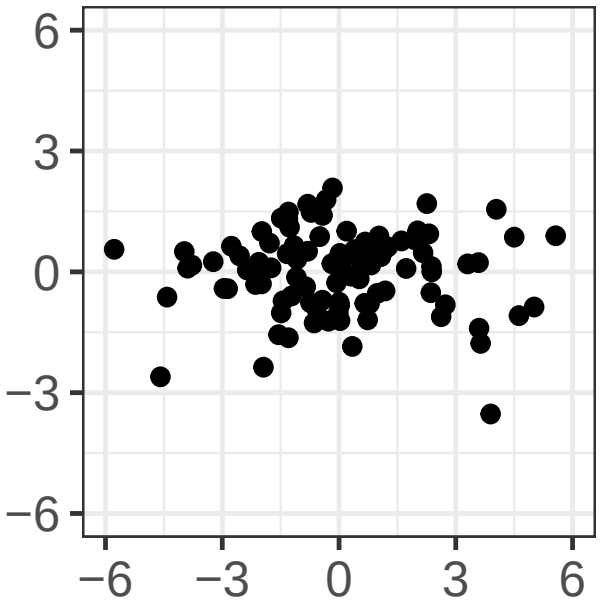
Given observations $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^D$, the variance-covariance matrix $\mathbf{S}$ is defined as:

$$\mathbf{S} = \frac{1}{N} \sum_{k=1}^{N} \left(\mathbf{x}_k \mathbf{x}_k^T - \bar{\mathbf{x}} \bar{\mathbf{x}}^T \right).$$

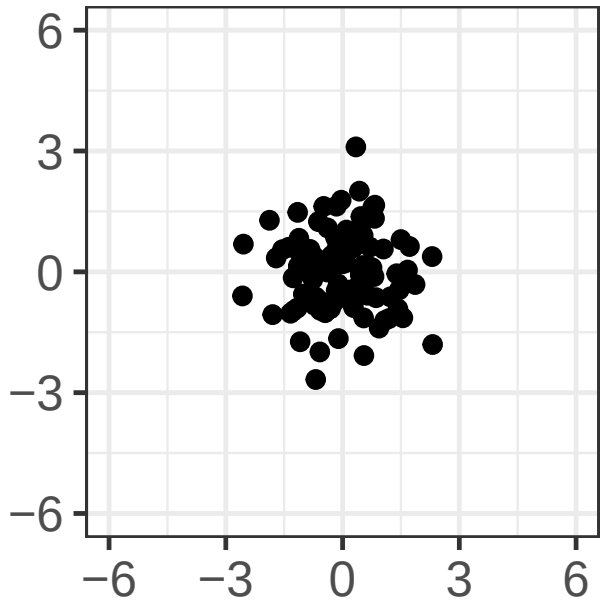
Structure: variances and covariances between components of the data.

$$\mathbf{S} = \begin{bmatrix} \mathsf{Var}(x_1) & \mathsf{Cov}(x_1, x_2) & \cdots & \mathsf{Cov}(x_1, x_D) \\ \mathsf{Cov}(x_2, x_1) & \mathsf{Var}(x_2) & \cdots & \mathsf{Cov}(x_2, x_D) \\ \vdots & \vdots & \ddots & \vdots \\ \mathsf{Cov}(x_D, x_1) & \mathsf{Cov}(x_D, x_2) & \cdots & \mathsf{Var}(x_D) \end{bmatrix}$$

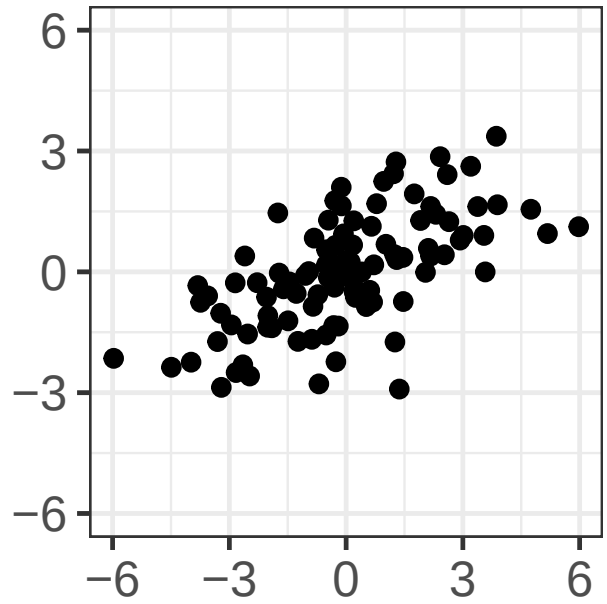
The covariance matrix: examples



$$\mathbf{S} = \begin{bmatrix} 6 & 0 \\ 0 & 1 \end{bmatrix}$$



$$\mathbf{S} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



$$\mathbf{S} = \begin{bmatrix} 5 & 2 \\ 2 & 2 \end{bmatrix}$$

Clearly the covariance matrix will help us find directions of maximum variability, but how?

Linear combination of features

- The **first principal component** $\mathbf{Z}_1$ is a linear combination of the columns of $\mathbf{X}$:

$$\mathbf{Z}_1 = v_1\mathbf{X}_1 + \cdots + v_D\mathbf{X}_D,$$

  where we will choose the coefficients v_i so that the variances is maximal, in some sense.

- The coefficients v_i are referred to as the **loadings** and the vector

$$\mathbf{v} = \begin{bmatrix} v_1 \\ \vdots \\ v_D \end{bmatrix}$$

  is the **loadings vector**.

- Variance of $\mathbf{Z}_1$:

$$\text{Var}(\mathbf{Z}_1) = \mathbf{v}^T\mathbf{S}\mathbf{v}.$$

Maximizing the variance

- Idea: choose loadings $\mathbf{v}$ so that $\text{Var}(\mathbf{Z}_1)$ is **maximal**.
- Problem: just by increasing the norm of $\mathbf{v}$, variance can become as large as we want. Solution: impose that $\|\mathbf{v}\| = 1$.

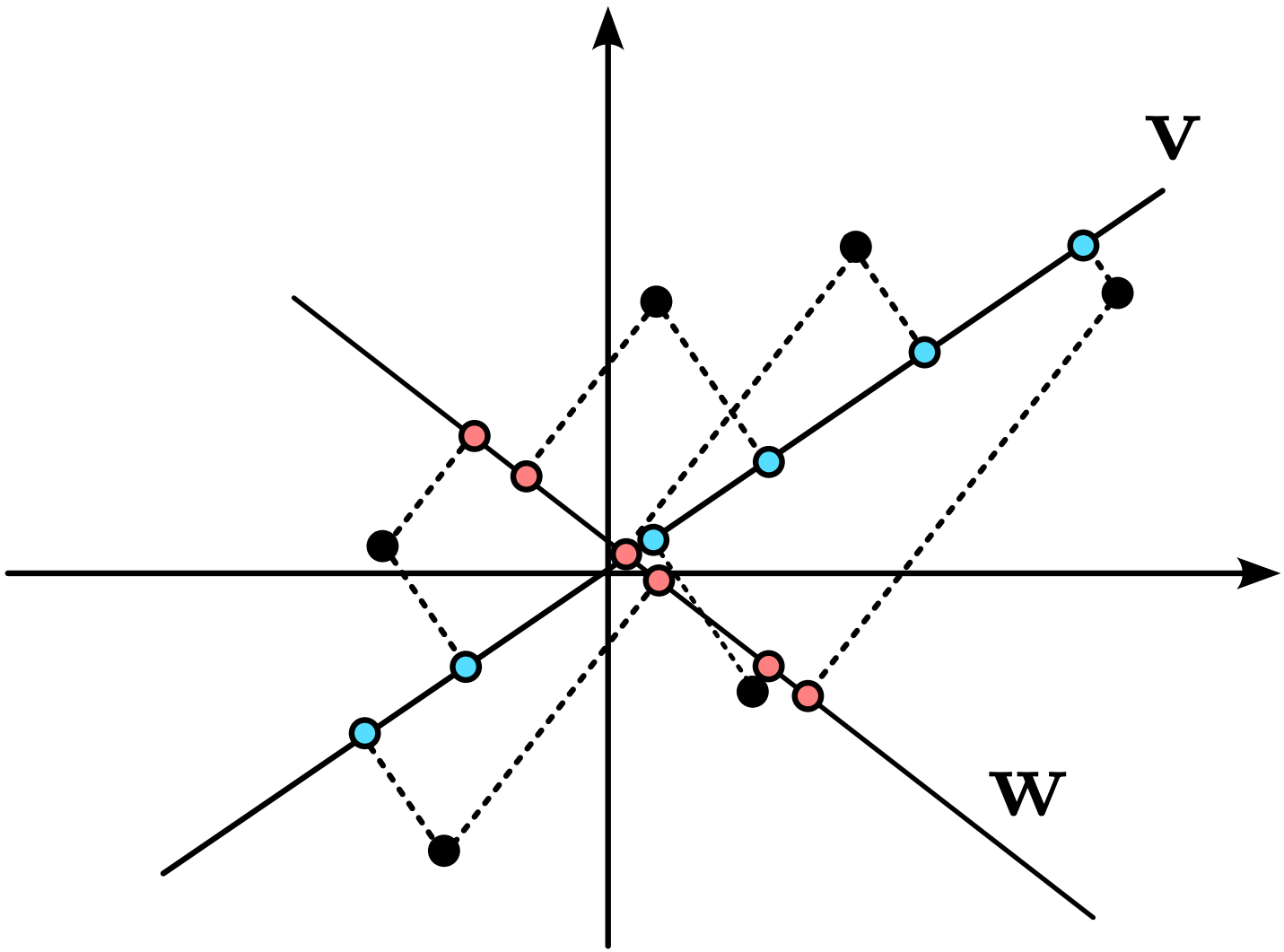
PC1: maximization of variance

The loadings vector $\mathbf{v}$ for the first principal component is found by solving the following maximization problem:

$$\text{maximize } \mathbf{v}^T\mathbf{S}\mathbf{v} \text{ so that } \mathbf{v}^T\mathbf{v} = 1.$$

Geometric interpretation

- $\mathbf{Z}_1$: projection of data on the line in the direction of $\mathbf{v}$.
- Find direction $\mathbf{v}$ so that variance is maximal (blue)



Eigenvalue problem

Through Lagrange multipliers, can show that maximizing variance is equivalent to finding eigenvalues and eigenvectors of $\mathbf{S}$.

PC1: eigenvalues

The loadings vector $\mathbf{v}$ for the first principal component is the eigenvector of $\mathbf{S}$ with the largest eigenvalue:

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

Eigenvectors are typically quite efficient to compute.

Amount of variance explained

Take the eigenvalue equation

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

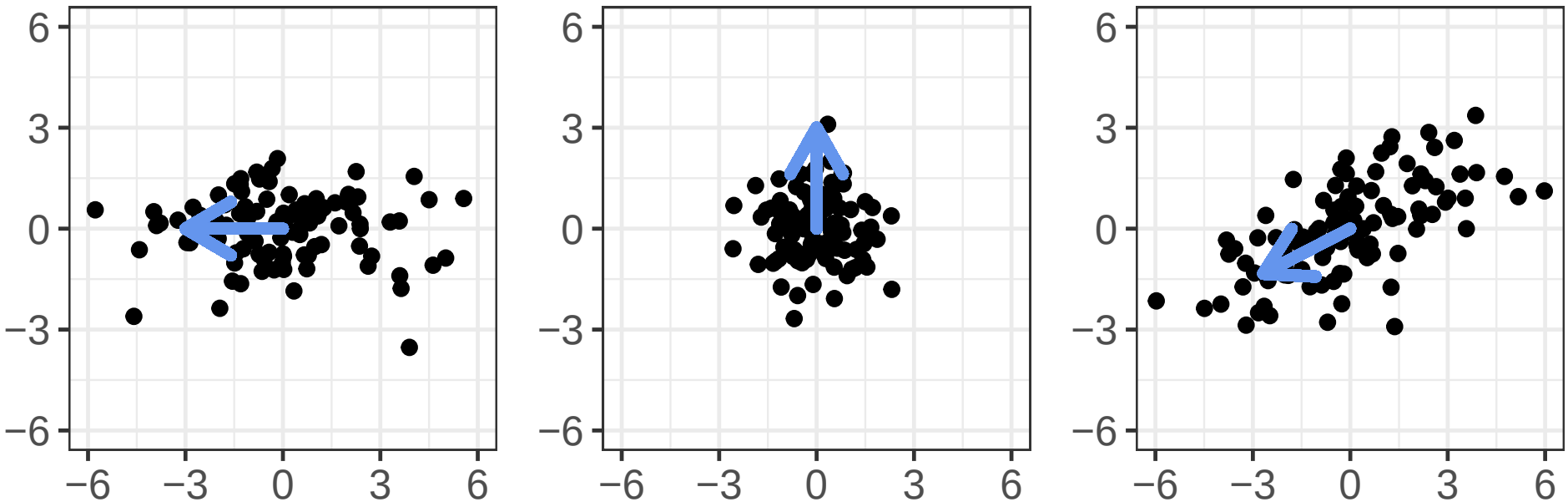
and left-multiply by $\mathbf{v}^T$ to get

$$\lambda = \lambda\mathbf{v}^T\mathbf{v} = \mathbf{v}^T\mathbf{S}\mathbf{v} = \mathrm{Var}(\mathbf{Z}_1).$$

Eigenvalues and eigenvectors

- The largest eigenvalue of $\mathbf{S}$ is equal to the variance contained in ("explained by") the first principal component $\mathbf{Z}_1$.
- The corresponding eigenvector gives the loadings vector $\mathbf{v}$.

Example



The loadings vectors may point in the opposite direction of what you expected... Why is this not a problem?

The remaining principal components

Next principal components $\mathbf{Z}_2, \mathbf{Z}_3, \ldots$ involve variation in the data after $\mathbf{Z}_1$ has been taken into account.

- For $\mathbf{Z}_2$:

$$\text{maximize } \text{Var}(\mathbf{Z}_2) \text{ so that } \text{Cov}(\mathbf{Z}_1, \mathbf{Z}_2) = 0$$

- Equivalent to: find second largest eigenvalue λ_2 and eigenvector $\mathbf{v}_2$.

Same story for remaining principal components.

> **Note**
>
> The principal components are *uncorrelated* linear combinations of features that *maximize variance*.

How many principal components are there?

Recall:

- $\mathbf{S}$ is a symmetric $D \times D$ matrix
- Such a matrix always has D eigenvalues and eigenvectors

When $D \leq N$ (more data points than features)

- In general, D non-zero principal components

When $D > N$:

- $\mathbf{S}$ has rank at most N: N non-zero principal components
- Can happen in high-dimensional datasets (e.g. gene assays)

Percentage of variance explained

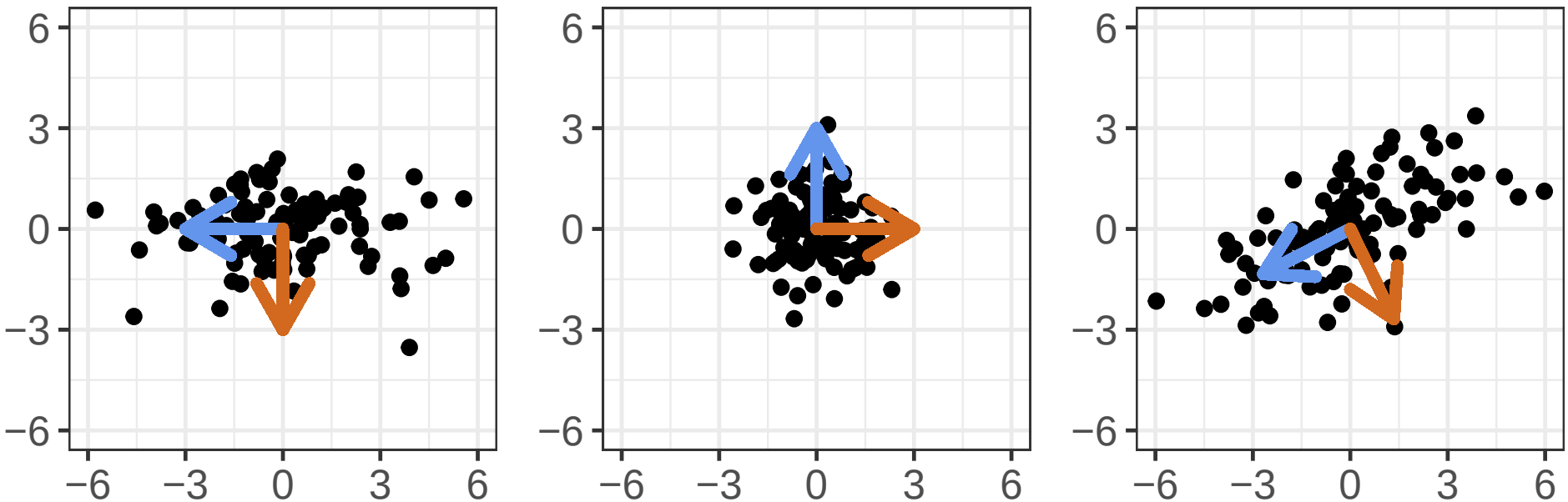
- Eigenvalue λ_i is amount of variance explained by PC i.
- Total amount of variance: $\lambda_1 + \lambda_2 + \cdots + \lambda_D$
- Percentage of variance explained by PC i:

$$\frac{\lambda_i}{\lambda_1 + \cdots + \lambda_D}$$

In many cases, the first few PCs will explain the majority of variance (80% to 90%).

Dimensionality reduction: we can omit the remaining principal components with only a small loss of information

Example

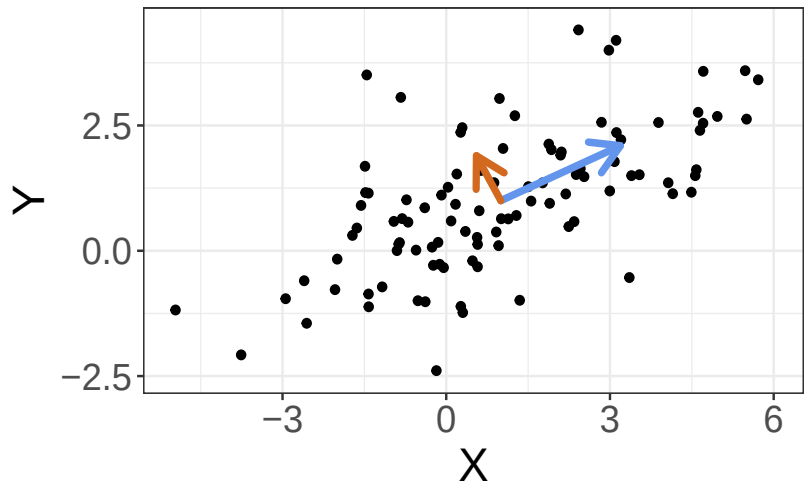


Blue: first PC, red: second PC.

Worked out example (by hand)

Dataset is chosen so that

$$\mathbf{S} = \begin{bmatrix} 5 & 2 \\ 2 & 2 \end{bmatrix}.$$



Eigenvalues:

$$\lambda_1 = 6, \quad \lambda_2 = 1.$$

Eigenvectors:

$$\mathbf{v}_1 = \frac{1}{\sqrt{5}} \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \quad \mathbf{v}_2 = \frac{1}{\sqrt{5}} \begin{bmatrix} -1 \\ 2 \end{bmatrix}.$$

Worked out example (with R)

```
prcomp(df)
```

```
Standard deviations (1, .., p=2):
[1] 2.44949 1.00000

Rotation (n x k) = (2 x 2):
         PC1         PC2
X 0.8944272 -0.4472136
Y 0.4472136  0.8944272
```

Note:

- Standard deviations are **square roots** of eigenvalues
- Columns of rotation matrix give loadings vectors

Example: body fat dataset

```
pca <- prcomp(bodyfat_predictors)
pca
```

```
Standard deviations (1, .., p=3):
[1] 7.2046011 3.7432587 0.1330841

Rotation (n x k) = (3 x 3):
                                 PC1        PC2        PC3
triceps.skinfold.thickness 0.6926671  0.1511979  0.7052315
thigh.circumference        0.6985058 -0.3842734 -0.6036751
midarm.circumference       0.1797272  0.9107542 -0.3717862
```

Percentage of variance explained

```
summary(pca)
```

```
Importance of components:
                          PC1    PC2     PC3
Standard deviation     7.2046 3.7433 0.13308
Proportion of Variance 0.7872 0.2125 0.00027
Cumulative Proportion  0.7872 0.9997 1.00000
```

- First 2 PCs explain over 99% of variance in data
- Interpretation:

$$\text{PC}_1 = 0.693 \cdot \texttt{triceps} + 0.699 \cdot \texttt{thigh} + 0.179 \cdot \texttt{midarm}$$
$$\text{PC}_2 = 0.151 \cdot \texttt{triceps} - 0.384 \cdot \texttt{thigh} - 0.910 \cdot \texttt{midarm}$$

Standardizing the features

Often, data are standardized before running PCA:

$$\mathbf{Y}_i = \frac{\mathbf{X}_i - \bar{\mathbf{X}}_i}{\text{SD}(\mathbf{X}_i)}$$

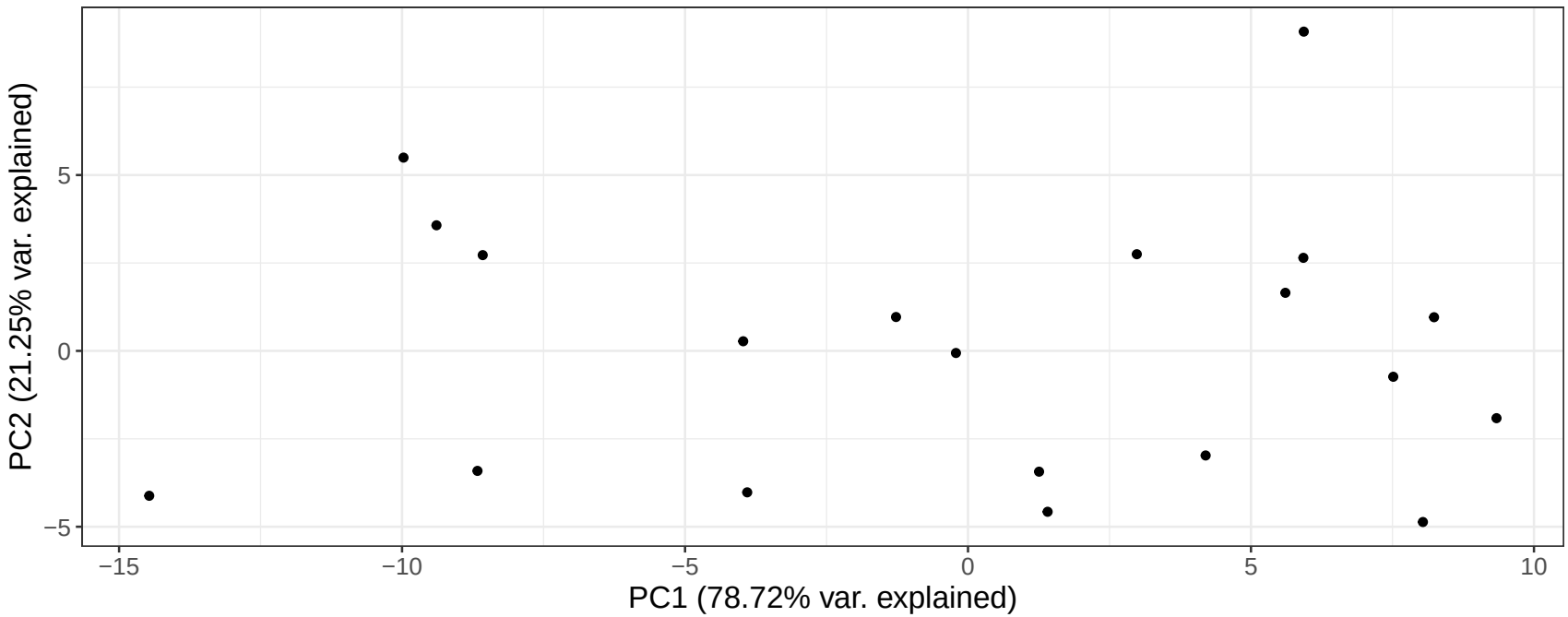
- Standardization puts all features on the same scale and affects the outcome of your PCA.
- Often a good idea when features have different units (e.g. mm, Watt, sec).
- **Not** a good idea when features have the same units (e.g. pixel intensities in an image).

In R: `prcomp(df, center = TRUE, scale = TRUE)`.

Interpretation of PCA results

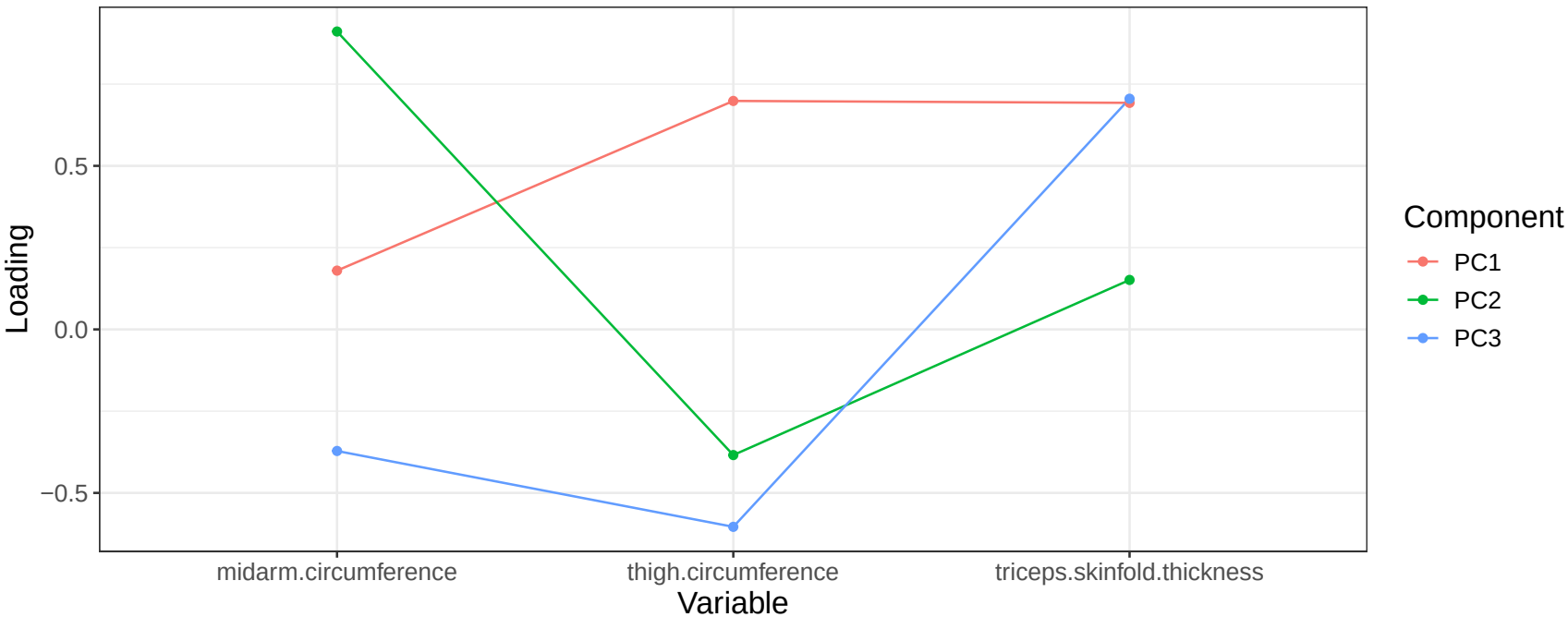
Score plot

- Scatter plot of two PC (usually PC1 and PC2)
- Can be used to spot patterns in data (see later)



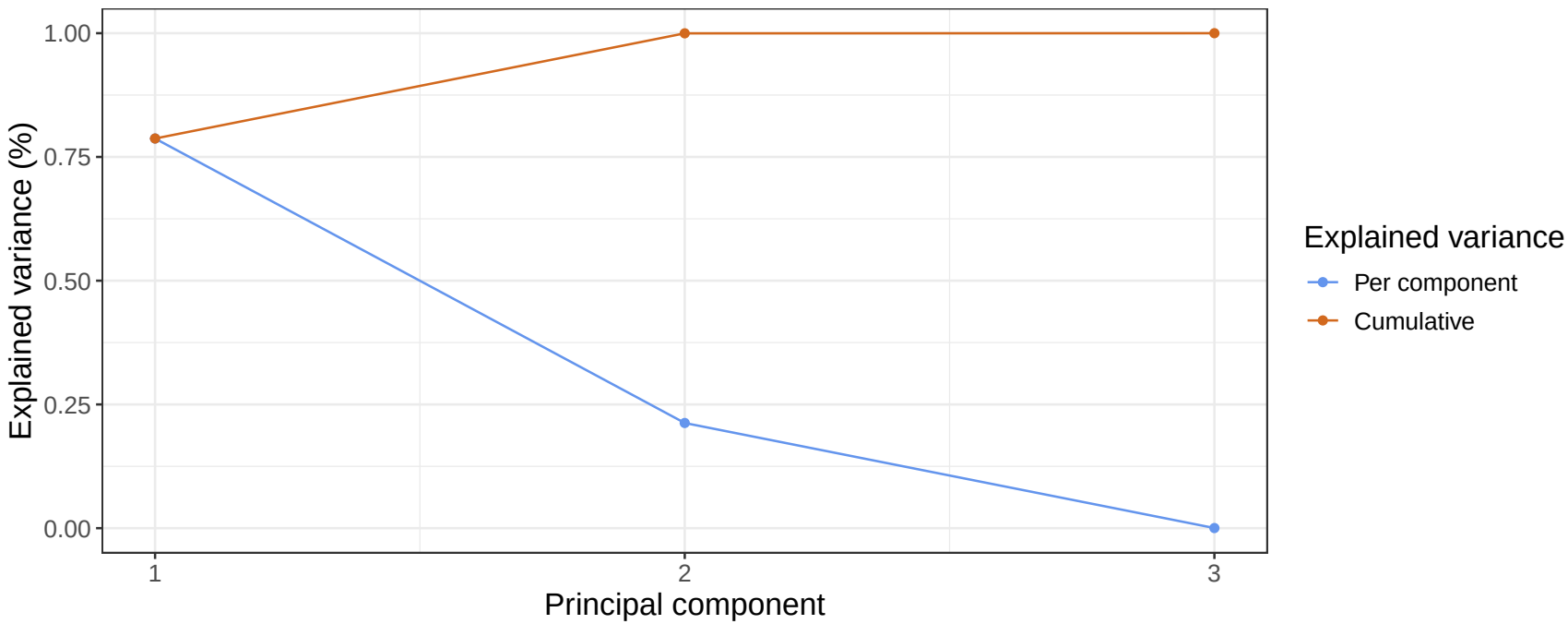
Loadings plot

- Shows how much each variable contributes to each PC
- Useful to discern patterns in PCs



Scree plot

- Shows percentage of variance explained per PC
- Useful to determine the PCs that contribute most to variance
- Can be made with R's `screeplot` command, but better to make your own (it's just a line plot)



Selecting the number of principal components to retain

Many heuristics exist for selecting "optimal" number of PCs:

- Explain fixed percentage (e.g. 80%) of variance
- "Elbow" in scree plot
- ...

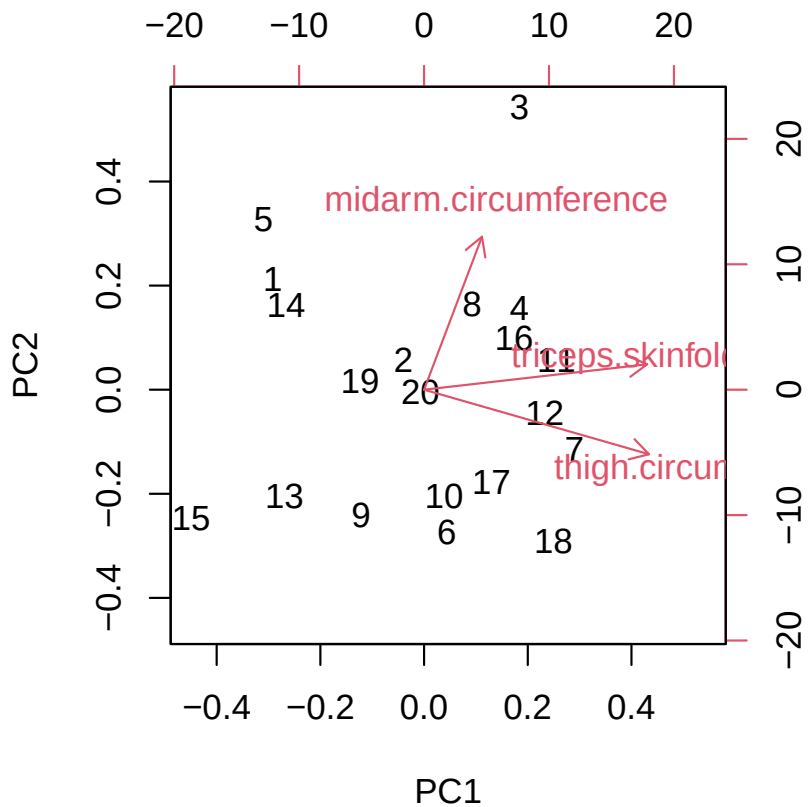
Can also determine number of PCs dynamically (e.g. if doing regression on PCs, look at R^2)

Image credit: https://en.wikipedia.org/wiki/Scree (Kevin Lenz, CC BY-SA 2.5)

Biplot

- Biplot = loadings plot + score plot
- Numbers: data for first two PC
- Arrows: contribution of variables to first two PC

```
biplot(pc)
```



Principal component regression

Principal component regression (PCR)

Idea:

- Do a PCA (usually with standardized features)
- Build a linear model on reduced number of PCs

Why do we do this?

- PCs are uncorrelated, so takes care of multicollinearity
- Results in a simpler model where only important features play a role

Not always the right thing to do, alternatives exist (e.g. ridge regression, see later)

PCR by hand: starting with PC 1

```r
pc1 <- pc$x[, "PC1"]
model_1 <- lm(bodyfat$bodyfat ~ pc1)
summary(model_1)
```

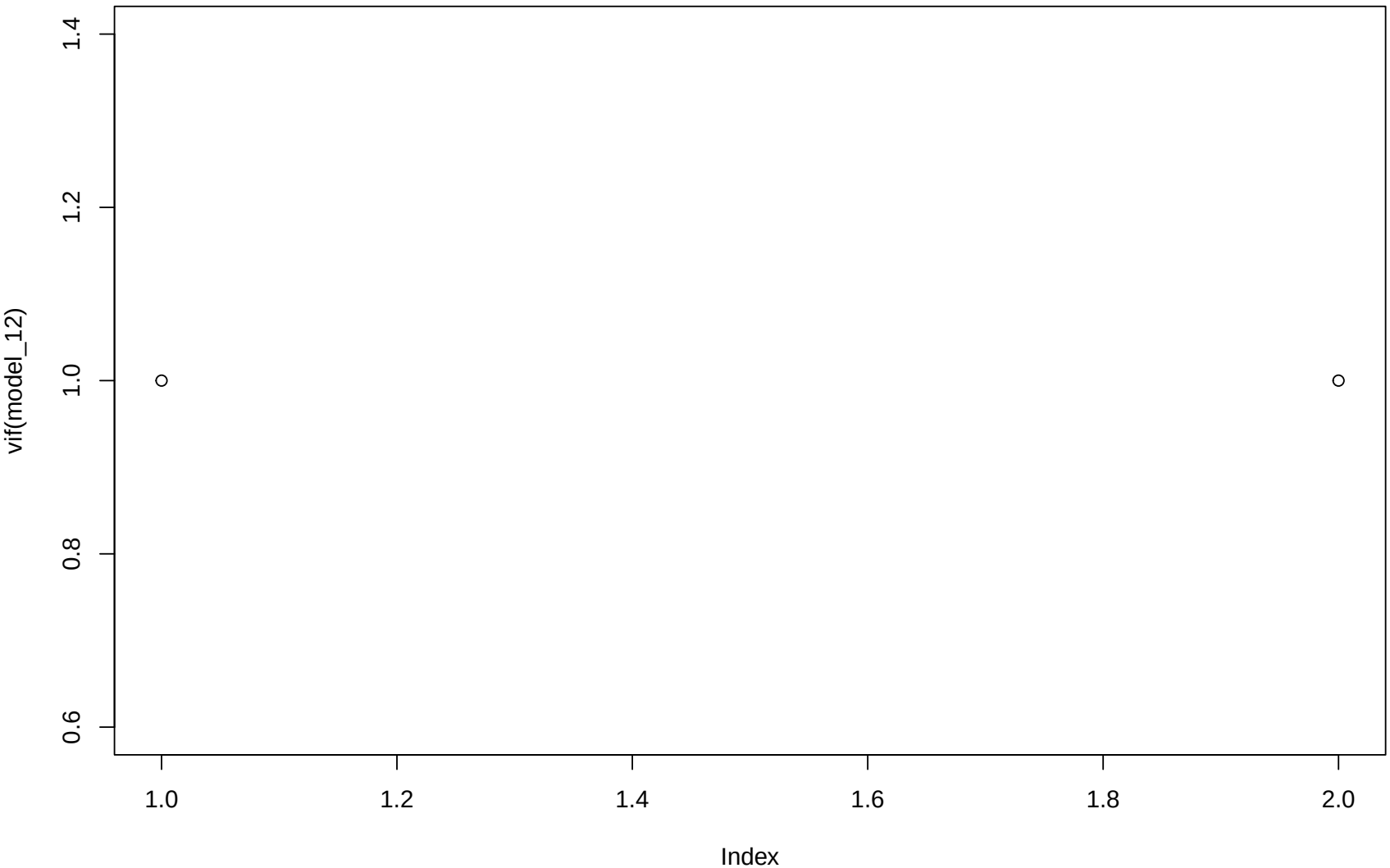
```
Call:
lm(formula = bodyfat$bodyfat ~ pc1)

Residuals:
    Min      1Q  Median      3Q     Max
-5.1357 -1.8821  0.2682  1.7107  3.4992

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 20.19500    0.58688  34.411  < 2e-16 ***
pc1          0.61366    0.08358   7.343 8.13e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.625 on 18 degrees of freedom
Multiple R-squared:  0.7497,    Adjusted R-squared:  0.7358
F-statistic: 53.91 on 1 and 18 DF,  p-value: 8.128e-07
```

PCR by hand: adding PC 2

```
Call:
lm(formula = bodyfat$bodyfat ~ pc1 + pc2)

Residuals:
    Min      1Q  Median      3Q     Max
-3.9876 -1.8822  0.2562  1.3209  4.0285

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 20.19500    0.56604  35.678  < 2e-16 ***
pc1          0.61366    0.08061   7.613 7.12e-07 ***
pc2         -0.23785    0.15514  -1.533    0.144
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.531 on 17 degrees of freedom
Multiple R-squared:  0.7801,    Adjusted R-squared:  0.7542
F-statistic: 30.15 on 2 and 17 DF,  p-value: 2.564e-06
```

Aside: variance inflation factors

Reminder: principal components are uncorrelated by definition, so all the VIFs will be equal to 1.



PCR by hand: putting together the final model

- We would probably select `model_1`:

$$\texttt{bodyfat} = 20.195 + 0.614 \cdot \texttt{PC}_1$$

- This model uses the principal components as predictors, but we typically want the original predictors. Recall

$$\texttt{PC}_1 = 0.693 \cdot \texttt{triceps} + 0.699 \cdot \texttt{thigh} + 0.179 \cdot \texttt{midarm}$$

- Putting these two together gives the final PCR model:

$$\texttt{bodyfat} = 20.195 + 0.426 \cdot \texttt{triceps} + 0.429 \cdot \texttt{thigh} + 0.110 \cdot \texttt{midarm}$$

Stepwise building a model and rewriting it back in terms of the original predictors is a lot of work. Is there a better way?

PCR via the pls package

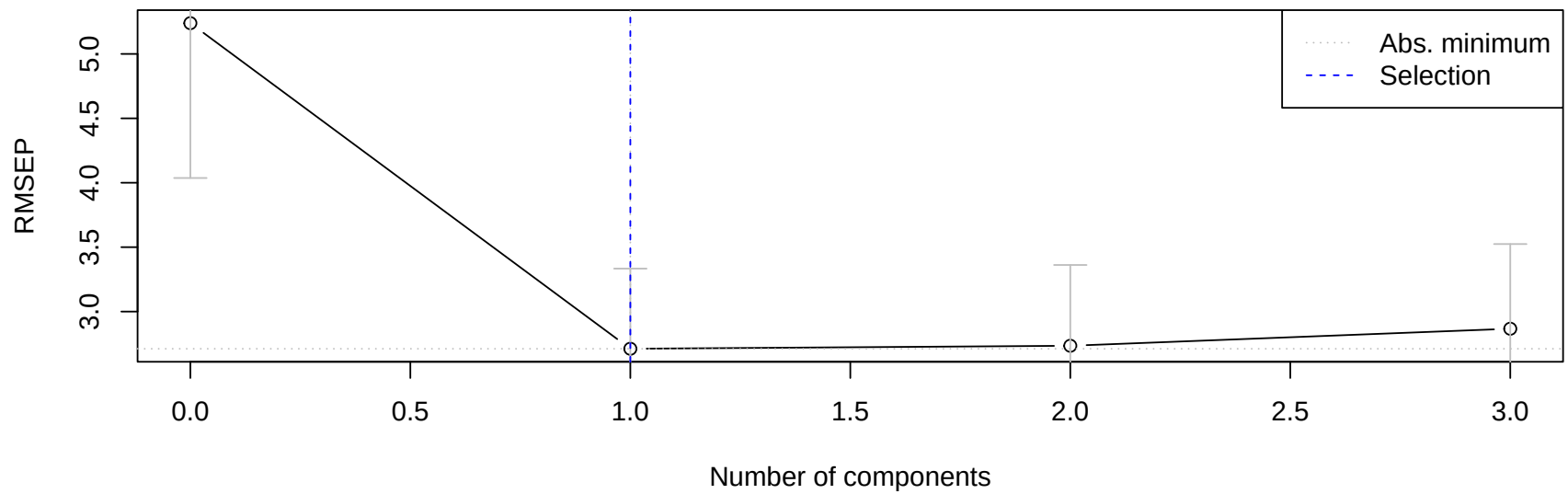
```r
library(pls)

pcr_model <- pcr(bodyfat ~ ., data = bodyfat, validation = "CV")
summary(pcr_model)
```

```
Data:   X dimension: 20 3
    Y dimension: 20 1
Fit method: svdpc
Number of components considered: 3

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps
CV           5.239    2.711    2.735    2.867
adjCV        5.239    2.695    2.713    2.832

TRAINING: % variance explained
         1 comps  2 comps  3 comps
X          78.72    99.97   100.00
bodyfat    74.97    78.01    80.14
```

Selecting the optimal number of components

- "1-sigma rule": select model with least number of components whose cross-validation error is at most 1 standard deviation away from optimal model
- In human language: 2 PCs gives lowest RMSEP, but we can go down to 1 PC without losing too much

```
selectNcomp(pcr_model, method = "onesigma", plot = TRUE)
```




```
[1] 1
```

Ridge regression (optional)

- PCR is not the only way to "regularize" a regression model
- Ridge regression: like ordinary regression, but punish model for coefficients that become too large.
- Objective function:

$$J_{\text{ridge}}(\alpha, \beta) = J(\alpha, \beta) + \lambda(\alpha^2 + \beta^2).$$

- Parameter λ set to a fixed value or determined via cross-validation.
  - $\lambda = 0$: ridge regression = ordinary regression
  - $\lambda \to \infty$: all coefficients become zero
- Can be done with the `glmnet` package (not very userfriendly)

Example session

```
library(glmnet)
predictors <- data.matrix(bodyfat_predictors)
outcome <- bodyfat$bodyfat

lambdas <- 10^seq(2, -2, by = -.1)
ridge_cv <- cv.glmnet(predictors, outcome, alpha = 0, lambda = lambdas)

best <- ridge_cv$lambda.min
best
```

```
[1] 0.06309573
```

```
best_ridge <- glmnet(predictors, outcome, alpha = 0, lambda = best)
coef(best_ridge)
```

```
4 x 1 sparse Matrix of class "dgCMatrix"
                                      s0
(Intercept)                -4.6532088
triceps.skinfold.thickness  0.6433295
thigh.circumference         0.2965686
midarm.circumference       -0.2391984
```