# Genomics Data

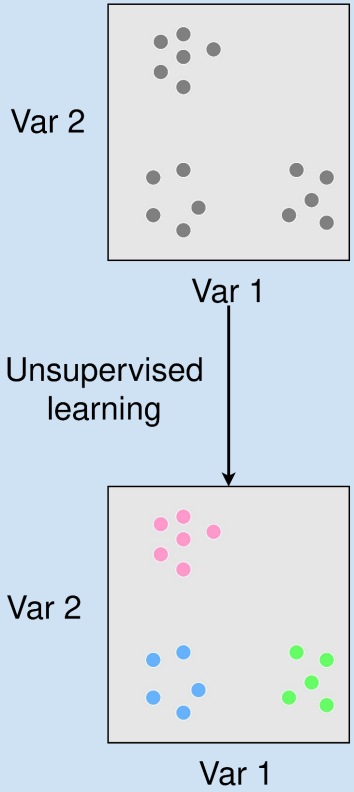
## Supervised Learning for High-Dimensional Data

TU Omics

# Supervised Learning

## Unsupervised learning

- Exploring patterns in data
- No “correct” response
- PCA, clustering

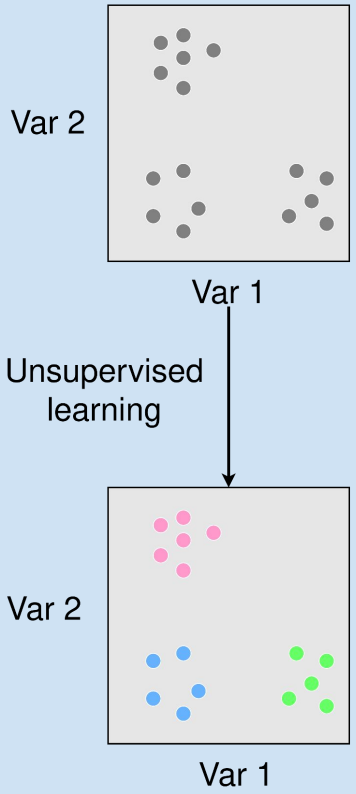


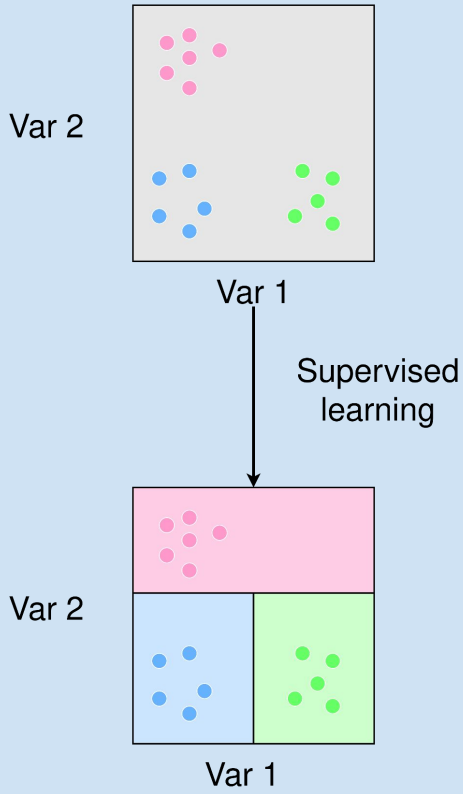
## Unsupervised learning

- Exploring patterns in data
- No “correct” response
- PCA, clustering

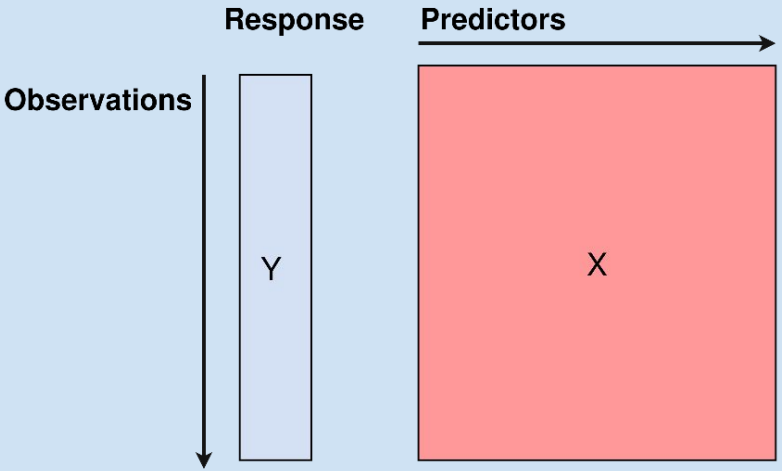
## Supervised learning

- Learn the relationship between predictors and a response
- Linear regression, random forest,...

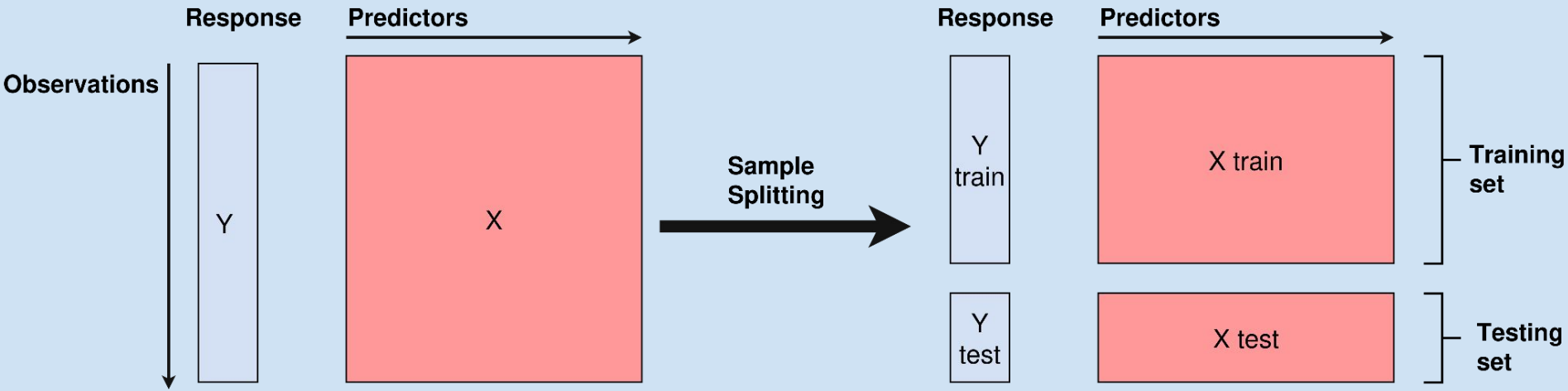


# Problem setup

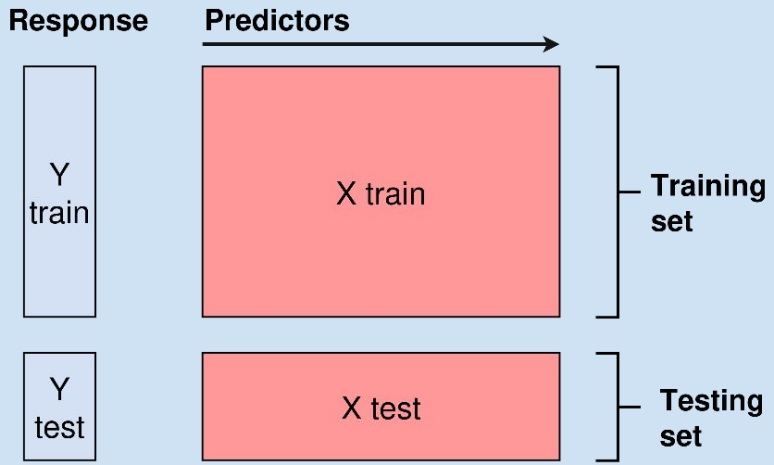


# Problem setup



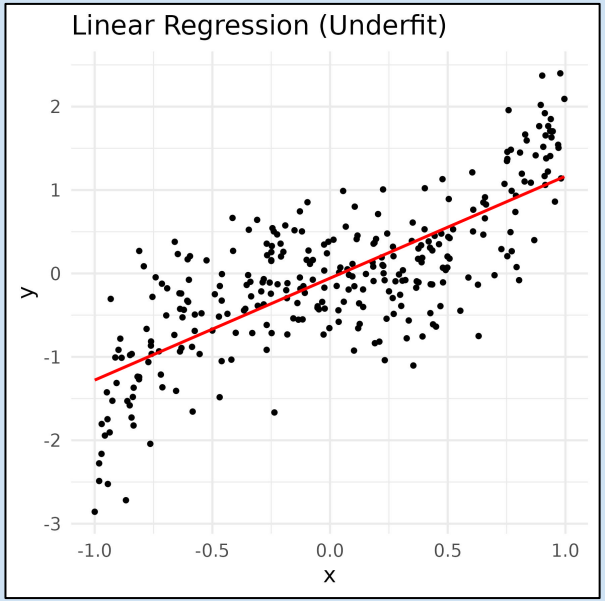
# Terminology in supervised learning

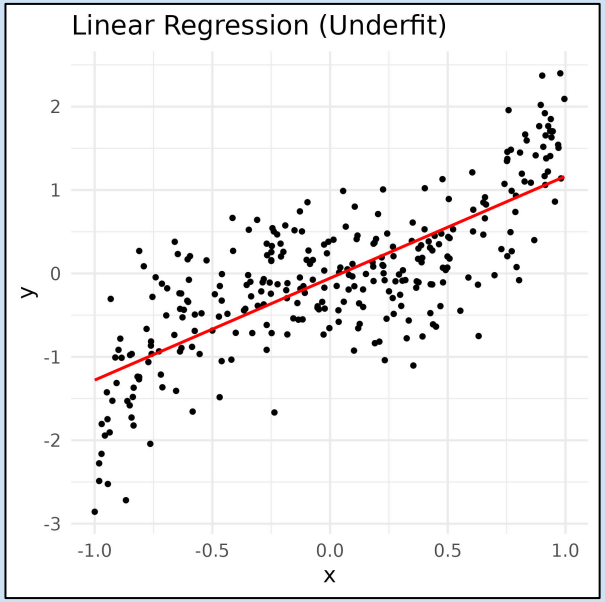
- **Observations:** set of input samples
- **Response:** variable to be predicted, Y
- **Predictors:** explanatory variables, X
- **Training set:** data used to learn the model parameters
- **Testing set:** data used to evaluate the model's performance

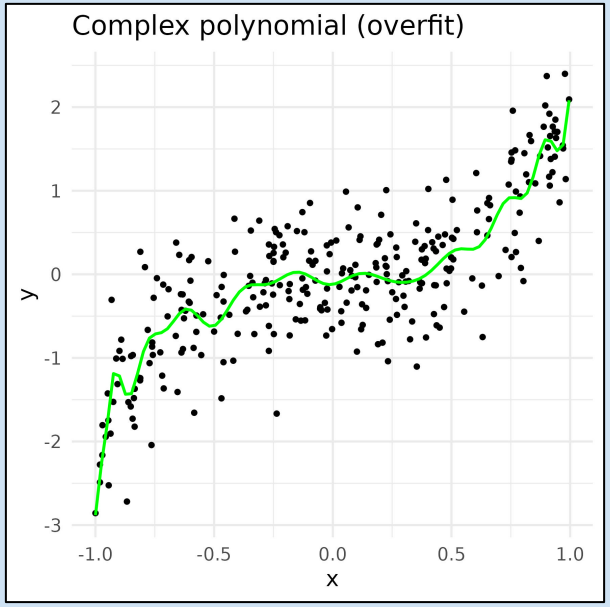


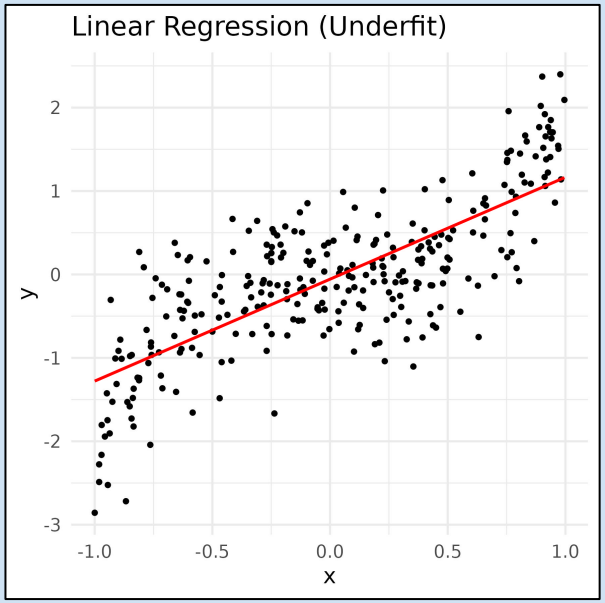
# Terminology for model evaluation

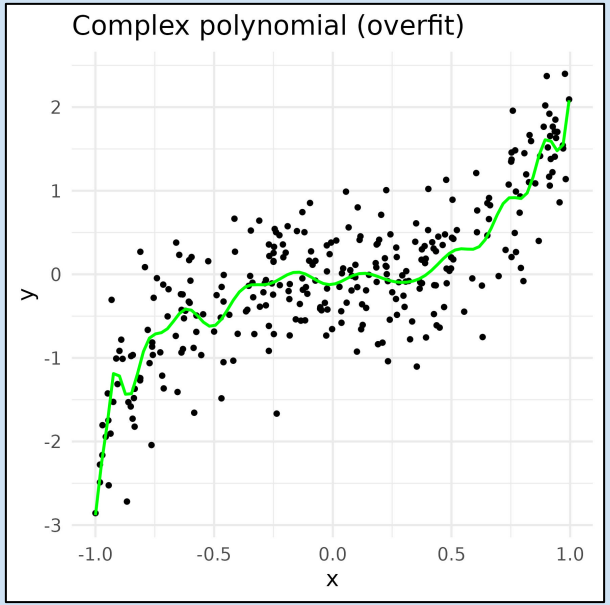
- **Loss function :** model performance metric (e.g. root mean squared error, $R^2$)
- **Bias :** systematic error in predicting true relationship
- **Variance :** model dependency on training data

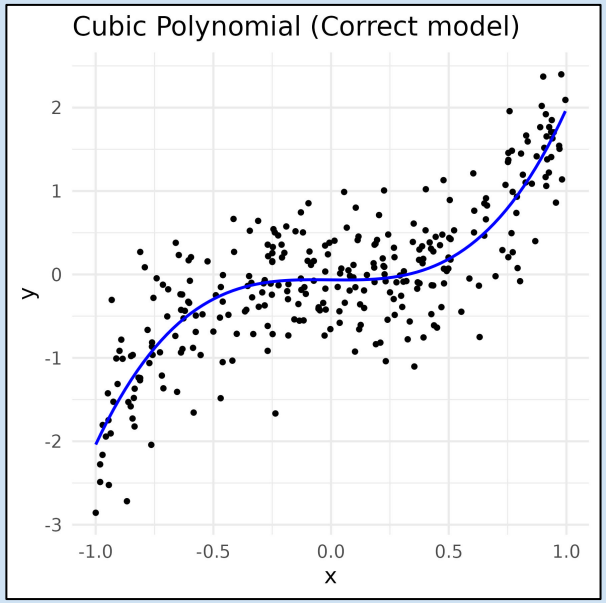

Linear Regression (Underfit)
2
1
0
-1
-2
-3
y
-1.0
-0.5
0.0
0.5
1.0
x

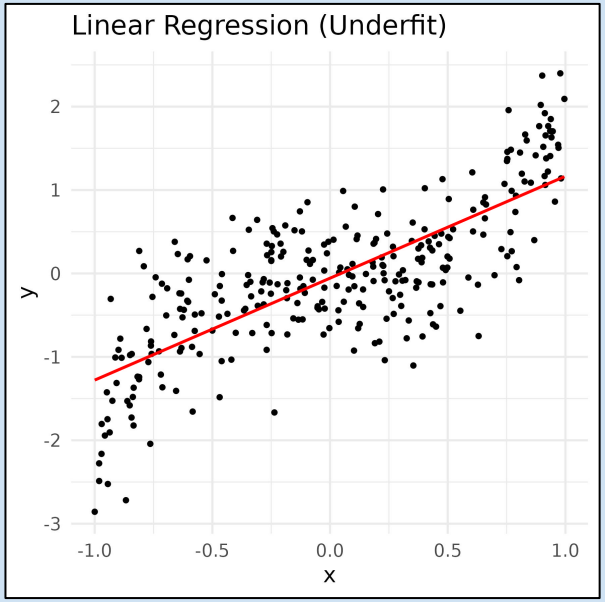

Linear Regression (Underfit)
2
1
0
-1
-2
-3
y
-1.0
-0.5
0.0
0.5
1.0
x

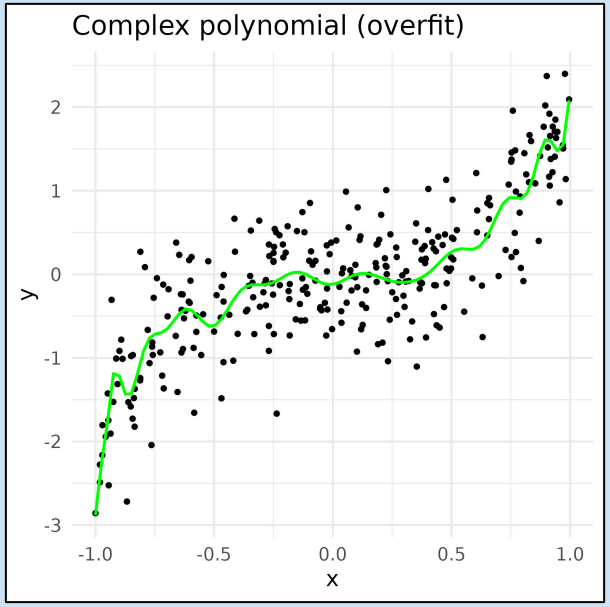

Complex polynomial (overfit)
2
1
0
-1
-2
-3
y
-1.0
-0.5
0.0
0.5
1.0
x

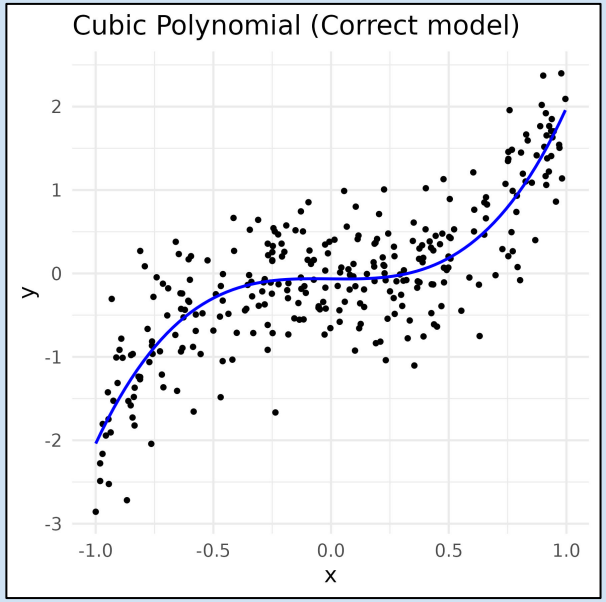

Linear Regression (Underfit)
y
x
2
1
0
-1
-2
-3
-1.0
-0.5
0.0
0.5
1.0


Complex polynomial (overfit)
y
x
2
1
0
-1
-2
-3
-1.0
-0.5
0.0
0.5
1.0


Cubic Polynomial (Correct model)
y
x
2
1
0
-1
-2
-3
-1.0
-0.5
0.0
0.5
1.0



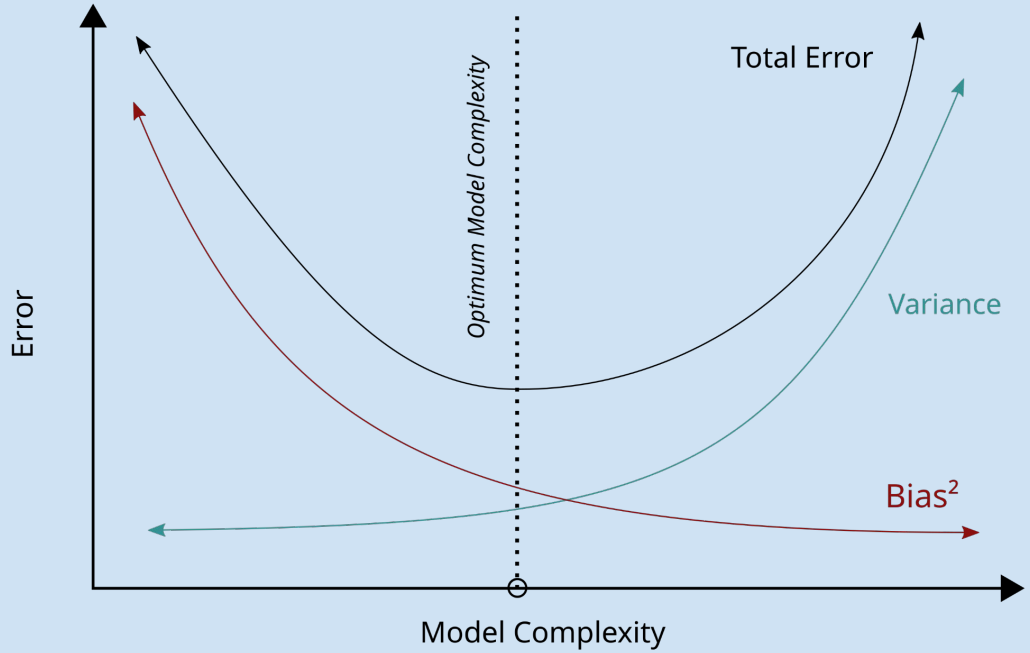




**Underfitting** = low variance, high bias

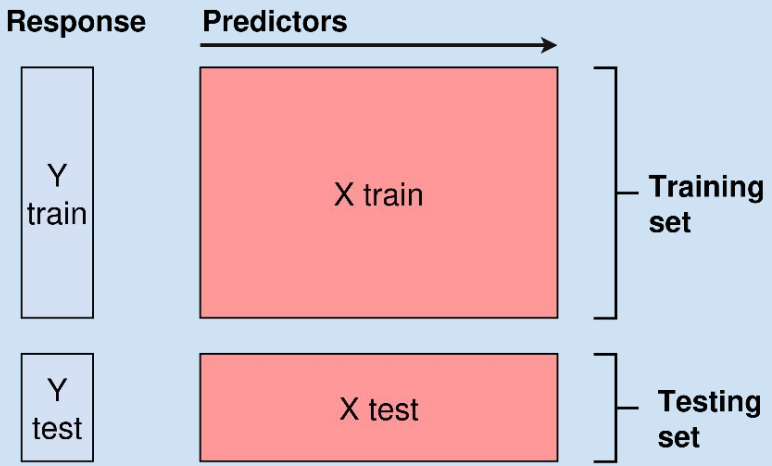
**Overfitting** = high variance, low bias

# Bias-Variance trade-off

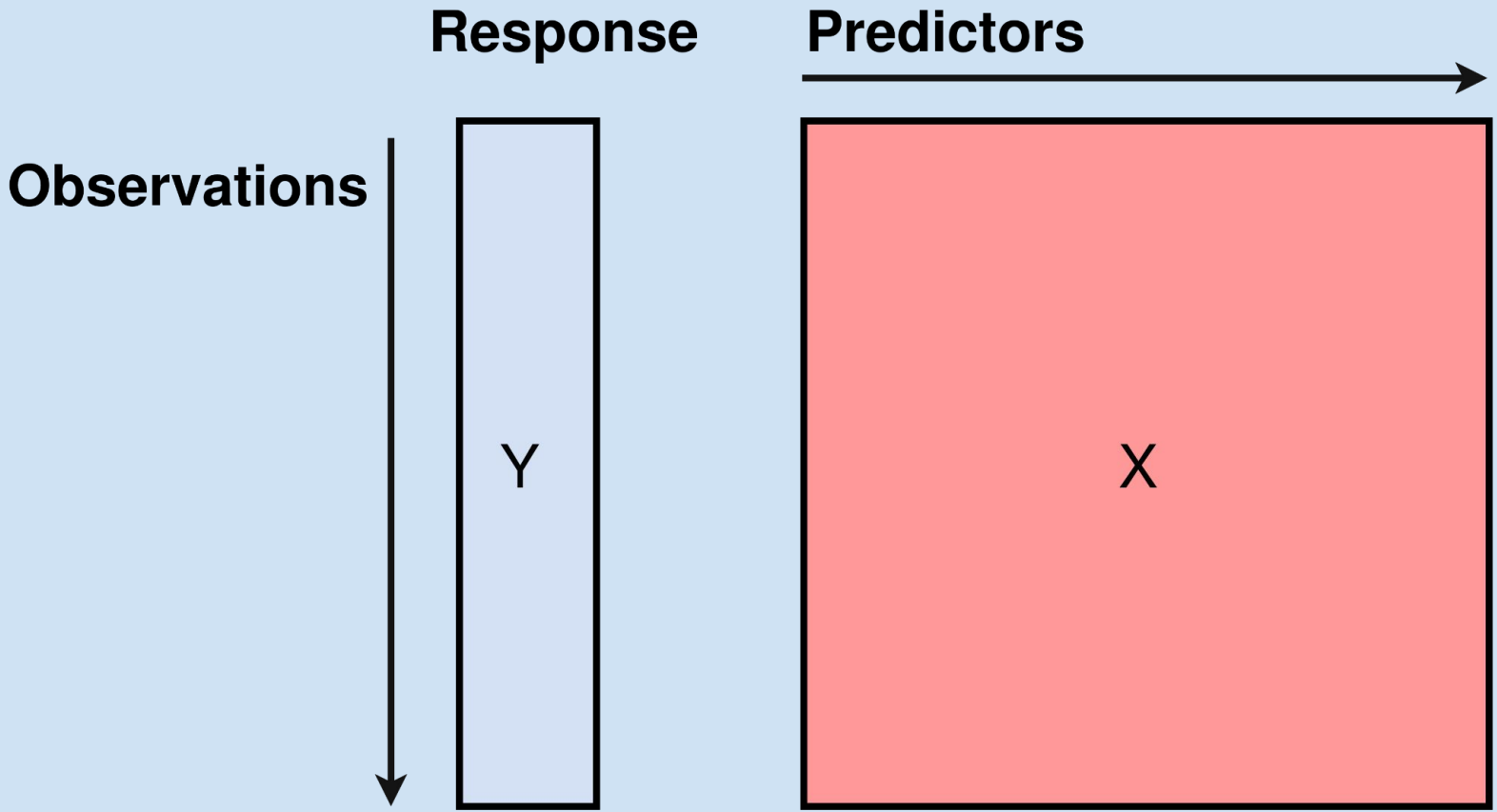


# How to evaluate true model performance?

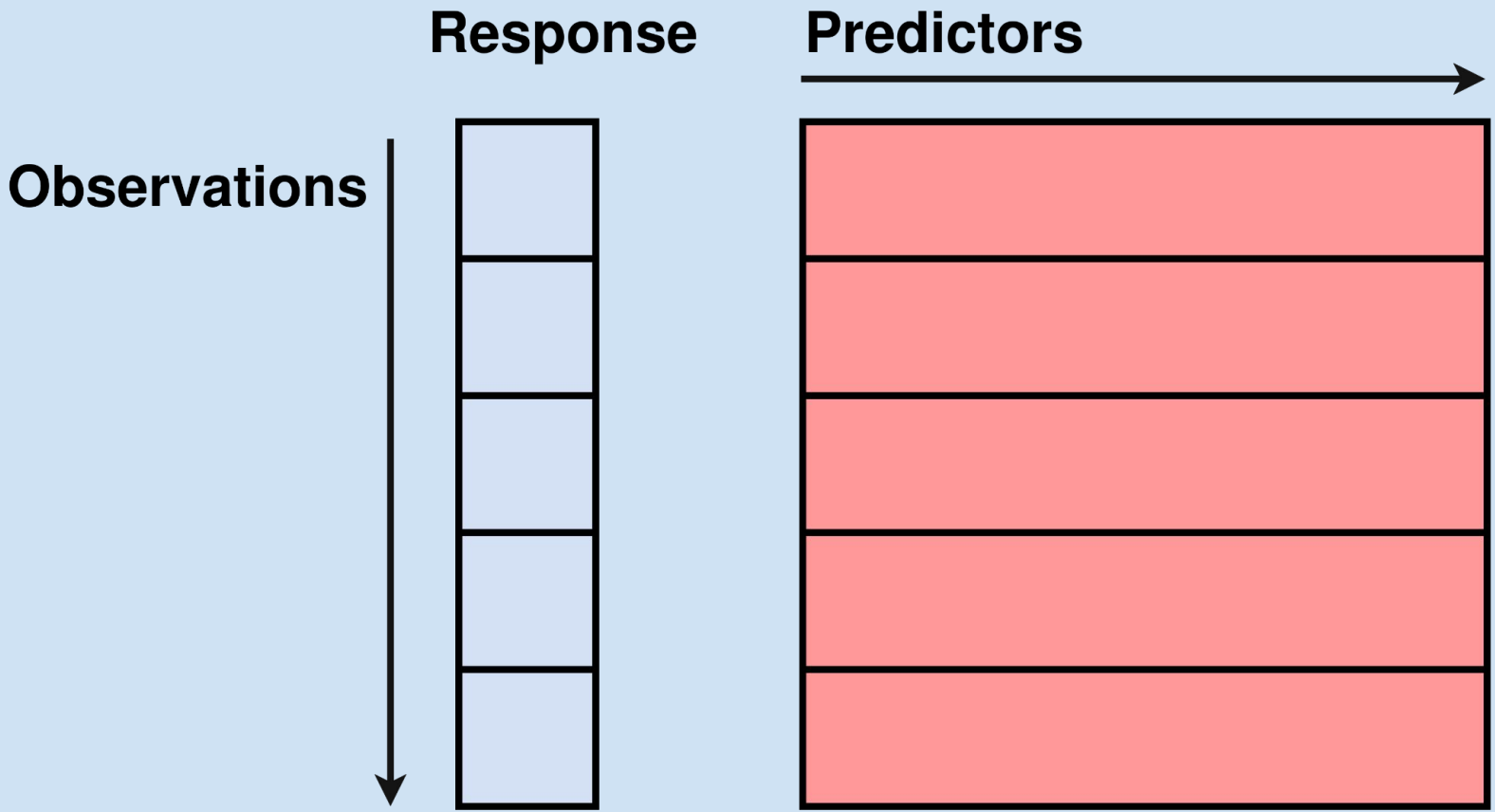
- **Naïve approach : train/test split**
- **Wasteful :** Reduce the size of the training set
- **Variable :** results will depend on the random data split



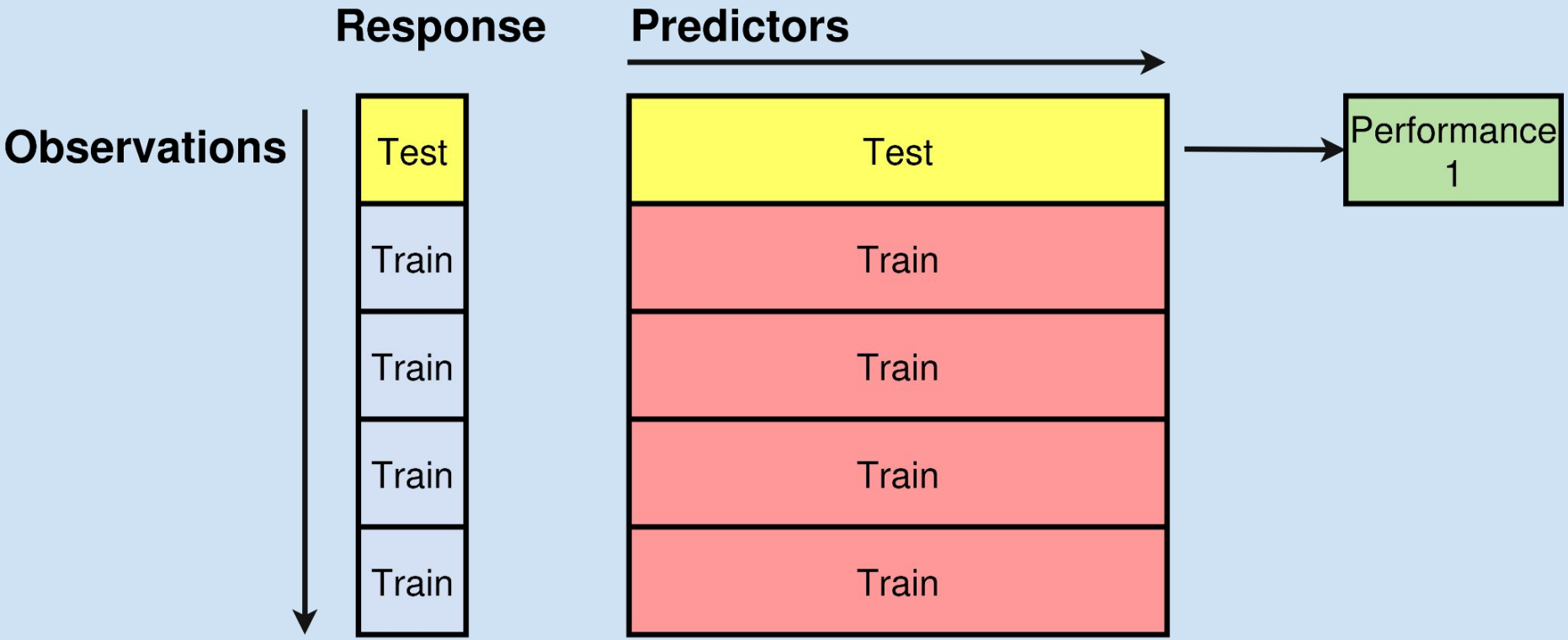
# K-fold cross-validation



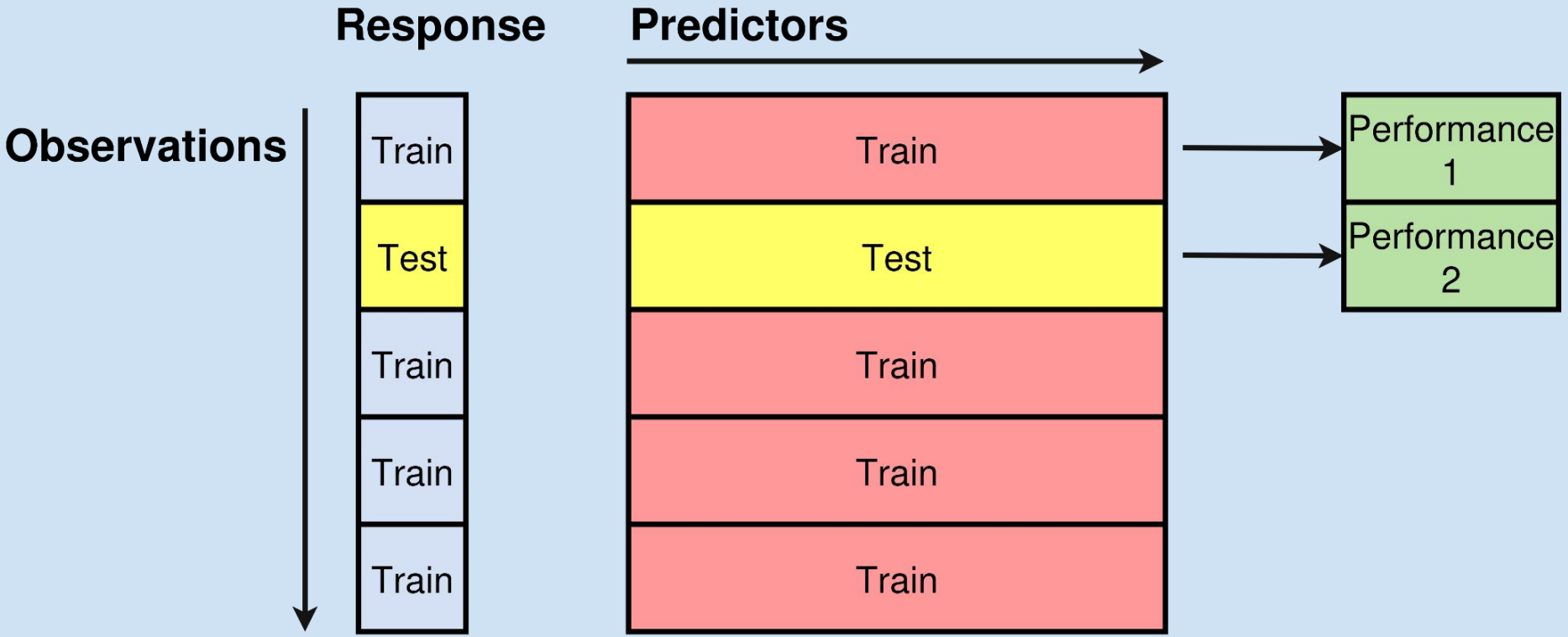
# K-fold cross-validation



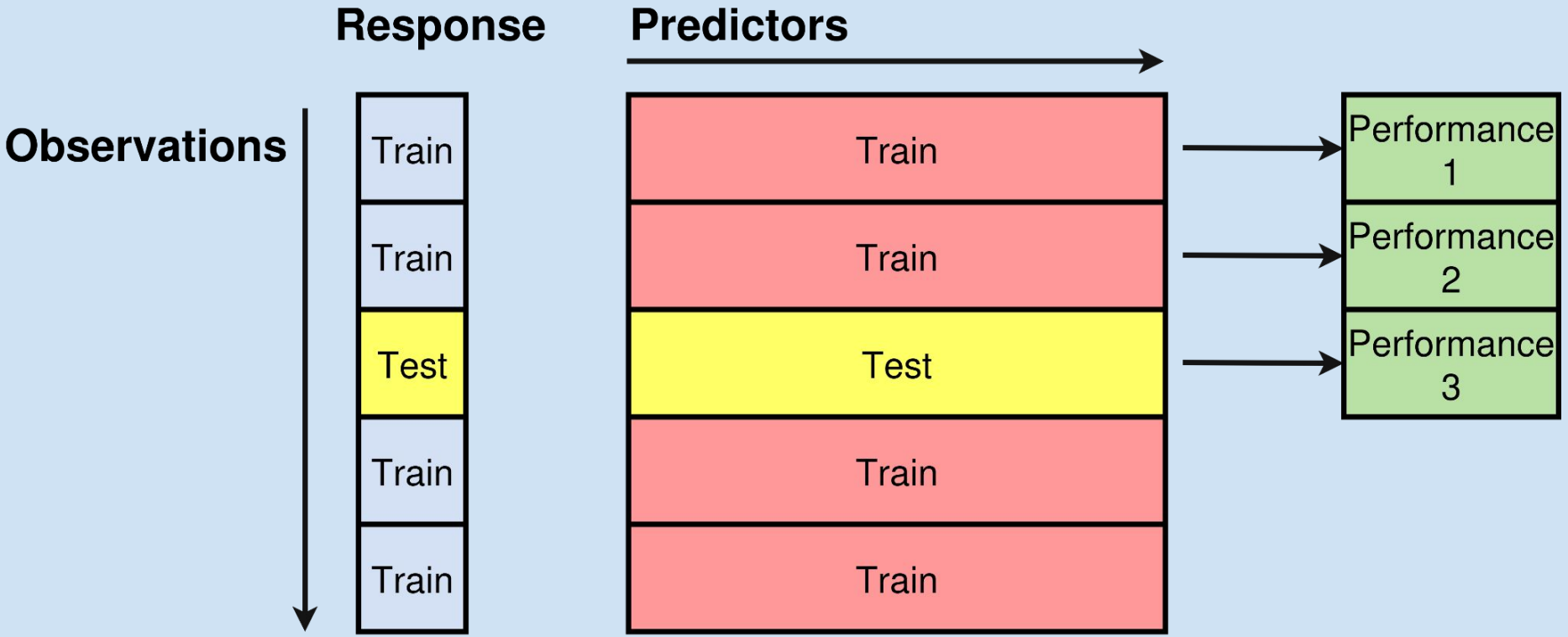
# K-fold cross-validation



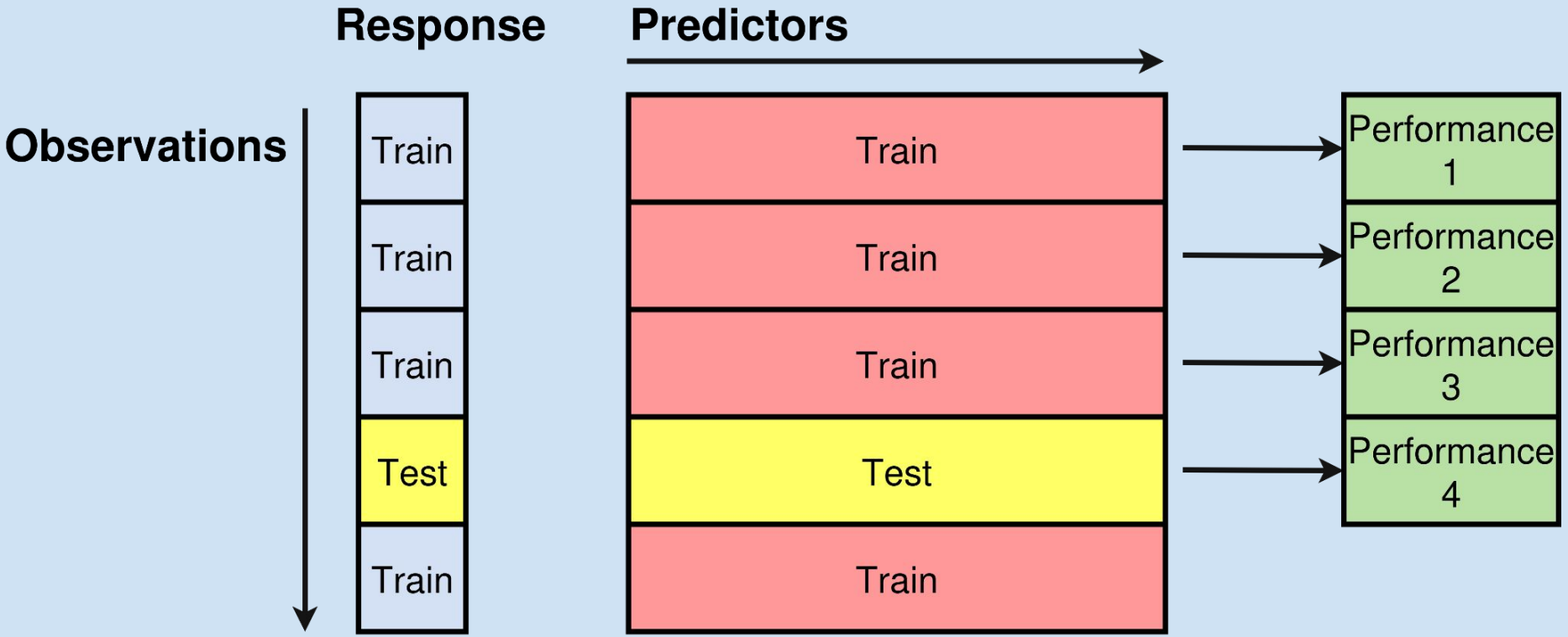
# K-fold cross-validation



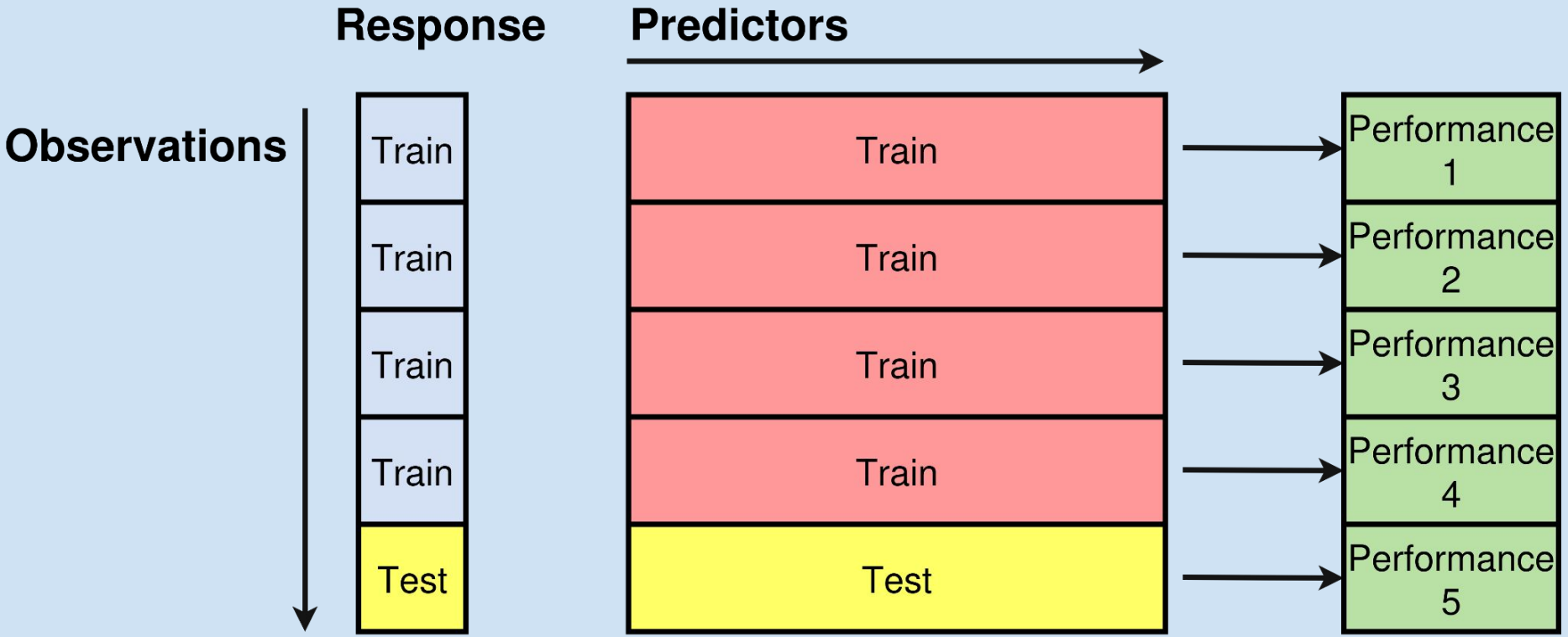
# K-fold cross-validation



# K-fold cross-validation



# K-fold cross-validation



# Introduction summary

- **Supervised learning** seeks to **learn the relationships between a response Y and predictors X** using a statistical model
- Model error can be described in terms of bias and variance
  - **Bias** is the **systematic error** in prediction
  - **Variance** is the model's **sensitivity** to the particular set of **training samples**
  - A good model is neither underfit (high bias) or overfit (high variance)
- K-fold cross validation is an efficient way to estimate the test error (model performance in unseen data)

# Supervised learning models for high-dimensional data

# Which model should we choose?

# Which model should we choose?

- **Task**
  - **Classification**: prediction of categorical response (e.g. healthy versus diseased, high-versus low treatment responder)
  - **Regression**: prediction of continuous response (e.g. antibody levels after vaccination, age of death after disease onset)
- **Hypotheses about the data**
- **Model interpretability**
  - Trade-off between complexity and interpretability
- **High-dim data has special problems**
  - overfitting, multicollinearity, signal sparsity, computational burden…

# Linear regression

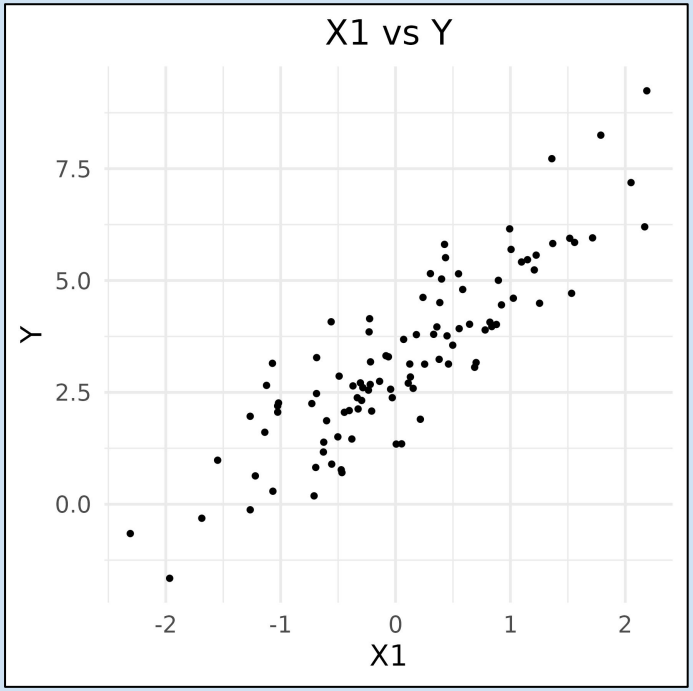
# Motivation : Linear regression

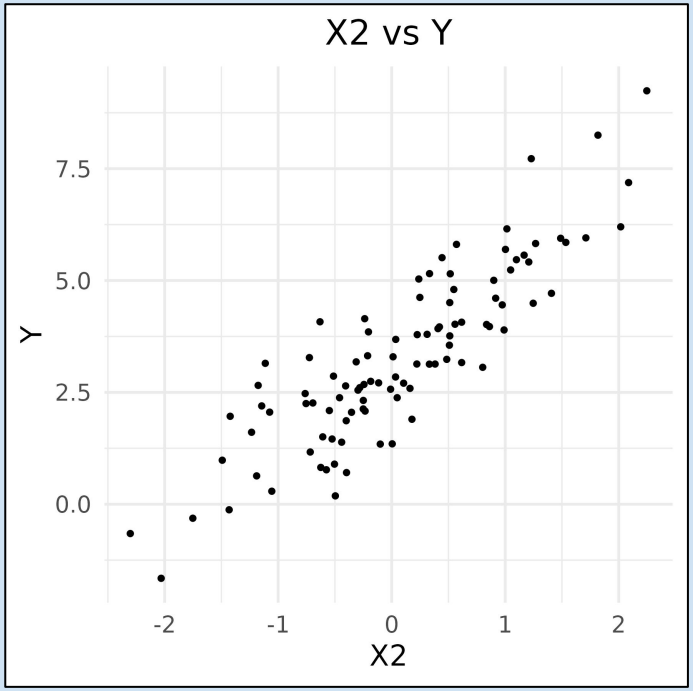
Simple data simulation

- response y
- two **highly correlated predictors x1, x2**

| y | x1 | x2 |
|---|---|---|
| 4.077859 | -0.56047565 | -0.63151630 |
| 3.852058 | -0.23017749 | -0.20448912 |
| 5.852272 | 1.55870831 | 1.53403913 |
| 3.684211 | 0.07050839 | 0.03575413 |
| 2.844236 | 0.12928774 | 0.03412588 |
| 5.953883 | 1.71506499 | 1.71056221 |

# Both x1 and x2 are predictive of y



# Regression of y on x1 and x2

**Model : y =** $\alpha 0 + \alpha 1 * x1 + \alpha 2 * x2$

**What is wrong here?**

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.12389    0.09857  31.693   <2e-16 ***
x1           2.05465    0.95083   2.161   0.0332 *
x2          -0.26485    0.94944  -0.279   0.7809
```

# Regression of y on x1 and x2

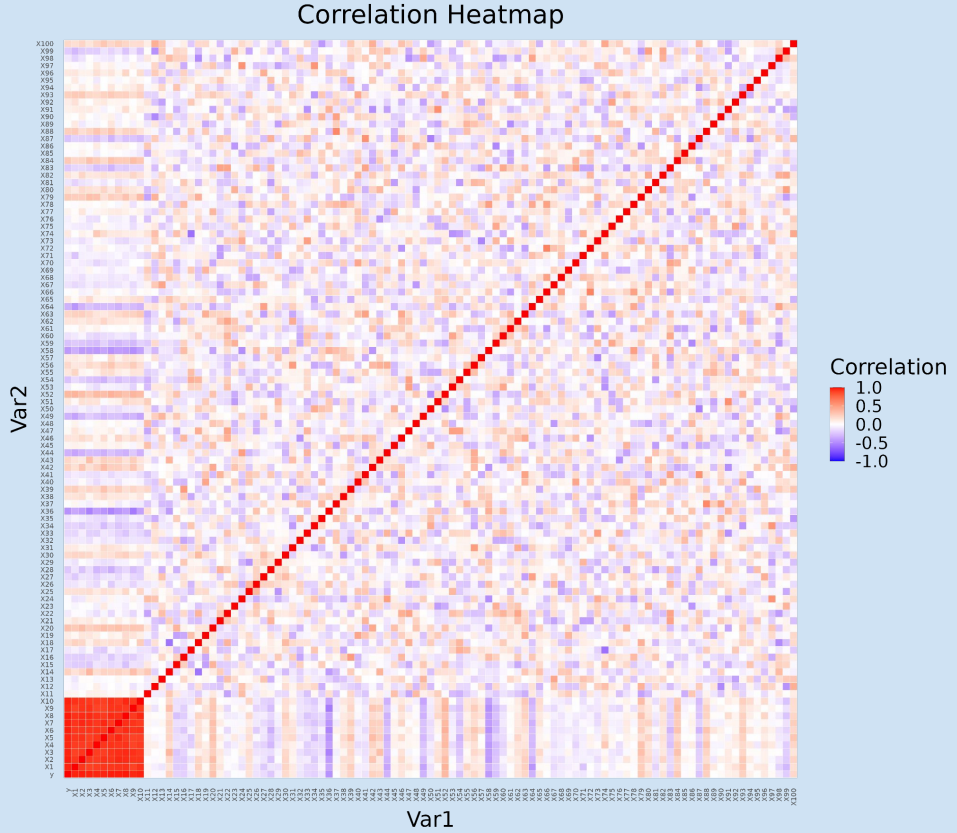
**Model : y =** $\alpha 0 + \alpha 1 * x1 + \alpha 2 * x2$

**What is wrong here?**

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.12389    0.09857  31.693   <2e-16 ***
x1           2.05465    0.95083   2.161   0.0332 *
x2          -0.26485    0.94944  -0.279   0.7809
```

# What happens in high-dimension

- 100 predictors, 30 observations
- **First 10 predictors** multicollinear and correlated with true coefficients = 2
- **Other 90 are noise**



# Linear regression output

```
Coefficients: (71 not defined because of singularities)
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   6.09614        NaN     NaN      NaN
X1           17.19978        NaN     NaN      NaN
X2           -0.70897        NaN     NaN      NaN
X3           35.43475        NaN     NaN      NaN
X4          -16.38034        NaN     NaN      NaN
X5          -13.82239        NaN     NaN      NaN
```

…

```
X28         -10.63382        NaN     NaN      NaN
X29          -5.95420        NaN     NaN      NaN
X30                NA         NA      NA       NA
X31                NA         NA      NA       NA
X32                NA         NA      NA       NA
```

# Some problems with linear regression

- OLS estimates are **not interpretable in multicollinear** settings
  - Coefficients are nonsensical and standard errors are high
- **In addition :** OLS is not possible when n<p!

# Regularised methods

## Ridge, lasso, elastic net regression

# Regression models minimise a loss function

Regression methods estimate parameters by minimising a loss function :

$$\hat{\boldsymbol{\beta}} = \arg\min_{\beta} \underbrace{\mathcal{L}(\boldsymbol{\beta}; \boldsymbol{X}, y)}_{\text{loss function}},$$

For example :

- **Squared error loss (linear regression):**

$$\mathcal{L}(\beta; X, y) = \sum_{i=1}^{n} (y_i - x_i^\top \beta)^2$$

  - Measures the sum of squared differences between observed and predicted responses.

# Regularised methods

Regularised methods add a **penalty term** that *shrinks* the parameter estimates:

$$\hat{\boldsymbol{\beta}} = \arg\min_{\beta} \underbrace{\mathcal{L}(\boldsymbol{\beta}; \boldsymbol{X}, y)}_{\text{loss function}} + \underbrace{\lambda\, \Omega(\boldsymbol{\beta})}_{\text{penalty}},$$

- $\Omega(\boldsymbol{\beta})$ is a **penalty function** encoding preferences on the parameters (e.g., small magnitude, sparsity, smoothness).
- $\lambda \geq 0$ controls the strength of the penalty: higher $\lambda$ imposes stronger regularisation.

# Regularised methods

Different penalty functions have different properties:

- **Ridge regression:**

$$\Omega(\boldsymbol{\beta}) = \sum_{j=1}^{p} \beta_j^2 = \|\boldsymbol{\beta}\|_2^2$$

- **Lasso regression:**

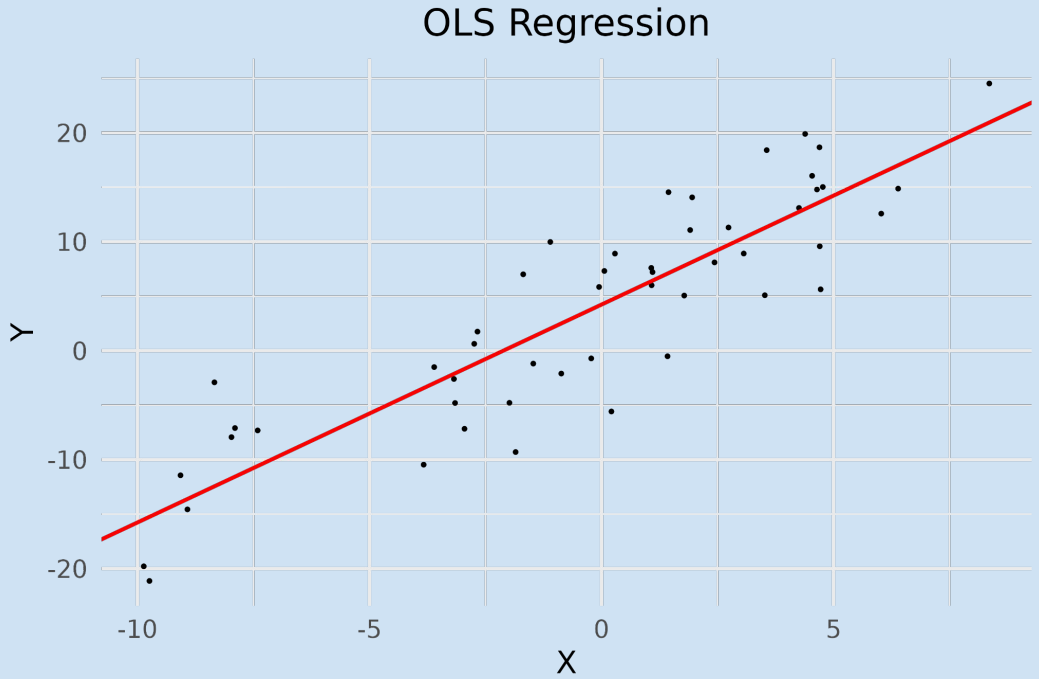
$$\Omega(\boldsymbol{\beta}) = \sum_{j=1}^{p} |\beta_j| = \|\boldsymbol{\beta}\|_1$$

- **Elastic net regression:**

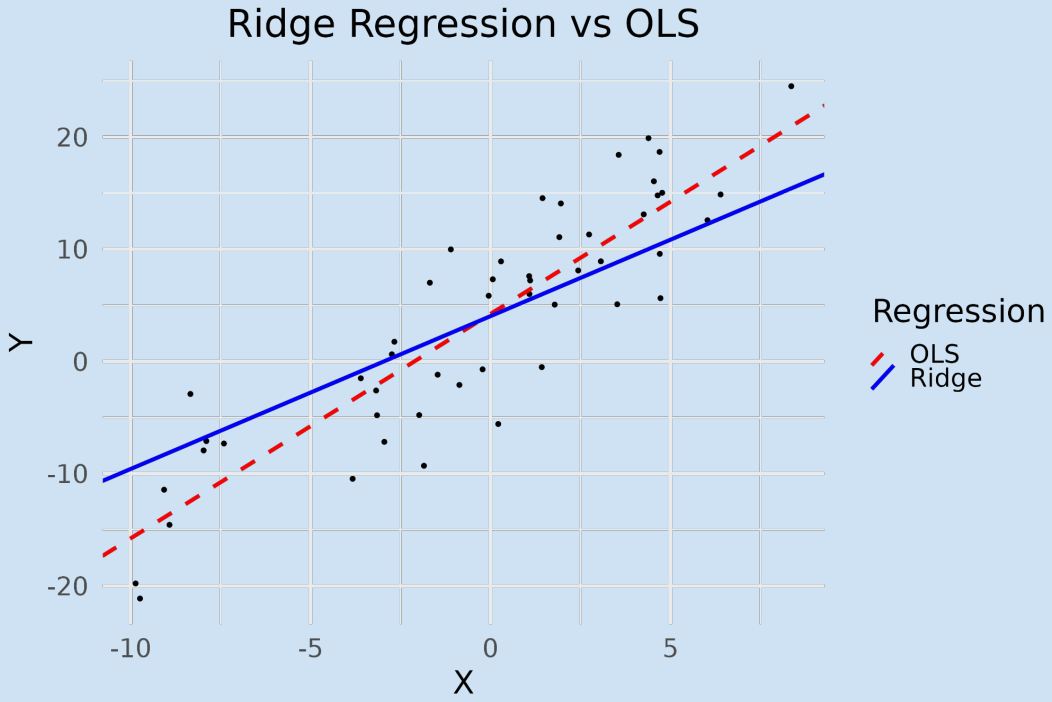
$$\Omega(\boldsymbol{\beta}) = \alpha\|\boldsymbol{\beta}\|_1 + (1-\alpha)\|\boldsymbol{\beta}\|_2^2, \quad \alpha \in [0,1]$$

# What does this mean in practice?

- OLS based on 1 predictor of Y

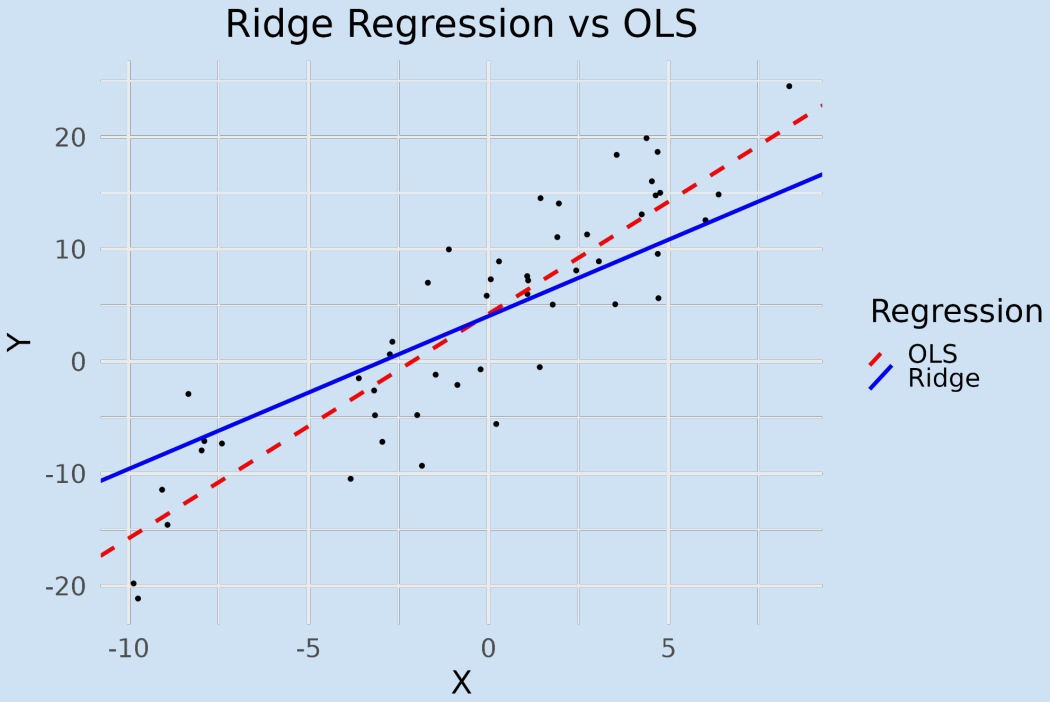


# What does this mean in practice?



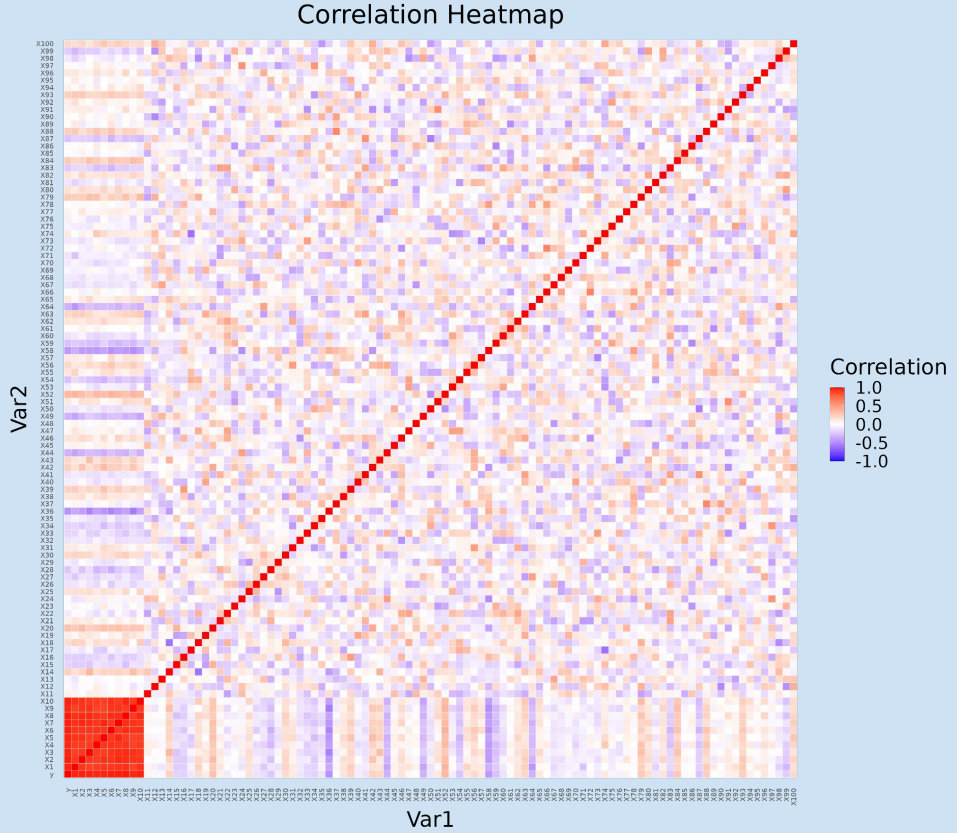
# What does this mean in practice?

- Ridge regression *shrinks* the coefficient of X
- The line is **less specific to the training data**



# What happens in high-dimension

- 100 predictors, 30 observations
- **First 10 predictors** multicollinear and correlated with true coefficients = 2
- **Other 90 are noise**



# Linear regression output

```
Coefficients: (71 not defined because of singularities)
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   6.09614        NaN     NaN      NaN
X1           17.19978        NaN     NaN      NaN
X2           -0.70897        NaN     NaN      NaN
X3           35.43475        NaN     NaN      NaN
X4          -16.38034        NaN     NaN      NaN
X5          -13.82239        NaN     NaN      NaN
```

…

```
X28         -10.63382        NaN     NaN      NaN
X29          -5.95420        NaN     NaN      NaN
X30                NA         NA      NA       NA
X31                NA         NA      NA       NA
X32                NA         NA      NA       NA
```

# Ridge regression, small penalty term

| Predictor | Coefficient |
|---|---|
| (Intercept) | 4.199203850 |
| X1 | 1.722511160 |
| X2 | 1.563138138 |
| X3 | 1.618239257 |
| X4 | 1.545498028 |
| X5 | 1.647217751 |
| X6 | 1.653767428 |
| X7 | 1.607043560 |
| X8 | 1.602866955 |
| X9 | 1.541996515 |
| X10 | 1.522906678 |
| X11 | -0.083351967 |
| X12 | -0.217714240 |
| X13 | 0.041424806 |

- **Ridge regression:**

$$\Omega(\boldsymbol{\beta}) = \sum_{j=1}^{p} \beta_j^2 = \|\boldsymbol{\beta}\|_2^2$$

# Ridge regression, small penalty term

| Predictor | Coefficient |
|---|---|
| (Intercept) | 4.199203850 |
| X1 | 1.722511160 |
| X2 | 1.563138138 |
| X3 | 1.618239257 |
| X4 | 1.545498028 |
| X5 | 1.647217751 |
| X6 | 1.653767428 |
| X7 | 1.607043560 |
| X8 | 1.602866955 |
| X9 | 1.541996515 |
| X10 | 1.522906678 |
| X11 | -0.083351967 |
| X12 | -0.217714240 |
| X13 | 0.041424806 |

- Coefficients for first 10 predictors are **underestimated** (penalised), but reasonable

# Ridge regression, large penalty term

| Predictor | Coefficient |
|---|---|
| (Intercept) | 5.824742599 |
| X1 | 0.920329883 |
| X2 | 0.874152089 |
| X3 | 0.900291651 |
| X4 | 0.876054764 |
| X5 | 0.927766602 |
| X6 | 0.921230947 |
| X7 | 0.895915991 |
| X8 | 0.892476809 |
| X9 | 0.860369210 |
| X10 | 0.887186751 |
| X11 | -0.068315490 |
| X12 | -0.169504537 |
| X13 | 0.078989765 |

# Ridge regression, large penalty term

| Predictor | Coefficient |
|---|---|
| (Intercept) | 5.824742599 |
| X1 | 0.920329883 |
| X2 | 0.874152089 |
| X3 | 0.900291651 |
| X4 | 0.876054764 |
| X5 | 0.927766602 |
| X6 | 0.921230947 |
| X7 | 0.895915991 |
| X8 | 0.892476809 |
| X9 | 0.860369210 |
| X10 | 0.887186751 |
| X11 | -0.068315490 |
| X12 | -0.169504537 |
| X13 | 0.078989765 |

- Coefficients even more penalised
- However, coefficients for the noise predictors are still non-zero
- In fact, the ridge penalty **cannot** shrink coefficients all the way to zero

# Lasso regression - regularisation and variable selection

- Lasso regression solves this problem by using a penalty which allows coefficients to be **shrunk to exactly 0**
- This allows uninformative features to be removed from the model and for informative features to be easier to interpret!

- **Lasso regression:**

$$\Omega(\boldsymbol{\beta}) = \sum_{j=1}^{p} |\beta_j| = \|\boldsymbol{\beta}\|_1$$

# Lasso regression - small penalty term

| Predictor | Coefficient |
|---|---|
| (Intercept) | 3.2138461 |
| X1 | 2.9146758 |
| X2 | 2.6461699 |
| X3 | 1.6067177 |
| X4 | 0.8452229 |
| X5 | 2.7256999 |
| X6 | 2.0209490 |
| X7 | 0.3937528 |
| X8 | 2.2824053 |
| X9 | 2.1998183 |
| X10 | 2.2016101 |
| X11 | 0.0000000 |
| X12 | 0.0000000 |
| X13 | 0.0000000 |

# Lasso regression - small penalty term

| Predictor | Coefficient |
|---|---|
| (Intercept) | 3.2138461 |
| X1 | 2.9146758 |
| X2 | 2.6461699 |
| X3 | 1.6067177 |
| X4 | 0.8452229 |
| X5 | 2.7256999 |
| X6 | 2.0209490 |
| X7 | 0.3937528 |
| X8 | 2.2824053 |
| X9 | 2.1998183 |
| X10 | 2.2016101 |
| X11 | 0.0000000 |
| X12 | 0.0000000 |
| X13 | 0.0000000 |

- Now, we only have non-zero coefficients for the first 10 predictors
- The model is more ***sparse and interpretable***

# Lasso regression - large penalty term

| Predictor | Coefficient |
|---|---|
| (Intercept) | 5.7733620 |
| X1 | 0.3473931 |
| X2 | 0.0000000 |
| X3 | 0.1216453 |
| X4 | 0.0000000 |
| X5 | 0.1664646 |
| X6 | 2.9953686 |
| X7 | 0.0000000 |
| X8 | 0.0000000 |
| X9 | 1.6641831 |
| X10 | 1.3486280 |
| X11 | 0.0000000 |
| X12 | 0.0000000 |
| X13 | 0.0000000 |

# Lasso regression - large penalty term

| Predictor | Coefficient |
|---|---|
| (Intercept) | 5.7733620 |
| X1 | 0.3473931 |
| X2 | 0.0000000 |
| X3 | 0.1216453 |
| X4 | 0.0000000 |
| X5 | 0.1664646 |
| X6 | 2.9953686 |
| X7 | 0.0000000 |
| X8 | 0.0000000 |
| X9 | 1.6641831 |
| X10 | 1.3486280 |
| X11 | 0.0000000 |
| X12 | 0.0000000 |
| X13 | 0.0000000 |

- A slight problem : if predictors are **multicollinear**, lasso tends to **select one** and discards the rest
- This could lead to **erroneous interpretations** of which predictors are important

# Elastic Net regression - mix of lasso and ridge

- Elastic net regression solves this problem by *combining* the penalty terms of lasso and ridge

- **Elastic net regression:**

$$\Omega(\boldsymbol{\beta}) = \alpha\|\boldsymbol{\beta}\|_1 + (1-\alpha)\|\boldsymbol{\beta}\|_2^2, \quad \alpha \in [0,1]$$

# Elastic Net regression - large penalty term

| Predictor | Coefficient |
|---|---|
| (Intercept) | 3.591873 |
| X1 | 2.031664 |
| X2 | 1.921551 |
| X3 | 1.659616 |
| X4 | 1.575456 |
| X5 | 2.013029 |
| X6 | 1.930660 |
| X7 | 1.108460 |
| X8 | 1.760805 |
| X9 | 1.775735 |
| X10 | 1.899880 |
| X11 | 0.000000 |
| X12 | 0.000000 |
| X13 | 0.000000 |

# Elastic Net regression - large penalty term

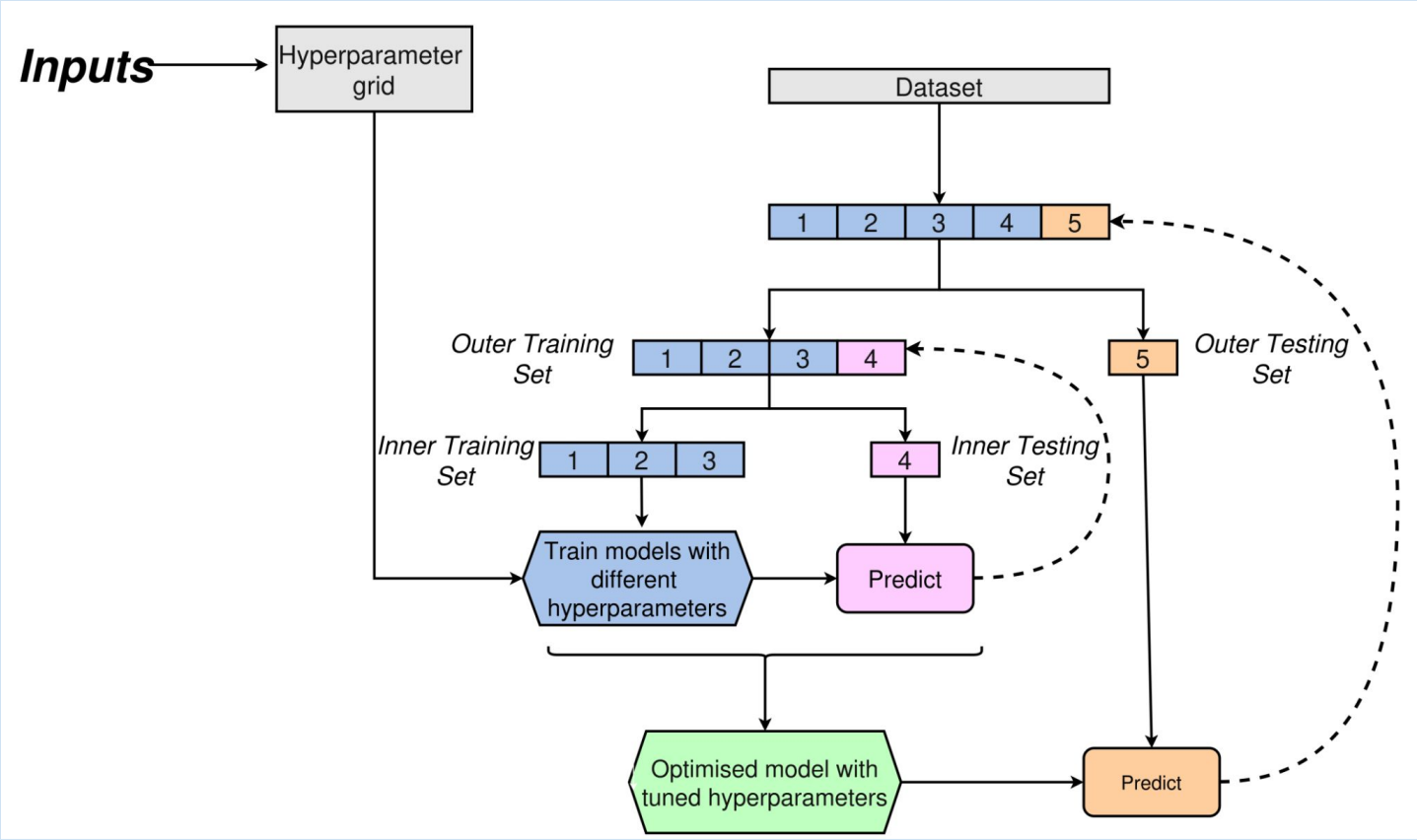
| Predictor | Coefficient |
|---|---|
| (Intercept) | 3.591873 |
| X1 | 2.031664 |
| X2 | 1.921551 |
| X3 | 1.659616 |
| X4 | 1.575456 |
| X5 | 2.013029 |
| X6 | 1.930660 |
| X7 | 1.108460 |
| X8 | 1.760805 |
| X9 | 1.775735 |
| X10 | 1.899880 |
| X11 | 0.000000 |
| X12 | 0.000000 |
| X13 | 0.000000 |

- Elastic net combines the properties of ridge and lasso
- **Ridge :** tends to shrink multicollinear predictors in proportion, but cannot *select* predictors
- **Lasso :** can select predictors, but may discard important ones when multicollinear
- **Elastic Net :** Can select groups of multicollinear predictors

# Summary on regularised methods

| | Handles overfitting? | Handles multicollinearity? | Variable selection? |
|---|---|---|---|
| **Linear regression** | no | Nonsensical estimates, large error | no |
| **Ridge regression** | Penalty term reduces dependency of model estimates on data | yes | no |
| **Lasso regression** | Penalty term reduces dependency of model estimates on data | Tendency to discard predictors in multicollinear groups | yes |
| **Elastic net regression** | Penalty term reduces dependency of model estimates on data | Selects groups of multicollinear predictors | yes |

# How to choose the hyperparameters? Nested K-fold CV



# Latent Variable Methods
## Principal Component Regression

# Latent variable approaches

- ***Latent*** **= hidden**
- A **latent variable cannot be observed** but can be estimated from observable variables
- **Example :** socio-economic status
  - This variable cannot be directly measured but could be constructed from a combination of e.g. income, education level, occupation, housing quality

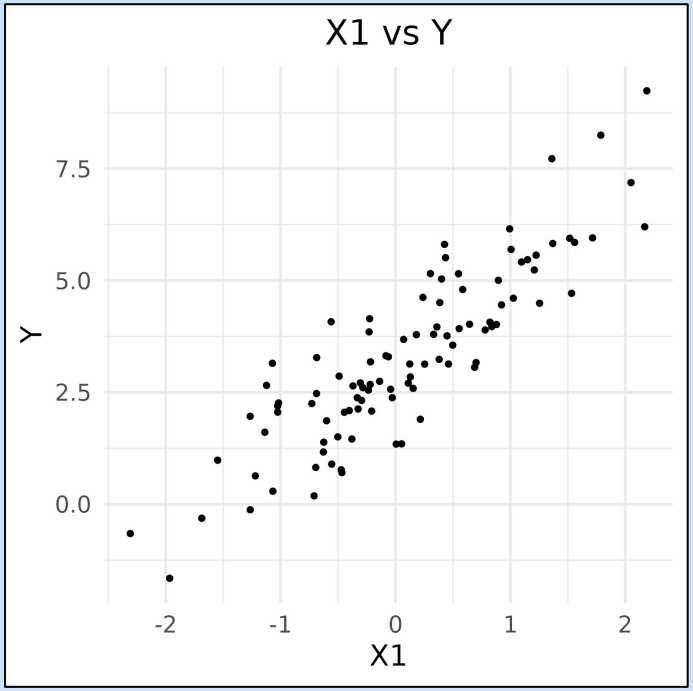
# Motivation : Linear regression

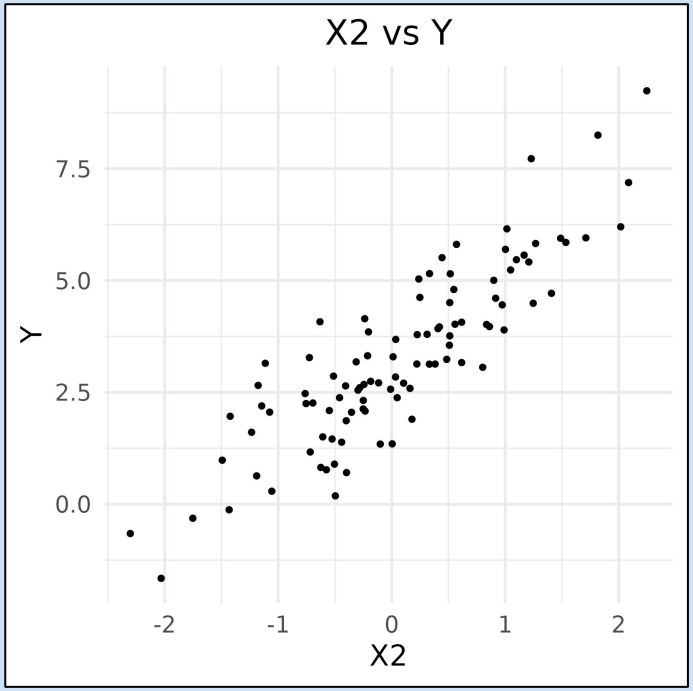
Simple data simulation

- response y
- two **highly correlated predictors x1, x2**

| y | x1 | x2 |
|---|---|---|
| 4.077859 | -0.56047565 | -0.63151630 |
| 3.852058 | -0.23017749 | -0.20448912 |
| 5.852272 | 1.55870831 | 1.53403913 |
| 3.684211 | 0.07050839 | 0.03575413 |
| 2.844236 | 0.12928774 | 0.03412588 |
| 5.953883 | 1.71506499 | 1.71056221 |

# Both x1 and x2 are predictive of y



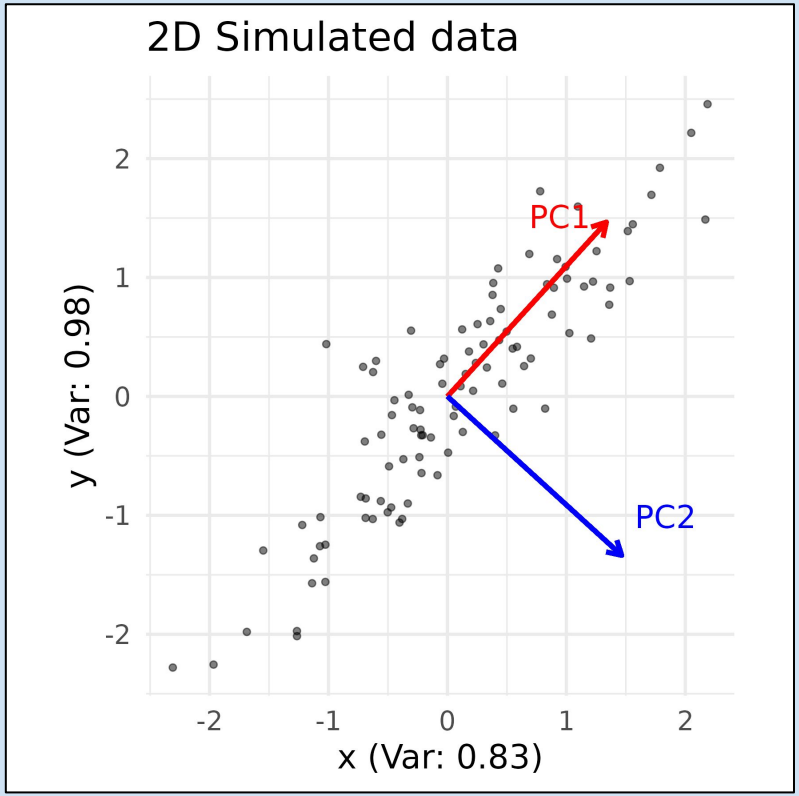
# Linear regression fails with multicollinear predictors

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.12389    0.09857  31.693   <2e-16 ***
x1           2.05465    0.95083   2.161   0.0332 *
x2          -0.26485    0.94944  -0.279   0.7809
```

**Idea :** transform the data to capture signal in a smaller number of uncorrelated variables

# Principal Component Analysis reminder

- **Principal Components** are weighted combinations of the original variables
- In this example :
  - PC1 = 0.67x + 0.74y
  - PC2 = 0.74x - 0.67y
- The first PCs capture the majority of the variability in the data
- By construction, PCs are completely uncorrelated (orthogonality)



# Principal Component Regression

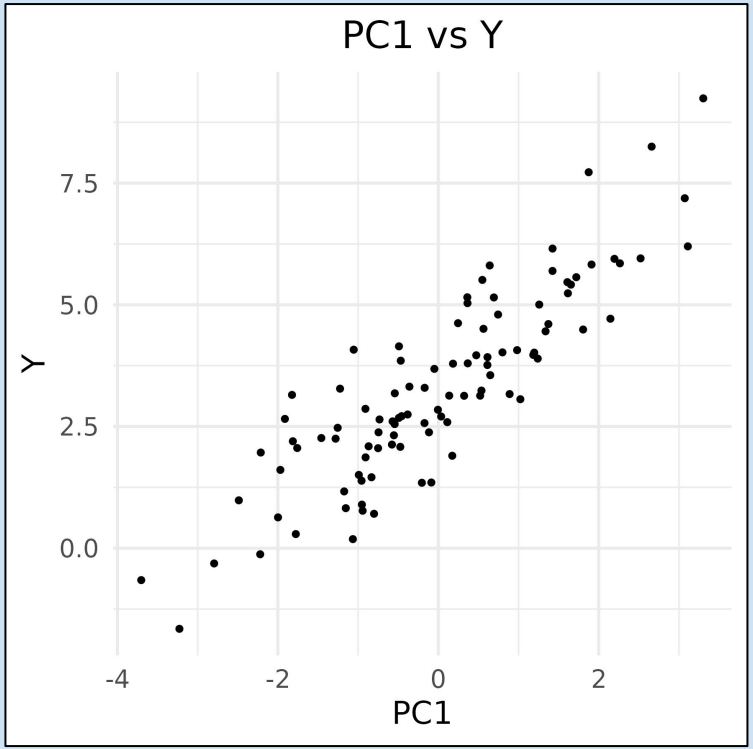
1. **Transform the data** with PCA
2. **Reduce the dimension** by discarding PCs which do not explain much of the variability in the data (scree plot)
3. **Perform regression** using reduced number of PCs instead of original predictors

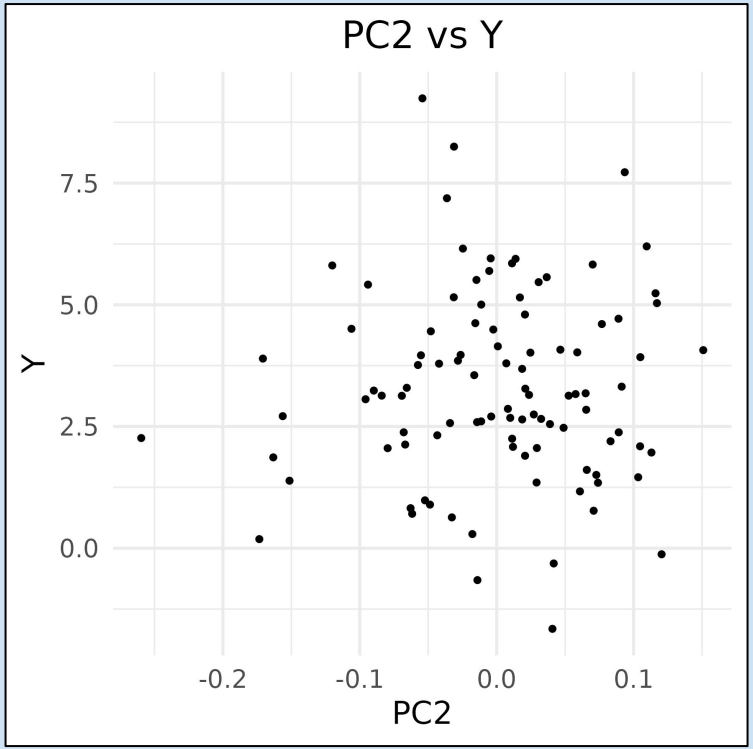
# Transformation of the data

- PC1 = 0.71*x1 + 0.71*x2
- PC2 = 0.71*x1 - 0.71*x2
- cor(PC1, PC2) = 0

| y | x1 | x2 | PC1 | PC2 |
|---|---|---|---|---|
| 4.077859 | -0.56047565 | -0.63151630 | -1.054901952 | 0.046499773 |
| 3.852058 | -0.23017749 | -0.20448912 | -0.468365196 | -0.028310420 |
| 5.852272 | 1.55870831 | 1.53403913 | 2.263632953 | 0.011188532 |
| 3.684211 | 0.07050839 | 0.03575413 | -0.049405709 | 0.018578749 |
| 2.844236 | 0.12928774 | 0.03412588 | -0.005133536 | 0.065372635 |
| 5.953883 | 1.71506499 | 1.71056221 | 2.521446325 | -0.004383515 |

# PC1 is related to Y but PC2 is not



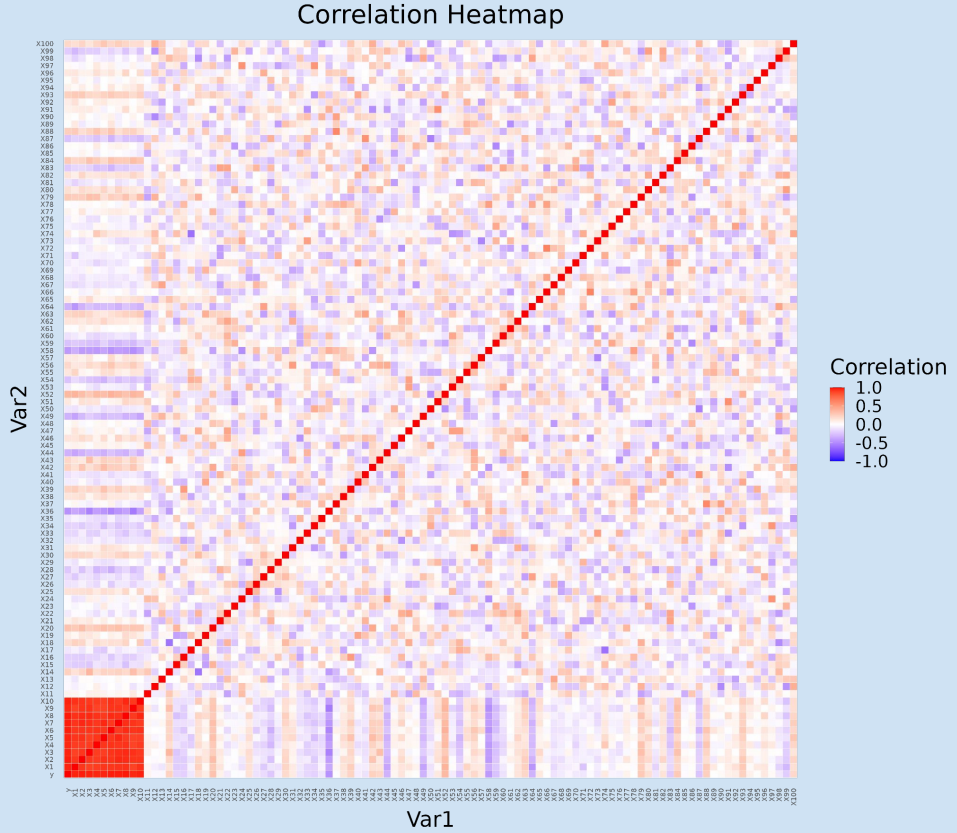
# Linear regression on principal components

**Model : y =** $\alpha 0 + \alpha 1 * PC1 + \alpha 2 * PC2$

| | Estimate | Std. Error | t value | Pr(>\|t\|) | |
|---|---|---|---|---|---|
| (Intercept) | 3.30128 | 0.09513 | 34.703 | <2e-16 | *** |
| PC1 | 1.20502 | 0.06770 | 17.799 | <2e-16 | *** |
| PC2 | 0.89752 | 1.27661 | 0.703 | 0.484 | |

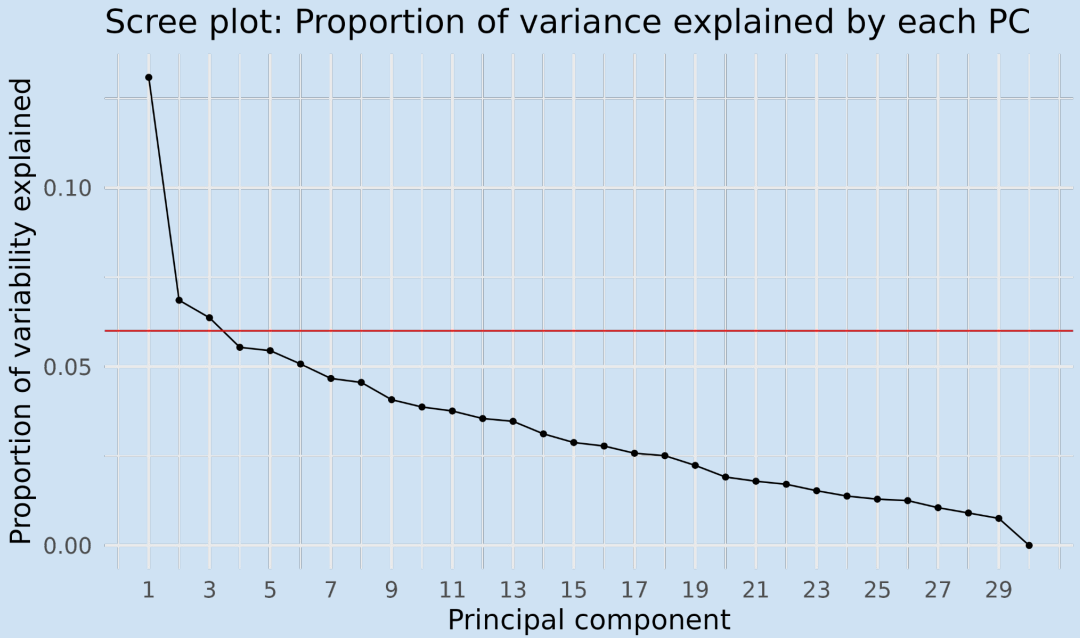
# PCR in high-dimension

- 100 predictors, 30 observations
- **First 10 predictors** multicollinear and correlated with true coefficients = 2
- **Other 90 are noise**

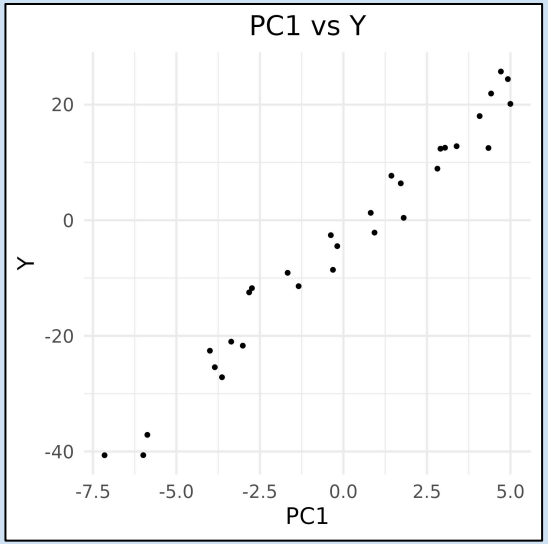


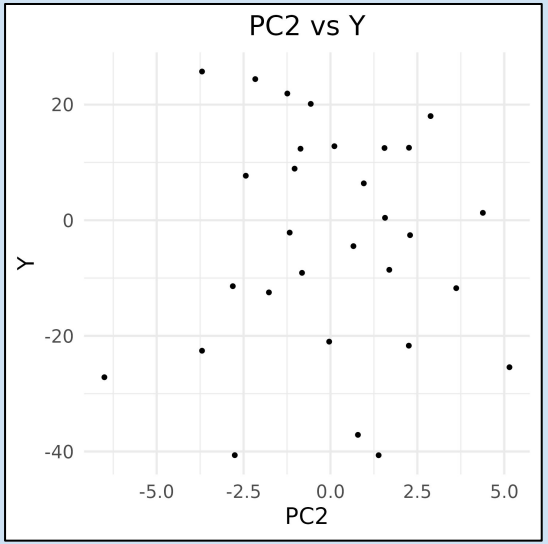
# PCR in high-dimension : choosing the number of PCs

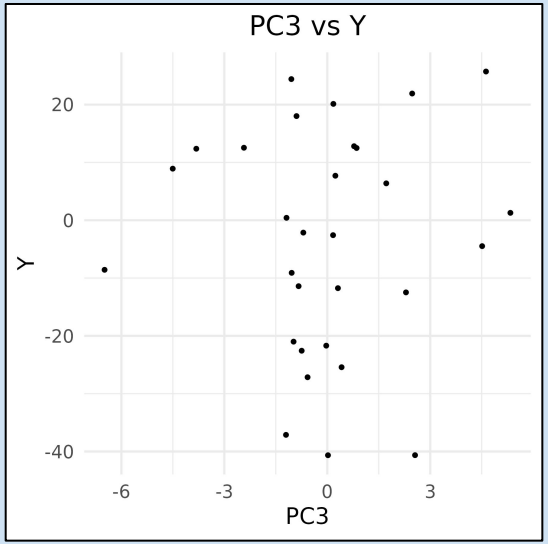
- First few PCs capture a lot of the variance
- We will keep 3 for the regression (elbow method)



# Examine the new predictors



# Linear regression output

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -3.78726    0.61919  -6.116 1.82e-06 ***
PC1          5.29136    0.17409  30.394  < 2e-16 ***
PC2         -0.03108    0.24054  -0.129    0.898
PC3          0.34242    0.24961   1.372    0.182
```

# How can we interpret the results?

Look at the ***loading weights*** for the important PCs

These give the contributions of each variable to the PC

- As expected, the 10 truly associated predictors all contribute positively to PC1, the most predictive PC

| | Predictor | Loading |
|---|---|---|
| X8 | X8 | 0.26683184 |
| X9 | X9 | 0.26669971 |
| X3 | X3 | 0.26490462 |
| X1 | X1 | 0.26336880 |
| X10 | X10 | 0.26304947 |
| X7 | X7 | 0.26167679 |
| X2 | X2 | 0.26164914 |
| X6 | X6 | 0.25900956 |
| X5 | X5 | 0.25889924 |
| X4 | X4 | 0.25757469 |
| X80 | X80 | -0.12902728 |
| X49 | X49 | 0.12614039 |
| X45 | X45 | -0.12319149 |

# Maths of PC regression

**Principal Component Regression**

- $X \in \mathcal{R}^{n \times p}$ matrix with $n$ rows of observations of $p$ columns of predictors,
- $y$ a continuous outcome,
- both predictors and outcome centered, scaled

# Maths of PC regression

1. **Find** $PC_1 \in \mathbb{R}^{n\times 1}$, a linear combination of the predictors:

$$PC_1 = Xw_1,$$

where

- $w_1 \in \mathbb{R}^{p\times 1}$ is the **weight** vector that defines the linear combination of the predictors, chosen to **maximize the variance** of the score:

$$w_1 = \arg\max_{\|w\|=1} \mathrm{Var}(Xw)$$

2. **Adjust** (deflate) $X$ to remove the contribution of the first principal component:

$$p_1 = \frac{X^\top PC_1}{PC_1^\top PC_1} \qquad \text{(PC loading vector)}$$

$$X_1 = X - PC_1 p_1^\top.$$

# Maths of PC regression

3. **Repeat** to find additional principal component scores $PC_2, PC_3, \ldots$ using deflated predictors $X_1, X_2, \ldots$.
4. **Keep** $p^*$ informative PCs (choose with cumulative proportion of variance explained)
5. **Regression step:** regress $y$ on $PC = (PC_1, ..., PC_{p^*})$

# A couple of problems with PC regression

- **Explaining the variability in the predictors does not guarantee explaining the variability in the response!**
  - In this example, PC2 and PC3 were useless predictors

- **Every predictor contributes a non-zero weight to every PC**
  - i.e. the more useless predictors, the more we accumulate noise in the PCs
  - Perhaps this could be **solved with regularisation** on the loading weights?

# Latent Variable Methods

## Partial Least Squares

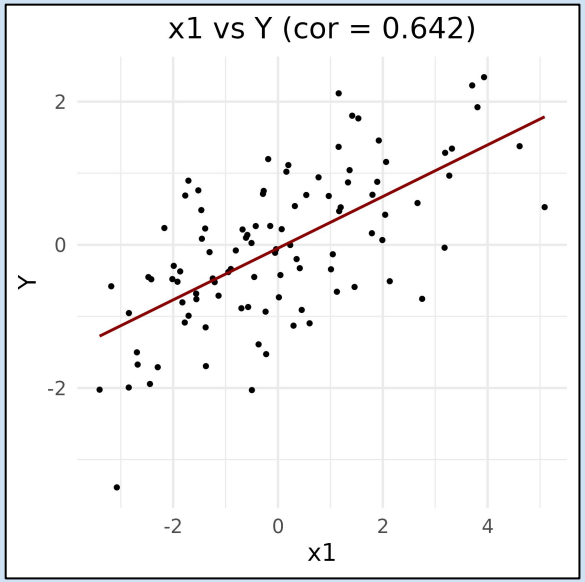
# Idea of partial least squares

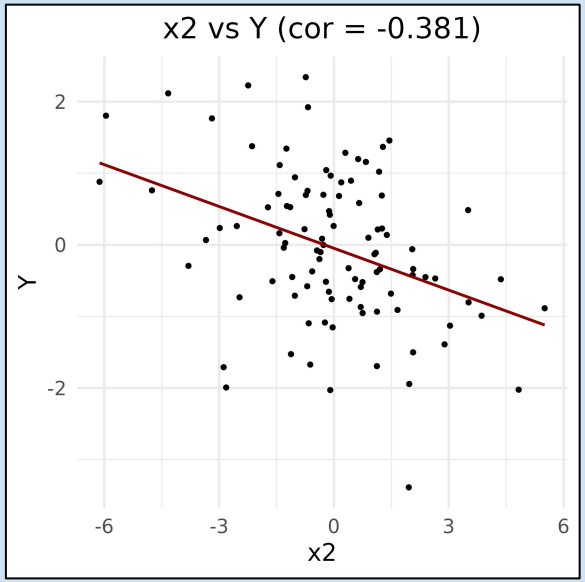
- PCA forms latent variables by linear combinations which best explain the variability in X
- **PLS forms latent variables by linear combinations which best explain the variability in Y!**
    - Like PCA, these latent variables are **uncorrelated**

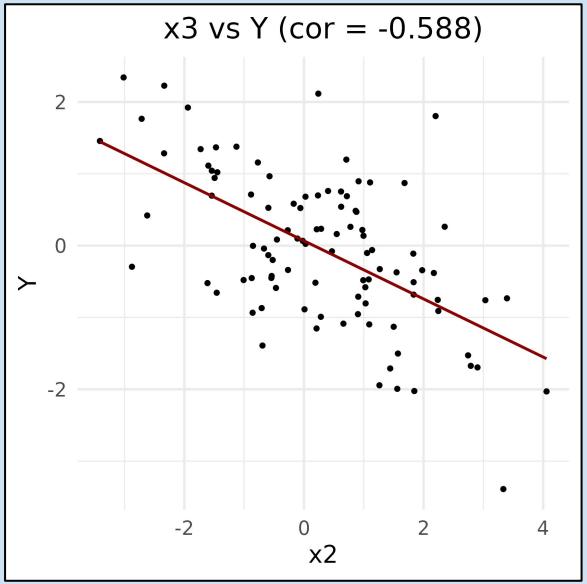
# Partial Least Squares Regression

1. **Transform the data** into latent variables (LVs) capturing the covariability between the predictors and response
2. **Reduce the dimension** by discarding LVs which do not explain much of the co-variability in the data (scree plot)
3. **Perform regression** using reduced number of LVs instead of original predictors

# Low dimensional example

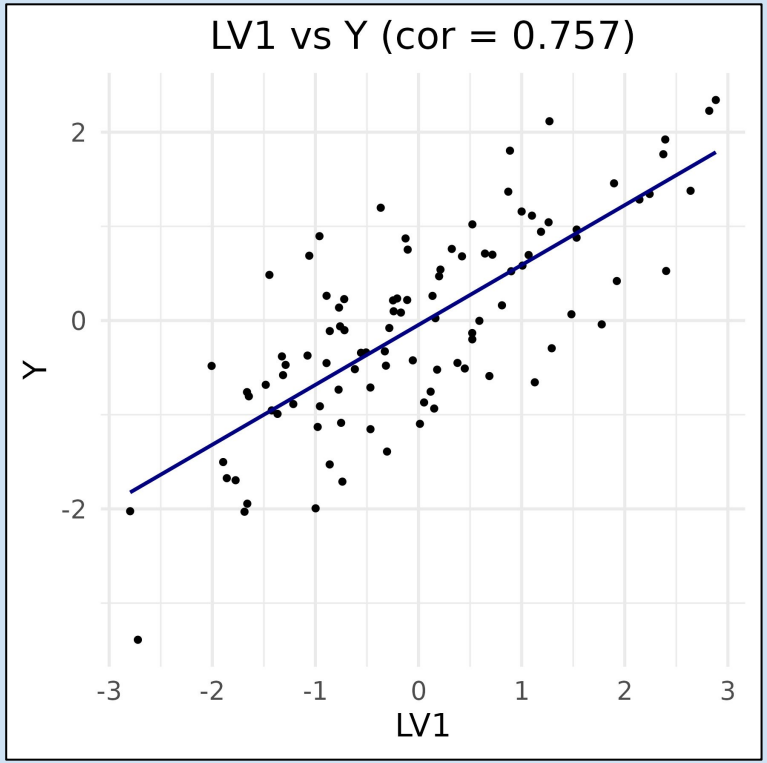


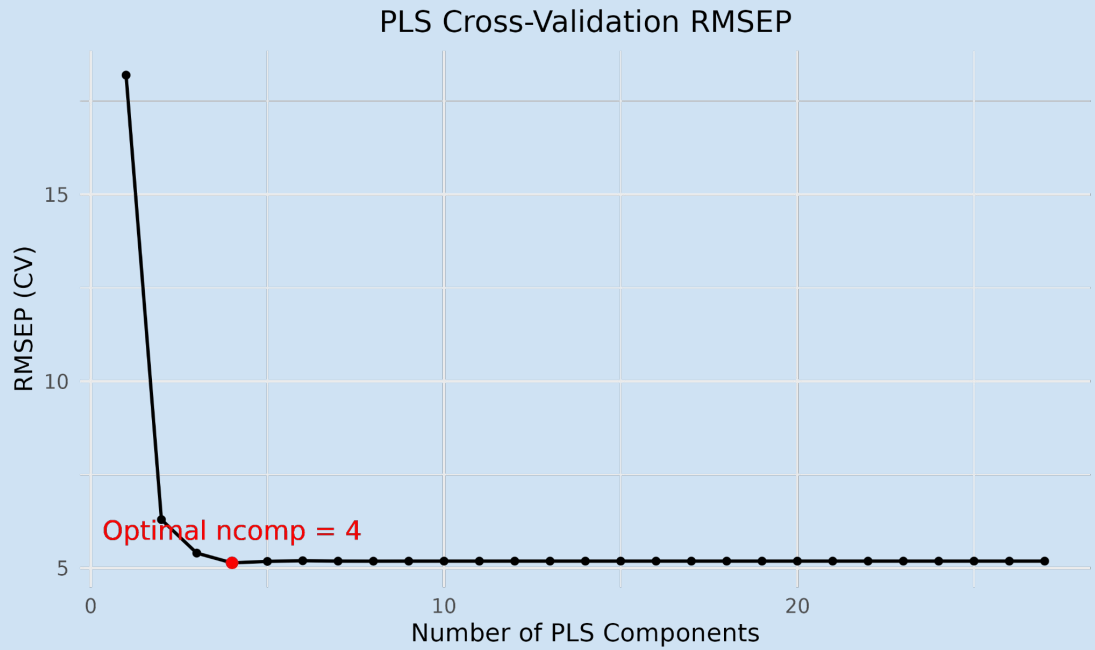
# Low-dimensional example

- **We combine the original predictors** into a latent variable which has **more correlation with Y** than any of the original predictors alone



# High-dimensional example - tuning number of LVs to keep

- CV error is pretty stable after 4 but we prefer smaller models (parsimony)
- We can therefore perform linear regression using the 4 first latent variables



# High-dimensional example - linear regression output

- All 4 LVs are associated with Y, with decreasing strength

```
Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)      3.19182    0.06270  50.909  < 2e-16 ***
LV_scoresComp 1  5.04198    0.01826 276.195  < 2e-16 ***
LV_scoresComp 2  1.27878    0.02840  45.023  < 2e-16 ***
LV_scoresComp 3  0.64658    0.03839  16.844 3.72e-15 ***
LV_scoresComp 4  0.29620    0.03682   8.044 2.13e-08 ***
---
```

# Interpretation of LV1

- Similar with PC regression, we can interpret the loading weights of the important LVs
- LV1 is dominated by the truly important variables X1,...,X10

| variable | loading |
|---|---|
| X8 | 0.2756075 |
| X2 | 0.2738030 |
| X3 | 0.2736750 |
| X1 | 0.2721593 |
| X4 | 0.2690283 |
| X7 | 0.2683456 |
| X10 | 0.2663741 |
| X9 | 0.2660805 |
| X6 | 0.2658825 |
| X5 | 0.2643691 |
| X93 | -0.1397060 |
| X70 | -0.1359411 |

# Interpretation of LV2

- LV2 is correlated with Y, but it is formed of variables which are just noise…

| variable | loading |
|---|---|
| X48 | 0.2727252 |
| X55 | 0.2478832 |
| X91 | -0.2429874 |
| X58 | 0.2019630 |
| X87 | 0.1913649 |
| X46 | 0.1901583 |
| X67 | -0.1892773 |
| X88 | 0.1750529 |

# Maths of PLS regression

**Partial least squares**

- $X \in \mathcal{R}^{n \times p}$ matrix with $n$ rows of observations of $p$ columns of predictors,
- $y$ a continuous outcome,
- both predictors and outcome centered, scaled

# Maths of PLS regression

1. **Find** $LV_1 \in \mathcal{R}^{n \times 1}$ , linear combination of the predictors:

$$LV_1 = Xw_1$$

Where:

- $w_1 \in \mathcal{R}^{p \times 1}$ is the **weight vector** that defines the linear combination of the predictors, chosen to **maximize the covariance** between the latent variable $LV_1$ and $y$:

$$w_1 = \arg \max_{\|w\|=1} \text{Cov}(Xw, y)$$

2. **Adjust** (deflate) $X$ to remove the influence of the first LV:

$$p_1 = \frac{X^\top LV_1}{LV_1^\top LV_1}.$$

$$X_1 = X - LV_1 p_1^T$$

Where $p_1$ is a vector that relates the first latent variable $LV_1$ back to the original predictors.

# Maths of PLS regression

3. **Repeat** to find additional latent variables $LV_2, LV_3, \ldots$ using deflated predictors $X_1, X_2, \ldots$.
4. **Keep** $p^*$ informative latent variables (choose with cross-validation)
5. **Regression step** : regress $y$ on $LV = (LV_1, ..., LV_{p^*})$

# Problems with PLS

- **Every predictor still contributes a non-zero weight to every LV**
  - This is likely to reduce predictive accuracy as we accumulate noise
- **High potential for overfitting**
  - The more predictors available, the more likely we are to find some combination of noise which produces a predictive latent variable

# Latent Variable + Regularisation

## Sparse Partial Least Squares

# Idea of sparse partial least squares

- Latent variable methods like PCr and PLSr deal well with multicollinearity but still suffer in the high-dimensional setting
  - Overfitting, noise accumulation, interpretation…

- Idea : combine partial least squares with lasso regularisation to construct ***sparse latent variables***

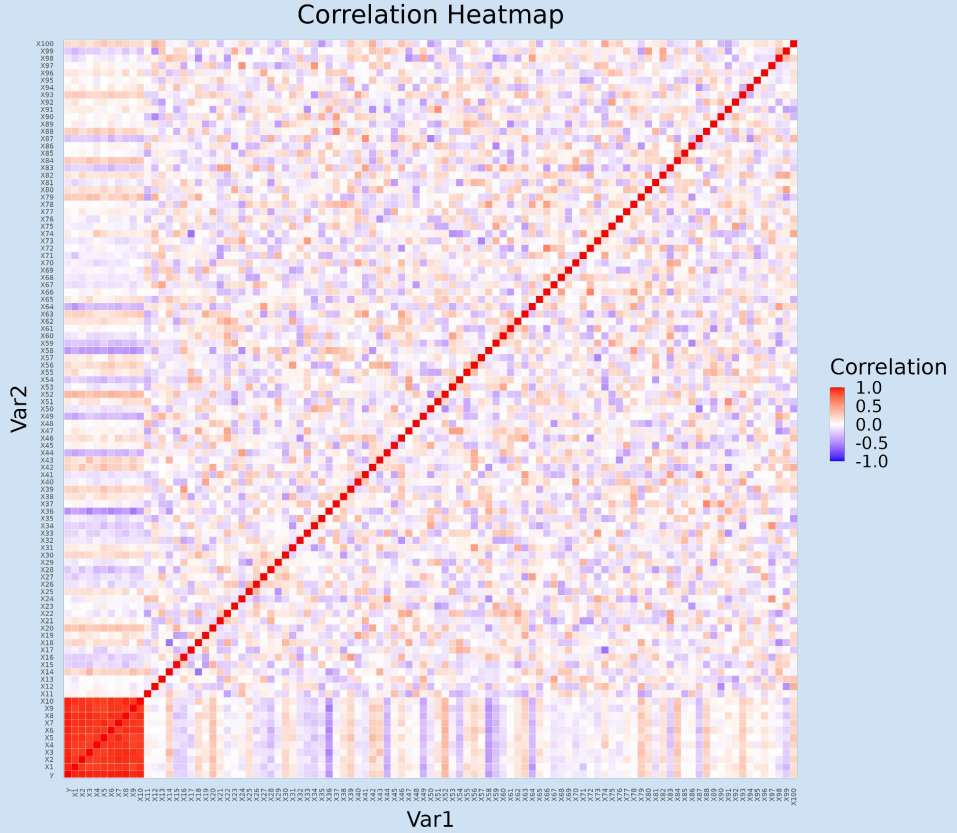
**sPLS = PLS + LASSO**

# Sparse Partial Least Squares Regression

1. **Transform the data** into **sparse** latent variables (LVs) capturing the covariability between the predictors and response, where only a few of the predictors contribute to each LV
2. **Reduce the dimension** by discarding LVs which do not explain much of the co-variability in the data (scree plot)
3. **Perform regression** using reduced number of LVs instead of original predictors

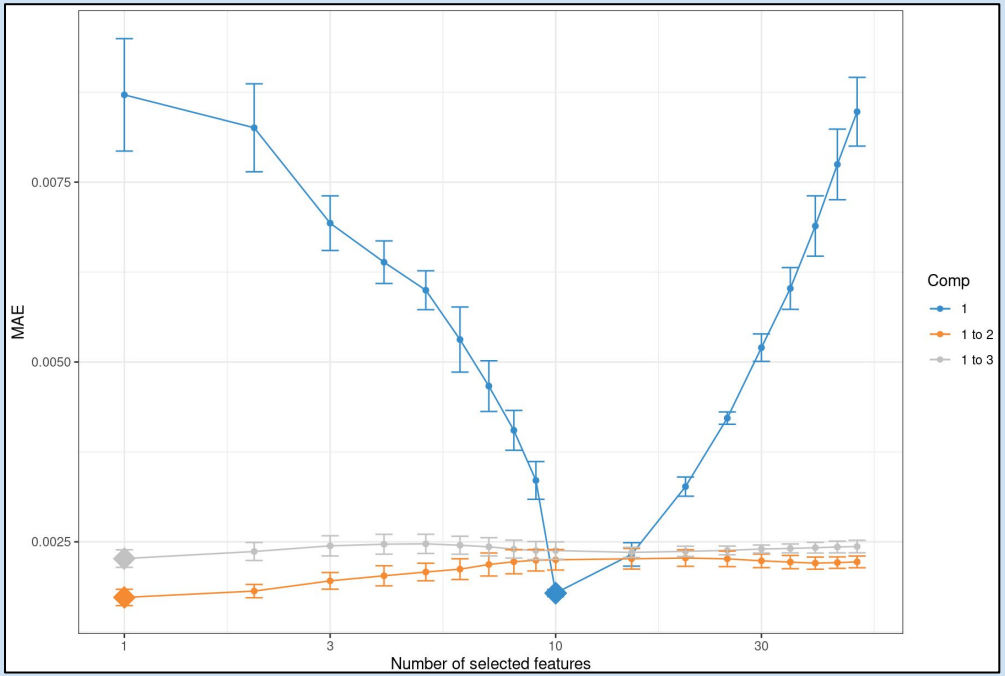
# sPLS in high-dimension

- 100 predictors, 30 observations
- **First 10 predictors** multicollinear and correlated with true coefficients = 2
- **Other 90 are noise**



# High-dimensional example - tuning

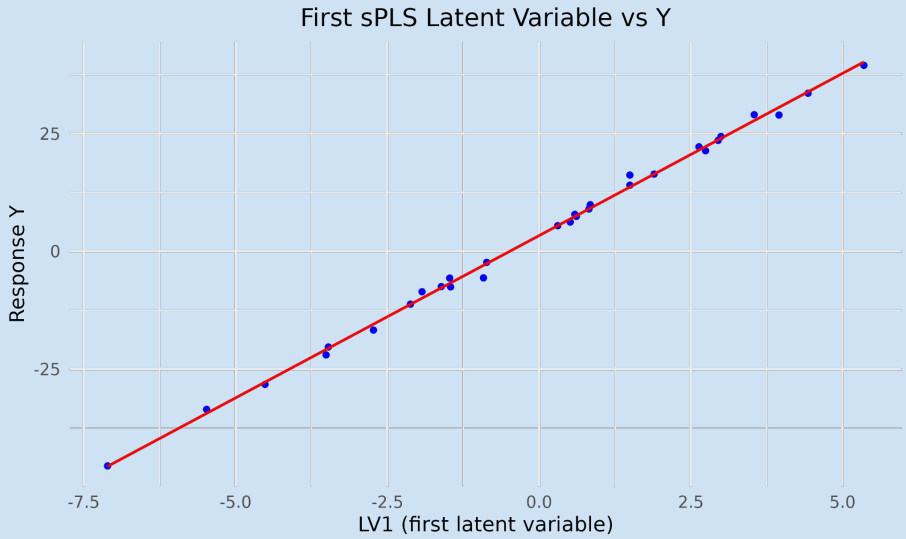
- In MixOmics package - 2 hyperparameters to tune for sPLS
  - Number of components to keep
  - Number of variables to keep on each component (equivalent to tuning sparsity parameter)
- This plot says 1 component with 10 features is optimal



# Look at LV1

The 10 truly important variables were selected for LV1, no others were selected

| | value.var |
|---|---|
| X7 | 0.3244752 |
| X4 | 0.3193589 |
| X5 | 0.3190780 |
| X1 | 0.3173010 |
| X8 | 0.3160257 |
| X10 | 0.3153129 |
| X6 | 0.3143405 |
| X2 | 0.3137792 |
| X9 | 0.3130805 |
| X3 | 0.3092797 |



# Maths of sPLS

**Sparse partial least squares (sPLS)**

- $X \in \mathcal{R}^{n \times p}$ matrix with $n$ rows of observations of $p$ columns of predictors,
- $y$ a continuous outcome,
- both predictors and outcome centered, scaled

1. **Find** $LV_1 \in \mathcal{R}^{n \times 1}$ , linear combination of the predictors:

$$LV_1 = Xw_1$$

Where:

- $w_1 \in \mathcal{R}^{p \times 1}$ is the **sparse weight vector** that defines the linear combination of the predictors, chosen to **maximize the covariance** between the latent variable $LV_1$ and $y$ while enforcing sparsity via an L1 penalty:

$$w_1 = \arg \max_{\|w\|_2 = 1} \Big[ \mathrm{Cov}(Xw, y) - \lambda \|w\|_1 \Big],$$

where $\|w\|_1 = \sum_{j=1}^{p} |w_j|$ is the L1 norm of $w$ and $\lambda > 0$ controls the degree of sparsity (larger $\lambda$ $\rightarrow$ more zeros in $w_1$).

# Take-Home messages

| | Regularisation? | Sparsity/variable selection? | Handles Multicollinearity? | Predictive power? |
|---|---|---|---|---|
| **Linear regression** | no | no | Fails in multicollinear settings | Bad |
| **Ridge regression** | Yes, L2 penalty | No | Yes | Good |
| **Lasso regression** | Yes, L1 penalty | Yes | Tendency to discard predictors in multicollinear groups | Slightly worse than ridge |
| **Elastic net regression** | Yes, L1 + L2 penalty | Yes | Yes, selects groups of multicollinear predictors | Good |
| **Principal component regression** | No | No | Yes | PCs are not necessarily predictive |
| **Partial least squares** | No | No | Yes | Good but can accumulate noise in predictors |
| **Sparse partial least squares** | Yes, L1 penalty | Yes | Yes | Good |

# Conclusion

- **Curse of high-dimensionality makes predictive modelling difficult**, specialist methods are required
  - Main problems : large amount of irrelevant predictors, multicollinearity, overfitting
- **Regularisation** approaches
  - **Mitigate overfitting** by reducing the impact of the training data on the parameter estimates
  - Allow for **variable selection**
- **Latent variable** approaches allow to
  - Find **parsimonious “hidden” signals** in the data
- LV + regularisation approaches are powerful (sPLS)

# Practical work

We are going to analyse a public dataset containing transcriptomic samples from participants vaccinated with seasonal influenza vaccine. The goal will be to predict the vaccine-induced immune response (antibodies) using the earlier gene expression response.

Complete the R markdown file

PHDS_omics_supervisedlearning_2026_questions.Rmd