Statistical Methods for High-dimensional Biology

Supervised Learning II: advanced topics

Yongjin Park, UBC Path&Lab, STAT, BC Cancer

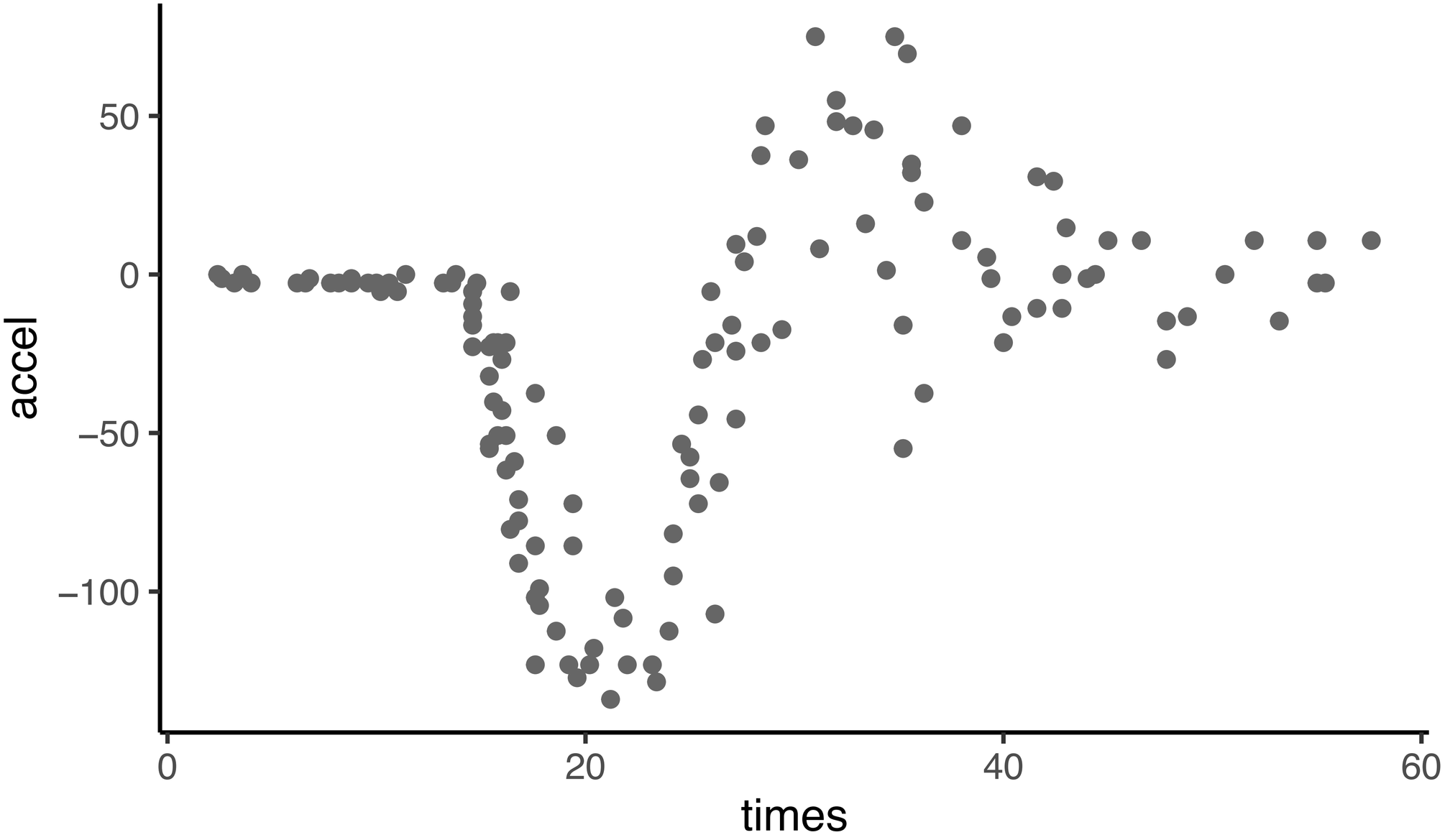
Today's lecture: Supervised Learning in Genomics

- **Non-parametric prediction methods**
  - When we don't have any idea of data-generation mechanisms.
  - Kernel (local) regression
  - Gaussian Process
- **Ensemble learning: the collective power of weak learners**
  - Expectation maximization
  - Mixture of linear regressions
- **Variable selection**
  - Challenges in high-dimensional prediction problems
  - Sparse regression models

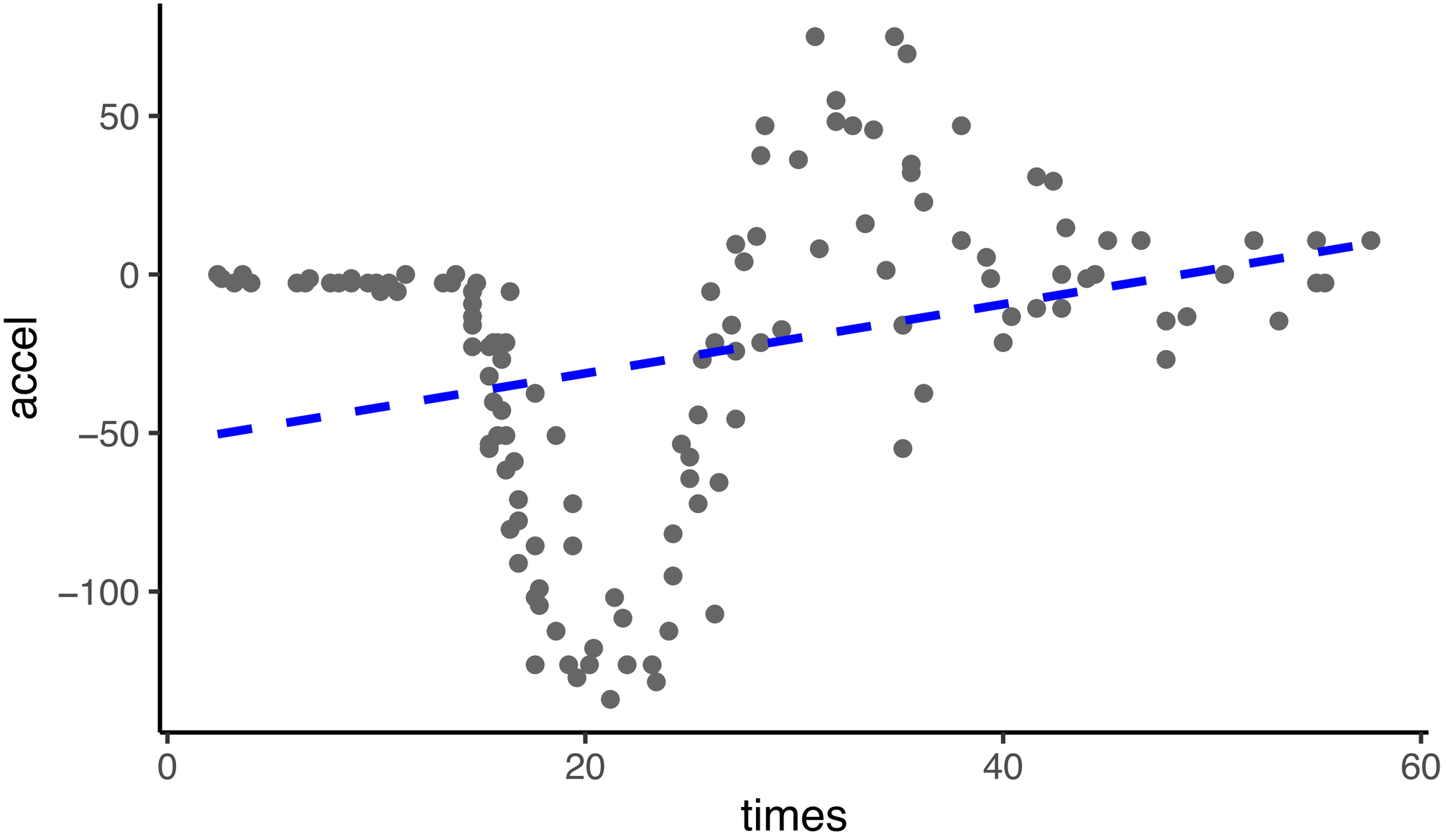
Why do learn non-parametric methods?

- We are already using it in many statistics applications!
  - E.g., mean-variance relationship modelling by `lowess` in `voom`
  - E.g., `geom_smooth` in the `ggplot` ecosystem
- Interpolation, extrapolation, and smoothing trends

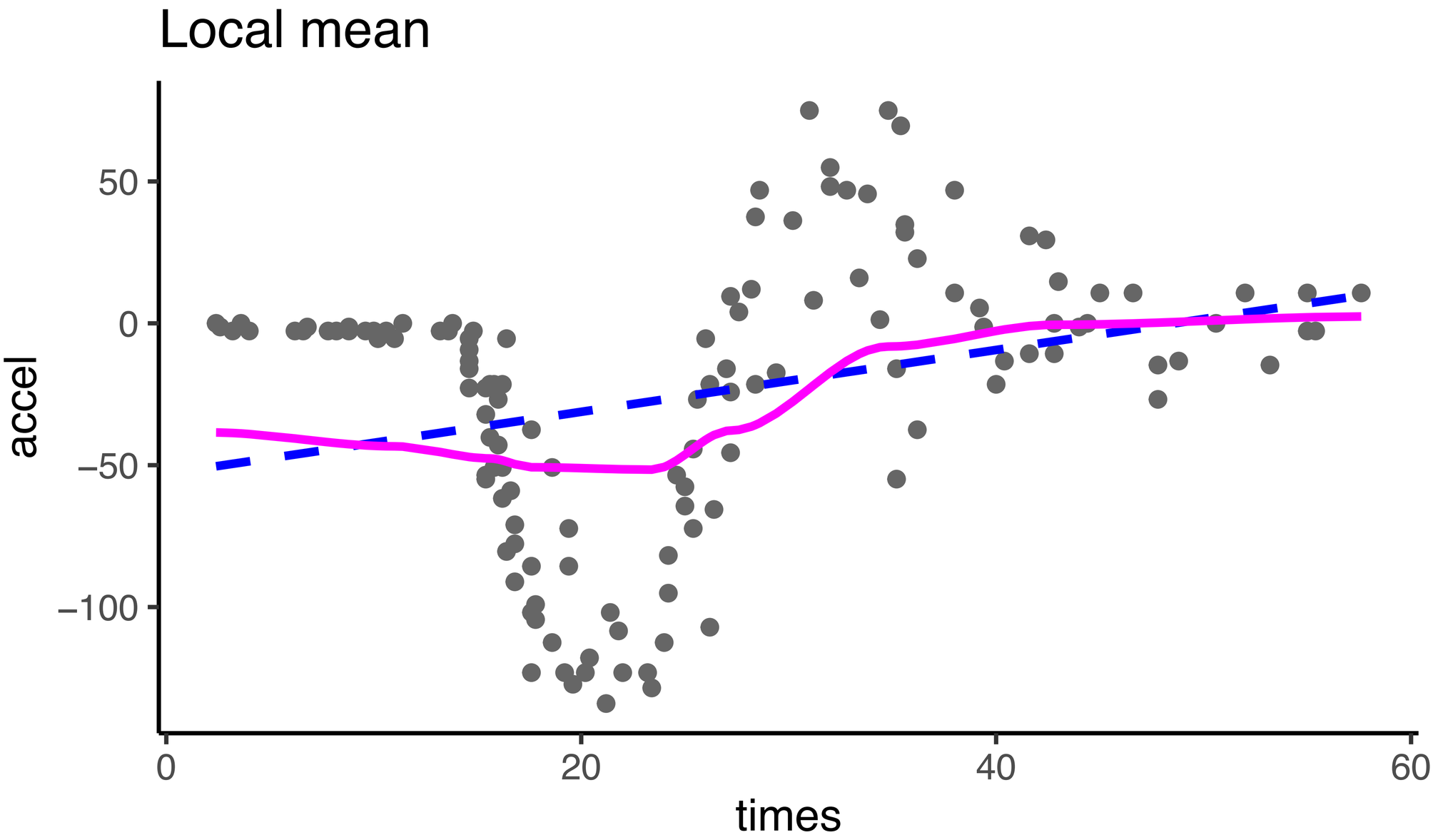
There are many things that a linear model cannot capture



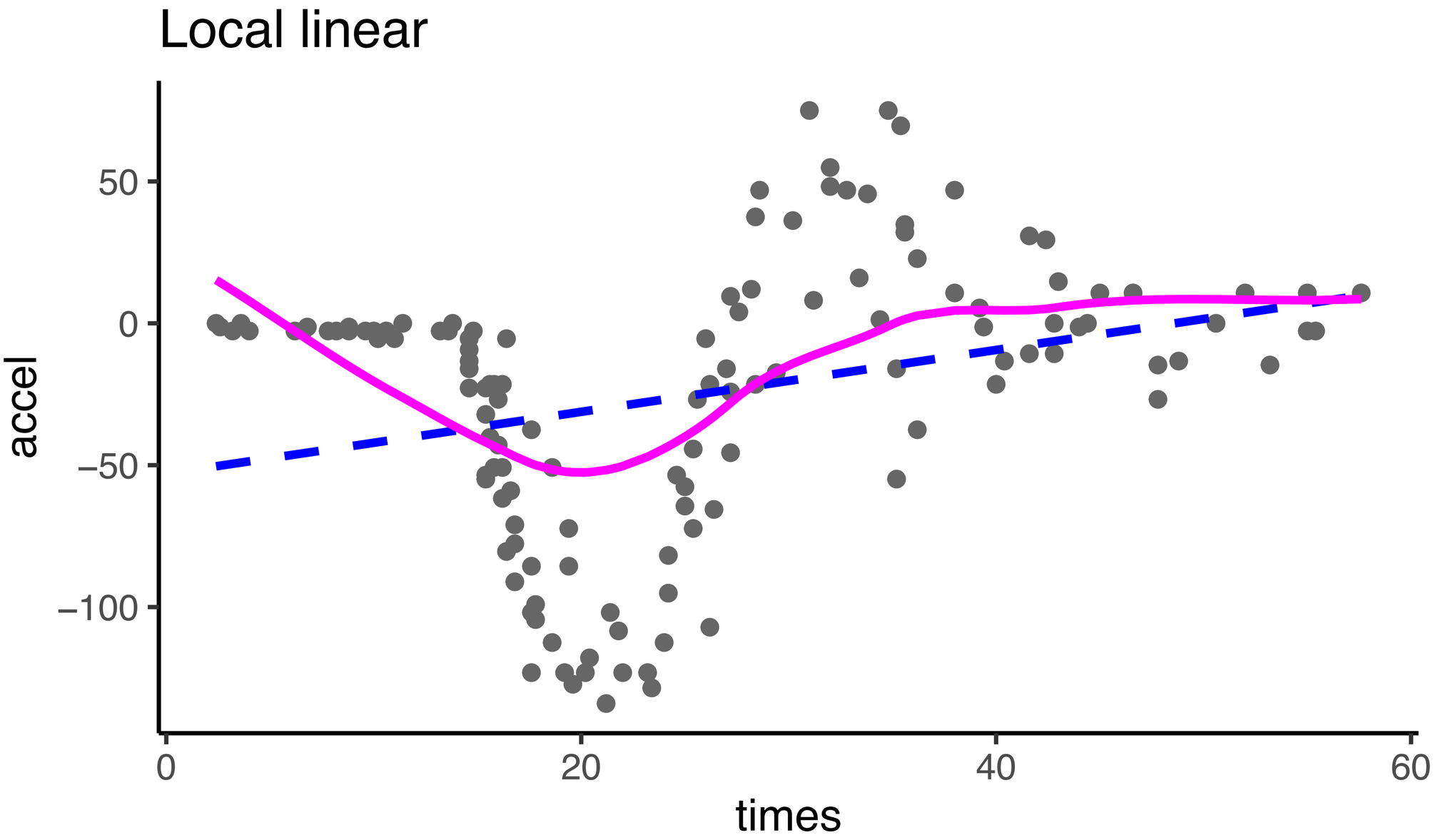
There are many things that a linear model cannot capture



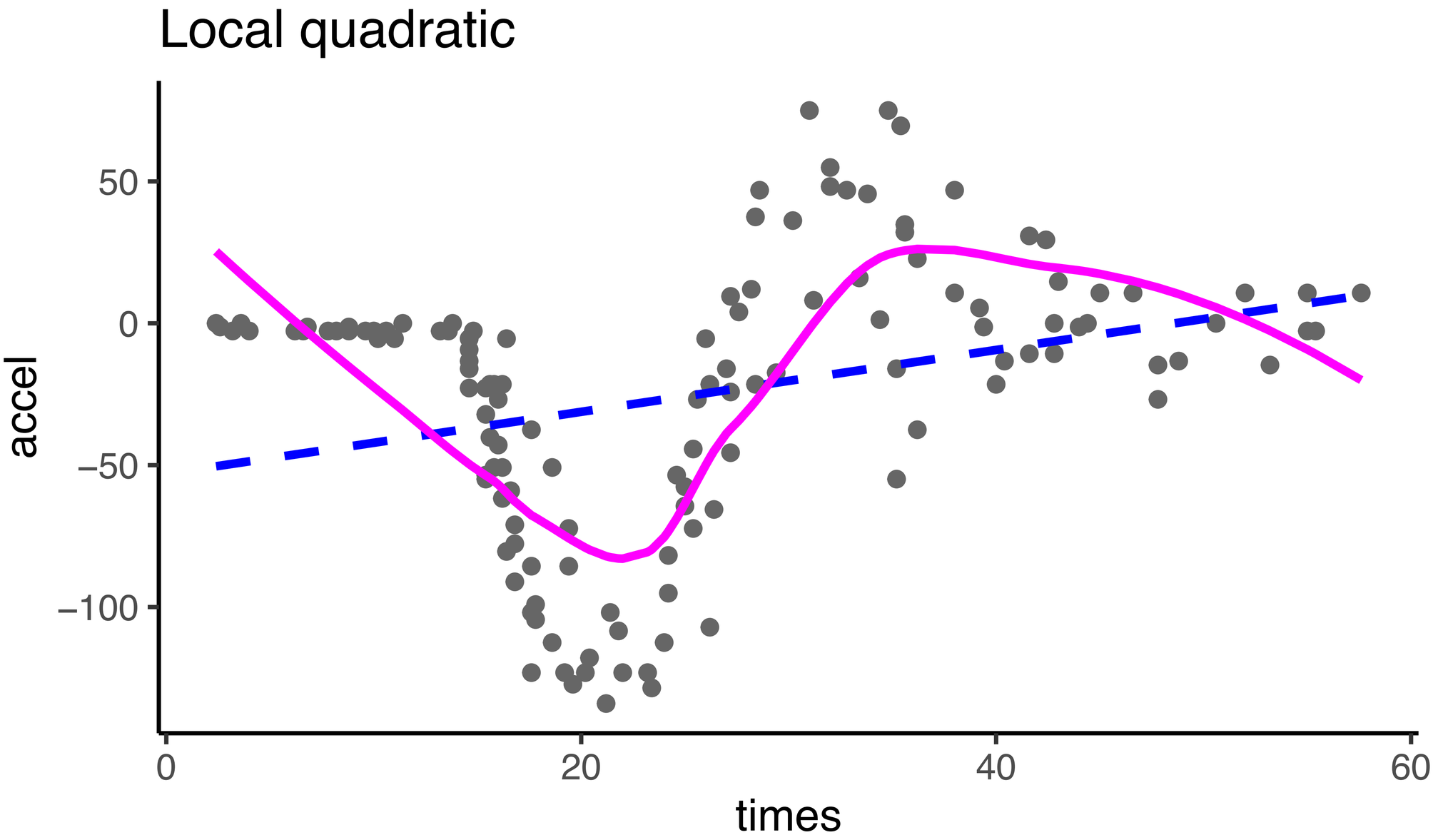
Let's think locally ... local polynomial regression `loess`



Let's think locally ... local polynomial regression `loess`



Let's think locally ... local polynomial regression `loess`

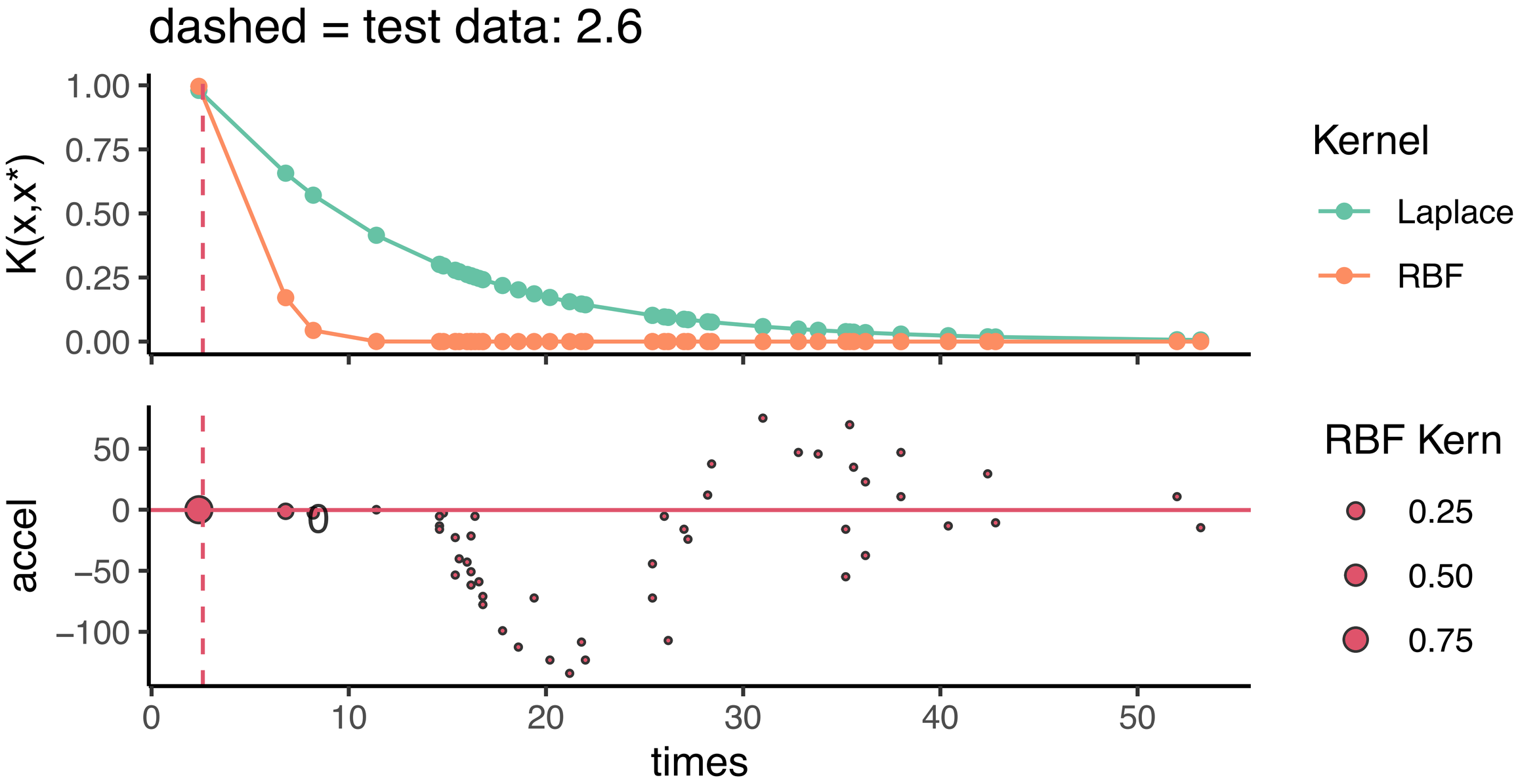


Nadaraya-Watson (kernelized local) estimator

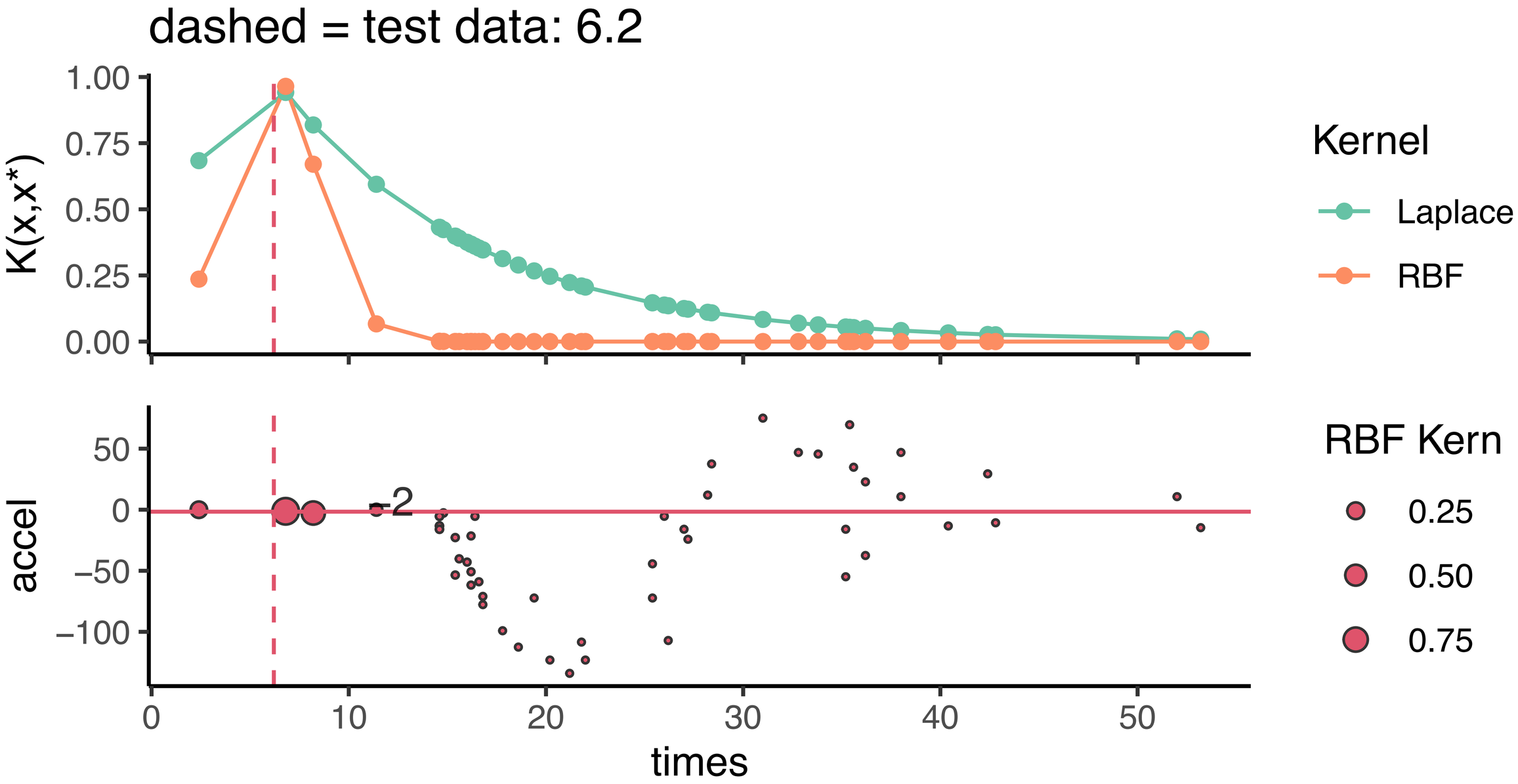
```
nn <- nrow(mcycle)
train.idx <- sample(nn, 50)
x.test <-
    mcycle[-train.idx, 1, drop = FALSE] %>%
    as.matrix
y.test <-
    mcycle[-train.idx, 2, drop = FALSE] %>%
    as.matrix
x.train <-
    mcycle[train.idx, 1, drop = FALSE] %>%
    as.matrix
y.train <-
    mcycle[train.idx, 2, drop = FALSE] %>%
    as.matrix
```

```
#' @param x.test testing data points
#' @param X training X
#' @param Y training Y
#' @param h bandwidth h
nw.rbf <- function(x.test, X, Y, h) {
    rbf <- kernlab::rbfdot(sigma=h)
    K <- kernlab::kernelMatrix(rbf, x.test, X)
    W <- K / rowSums(K)
    W %*% as.matrix(Y)
}
```

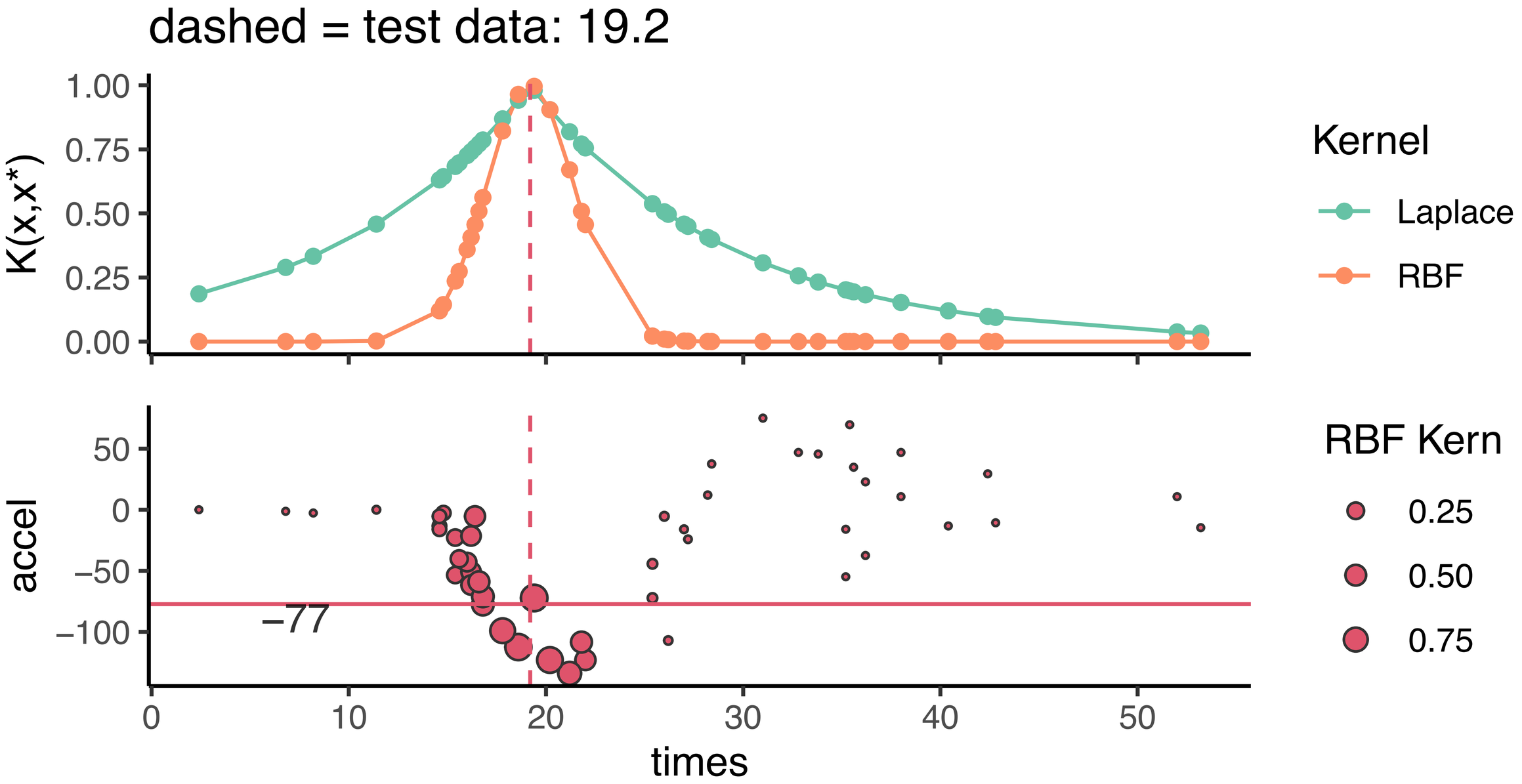
Basic idea: borrow neighbours' Y values in training data



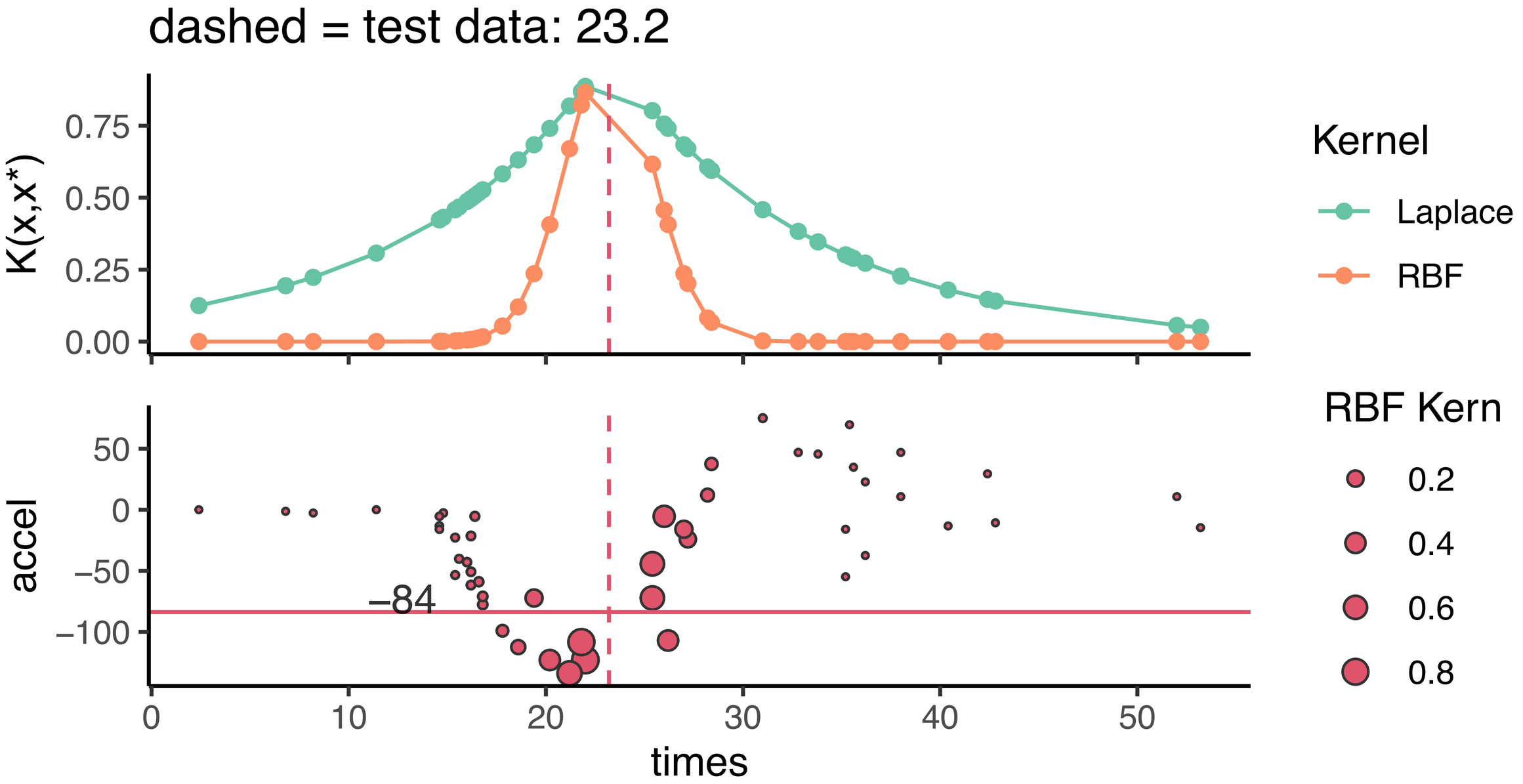
Basic idea: borrow neighbours' Y values in training data



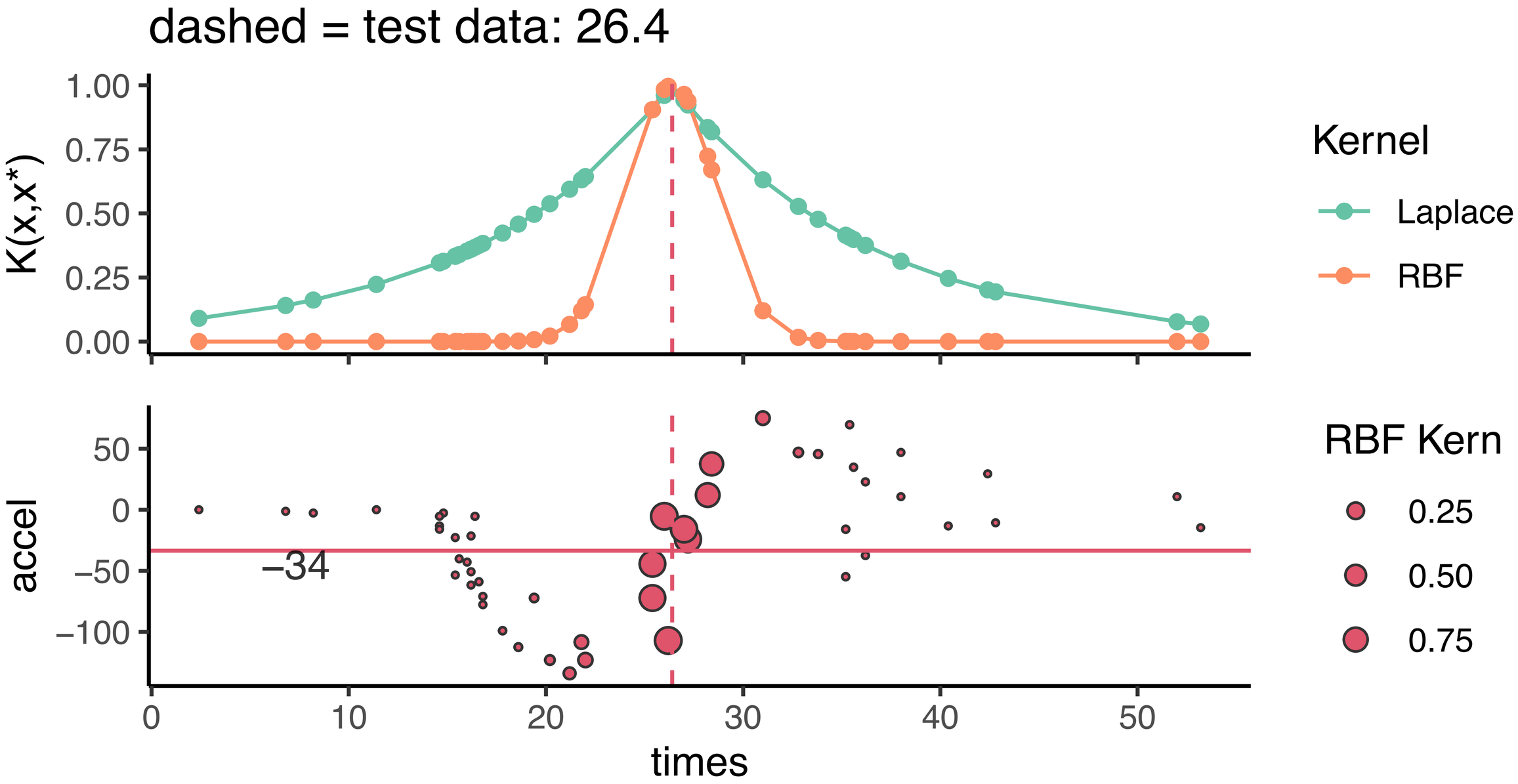
Basic idea: borrow neighbours' Y values in training data



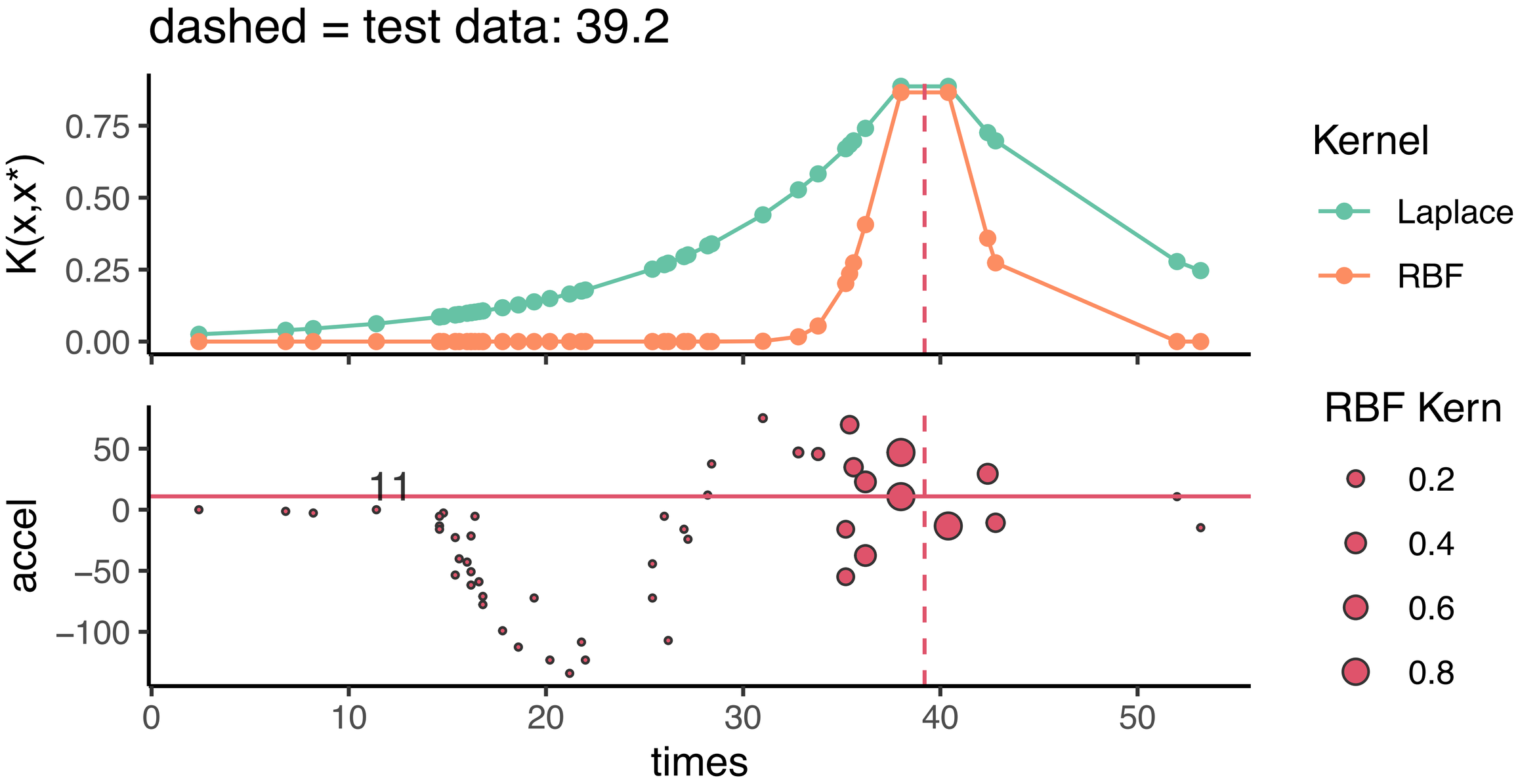
Basic idea: borrow neighbours' Y values in training data



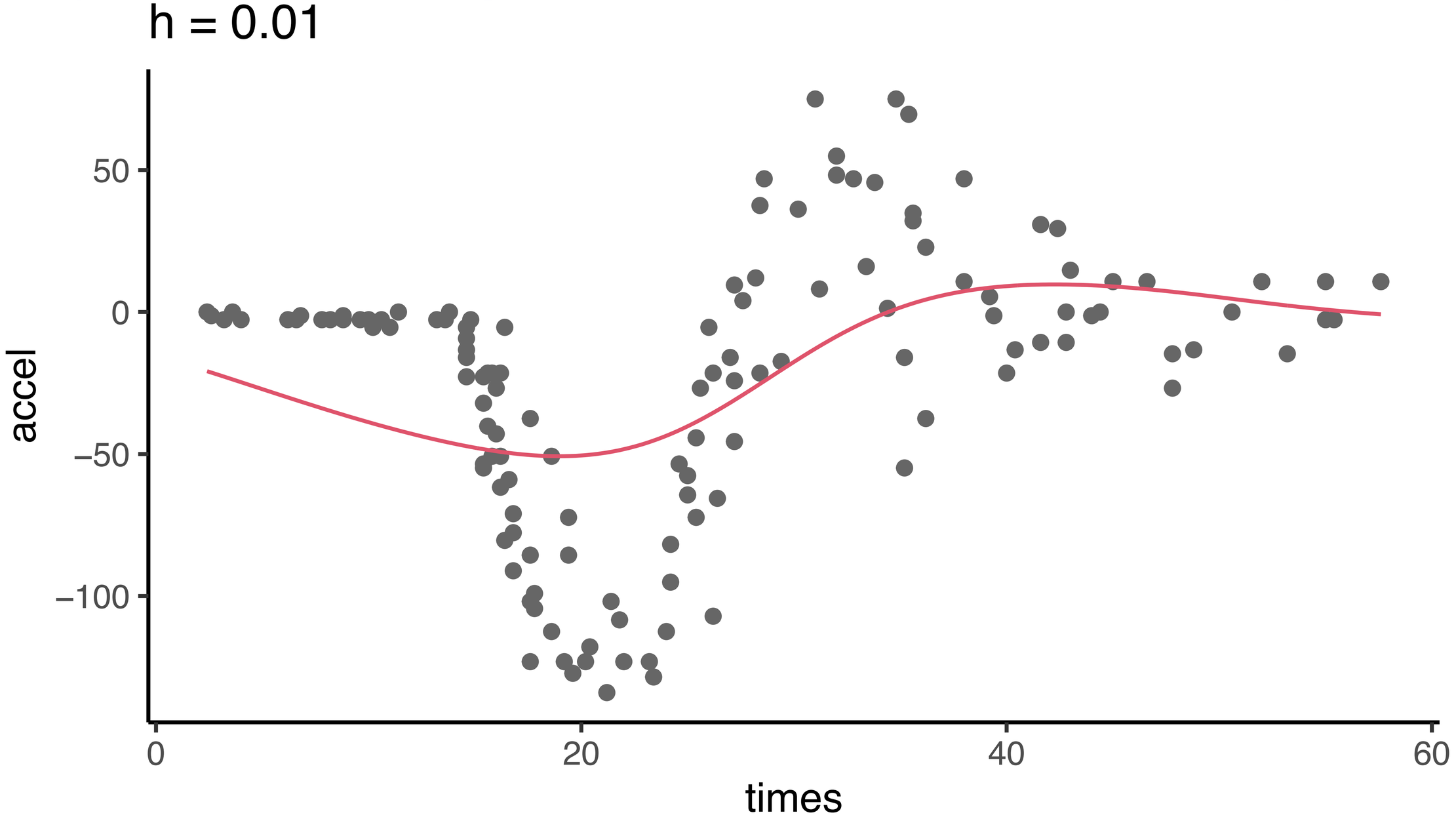
Basic idea: borrow neighbours' Y values in training data



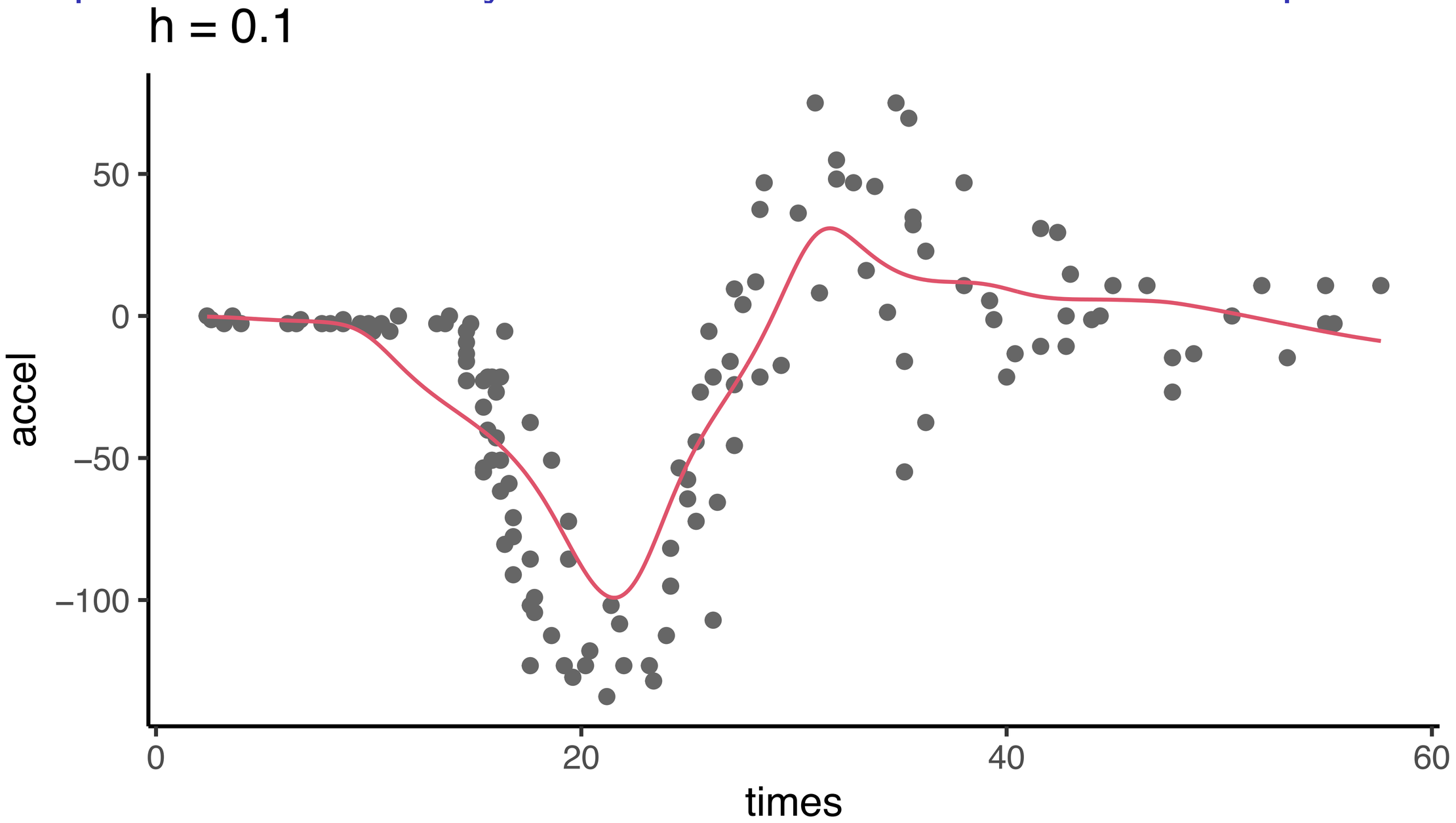
Basic idea: borrow neighbours' Y values in training data



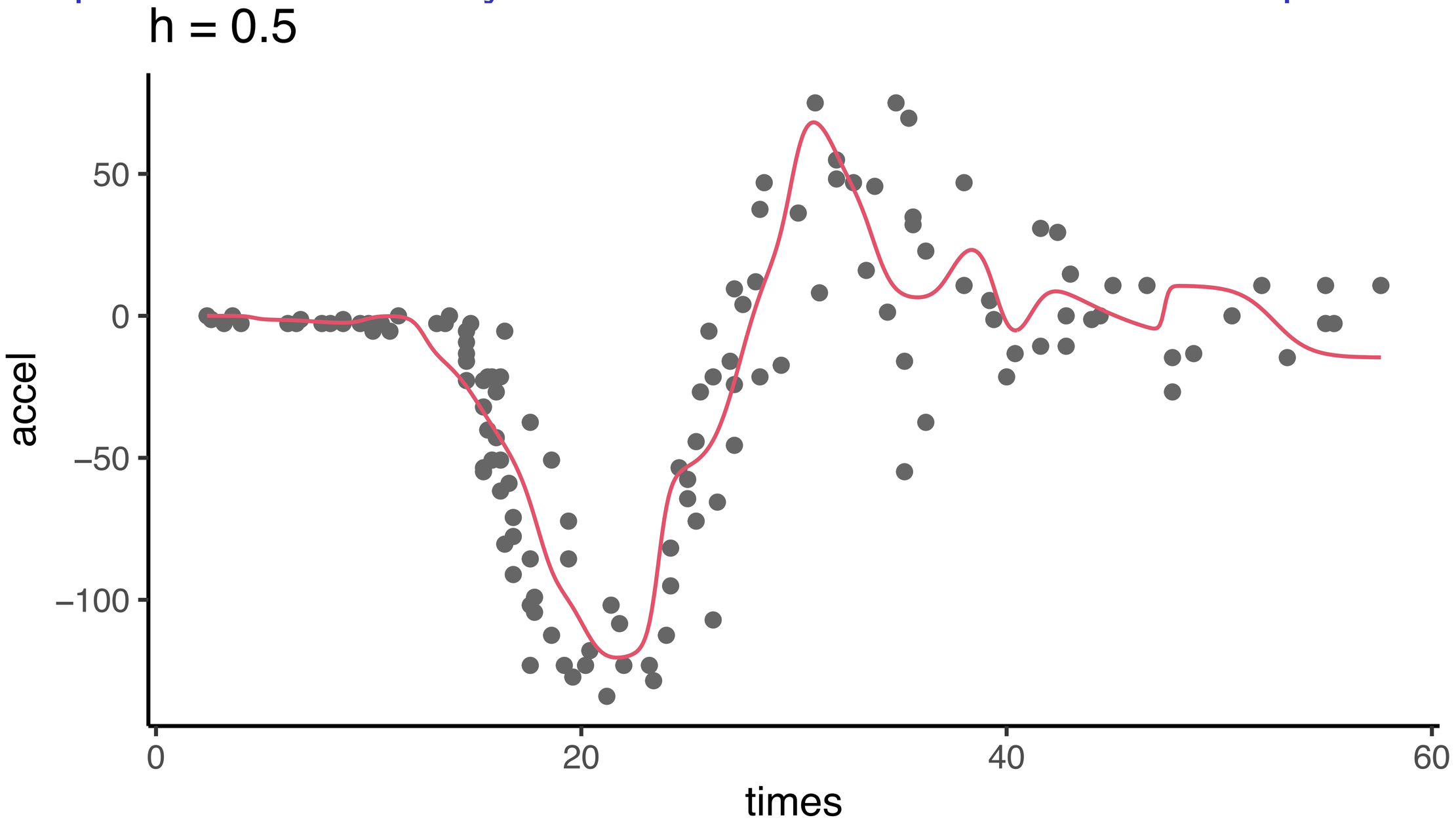
A full spectrum of Nadaraya-Watson with different bandwidth parameters



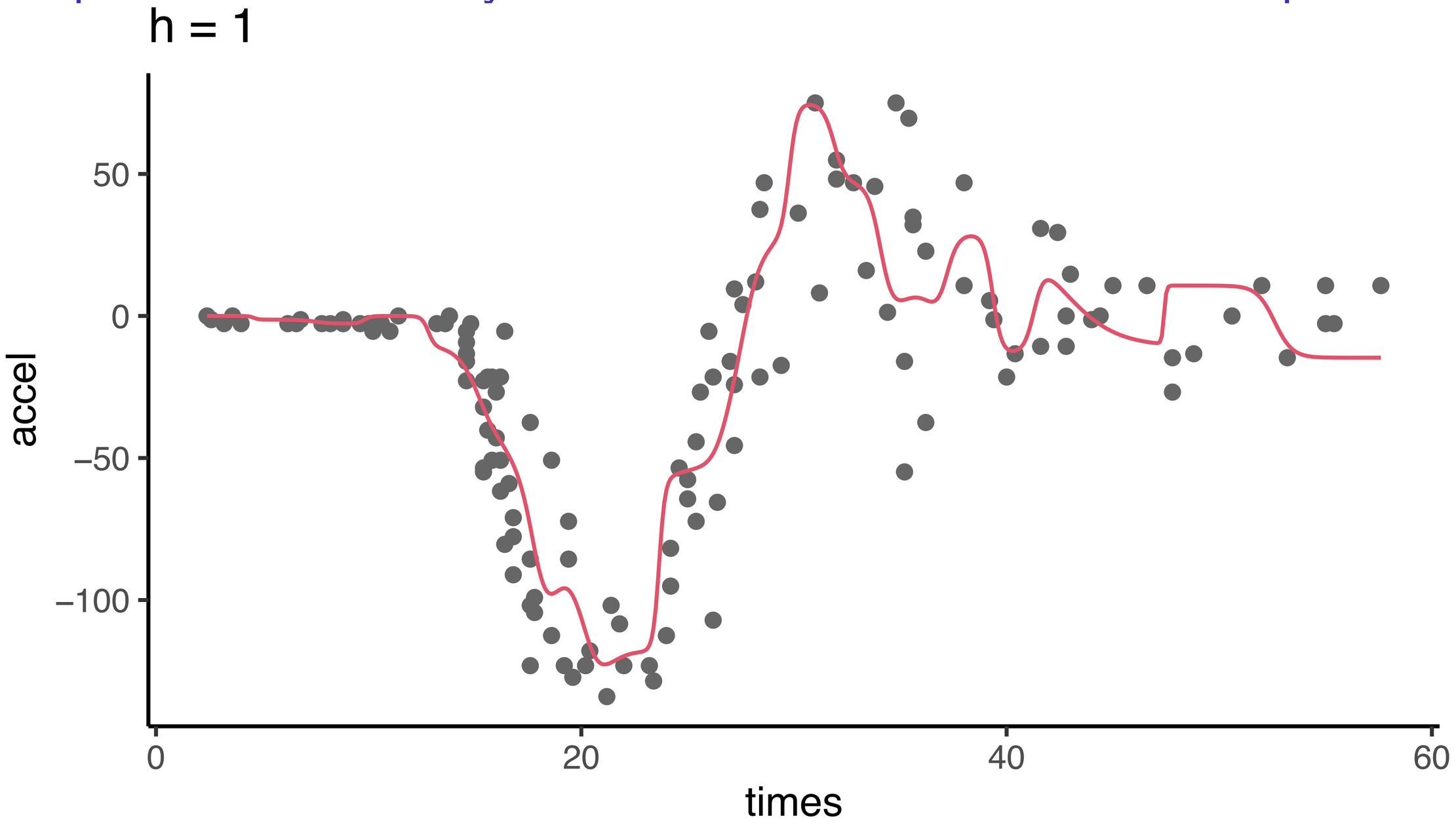
A full spectrum of Nadaraya-Watson with different bandwidth parameters



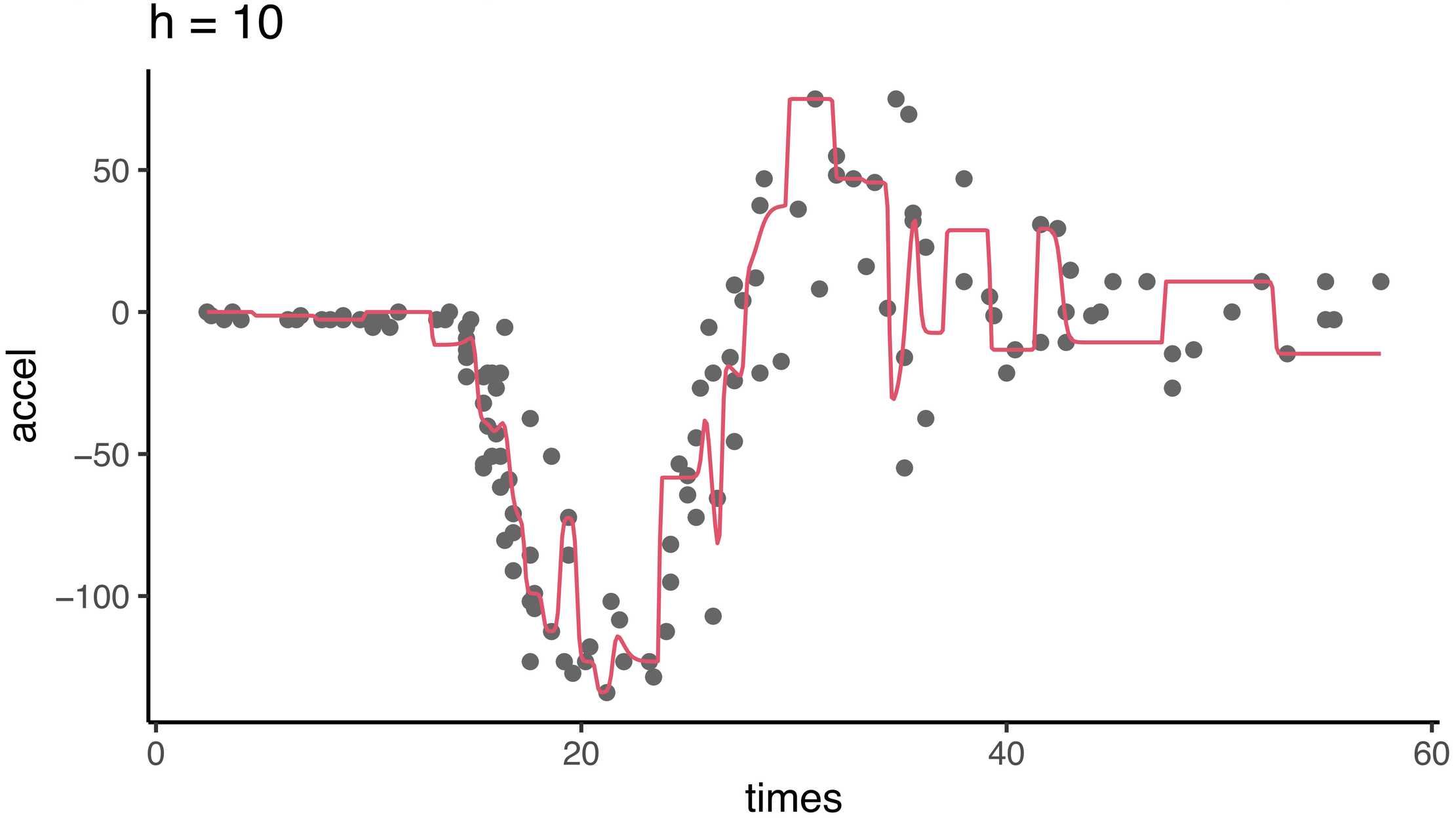
A full spectrum of Nadaraya-Watson with different bandwidth parameters



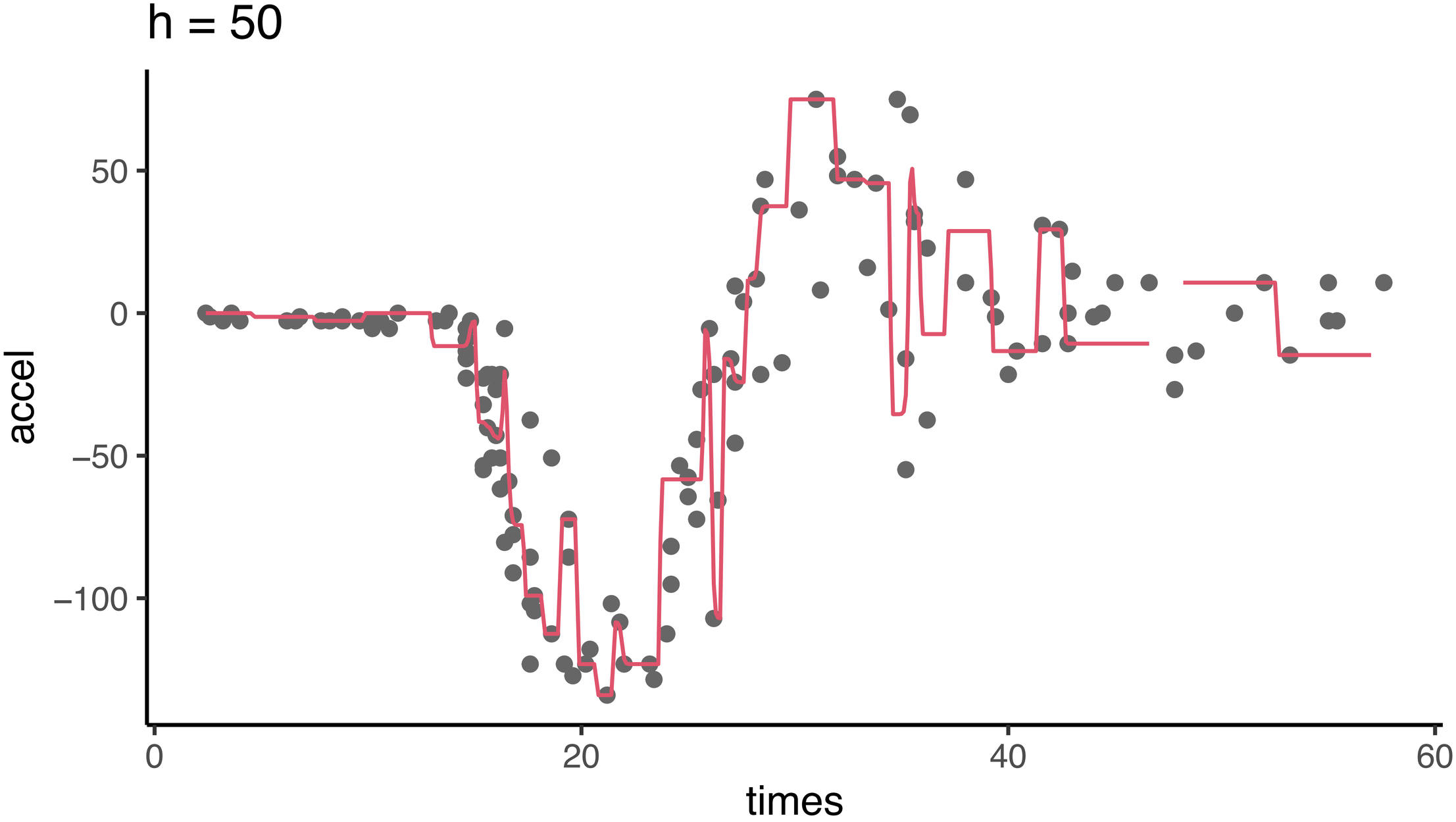
A full spectrum of Nadaraya-Watson with different bandwidth parameters



A full spectrum of Nadaraya-Watson with different bandwidth parameters



A full spectrum of Nadaraya-Watson with different bandwidth parameters



Bias-variance tradeoff in generalization error

$$\mathbb{E}\Big[\overset{\text{true unknown model}}{f(X)} - \overset{\text{our attempt}}{\hat{f}(X)}\Big]^2 = \underbrace{\mathbb{E}\Big[f(X) - \mathbb{E}\Big[\hat{f}\Big]\Big]^2}_{\text{bias}^2} + \underbrace{\mathbb{E}\Big[\mathbb{E}\Big[\hat{f}\Big] - \hat{f}(X))\Big]^2}_{\text{variance}}$$

Wasserman, *All of Statistics* (2005)

Today's lecture: Supervised Learning in Genomics

- **Non-parametric prediction methods**
  - When we don't have any idea of data-generation mechanisms.
  - Kernel (local) regression
  - Gaussian Process
- **Ensemble learning: the collective power of weak learners**
  - Expectation maximization
  - Mixture of linear regressions
- **Variable selection**
  - Challenges in high-dimensional prediction problems
  - Sparse regression models

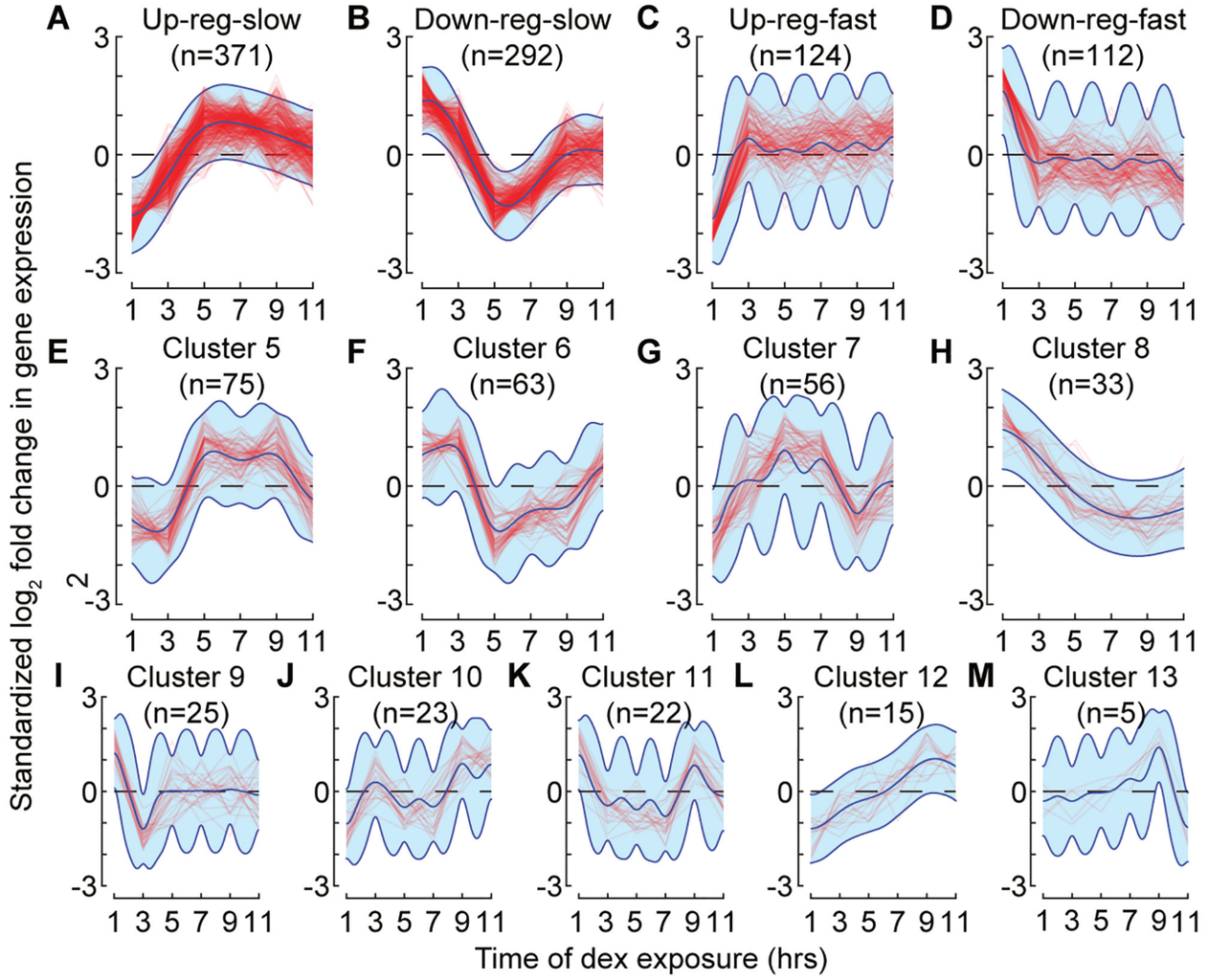
Can we learn an arbitrary shape of gene expression trajectories from data?

RESEARCH ARTICLE

Clustering gene expression time series data using an infinite Gaussian process mixture model

Ian C. McDowell[1,2], Dinesh Manandhar[1,2], Christopher M. Vockley[2,3], Amy K. Schmid[2,4], Timothy E. Reddy[1,2,3]*, Barbara E. Engelhardt[5,6]*

1 Computational Biology & Bioinformatics Graduate Program, Duke University, Durham, North Carolina, United States of America, **2** Center for Genomic & Computational Biology, Duke University, Durham, North Carolina, United States of America, **3** Department of Biostatistics & Bioinformatics, Duke University Medical Center, Durham, North Carolina, United States of America, **4** Biology Department, Duke University, Durham, North Carolina, United States of America, **5** Department of Computer Science, Princeton University, Princeton, New Jersey, United States of America, **6** Center for Statistics and Machine Learning, Princeton University, Princeton, New Jersey, United States of America



What is Gaussian process?

- A non-parametric Bayesian approach to represent distribution over functions.
- A high-level generative model:
  1. Sample some function f from GP
  2. Evaluate $f_i \leftarrow f(\mathbf{x}_i)$
- A function evaluated at each data point i, f_i, follows multivariate Gaussian

$$\mathbf{f} \sim \mathcal{N}\left(\underbrace{\mathbf{m}(\{\mathbf{x}_i\})}_{n\times 1\text{mean function}}, \underbrace{K(\{\mathbf{x}_i\},\{\mathbf{x}_i\})}_{n\times n\text{covariance}}\right)$$

- We will only need to handle the sampled $\mathbf{f}$ vector, not an abstract $f \sim \mathcal{GP}$.

Why is GP useful in practice?

- When we don't have a clear idea of a type of function f
- High-dimensional design matrix X where $d \gg n$
- Can avoid the curse of dimensionality!
- The goal is prediction!

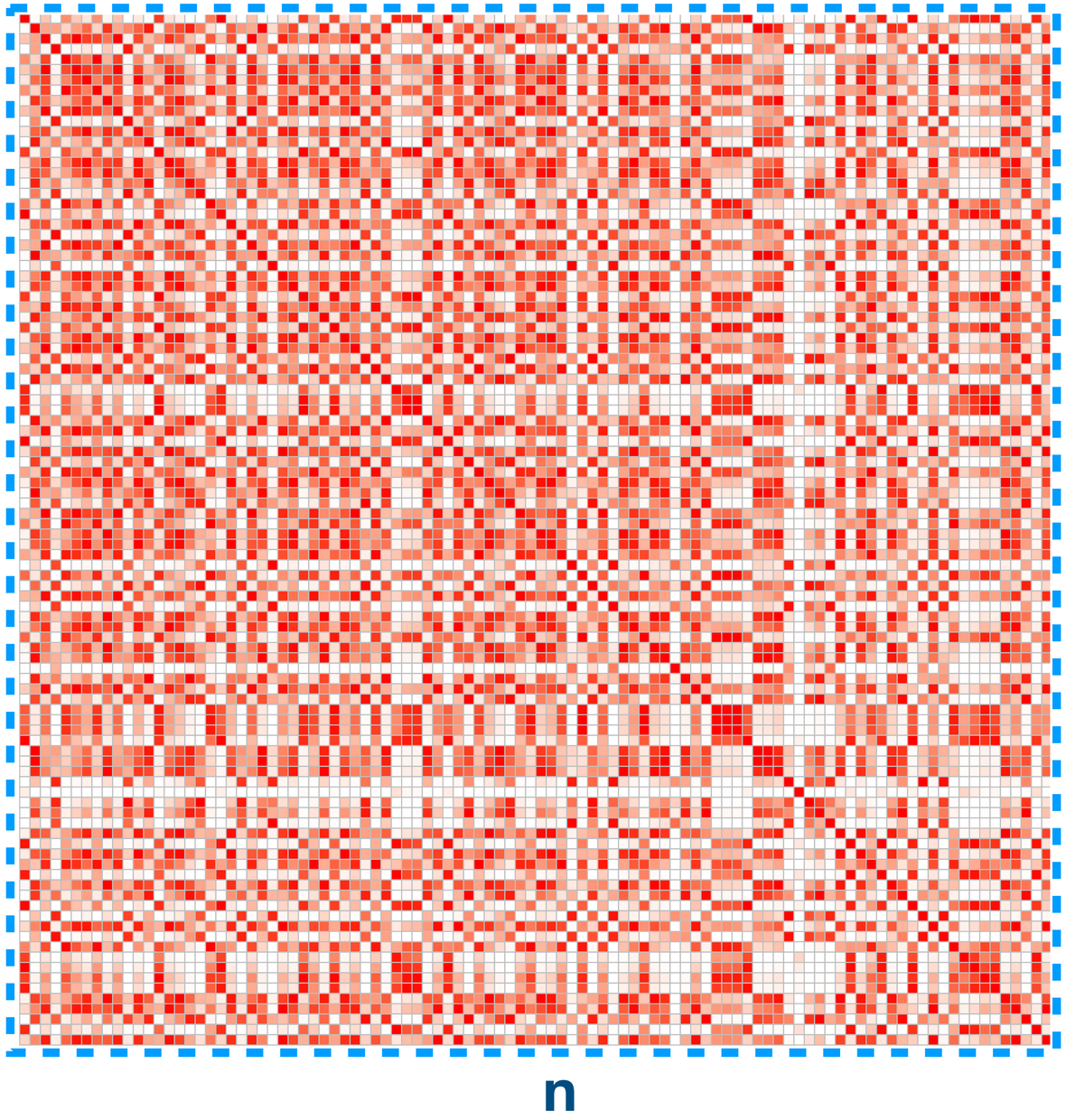
$$\int df \, p(\underbrace{\mathbf{y}^\star}_{\text{new data}} \mid \underbrace{X^\star}_{\text{new data}}, \underbrace{f}_{\text{unknown function}}) p(\underbrace{f}_{\text{unknown function}} \mid \underbrace{X, \mathbf{y}}_{\text{training data}}) \mathcal{GP}(f)$$

$$\rightarrow p(\mathbf{y}^\star | X^\star, \mathbf{y}, X)$$

Covariance (Kernel) matrix when data points are scattered

```
n <- 100
d <- 1
x <- .rnorm(n, d)

rbf <- kernlab::rbfdot(sigma=1)
K <- kernlab::kernelMatrix(rbf, x)
```

Will the dims change with d > 1?



We will use *stan* to infer/simulate the "posterior" distribution of unknown functions

https://mc-stan.org/

Writing down Gaussian Process in stan

```
data {
  int<lower=1> N;                        // N data points
  int<lower=1> D;                        // Dimensionality
  array[N] vector[D] X;                  // N x D data
}
transformed data {
  matrix[N, N] K;                        // (realized) Kernel matrix
  vector[N] mu = rep_vector(0, N);       // mean vector
  K = gp_exp_quad_cov(X, 1.0, 1.0);      // RBF kernel
  for (i in 1:N)                         // regularizer
    K[i, i] = K[i, i] + 0.1;             //
  matrix[N, N] L;                        // Cholesky
  L = cholesky_decompose(K);             // for faster N(mu,K)
}
parameters {
  vector[N] y;
}
model {
  y ~ multi_normal_cholesky(mu, L);
}
```

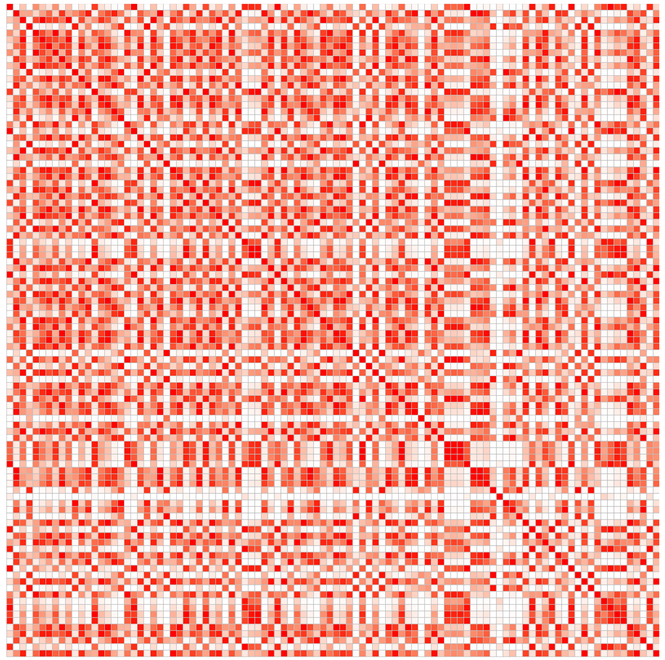
See Stan's GP tutorial

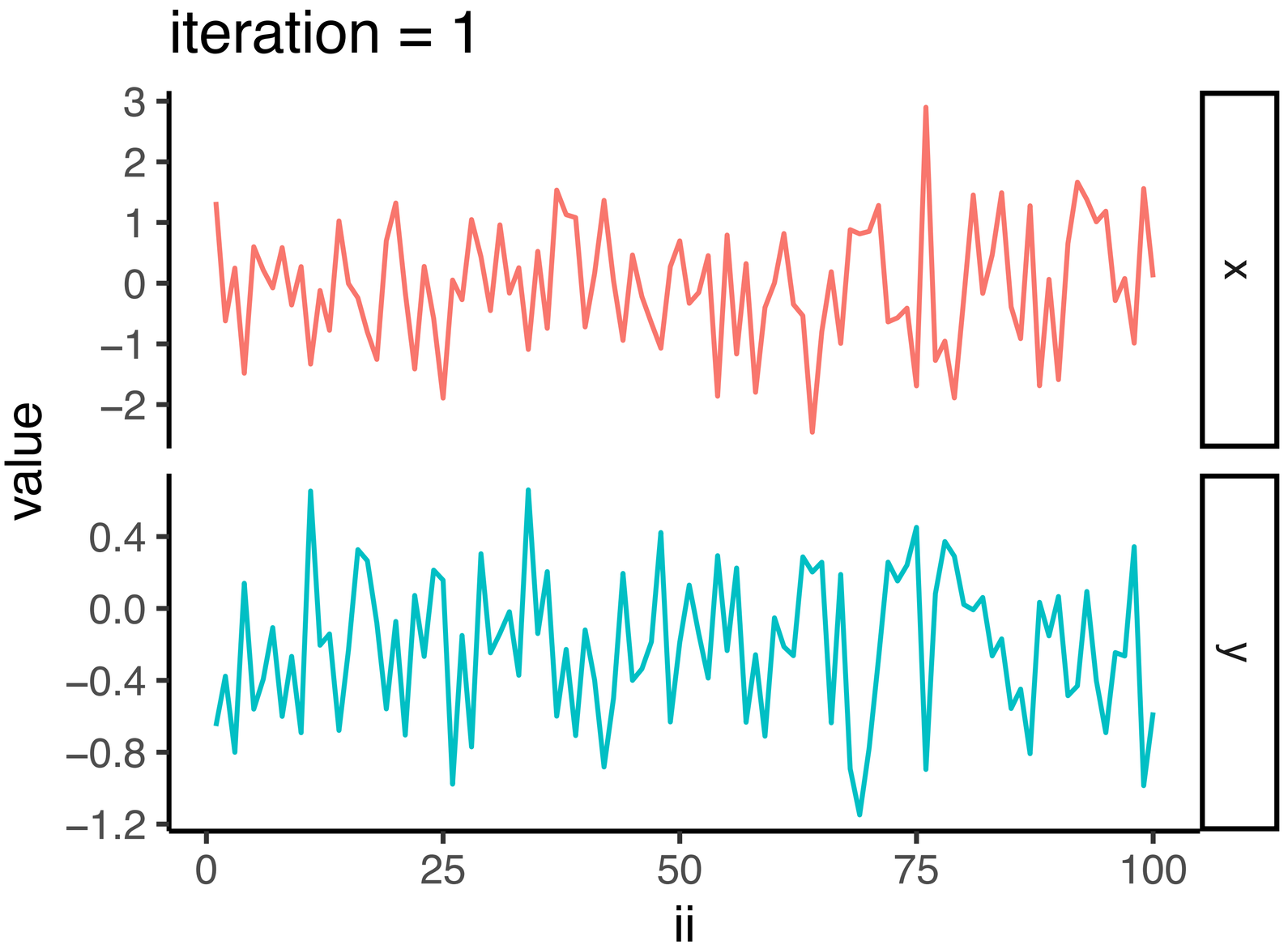
Or you can have a separate file for the stan code and import it to R

```
gp <- cmdstan_model("example_gp1.stan")
```

Let's generate "functions" with GP prior

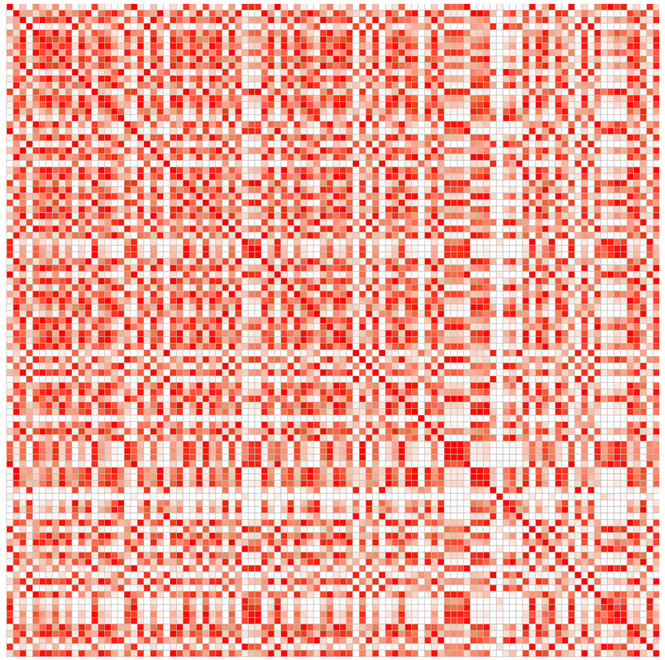
```
.data <- list(N=n, D=d, X=x)

gp.sample <-
    gp$sample(data = .data,
              chains = 1,
              iter_warmup=10,
              iter_sampling=111)
```

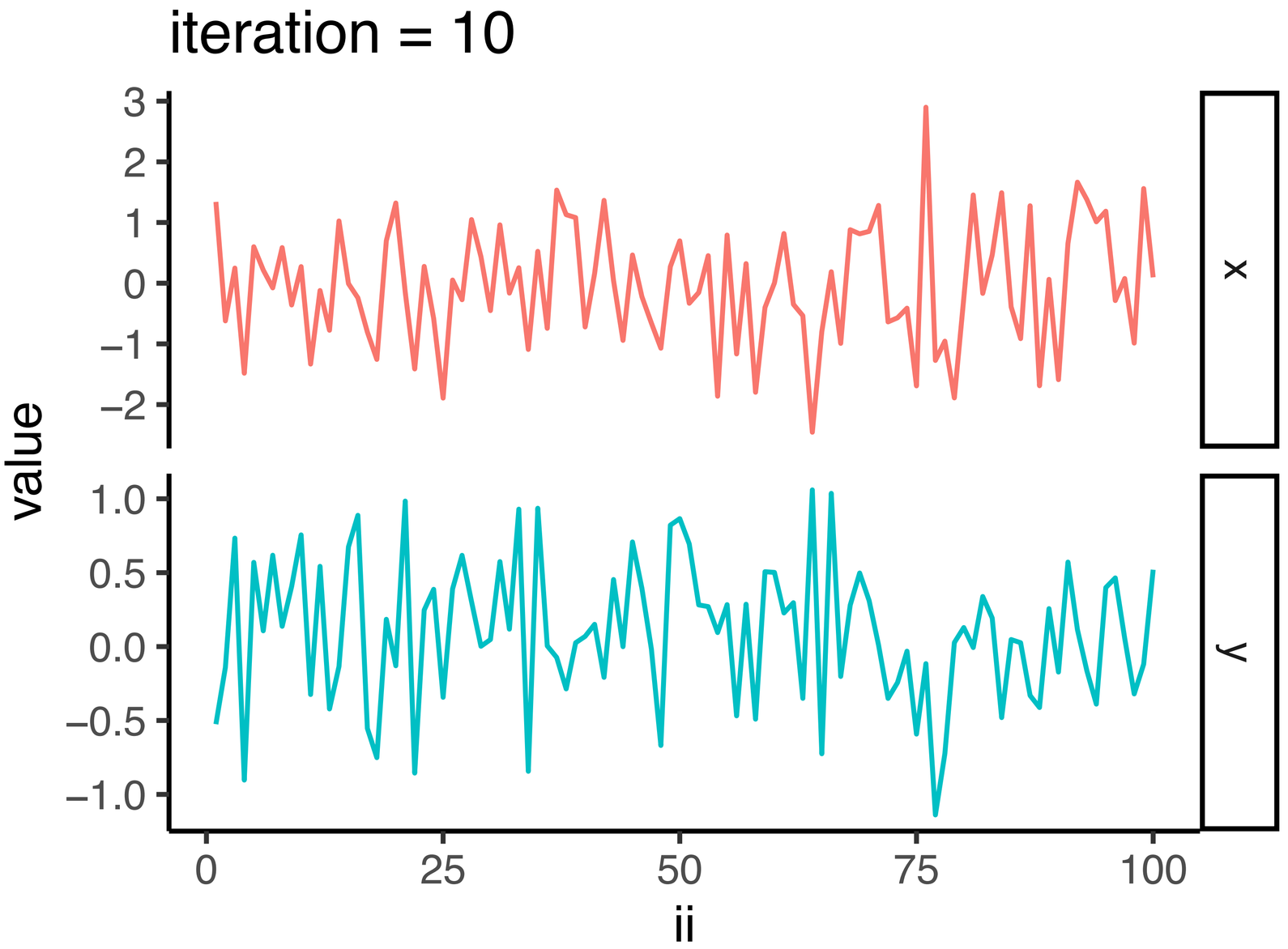




Let's generate "functions" with GP prior

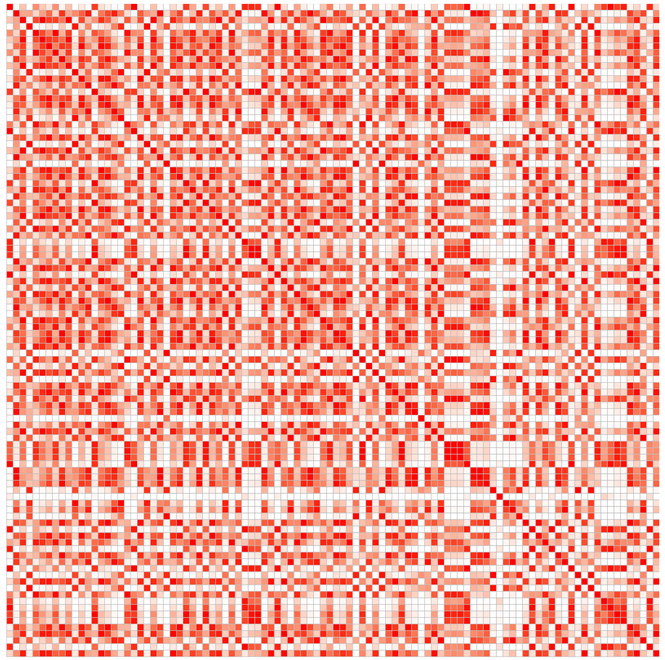
```
.data <- list(N=n, D=d, X=x)

gp.sample <-
    gp$sample(data = .data,
              chains = 1,
              iter_warmup=10,
              iter_sampling=111)
```

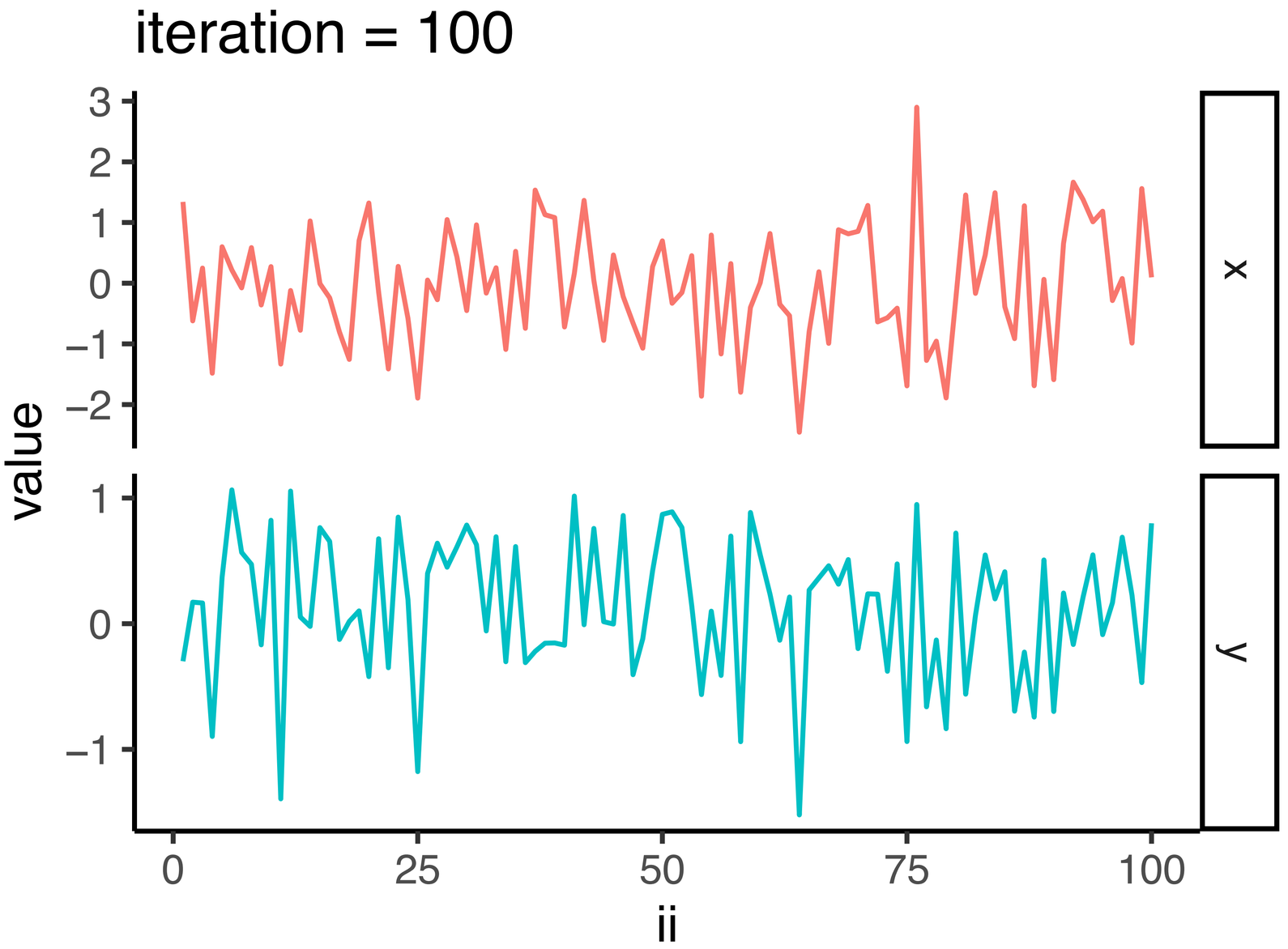




Let's generate "functions" with GP prior

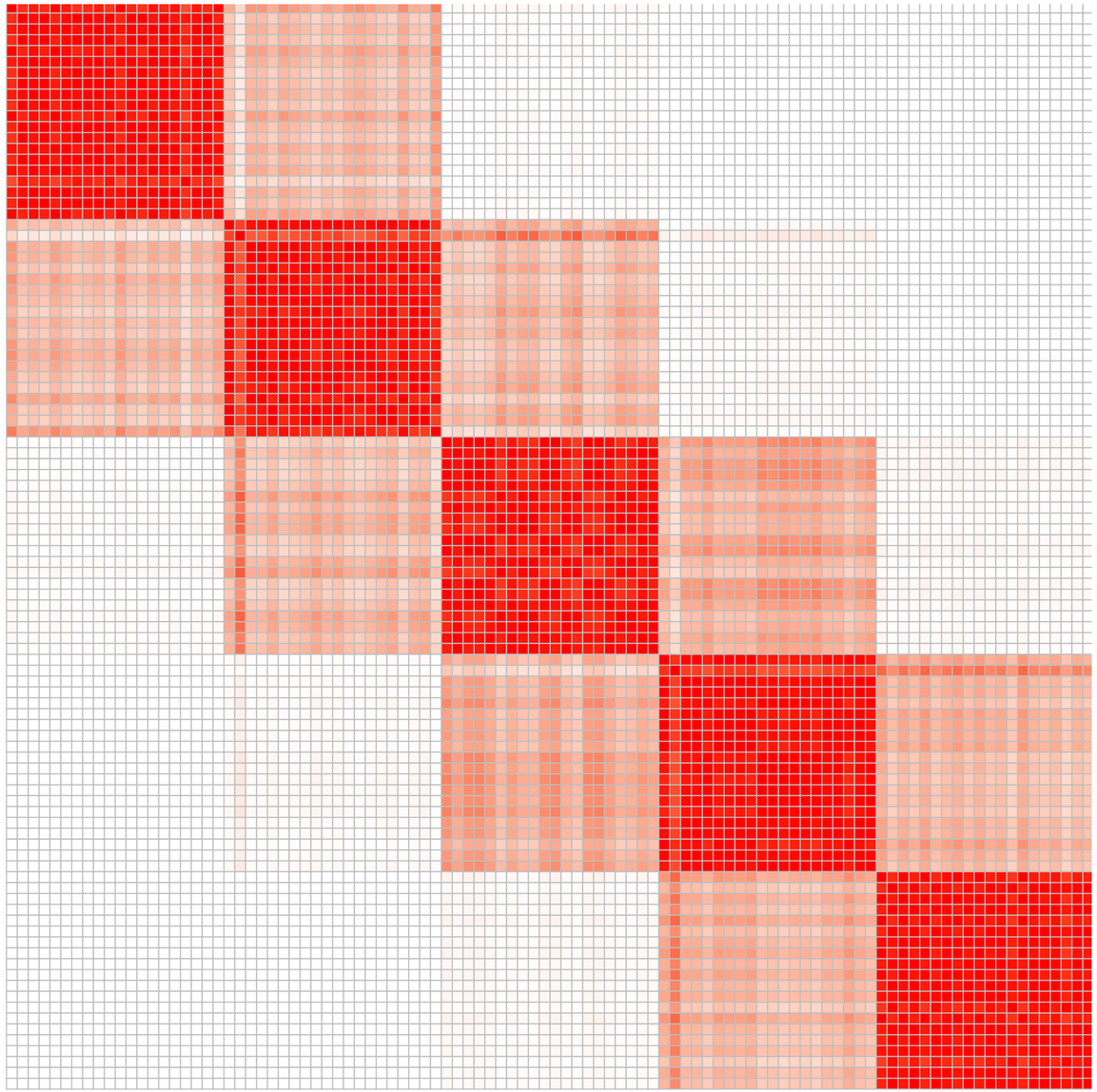
```
.data <- list(N=n, D=d, X=x)

gp.sample <-
    gp$sample(data = .data,
              chains = 1,
              iter_warmup=10,
              iter_sampling=111)
```





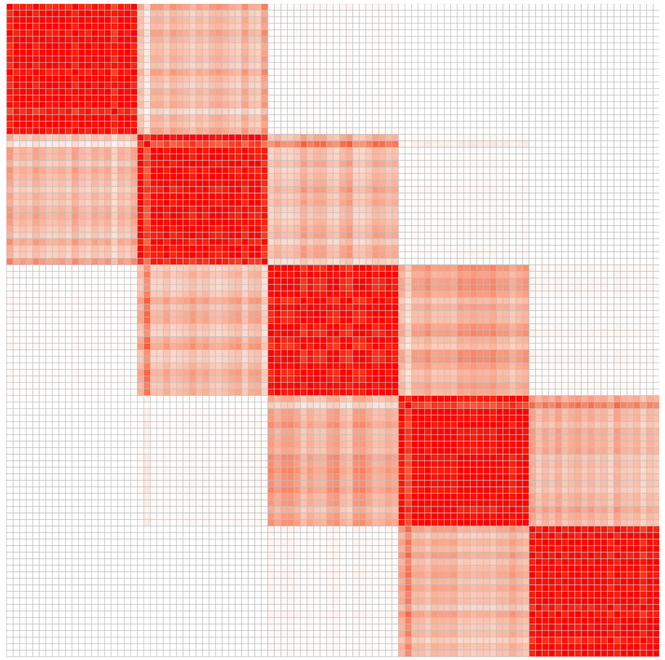
When data points are “similar” to each other

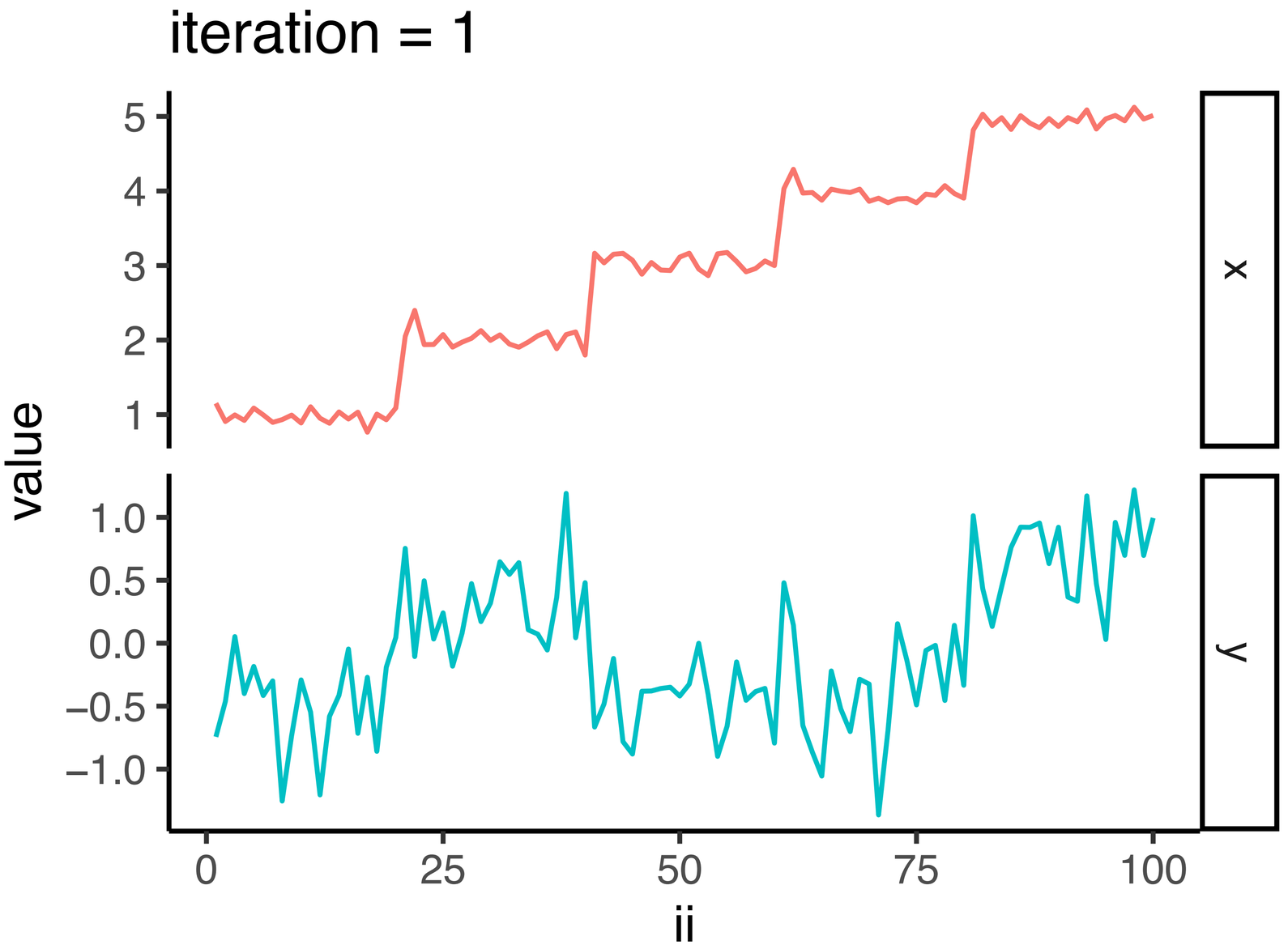
```
xx <- rep(1:5, each=20) + .rnorm(n, 1) * .1
x <- matrix(xx, nrow = n, ncol = 1)

rbf <- kernlab::rbfdot(sigma=1)
K <- kernlab::kernelMatrix(rbf, x)
```



Let's generate "functions" with the different kernel

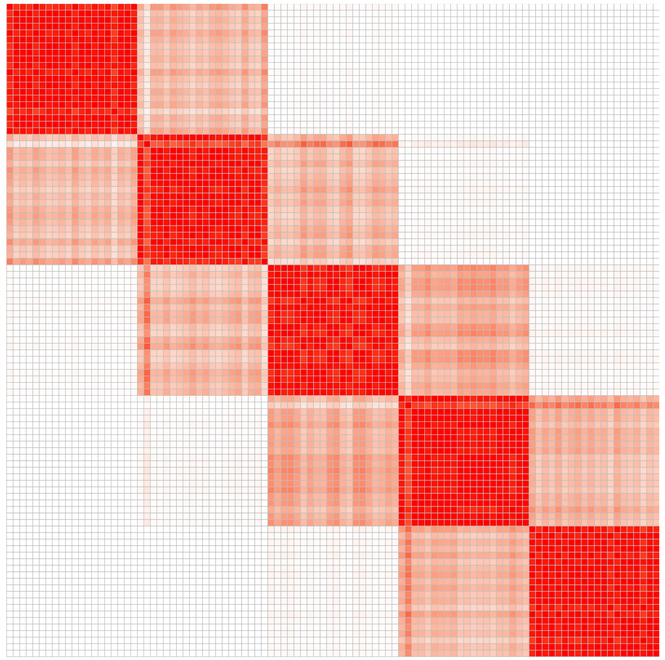
```
.data <- list(N=n, D=d, X=x)

gp.sample <-
    gp$sample(data = .data,
              chains = 1,
              iter_warmup=10,
              iter_sampling=111)
```

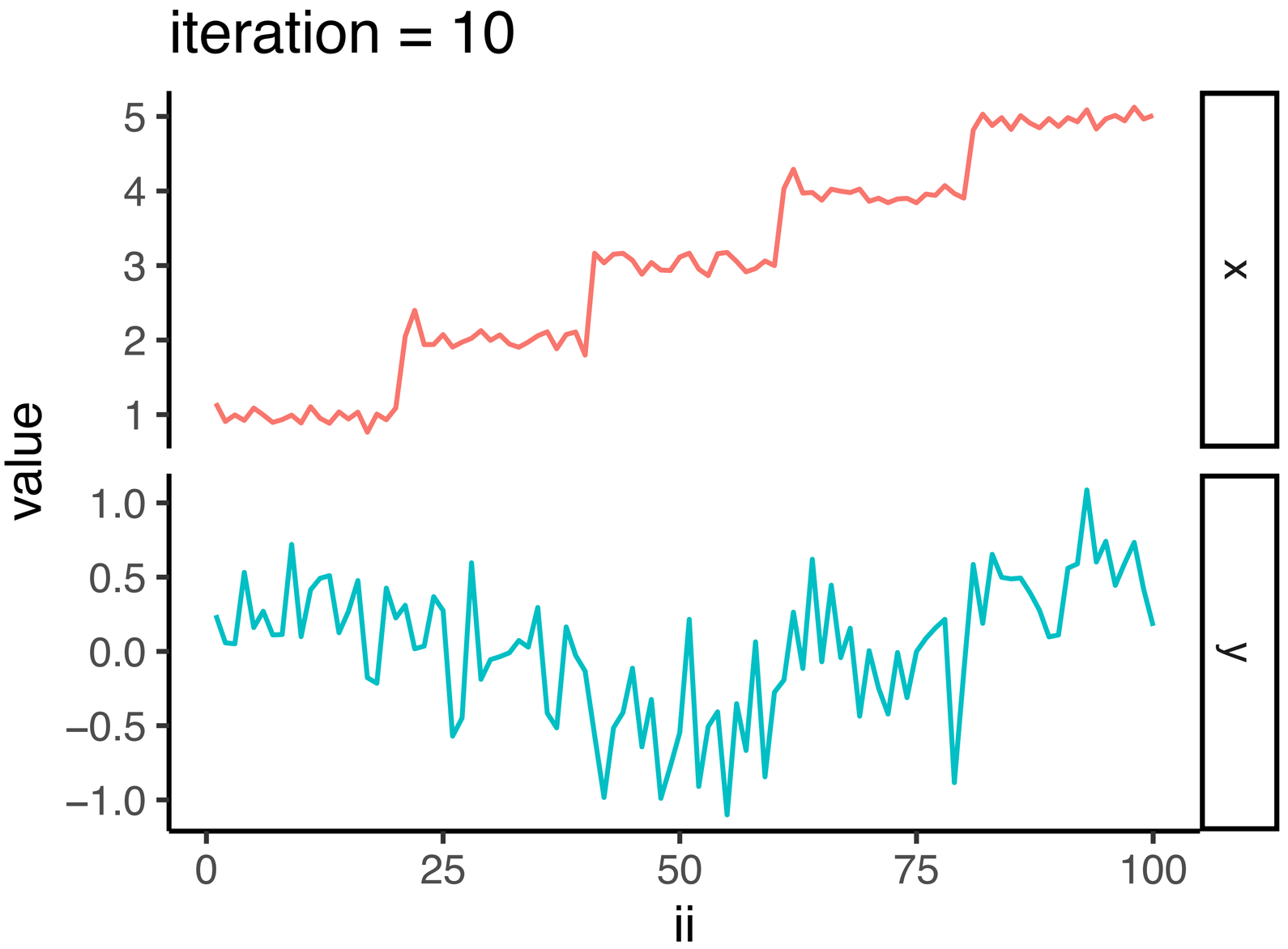




Let's generate "functions" with the different kernel

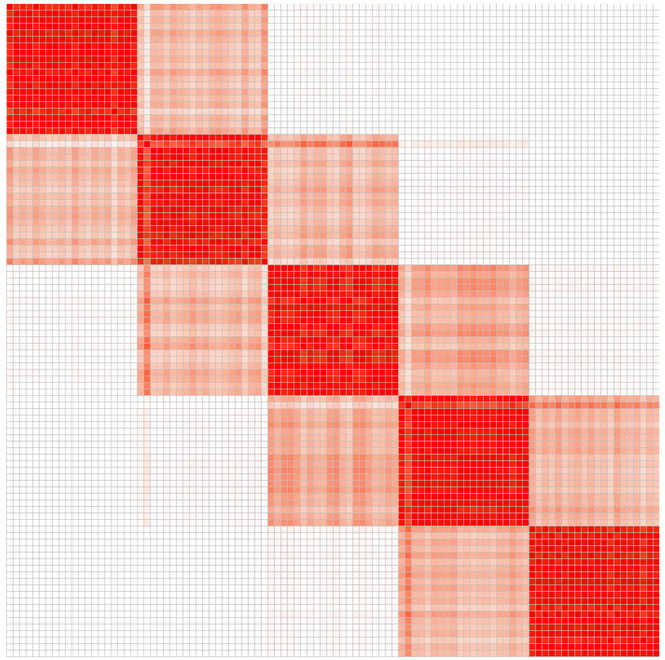
```
.data <- list(N=n, D=d, X=x)

gp.sample <-
    gp$sample(data = .data,
              chains = 1,
              iter_warmup=10,
              iter_sampling=111)
```

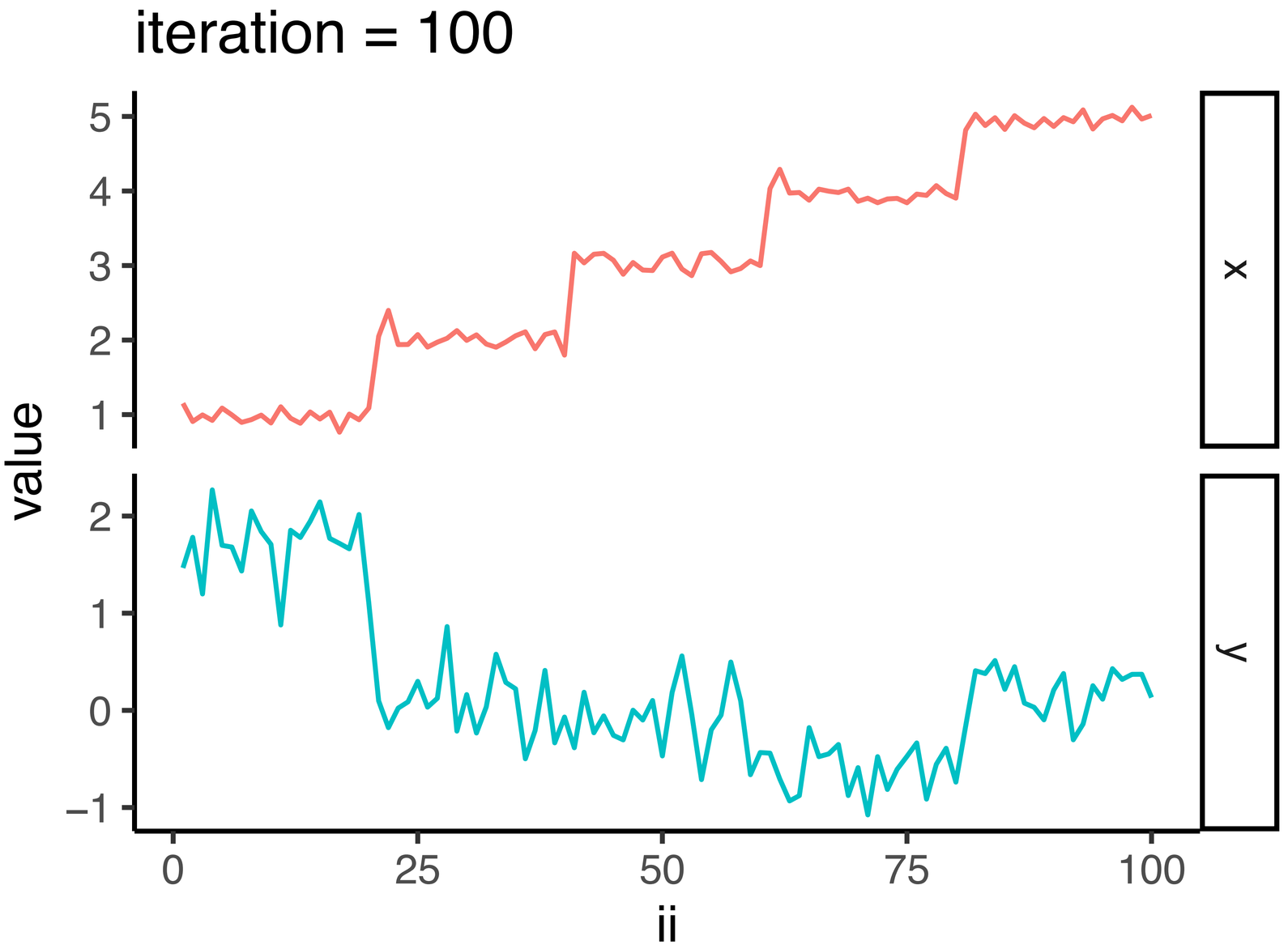




Let's generate "functions" with the different kernel

```
.data <- list(N=n, D=d, X=x)

gp.sample <-
    gp$sample(data = .data,
              chains = 1,
              iter_warmup=10,
              iter_sampling=111)
```



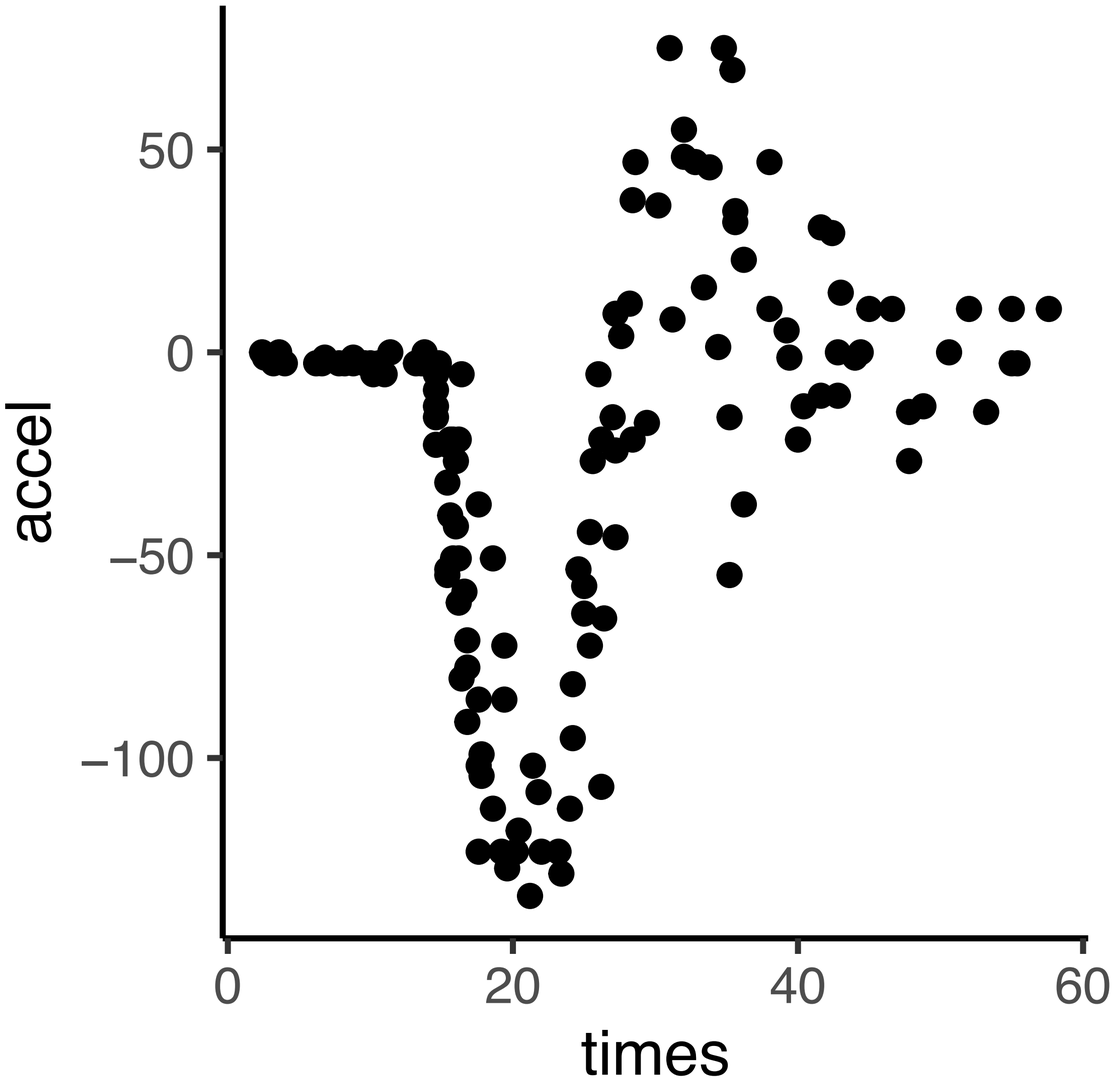


Gaussian Process regression

```
data(mcycle, package="MASS")

rbf <- kernlab::rbfdot(sigma=1)
xx <- mcycle[,1]
K <- kernlab::kernelMatrix(rbf, xx)
```

$$f \sim \mathcal{GP}(K)$$

$$\mathbf{y} \sim \mathcal{N}(\mathbf{f}, \sigma^2 I)$$

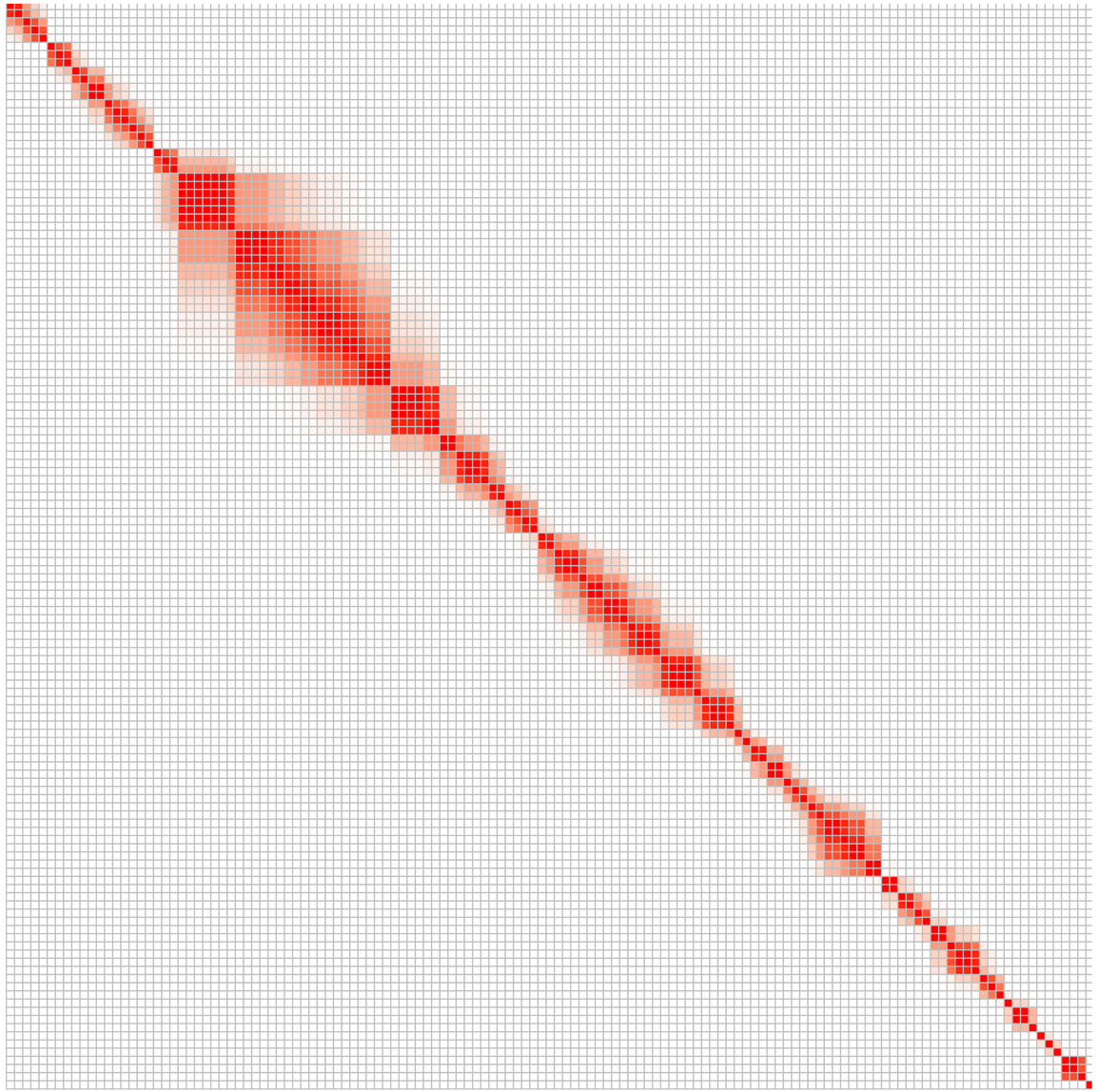


Gaussian Process regression

```
data(mcycle, package="MASS")

rbf <- kernlab::rbfdot(sigma=1)
xx <- mcycle[,1]
K <- kernlab::kernelMatrix(rbf, xx)
```

$$f \sim \mathcal{GP}(K)$$

$$\mathbf{y} \sim \mathcal{N}(\mathbf{f}, \sigma^2 I)$$



Writing down GP regression in stan

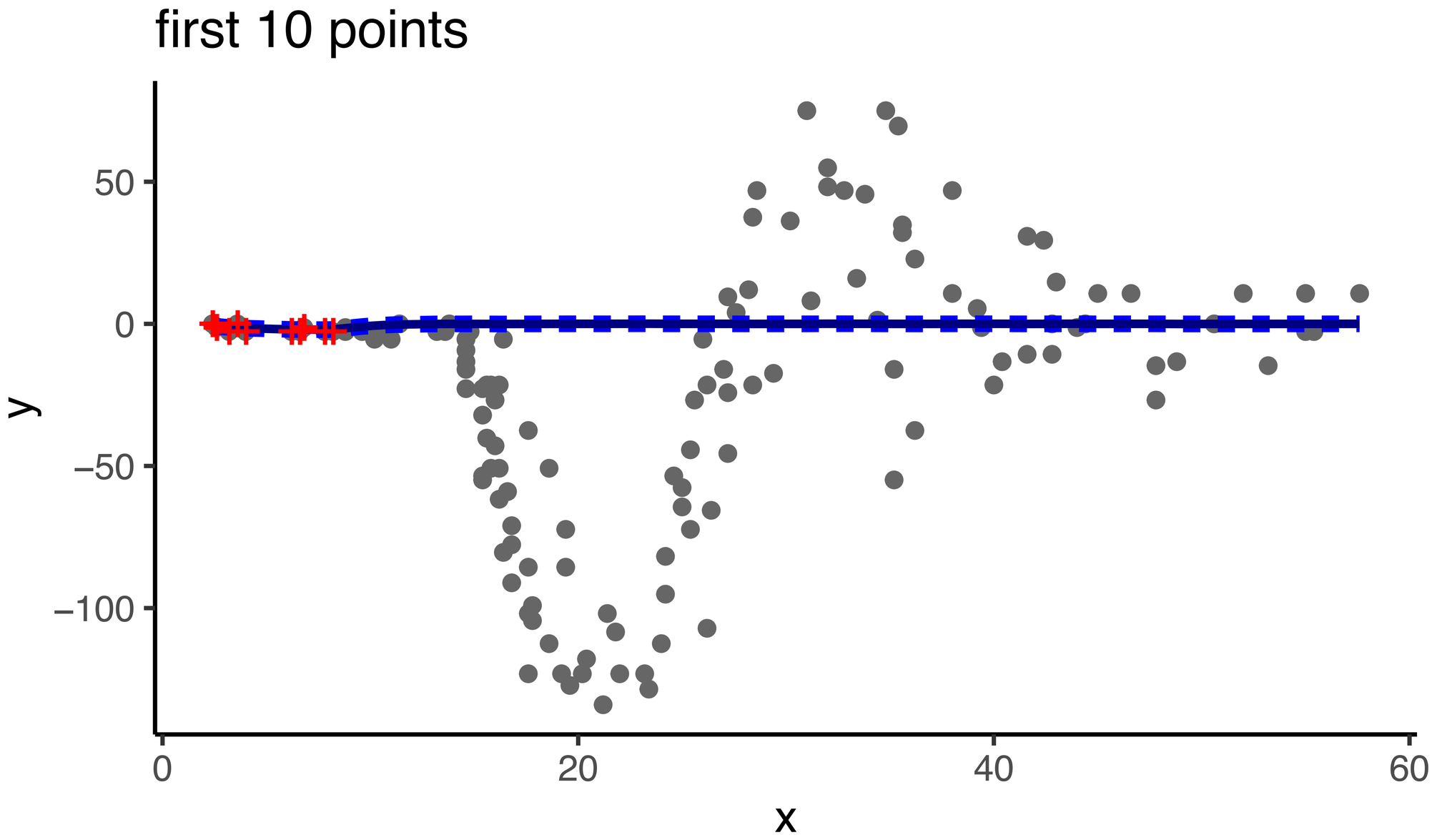
```
gp.model <- cmdstan_model("example_gp2.stan")
```

We can train Gaussian Process Regression models by MCMC

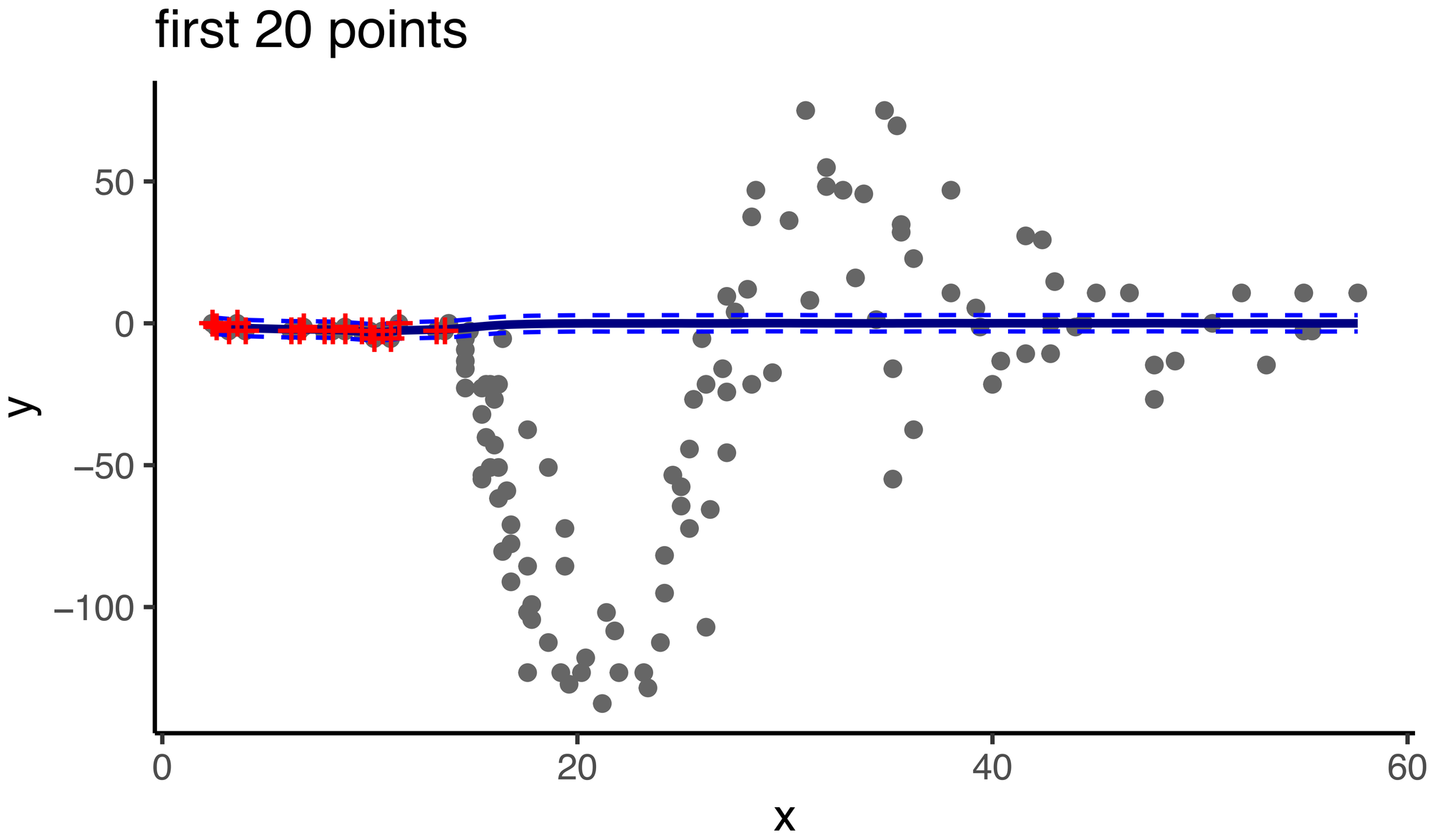
```
data(mcycle, package="MASS")
x <- mcycle[, 1, drop = FALSE]
y <- mcycle[, 2]
n <- nrow(x)
d <- ncol(x)
```

```
.data <- list(N=n, D=d, x=x, y=y, Ntest=n, xtest=x)
gp.sample <- gp.model$sample(data = .data,
                             chains = 5,
                             parallel_chains = 5)
```

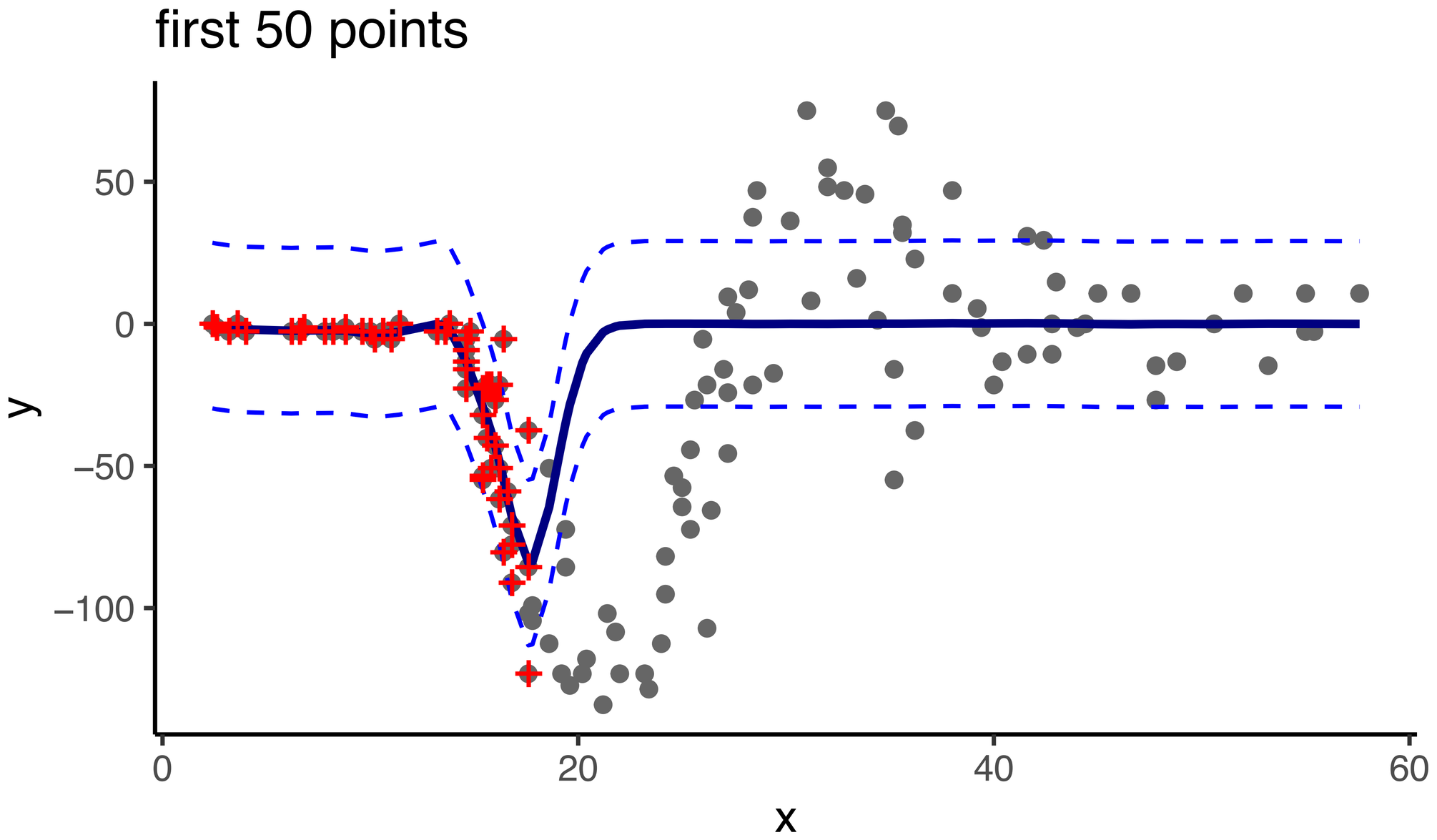
What if we give only a partial data set?



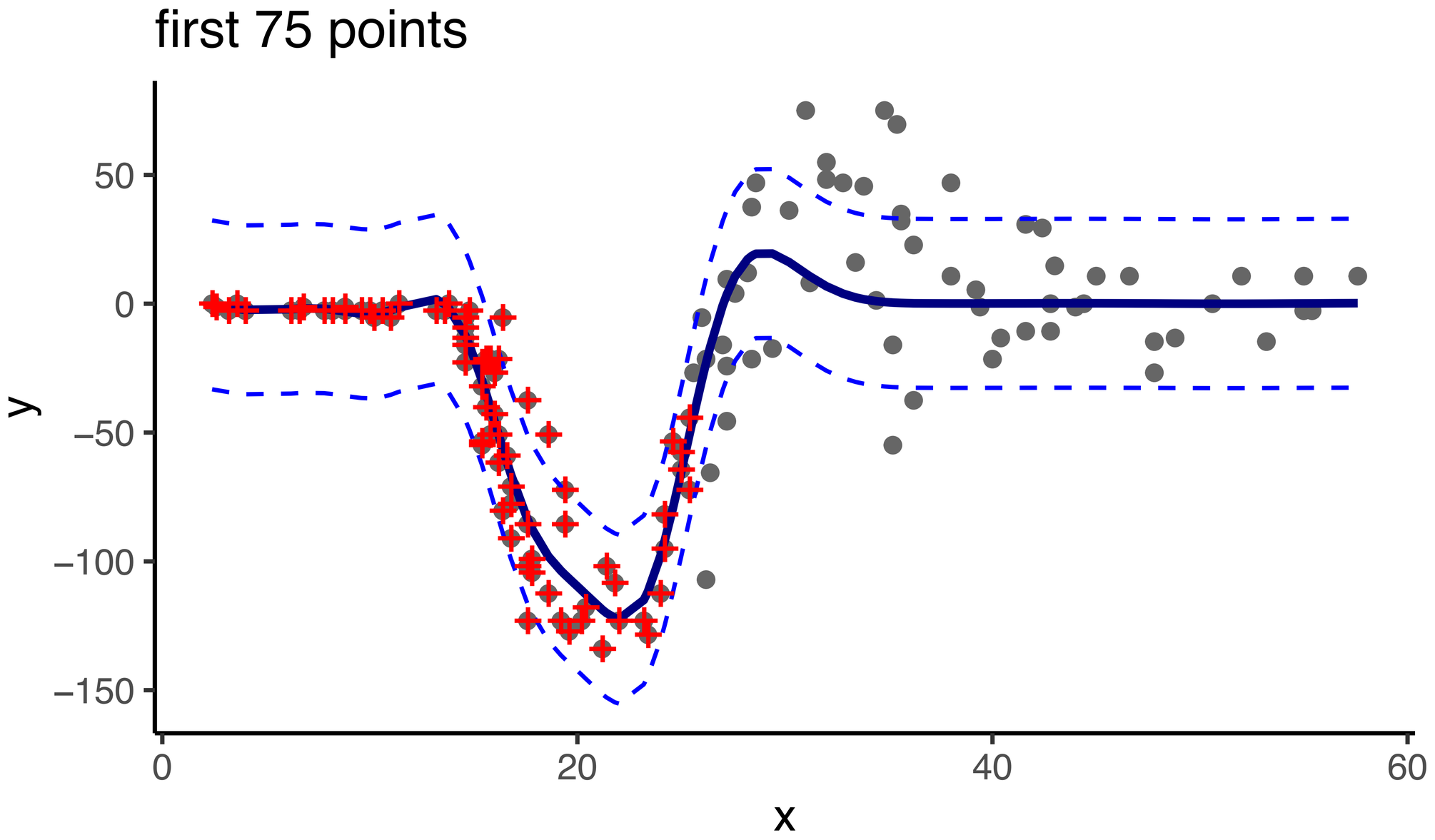
What if we give only a partial data set?



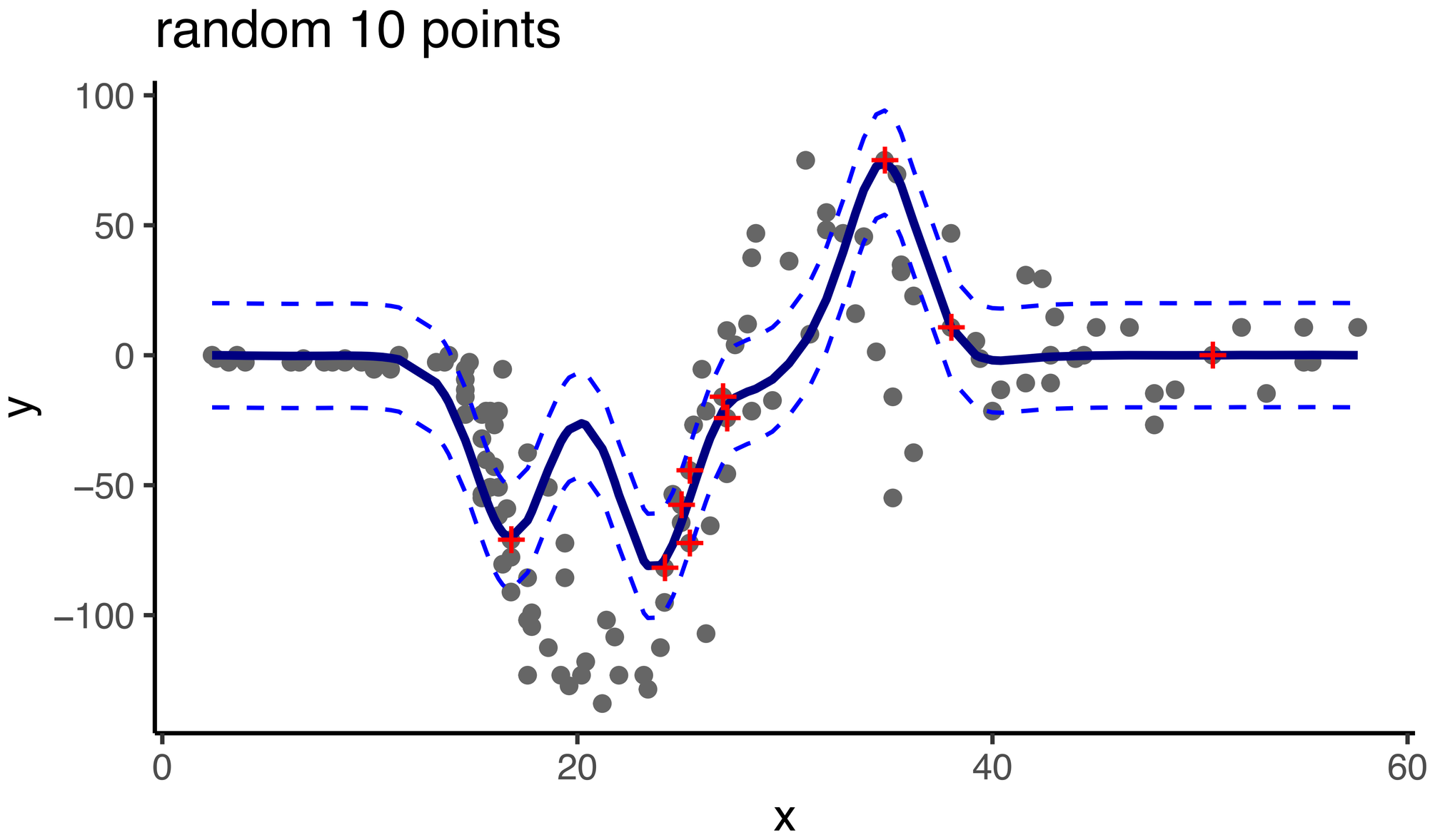
What if we give only a partial data set?



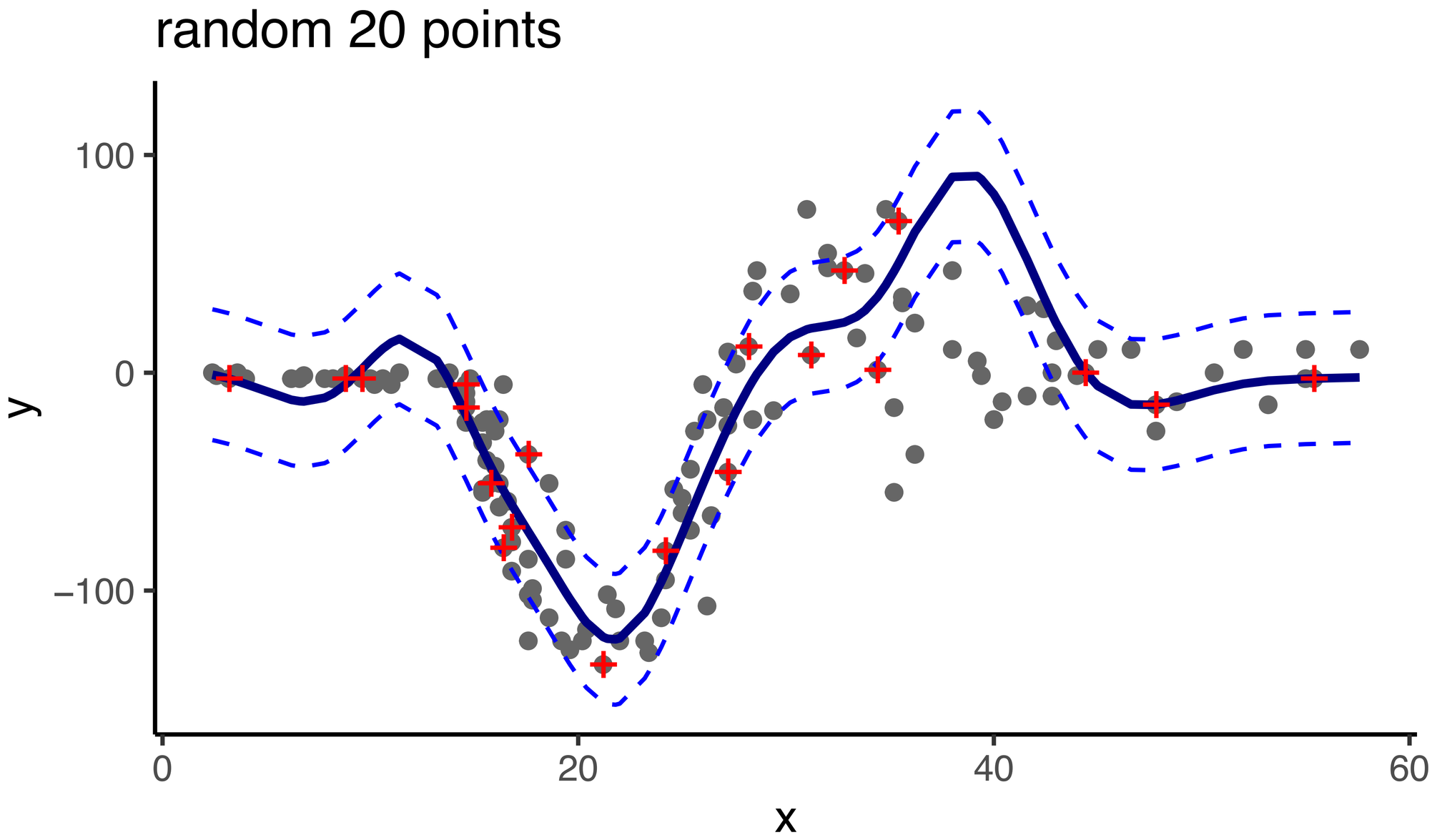
What if we give only a partial data set?



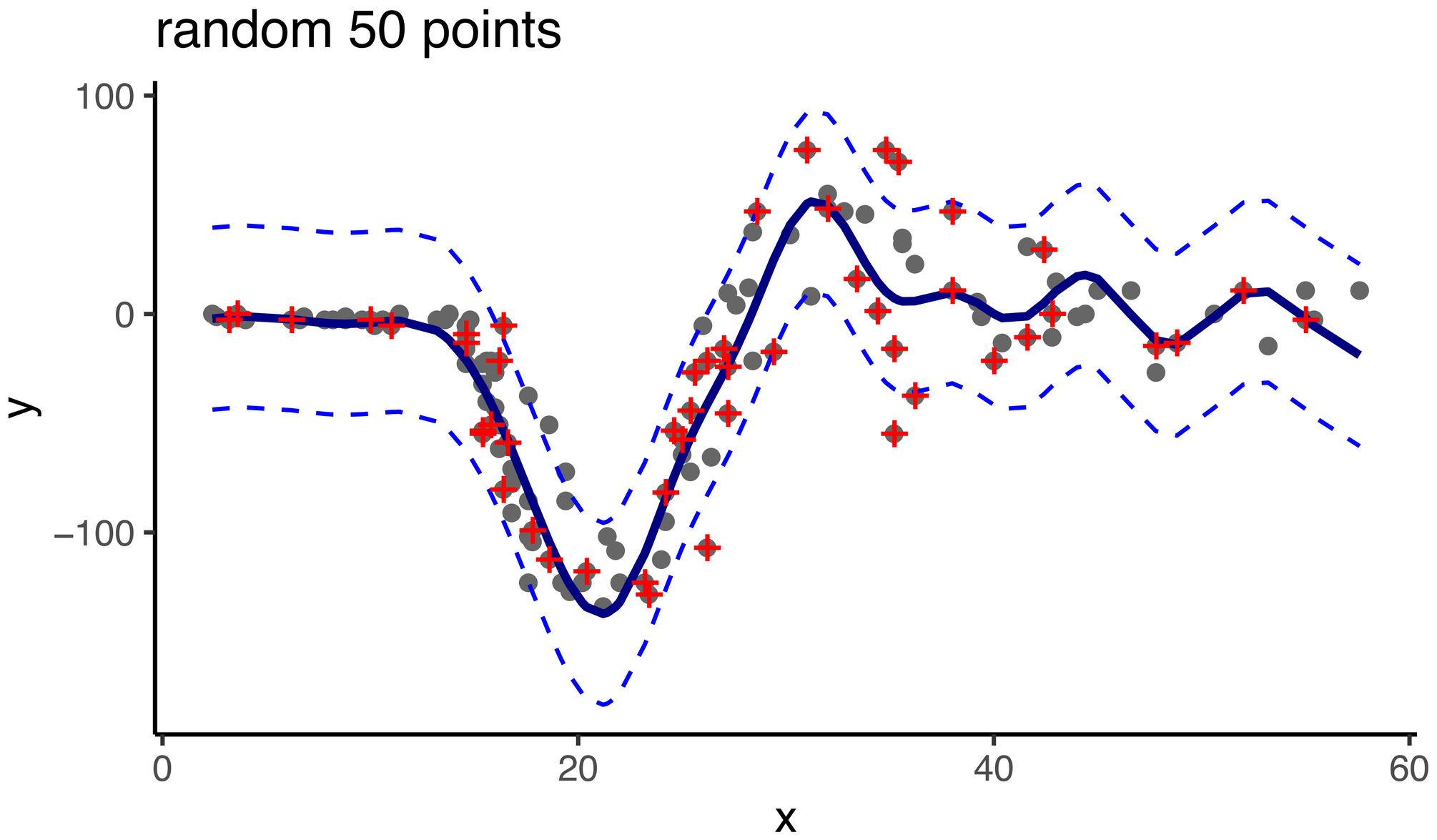
What if we give only a partial data set?



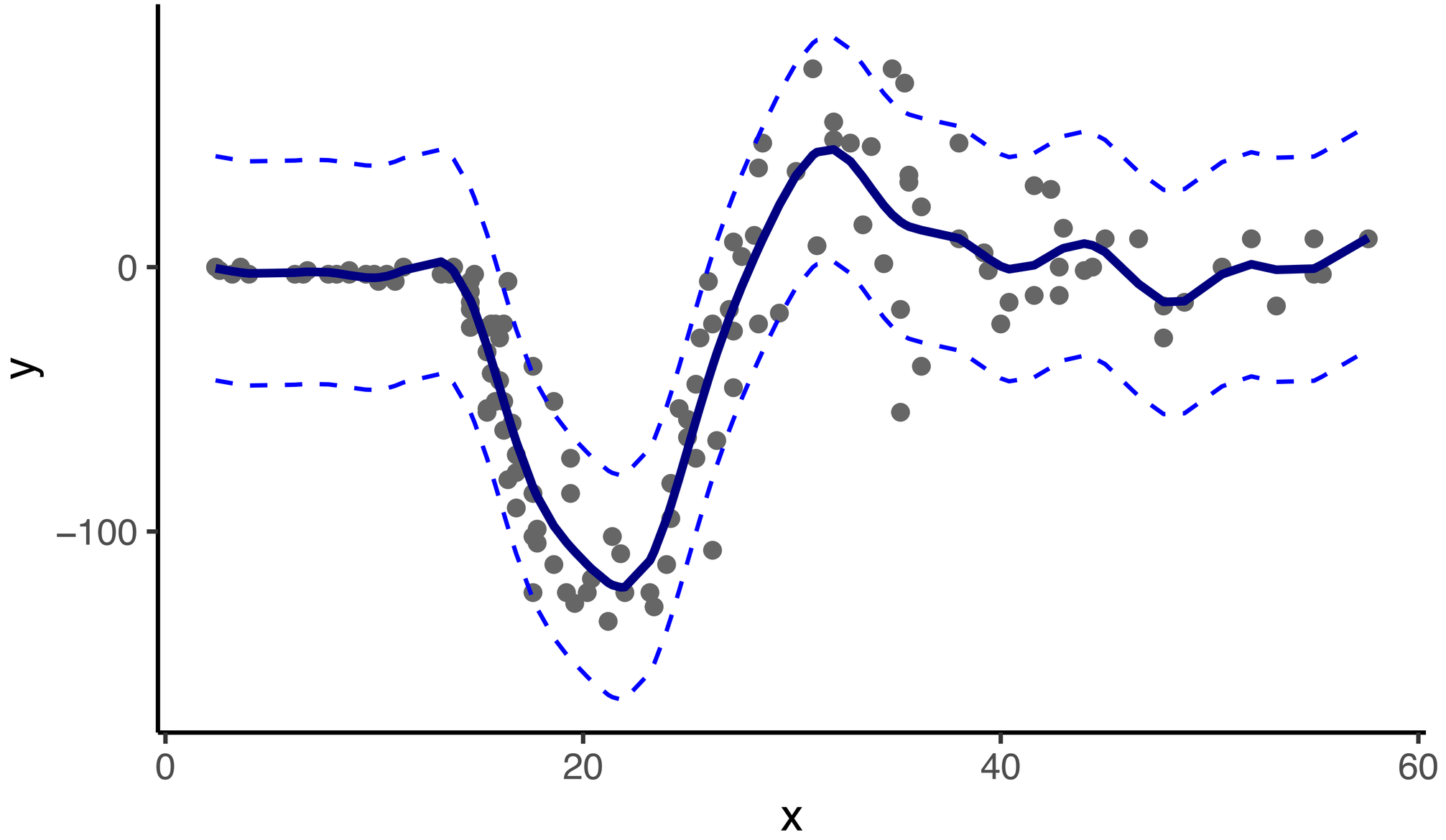
What if we give only a partial data set?



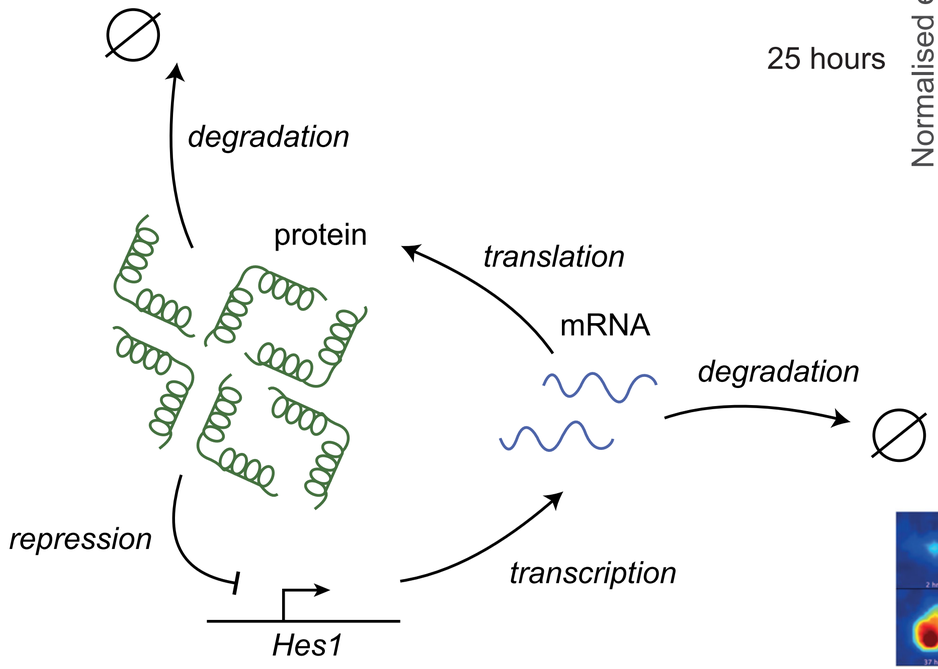
What if we give only a partial data set?



Full Gaussian Process regression training



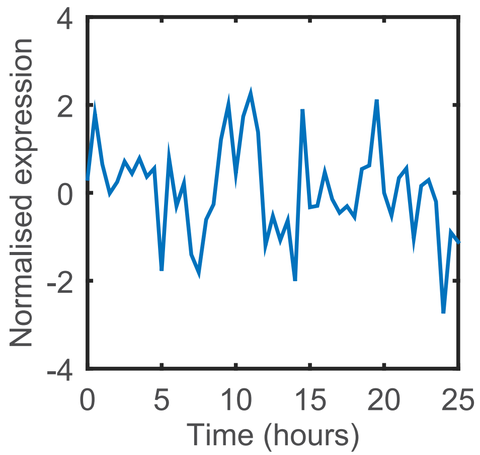
GP is a versatile tool for longitudinal studies

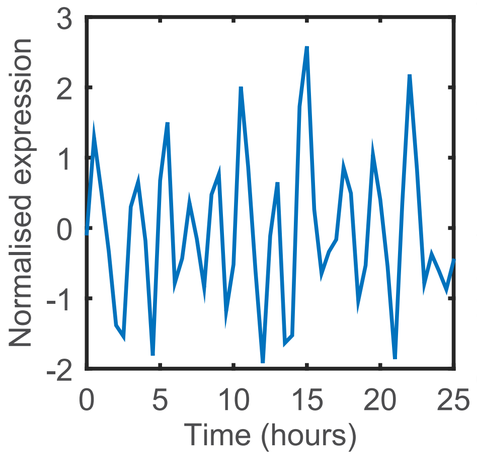


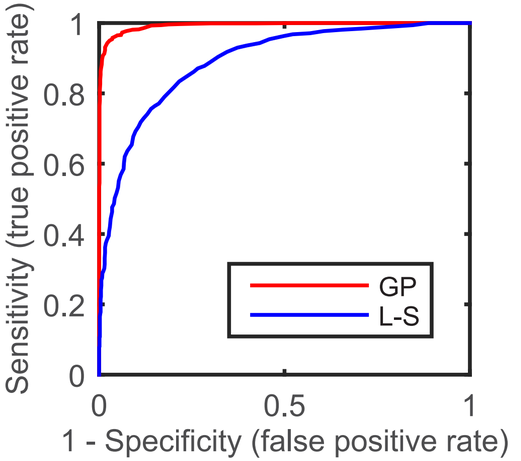
Goal: predict expression is oscillating or not

$\sigma_N^2 = 0.1$

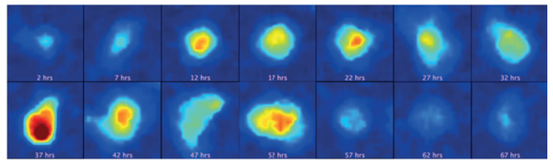
25 hours

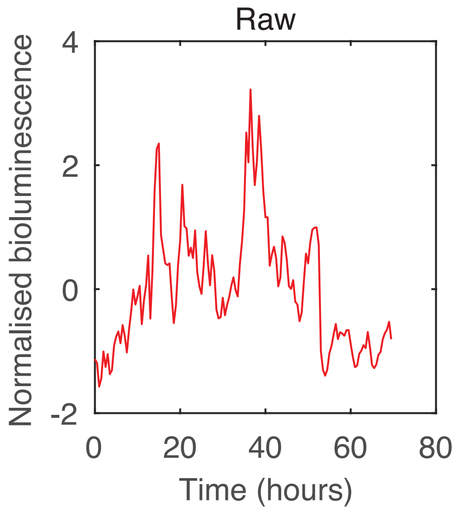


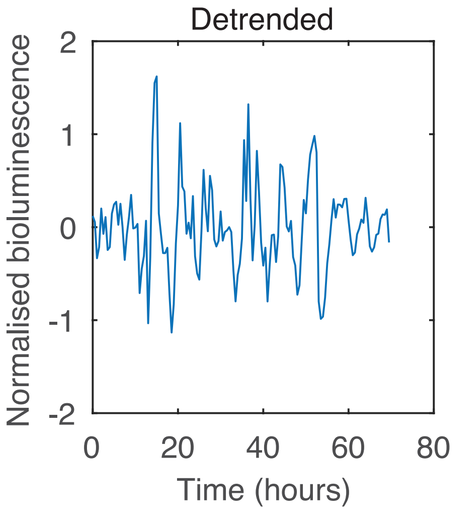




Hes1 promoter





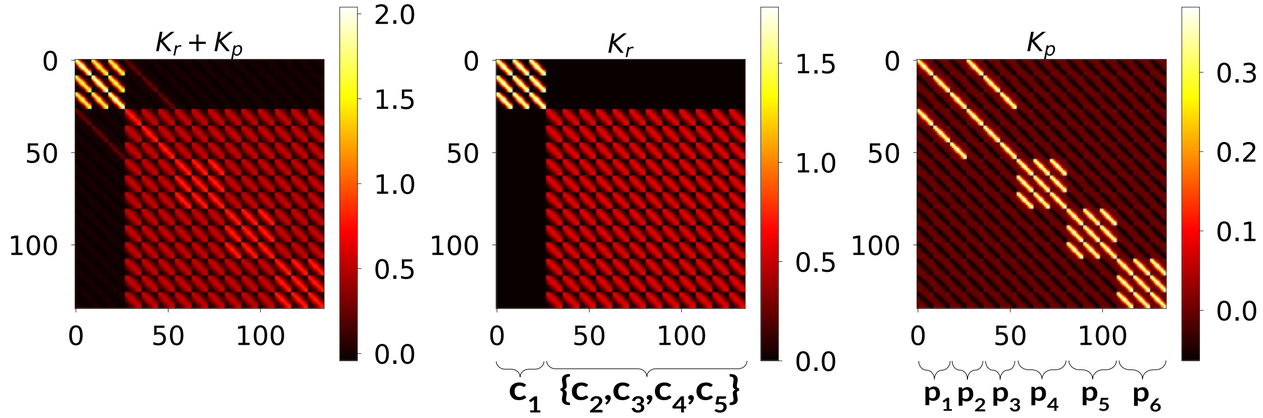


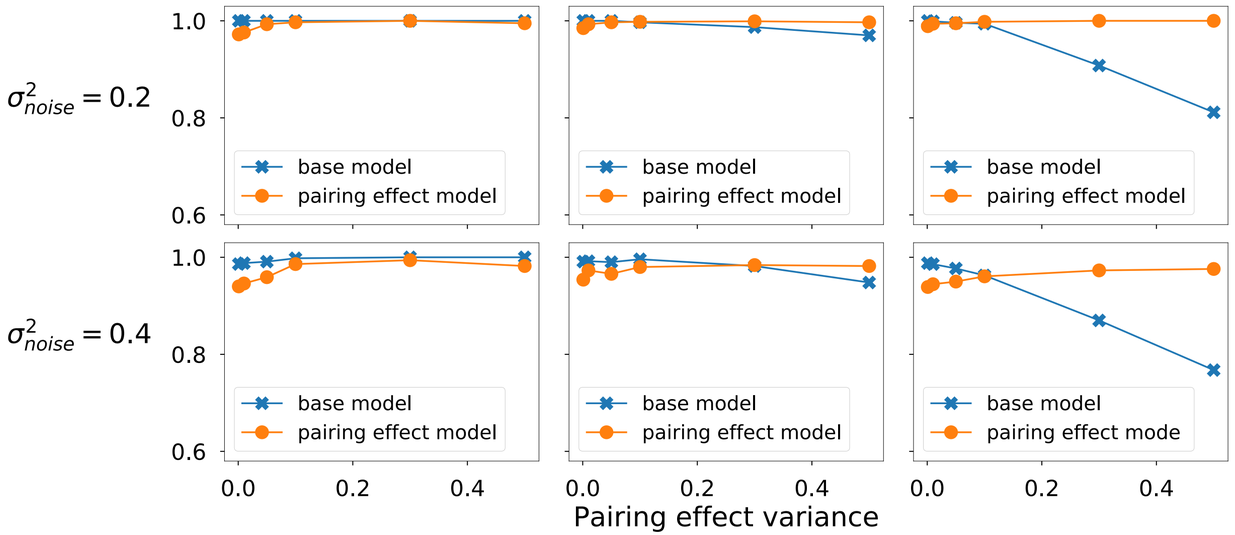
Use GP to identify general trends w/o overfitting

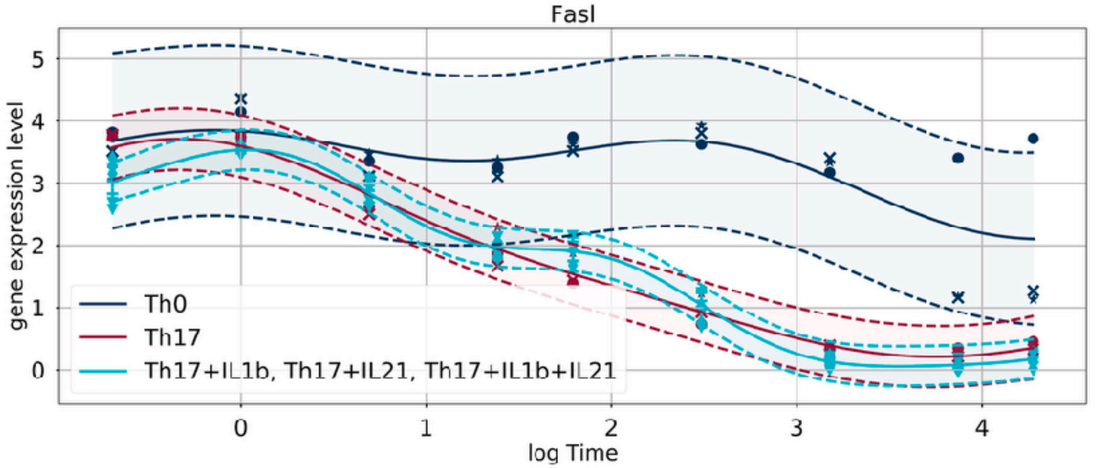
LSP: Lomb-Scargle Periodogram

Phillips *et al. PLoS Comp. Bio.* (2017)

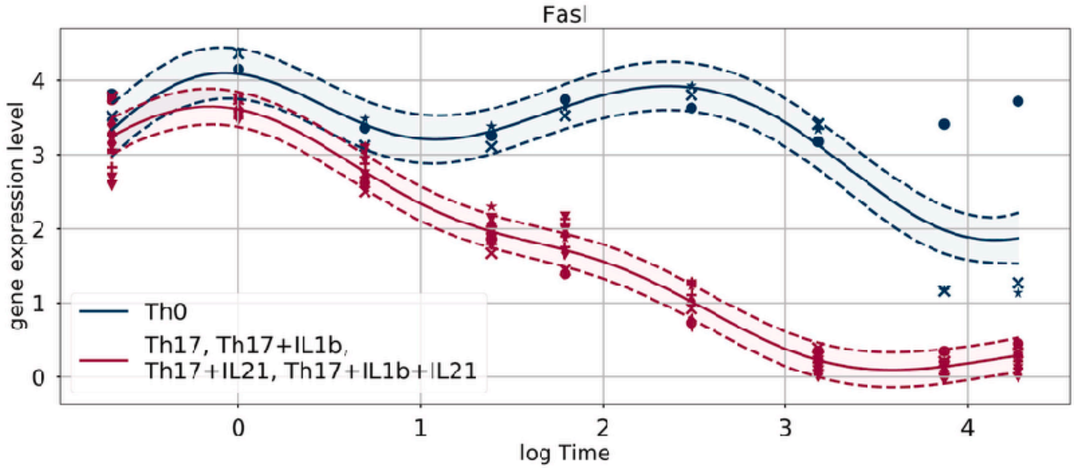
GP is a versatile tool for longitudinal studies







(a)



Vantini *et al.* (2022)

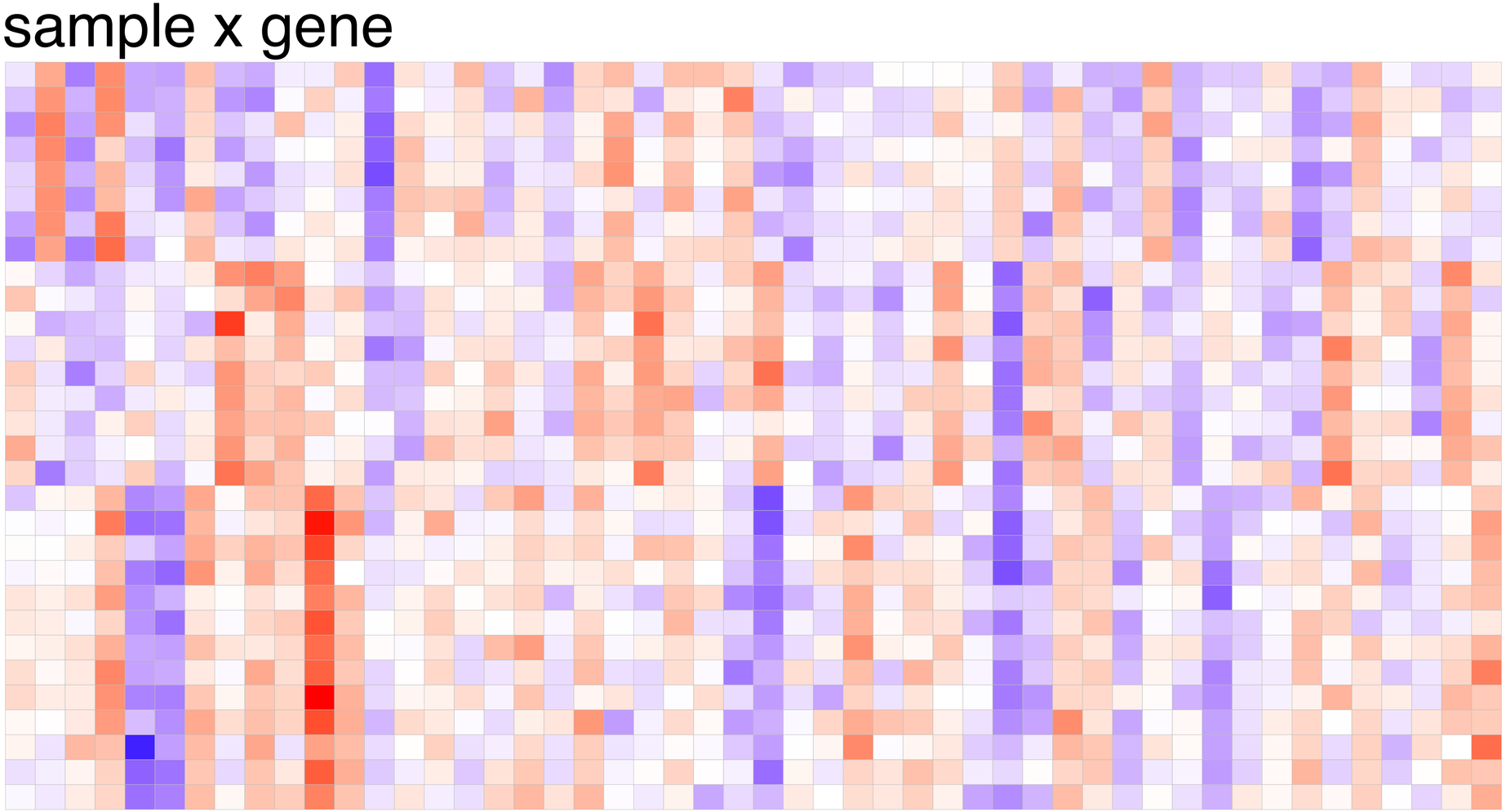
Today's lecture: Supervised Learning in Genomics

- **Non-parametric prediction methods**
  - When we don't have any idea of data-generation mechanisms.
  - Kernel (local) regression
  - Gaussian Process
- **Ensemble learning: the collective power of weak learners**
  - Expectation maximization
  - Mixture of linear regressions
- **Variable selection**
  - Challenges in high-dimensional prediction problems
  - Sparse regression models

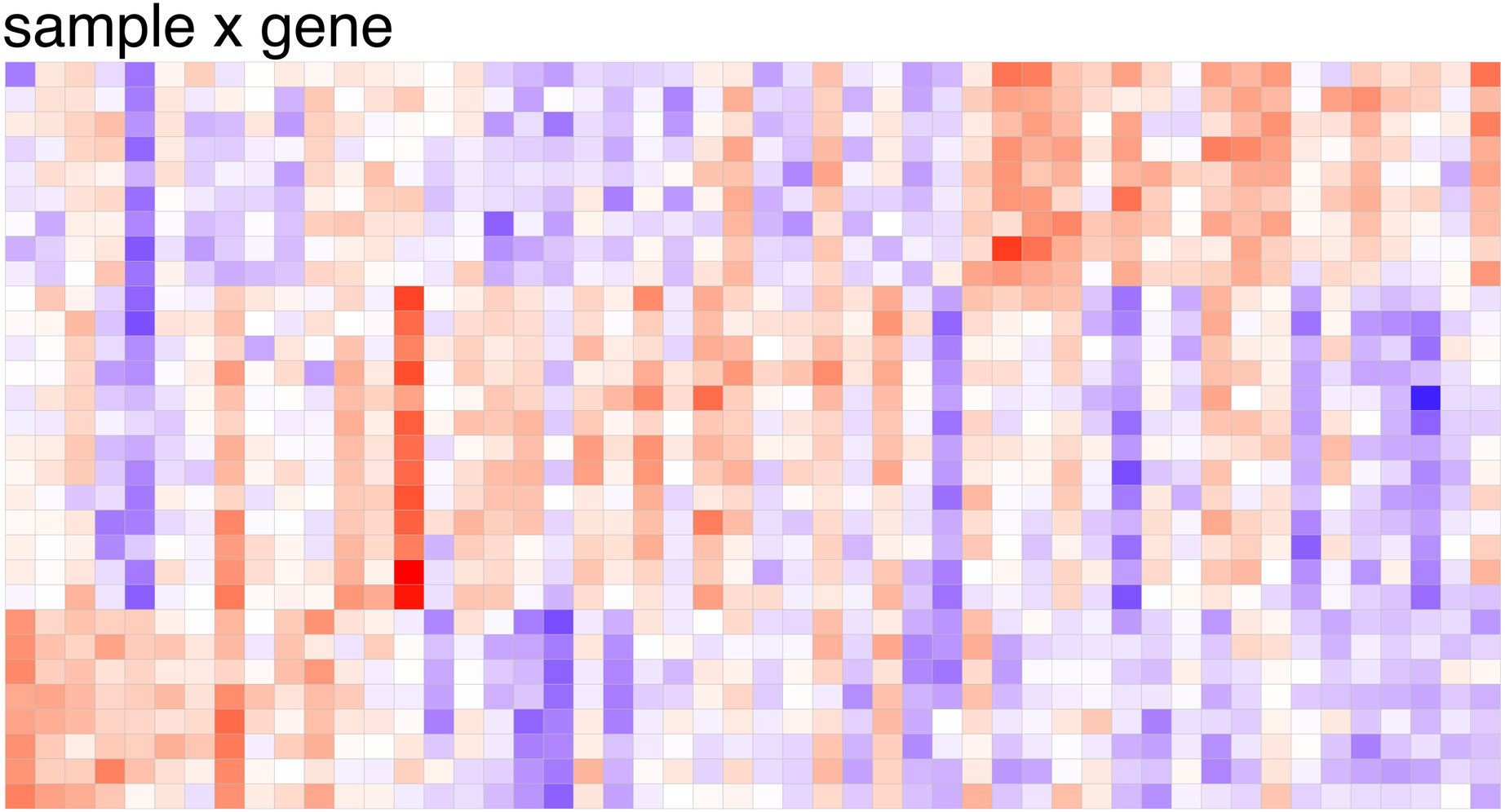
Motivation for Ensemble learning

- Training a highly-expressive model can be “expensive” requiring many data points and increasing the risk of overfitting
- A simple local rule/model/classifier is often powerful enough to capture local patterns

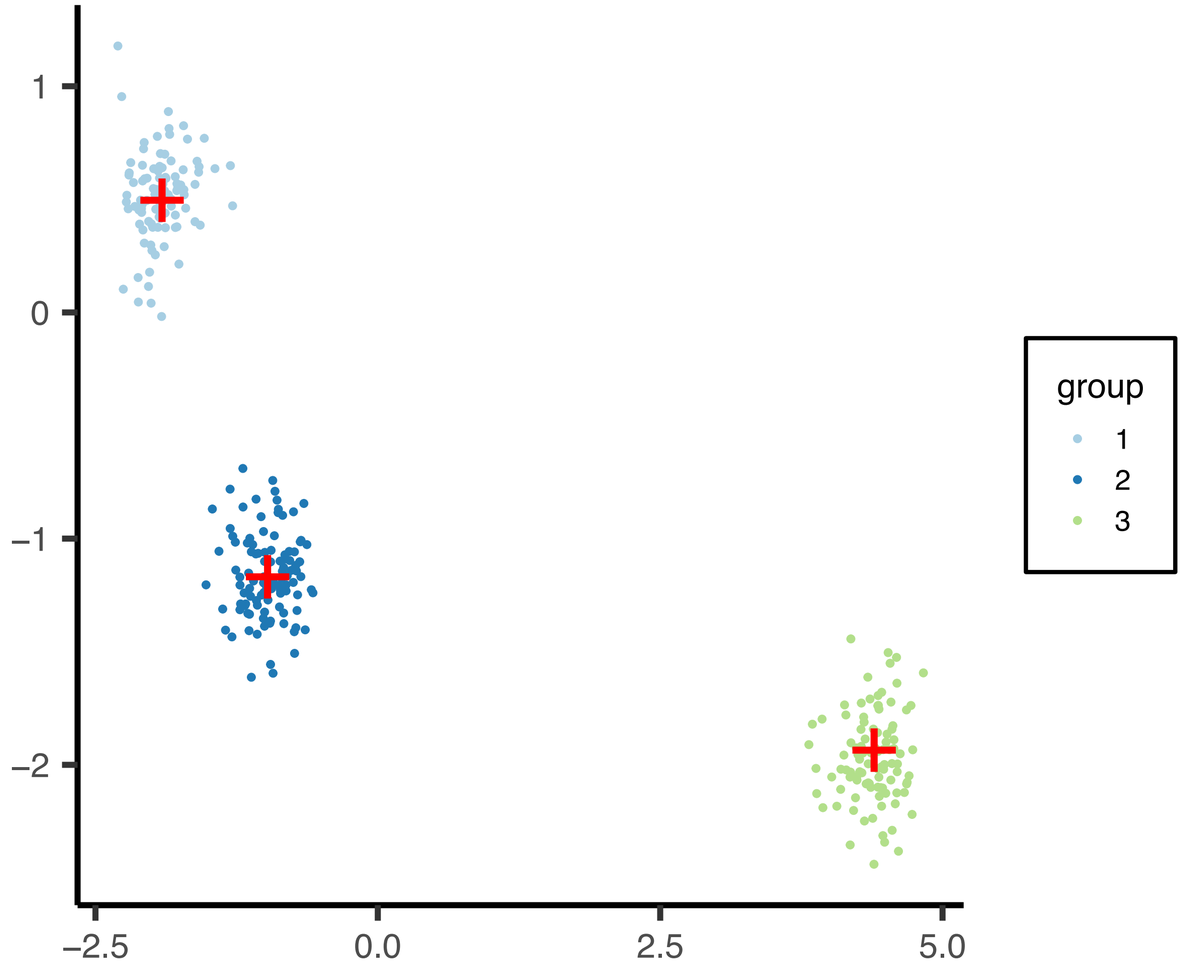
Warm-up example: mixture of Gaussian distributions



Warm-up example: mixture of Gaussian distributions



What do we want to know from data?



Two goals: recover (1) group membership and (2) the centroids (red marks)

A chicken-and-egg problem: guessing latent membership vs. parametric inference

- If we knew the membership of all the points, we can simply estimate the centre (e.g., taking sample mean within each cluster)
- If we knew the centre coordinates, we would be able to assign points to most probable groups easily based on distance from the centre points.
- Statistical answer: Solve the underlying inference problem.
- To a parameter estimator, the membership assignments are *hidden* (latent).

Gaussian Mixture Model (k-means)

1. Initialize μ_k (the centre of each group) and σ_k (the spread within each group)
2. Randomly assign group membership, $Z_{ik} = 1$ iff a point i belongs to a group k.
3. Generate: $\mathbf{x}_i | Z_{ik} = 1, \mu_k \sim \mathcal{N}(\mu_k, \sigma^2 I)$

How do we infer Z and μ, σ?

Expectation Maximization for GMM MLE

$$J \equiv \log \prod_{i=1}^{n} p(\mathbf{x}_i|\mu, \sigma)$$

Expectation Maximization for GMM MLE

$$
\begin{aligned}
J &\equiv \log \prod_{i=1}^{n} p(\mathbf{x}_i|\mu, \sigma) \\
&= \log \prod_{i=1}^{n} \sum_{Z} p(\mathbf{x}_i|Z, \mu, \sigma) p(Z)
\end{aligned}
$$

* It might be difficult to enumerate all the Z's... so let's introduce some other distributions that will help our "guessing" work, which we call it $q(Z)$

EM algorithm: What is the best way to guess latent variables?

$$\sum_i \log p(\mathbf{x}_i|\mu,\sigma) \quad = \quad \sum_{i=1}^{n} \log \sum_{Z_i} \frac{q(Z_i)}{q(Z_i)} p(\mathbf{x}_i|Z_i,\mu,\sigma)p(Z_i)$$

EM algorithm: What is the best way to guess latent variables?

$$\sum_i \log p(\mathbf{x}_i|\mu,\sigma) = \sum_{i=1}^{n} \log \sum_{Z_i} \frac{q(Z_i)}{q(Z_i)} p(\mathbf{x}_i|Z_i,\mu,\sigma)p(Z_i)$$

$$\underset{\text{Jensen}}{\geq} \sum_{i=1}^{n} \sum_{Z_i} q(Z_i) \log \frac{p(\mathbf{x}_i|Z_i,\mu,\sigma)p(Z_i)}{q(Z_i)}$$

EM algorithm: What is the best way to guess latent variables?

$$\sum_i \log p(\mathbf{x}_i|\mu,\sigma) = \sum_{i=1}^{n} \log \sum_{Z_i} \frac{q(Z_i)}{q(Z_i)} p(\mathbf{x}_i|Z_i,\mu,\sigma)p(Z_i)$$

$$\geq_{\text{Jensen}} \sum_{i=1}^{n} \sum_{Z_i} q(Z_i) \log \frac{p(\mathbf{x}_i|Z_i,\mu,\sigma)p(Z_i)}{q(Z_i)}$$

What is the best $q(Z)$?

EM algorithm: optimal E-step is to take the posterior probability

If $q(Z) = p(Z|\mathbf{x}_i, \mu, \sigma)$ (by Bayes rule),

$$= \sum_{i}^{n} \sum_{Z_i} p(Z_i|\mathbf{x}_i, \mu, \sigma) \log \frac{p(\mathbf{x}_i|Z_i, \mu, \sigma)p(Z_i)}{p(Z_i|\mathbf{x}_i\mu, \sigma)}$$

EM algorithm: optimal E-step is to take the posterior probability

If $q(Z) = p(Z|\mathbf{x}_i, \mu, \sigma)$ (by Bayes rule),

$$
\begin{aligned}
&= \sum_{i}^{n} \sum_{Z_i} p(Z_i|\mathbf{x}_i, \mu, \sigma) \log \frac{p(\mathbf{x}_i|Z_i, \mu, \sigma)p(Z_i)}{p(Z_i|\mathbf{x}_i\mu, \sigma)} \\
&= \sum_{i}^{n} \sum_{Z_i} p(Z_i|\mathbf{x}_i, \mu, \sigma) \log \frac{p(\mathbf{x}_i, Z_i|\mu, \sigma)p(\mathbf{x}_i|\mu, \sigma)}{p(\mathbf{x}_i, Z_i|\mu, \sigma)}
\end{aligned}
$$

EM algorithm: optimal E-step is to take the posterior probability

If $q(Z) = p(Z|\mathbf{x}_i, \mu, \sigma)$ (by Bayes rule),

$$
\begin{aligned}
&= \sum_i^n \sum_{Z_i} p(Z_i|\mathbf{x}_i, \mu, \sigma) \log \frac{p(\mathbf{x}_i|Z_i, \mu, \sigma)p(Z_i)}{p(Z_i|\mathbf{x}_i\mu, \sigma)} \\
&= \sum_i^n \sum_{Z_i} p(Z_i|\mathbf{x}_i, \mu, \sigma) \log \frac{p(\mathbf{x}_i, Z_i|\mu, \sigma)p(\mathbf{x}_i|\mu, \sigma)}{p(\mathbf{x}_i, Z_i|\mu, \sigma)} \\
&= \sum_{i=1}^n \underbrace{\left[\sum_{Z_i} p(Z_i|\mathbf{x}_i, \mu, \sigma)\right]}_{=1} \log p(\mathbf{x}_i|\mu, \sigma) \\
&= \sum_i \log p(\mathbf{x}_i|\mu, \sigma)
\end{aligned}
$$

The inequality becomes equality.

Expectation Maximization algorithm = expected MLE

The goal:

$$\log p(X|\mu,\sigma) = \log \sum_Z p(X,Z|\mu,\sigma) \tag{1}$$

$$\geq_{\text{Jensen}} \sum_Z \underbrace{p(Z|X,\mu,\sigma)}_{\text{E-step}} \underbrace{\log p(X|Z,\mu,\sigma)}_{\text{M-step}} \tag{2}$$

$$= \underbrace{\mathbb{E}_{p(Z|X,\mu,\sigma)}}_{\text{E-step}} \left[\underbrace{\log p(X|Z,\mu,\sigma)}_{\text{M-step}} \right] \tag{3}$$

(Z is discrete, e.g., a membership indicator)

Solution: **Maximize** the lower bound by taking **the expectation** over the posterior probability.

EM algorithm of GMM: E-step

Log-likelihood under some group (μ_k and σ_k):

$$\log p(\mathbf{x}_i|\mu_k, \sigma_k) = \log \mathcal{N}(\mathbf{x}_i|\mu_k, \sigma_k)$$

How to estimate the posterior?

$$p(Z_{ik}|\mathbf{x}_i, \mu, \sigma) = \frac{\exp\{\log p(\mathbf{x}_i|\mu_k, \sigma_k)\}}{\sum_{k'} \exp\{\log p(\mathbf{x}_i|\mu_{k'}, \sigma_{k'})\}}$$

Remark: We can stochastically sample $Z_{ik} = 1$ with the posterior probability.

EM algorithm of GMM: M-step

Maximization step to optimize model parameters

Let this expected lower-bound (ELBO)

$$\mathcal{L}(\mathbf{x}_i; \{\mu_k\}, \{\sigma_k\}) = \sum_{i=1}^{n} \sum_{k=1}^{K} Z_{ik} \log p(\mathbf{x}_i | Z_{ik}, \mu_k, \sigma_k)$$

Given Z, what are the unknown? We can take gradient steps (e.g., `torch`)

$$\mu_k^{(t)} \leftarrow \mu_k^{(t-1)} + \rho \nabla_{\mu_k} \sum_i \mathcal{L}(\mathbf{x}_i)$$

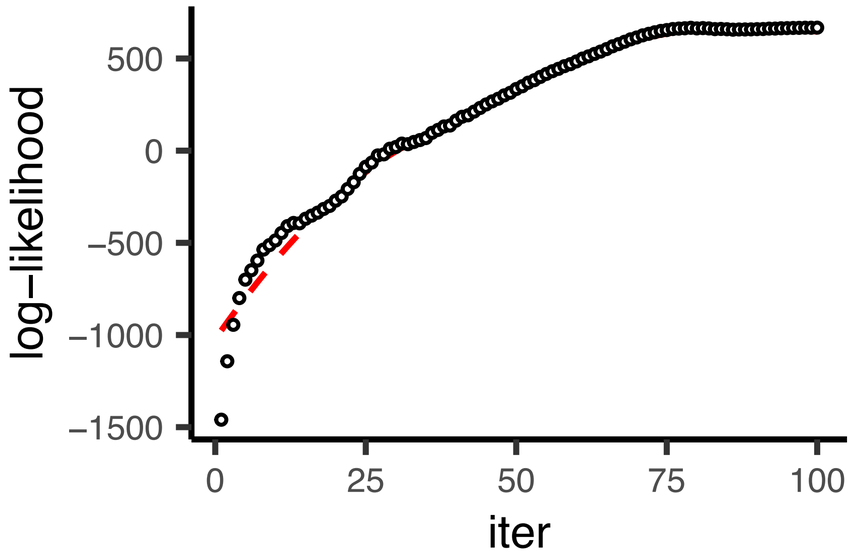
$$\sigma_k^{(t)} \leftarrow \sigma_k^{(t-1)} + \rho \nabla_{\sigma_k} \sum_i \mathcal{L}(\mathbf{x}_i)$$

Alternate E- and M-step until convergence

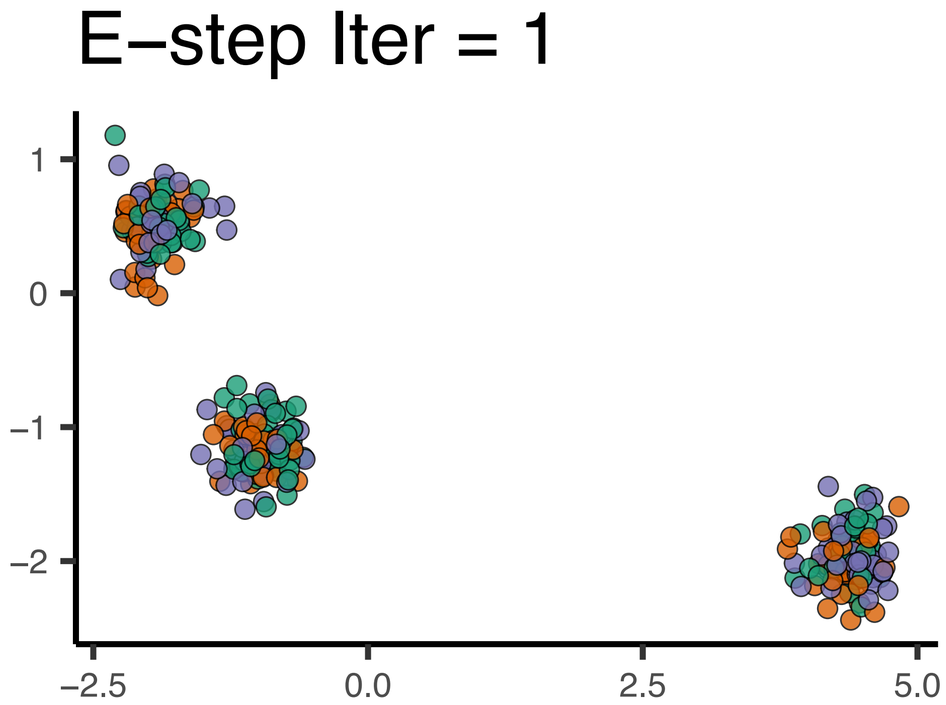
```
for(tt in 1:100){
    rand.idx <- take.estep()
    z <- nnf_one_hot(rand.idx)
    llik <- take.mstep(z)
}
```

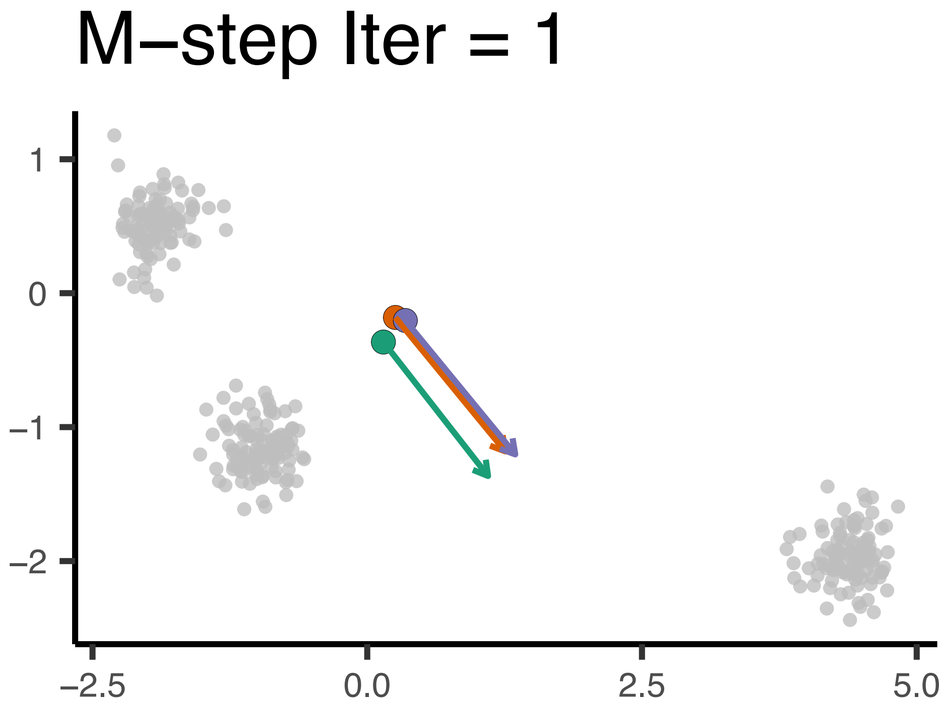
Find the details here:

`https://github.com/STAT540-UBC/lectures`



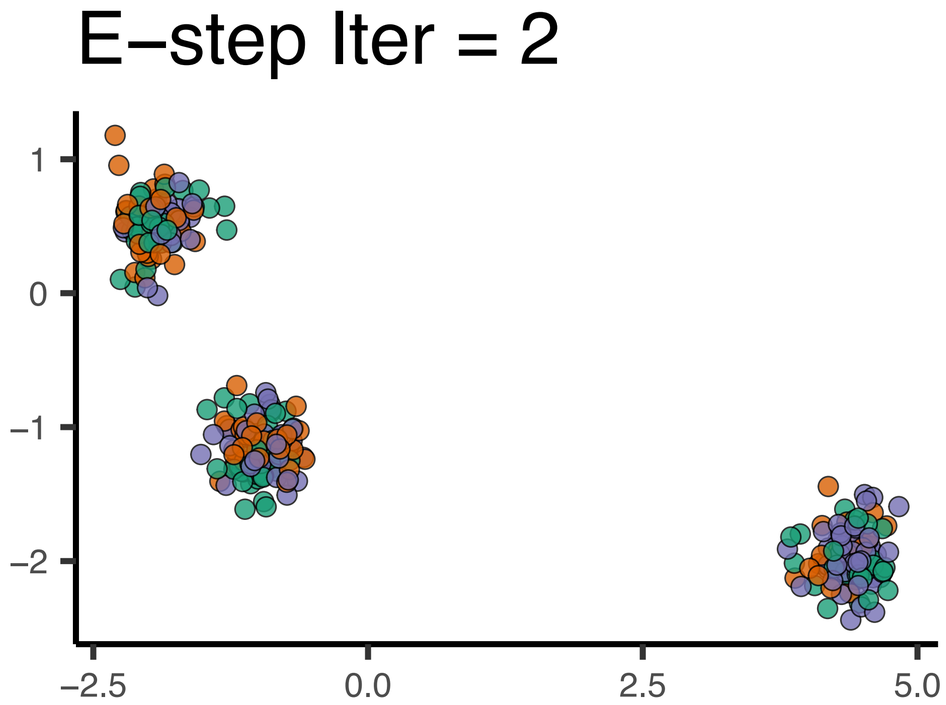
Show the trace of the EM algorithm

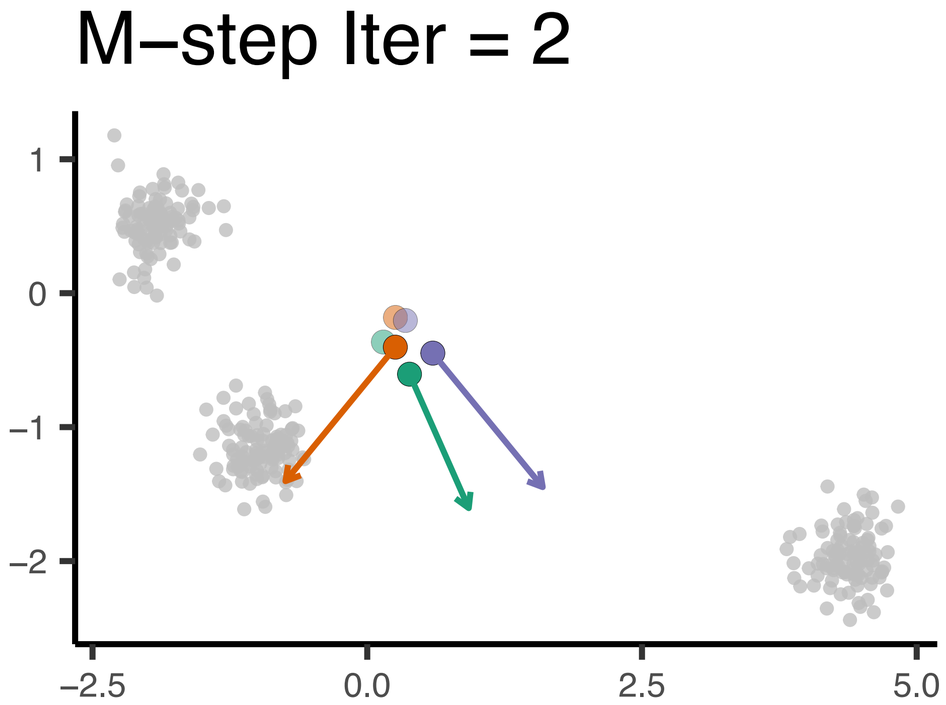




- Arrows: stochastic gradient $\nabla\mu$
- Colour: latent membership

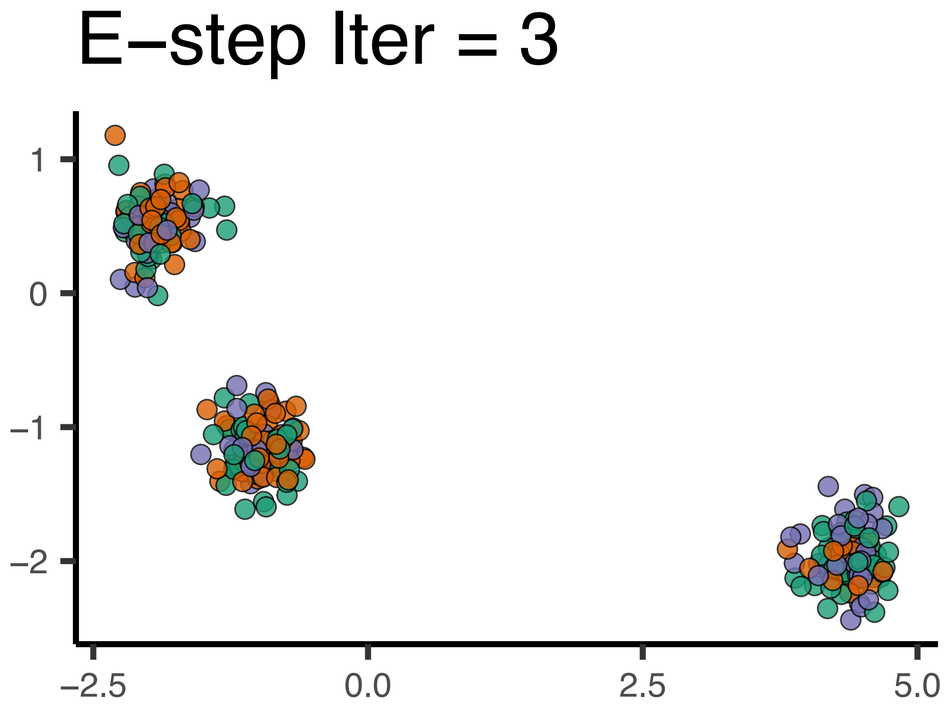
Show the trace of the EM algorithm

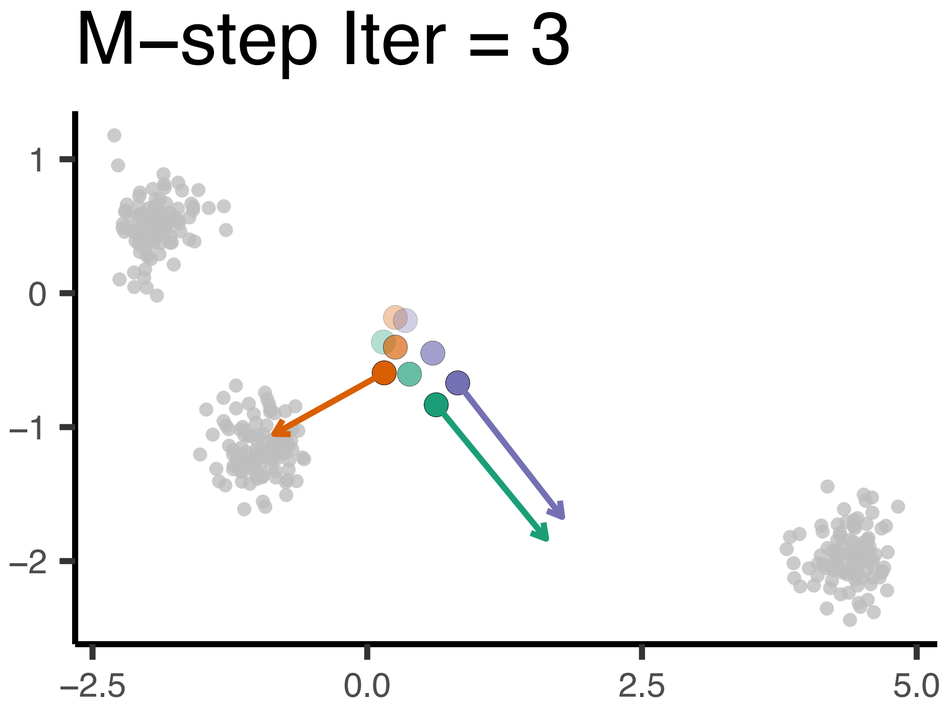




- Arrows: stochastic gradient $\nabla\mu$
- Colour: latent membership

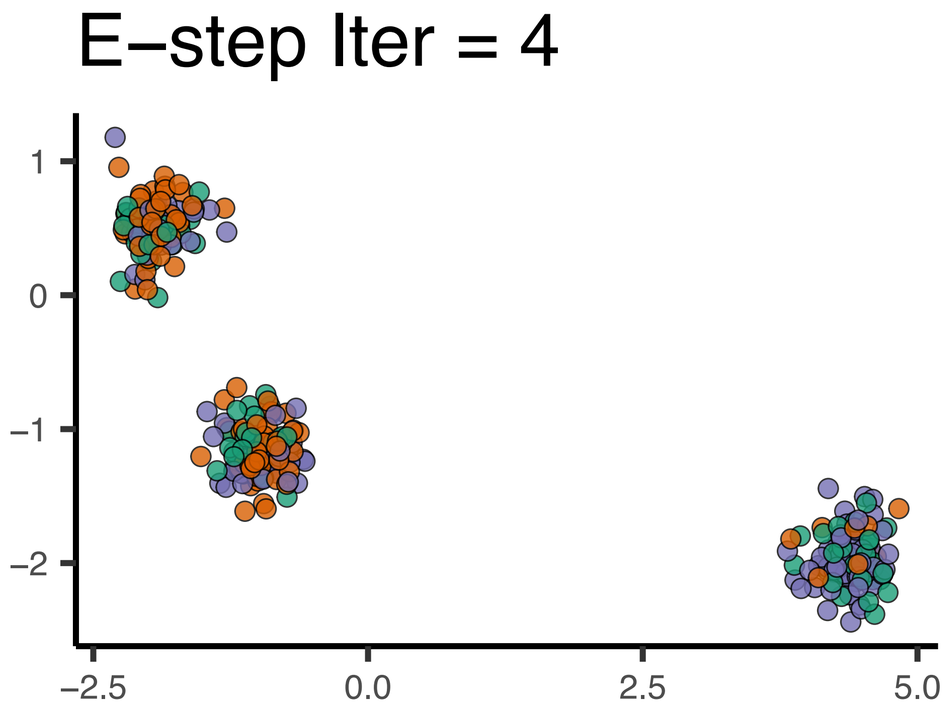
Show the trace of the EM algorithm

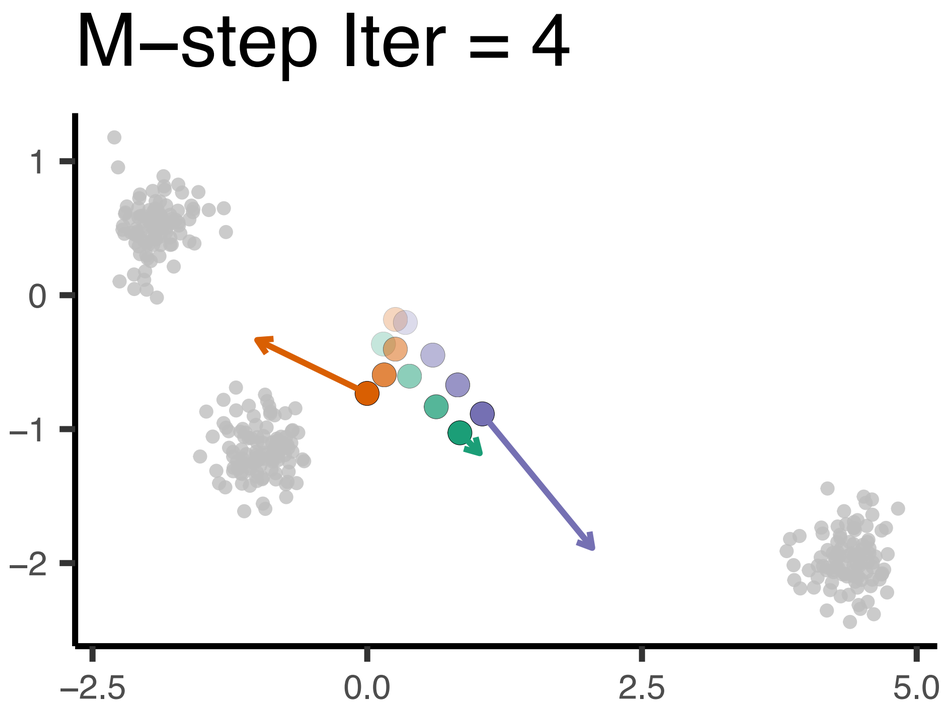




- Arrows: stochastic gradient $\nabla\mu$
- Colour: latent membership

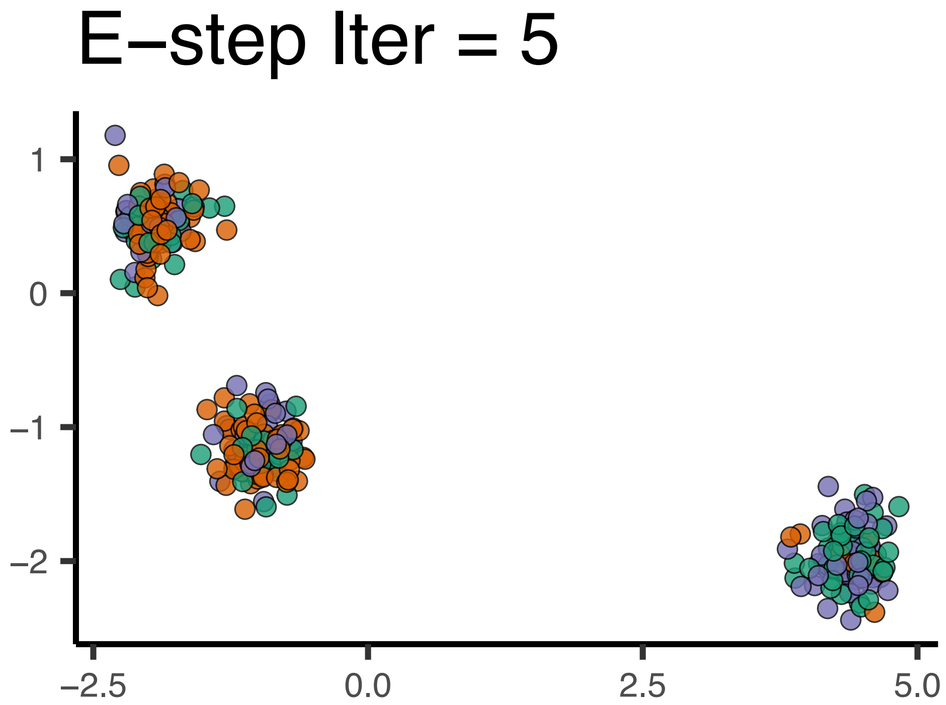
Show the trace of the EM algorithm

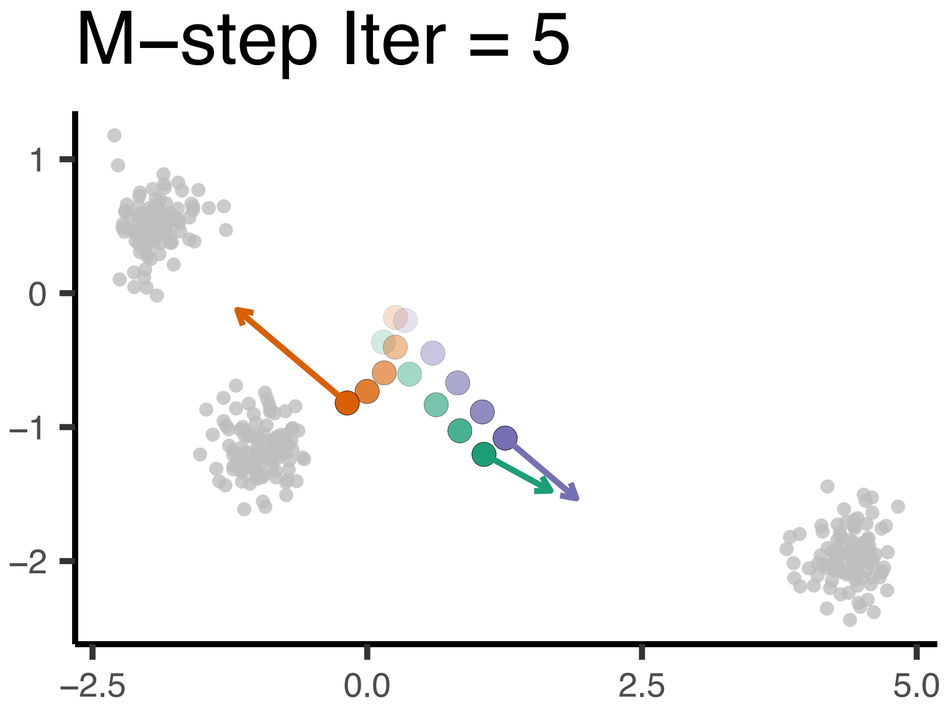




- Arrows: stochastic gradient $\nabla\mu$
- Colour: latent membership

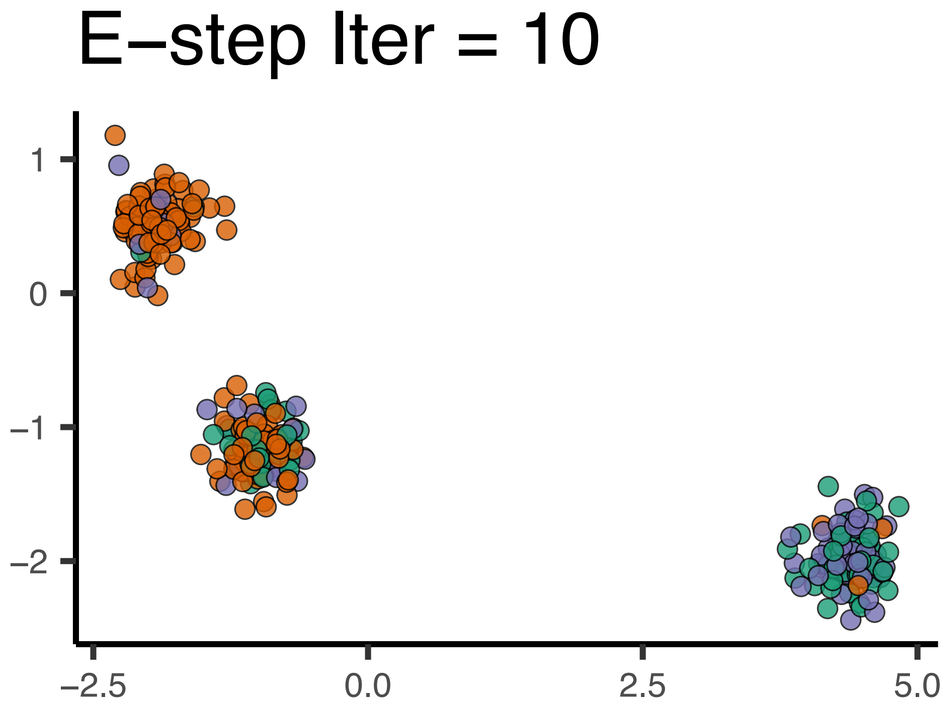
Show the trace of the EM algorithm

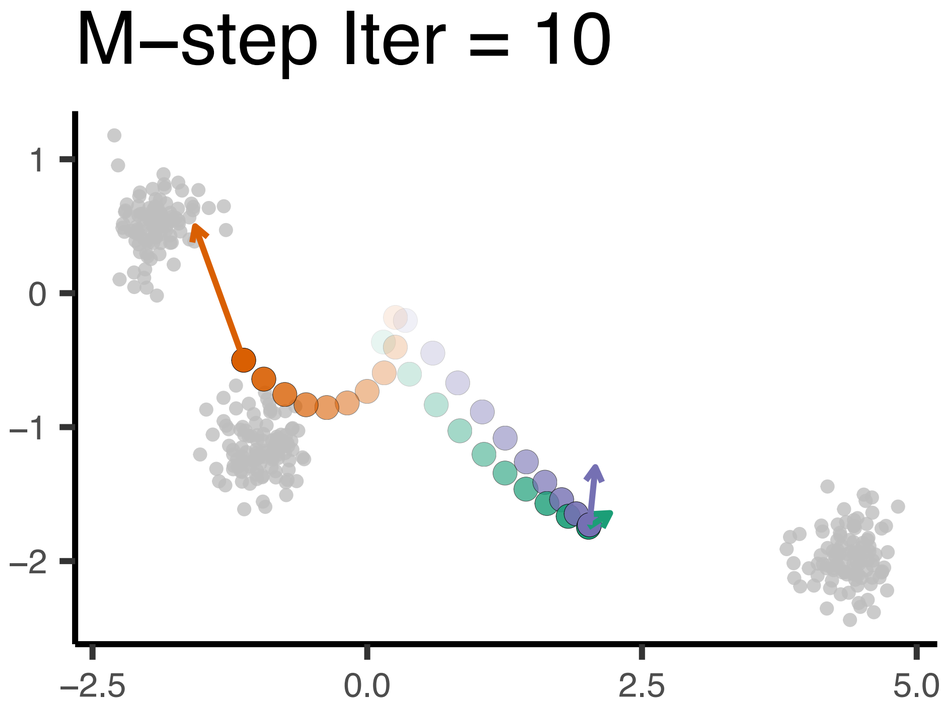




- Arrows: stochastic gradient $\nabla\mu$
- Colour: latent membership

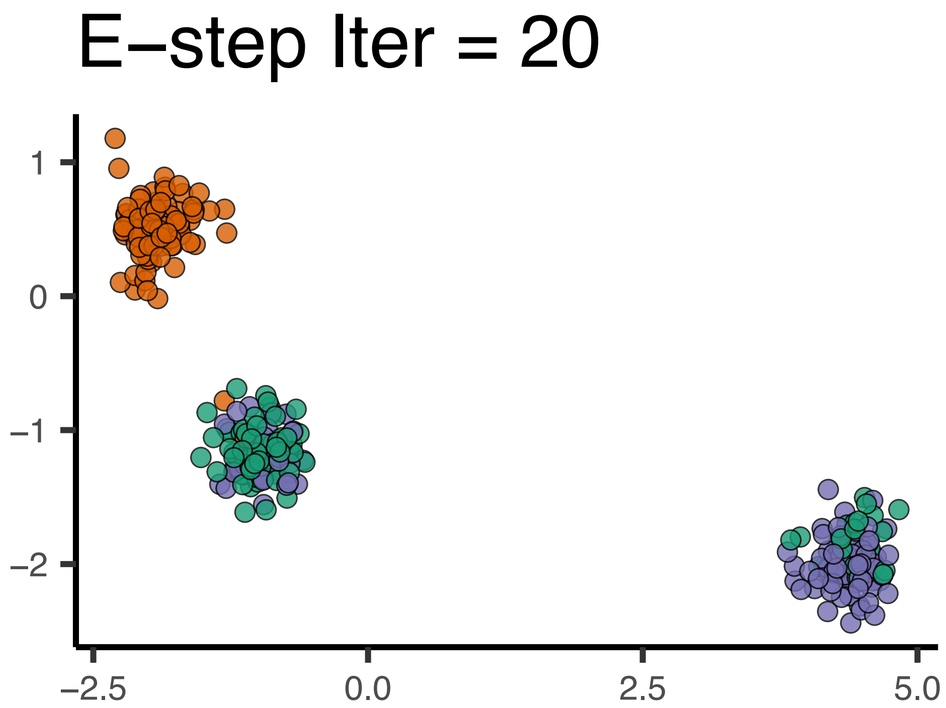
Show the trace of the EM algorithm

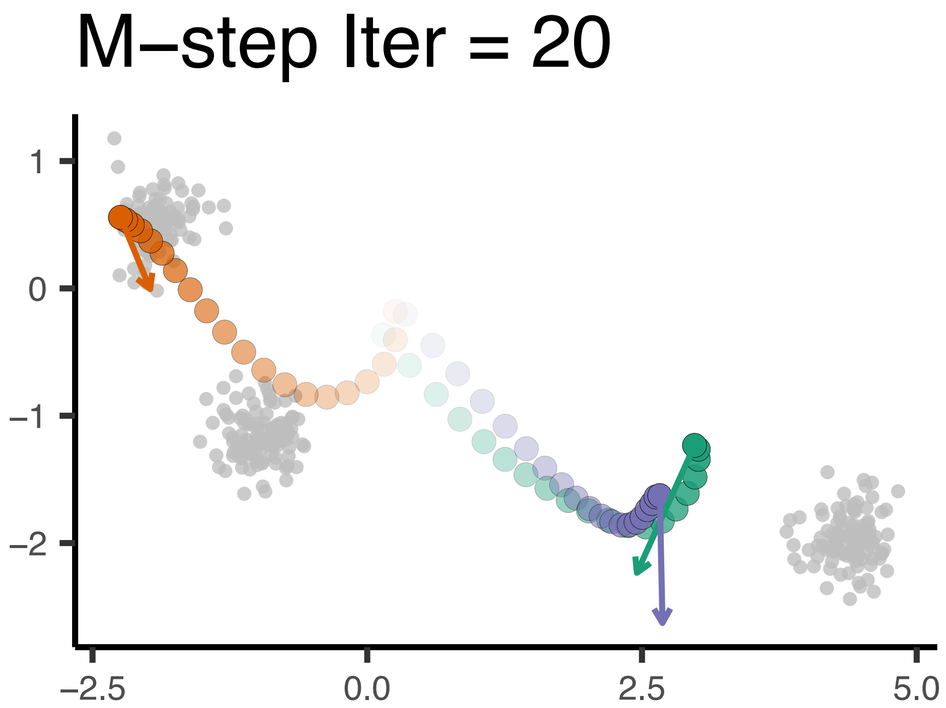




- Arrows: stochastic gradient $\nabla\mu$
- Colour: latent membership

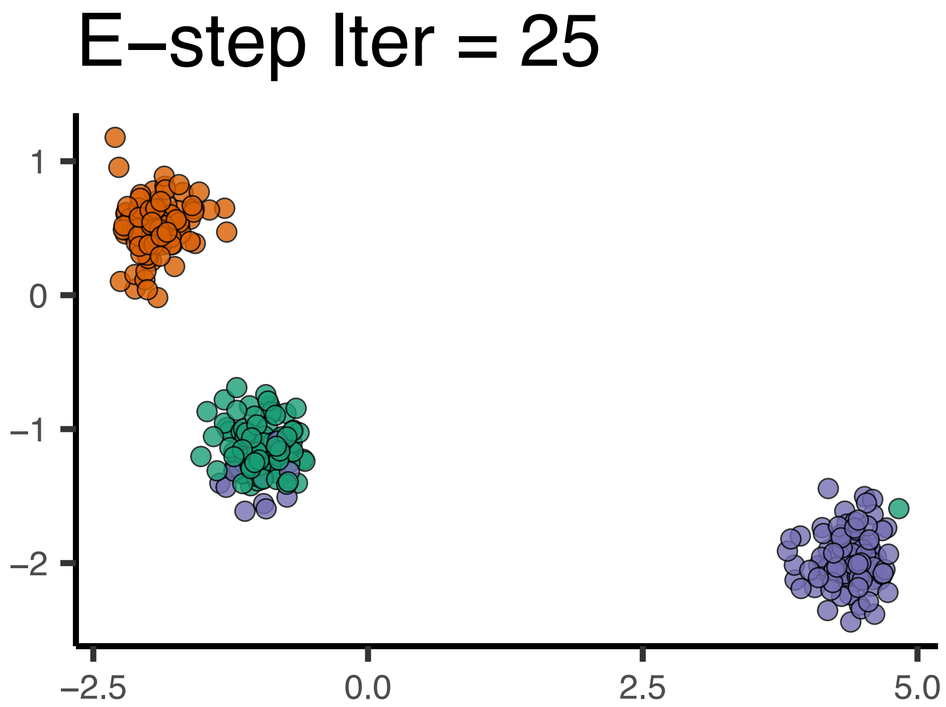
Show the trace of the EM algorithm

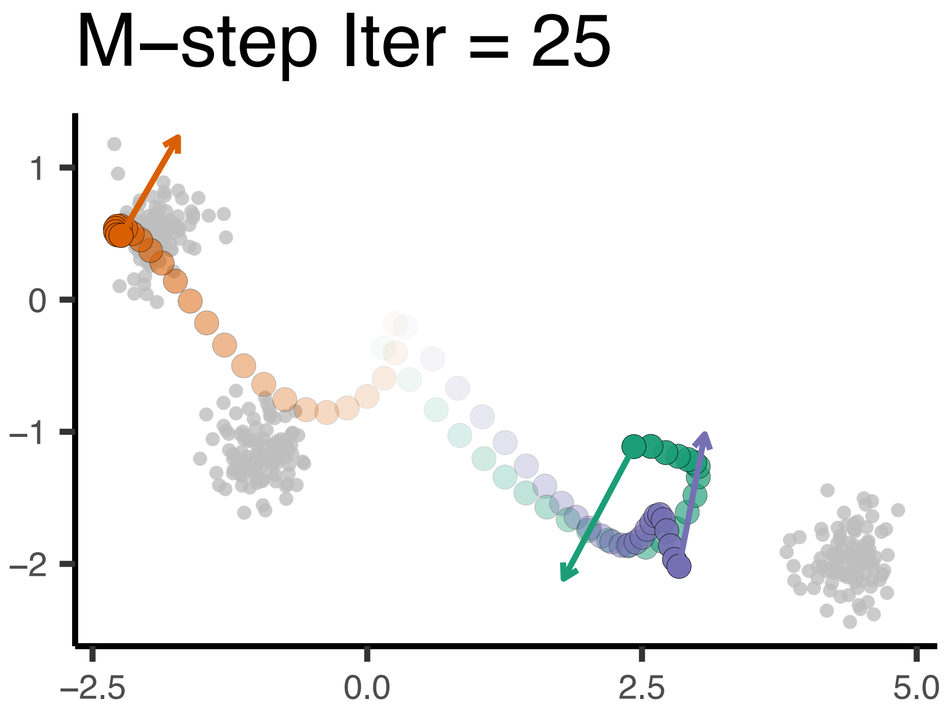




- Arrows: stochastic gradient $\nabla\mu$
- Colour: latent membership

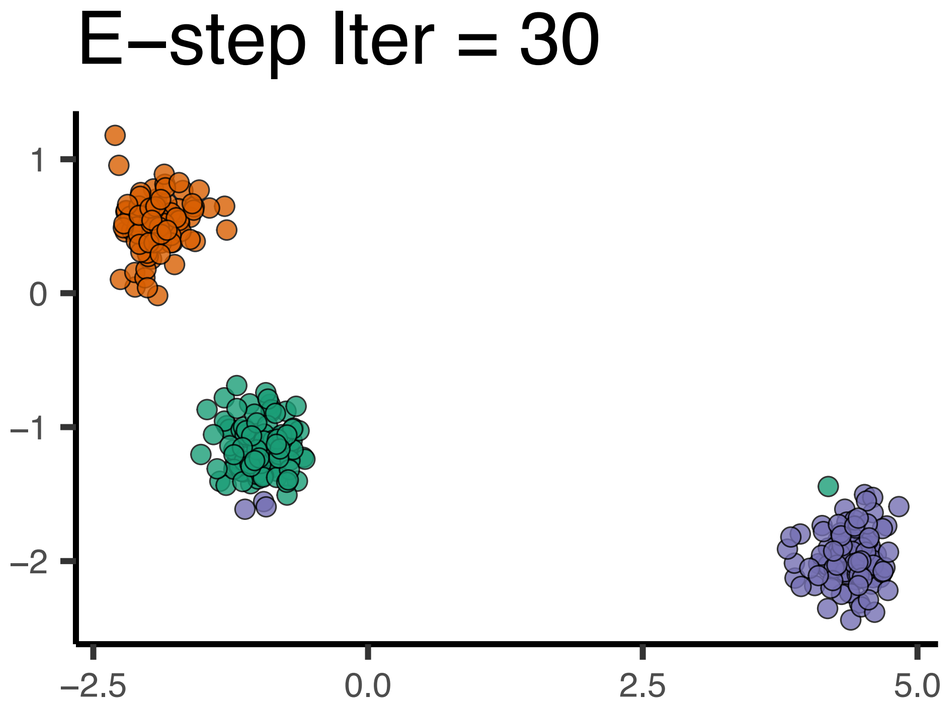
Show the trace of the EM algorithm

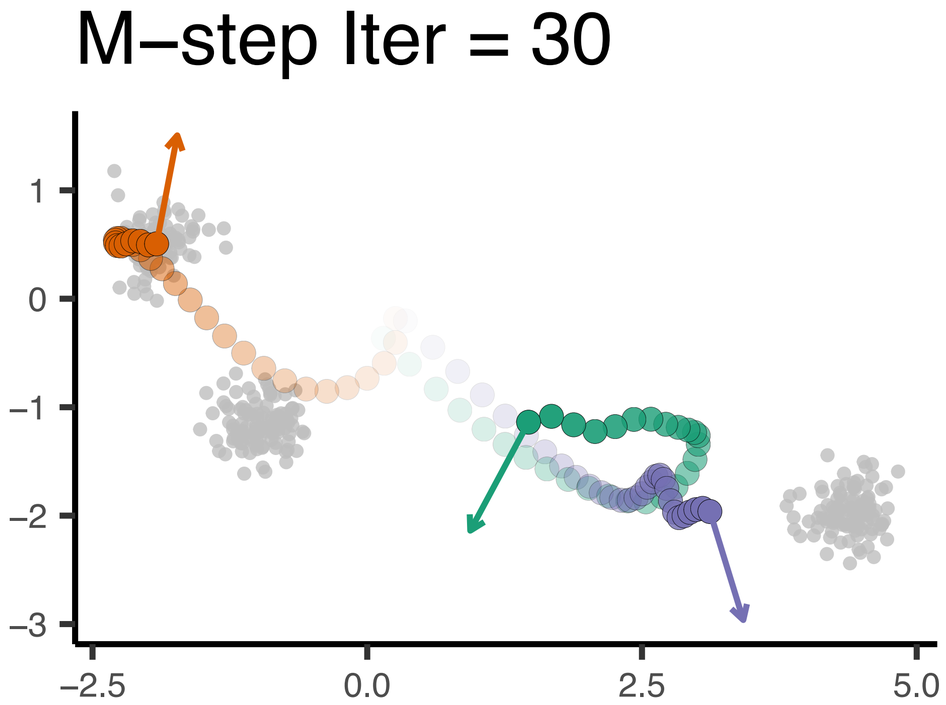




- Arrows: stochastic gradient $\nabla\mu$
- Colour: latent membership

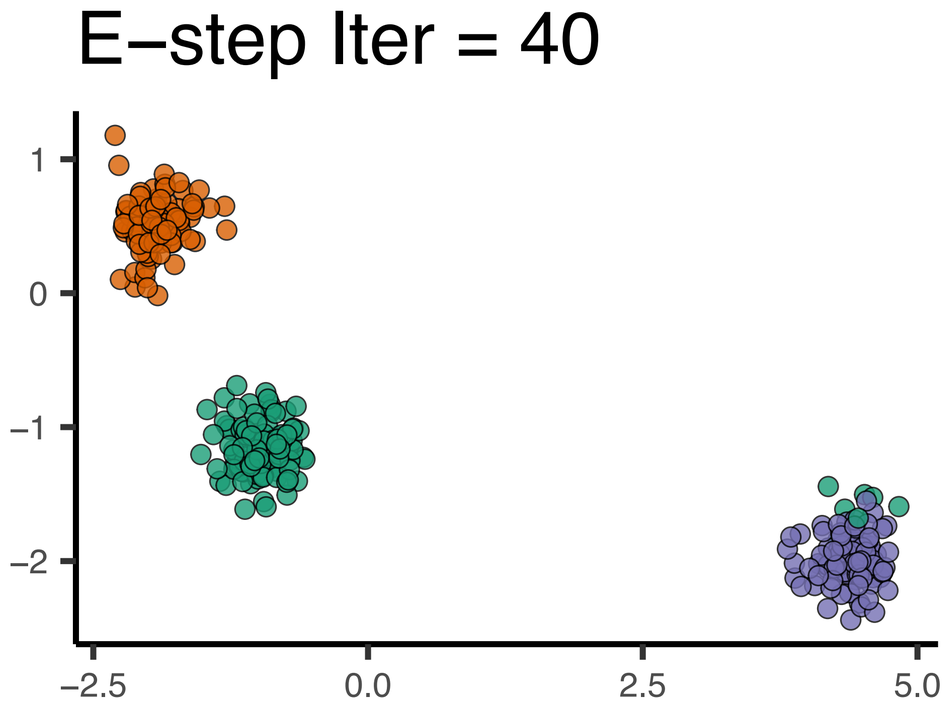
Show the trace of the EM algorithm

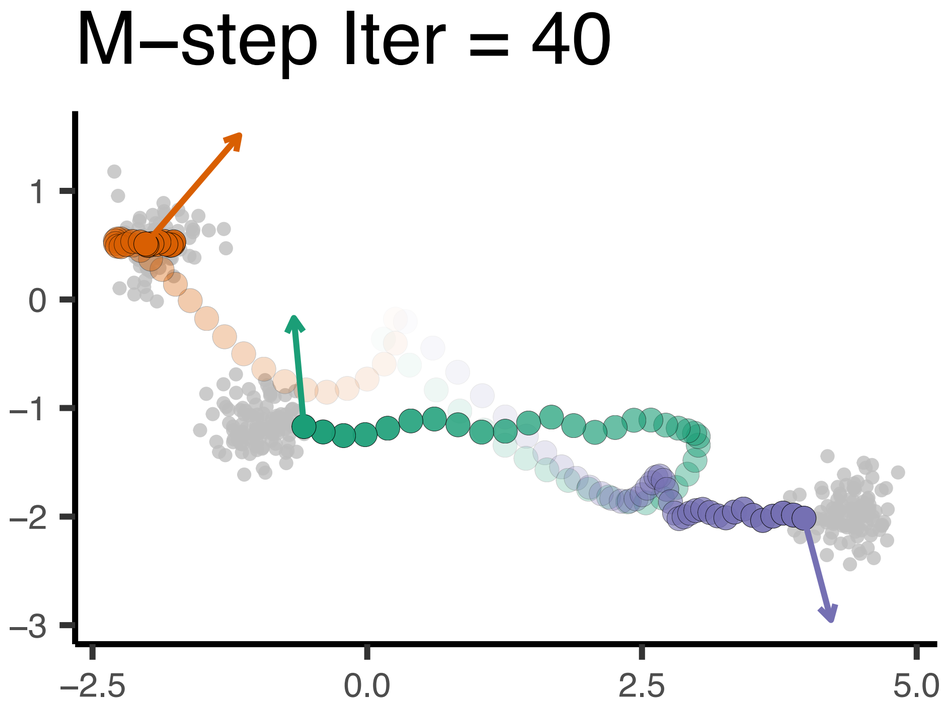




- Arrows: stochastic gradient $\nabla\mu$
- Colour: latent membership

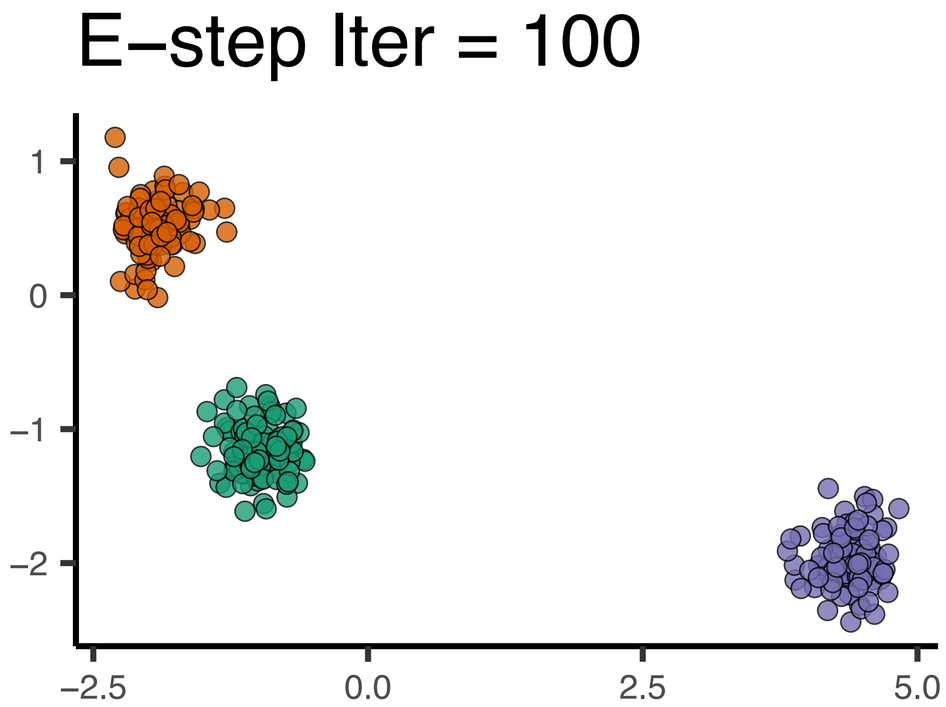
Show the trace of the EM algorithm

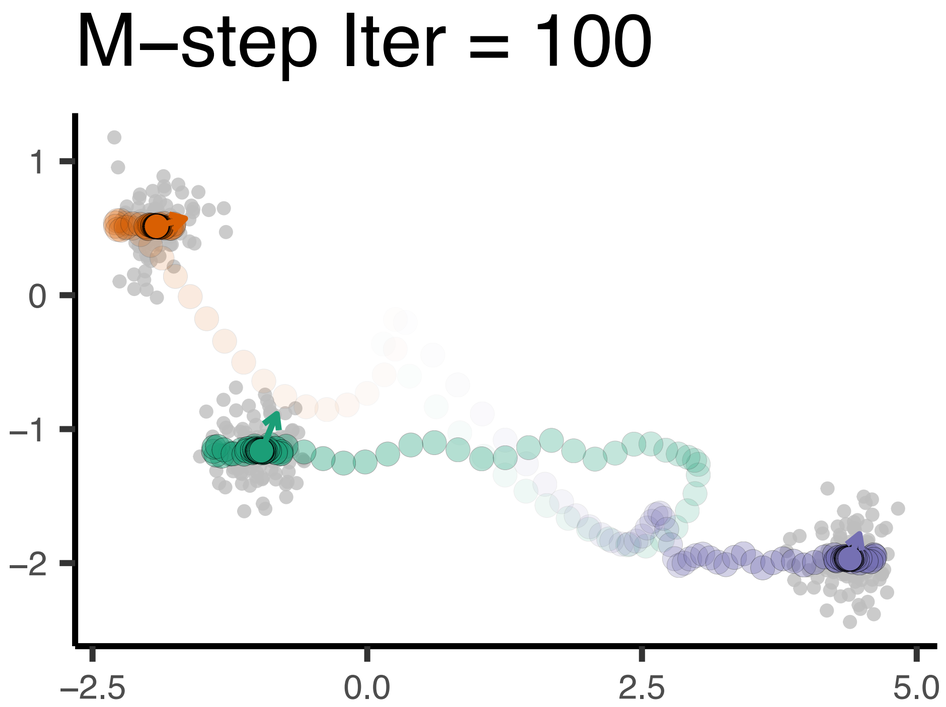




- Arrows: stochastic gradient $\nabla\mu$
- Colour: latent membership

Show the trace of the EM algorithm





- Arrows: stochastic gradient $\nabla \mu$
- Colour: latent membership

Today's lecture: Supervised Learning in Genomics

- **Non-parametric prediction methods**
  - When we don't have any idea of data-generation mechanisms.
  - Kernel (local) regression
  - Gaussian Process
- **Ensemble learning: the collective power of weak learners**
  - Expectation maximization
  - Mixture of linear regressions
- **Variable selection**
  - Challenges in high-dimensional prediction problems
  - Sparse regression models

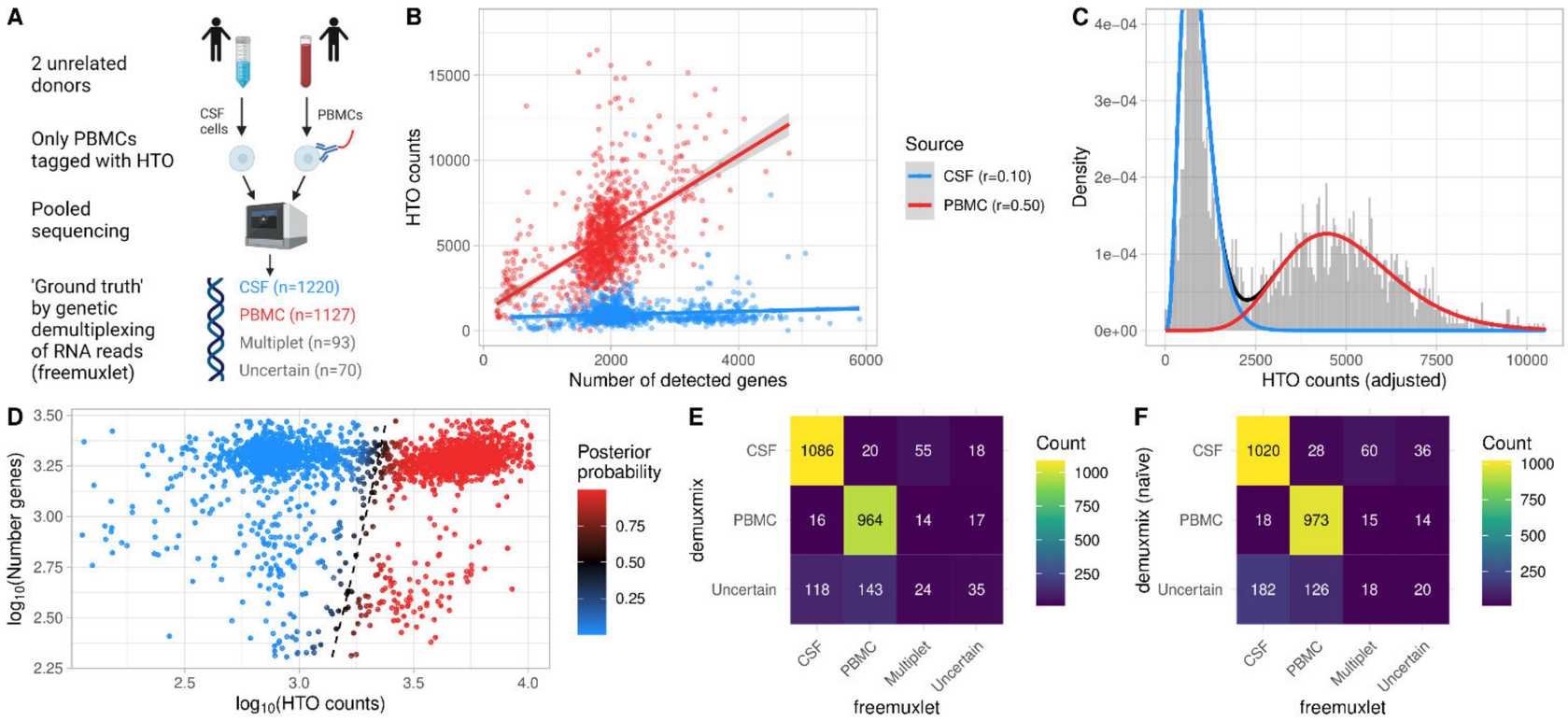
Mixture-of-regression models are useful in demultiplexing multiple input-output patterns

demuxmix: Demultiplexing oligonucleotide-barcoded single-cell RNA sequencing data with regression mixture models

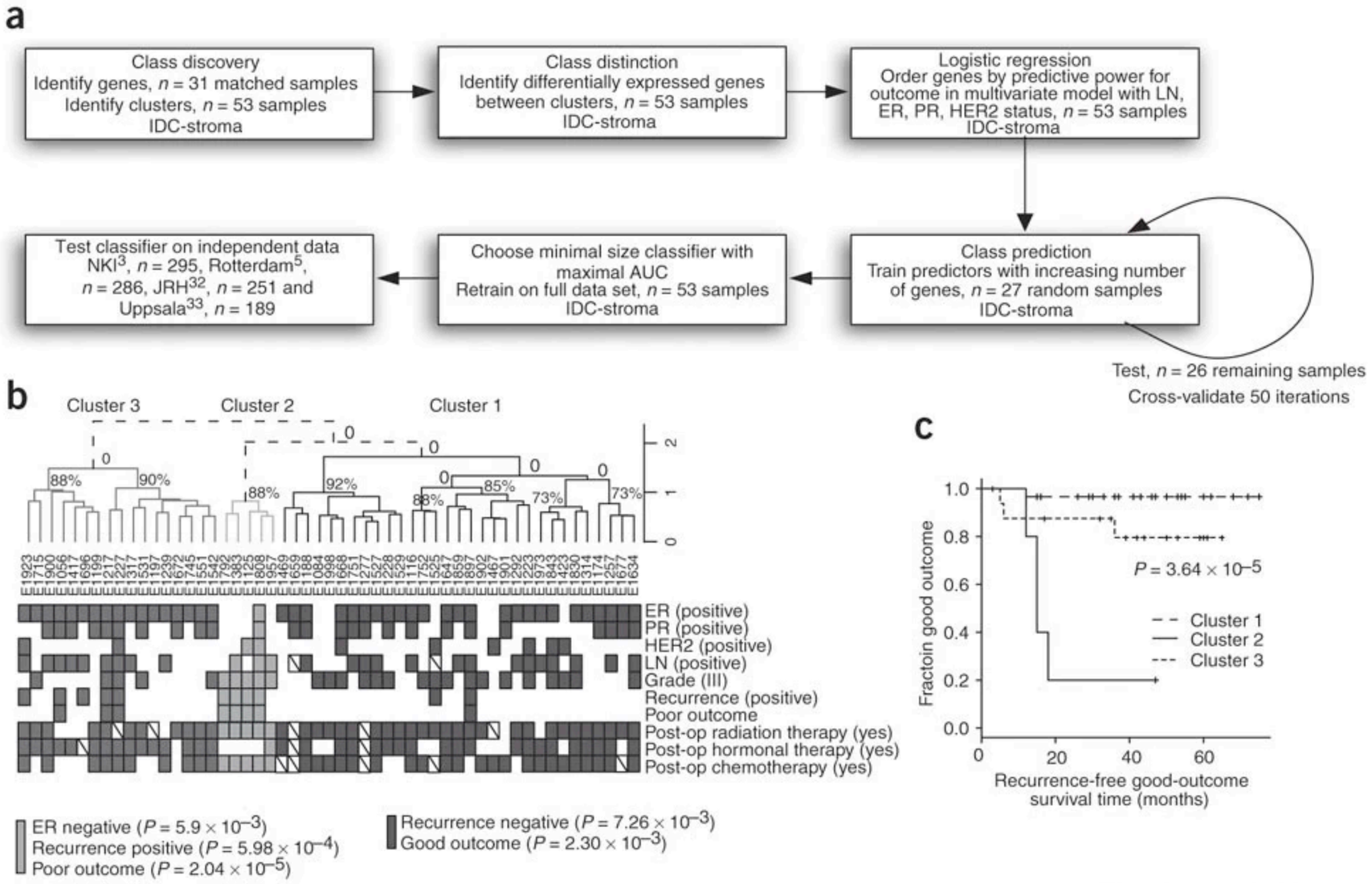
Hans-Ulrich Klein[1,2,*]

[1]Center for Translational & Computational Neuroimmunology, Department of Neurology, Columbia University Irving Medical Center, New York, NY, USA
[2]Taub Institute for Research on Alzheimer's Disease and the Aging Brain, Columbia University Irving Medical Center, New York, NY, USA

*Correspondence: hk2948@cumc.columbia.edu



Mixture models are adaptive to multiple conditions



Finak *et al. Nature Med.* (2008)

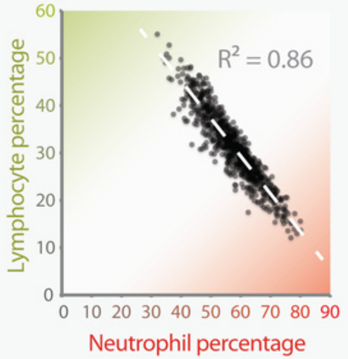
Mixture models naturally emerge in cell-type-specific eQTL prediction problems

I Identify probes that strongly correlate with neutrophil granulocyte percentage in whole peripheral blood: 58 probes in EGCUT (each R > 0.57). Build classifier on 58 probes (R = 0.74 in EGCUT, n = 826)

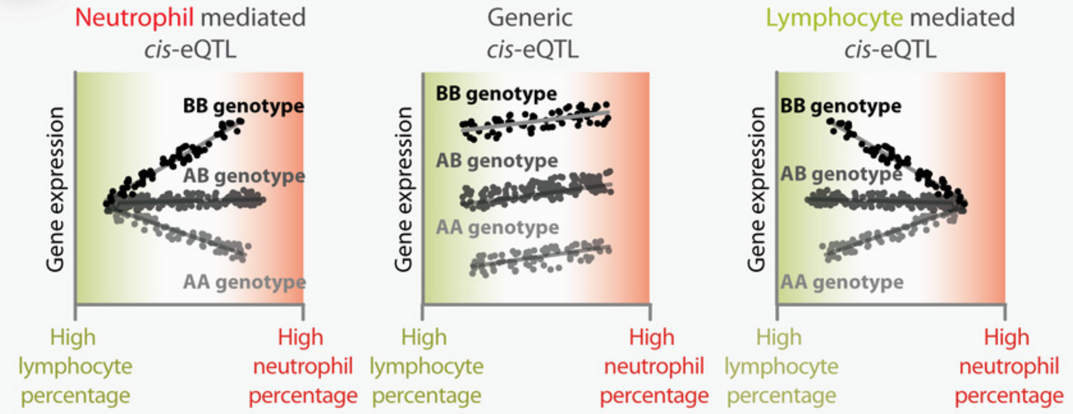
Replicate classifier based on these 58 probes in independent cohort (SHIP-TREND; n = 962, R = 0.82)

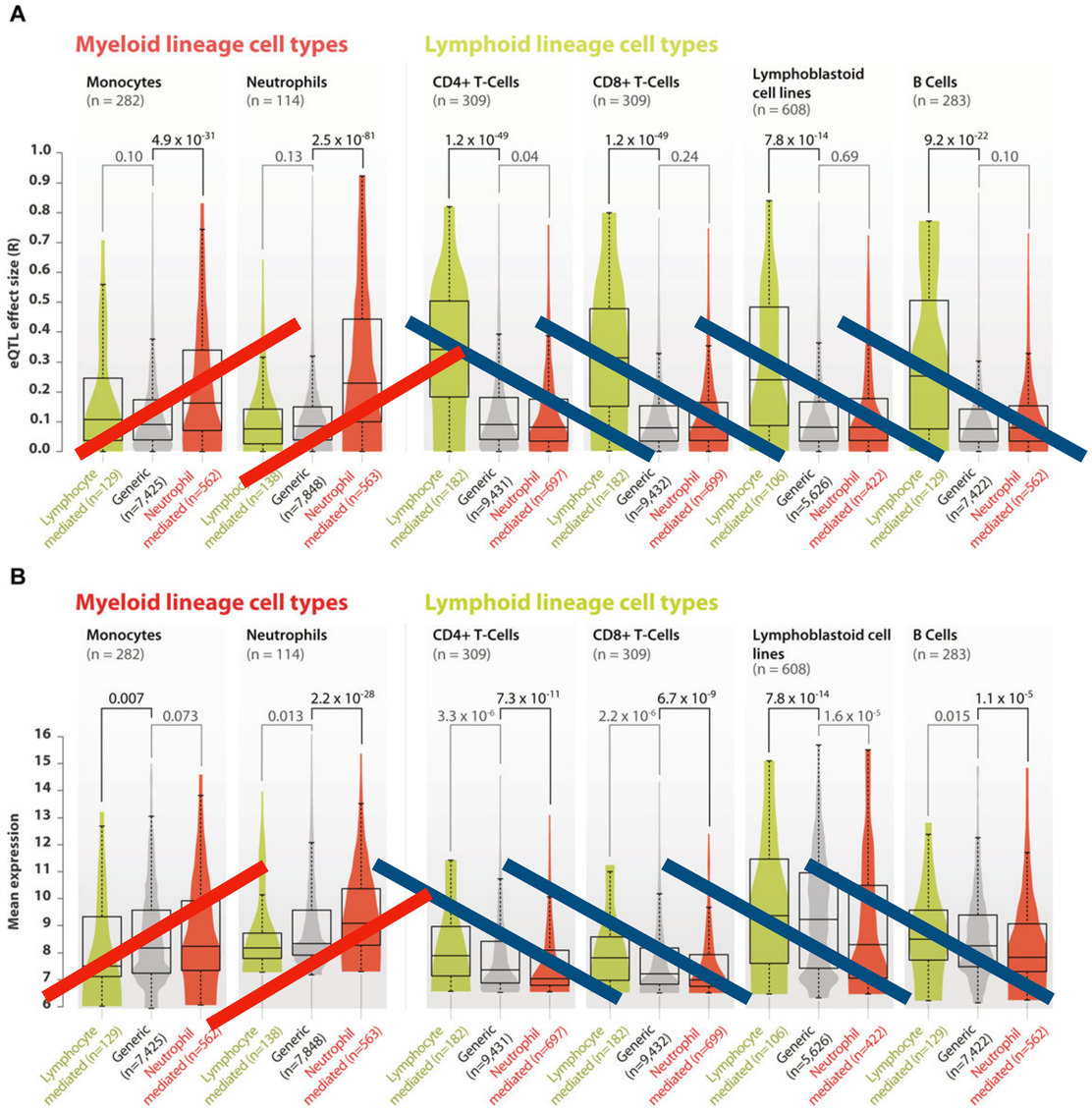
II Apply classifier to other cohorts with no neutrophil granulocyte percentage information available

Neutrophil percentage is negatively correlated with lymphocyte percentage (EGCUT cohort shown)



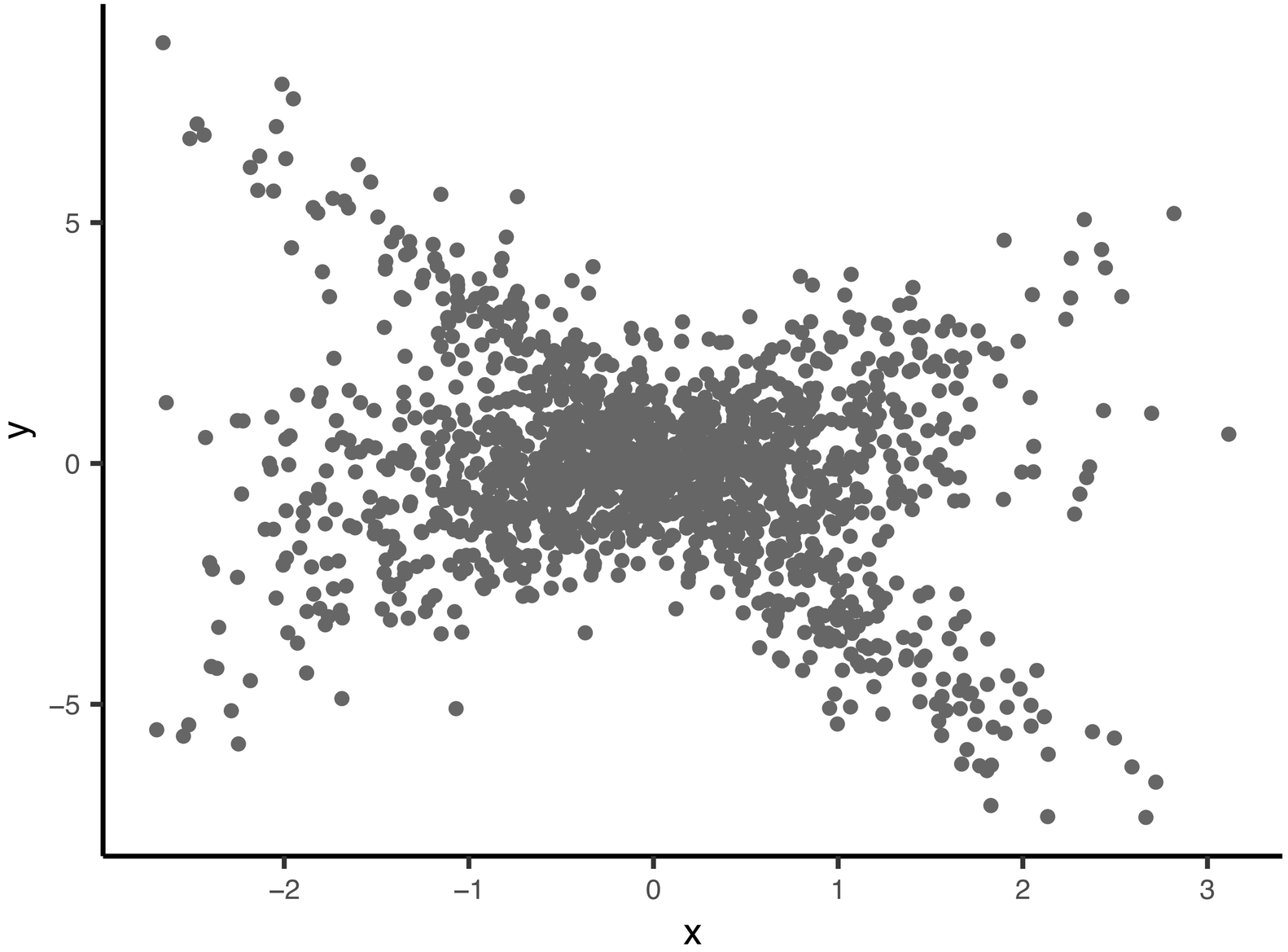
III Map cis-eQTLs with a G x E model with interaction term, using the neutrophil proxy as an environmental factor.



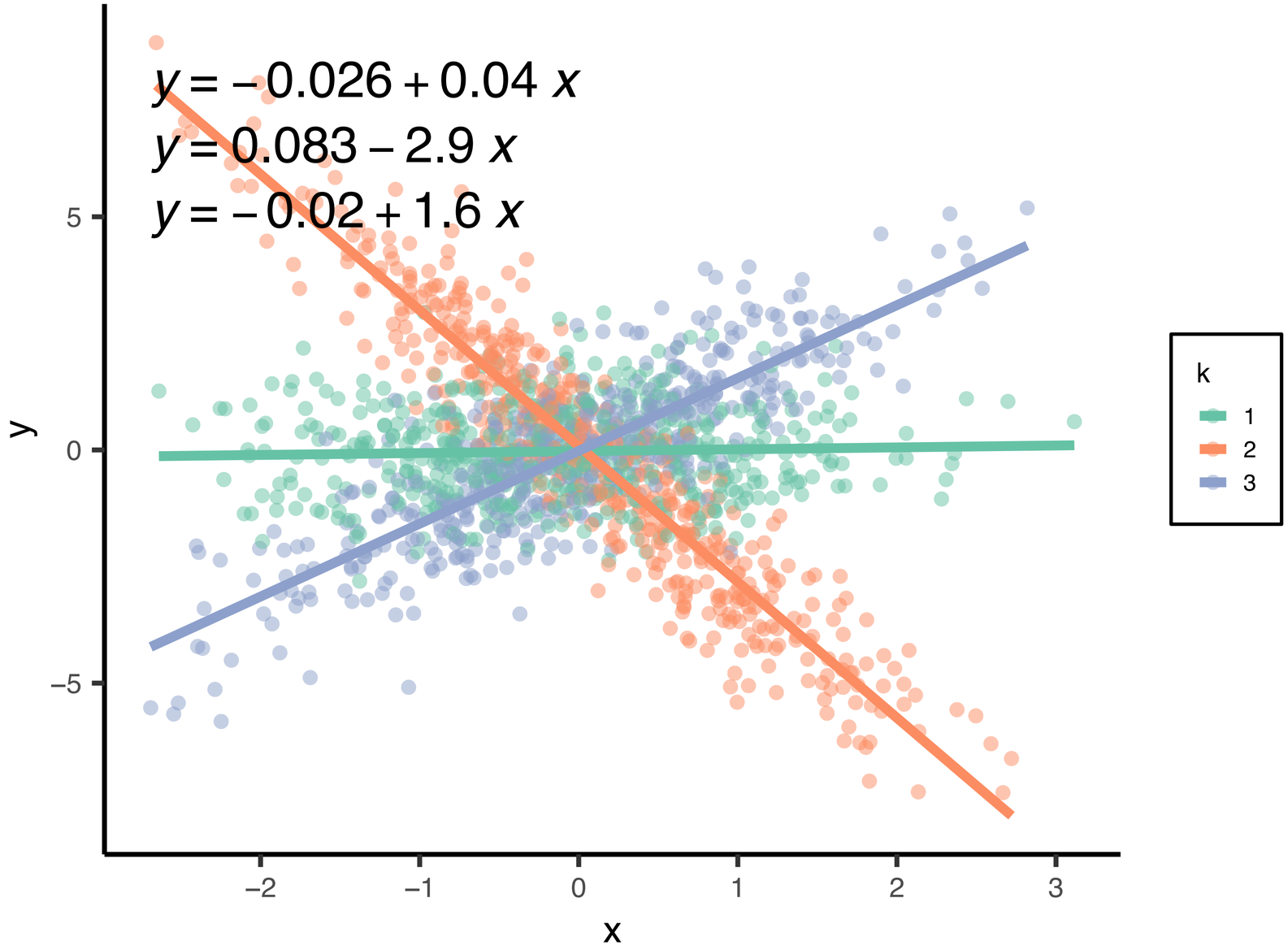


Westra *et al. PLoS Genetics* (2015)

Mixture of linear regression models



Modelling a mixture of regression models



What is the data generation process?

1. Initialize β_k (the slope of each model) and σ_k (the spread within each model)
2. Randomly assign group membership, $Z_{ik} = 1$ iff a point i belongs to a group k.
3. Generate: $Y_i | X_i, Z_{ik} = 1, \mu_k \sim \mathcal{N}(X_i \beta_k, \sigma_k^2)$

What is the expected log-likelihood?

Expected log-likelihood (to maximize):

$$\mathcal{L} = \sum_{i=1}^{n} \sum_{k=1}^{K} p(Z_{ik}|Y_i, X_i, \beta_k, \sigma_k) \log p(Y_i|X_i, \beta_k, \sigma_k)$$

where

$$\log p(Y_i|X_i, \beta_k, \sigma_k) = \log \mathcal{N}(Y_i|X_i\beta_k, \sigma_k^2) = -\frac{1}{2\sigma_k^2}(Y_i - X_i\beta_k)^2 - \frac{1}{2}\log \sigma_k^2$$

So, it's equivalent to finding weighted least square estimates (if $\sigma_k = 1$):

$$\min \sum_{k=1}^{K} \sum_{i=1}^{n} \underbrace{\mathbb{E}[Z_{ik}]}_{\text{E-step}} \underbrace{(Y_i - X_i\beta_k)^2}_{\text{M-step}}$$

E-step: What is the posterior probability of assigning each data point?

Given β_k and σ_k:

$$p(Z_{ik}|X_i, Y_i, \beta_k, \sigma_k) = \frac{\exp\left(-\frac{1}{2\sigma_k^2}\left(Y_i - \overbrace{X_i\beta_k}^{\text{prediction}}\right)^2 - \log\sigma_k\right)}{\sum_{k'}\exp\left(-\frac{1}{2\sigma_{k'}^2}\left(Y_i - \underbrace{X_i\beta_{k'}}_{\text{prediction}}\right)^2 - \log\sigma_{k'}\right)}$$

- Intuition: For the observed (X_i, Y_i), we ask, "how far is the predicted $X_i\beta_k$ from the observed Y_i?"
- Sample inversely proportional to the distance from different $X_i\beta_k$
- More rigorously: We need to introduce a Lagrangian multiplier to enforce the "sum to 1" constraint, $\sum_k Z_{ik} = 1$.

E-step: What is the posterior probability of assigning each data point?

Given β_k and σ_k:

$$p(Z_{ik}|X_i, Y_i, \beta_k, \sigma_k) = \frac{\exp\left(-\frac{1}{2\sigma_k^2}\left(Y_i - X_i\beta_k\right)^2 - \log\sigma_k\right)}{\sum_{k'}\exp\left(-\frac{1}{2\sigma_{k'}^2}\left(Y_i - X_i\beta_{k'}\right)^2 - \log\sigma_{k'}\right)}$$

- Intuition: For the observed (X_i, Y_i), we ask, "how far is the predicted $X_i\beta_k$ from the observed Y_i?"
- Sample inversely proportional to the distance from different $X_i\beta_k$
- More rigorously: We need to introduce a Lagrangian multiplier to enforce the "sum to 1" constraint, $\sum_k Z_{ik} = 1$.

M-step: Maximize regression model parameters

For each regression model k, we can optimize the parameters β_k and σ_k.

For example,

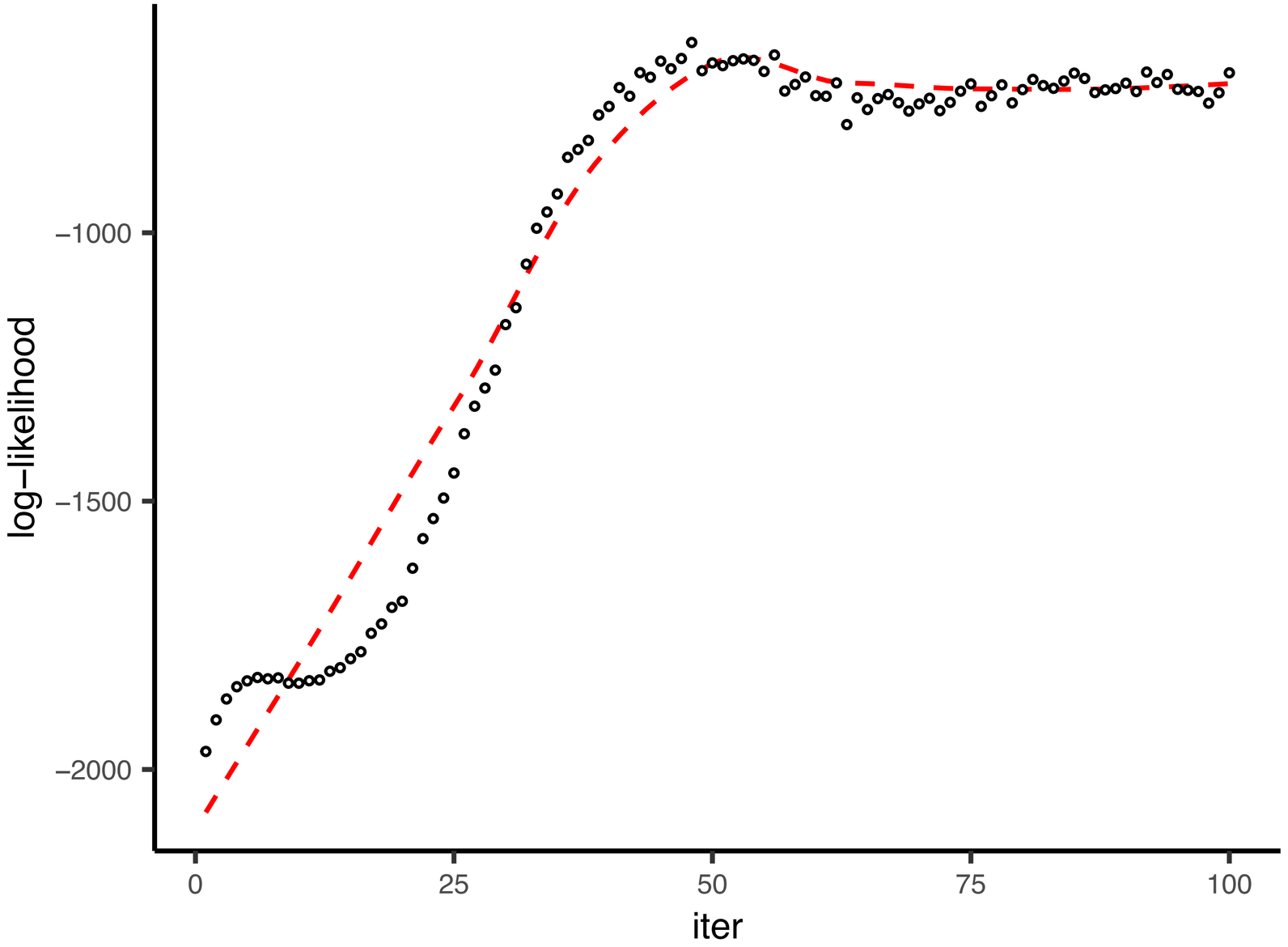
$$\nabla_{\beta_k,\sigma_k}\mathcal{L} = \overbrace{\sum_{i=1}^{n}}^{\text{could be a minibatch}} \underbrace{\mathbb{E}[Z_{ik}]}_{\text{from the E-step}} \underbrace{\nabla_{\beta_k,\sigma_k} \log p(Y_i|X_i,\beta_k,\sigma_k)}_{\text{local gradients}}$$

we can simply take stochastic gradient steps:

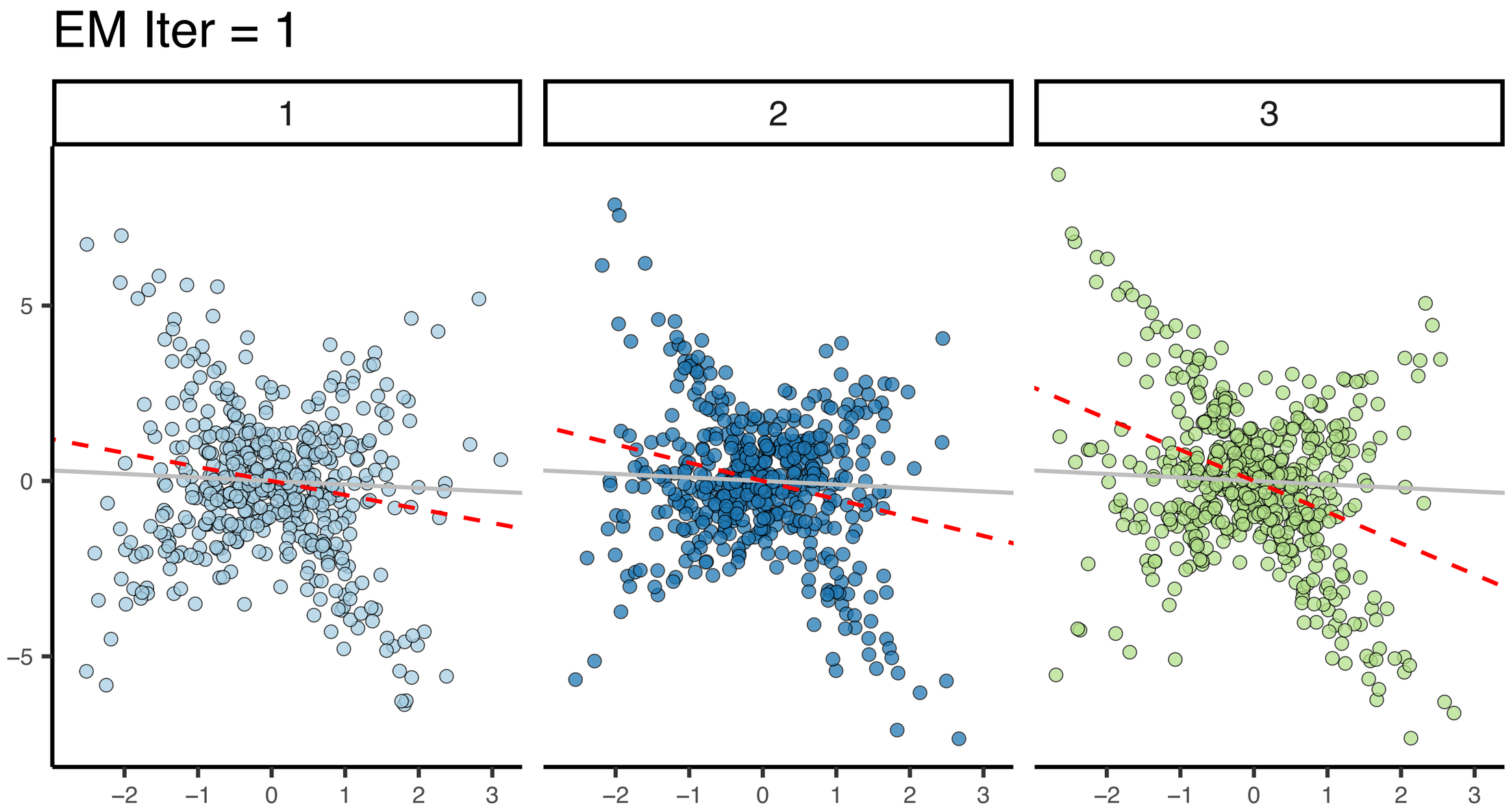
$$\beta_k \leftarrow \beta_k + \rho\nabla_{\beta_k}\mathcal{L}$$

$$\sigma_k \leftarrow \sigma_k + \rho\nabla_{\sigma_k}\mathcal{L}$$

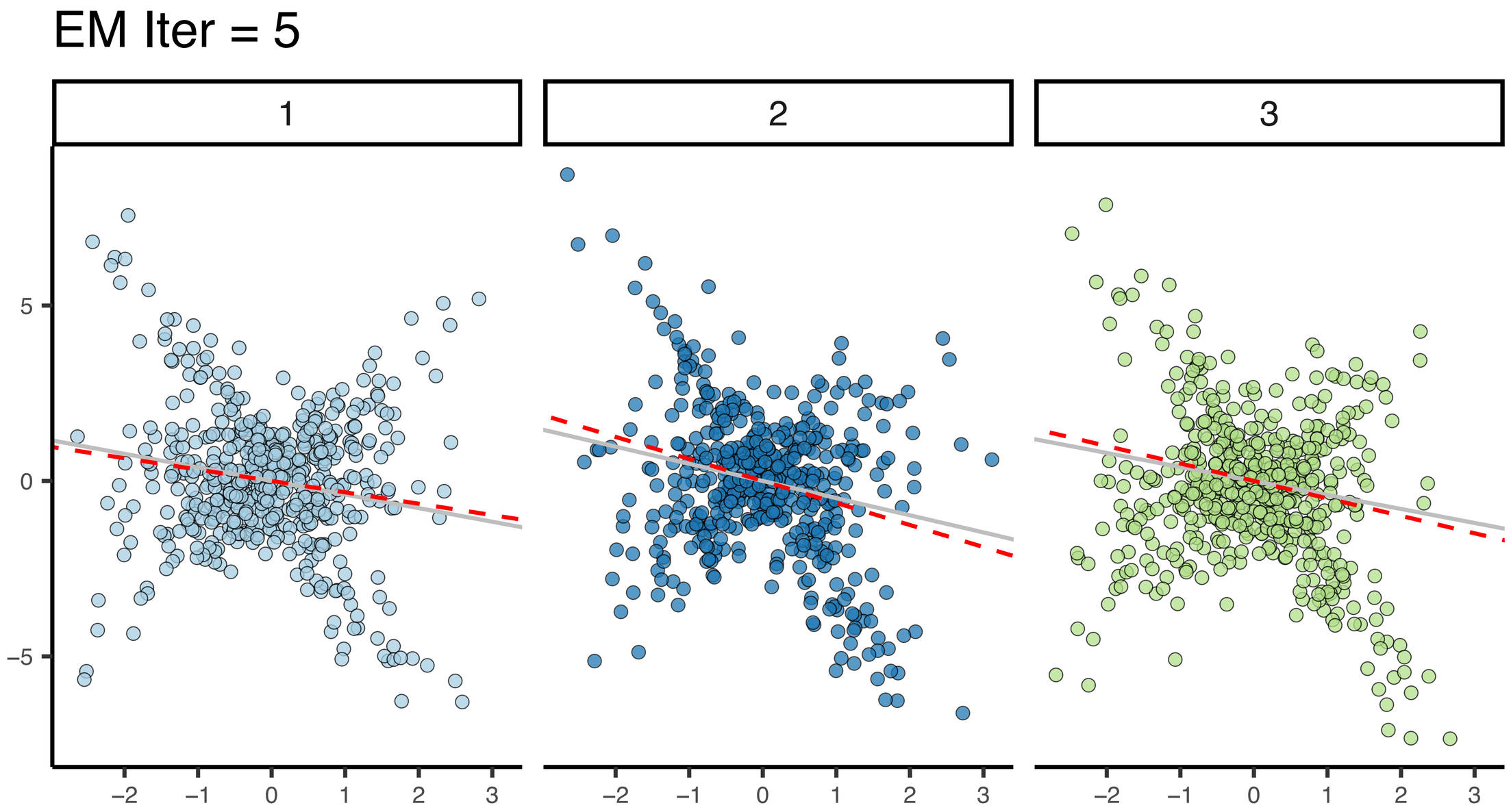
EM algorithm for a mixture of regression models



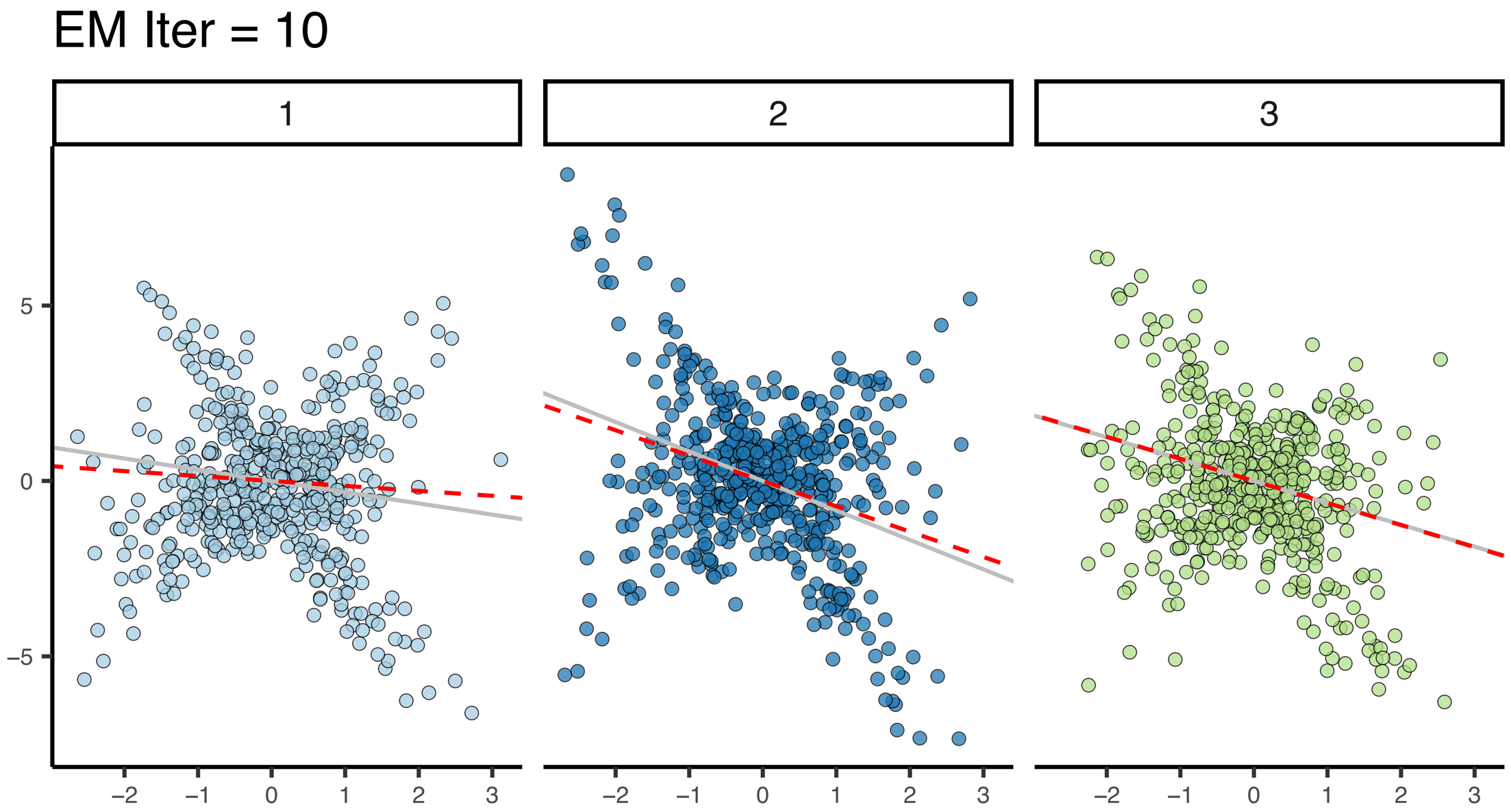
EM algorithm for a mixture of regression models



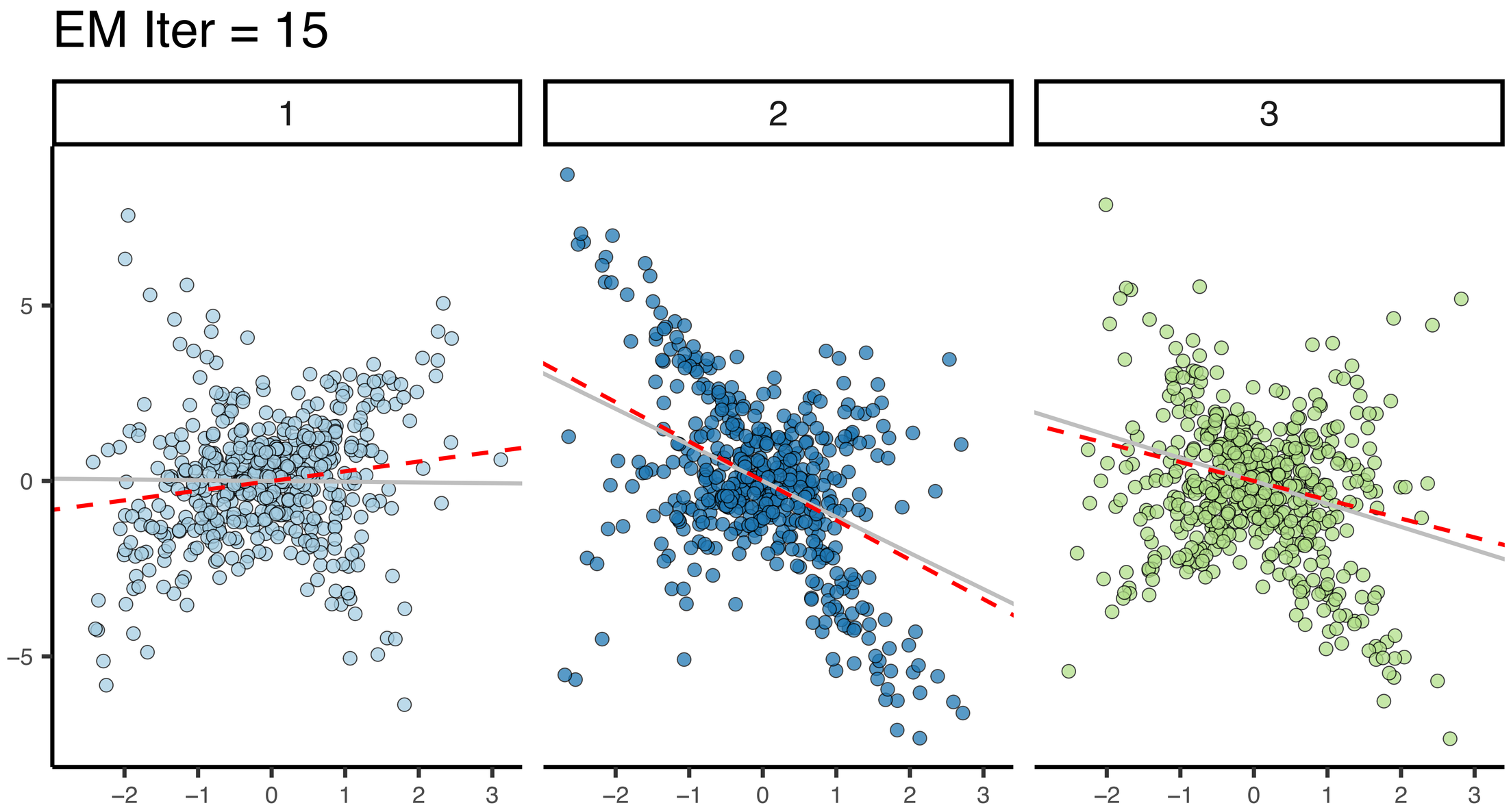
EM algorithm for a mixture of regression models



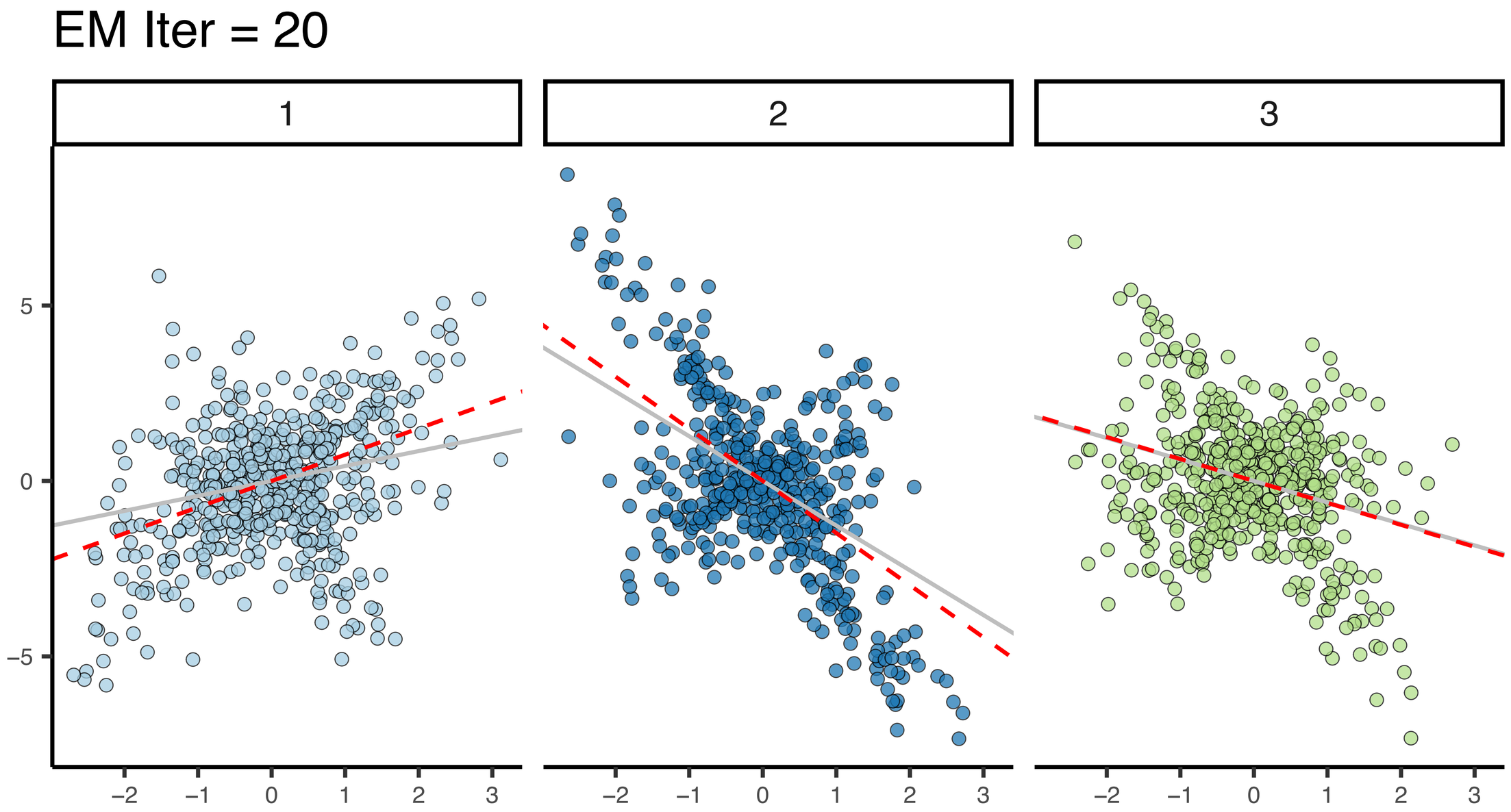
EM algorithm for a mixture of regression models



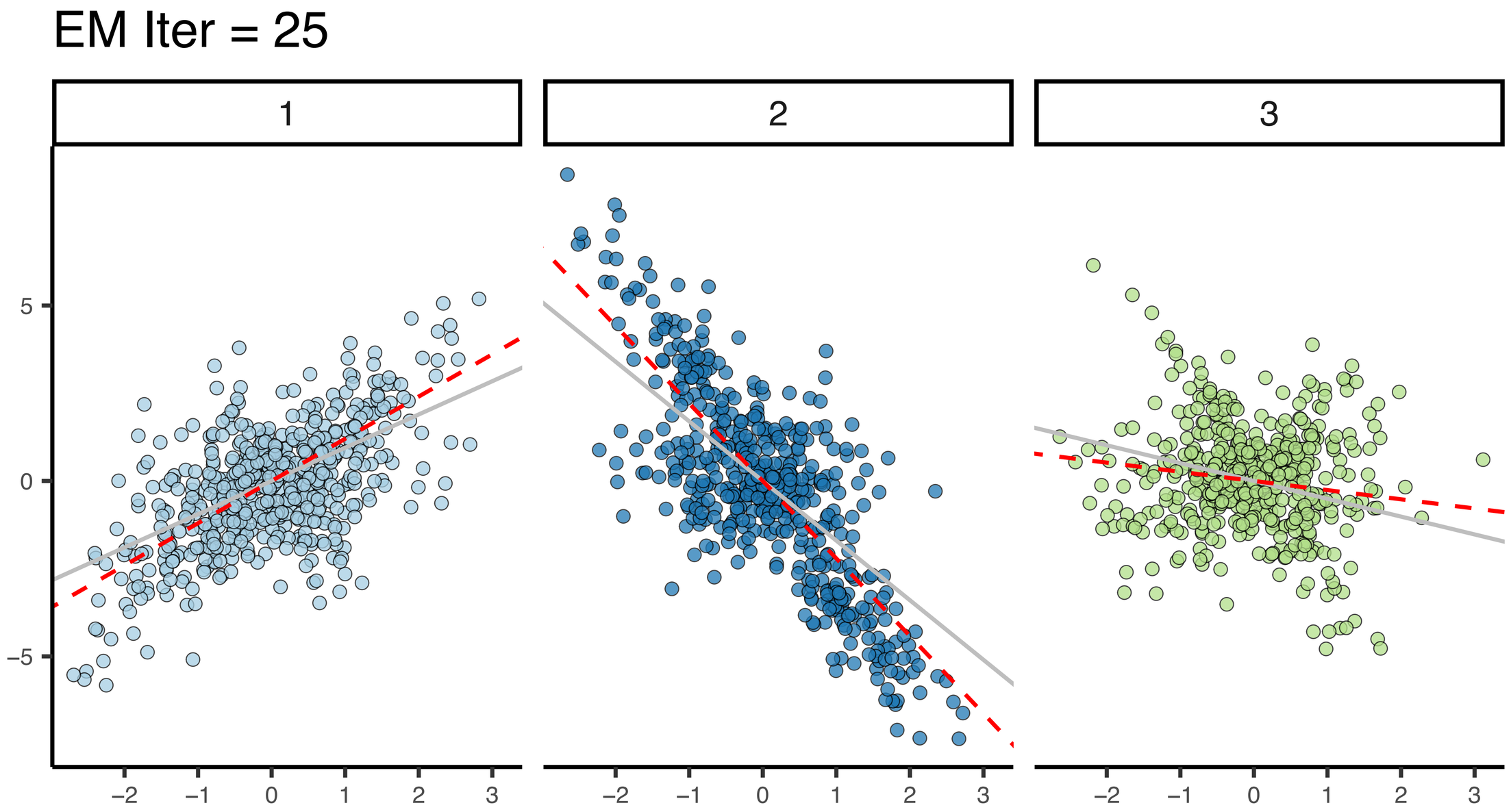
EM algorithm for a mixture of regression models



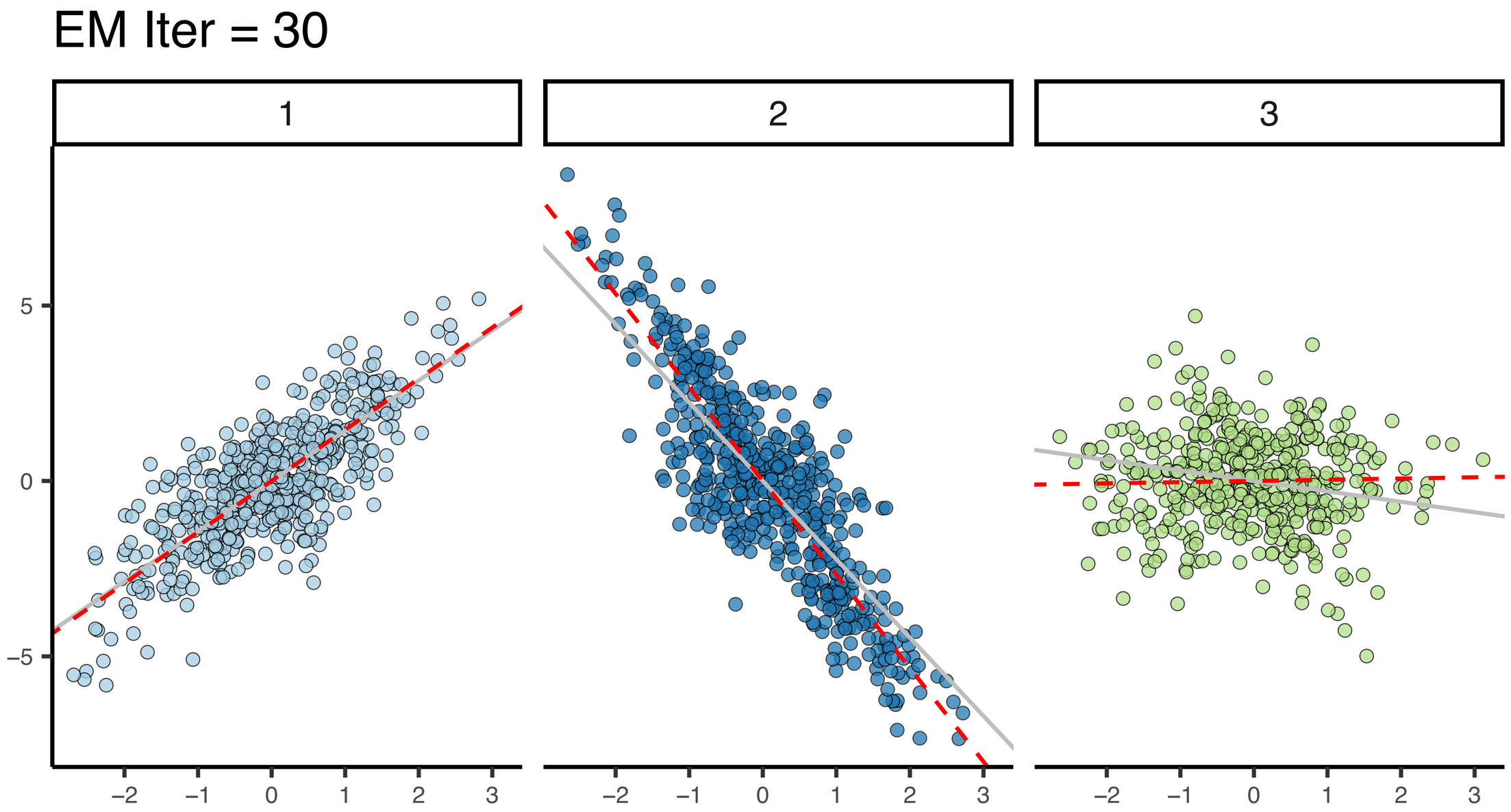
EM algorithm for a mixture of regression models



EM algorithm for a mixture of regression models



EM algorithm for a mixture of regression models



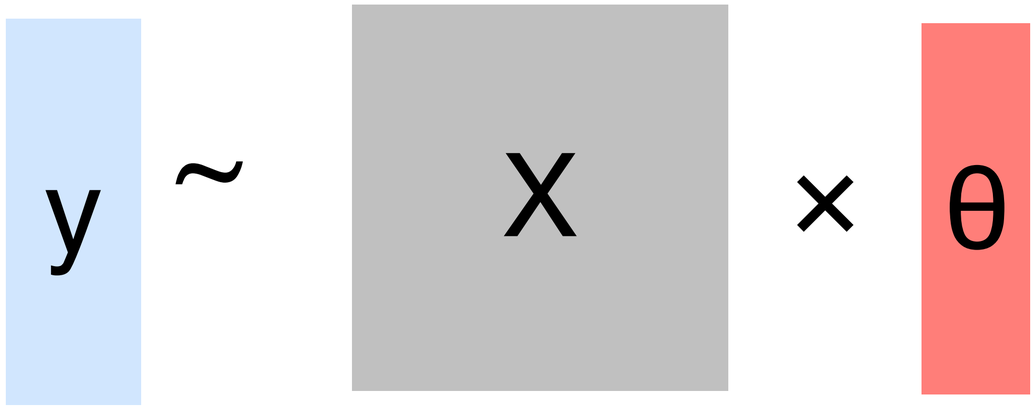
A Mixture of Regression models

- Divide and conquer
- What will be a potential application of this approach?
- Can we apply the same inference algorithm for non-linear regression models?
- What is the benefit of modelling a mixture of regressions models, as opposed to modelling a mixture of densities?

Today's lecture: Supervised Learning in Genomics

- **Non-parametric prediction methods**
  - When we don't have any idea of data-generation mechanisms.
  - Kernel (local) regression
  - Gaussian Process
- **Ensemble learning: the collective power of weak learners**
  - Expectation maximization
  - Mixture of linear regressions
- **Variable selection**
  - Challenges in high-dimensional prediction problems
  - Sparse regression models

Why human genetics problems are so hard?



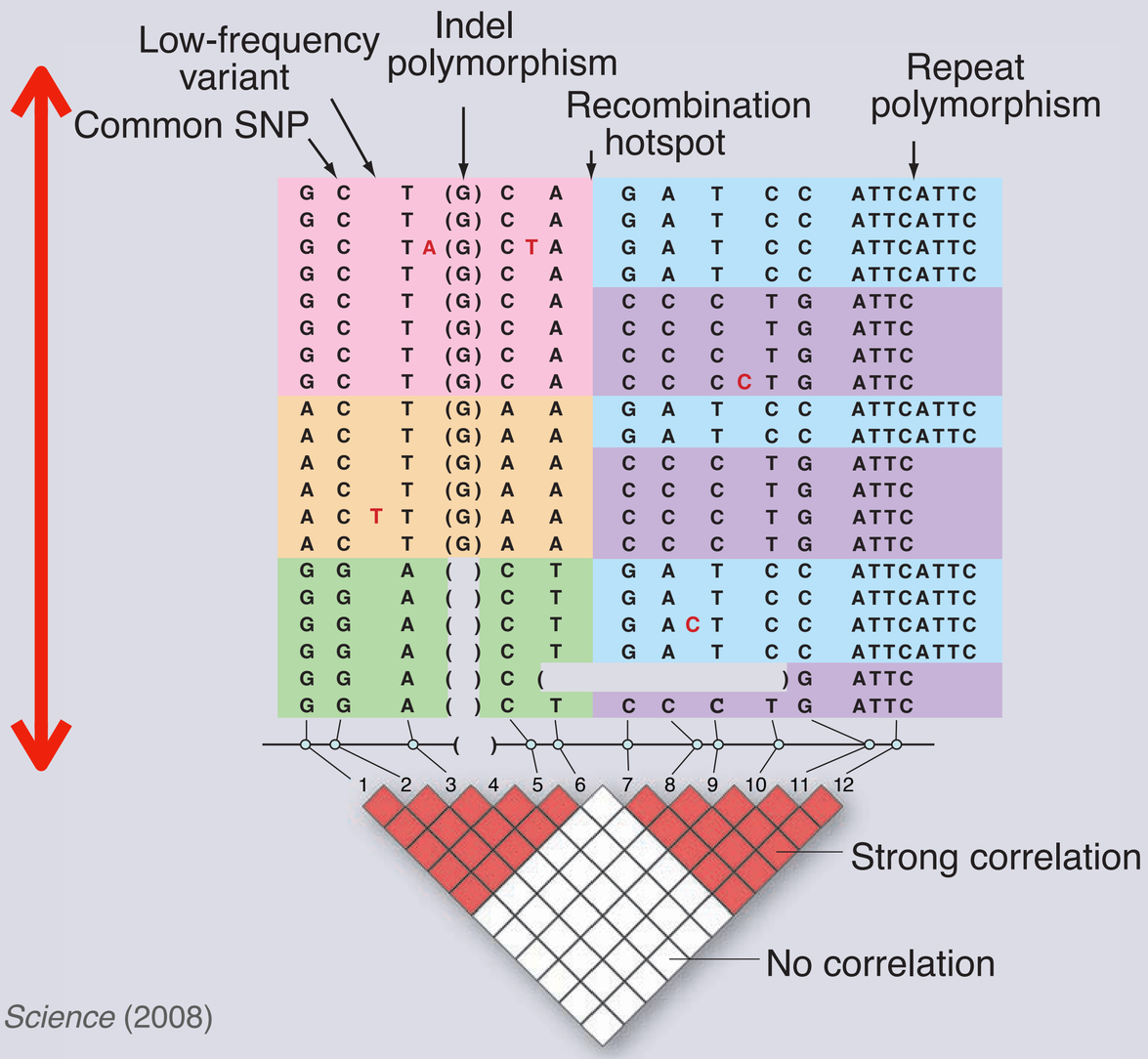
Gene expression (other phenotypes)

Genotype matrix individual x SNPs

SNPs x 1 effect size

Genetic variation in one figure

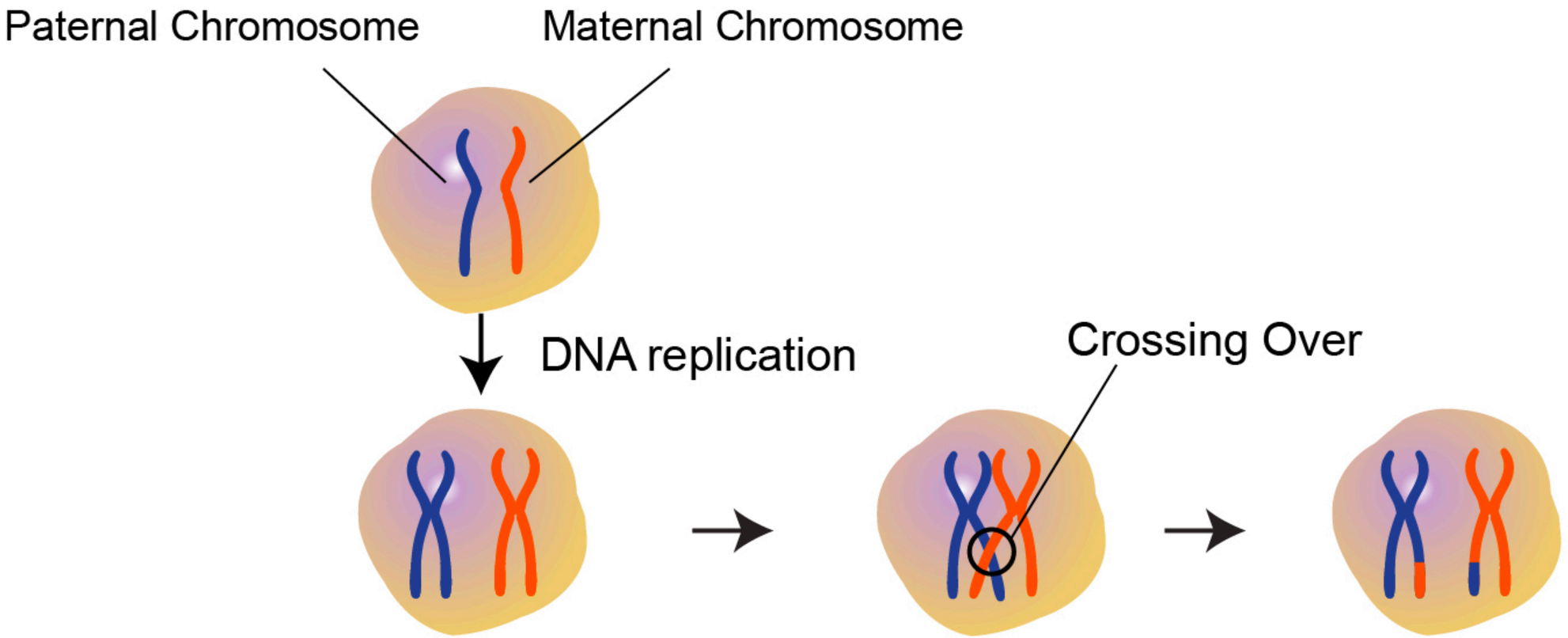
across many individuals (diploid genomes)



Common SNPs
Insertion/deletion
Other low-freq. variants
Other structural variants
Recombination hotspot

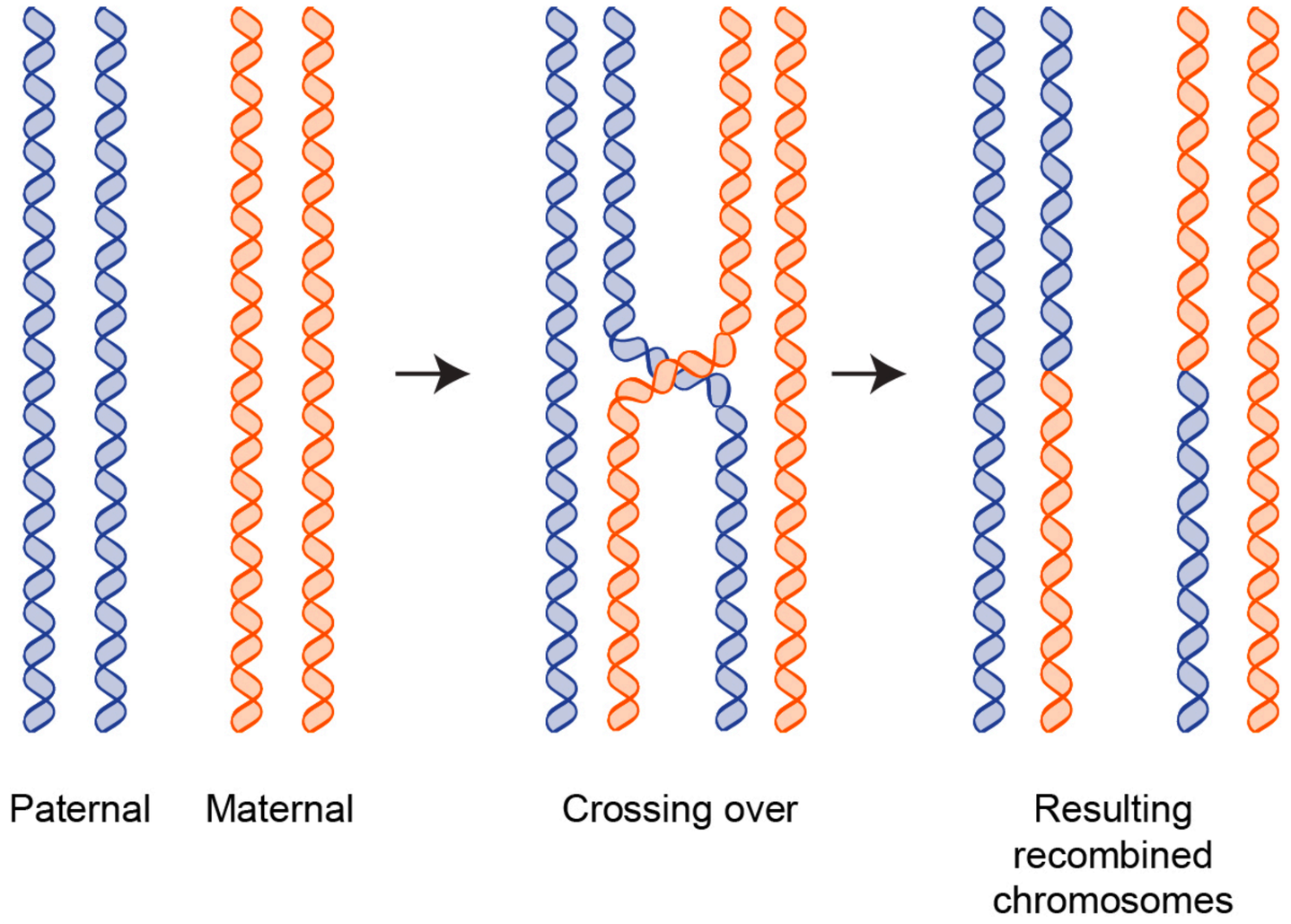
Altshuler, Daly, Lander, *Science* (2008)

Recombination: Mixing the maternal and paternal copies

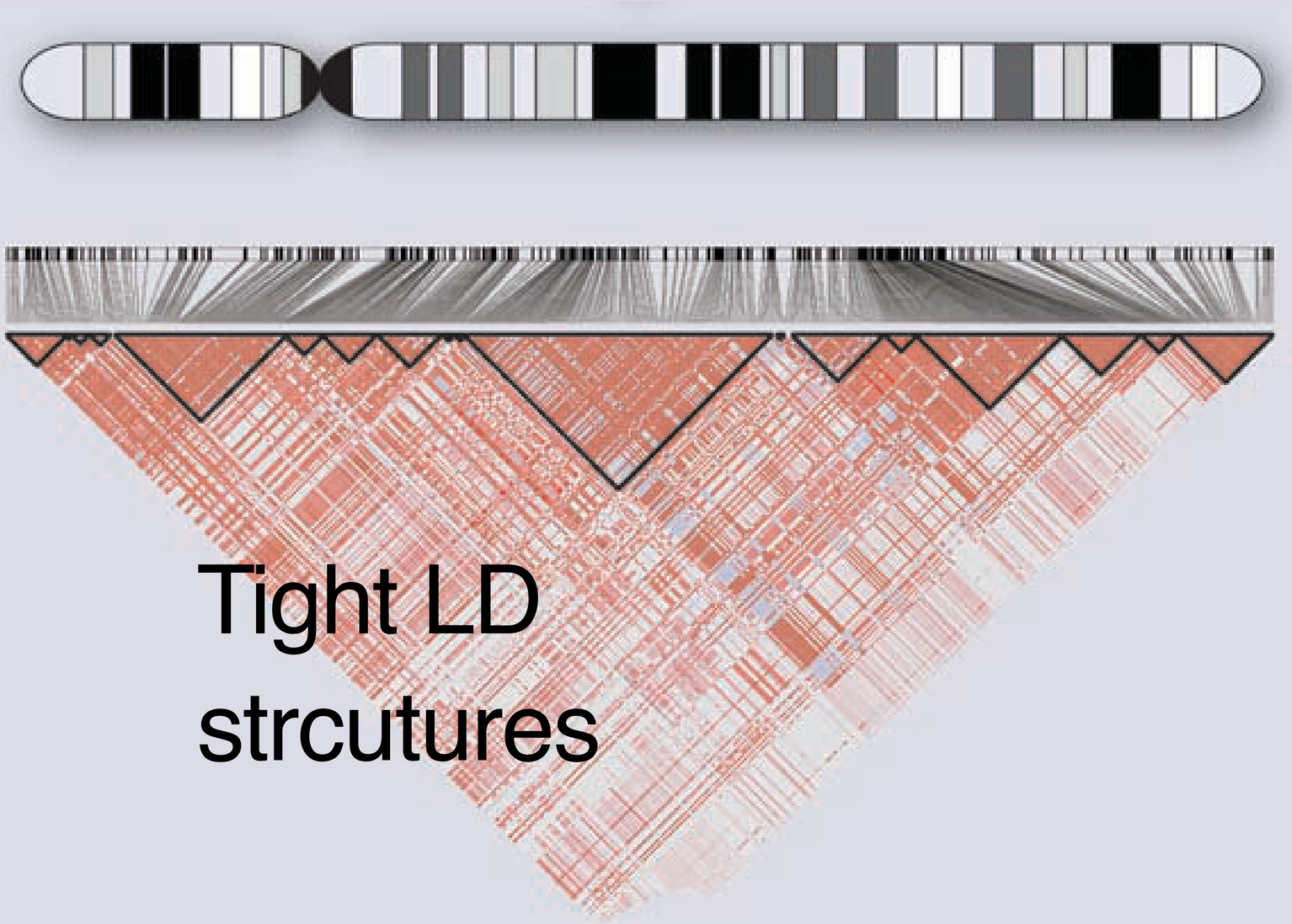


https://www.genome.gov/genetics-glossary

Recombination: Mixing the maternal and paternal copies



https://www.genome.gov/genetics-glossary



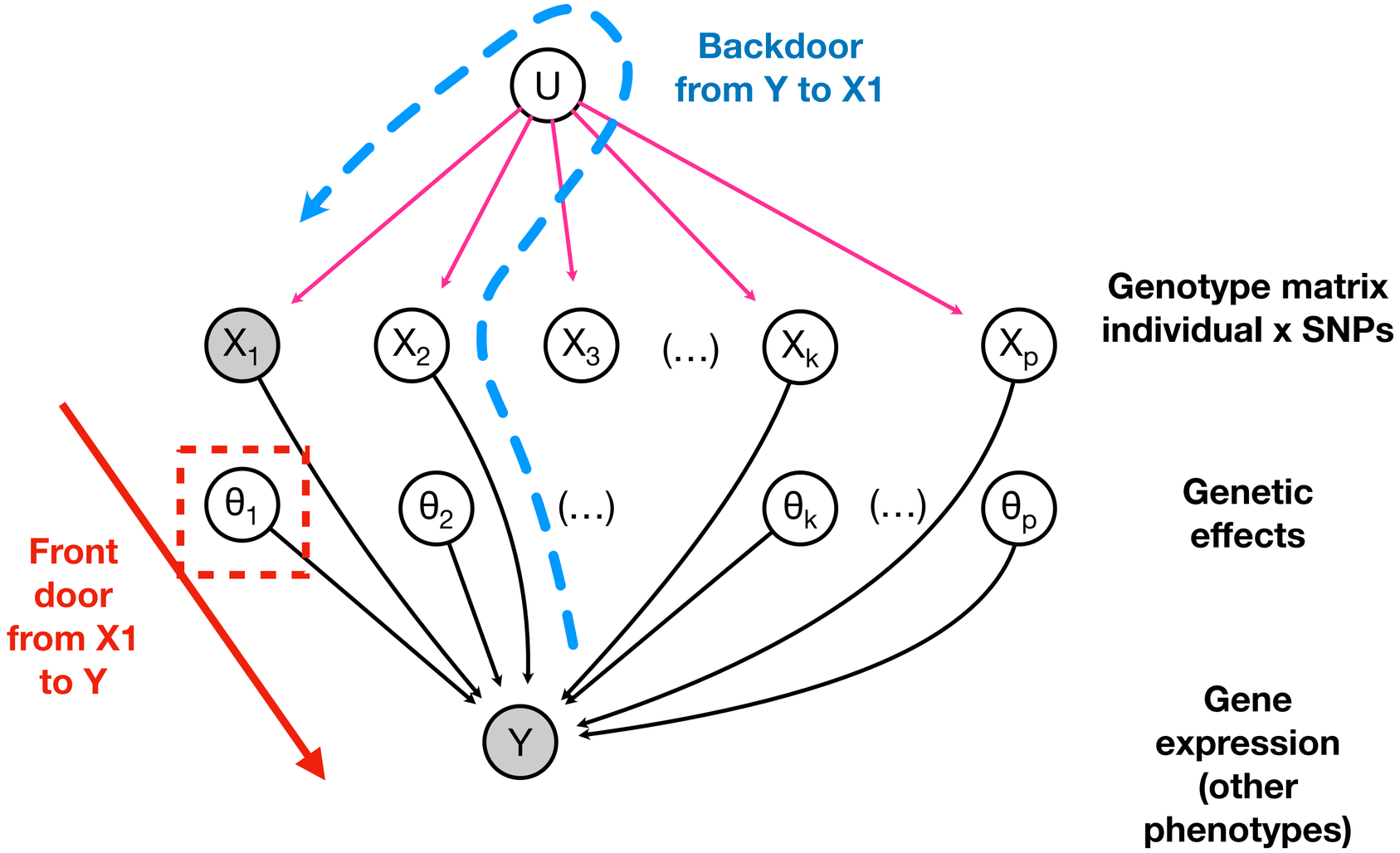
Tight LD
strcutures

1M "tagging" SNPs across the Human Genome

~ 1M independent loci (locations)

~ 5M variants

Genetics problems are hard due to unmeasured confounding factors

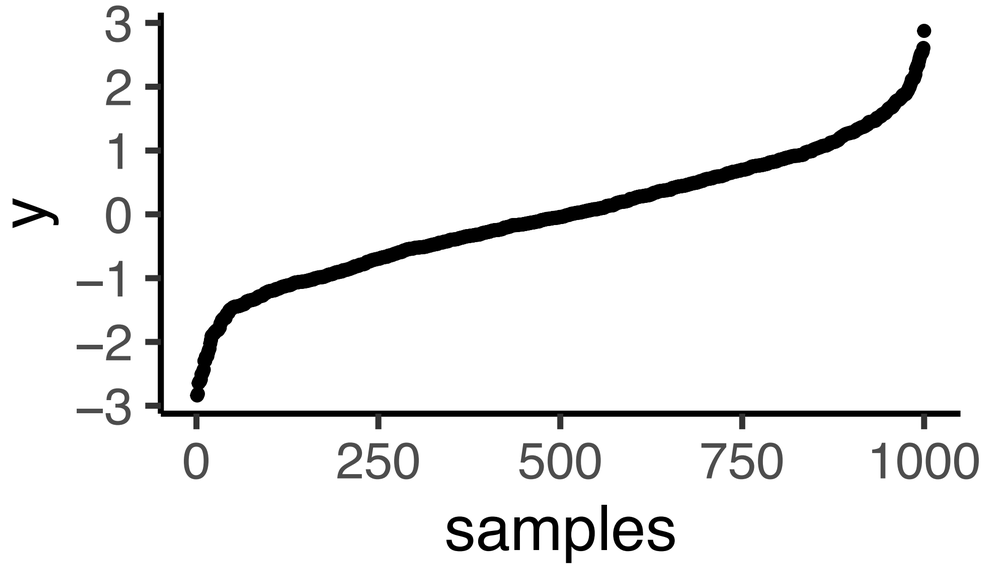


Revisit a multivariate liner regression problem

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \theta_1 \begin{pmatrix} X_{11} \\ X_{21} \\ \vdots \\ X_{n1} \end{pmatrix} + \cdots \theta_p \begin{pmatrix} X_{1p} \\ X_{2p} \\ \vdots \\ X_{np} \end{pmatrix}.$$

- Different number of variables will define a class of potential models
- E.g., $\mathcal{F}_1$: a class of models with one variable
- $\mathcal{F}_2$: a class of models with two variables
- (...)
- $\mathcal{F}_q$: a class of models with q variables

Challenges in a $p \gg n$ regression problem in genomics

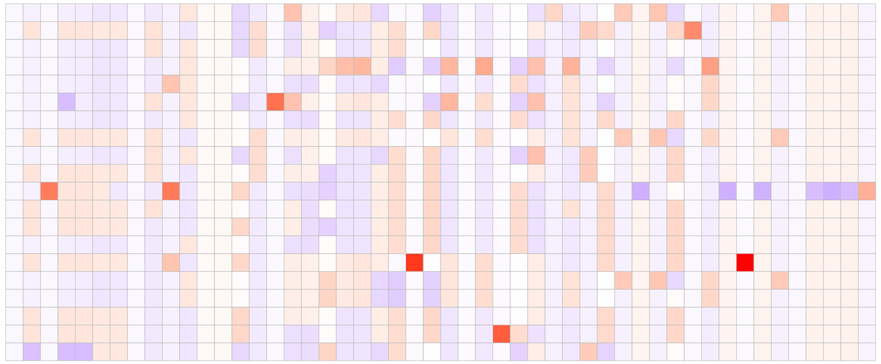



```
dim(y)
```

```
[1] 1000 1
```

There are 10 true non-zero variables.

1000 x 2000



...

...

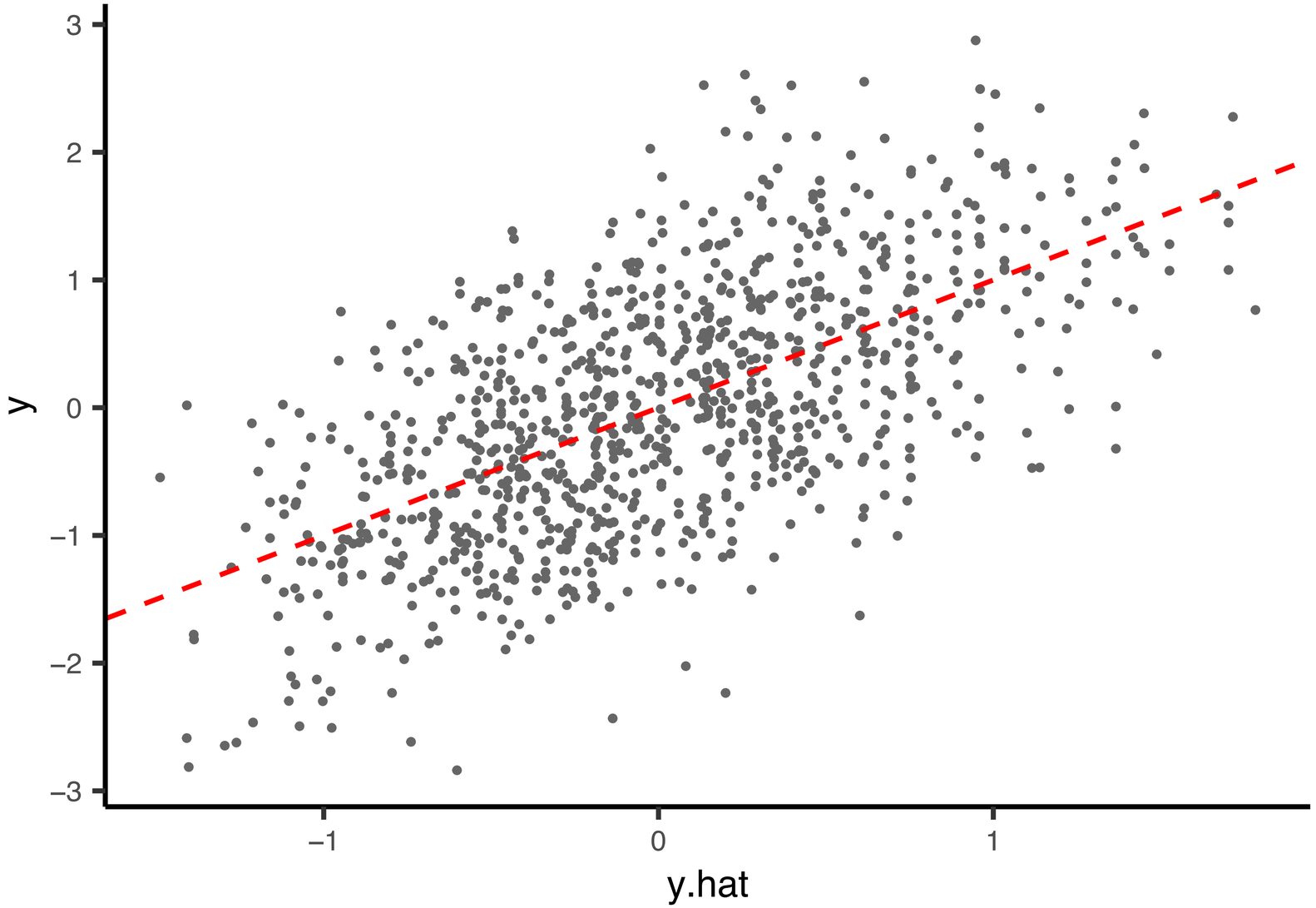
...

```
dim(X)
```

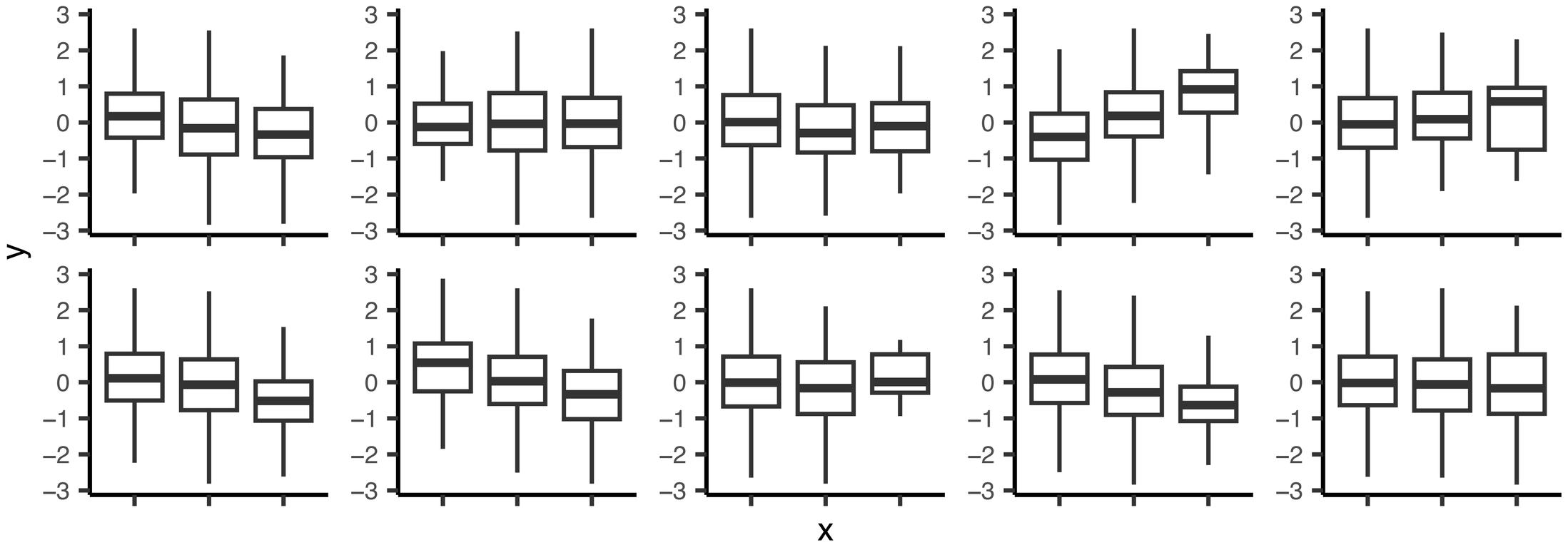
```
[1] 1000 2000
```

True causal variables explain a large fraction of variation

```
.lm <- lm(y ~ X[, sim$causal, drop = FALSE] - 1)
```



Variant-by-variant correlations



How do we know “causal” variables from 2,000 variables?

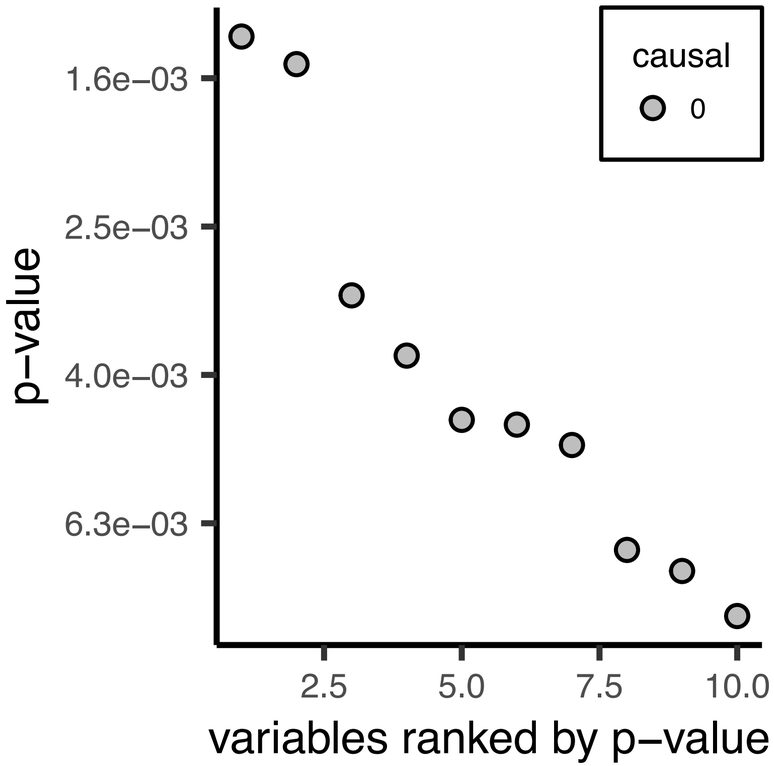
- Let’s try out one by one and rank them by univariate

```
cor.test(x,y)
```

How do we know "causal" variables from 2,000 variables?

- Let's try out one by one and rank them by univariate

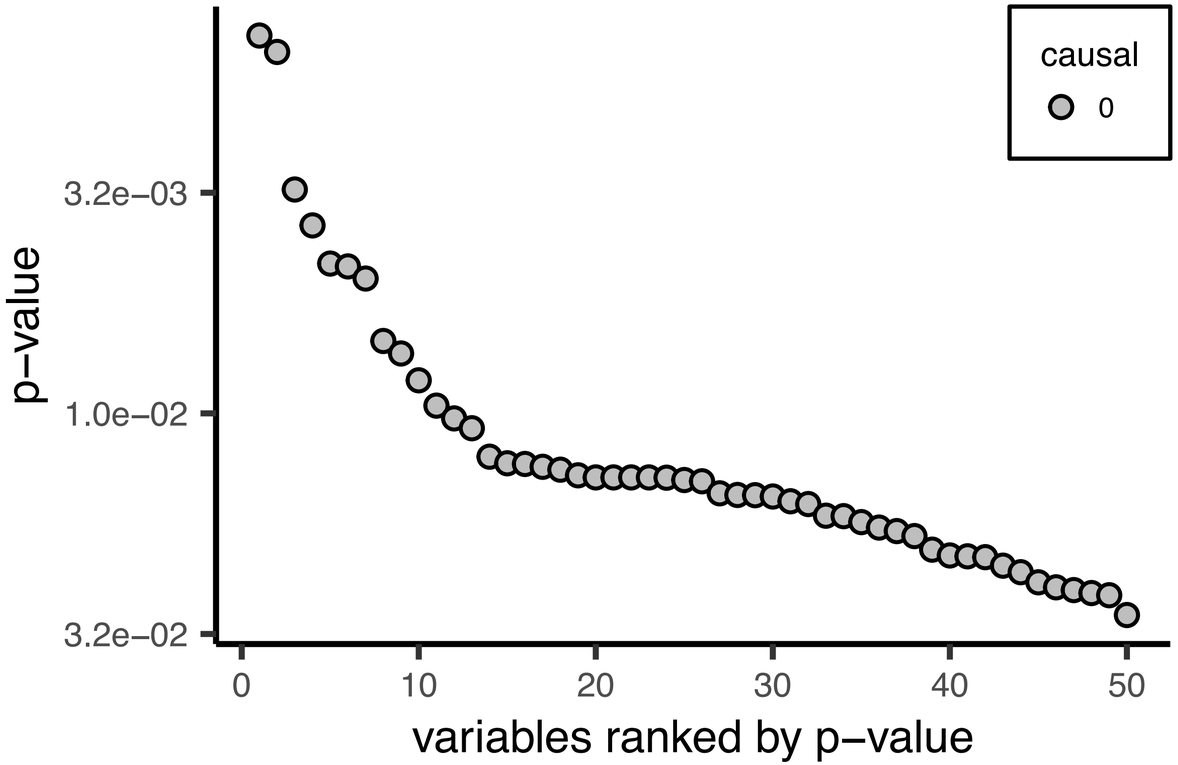
```
cor.test(x,y)
```



How do we know "causal" variables from 2,000 variables?

- Let's try out one by one and rank them by univariate

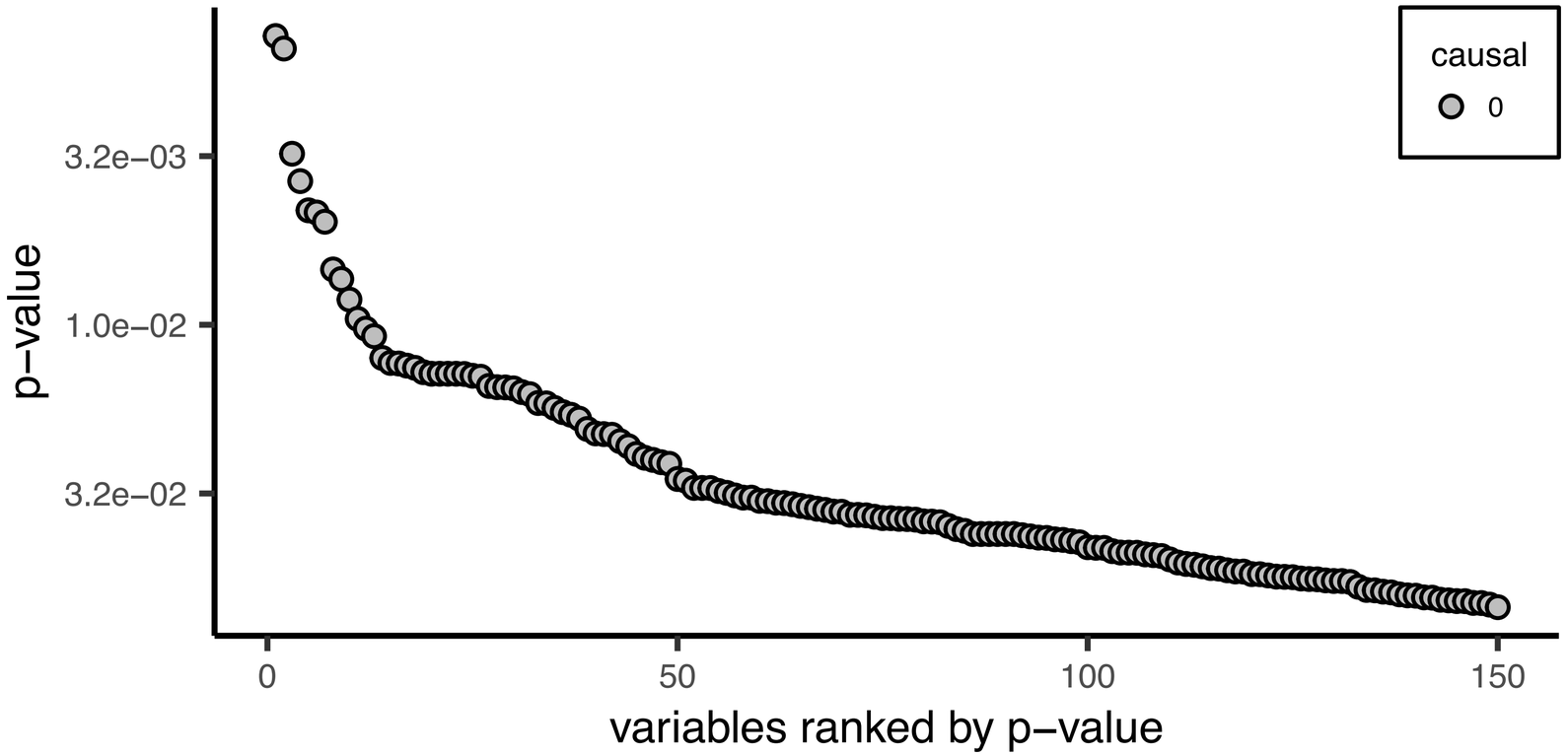
```
cor.test(x,y)
```



How do we know "causal" variables from 2,000 variables?

- Let's try out one by one and rank them by univariate

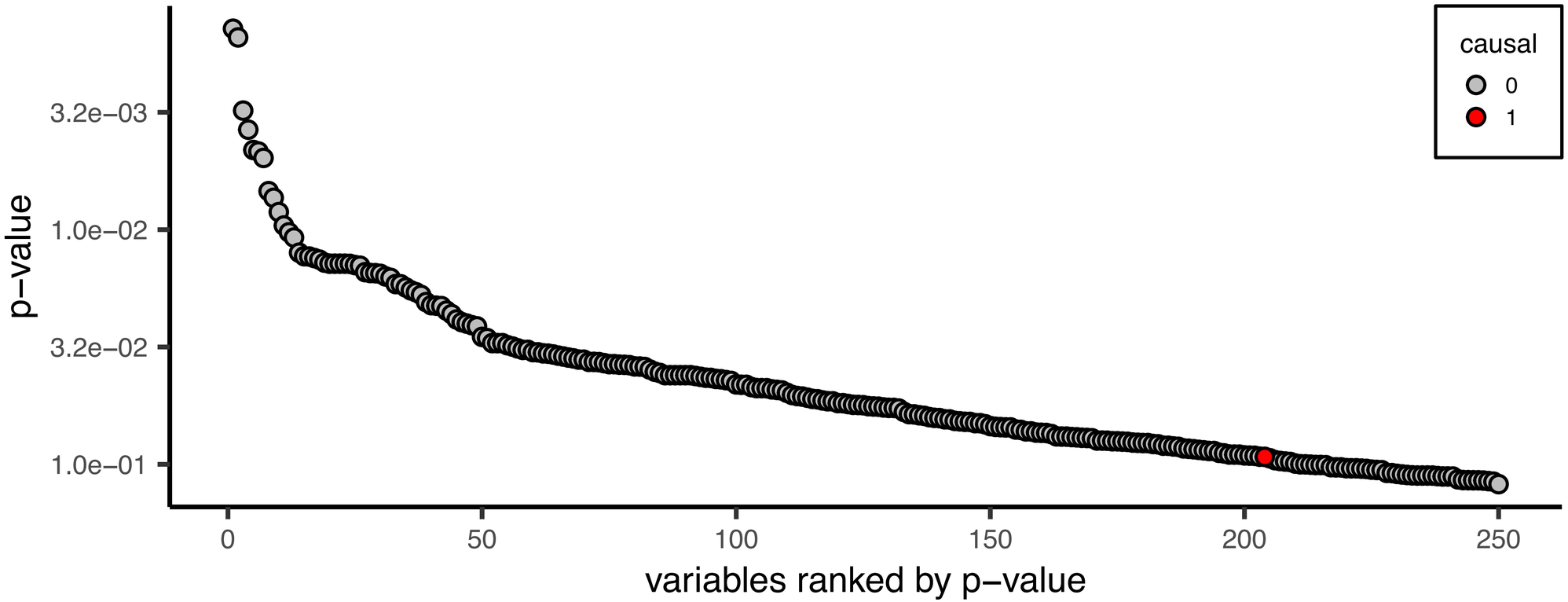
```
cor.test(x,y)
```



How do we know "causal" variables from 2,000 variables?

- Let's try out one by one and rank them by univariate

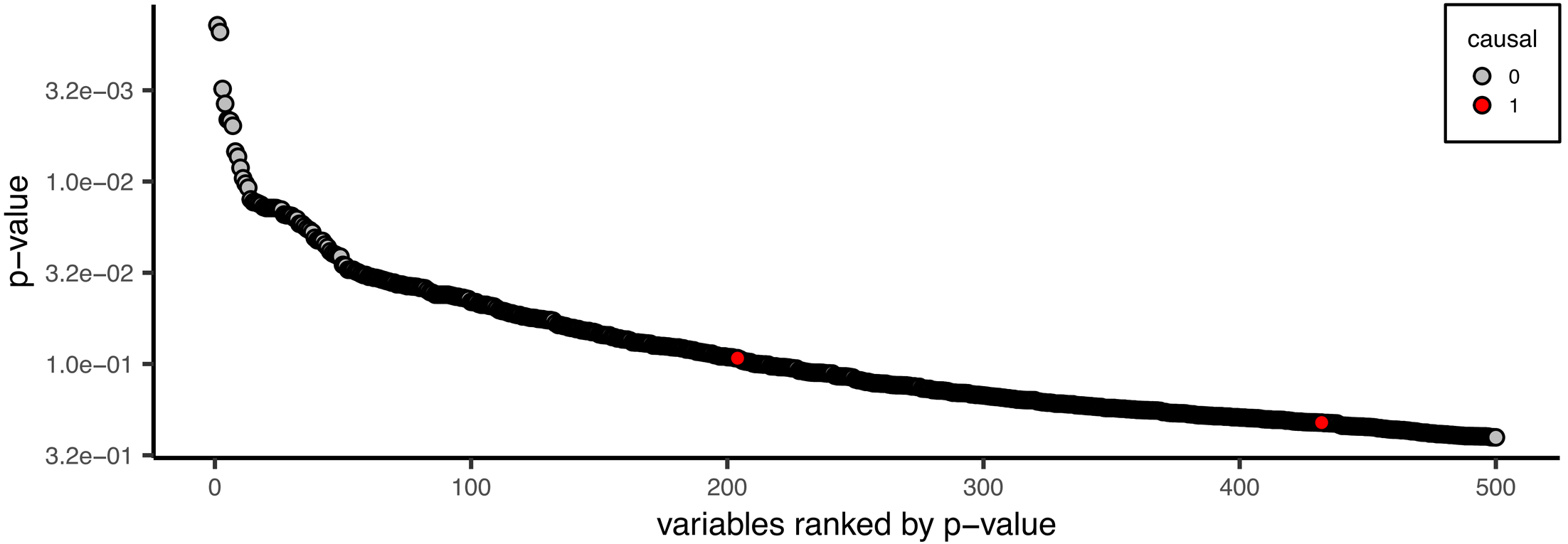
```
cor.test(x,y)
```



How do we know "causal" variables from 2,000 variables?

- Let's try out one by one and rank them by univariate

```
cor.test(x,y)
```



Classical statistics does not really help

- Classical variable selection by univariate (one-by-one) tests will not work for a $p \gg n$ regression problem
- Especially if we have col-linearity in the design matrix X

Variable selection in high-dimensional genotype matrix ($n \ll p$)

Regression analysis $=$ projecting the observed $\mathbf{y}$ vector on to column space of $\{\mathbf{x}_j : j \in [p]\}$,

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \theta_1 \begin{pmatrix} X_{11} \\ X_{21} \\ \vdots \\ X_{n1} \end{pmatrix} + \cdots \theta_p \begin{pmatrix} X_{1p} \\ X_{2p} \\ \vdots \\ X_{np} \end{pmatrix}.$$

Variable selection $=$ column selection.

- Intuitive idea : choose the best combination of variables. $\rightarrow 2^p$ choices (even harder).

Variable selection in high-dimensional genotype matrix ($n \ll p$)

Regression analysis $=$ projecting the observed $\mathbf{y}$ vector on to column space of $\{\mathbf{x}_j : j \in [p]\}$,

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \theta_1 \begin{pmatrix} X_{11} \\ X_{21} \\ \vdots \\ X_{n1} \end{pmatrix} + \cdots \theta_p \begin{pmatrix} X_{1p} \\ X_{2p} \\ \vdots \\ X_{np} \end{pmatrix}.$$

Variable selection $=$ column selection.

- Intuitive idea : choose the best combination of variables. $\rightarrow 2^p$ choices (even harder).
- Alternative idea : make as many θ_j's nearly zero values.

Variable selection in high-dimensional genotype matrix ($n \ll p$)

Regression analysis $=$ projecting the observed $\mathbf{y}$ vector on to column space of $\{\mathbf{x}_j : j \in [p]\}$,

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \theta_1 \begin{pmatrix} X_{11} \\ X_{21} \\ \vdots \\ X_{n1} \end{pmatrix} + \cdots \theta_p \begin{pmatrix} X_{1p} \\ X_{2p} \\ \vdots \\ X_{np} \end{pmatrix}.$$

Variable selection $=$ column selection.

- Intuitive idea : choose the best combination of variables. $\rightarrow 2^p$ choices (even harder).
- Alternative idea : make as many θ_j's nearly zero values.
- What prior does: penalize $|\theta_j| > 0$ so that only the strong enough variables take non-zero values.

Today's lecture: Supervised Learning in Genomics

- **Non-parametric prediction methods**
  - When we don't have any idea of data-generation mechanisms.
  - Kernel (local) regression
  - Gaussian Process
- **Ensemble learning: the collective power of weak learners**
  - Expectation maximization
  - Mixture of linear regressions
- **Variable selection**
  - Challenges in high-dimensional prediction problems
  - Sparse regression models

Reconciling two related concepts – MLE and MSE

Equivalence of maximum-likelihood estimation and mean square error minimization (isotropic Gaussian error distribution).

[MLE] Find θ maximizing

$$\ln p(\mathbf{y}|X,\theta) = -\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \mathbf{x}_i\theta)^2 + \text{const.}$$

without prior contribution of parameter, and σ is known.

[MSE] Find θ minimizing

$$\sum_{i=1}^{n}(y_i - \mathbf{x}_i\theta)^2.$$

MLE, MSE, an optimization problem

Minimization of the convex loss function:

$$L(\theta) = (\mathbf{y} - X\theta)^\top (\mathbf{y} - X\theta)$$

MLE, MSE, an optimization problem

Minimization of the convex loss function:

$$L(\theta) = (\mathbf{y} - X\theta)^\top (\mathbf{y} - X\theta)$$

We can optimize setting the derivative with respect to θ to zero:

$$\nabla_\theta L = X^\top (\mathbf{y} - X\theta) = 0$$

Rearranging the equation

$$\mathbf{y}^\top X = X^\top X\theta \implies \hat{\theta}_{MLE} = (X^\top X)^{-1} X^\top y.$$

MLE, MSE, an optimization problem

Minimization of the convex loss function:

$$L(\theta) = (\mathbf{y} - X\theta)^\top (\mathbf{y} - X\theta)$$

We can optimize setting the derivative with respect to θ to zero:

$$\nabla_\theta L = X^\top (\mathbf{y} - X\theta) = 0$$

Rearranging the equation

$$\mathbf{y}^\top X = X^\top X\theta \implies \hat{\theta}_{MLE} = (X^\top X)^{-1} X^\top y.$$

- Approximately, $p(\theta|\mathbf{y}, X) \approx \mathcal{N}\left(\theta \middle| \hat{\theta}_{MLE}, \sigma^2 (X^\top X)^{-1}\right)$.

MLE, MSE, an optimization problem

Minimization of the convex loss function:

$$L(\theta) = (\mathbf{y} - X\theta)^\top (\mathbf{y} - X\theta)$$

We can optimize setting the derivative with respect to θ to zero:

$$\nabla_\theta L = X^\top (\mathbf{y} - X\theta) = 0$$

Rearranging the equation

$$\mathbf{y}^\top X = X^\top X\theta \implies \hat{\theta}_{MLE} = (X^\top X)^{-1} X^\top y.$$

- Approximately, $p(\theta|\mathbf{y}, X) \approx \mathcal{N}\left(\theta \middle| \hat{\theta}_{MLE}, \sigma^2 (X^\top X)^{-1}\right)$.
- How hard is $(X^\top X)^{-1}$ (i.e., inverse of $p \times p$ matrix)?

MLE, MSE, an optimization problem

Minimization of the convex loss function:

$$L(\theta) = (\mathbf{y} - X\theta)^\top (\mathbf{y} - X\theta)$$

We can optimize setting the derivative with respect to θ to zero:

$$\nabla_\theta L = X^\top (\mathbf{y} - X\theta) = 0$$

Rearranging the equation

$$\mathbf{y}^\top X = X^\top X\theta \implies \hat{\theta}_{MLE} = (X^\top X)^{-1} X^\top y.$$

- Approximately, $p(\theta|\mathbf{y}, X) \approx \mathcal{N}\left(\theta \middle| \hat{\theta}_{MLE}, \sigma^2 (X^\top X)^{-1}\right)$.
- How hard is $(X^\top X)^{-1}$ (i.e., inverse of $p \times p$ matrix)?
- What if $n \ll p$? What if we want to include $p(\theta)$?

Bayesian/regularization idea to add the missing probability component

We've been discussing the conditional likelihood

$$p(\mathbf{y}|X, \theta)$$

without a prior probability of regression coefficients,

$$p(\theta)$$

What will be a suitable prior distribution of θ?

Recall: Reconciling two related concepts – MLE and MSE

Equivalence of maximum-likelihood estimation and mean square error minimization (isotropic Gaussian error distribution).

[MLE] Find θ maximizing

$$\ln p(\mathbf{y}|X,\theta) = -\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \mathbf{x}_i\theta)^2 + \text{const.}$$

without prior contribution of parameter, and σ is known.

[MSE] Find θ minimizing

$$\sum_{i=1}^{n}(y_i - \mathbf{x}_i\theta)^2.$$

Ridge regression, a linear regression with Gaussian prior (L2)

Prior distribution

$$p(\theta) = \mathcal{N}(\theta|\mathbf{0}, \lambda^{-1}I) \propto \exp\left(-\frac{\lambda}{2}\|\theta\|^2\right)$$

where

$$\|\theta\|^2 = \sum_{j=1}^{p} \theta_j^2, \text{ L2-norm.}$$

Maximize

$$\ln p(\mathbf{y}|X, \theta) + \ln p(\theta|\lambda) = -\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \mathbf{x}_i\theta)^2 - \frac{\lambda}{2}\|\theta\|^2$$

Minimize L_2-regularized error

$$\sum_{i=1}^{n}(y_i - \mathbf{x}_i\theta)^2 + \frac{\lambda}{2}\|\theta\|^2$$

Lasso regression, a linear regression with Laplace prior (L1)

Prior distribution

$$p(\theta) = \text{Laplace}(\theta|\lambda) \propto \exp\left(-\lambda\|\theta\|_1\right)$$

where

$$\|\theta\|_1 = \sum_{j=1}^{p} |\theta_j|, \text{ L1-norm.}$$

Maximize

$$\ln p(\mathbf{y}|X,\theta) + \ln p(\theta|\lambda) = -\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \mathbf{x}_i\theta)^2 - \lambda\|\theta\|_1$$

Minimize L_1-regularized error

$$\sum_{i=1}^{n}(y_i - \mathbf{x}_i\theta)^2 + \lambda\|\theta\|_1$$

(Tibshirani, 1996)

Posterior inference of the regularized regression models

Our goal is to estimate (1) posterior distribution

$$p(\theta|\mathbf{y}, X) = \frac{p(\mathbf{y}|X, \theta)p(\theta)}{p(y|X)}.$$

Then (2) using $p(\theta|\mathbf{y}, X)$, predict $p(\mathbf{y}^\star|\mathbf{y}, X)$ by averaging over all possible θ sampled from the estimated posterior distribution.

- Usually posterior prediction (2) is can be easily simulated with accurate estimation of posterior distribution (1).
- Posterior inference can be done analytically or not, depending on the choice of $p(\theta)$[1].

[1] We term prior $p(\theta)$ a *conjugate prior* if its posterior $p(\theta|\mathbf{y}, X)$ is of the same type of distribution.

We can find an analytical solution in L2-regularized regression

$$\ln p(\theta|\mathbf{y}, X) = -\frac{1}{2\sigma^2}(\mathbf{y} - X\theta)^\top(\mathbf{y} - X\theta) - \frac{\lambda}{2}\theta^\top\theta + \text{const.}$$

We can find an analytical solution in L2-regularized regression

$$\ln p(\theta|\mathbf{y}, X) = -\frac{1}{2\sigma^2}(\mathbf{y} - X\theta)^\top(\mathbf{y} - X\theta) - \frac{\lambda}{2}\theta^\top\theta + \text{const.}$$

By taking derivative with respect to θ and setting it the zero vector:

$$\nabla_\theta = -\frac{1}{\sigma^2}X^\top(\mathbf{y} - X\theta) - \lambda\theta = 0$$

We can find an analytical solution in L2-regularized regression

$$\ln p(\theta|\mathbf{y}, X) = -\frac{1}{2\sigma^2}(\mathbf{y} - X\theta)^\top(\mathbf{y} - X\theta) - \frac{\lambda}{2}\theta^\top\theta + \text{const.}$$

By taking derivative with respect to θ and setting it the zero vector:

$$\nabla_\theta = -\frac{1}{\sigma^2}X^\top(\mathbf{y} - X\theta) - \lambda\theta = 0$$

Rearranging the equation:

$$X^\top\mathbf{y} = (X^\top X + \lambda\sigma^2 I)\theta \implies \hat{\theta} = (X^\top X + \lambda\sigma^2 I)^{-1}X^\top\mathbf{y}$$

We can find an analytical solution in L2-regularized regression

$$\ln p(\theta|\mathbf{y}, X) = -\frac{1}{2\sigma^2}(\mathbf{y} - X\theta)^\top(\mathbf{y} - X\theta) - \frac{\lambda}{2}\theta^\top\theta + \text{const.}$$

By taking derivative with respect to θ and setting it the zero vector:

$$\nabla_\theta = -\frac{1}{\sigma^2}X^\top(\mathbf{y} - X\theta) - \lambda\theta = 0$$

Rearranging the equation:

$$X^\top\mathbf{y} = (X^\top X + \lambda\sigma^2 I)\theta \implies \hat{\theta} = (X^\top X + \lambda\sigma^2 I)^{-1}X^\top\mathbf{y}$$

Remark: For $n \ll p$, the inverse $(X^\top X)^{-1}$ may not exists, but $(X^\top X + \lambda\sigma^2 I)^{-1}$ can exist with a proper λ.

For L1/L2 constraints, the greedy algorithm of `glmnet` works so well

Goal:

$$\min_{\theta} \quad \overbrace{(\mathbf{y} - X\theta)^\top (\mathbf{y} - X\theta)}^{\text{RSS}} + \underbrace{\lambda\alpha\|\theta\|_1}_{\text{variable selection}} + \underbrace{\lambda(1-\alpha)\|\theta\|_2}_{\text{shrinkage}}$$

For L1/L2 constraints, the greedy algorithm of `glmnet` works so well

Goal:

$$\min_{\theta} \quad \overbrace{(\mathbf{y} - X\theta)^\top (\mathbf{y} - X\theta)}^{\text{RSS}} + \underbrace{\lambda\alpha\|\theta\|_1}_{\text{variable selection}} + \underbrace{\lambda(1-\alpha)\|\theta\|_2}_{\text{shrinkage}}$$

The variable-by-variable update equation makes sense:

For each θ_j,

$$\hat{\theta}_j^{\text{glmnet}} \leftarrow \frac{S\left(\sum_{i=1}^n X_{ij}(y_i - \hat{y}_i^{(-j)}), \lambda\alpha\right)}{\sum_{i=1}^n X_{ij}^2 + \lambda(1-\alpha)} \quad \text{vs.} \quad \theta_j^{\text{MLE}} \leftarrow \frac{\sum_{i=1}^n X_{ij}\left(y_i - \sum_{k \neq j} X_{ik}\hat{\theta}_k\right)}{\sum_{i=1}^n X_{ij}^2}$$

Friedman *et al.*, Regularization Paths for Generalized Linear Models via Coordinate Descent (2010)

For L1/L2 constraints, the greedy algorithm of `glmnet` works so well

Goal:

$$\min_{\theta} \quad \overbrace{(\mathbf{y} - X\theta)^\top (\mathbf{y} - X\theta)}^{\text{RSS}} + \underbrace{\lambda\alpha\|\theta\|_1}_{\text{variable selection}} + \underbrace{\lambda(1-\alpha)\|\theta\|_2}_{\text{shrinkage}}$$

The variable-by-variable update equation makes sense:

For each θ_j,

$$\hat{\theta}_j \leftarrow \frac{\overset{\text{threshold}}{S}\left(\sum_{i=1}^n X_{ij} \quad \overbrace{(y_i - y_i^{(-j)})}^{\text{residual w/o the variable } \theta_j} \quad , \lambda\alpha\right)}{\sum_{i=1}^n X_{ij}^2 + \underbrace{\lambda(1-\alpha)}_{\text{shrinkage}}}$$

where $S(z, \tau)$ will set it to zero if $|z| < \tau$.

Cross-validation: How do we tune hyper-parameters (e.g., λ)?

1. Divide the total training data $\mathcal{D}^{\text{train}} = \{(X, y)\}$ into two parts:
   - (1) cross-validation training $\{(X, y)\}$ and
   - (2) CV testing data $\{(X^\star, y^\star)\}$
2. For each different (λ, α) combination,
   - Train coefficients θ using CV training $\{(X, y)\} \subset \mathcal{D}^{\text{train}}$
   - Test how well $\sum_j X^\star_{ij} \hat{\theta}_j$ predicts $y^\star$?
3. Choose the optimal $(\lambda^\star, \alpha^\star)$

How do we tune hyper-parameters (e.g., λ)?

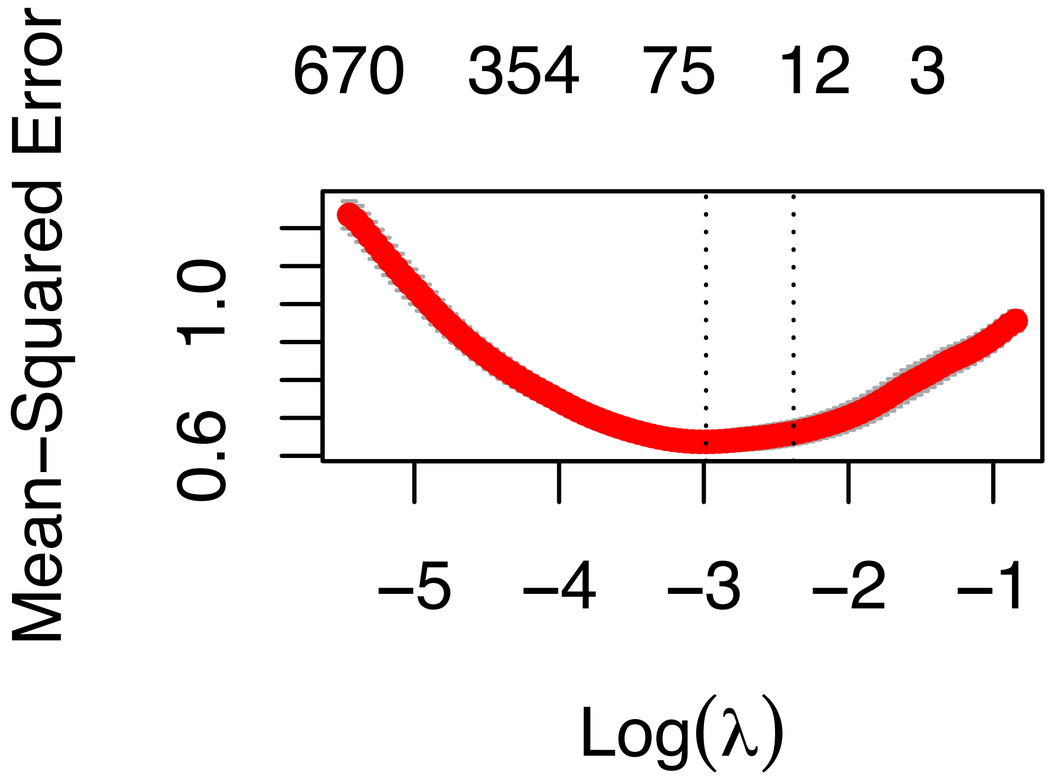
Well, in R, we simply run

```
glm.cv.out <-
 glmnet::cv.glmnet(X,
                   y,
                   nfolds=5,
                   alpha=1)
```

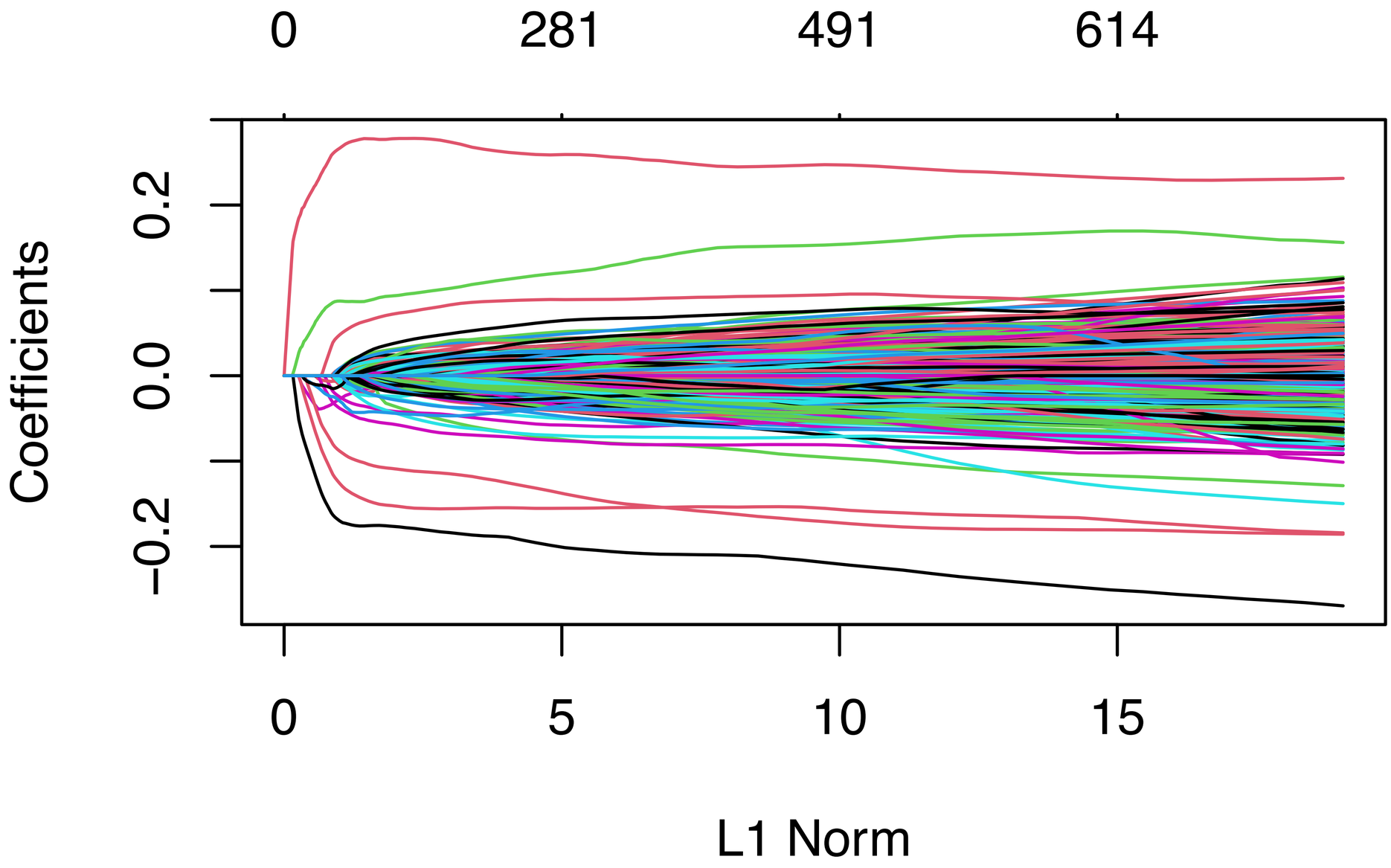
How do we tune hyper-parameters (e.g., λ)?

Well, in R, we simply run

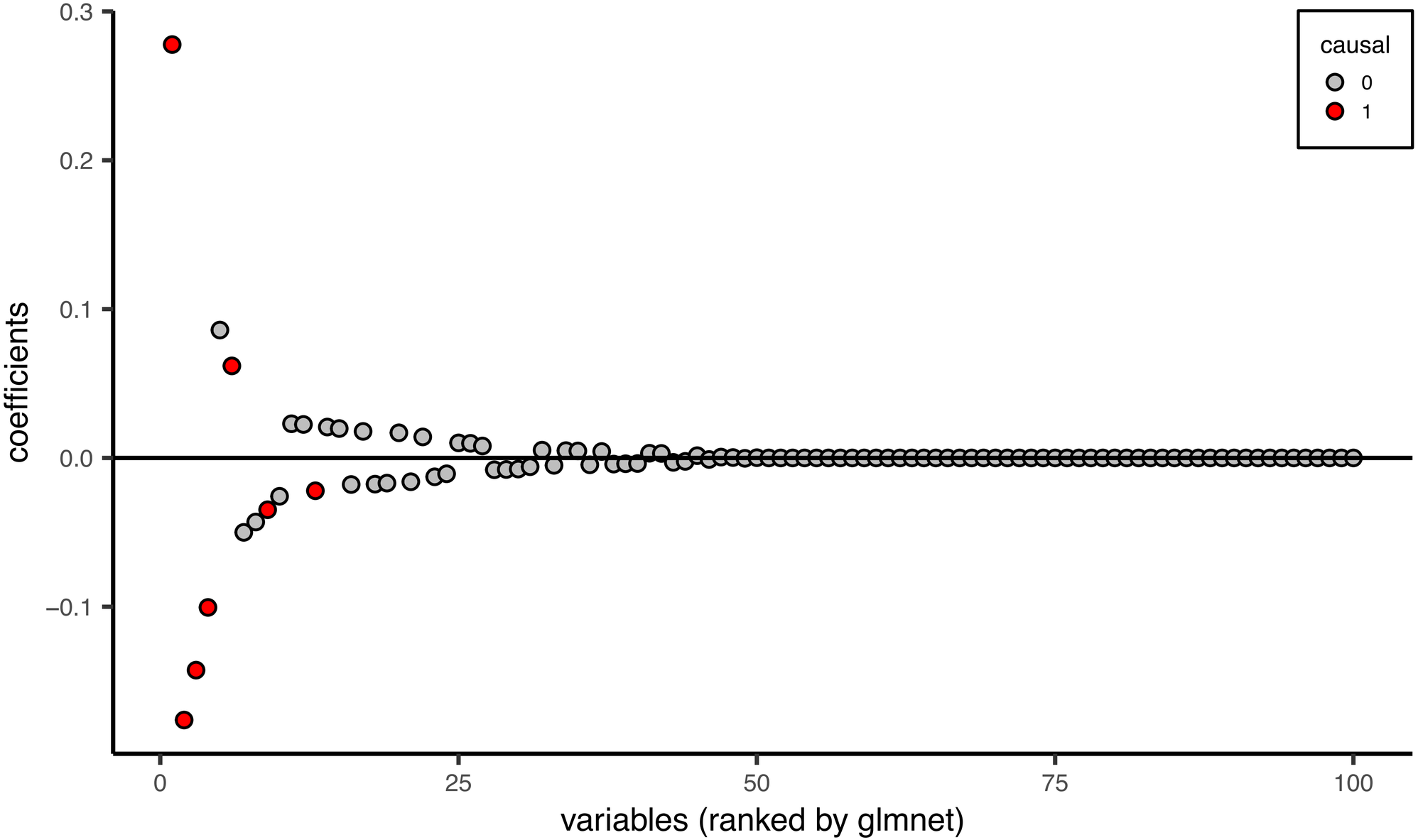
```
glm.cv.out <-
 glmnet::cv.glmnet(X,
                   y,
                   nfolds=5,
                   alpha=1)
```



Revisit our working example with L1-regularization (`glmnet`)



At the optimal λ value found by `cv.glmnet`...



Bias-variance tradeoff explains why a regularized regression works in practice

$$\min_{\theta} \quad \overbrace{(\mathbf{y} - X\theta)^\top (\mathbf{y} - X\theta)}^{\approx \text{bias}} + \underbrace{\lambda\alpha\|\theta\|_1}_{\text{variable selection}} + \underbrace{\lambda(1-\alpha)\|\theta\|_2}_{\text{shrinkage}}$$

- The second and the third terms control the model variance

Today's lecture: Supervised Learning in Genomics

- **Non-parametric prediction methods**
  - When we don't have any idea of data-generation mechanisms.
  - Kernel (local) regression
  - Gaussian Process
- **Ensemble learning: the collective power of weak learners**
  - Expectation maximization
  - Mixture of linear regressions
- **Variable selection**
  - Challenges in high-dimensional prediction problems
  - Sparse regression models

SuSiE: Sum of Single Effect Regression models

ROYAL STATISTICAL SOCIETY
DATA | EVIDENCE | DECISIONS

Journal of the Royal Statistical Society
Statistical Methodology
Series B

J. R. Statist. Soc. B (2020)
82, *Part* 5, *pp.* 1273–1300

A simple new approach to variable selection in regression, with application to genetic fine mapping

Gao Wang, Abhishek Sarkar, Peter Carbonetto and Matthew Stephens
University of Chicago, USA

[Received December 2018. Final revision May 2020]

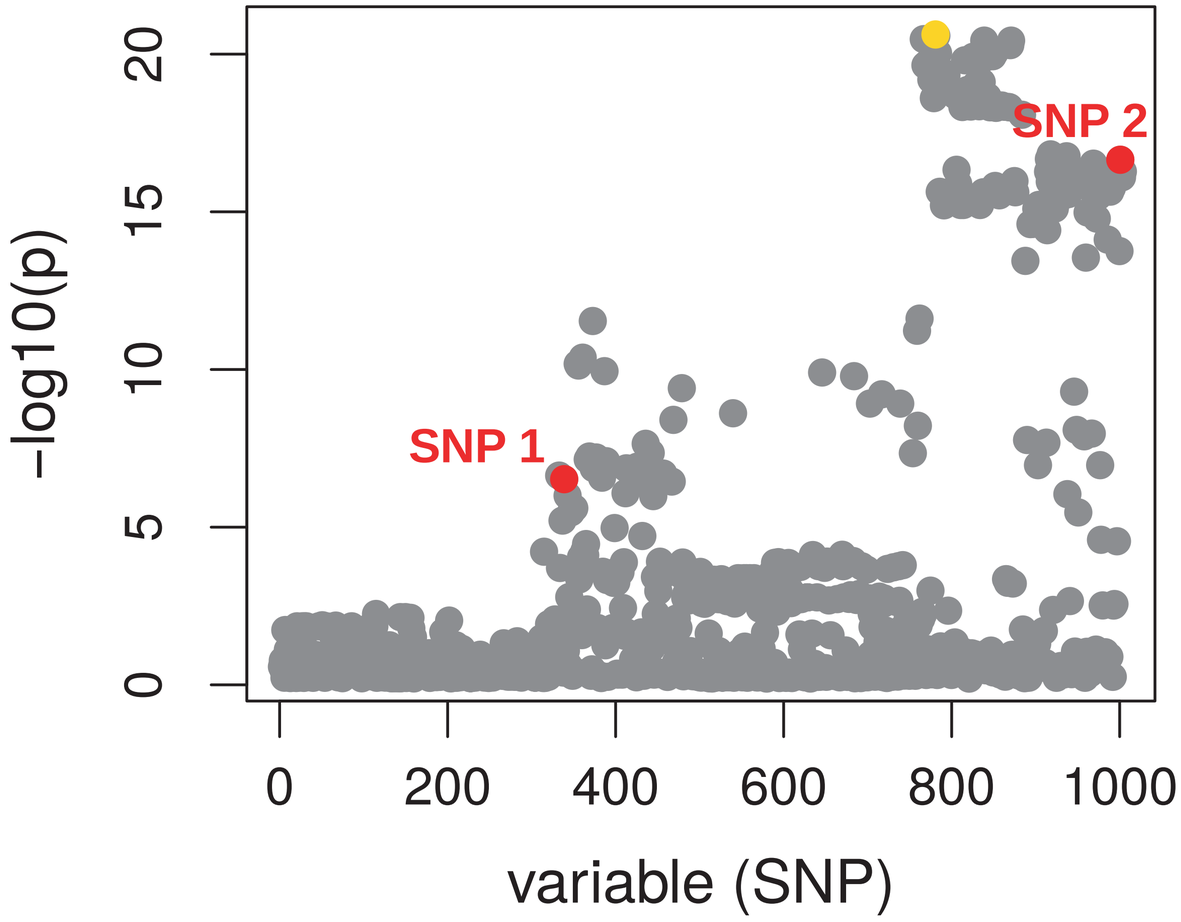
<u>Intuition</u>: Estimate a single-effect regression iterative!

$$\mathbf{y} = X\boldsymbol{\theta} + \epsilon \quad \Rightarrow \quad \boldsymbol{\theta} = \sum_{l=1}^{L} \boldsymbol{\beta}_l \quad \Rightarrow \quad \beta_{jl} = b_l \pi_{jl}, \ \sum_{j=1}^{p} \pi_{jl} = 1$$

single effect

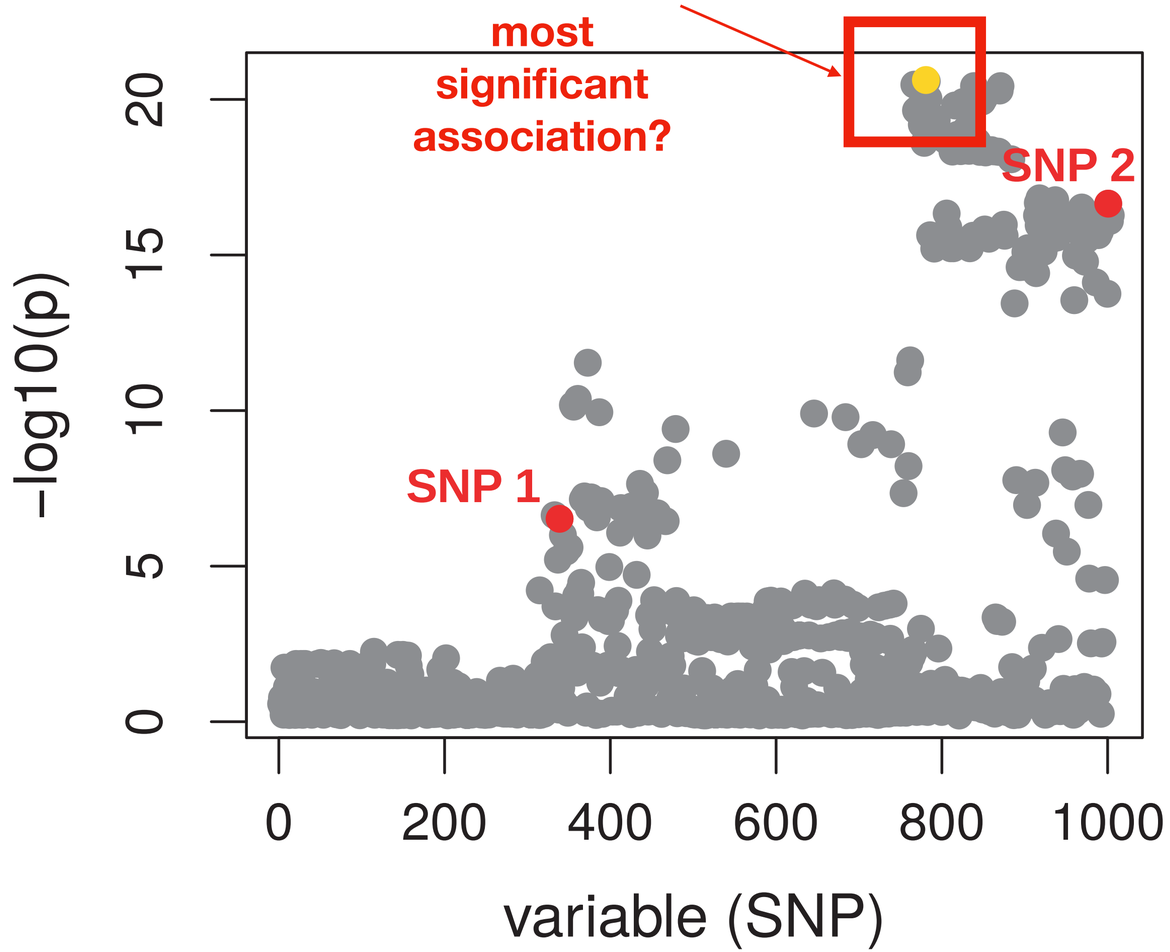
gene selection

How do we select a "causal" variant out of a thousand correlated ones?



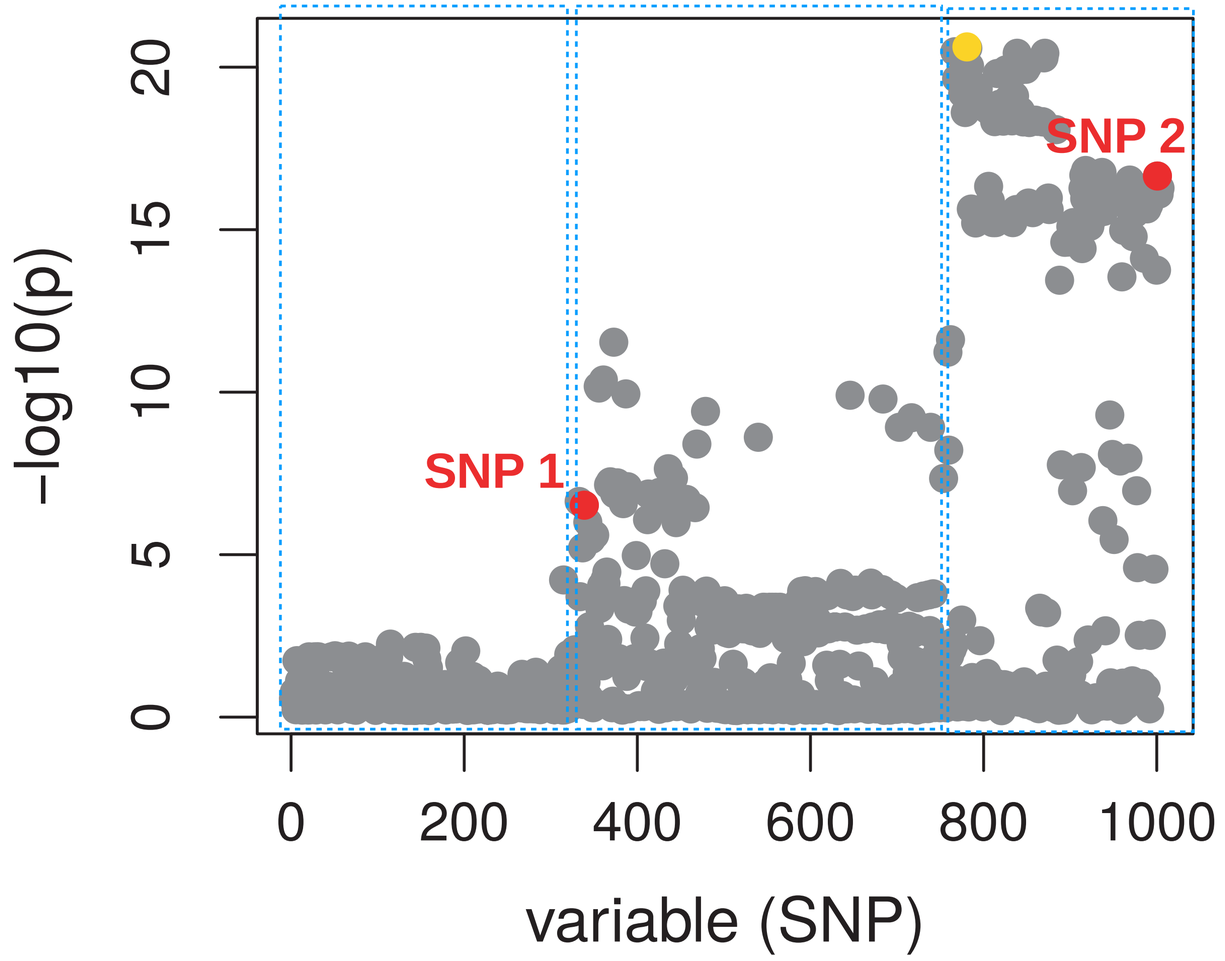
Wang, Sarkar, Carbonetto, Stephens (2020)

Can we pick the most significant one?



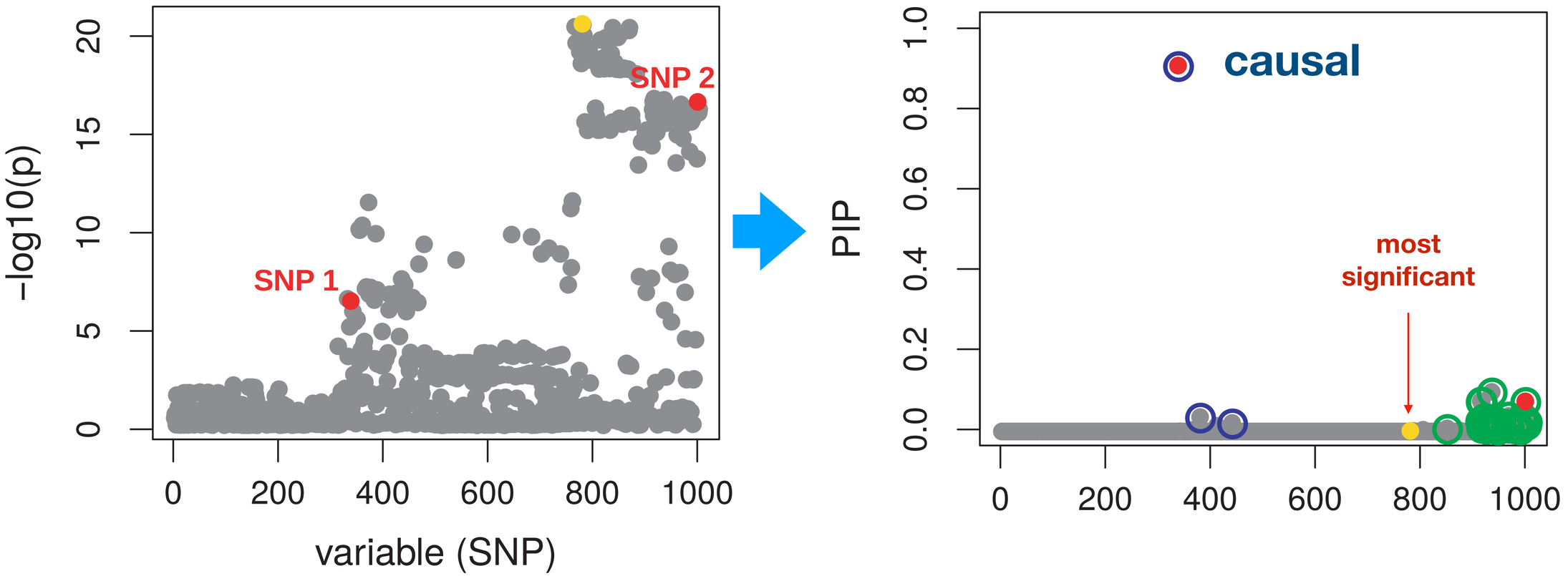
Wang, Sarkar, Carbonetto, Stephens (2020)

Are all the variants equal?



Wang, Sarkar, Carbonetto, Stephens (2020)

We need to establish a sparse causal set of variants!



PIP: posterior inclusion probability

Wang, Sarkar, Carbonetto, Stephens (2020)

Sum of Single Effect (SuSiE) regression

```
library(susieR)
susie.out <- susie(X, y)
```

$$\mathbf{y} = \sum_{l=1}^{L} \underbrace{\sum_{j} \mathbf{x}_j \quad \alpha_j \quad \beta_j}_{\text{layer-by-layer}} + \epsilon$$

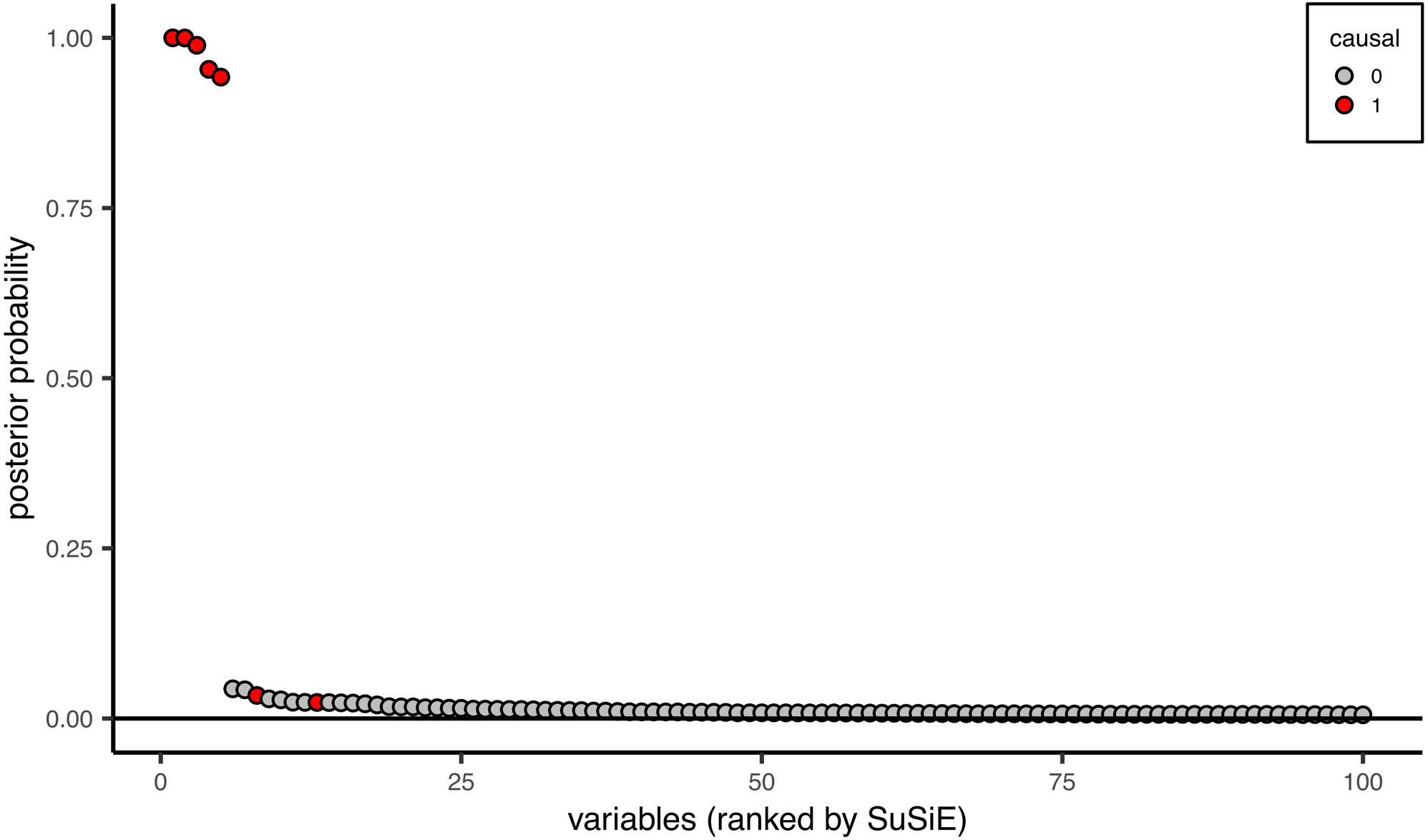
probabilistic selection

single variant effect

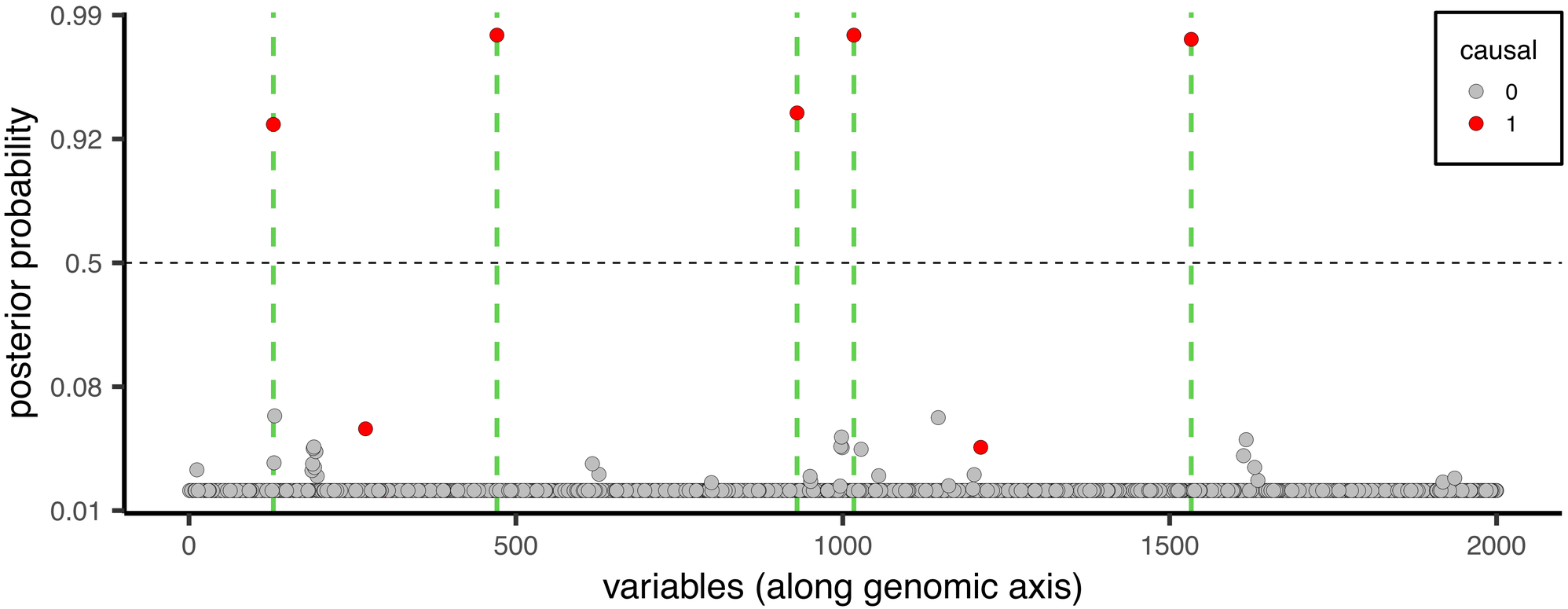
where $\sum_{j=1}^{p} \alpha_j = 1$.

Wang .. Stephens, *Journal of the Royal Statistical Society* (2020)

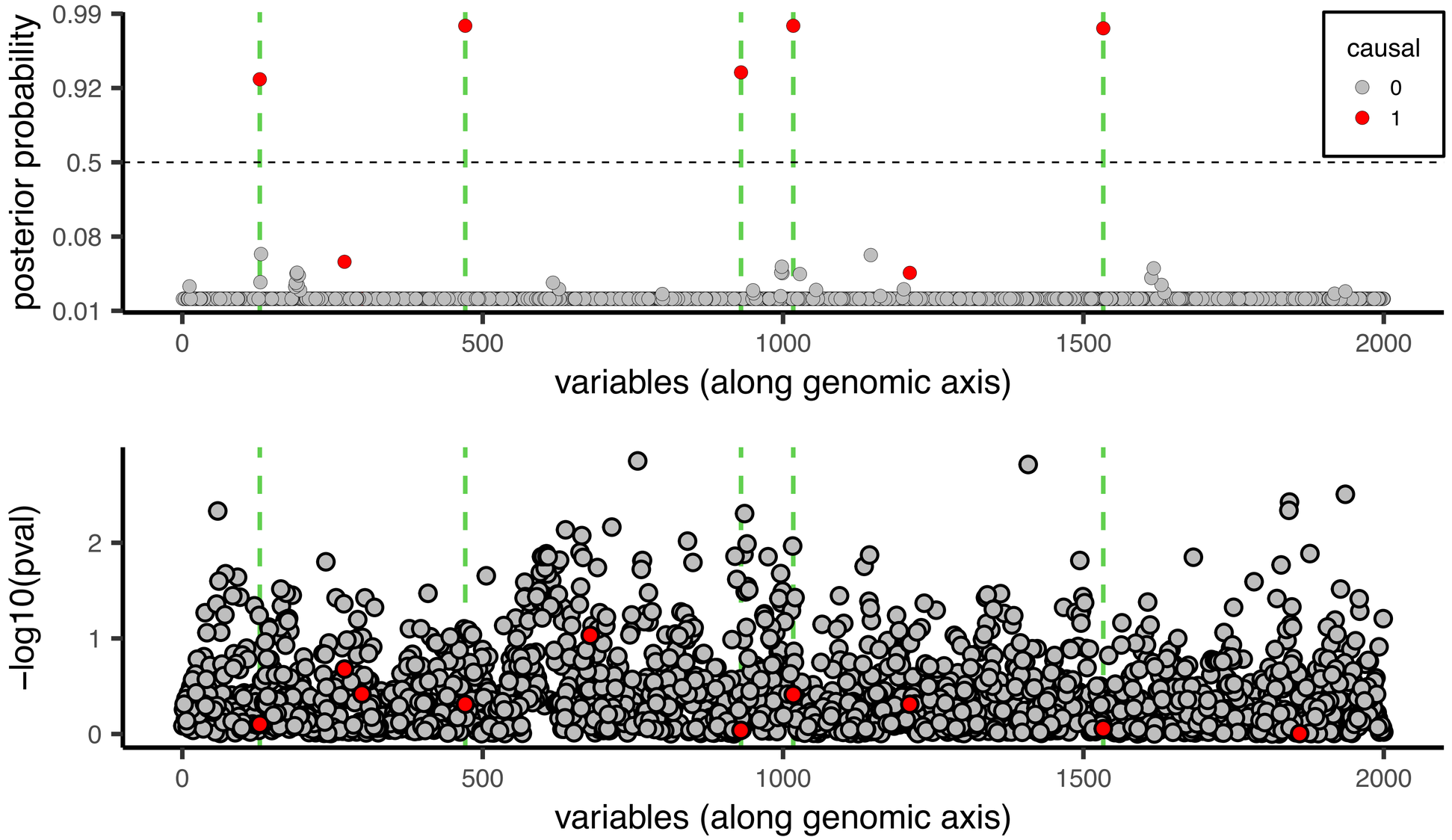
SuSiE identifies top causal variants



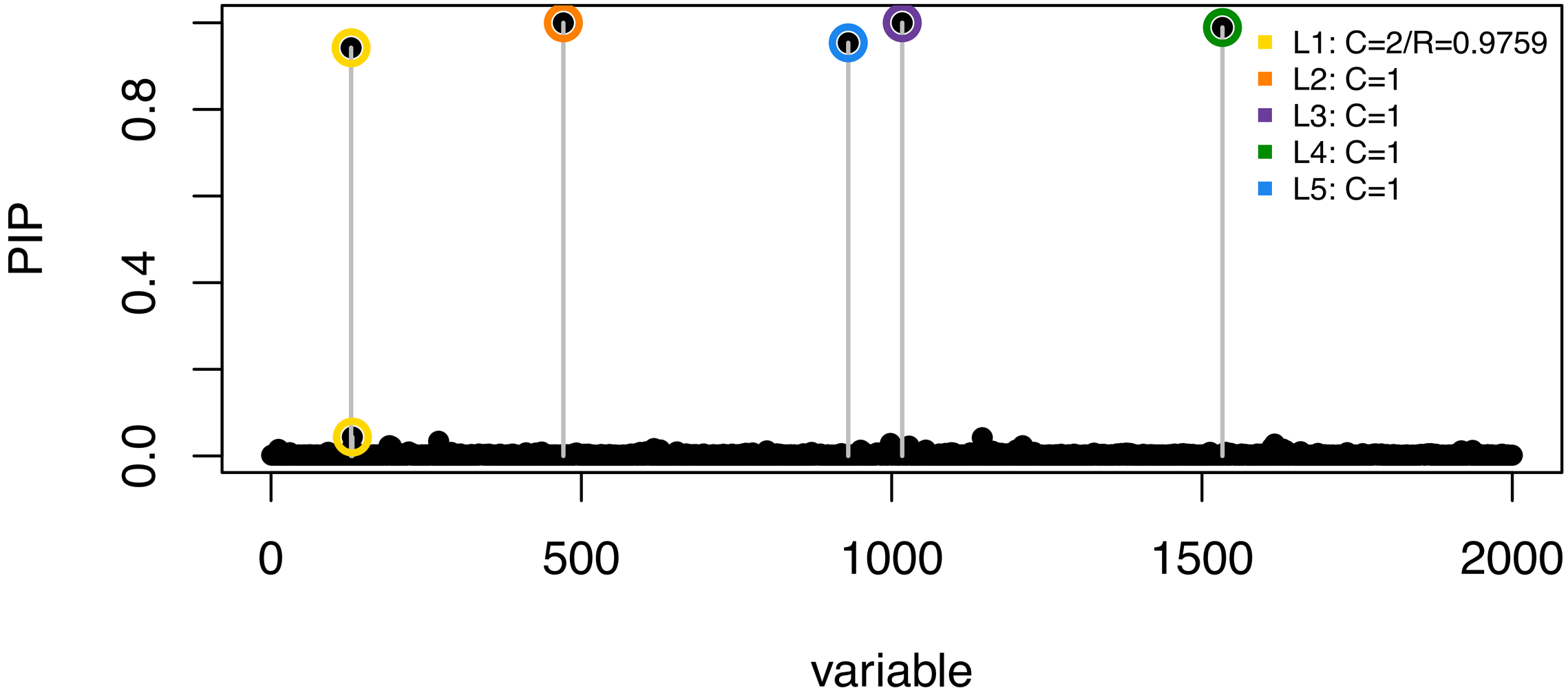
SuSiE can avoid the col-linearity problem



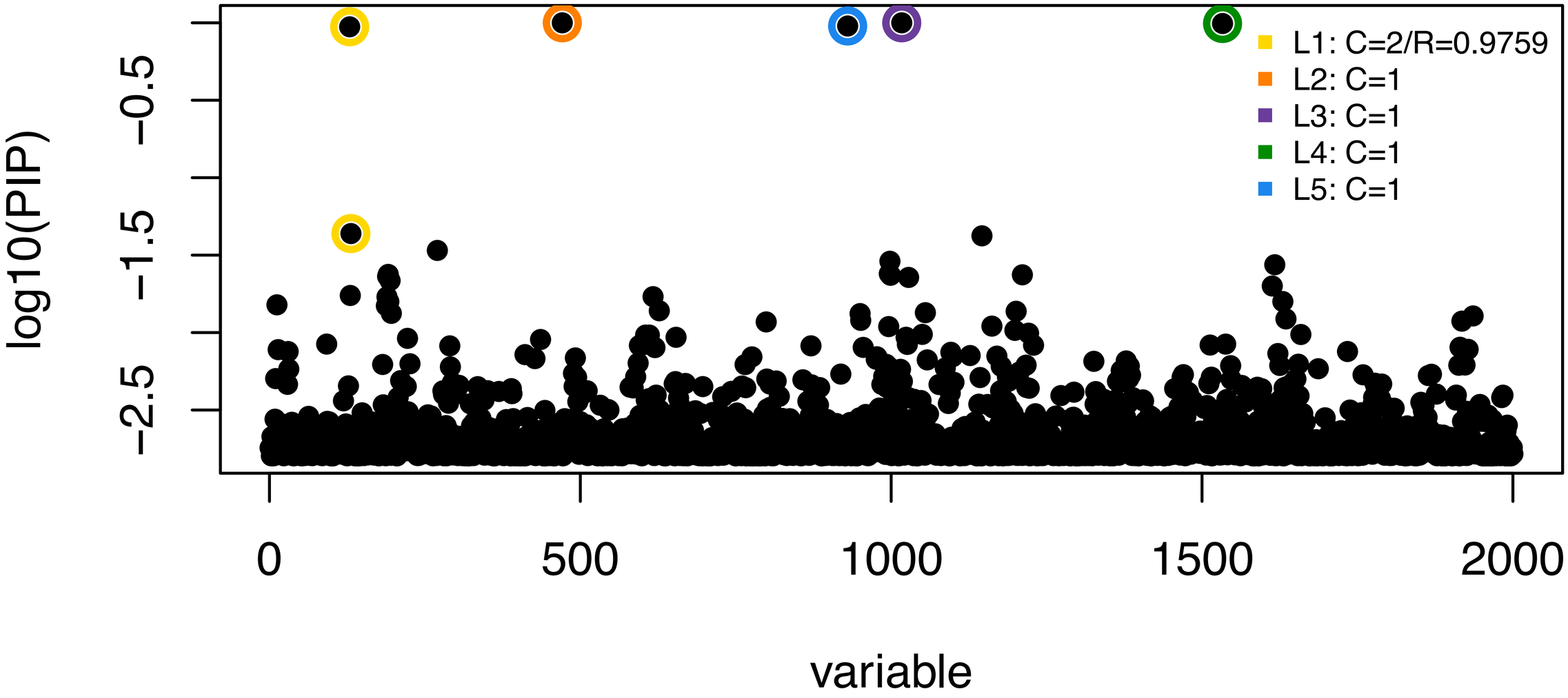
SuSiE can avoid the col-linearity problem



SuSiE can avoid the col-linearity problem



SuSiE can avoid the col-linearity problem



Today's lecture: Supervised Learning in Genomics

- **Non-parametric prediction methods**
  - When we don't have any idea of data-generation mechanisms.
  - Kernel (local) regression
  - Gaussian Process
- **Ensemble learning: the collective power of weak learners**
  - Expectation maximization
  - Mixture of linear regressions
- **Variable selection**
  - Challenges in high-dimensional prediction problems
  - Sparse regression models