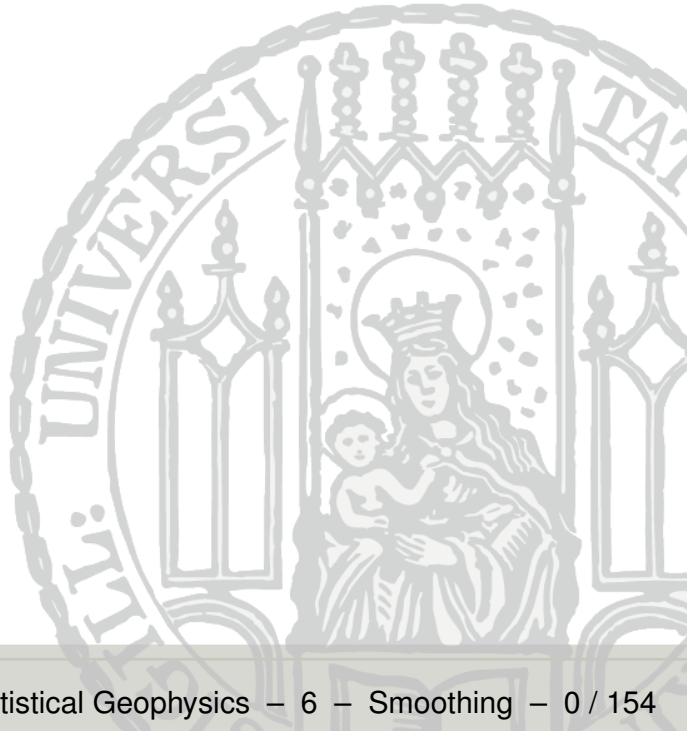
LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Statistical Geophysics

Chapter 6

Smoothing



Smoothing

Univariate Smoothing

Univariate Smoothing

This Section discusses the problem of estimating an unknown functional relationship between some response y and a continous covariate z. The representation of smooth functions is best introduced considering the following model:

$$y_i = f(z_i) + \varepsilon_i$$

where f is a smooth function and

$$E(\varepsilon_i) = 0 \text{ and } Var(\varepsilon_i) = \sigma^2.$$

Here we assume that the errors are

$$\varepsilon_i \; i.i.d. \; N(0, \sigma^2).$$

Univariate Smoothing

Recall some assumptions about the linear regression model:

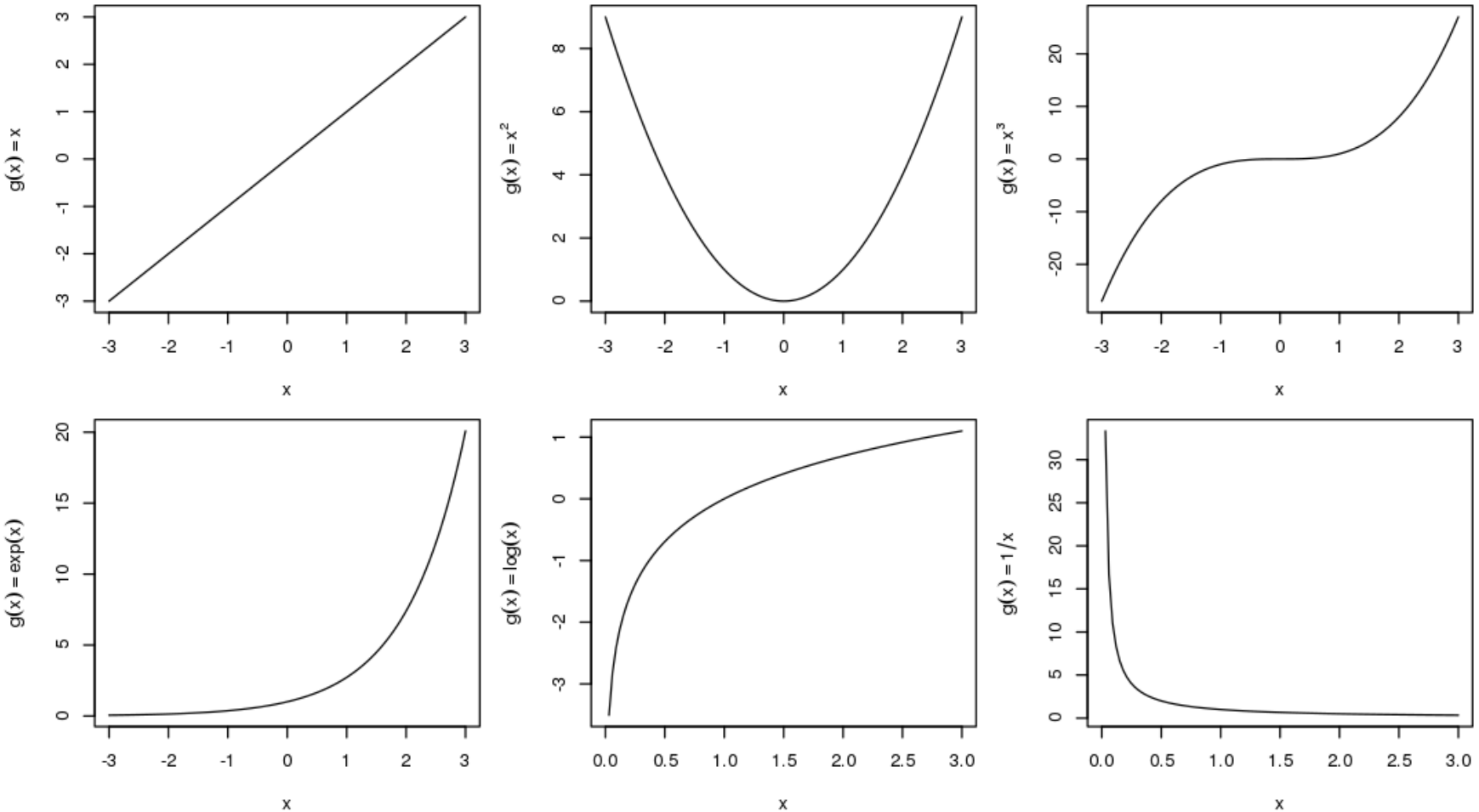
1. $E(y|\mathbf{x}) = \mathbf{x}^\top \boldsymbol{\beta}$;
2. $Var(y|\mathbf{x})$ is constant;
3. $\varepsilon = y - E(y|\mathbf{x})$ is normally distributed; and
4. ε_i and ε_j are uncorrelated if $i \neq j$.

Transformations can be used to meet one or more of assumptions 1 - 3. Unfortunately, data transformation cannot help if the data are correlated.

Transforming either the covariates or the response can help meet assumption 1, but only transformation of the response affects the distribution of the response and can help meet assumption 2 - 3.

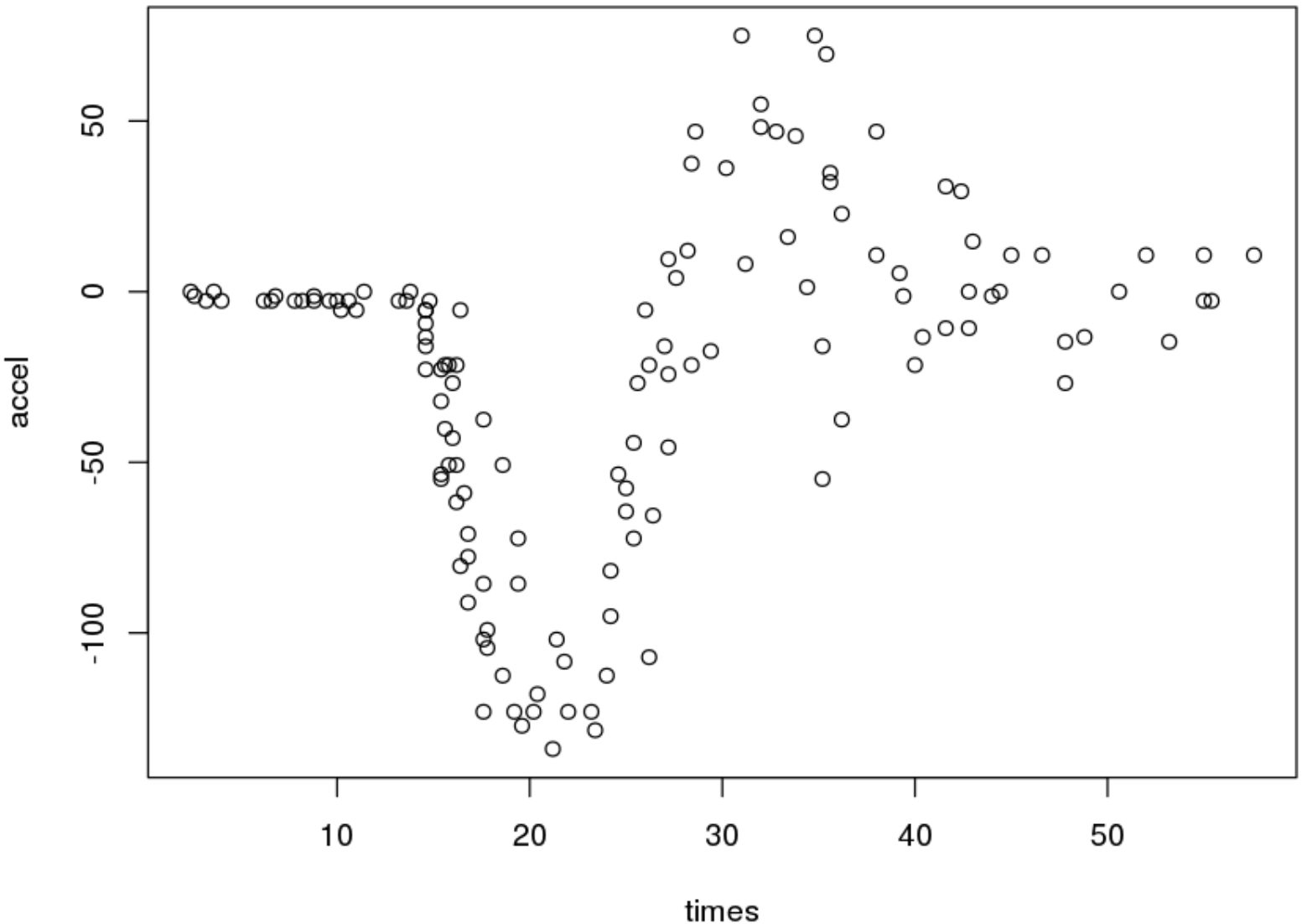
In the following, we will concentrate on problems concerning assumption 1.

Univariate Smoothing



Commonly used transformations of covariates.

Univariate Smoothing



Univariate Smoothing

Polynom Splines (Regression Splines)

- If a linear fit is too simple we could use a polynomial model

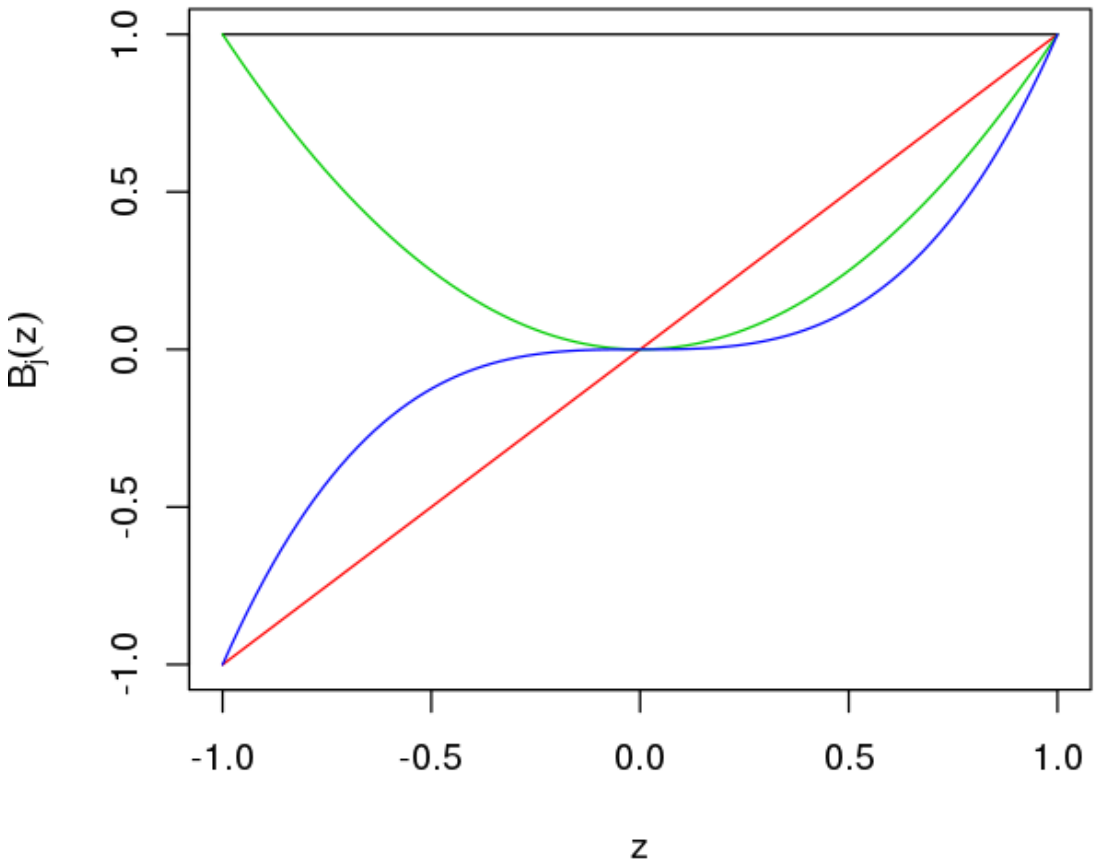
$$\texttt{accel}_i = \gamma_0 + \gamma_1 \texttt{times}_i + \ldots + \gamma_l \texttt{times}_i^l.$$

- The parameters can be estimated by ordinary least squares.
- Note that we write γ instead of β to better distinguish between simple linear and nonlinear effects here.
- The design matrix has the following form

$$\mathbf{Z} = \begin{pmatrix} 1 & \texttt{times}_1 & \ldots & \texttt{times}_1^l \\ \vdots & \vdots & \vdots & \vdots \\ 1 & \texttt{times}_n & \ldots & \texttt{times}_n^l \end{pmatrix}$$

Univariate Smoothing

- The columns of Z are also called basis functions $B_j(z)$, $j = 0, \ldots, l$. In this case a polynomial basis.
- With sorted z they have a nice visual representation.



Univariate Smoothing

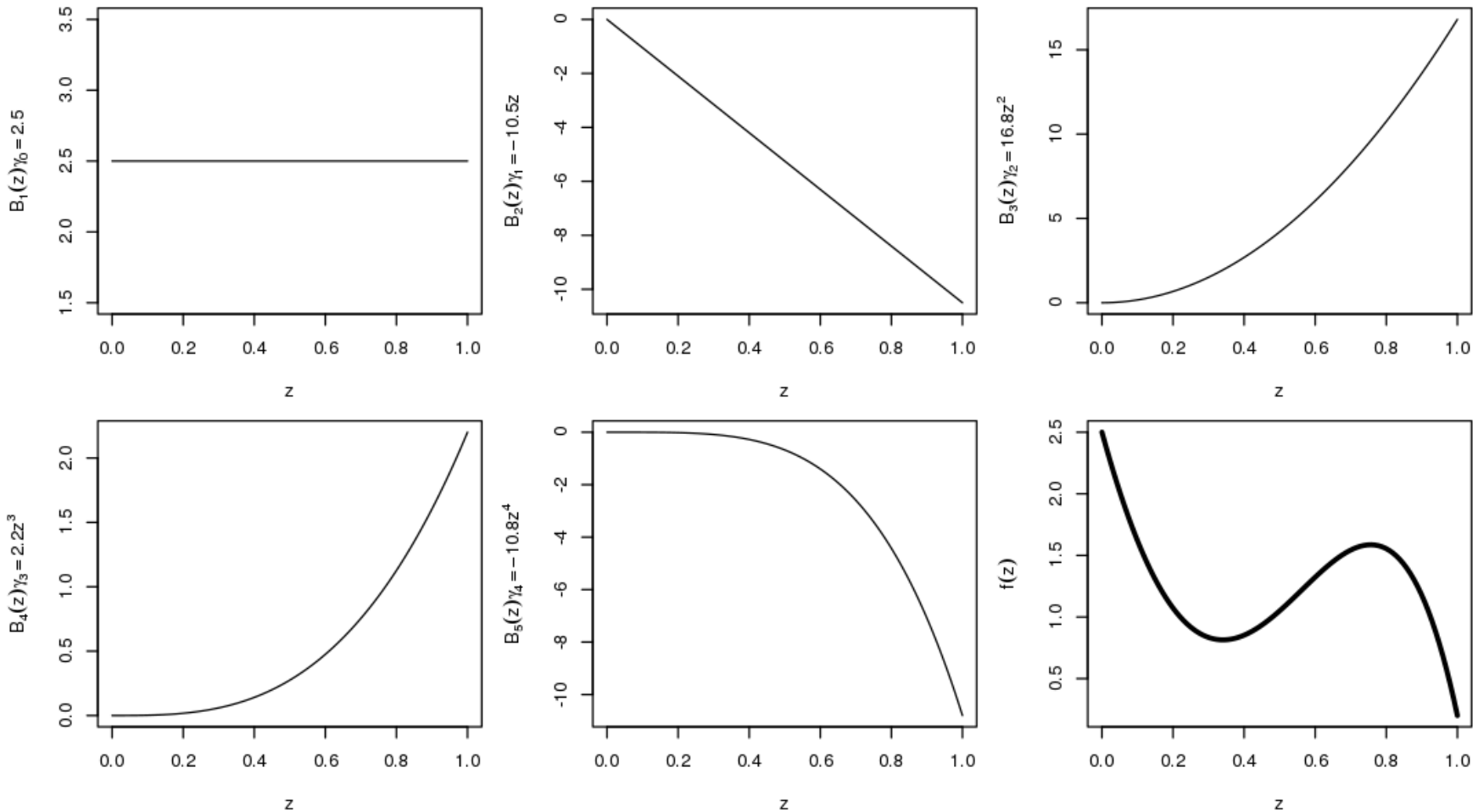
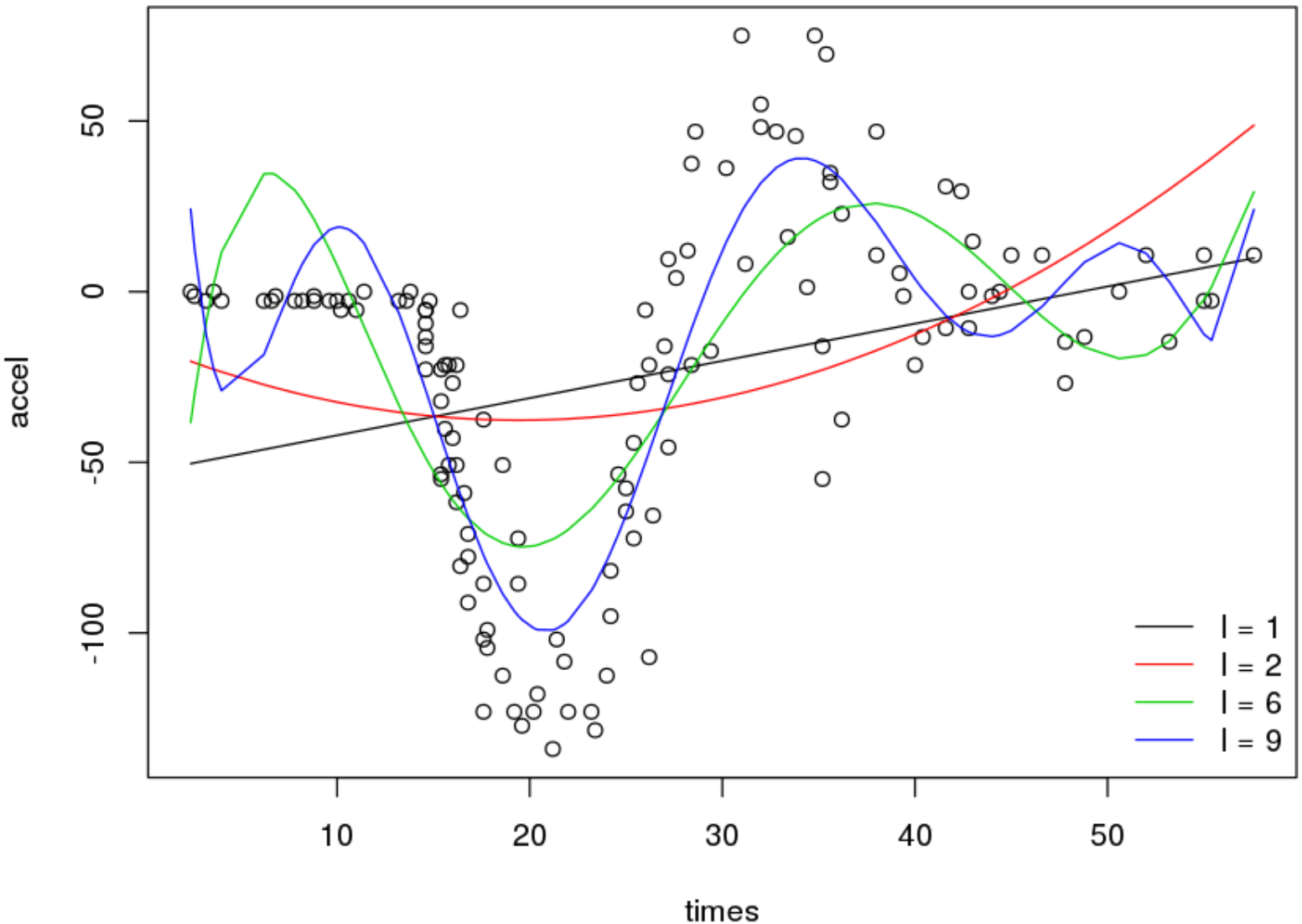
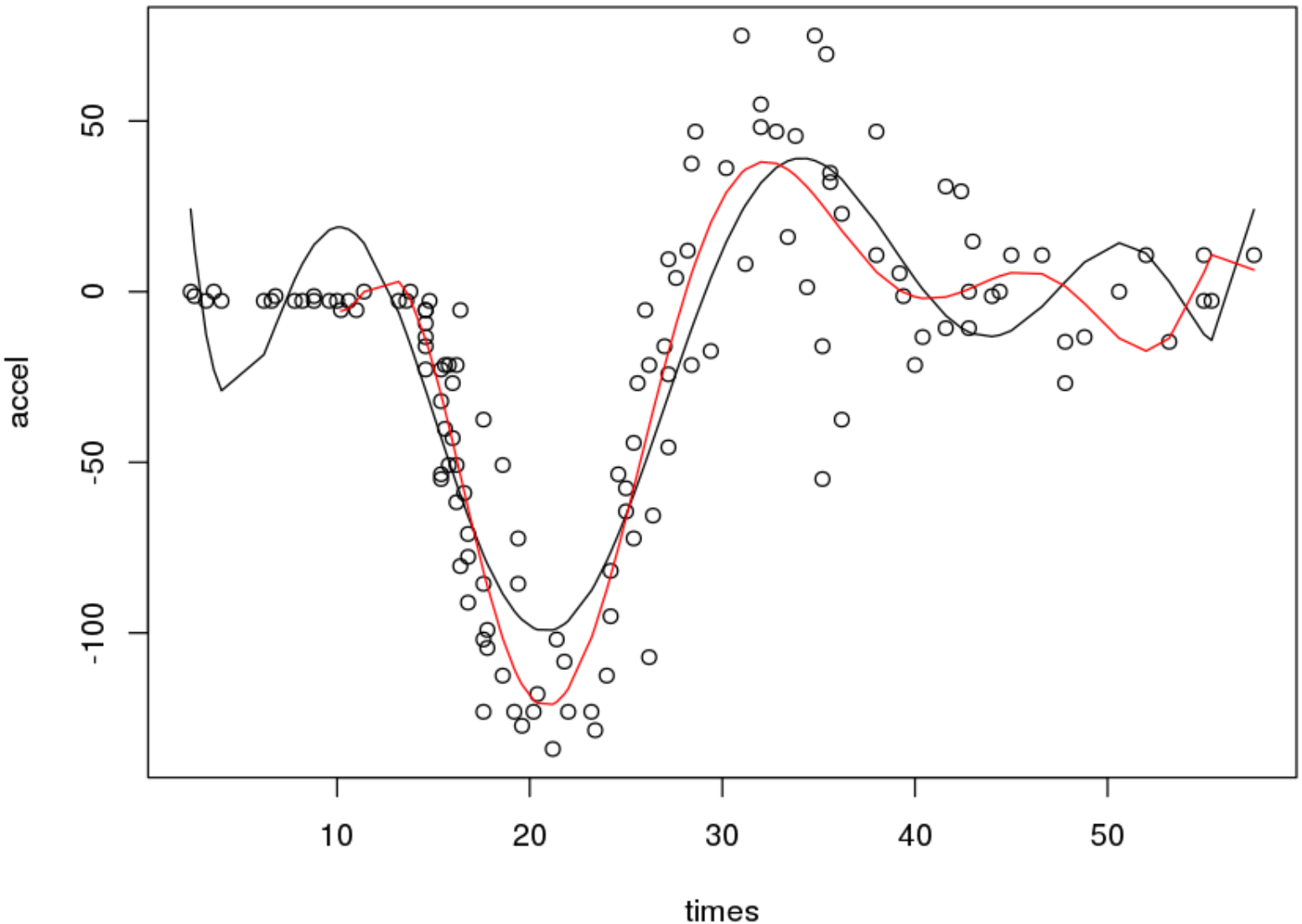


Illustration of how $f(z)$ is represented in terms of $B_j(z)\gamma_j$.

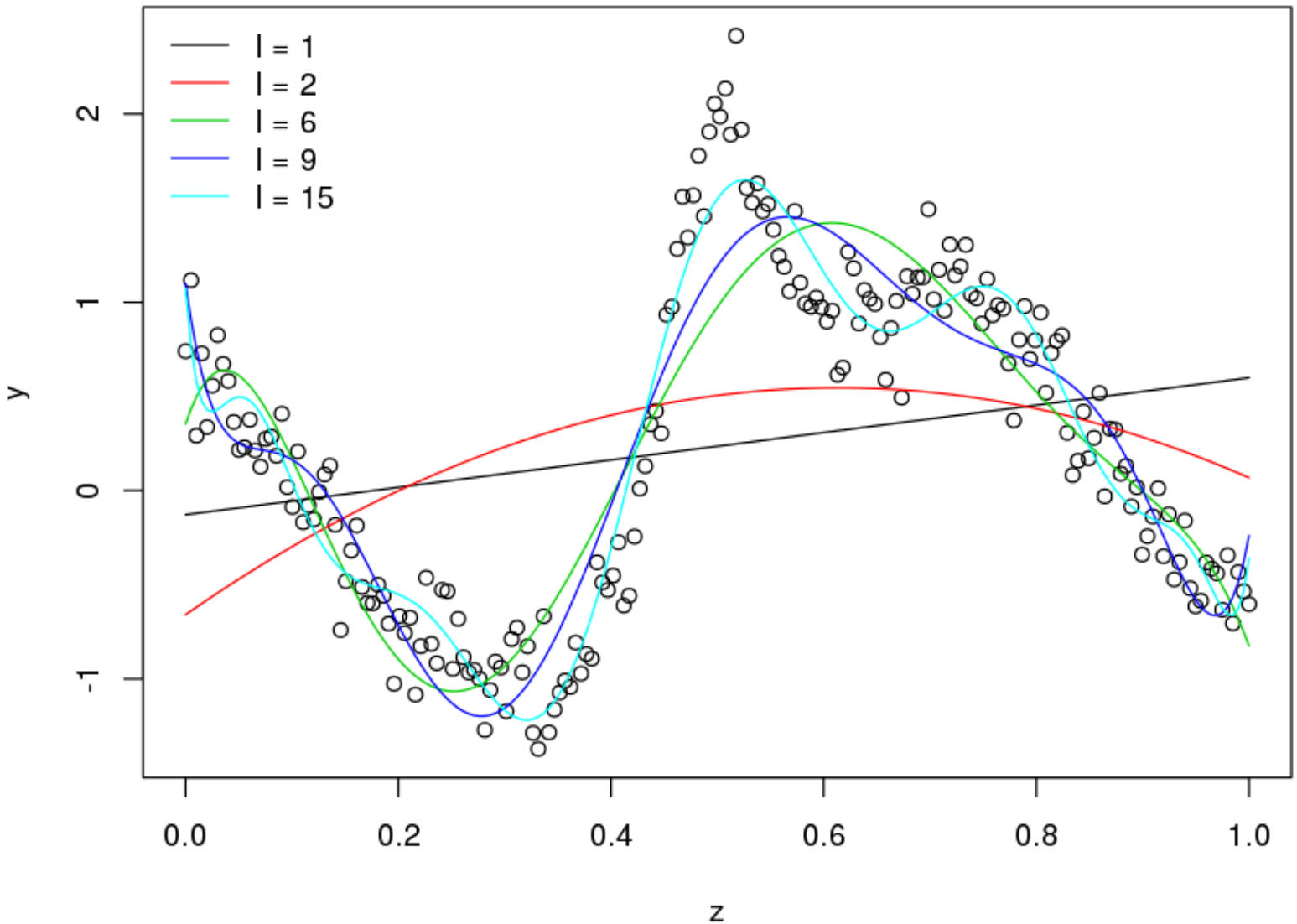
Univariate Smoothing



Univariate Smoothing



Univariate Smoothing



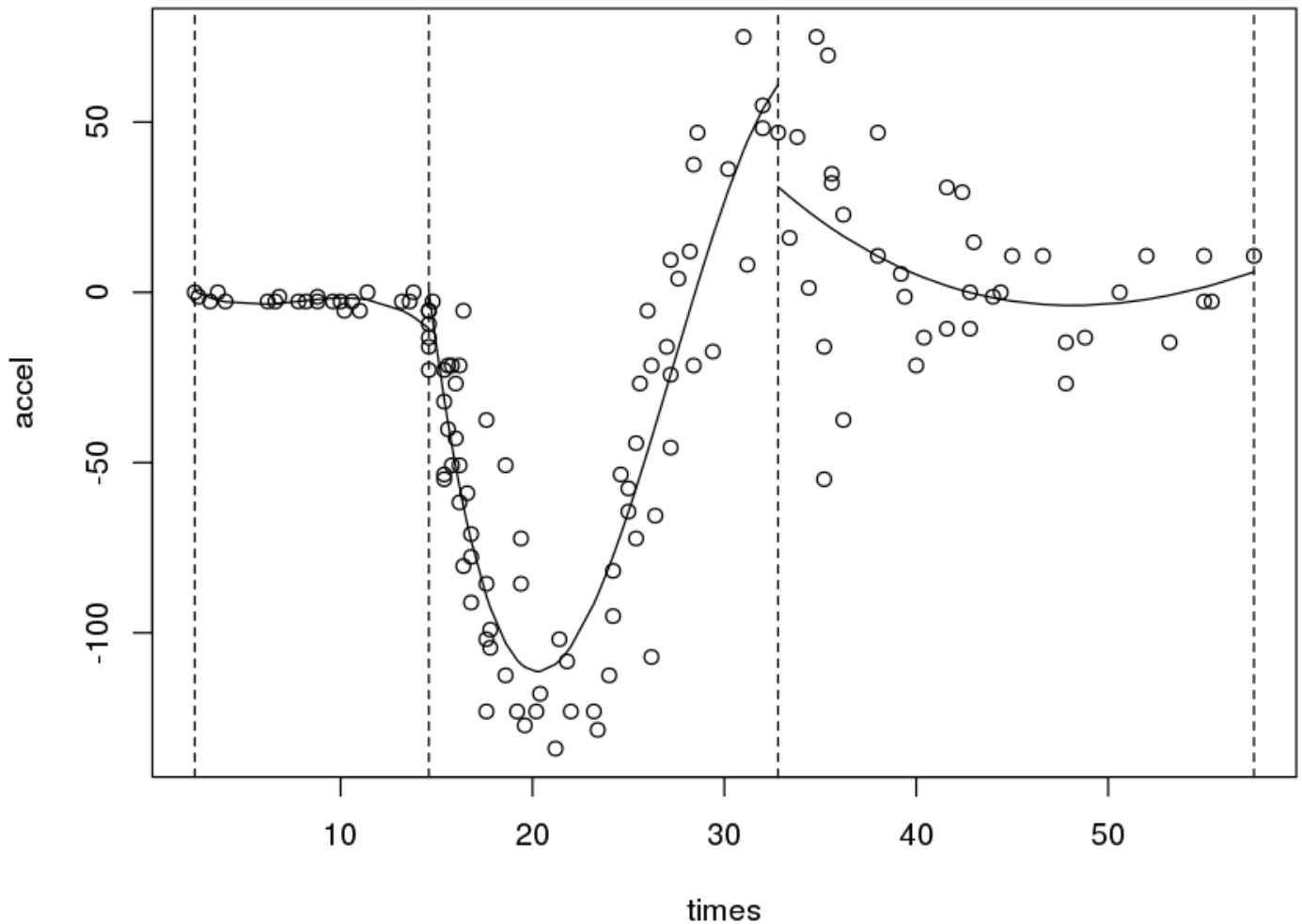
Univariate Smoothing

Problems

- High degree needed for decent curve fit.
- Higher degree polynomials are numerically unstable.
- Basis funtions are global.
- Unexpected wiggles.
- Round-off problems with $\hat{\gamma} = (\mathbf{Z}^\top \mathbf{Z})^{-1} \mathbf{Z}^\top \mathbf{y}$.
- Partial remedy: center and normalize z.
- Better use orthogonal polynomials instead, see also function `poly()`.

Univariate Smoothing

Divide the range of z in equidistant intervals with boundaries $\boldsymbol{\kappa}$ (knots) and fit polynomial models within each section.



Univariate Smoothing

- The fitted functions don't form a nice overall smooth function, see the jumps at the boundaries.
- We need additional requirements to construct a smooth functional form.

A function f is called polynomial spline of degree $l \geq 0$ with knots $min(z) = \kappa_1 < \ldots < \kappa_m = max(z)$ (the interval boundaries), if it satisfies

1. $f(z)$ is $(l-1)$ times continuously differentiable,
2. $f(z)$ is a polynom of degree l in each interval $[\kappa_j, \kappa_{j+1})$.

Every spline may be represented by a linear combination of basis functions, i.e.

$$f(z_i) = \gamma_1 \cdot B_1(z_i) + \gamma_2 \cdot B_2(z_i) + \ldots + \gamma_{l+m-1} \cdot B_{l+m-1}(z_i).$$

Univariate Smoothing

Polynom splines with truncated powers

- $$y_i = \gamma_1 + \gamma_2 z_i + \ldots + \gamma_{l+1} z_i^l + \sum_{j=2}^{m-1} \gamma_{l+j} (z_i - \kappa_j)_+^l + \varepsilon_i.$$

  where

  $$(z_i - \kappa_j)_+^l = \begin{cases} (z_i - \kappa_j)^l & z_i \geq \kappa_j \\ 0 & \text{else.} \end{cases}$$

- Corresponding basis functions

  $$B_1(z_i) = 1, \quad B_2(z_i) = z_i, \quad \ldots \quad , \quad B_{l+1}(z_i) = z_i^l,$$
  $$B_{l+2}(z_i) = (z_i - \kappa_2)_+^l, \quad \ldots \quad , \quad B_k(z_i) = (z_i - \kappa_{m-1})_+^l.$$

Univariate Smoothing

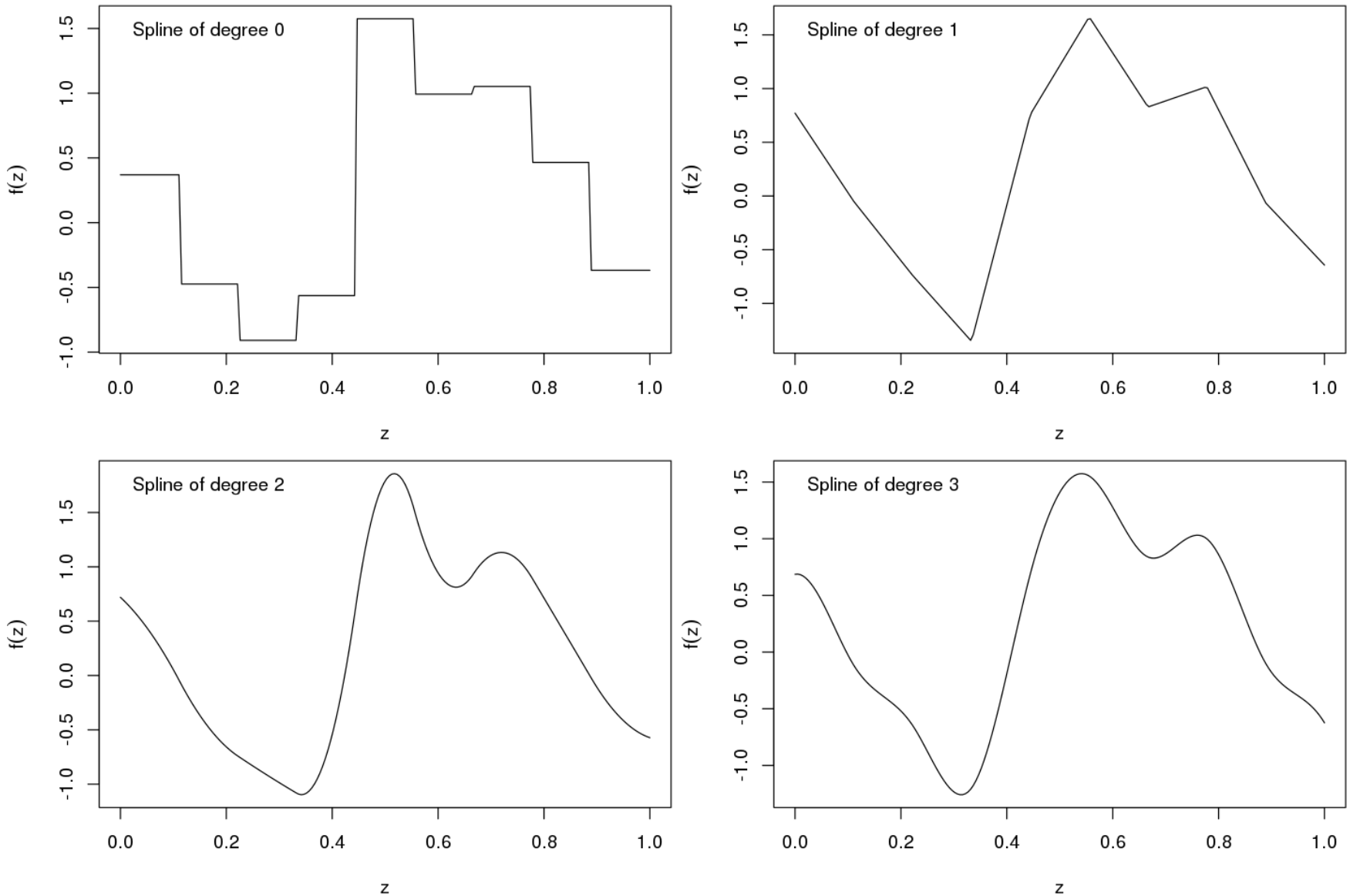
- $$y_i = f(z_i) + \varepsilon_i = \sum_{j=1}^{k} \gamma_j B_j(z_i) + \varepsilon_i.$$

- The corresponding design matrix is

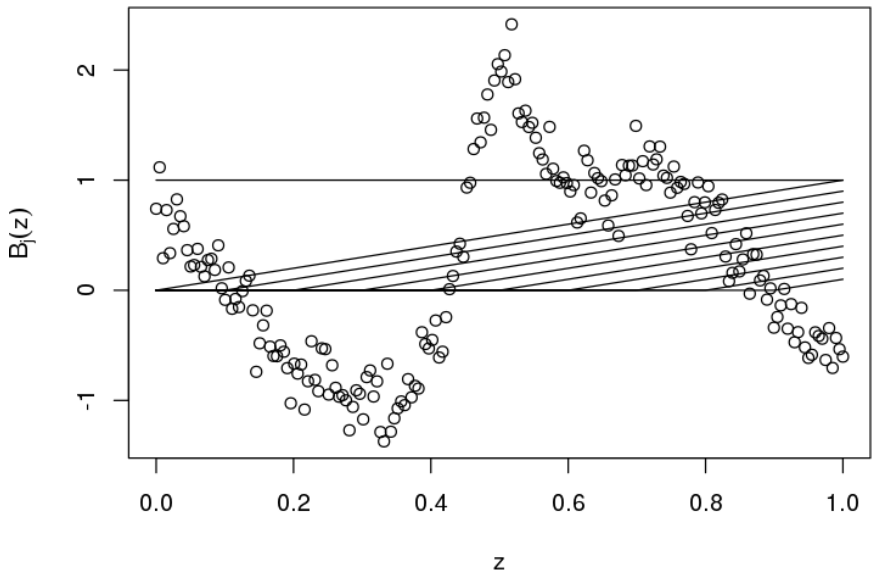
$$\begin{aligned}
\mathbf{Z} &= \begin{pmatrix} B_1(z_1) & \dots & B_k(z_1) \\ \vdots & & \vdots \\ B_1(z_n) & \dots & B_k(z_n) \end{pmatrix} \\
&= \begin{pmatrix} 1 & z_1 & \dots & z_1^l & (z_1 - \kappa_2)_+^l & \dots & (z_1 - \kappa_{m-1})_+^l \\ \vdots & & & & & & \vdots \\ 1 & z_n & \dots & z_n^l & (z_n - \kappa_2)_+^l & \dots & (z_n - \kappa_{m-1})_+^l \end{pmatrix},
\end{aligned}$$

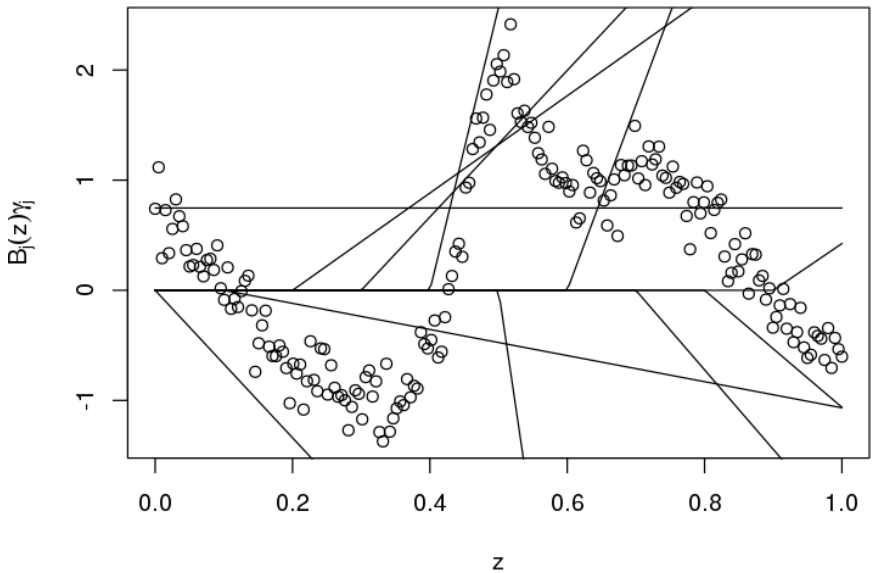
- $$\mathbf{y} = \mathbf{Z}\boldsymbol{\gamma} + \boldsymbol{\varepsilon} \text{ and } \hat{\boldsymbol{\gamma}} = (\mathbf{Z}^\top \mathbf{Z})^{-1} \mathbf{Z}^\top \mathbf{y}.$$

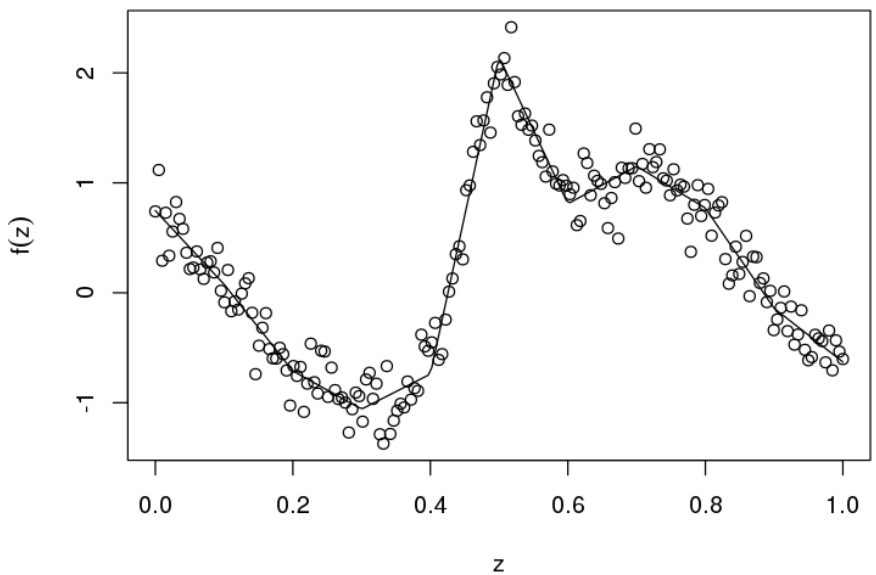
Univariate Smoothing



Univariate Smoothing







Univariate Smoothing

Knot Selection

But how many knots should be used and where should we place them? We could do the following:

1. Equidistant knots. Compute $h = (max(\mathbf{z}) - min(\mathbf{z}))/(m-1)$, then the knots are

$$\kappa_j = min(\mathbf{z}) + (j-1) \cdot h, \quad j = 1, \ldots, m.$$

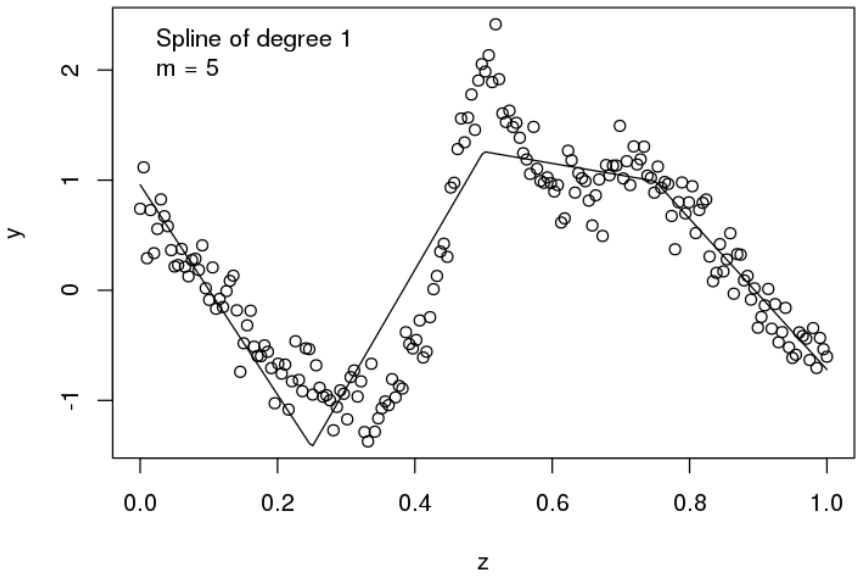
2. Quantile based knots. Choose the

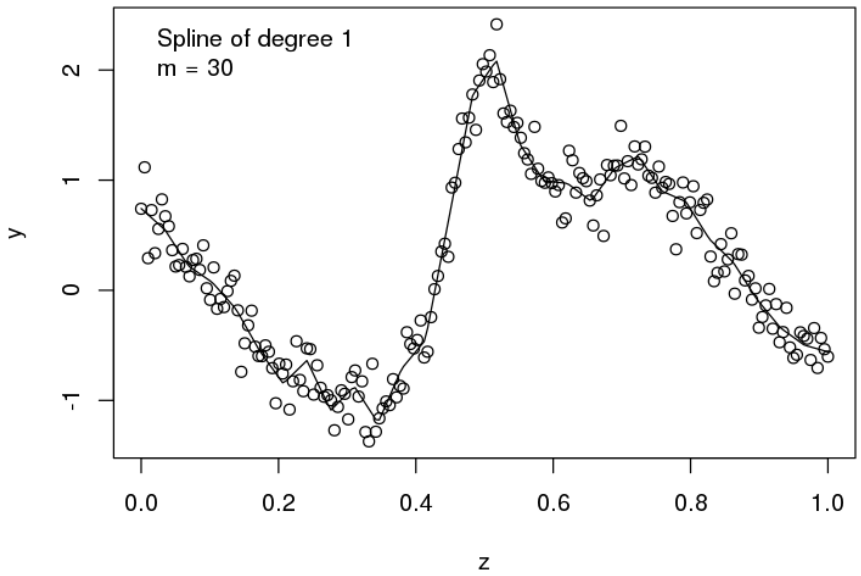
$$(j-1)/(m-1)\text{-quantiles}, \quad j = 1, \ldots, m,$$

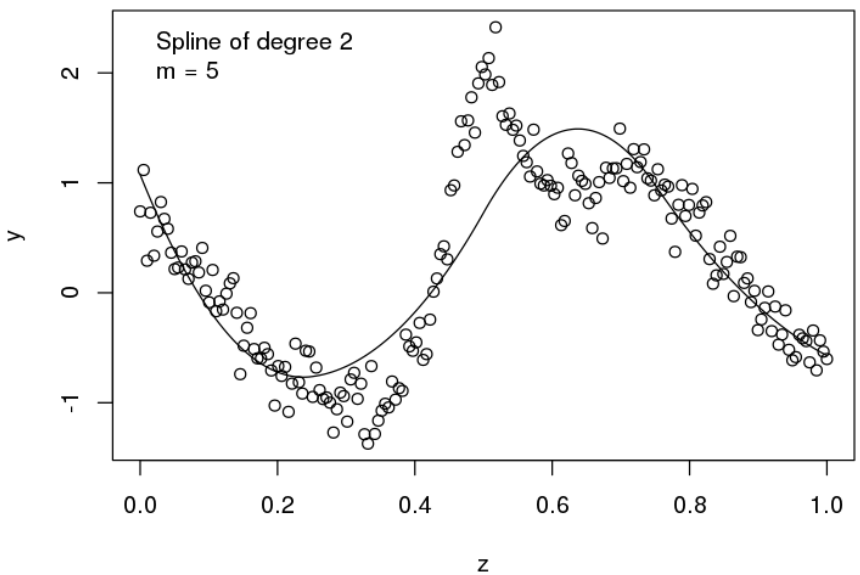
   of observations $z_1, \ldots, z_n$.

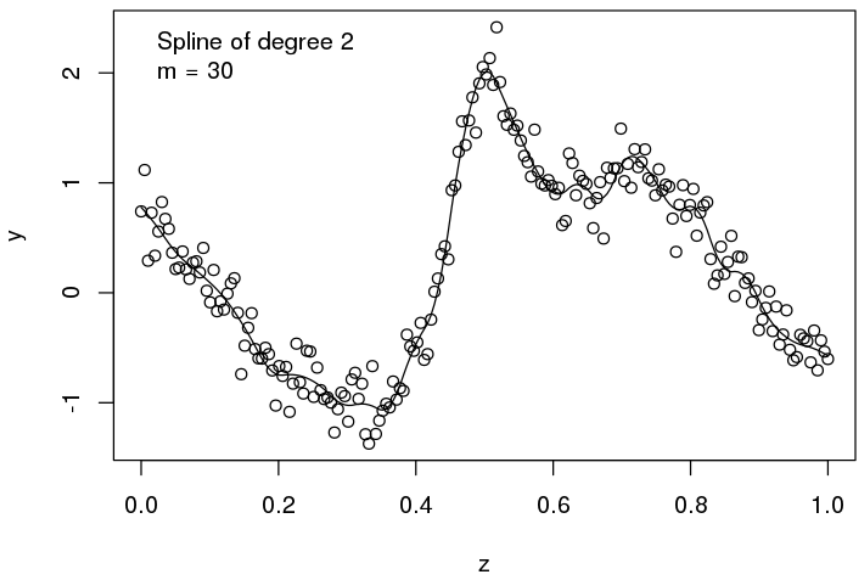
3. Visual inspection.

Univariate Smoothing









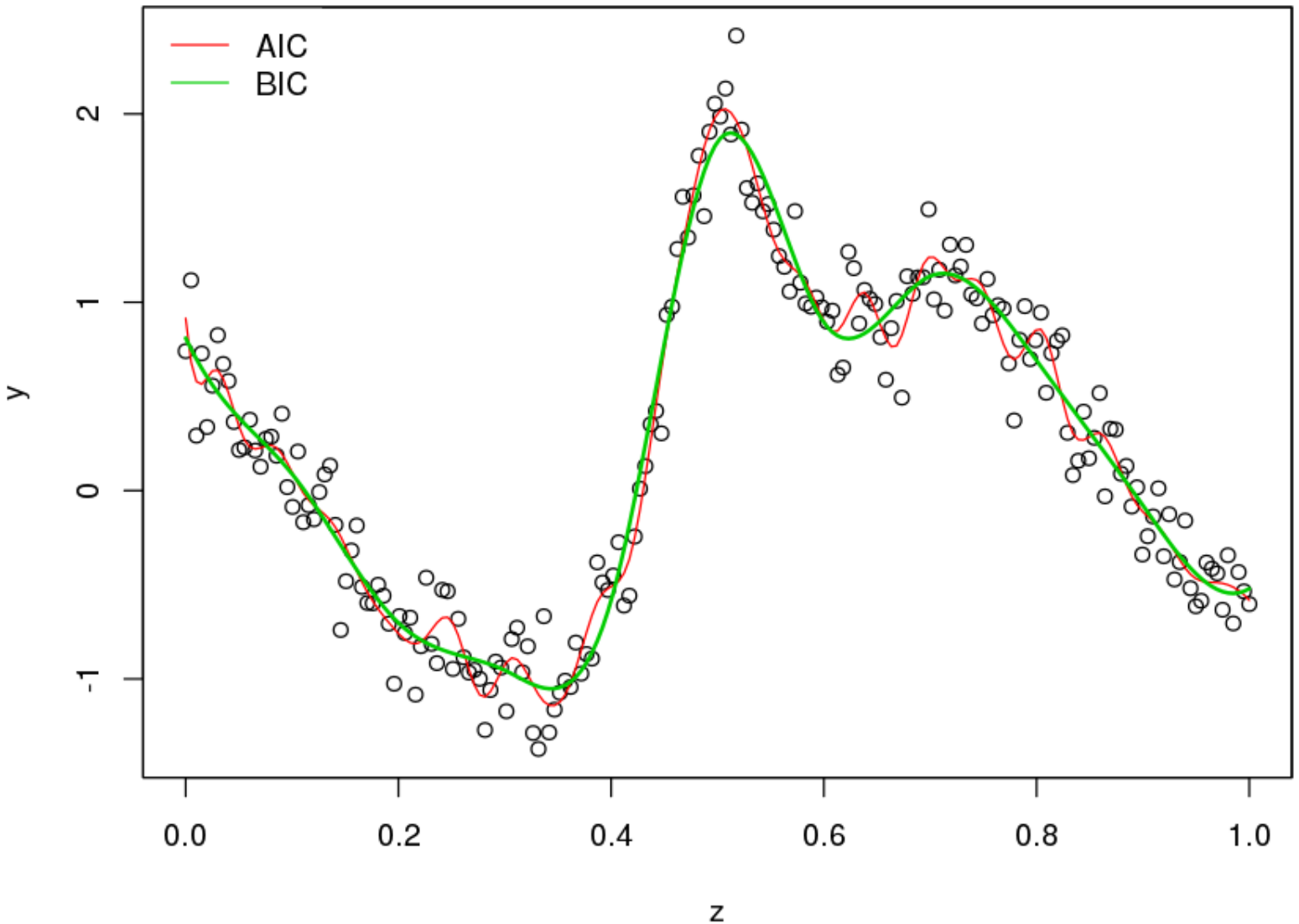
Univariate Smoothing

Automatic Selection – Knot Search Algorithm

A first naive way for selecting the number of equidistant intervals would be a simple grid search, i.e. for a certain maximum number of knots m, choose the optimum number of knots $\alpha \in \{2, \ldots, m\}$ that minimzes e.g. the AIC.

```
R> search.nknots <- function(y, z, IC = AIC, m = 50, ...) {
+    knots <- 2:m; ic <- NULL
+    for(i in knots)
+      ic <- c(ic, IC(lm(y ~ -1 + tp(z, knots = i, ...))))
+    return(knots[which.min(ic)])
+  }

R> search.nknots(y, z, IC = AIC)

[1] 37

R> search.nknots(y, z, IC = BIC)

[1] 11
```

Univariate Smoothing



Univariate Smoothing

Automatic Selection – MARS

Forward

- Create a full design matrix Z.
  1. Start with the smallest possible model, i.e. start with the first column of Z (intercept).
  2. Successively add basis functions $B_j(z_i)$ (columns) that have not been used, one at a time.
  3. Choose the basis function which, e.g., minimizes the RSS the most to be added to the model.
  4. Calculate an information criterion from the current model.
  5. Repeat 1 – 4 until all basis functions are included in the model.

Univariate Smoothing

Backward

- Take the full design matrix Z.
  1. Successively remove basis functions $B_j(z_i)$ (columns) that have not been removed yet, one at a time.
  2. Choose the basis function which, e.g., maximizes the RSS the most to be removed from the model.
  3. Calculate an information criterion from the current model.
  4. Repeat 1 – 3 until one basis function is left in the model.

Finally, choose the set of basis functions that minimizes the information criterion obtained from the forward and backward runs.

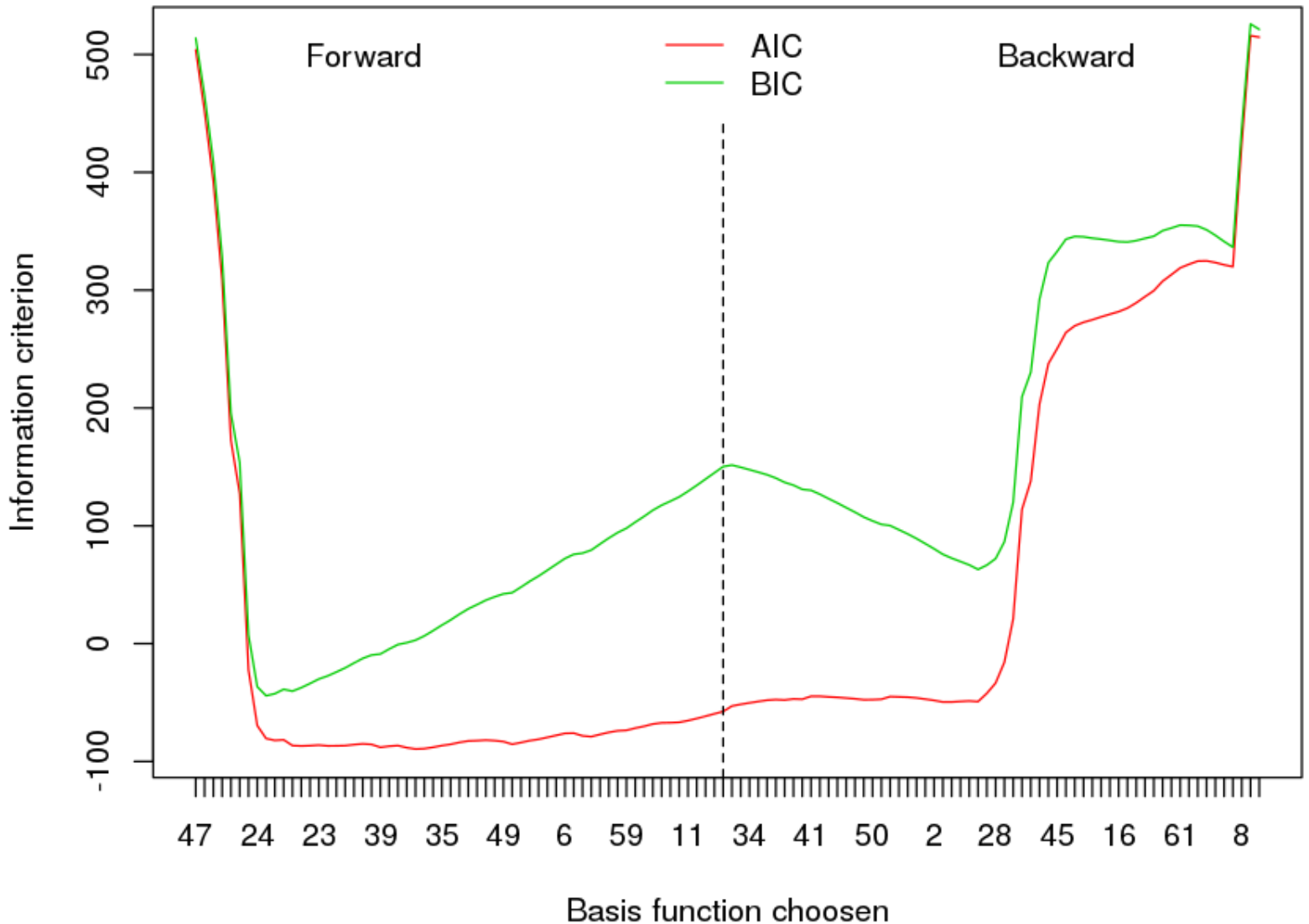
Univariate Smoothing

```
R> mars <- function(y, z, IC = AIC, m = 60, ...) {
+    Z <- tp(z, knots = m, ...)
+    id <- 2:ncol(Z)
+    icf <- icb <- taken <- NULL
+    while(!all(id %in% taken)) {
+      nottaken <- id[!(id %in% taken)]
+      rss <- NULL
+      for(k in nottaken) {
+        cid <- c(1, taken, k)
+        e <- residuals(lm(y ~ -1 + Z[, cid]))
+        rss <- c(rss, crossprod(e))
+      }
+      taken <- c(taken, nottaken[which.min(rss)])
+      icf <- c(icf, IC(lm(y ~ -1 + Z[, c(1, taken)])))
+    }
+
+    ## continued on next page ...
```

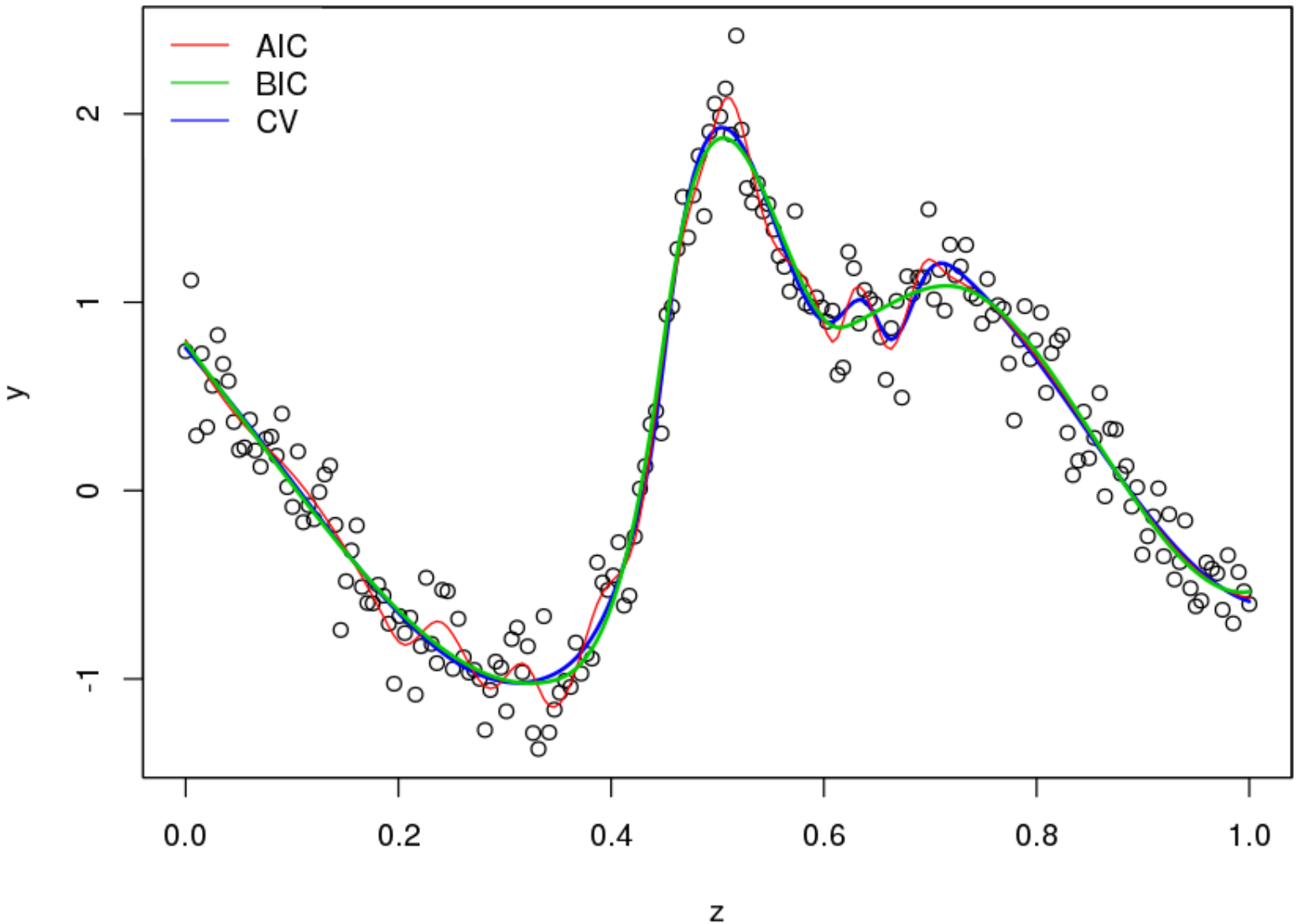
Univariate Smoothing

```
+     ## ... continued from previous page
+
+     dropped <- NULL
+     while(!all(id %in% dropped)) {
+       nottaken <- id[!(id %in% dropped)]
+       rss <- NULL
+       for(k in nottaken) {
+         cid <- c(1, id[!(id %in% c(dropped, k))])
+         e <- residuals(lm(y ~ -1 + Z[, cid]))
+         rss <- c(rss, crossprod(e))
+       }
+       dropped <- c(dropped, nottaken[which.max(rss)])
+       icb <- c(icb,
+         IC(lm(y ~ -1 + Z[, c(1, id[!(id %in% dropped)])])))
+     }
+     Z <- if(min(icf) < min(icb)) {
+       Z[, c(1, taken[1:which.min(icf)])]
+     } else Z[, c(1, id[!(id %in% dropped[1:which.min(icb)])])]
+     return(list("Z" = Z, "ic" = c(icf, icb),
+       "selected" = c(taken, dropped)))
+   }
```

Univariate Smoothing



Univariate Smoothing

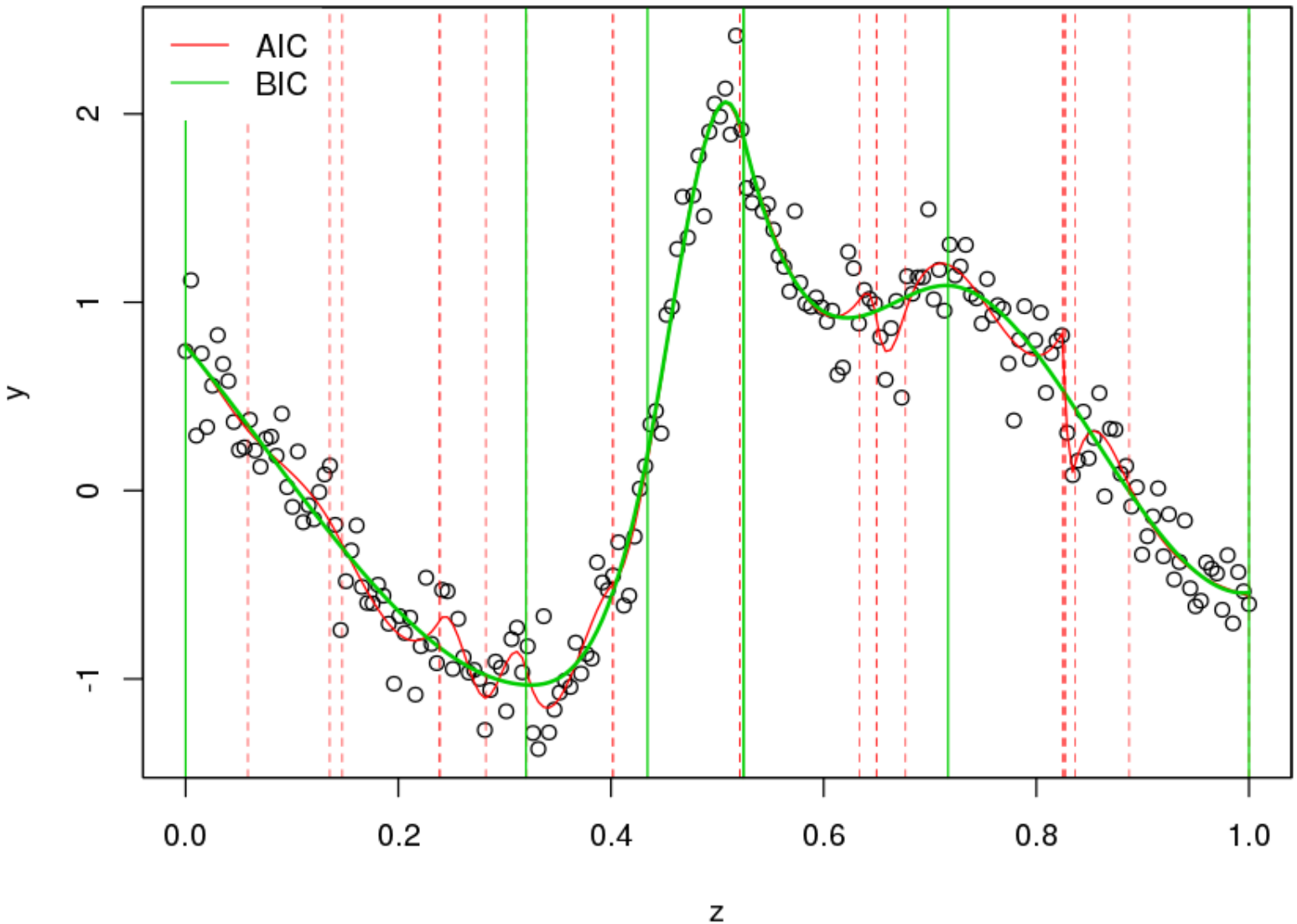


Univariate Smoothing

Automatic Selection – Free Knots Algorithm

```
R> search.knots <- function(y, z, IC = AIC, m = 30, ...) {
+    objfun <- function(knots, y, z, IC, ...) {
+      IC(lm(y ~ -1 + tp(z, knots = knots, ...)))
+    }
+    ic <- NULL; knots <- list(); zr <- range(z)
+    for(i in 2:m) {
+      kn <- seq(zr[1], zr[2], length = i)
+      par <- optim(kn, fn = objfun, y = y, z = z, IC = IC,
+        method = "L-BFGS-B", lower = zr[1], upper = zr[2])
+      ic <- c(ic, par$value); knots[[i - 1]] <- par$par
+    }
+    return(knots[[which.min(ic)]])
+  }

R> knotsAIC <- search.knots(y, z, IC = AIC)
R> knotsBIC <- search.knots(y, z, IC = BIC)

R> c(length(knotsAIC), length(knotsBIC))

[1] 23  7
```

Univariate Smoothing



Univariate Smoothing

Penalized Regression

- Although most of the automatic knot selection procedures have exhibited good performance, they are usually quite complicated and computational intensive.
- We therefore seek a simpler method for flexible spline-based regression.
- As mentioned before, the roughness of a fit is due to there being too many knots in the model.
- Another way to overcome this problem is to retain all of the knots but to constrain their influence.
- The hope is that this will result in a less variable fit.

Univariate Smoothing

Penalized Regression with TP-Splines

- Consider the truncated polynomial model

$$f(z_i) = \gamma_1 + \gamma_2 z_i + \ldots + \gamma_{l+1} z_i^l + \sum_{j=2}^{m-1} \gamma_{l+j} (z_i - \kappa_j)_+^l .$$

- The wiggliness of the fit is mainly the result of too large variability of the coefficients of the truncated bases.
- Constraints on the γ_{l+j} that might rectify this situation are
  1. $max|\gamma_{l+j}| < C$,
  2. $\sum |\gamma_{l+j}| < C$, and
  3. $\sum \gamma_{l+j}^2 < C$.
- Each of these will lead to a smoother fit, however, the third constraint is much easier to implement.

Univariate Smoothing

- Define the $((m-2)+l) \times ((m-2)+l)$ matrix

$$\mathbf{K} = \begin{pmatrix} \mathbf{0}_{l \times l} & \mathbf{0}_{l \times (m-2)} \\ \mathbf{0}_{(m-2) \times l} & \mathbf{I}_{(m-2) \times (m-2)} \end{pmatrix},$$

  then our minimization problem can be written as

$$min||\mathbf{y} - \mathbf{Z}\boldsymbol{\gamma}||^2 \text{ subject to } \boldsymbol{\gamma}^\top \mathbf{K} \boldsymbol{\gamma} < C.$$

- Using a Lagrange multiplier argument, it can be shown that this is equivalent to choosing γ to minimize

$$||\mathbf{y} - \mathbf{Z}\boldsymbol{\gamma}||^2 + \lambda \boldsymbol{\gamma}^\top \mathbf{K} \boldsymbol{\gamma}$$

  for some $\lambda \geq 0$.

Univariate Smoothing

- The solution is then given by

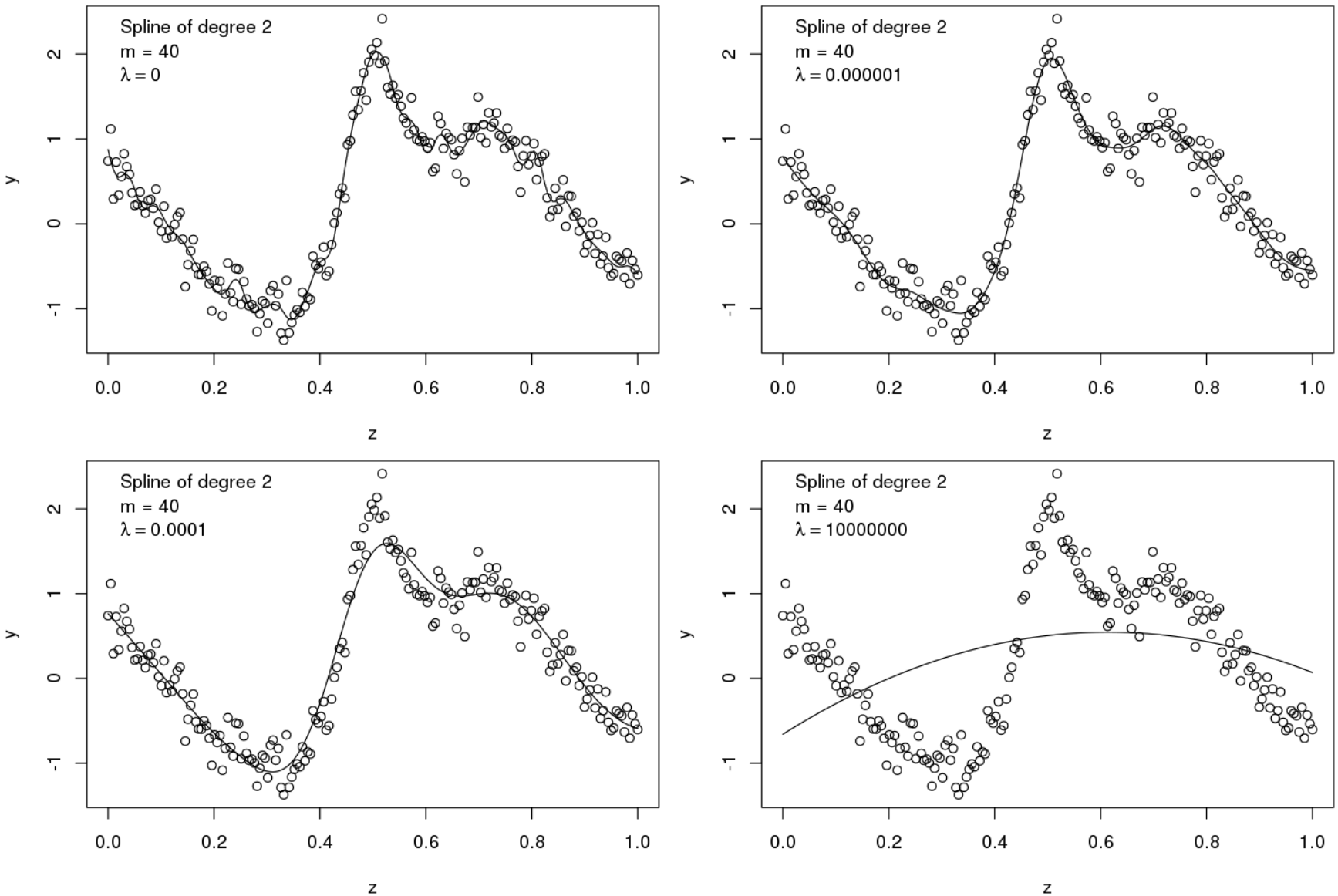
$$\hat{\gamma} = \left(\mathbf{Z}^\top \mathbf{Z} + \lambda \mathbf{K}\right)^{-1} \mathbf{Z}^\top \mathbf{y}.$$

- The fitted values for a penalized spline regression are

$$\hat{\mathbf{y}} = \mathbf{Z}\left(\mathbf{Z}^\top \mathbf{Z} + \lambda \mathbf{K}\right)^{-1} \mathbf{Z}^\top \mathbf{y} = \mathbf{S}_\lambda \mathbf{y},$$

  where $\mathbf{S}_\lambda$ is called smoother matrix.

Univariate Smoothing

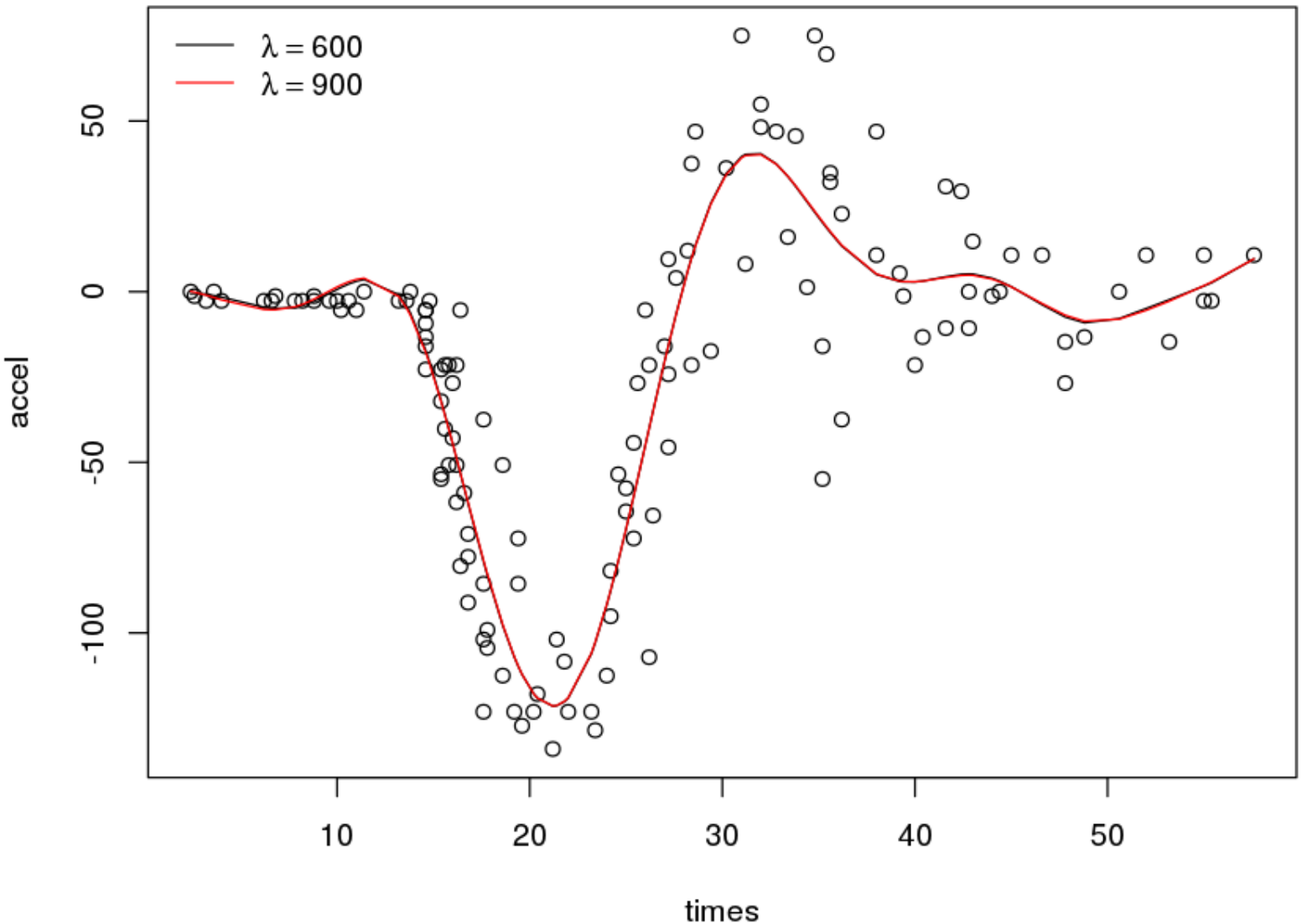


Univariate Smoothing

Degrees of Freedom of a Smoother

- A small value of λ leads to a wiggly scatterplot smooth, whereas a very large λ results in a parametric fit that depends on the degree of the spline basis functions.
- What is not so clear is the relationship between intermediate values of λ and the resulting amount of smoothing.
- The problem is illustrated in the next slide which shows 20-knot cubic penalized spline smooths of the `mcycle` data with $\lambda = 600$ and $\lambda = 900$.
- The second smoothing parameter is 50% larger than the first, but the two fits look almost identical.

Univariate Smoothing



Univariate Smoothing

- Hence, the value of the smoothing parameter does not have a direct interpretation as to how much structure is being imposed to the fit.
- A transformation $t(\lambda)$ of λ that provides a reasonable solution to this problem is

$$t(\lambda) = \mathit{trace}(\mathbf{S}_\lambda),$$

  which is called the *degrees of freedom* of the fit corresponding to λ.
- Since

$$\mathit{trace}(\mathbf{H}) = p = \mathit{df},$$

  so $\mathit{trace}(\mathbf{S}_\lambda)$ is a generalization of this definition for scatterplot smooths.

Univariate Smoothing

- For a penalized spline smooth with m knots and degree l, it is easily shown that

$$trace(\mathbf{S}_0) = l + m - 1.$$

- At the other extreme,

$$trace(\mathbf{S}_\lambda) \to l + 1 \text{ as } \lambda \to \infty.$$

- So positive values of λ correspond to

$$l + 1 < df < l + m - 1.$$

Univariate Smoothing

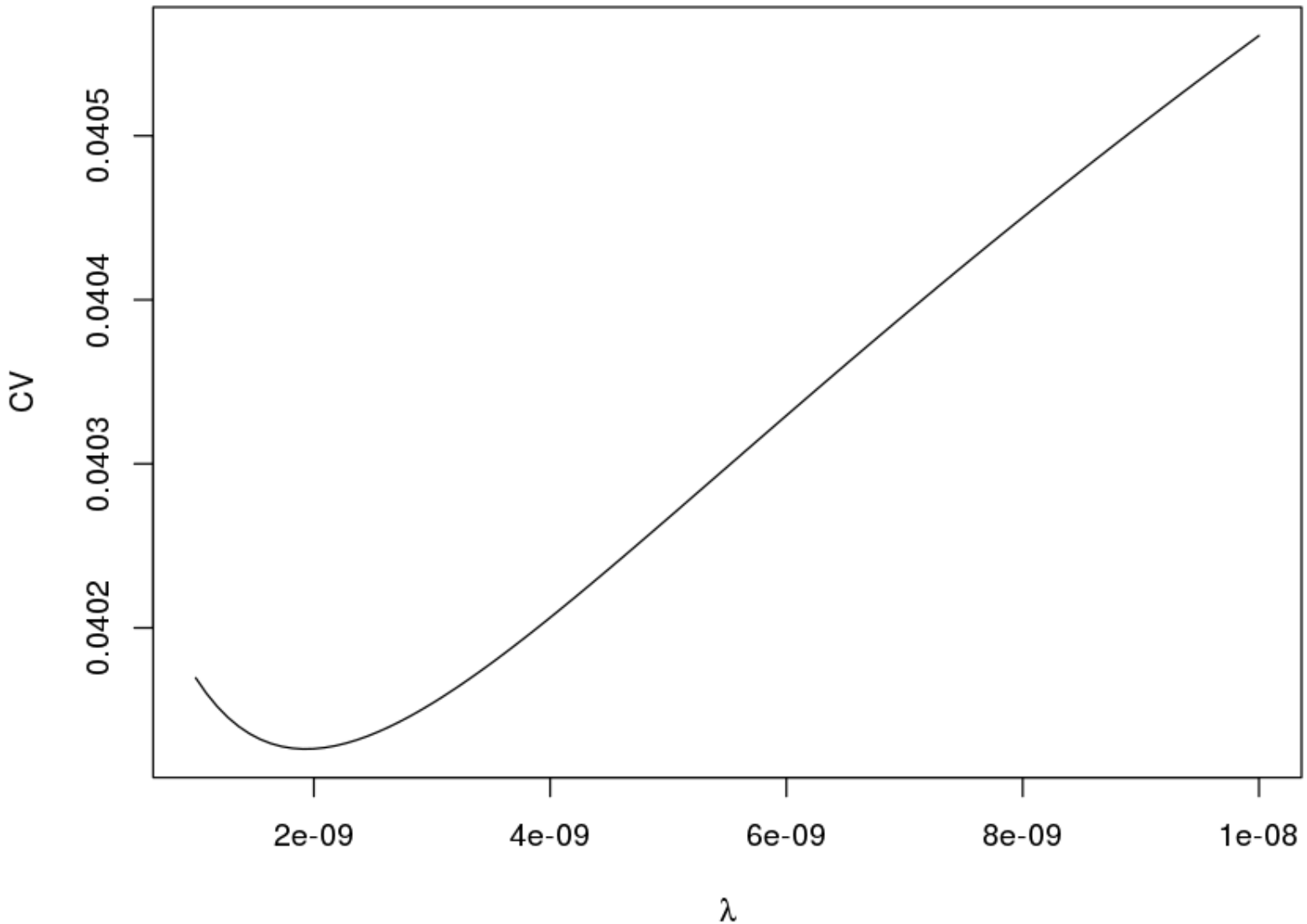
Choosing the Smoothing Parameter

- If λ is too high the data will be over smoothed, and if it is too low then the data will be under smoothed.
- In both cases, this means that the spline estimate $\hat{f}$ will not be close to the true function.
- Therefore, choose the smoothing parameter λ, e.g. by *cross validation*.
- For univariate regression models this could be done by a simple grid search.

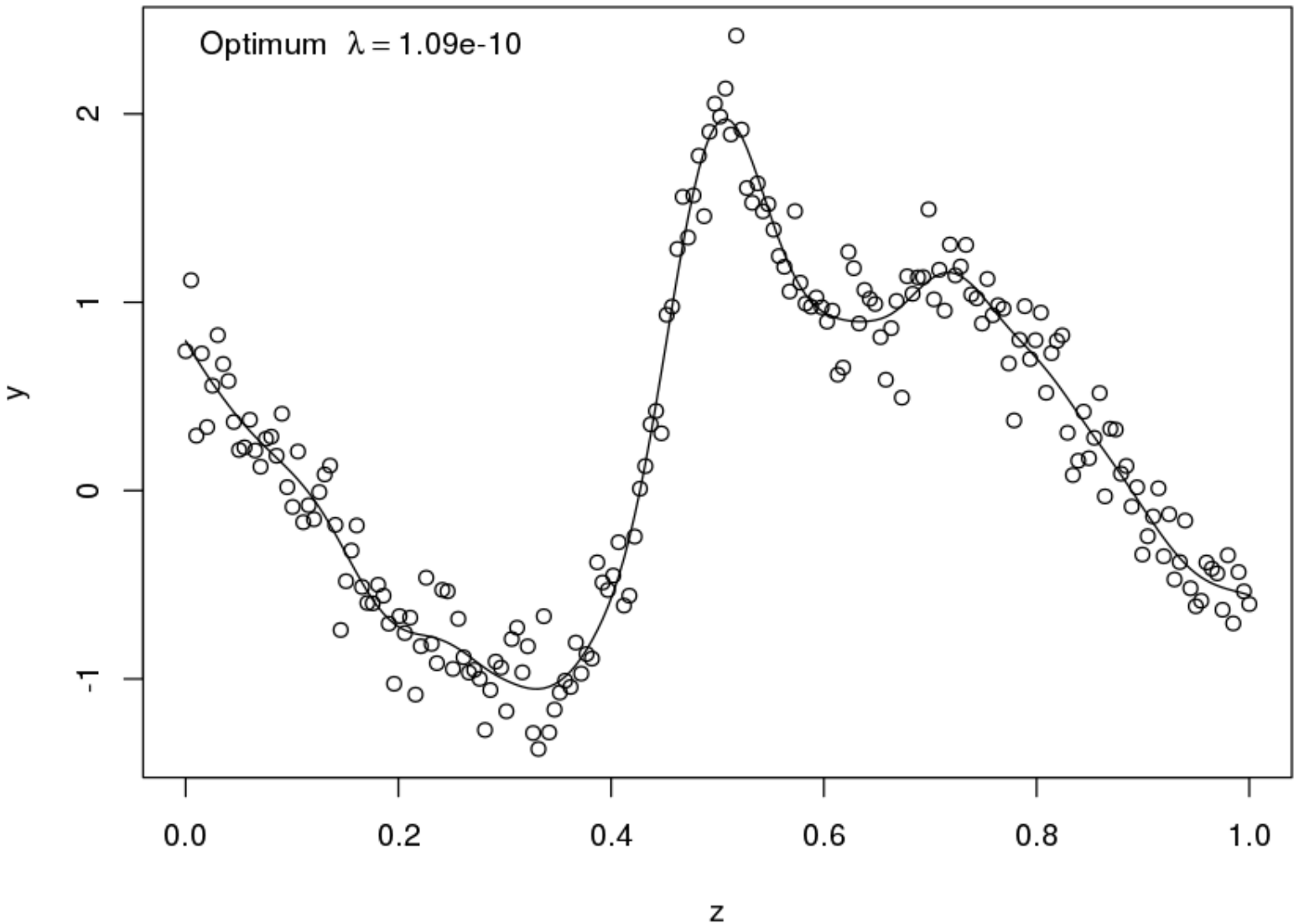
Univariate Smoothing

```
R> degree <- 3
R> Z <- tp(z, degree = degree, knots = 40)
R> K <- diag(c(rep(0, degree + 1), rep(1, ncol(Z) - degree - 1)))

R> lambdas <- seq(1e-9, 1e-8, length = 100)
R> cv <- NULL; n <- length(y)

R> for(i in lambdas) {
+    S <- Z %*% solve(t(Z) %*% Z + i * K) %*% t(Z)
+    yhat <-  S %*% y
+    trS <- sum(diag(S))
+    rss <- sum((y - yhat)^2)
+    cv <- c(cv, n * rss / (n - trS)^2)
+  }
```

Univariate Smoothing



Univariate Smoothing



Univariate Smoothing

P-Splines

- Truncated power bases can sometimes lead to numerical instability when there is a large number of knots and the smoothing parameter is small.
- Therefore, in practical use it is advisable to work with equivalent bases with more stable numerical properties.
- The most common choice is the B-spline basis.
- B-spline basis can represent cubic splines (and also higher or lower orders).
- The advantage of B-splines is that they are strictly local – each basis function is only non-zero over $l + 1$ adjacent knots.

Univariate Smoothing

- To define a B-spline with k basis functions we need to set up $m + l + 1$ knots

$$\kappa_1 < \kappa_2 < \ldots < \kappa_{m+l+1},$$

  where the interval over which the spline is to be evaluated is $[\kappa_{l+1}, \kappa_k]$, i.e., the first and the last l knot locations are essentially arbitrary.
- Every basis function overlaps with $2l$ neighboring basis functions and is positive over $l + 2$ neighboring knots.
- The B-spline is $l - 1$ times continuously differentiable.
- A lth order B-spline is then represented by

$$f(z_i) = \sum_{j=1}^{k} B_j^l(z_i)\gamma_j.$$

Univariate Smoothing

- The B-spline basis functions are most conveniently defined recursively as follows:

$$B_j^l(z_i) = \frac{z_i - \kappa_j}{\kappa_{j+l} - \kappa_j} B_j^{l-1}(z_i) + \frac{\kappa_{j+l+1} - z_i}{\kappa_{j+l+1} - \kappa_{j+1}} B_{j+1}^{l-1}(z_i),$$

- where

$$B_j^0(z_i) = \begin{cases} 1 & \kappa_j \leq z_i < \kappa_{j+1}, \\ 0 & \text{else.} \end{cases}$$

- A common choice is a cubic spline basis with $l = 3$.

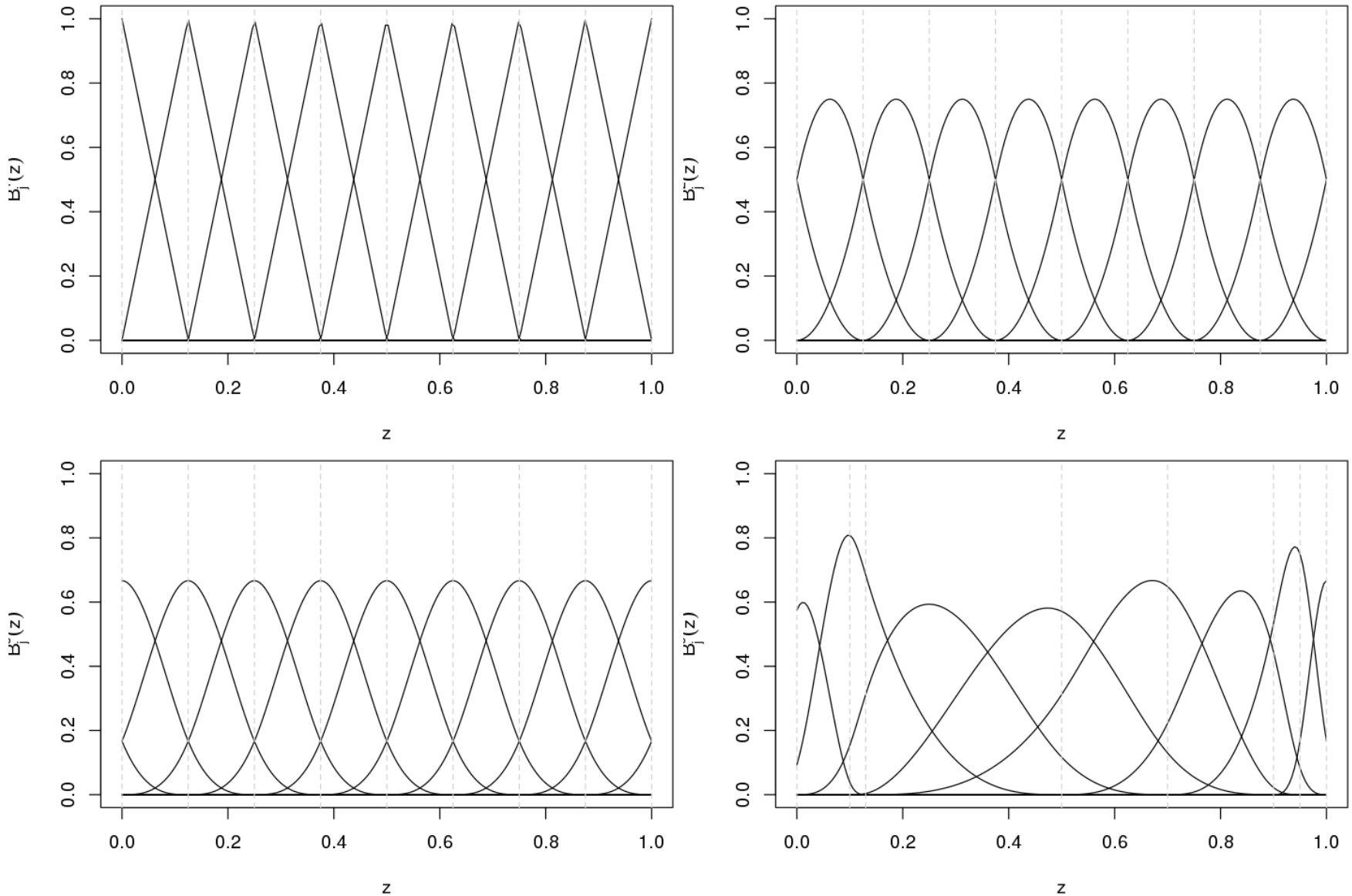
Univariate Smoothing

```
R> bsDesign <- function(z, degree = 3, knots = NULL) {
+
+    ## basis function constructor function
+    bsbasis <- function(z, knots, j, degree) {
+      if(degree == 0)
+        B <- 1 * (knots[j] <= z & z < knots[j + 1])
+      if(degree > 0) {
+        b1 <- (z - knots[j]) / (knots[j + degree] - knots[j])
+        b2 <- (knots[j + degree + 1] - z) /
+          (knots[j + degree + 1] - knots[j + 1])
+        B <- b1 * bsbasis(z, knots, j, degree - 1) +
+          b2 * bsbasis(z, knots, j + 1, degree - 1)
+      }
+      B[is.na(B)] <- 0
+      return(B)
+    }
+
+    ## continued on next page ...
```

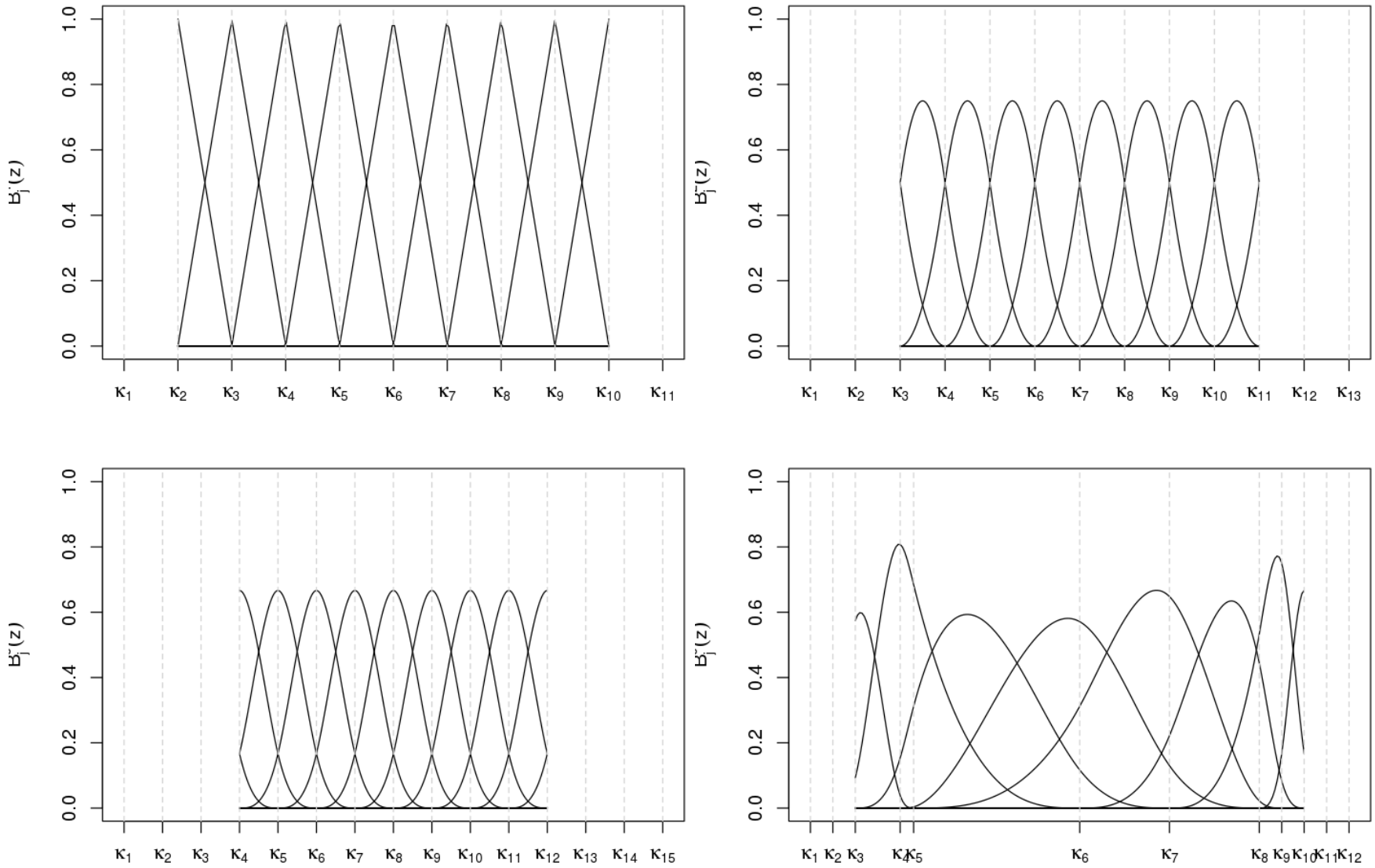
Univariate Smoothing

```
+    ## ... continued from previous page
+
+    ## compute knots
+    if(is.null(knots))
+      knots <- 40
+    if(length(knots) < 2) {
+      step <- (max(z) - min(z)) / (knots - 1)
+      knots <- seq(min(z) - degree * step,
+        max(z) + degree * step, by = step)
+    }
+
+    ## evaluate each basis function
+    ## and return the full design matrix B
+    B <- NULL
+    for(j in 1:(length(knots) - degree - 1))
+      B <- cbind(B, bsbasis(z, knots, j, degree))
+    return(B)
+  }
```

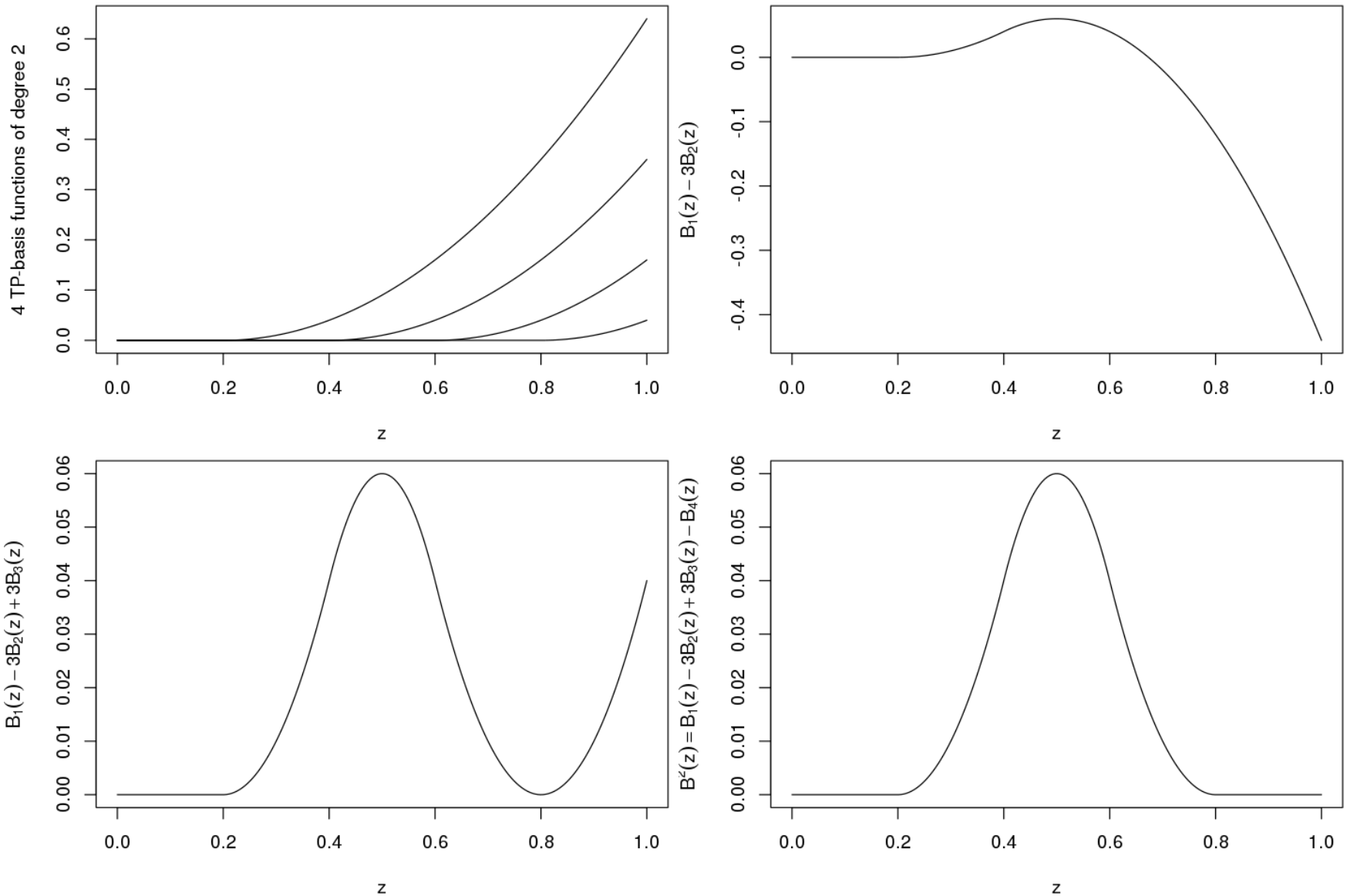
Univariate Smoothing



Univariate Smoothing



Univariate Smoothing



Univariate Smoothing

- With B-spline basis functions a penalty on the regression coefficients is not obvious, since we do not divide in a parametric and non-parametric part.
- Since we want an overall smooth function we could use the following panalty

$$\lambda \int (f''(z))^2 dz.$$

- For B-splines we can construct simpler equivalent penalty terms

$$||\mathbf{y} - \mathbf{Z}\boldsymbol{\gamma}||^2 + \lambda \boldsymbol{\gamma}^\top \mathbf{K} \boldsymbol{\gamma}$$

with

$$\lambda \boldsymbol{\gamma}^\top \mathbf{K} \boldsymbol{\gamma} = \sum_{j=k+1}^{k} (\Delta^d \gamma_j)^2.$$

Univariate Smoothing

- Δ^d is the dth order difference which is defined recursively

$$\begin{aligned}
\Delta^1\gamma_j &= \gamma_j - \gamma_{j-1} \\
\Delta^2\gamma_j &= \Delta^1\Delta^1\gamma_j = \Delta^1\gamma_j - \Delta^1\gamma_{j-1} = \gamma_j - 2\gamma_{j-1} + \gamma_{j-2} \\
&\vdots \\
\Delta^d\gamma_j &= \Delta^{d-1}\gamma_j - \Delta^{d-1}\gamma_{j-1}.
\end{aligned}$$

- The first order difference matrix is then given by

$$\mathbf{D}_1 = \begin{pmatrix} -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \end{pmatrix} \text{ with } \mathbf{D}_1\boldsymbol{\gamma} = \begin{pmatrix} \gamma_2 - \gamma_1 \\ \vdots \\ \gamma_k - \gamma_{k-1} \end{pmatrix}.$$

Univariate Smoothing

- The difference matrices can be computed recursively with

$$\mathbf{D}_d = \mathbf{D}_1 \mathbf{D}_{d-1}.$$

- Now, the resulting penalty matrix $\mathbf{K}$ is

$$\mathbf{K} = \mathbf{D}_k^\top \mathbf{D}_k.$$

```
R> penalty <- function(order = 2, k = 7) {
+    D <- diag(k)
+    for(i in 1:order)
+      D <- diff(D)
+    K <- crossprod(D, D)
+    return(K)
+  }
```

Univariate Smoothing

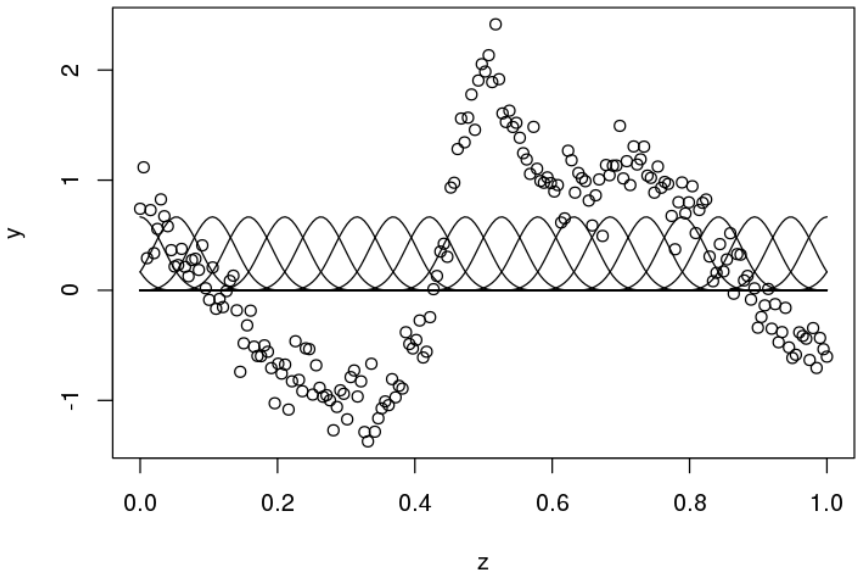
```
R> penalty(order = 1)
```

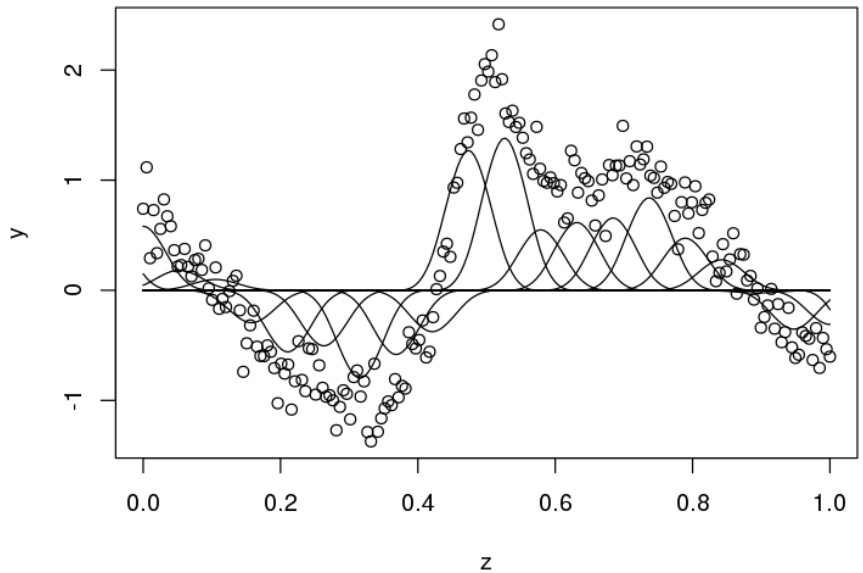
```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]    1   -1    0    0    0    0    0
[2,]   -1    2   -1    0    0    0    0
[3,]    0   -1    2   -1    0    0    0
[4,]    0    0   -1    2   -1    0    0
[5,]    0    0    0   -1    2   -1    0
[6,]    0    0    0    0   -1    2   -1
[7,]    0    0  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

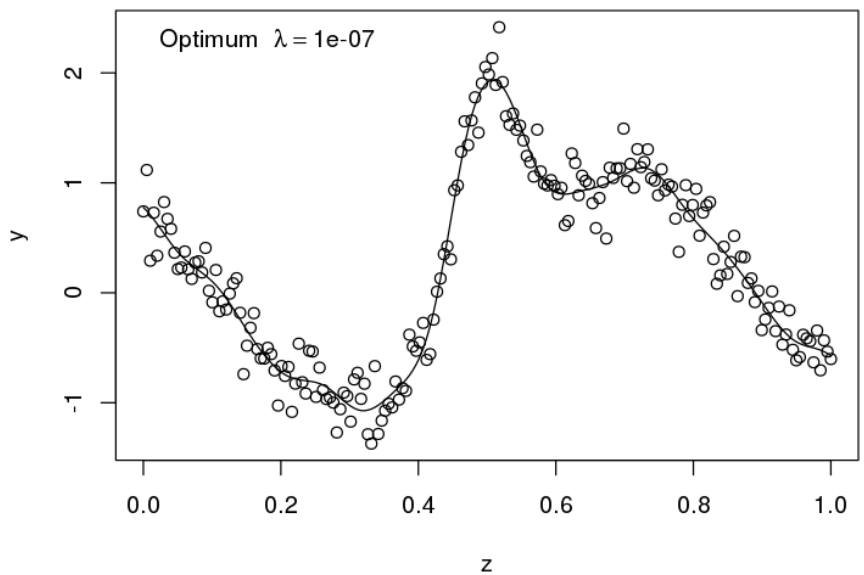
```
R> penalty(order = 2)
```

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

Univariate Smoothing





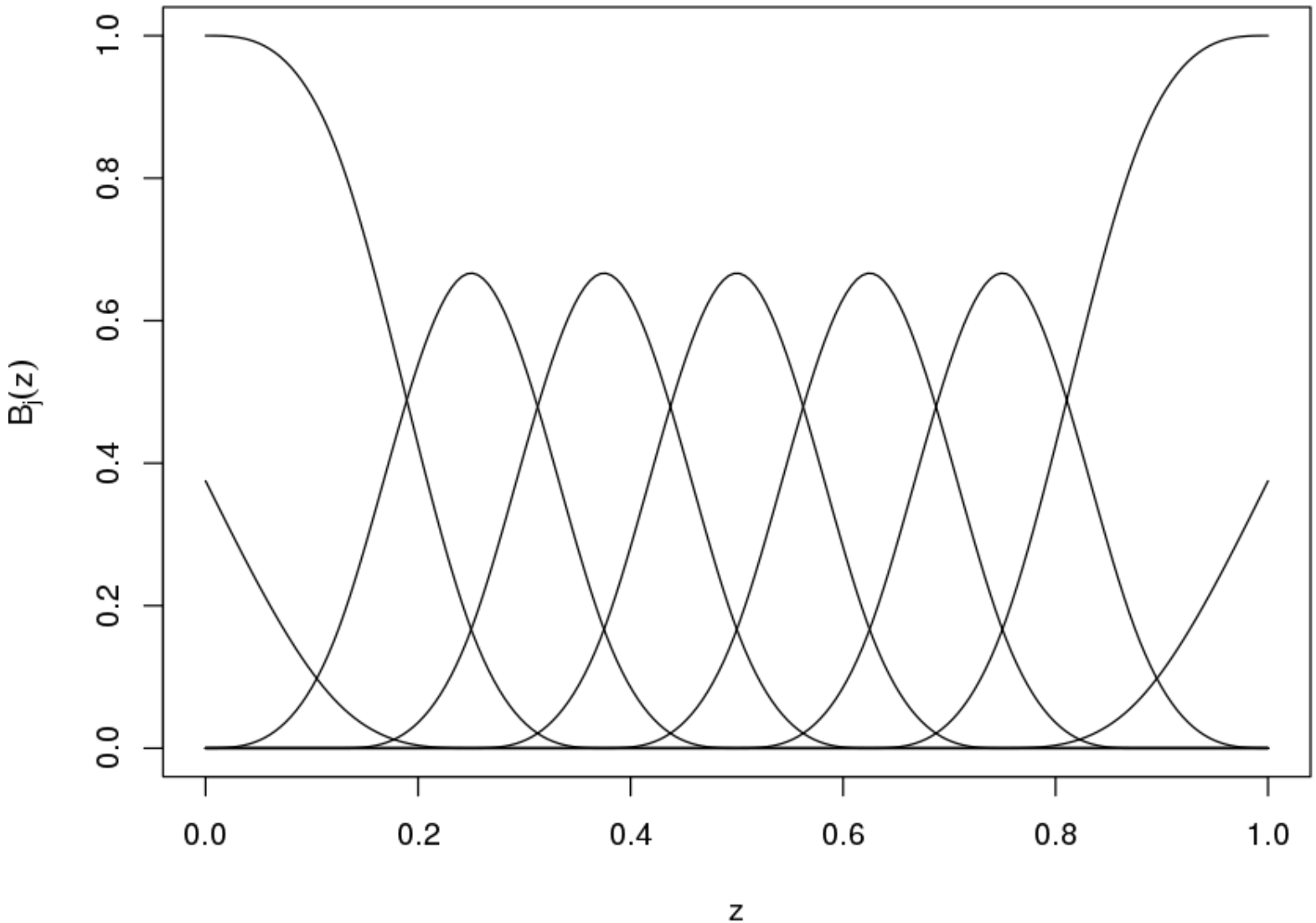


Univariate Smoothing

Natural Cubic Splines

- A commonly used modification of the cubic spline model is the *natural cubic spline basis*.
- Natural cubic splines are cubic splines with the constraint that they are linear within $[min(z), \kappa_2]$ and $[\kappa_{m-1}, max(z)]$.
- The linearity is enforced through constraints that the spline f satisfies $f''(min(z)) = f''(max(z)) = 0$.

Univariate Smoothing



Univariate Smoothing

- A *smoothing spline* minimizes the residual sum of squares plus a penalty on the integral of the squared second derivative

$$||\mathbf{y} - \mathbf{Z}\gamma||^2 + \lambda \int (f''(z))^2 dz.$$

- Although there is no prior constraint imposed on $\hat{f}$ that it even be a spline, it has been proved that the minimizer of this penalized sum of squares is a natural cubic spline with knots at the z_i.
- Therefore natural cubic splines are called “natural” since they arise as the solution of an optimization problem.
- The smoothing spline penalty can e.g. be implemented with P-splines as well

$$\int (f''(z))^2 dz = \sum_{i=1}^{k} \sum_{j=1}^{k} \gamma_i \gamma_j \int \mathbf{B}_i''(z) \mathbf{B}_j''(z) dz = \gamma^\top \mathbf{K} \gamma.$$

Univariate Smoothing

Derivative Penalty for P-splines

- Since B-splines are constructed through polynomial pieces, a derivative formula for each basis function is easily set up by

$$\frac{\partial}{\partial z} B_j^l(z) = l \cdot \left(\frac{1}{\kappa_{j+l} - \kappa_j} B_j^{l-1}(z) - \frac{1}{\kappa_{j+l+1} - \kappa_{j+1}} B_{j+1}^{l-1}(z) \right).$$

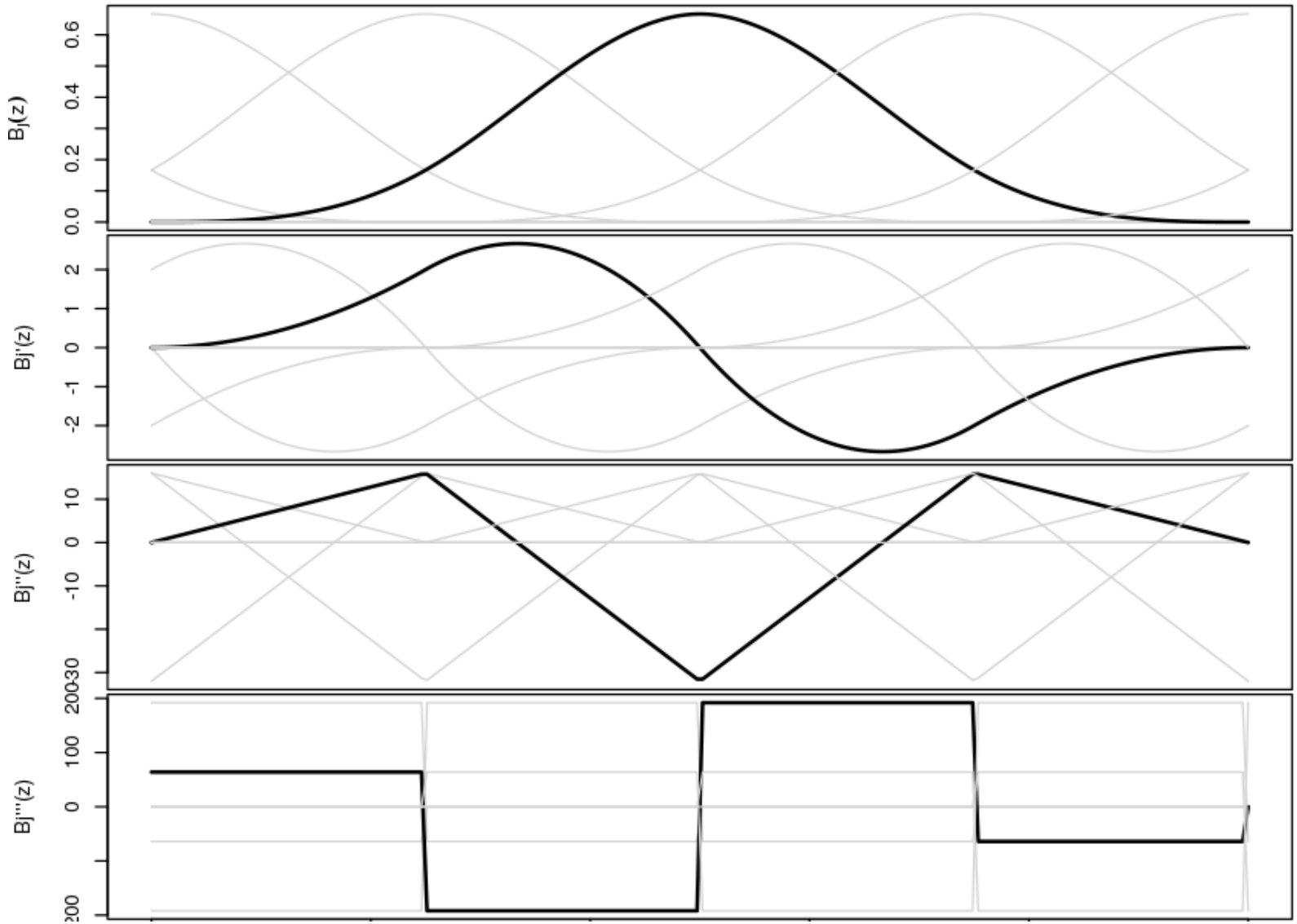
- The derivative for the polynom spline is then given by

$$\frac{\partial}{\partial z} \sum_{j=1}^{k} \gamma_j B_j^l(z) = l \cdot \sum_{j=1}^{k} \frac{\gamma_j - \gamma_{j-1}}{\kappa_{j+l} - \kappa_j} B_j^{l-1}(z).$$

Univariate Smoothing

```
+    bsderiv <- function(z, d, knots, j, degree) {
+      if(degree == 0)
+        B <- rep(0, length(z))
+      if(degree > 0) {
+        b1 <- degree / (knots[j + degree] - knots[j])
+        b2 <- degree / (knots[j + degree + 1] - knots[j + 1])
+        if(d == 1) {
+          B <- b1 * bsbasis(z, knots, j, degree - 1) -
+            b2 * bsbasis(z, knots, j + 1, degree - 1)
+        } else {
+          B <- b1 * bsderiv(z, d - 1, knots, j, degree - 1) -
+            b2 * bsderiv(z, d - 1, knots, j + 1, degree - 1)
+      }
+      B[is.na(B)] <- 0
+      return(B)
+    }
```

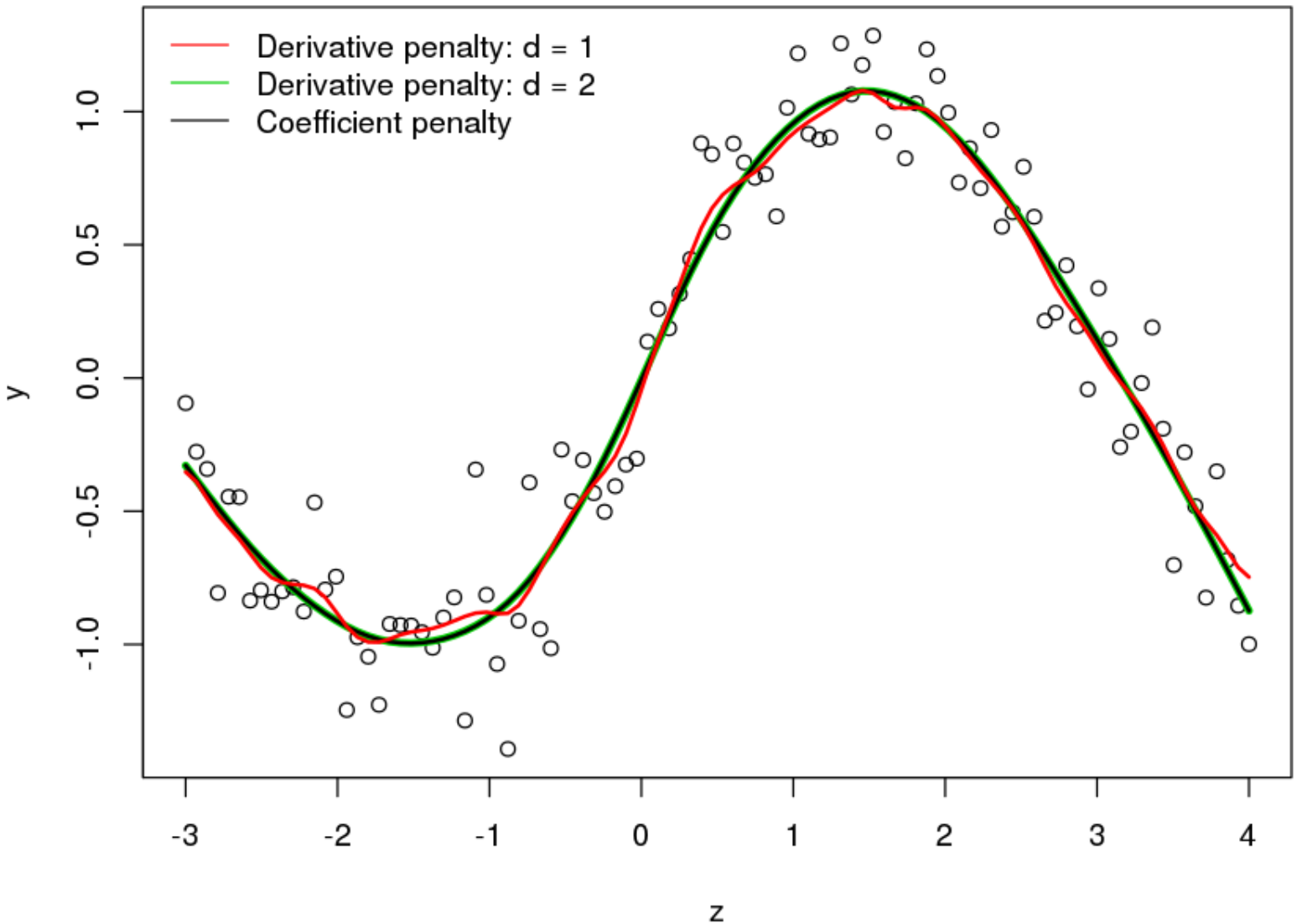
Univariate Smoothing



Univariate Smoothing

```
R> set.seed(111)
R> n <- 100
R> z <- seq(-3, 4, length = n)
R> y <- sin(z) + rnorm(n, sd = 0.2)
R> Z <- bsDesign(z)
R> Zderiv1 <- bsDeriv(z, d = 1)
R> Zderiv2 <- bsDeriv(z, d = 2)
R> K1 <- crossprod(Zderiv1)
R> K2 <- crossprod(Zderiv2)
R> lambda1 <- find.hyper(y, Z, K1, method = "GCV")
R> lambda2 <- find.hyper(y, Z, K2, method = "GCV")
R> c(lambda1, lambda2)

[1] 0.03223946 0.01578443

R> yhat1 <- Z %*% solve(t(Z) %*% Z + lambda1 * K1) %*% t(Z) %*% y
R> yhat2 <- Z %*% solve(t(Z) %*% Z + lambda2 * K2) %*% t(Z) %*% y
```

Univariate Smoothing



Univariate Smoothing

Monotonicity constraints

- Sometimes it might be necessary to constrain a smooth function estimate, e.g. for estimating price functions, we would like to have either strictly increasing or decreasing shapes.
- This is achieved relatively easy using P-splines, since one can show that e.g. a monotone increasing function has parameters

$$\gamma_1 < \gamma_2 < \ldots < \gamma_k.$$

- The previous presented difference penalty forces a smooth behavior of an estimated signal.
- Using asymmetric penalties, one can impose a variety of other useful properties, like a monotone or concave shape.

Univariate Smoothing

- Consider the following function

$$Q(\gamma) = ||\mathbf{y} - \mathbf{Z}\gamma||^2 + \lambda\gamma^\top \mathbf{K}\gamma + \alpha\gamma^\top \mathbf{D}_1^\top \mathbf{V}\mathbf{D}_1\gamma.$$

- Here,

$$\alpha\gamma^\top \mathbf{D}_1^\top \mathbf{V}\mathbf{D}_1\gamma$$

  is an asymetric penalty on the regression coefficients γ.
- For an increasing constraint, the matrix $\mathbf{V}$ is a diagonal matrix with weights $\mathbf{v}$ defined as follows

$$\begin{aligned} v_j &= 0 \quad \text{if} \quad \Delta\gamma_j > 0, \\ v_j &= 1 \quad \text{if} \quad \Delta\gamma_j \leq 0. \end{aligned}$$

  For decreasing functions vice versa.

Univariate Smoothing

- The parameter α is set quite large, e.g. $\alpha = 10^6$, to really penalize for monotonicity violations.
- To estimate the monotone function one has to solve

$$\hat{\gamma} = (\mathbf{Z}^\top \mathbf{Z} + \lambda \mathbf{K} + \alpha \mathbf{D}_1^\top \mathbf{V} \mathbf{D}_1)^{-1} \mathbf{Z}^\top \mathbf{y}$$

  iteratively by first computing $\hat{\gamma}$ the usual way and then determine the weights $\mathbf{v}$ in $\mathbf{V}$ by e.g. a simple search algorithm.

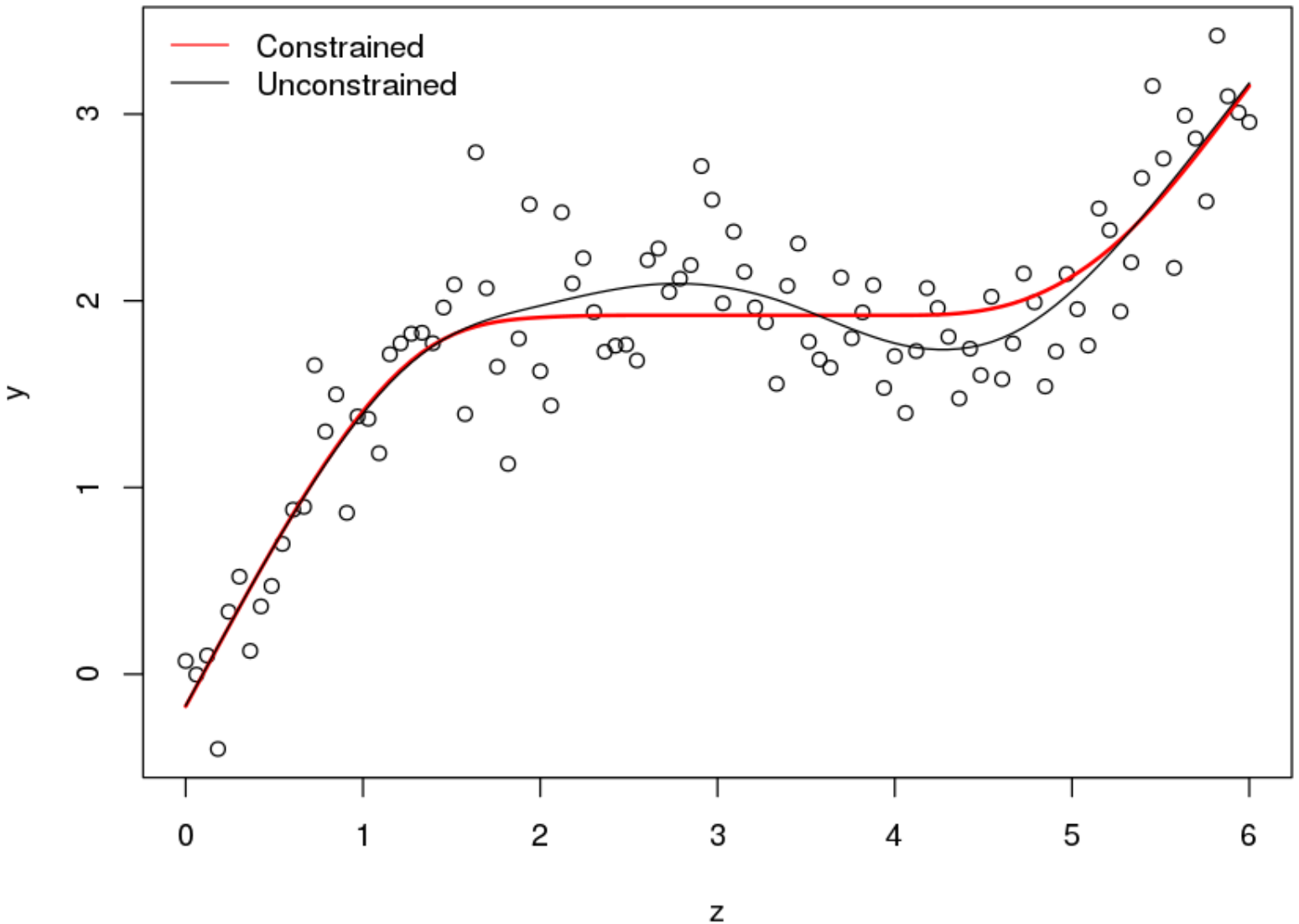
Univariate Smoothing

```
R> mPspline <- function(y, z, knots = 10, order = 2,
+    alpha = 1e+06, constr = 1, method = "GCV", eps = 0.0001)
+  {
+    Z <- bsDesign(z, knots = knots)
+    K <- penalty(order, ncol(Z))
+    D <- diff(diag(ncol(Z)))
+
+    vfun <- function(gamma, constr) {
+      v <- diff(drop(gamma))
+      if(constr < 2)
+        v <- (v < 0) * 1
+      else
+        v <- (v > 0) * 1
+      v
+    }
+
+  ## continued on next page
```

Univariate Smoothing

```
+  ## continued from previous page
+
+    lambda <- find.hyper(y, Z, K, method)
+    gamma <- solve(t(Z) %*% Z + lambda * K) %*% t(Z) %*% y
+    d <- 1
+    while(d > eps) {
+      v <- diag(vfun(gamma, constr))
+      gamma <- solve(t(Z) %*% Z + lambda * K +
+        alpha * t(D) %*% v %*% D) %*% t(Z) %*% y
+      d <- sum((v - diag(vfun(gamma, constr)))^2)
+    }
+    return(Z %*% gamma)
+  }
```

Univariate Smoothing



Univariate Smoothing

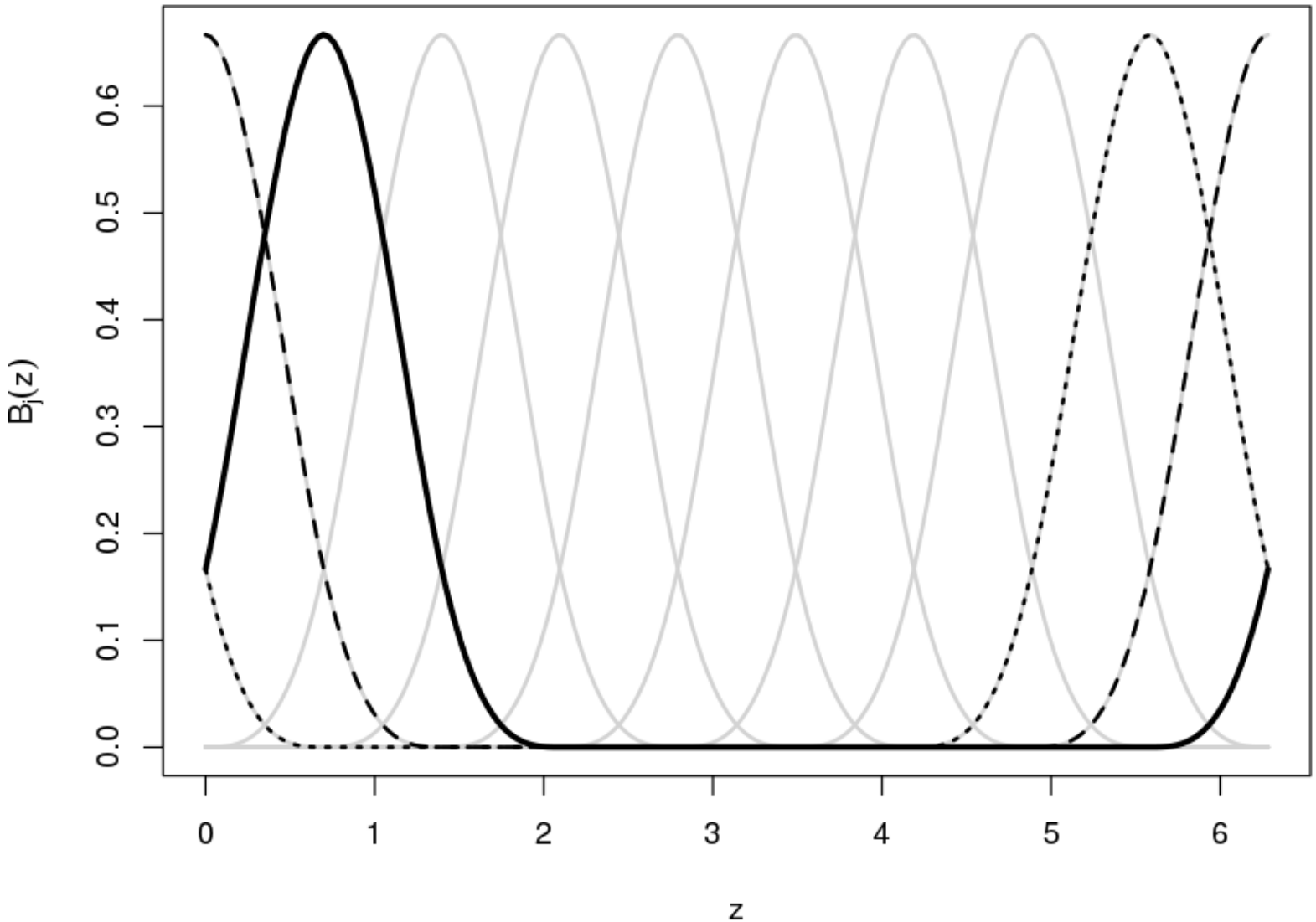
Cyclic constraints

- It is quite often appropriate for a model smooth to be “cyclic”, meaning that the function has the same value at its upper and lower boundaries.
- E.g., in most applications, it would not be appropriate for a smooth function of time of year to change discontinuously at the year end.
- For P-splines, to include cyclic penalties, it only requires to slightly modify the design matrix and the difference penalty.
- The cyclic B-spline basis functions are constructed without knot expansion. The B-splines are “wrapped” at the boundary knots.

Univariate Smoothing

```
R> cpsDesign <- function(z, degree = 3, knots = NULL, order = 2)
+  {
+    ## evaluate the B-spline design matrix
+    B <- bsDesign(z, degree, knots)
+    ## wrap basis functions
+    B <- cbind(B[ ,1:degree] + B[, (ncol(B) - (degree - 1)):ncol(B)],
+      B[, (degree + 1):(ncol(B) - degree)])
+
+    ## generate the penalty matrix the same way
+    K <- diff(diag(ncol(B) + order), differences = order)
+    ## save first columns
+    tmp <- K[, (1:order)]
+    ## drop first columns
+    K <- K[, -(1:order)]
+    indx <- (ncol(B) - order + 1):(ncol(B))
+    ## add first columns
+    K[, indx] <- K[, indx] + tmp
+
+    return(list("B" = B, "K" = crossprod(K)))
+  }
```

Univariate Smoothing

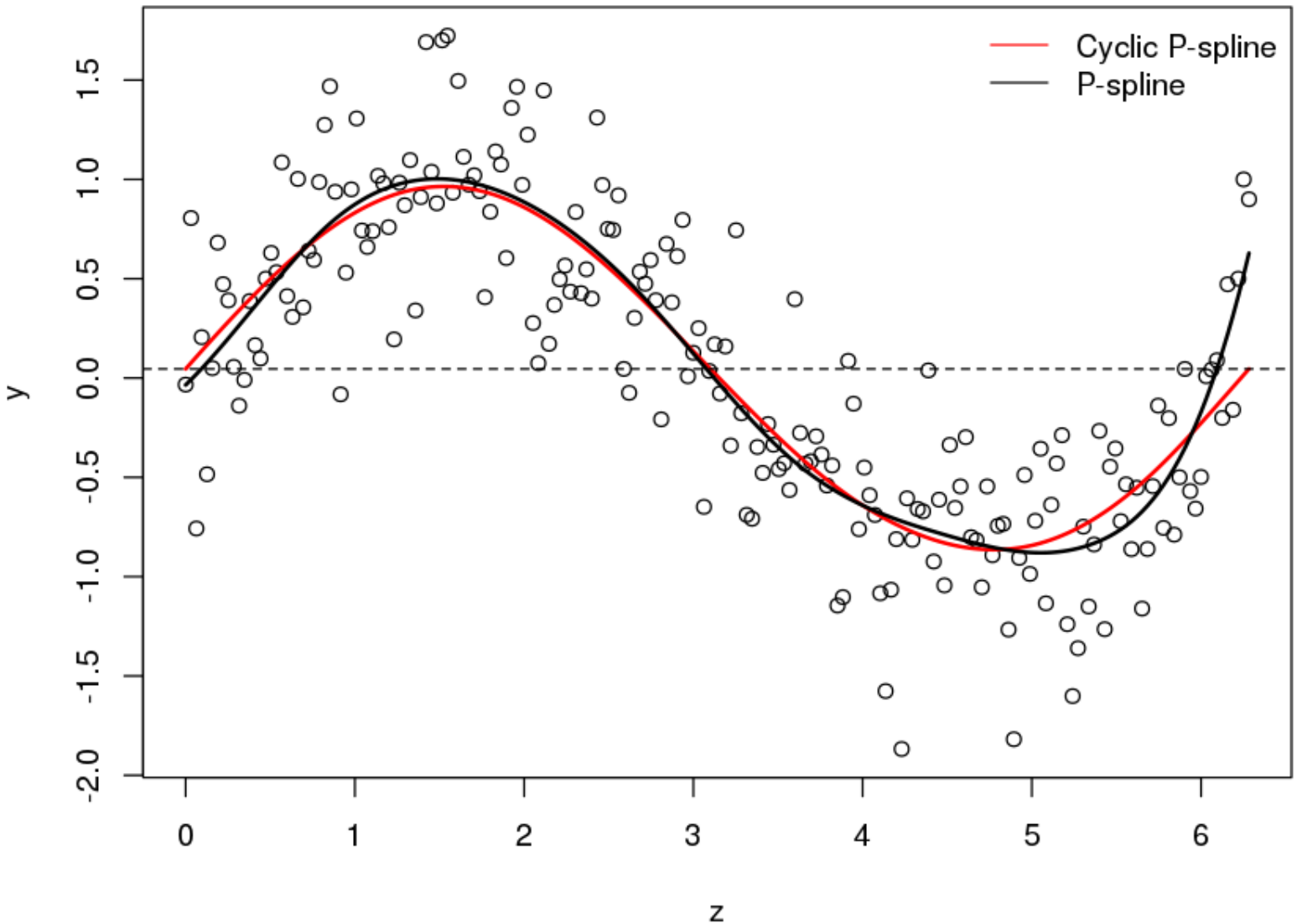


Univariate Smoothing

```
R> K
```

```
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
 [1,]    6   -4    1    0    0    0    0    1   -4
 [2,]   -4    6   -4    1    0    0    0    0    1
 [3,]    1   -4    6   -4    1    0    0    0    0
 [4,]    0    1   -4    6   -4    1    0    0    0
 [5,]    0    0    1   -4    6   -4    1    0    0
 [6,]    0    0    0    1   -4    6   -4    1    0
 [7,]    0    0    0    0    1   -4    6   -4    1
 [8,]    1    0    0    0    0    1   -4    6   -4
 [9,]   -4    1    0    0    0    0    1   -4    6
```

Univariate Smoothing



Univariate Smoothing

Random Walks

- Randow walk models are typically used in time series analysis.
- Let t be a time trend, which should also be modeled nonparametrically:

$$y_t = f(t) + \varepsilon_t, \quad t = 1, \ldots, T.$$

- Now we define parameters γ_t for each time point t and model

$$f(t) = \gamma_t = \gamma_{t-1} + u_t, \quad u_t \sim N(0, \tau^2),$$

or, for a second order random walk we get

$$f(t) = \gamma_t = 2\gamma_{t-1} - \gamma_{t-2} + u_t, \quad u_t \sim N(0, \tau^2).$$

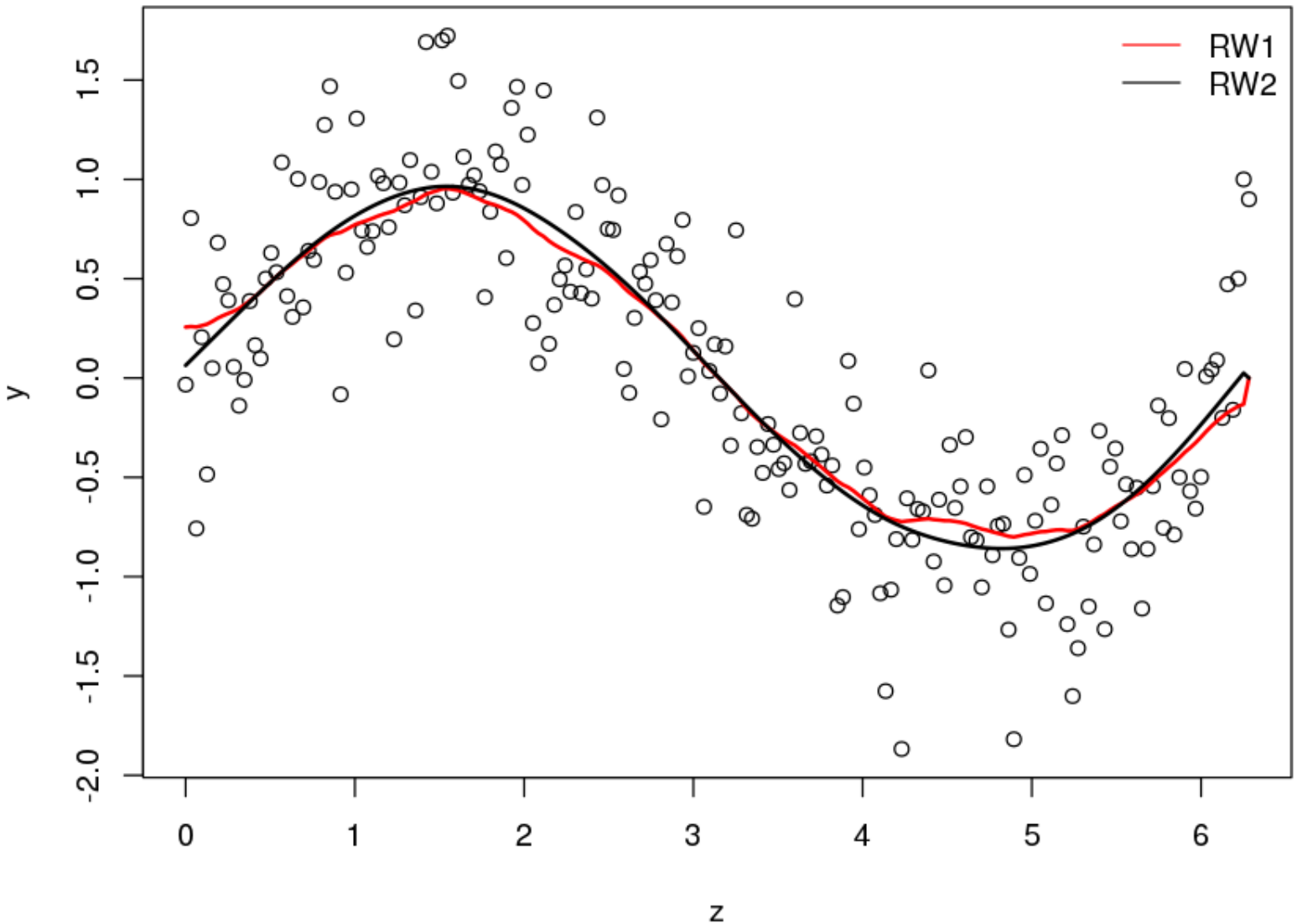
Univariate Smoothing

- Note that random walks can be interpreted as a special version of a P-spline of degree $l = 0$, with knot locations at the individual observations of covariate z.
- From this point of view we can again construct a penalized least squares criterion similar to P-splines (Hodrick-Prescott filter)

$$\sum_{t=1}^{T}(y_t - \gamma_t)^2 + \lambda \sum_{t=3}^{T}(\gamma_t - 2\gamma_{t-1} + \gamma_{t-2})^2 = ||\mathbf{y} - \boldsymbol{\gamma}||^2 + \lambda \boldsymbol{\gamma}^\top \mathbf{K} \boldsymbol{\gamma}.$$

```
R> rwDesign <- function(z, order = 2)
+  {
+    knots <- sort(unique(z))
+    B <- bsDesign(z, degree = 0, knots = knots)
+    K <- penalty(order, ncol(B))
+    return(list("B" = B, "K" = K))
+  }
```

Univariate Smoothing



Univariate Smoothing

Kriging

- The method is referred to as kriging, named after the South African mining engineer D. G. Krige, who used the approach when modeling the course of stone cores based on depth drillings.
- The main idea of kriging is to express (originally spatial) correlations among the data based on parametric correlation functions and to employ this information when fitting a regression model.
- Let

$$y_t = \mathbf{x}_t^\top + \varepsilon_t,$$

be a time series with $t = 1, \ldots, T$.

Univariate Smoothing

- In contrast to the classical linear model, the assumption of uncorrelated errors is usually violated in this situation, and it seems plausible to rather consider temporally correlated errors.
- Therefore we assume that the error ε is multivariate normal with expectation $\mathbf{0}$ and covariance matrix $\sigma^2\mathbf{I} + \tau^2\mathbf{R}$.
- The matrix $\mathbf{R}$ is a correlation matrix, described in terms of a parametric correlation function ρ, leading to the elements

$$\mathbf{R}[t, s] = \mathit{Corr}(\varepsilon_t, \varepsilon_s) = \rho(t, s).$$

- Hence, the covariance matrix is composed of an uncorrelated part $\sigma^2\mathbf{I}$ and a correlated part $\tau^2\mathbf{R}$.

Univariate Smoothing

- Most commonly, stationary error distributions are assumed, i.e. $\rho(t, s)$ only depends on the time difference $|t - s|$, and not on the exact locations of the time points t and s.
- This yields correlation functions $\rho(h)$ with $h = |t - s| \geq 0$.
  - $\rho(0) = 1$.
  - As $h \to \infty$, we have $\rho(h) \to 0$, i.e., observations having large temporal distance the correlation becomes negligible.
  - The correlation function $\rho(h)$ decreases monotonically, i.e., observations that are temporally close to each other should always show a higher correlation than observations that are more distant apart in time.

Univariate Smoothing

- *Spherical correlation functions*

$$\rho(h;\phi) = \begin{cases} 1 - \frac{3}{2}|h/\phi| + \frac{1}{2}|h/\phi|^3 & 0 \leq h \leq \phi, \\ 0 & h > \phi. \end{cases}$$

  Non-differentiable for $h = \phi$.

- *Power exponential family*

$$\rho(h;\phi,\upsilon) = \exp\left(-|h/\phi|^{\upsilon}\right).$$

  The power exponential correlation between two points remains positive regardless of their distance.

Univariate Smoothing

- In practice, the most flexible correlation functions is used, the *Matérn family*, given by

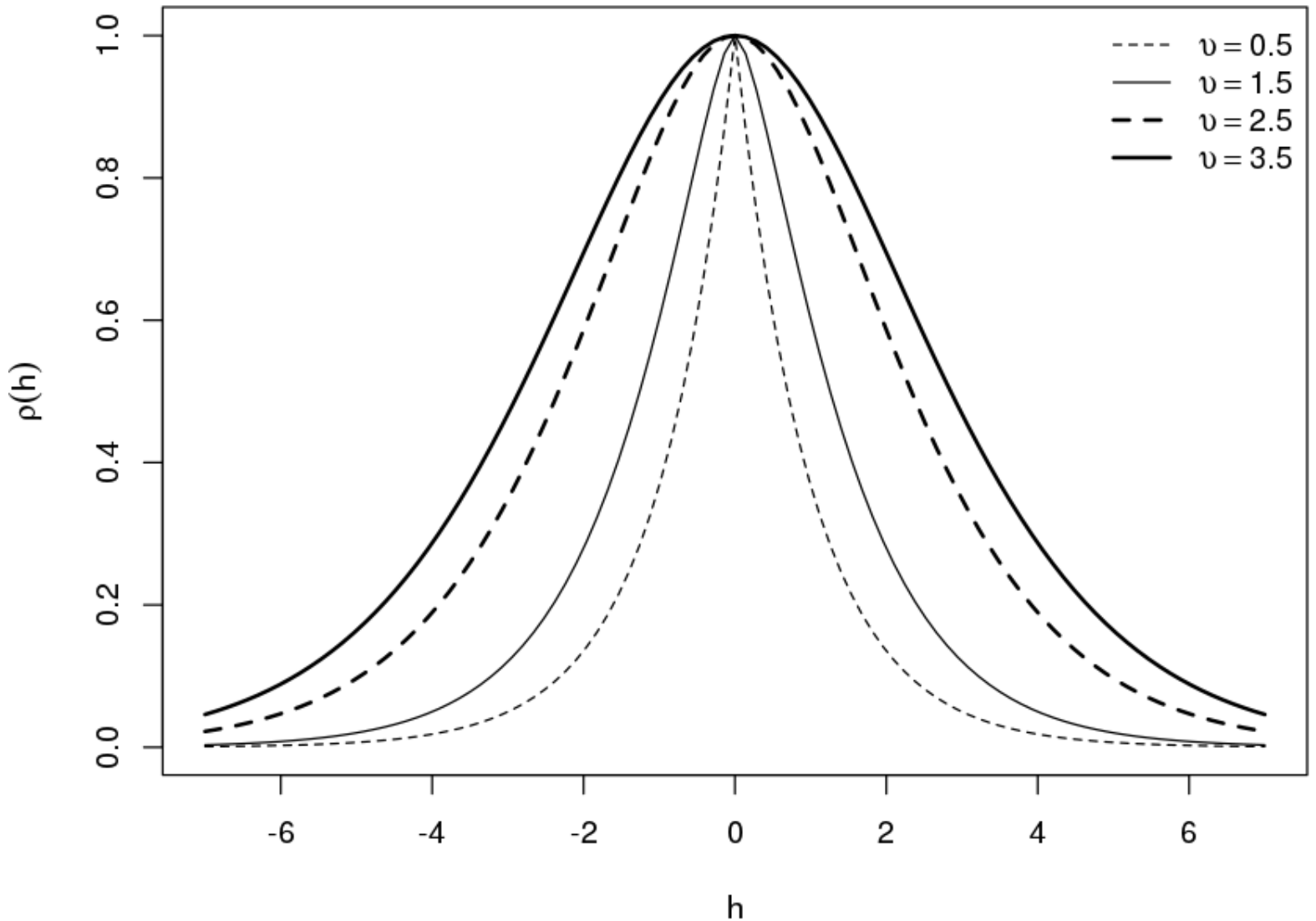
$$\rho(h; \phi, \upsilon) = (2^{\upsilon-1}\Gamma(\upsilon))^{-1}(h/\phi)^{\upsilon} K_{\upsilon}(h/\phi),$$

  where K_{υ} is the modified Bessel function of order υ.

- K_{υ} does not have a closed form for general υ's, however for $\upsilon = 0.5, 1.5, 2.5, \ldots$ we get

$$\begin{aligned}
\rho(h; \phi, \upsilon = 0.5) &= \exp(-|h/\phi|), \\
\rho(h; \phi, \upsilon = 1.5) &= \exp(-|h/\phi|)(1 + |h/\phi|), \\
\rho(h; \phi, \upsilon = 2.5) &= \exp(-|h/\phi|)(1 + |h/\phi| + \tfrac{1}{3}|h/\phi|^2), \\
\rho(h; \phi, \upsilon = 3.5) &= \exp(-|h/\phi|)(1 + |h/\phi| + \tfrac{2}{5}|h/\phi|^2 + \tfrac{1}{15}|h/\phi|^3).
\end{aligned}$$

Univariate Smoothing



Univariate Smoothing

- In order to motivate the use of correlation functions and stationary Gaussian processes in nonparametric function estimation, we decompose the error ε_t into a temporally correlated component γ_t and the remaining independent and identically distributed component u_t:

$$y_t = \mathbf{x}_t^\top \boldsymbol{\beta} + \gamma_t + u_t.$$

- In matrix notation we get

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\boldsymbol{\gamma} + \mathbf{u},$$

  with $\mathbf{Z} = \mathbf{I}$.

- In a second step, we further reparameterize the model to

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \underbrace{\mathbf{ZR}}_{\tilde{\mathbf{Z}}} \cdot \underbrace{\mathbf{R}^{-1}\boldsymbol{\gamma}}_{\tilde{\boldsymbol{\gamma}}} + \mathbf{u} = \mathbf{X}\boldsymbol{\beta} + \tilde{\mathbf{Z}}\tilde{\boldsymbol{\gamma}} + \mathbf{u}.$$

Univariate Smoothing

- This reparameterization evidently leads to the same distribution of $\mathbf{y}$ and, thus, to an equivalent model formulation.
- However, the interpretation of the design matrix $\tilde{\mathbf{Z}}$ changes. Due to the special structure of $\mathbf{Z}$, we obtain

$$\tilde{\mathbf{Z}}[t, s] = \rho(t, s).$$

  for the individual entries.
- If we compare this definition with the construction of the design matrix for B-splines or TP-splines, we find that the correlation function ρ is used like a basis function, and that the observed time points t adopt the role of knots.
- Thus, for the temporal trend, we have

$$f(t) = \gamma_t = \sum_{j=1}^{k} \tilde{\gamma}_j \rho(t, s).$$

Univariate Smoothing

- In order to use kriging for the estimation of nonparametric covariate effects, we return to the model

$$y_i = f(z_i) + \varepsilon_i.$$

- If $z_{(1)} < \ldots < z_{(k)}$ are the k unique and ordered covariate values, we define the parameters $\gamma_j = f(z_{(j)})$ and assume a stationary Gaussian process prior, with expectation 0, variance τ^2, and correlation function

$$\rho(\gamma_j, \gamma_d) = \rho(|z_{(j)} - z_{(d)}|).$$

Univariate Smoothing

- Accordingly, we can also view the correlation function as a basis function, and, as with smoothing splines, we can identify the knot positions at the covariate locations $z_{(j)}$. We thus obtain the representation

$$\mathbf{y} = \mathbf{Z}\boldsymbol{\gamma} + \boldsymbol{\varepsilon},$$

  with $\mathbf{Z}[i, j] = \rho(|z_i - z_{(j)}|)$ and $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_k)^\top$.
- This yields the penalized least squares criterion

$$||\mathbf{y} - \mathbf{Z}\boldsymbol{\gamma}||^2 + \lambda \boldsymbol{\gamma}^\top \mathbf{K} \boldsymbol{\gamma},$$

  with $\mathbf{K}[j, d] = \rho(|z_{(j)} - z_{(d)}|)$. Smoothing properties of the chosen correlation function carry over to the estimate $\hat{f}(z)$.

Univariate Smoothing

- The scale parameter determined via

$$\hat{\phi} = \max_{j,d} |z_{(j)} - z_{(d)}|/c.$$

- The value of the constant c is chosen such that

$$\rho(c; \phi = 1, \upsilon)$$

  is small, i.e. in essence, c determines the effective range of the correlation function.
- Choosing c in this manner ensures that the correlation functions are well distributed across the covariate domain. For polynomial splines, the choice of knots automatically guarantees such a distribution while in the case of kriging an effective range that is too small can result in non-overlapping correlation functions and induce unstable estimation of the nonparametric effects in the corresponding regions.

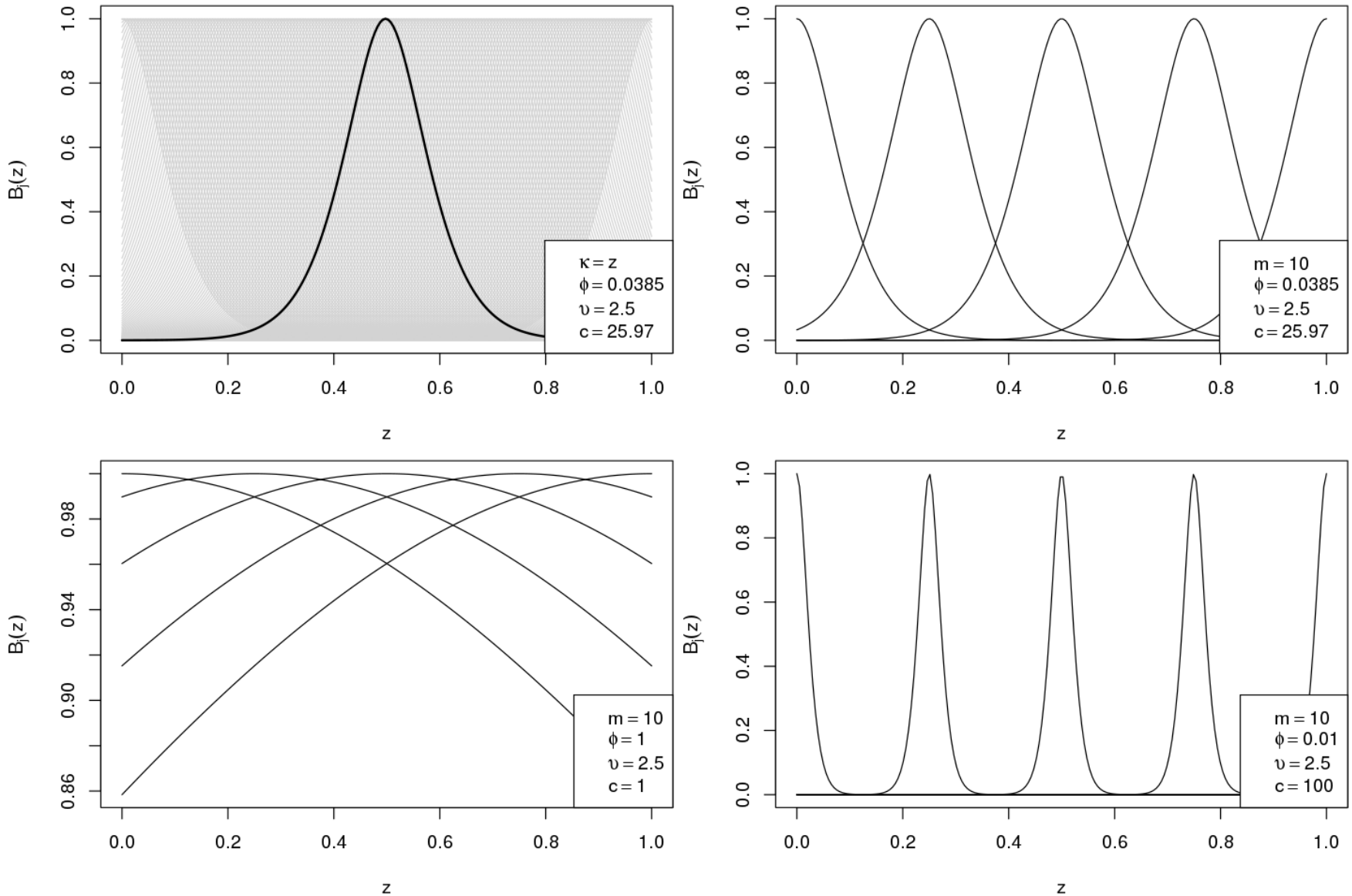
Univariate Smoothing

```
R> krDesign <- function(z, knots = NULL, rho = NULL,
+    phi = NULL, v = NULL, c = NULL) {
+    rho <- if(is.null(rho)) {
+      require("geoR")
+      geoR::matern
+    } else rho
+    knots <- if(is.null(knots)) sort(unique(z)) else knots
+    v <- if(is.null(v)) 2.5 else v
+    c <- if(is.null(c)) {
+      optim(1, matern, phi = 1, kappa = v,
          method = "L-BFGS-B", lower = 1e-10)$par
+    } else c
+    phi <- if(is.null(phi)) {
+      max(abs(diff(range(knots)))) / c
+    } else phi
+
+    ## continued on next page
```

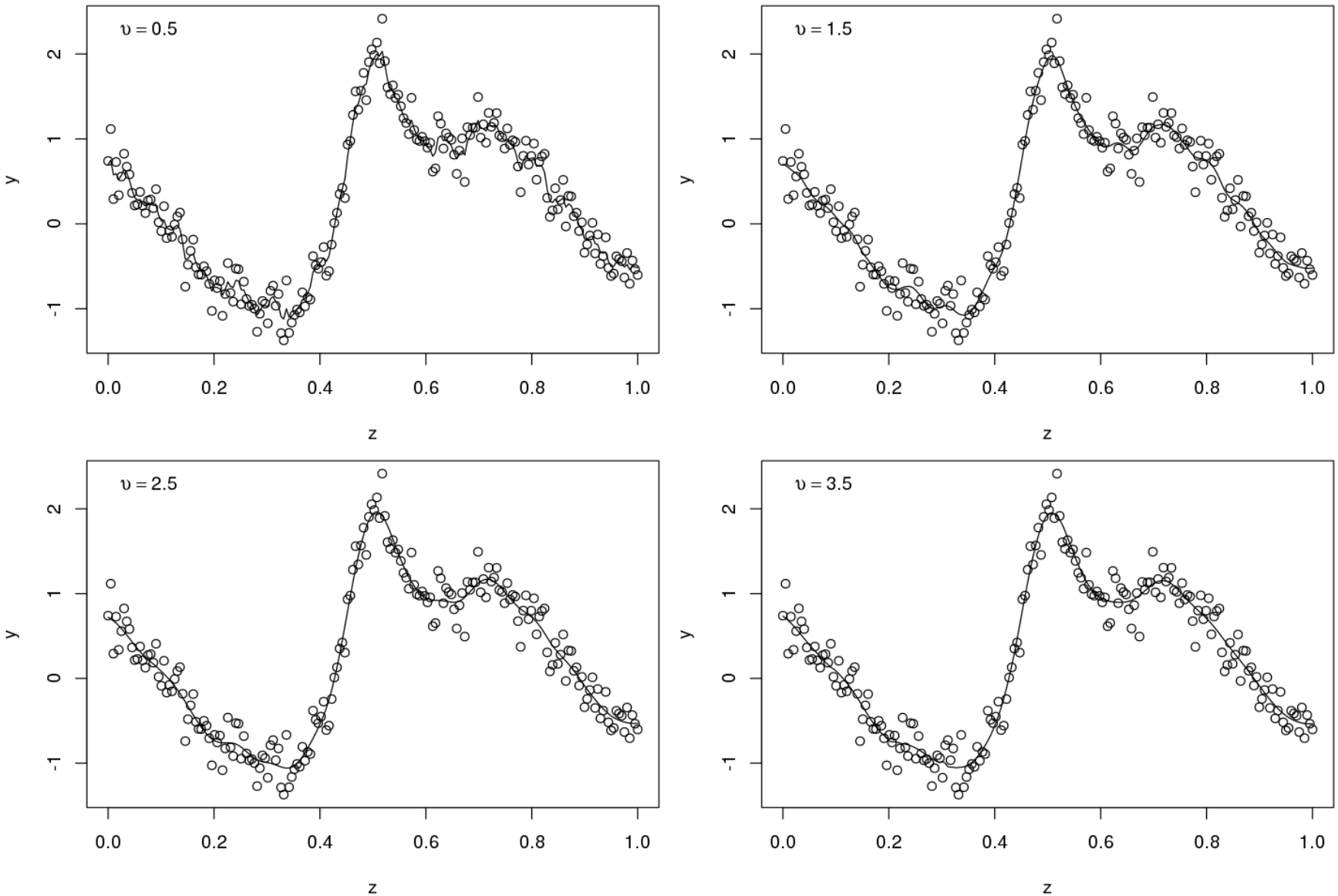
Univariate Smoothing

```
+    ## continued from previous page
+
+    B <- NULL
+    K <- as.matrix(dist(knots, diag = TRUE))
+    for(j in seq_along(knots)) {
+      h <- abs(z - knots[j])
+      B <- cbind(B, rho(h, phi, v))
+      K[, j] <- rho(K[, j], phi, v)
+    }
+    return(list("B" = B, "K" = K))
+  }
```

Univariate Smoothing



Univariate Smoothing



Univariate Smoothing

- One disadvantage of the full kriging model is the large number of parameters, i.e. the design matrix may have dimension $n \times n$.
- Again, a possible remedy is to use e.g. a maximum of $m = 40$ equidistant knots.
- Another option is to use a space filling algorithm, which searches for optimal knot locations by minimizing a coverage criterion, e.g.

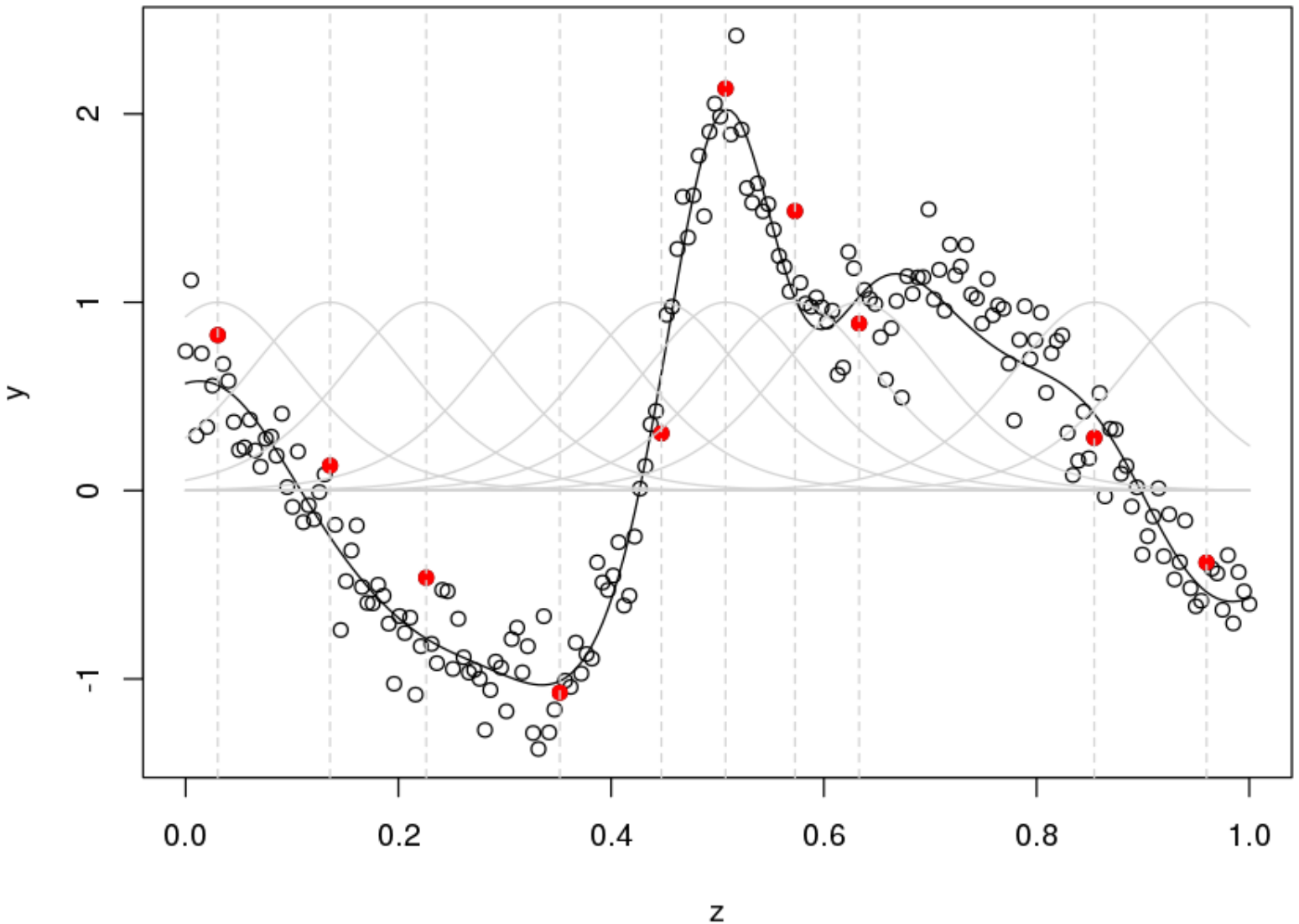
$$M(\mathbf{R}, \mathbf{C}) = \left(\sum_{j=1}^{m} \left(\sum_{i=1}^{n} dist(\mathbf{c}_j, \mathbf{r}_i) \cdot p \right) \cdot (q/p) \right) \cdot (1/q),$$

  with $p < 0$ and $q > 0$.
- This algorithm is implemented in the R package **fields**, the function is called `cover.design()`.

Univariate Smoothing

```
R> library("fields")
R> knots <- cover.design(R = cbind(z, y), nd = 10)
R> plot(z, y)
R> points(knots[, "z"], knots[, "y"],
+    pch = 21, col = "red", bg = "red")
R> D <- krDesign(z, knots = knots[, "z"], v = 3.5)
R> Z <- D$B
R> K <- D$K
R> lambda <- find.hyper(y, Z, K)
R> yhat <- sfit(y, Z, K, lambda)
R> lines(yhat ~ z)
```

Univariate Smoothing



Univariate Smoothing

Series-based smoothers

- Without loss of generality, assume that the regression function f is defined on the unit interval $[0, 1]$.
- Under certain regularity conditions, f admits the *Fourier series* representation

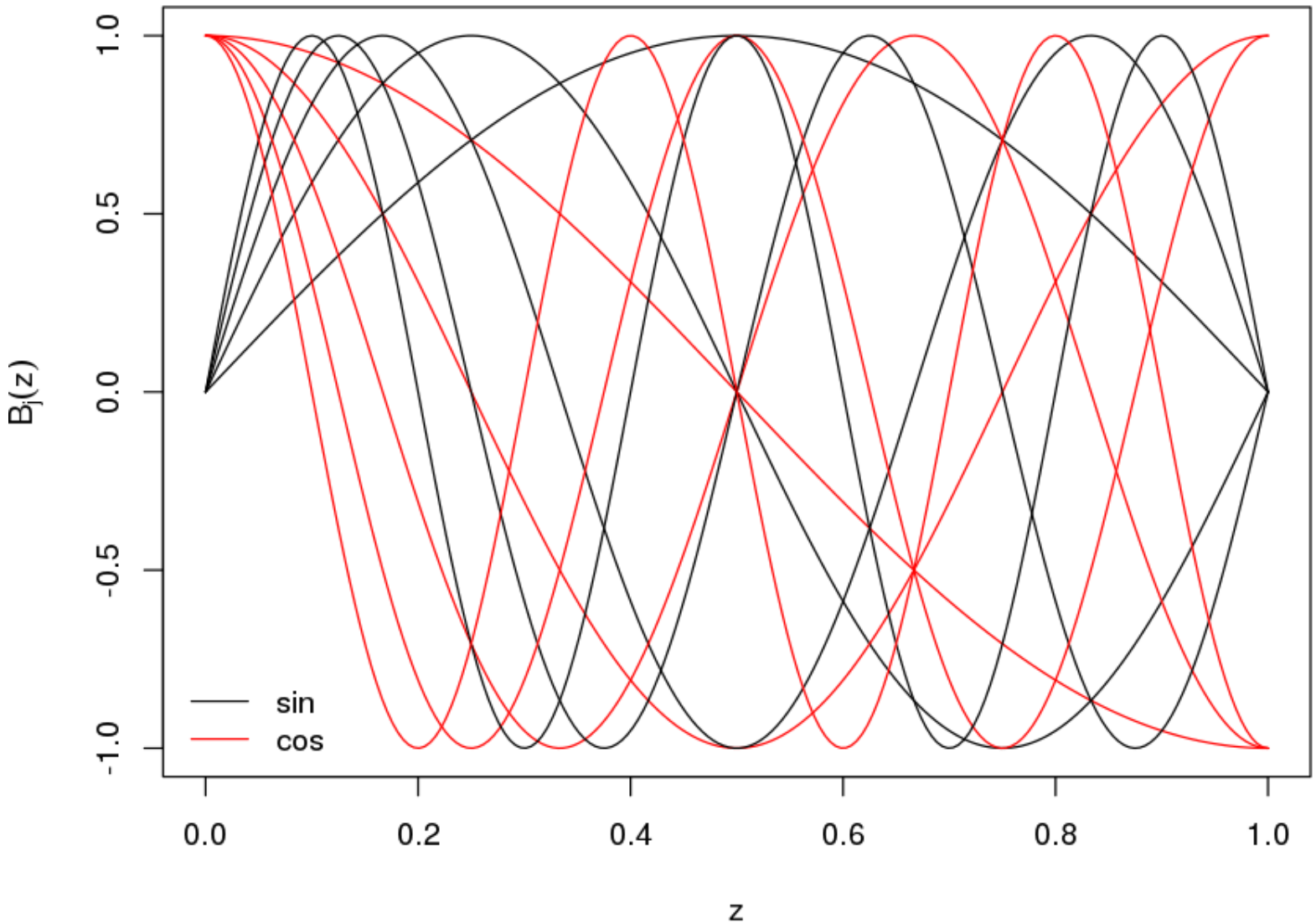
$$f(z) = \gamma_0 + \sum_{j=1}^{k} \left(\gamma_j^s sin(j\pi z) + \gamma_j^c cos(j\pi z) \right) .$$

- For higher values of j, the functions $sin(j\pi z)$ and $cos(j\pi z)$ become more oscillatory. The more oscillatory functions account for the finer structure in f.
- The cut-off value k is the smoothing parameter in this case.

Univariate Smoothing

```
R> sDesign <- function(z, k = 5) {
+    z <- (z - min(z)) / (max(z) - min(z))
+    B <- NULL; nc <- NULL
+    for(j in 1:k) {
+      B <- cbind(B, sin(j * pi * z), cos(j * pi * z))
+      nc <- c(nc, j, j)
+    }
+    colnames(B) <- nc
+    B
+  }
```

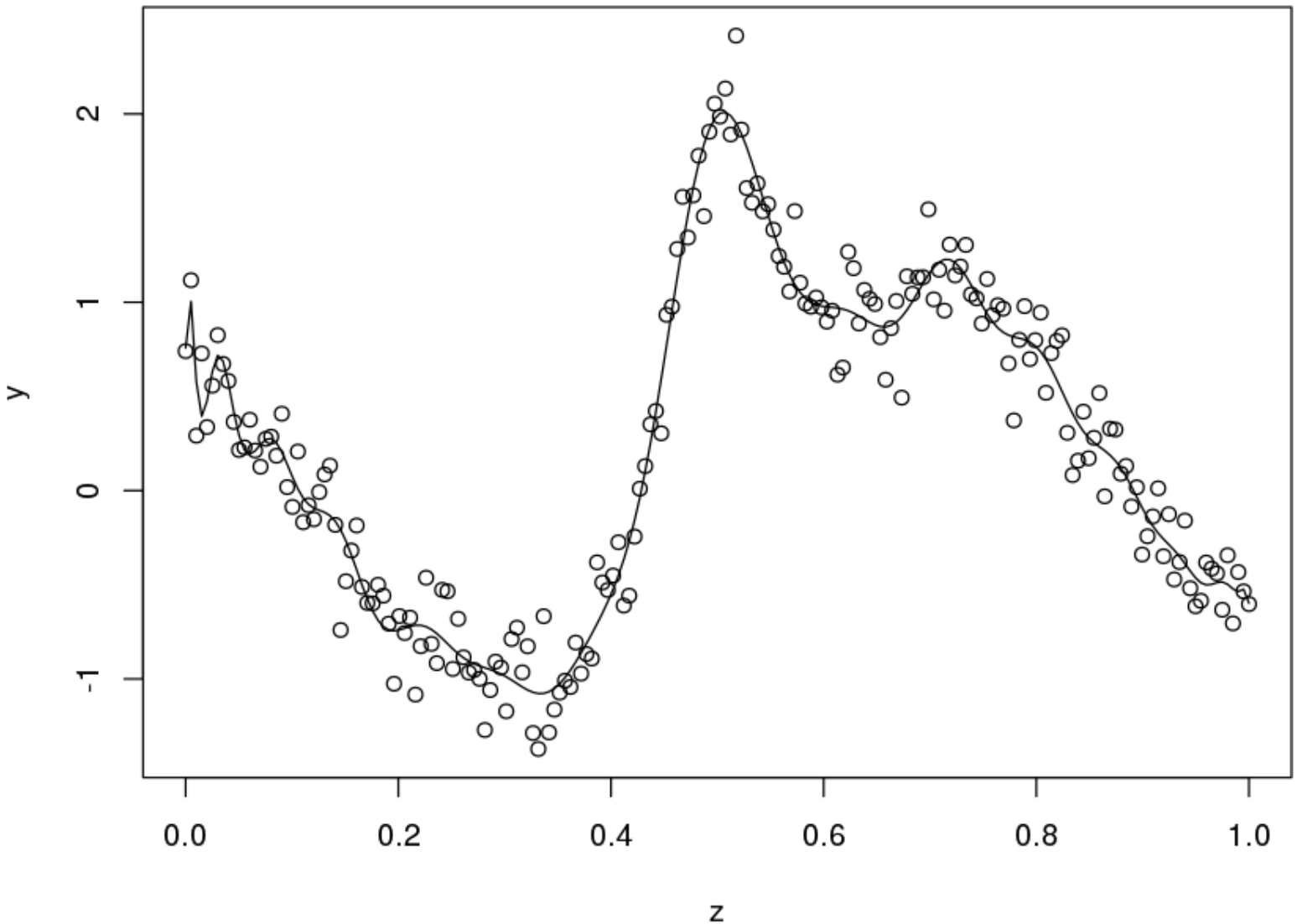
Univariate Smoothing



Univariate Smoothing

```
R> GCV <- function(x, ...) {
+    e <- residuals(x); n <- length(e)
+    RSS <- crossprod(e)
+    trH <- sum(influence(x)$hat[1:n])
+    drop(RSS * n / (n - trH)^2)
+  }

R> k <- 40
R> gcv <- NULL
R> for(j in 1:k) {
+    Z <- sDesign(z, j)
+    gcv <- c(gcv, GCV(lm(y ~ Z)))
+  }
R> which.min(gcv)

[1] 16
```

Univariate Smoothing



Univariate Smoothing

General penalization approaches

- The function f is represented through a large linear model, such that

$$\mathbf{y} = \mathbf{Z}\boldsymbol{\gamma} + \boldsymbol{\varepsilon}.$$

- Often this is achieved by approximating the function $f(z)$ using basis functions, i.e.

$$f(z) = \sum_{j=1}^{k} \gamma_j B_j(z).$$

- To regularize estimation, we assume a quadratic penalty of the form

$$\lambda \boldsymbol{\gamma}^\top \mathbf{K} \boldsymbol{\gamma}$$

for $\boldsymbol{\gamma}$.

Univariate Smoothing

- This leads to the penalized least squares criterion

$$||\mathbf{y} - \mathbf{Z}\gamma||^2 + \lambda\gamma^\top \mathbf{K}\gamma$$

  and the penalized least squares estimate

$$\hat{\gamma} = (\mathbf{Z}^\top\mathbf{Z} + \lambda\mathbf{K})^{-1}\mathbf{Z}^\top\mathbf{y}.$$

- The smoother matrix is given by

$$\mathbf{S}_\lambda = \mathbf{Z}(\mathbf{Z}^\top\mathbf{Z} + \lambda\mathbf{K})^{-1}\mathbf{Z}^\top.$$

- The hat-matrix $\mathbf{H}$ in the linear model is an equivalent to $\mathbf{S}_\lambda$, therefore we can write

$$\hat{y} = \underset{n\times n}{\mathbf{L}}\,\mathbf{y},$$

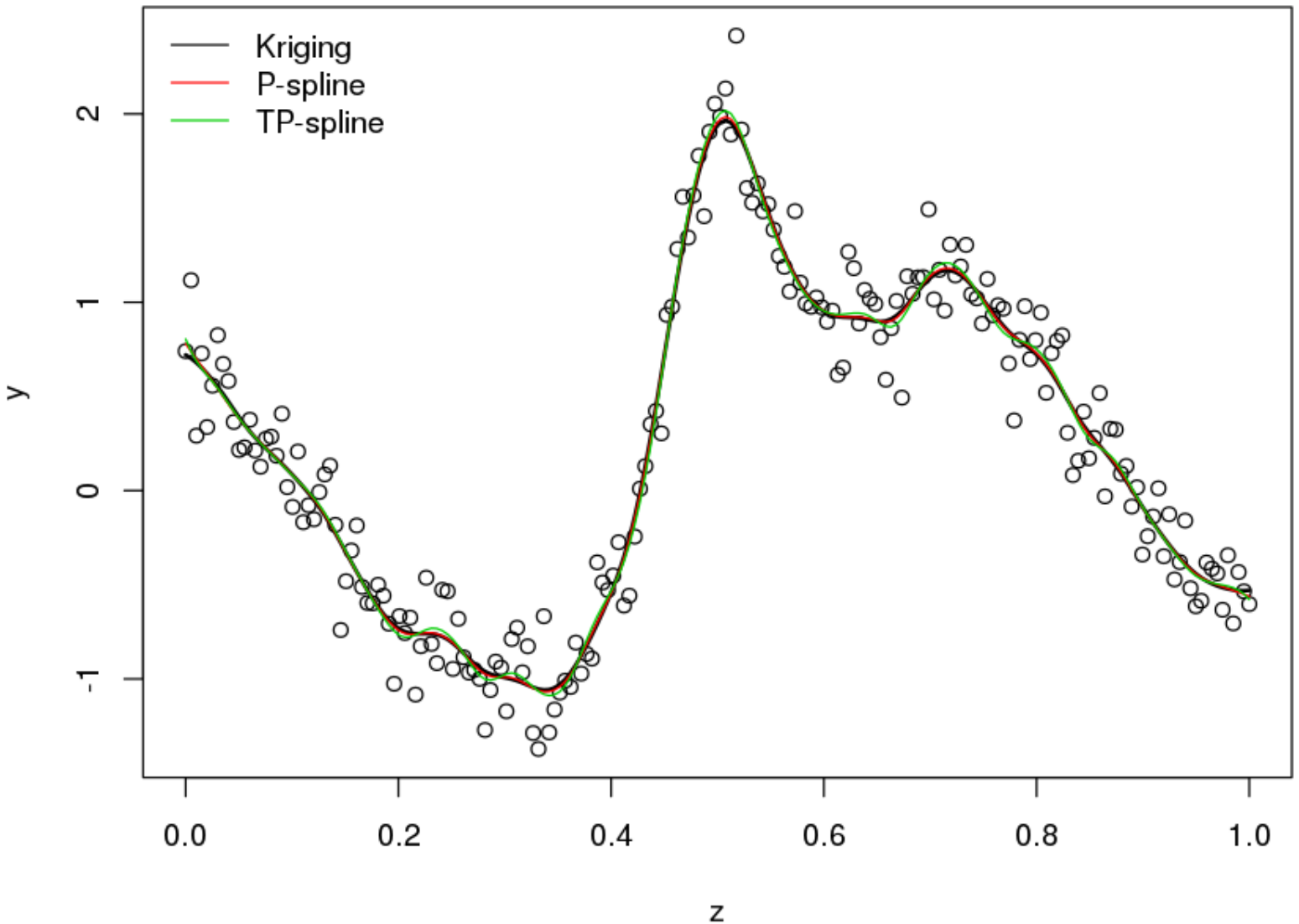
  which we call the class of *linear smoothers*.

Univariate Smoothing

Choosing a scatterplot smoother

- How should one decide between these various choices?
- First, it should be noted that there are approximate mathematical equivalences between the various linear smoothers.
- This means that, for fixed degrees of freedom, the fits from two different scatterplot smoothing methods are approximately the same.

Univariate Smoothing



Univariate Smoothing

Choosing a scatterplot smoother

1. *Convenience*. Is it available on the analyst's favorite computer package?
2. *Implementability*. If not immediately available, how easy is it to implement in the analyst's favorite programming language?
3. *Flexibility*. Is the smoother able to handle a wide range of types of relationships that may exist among the variables of interest?
4. *Simplicity*. Is it easy to understand how the technique processes the data to obtain answers?

Univariate Smoothing

5. *Tractability*. Is it easy to analyze the mathematical properties of the technique?
6. *Reliability*. Can the answers be trusted?
7. *Efficiency*. Does the technique use the data in the most efficient way?
8. *Extendibility*. Is the technique easily extended to more complicated settings?

Univariate Smoothing

Confidence intervals and confidence bands

- In addition to point estimates for $f(z)$, information on the variability of the estimate or corresponding confidence intervals is typically required in practice.
- When first considering the case of function estimate at a fixed covariate value z, using linear smoothers, a variance formula is given by

$$Var(\hat{f}(z)) = \sigma^2 \mathbf{s}(z)^\top \mathbf{s}(z),$$

where

$$\mathbf{s}(z)^\top = \mathbf{z}^\top (\mathbf{Z}^\top \mathbf{Z} + \lambda \mathbf{K})^{-1} \mathbf{Z}^\top$$

and

$$\hat{f}(z) = \mathbf{s}(z)^\top \mathbf{y}.$$

Univariate Smoothing

- If we additionally assume normally distributed errors, a $(1-\alpha)$-confidence interval is given by

$$\hat{f}(z) \pm z_{1-\alpha/2}\sigma\sqrt{\mathbf{s}(z)^\top \mathbf{s}(z)},$$

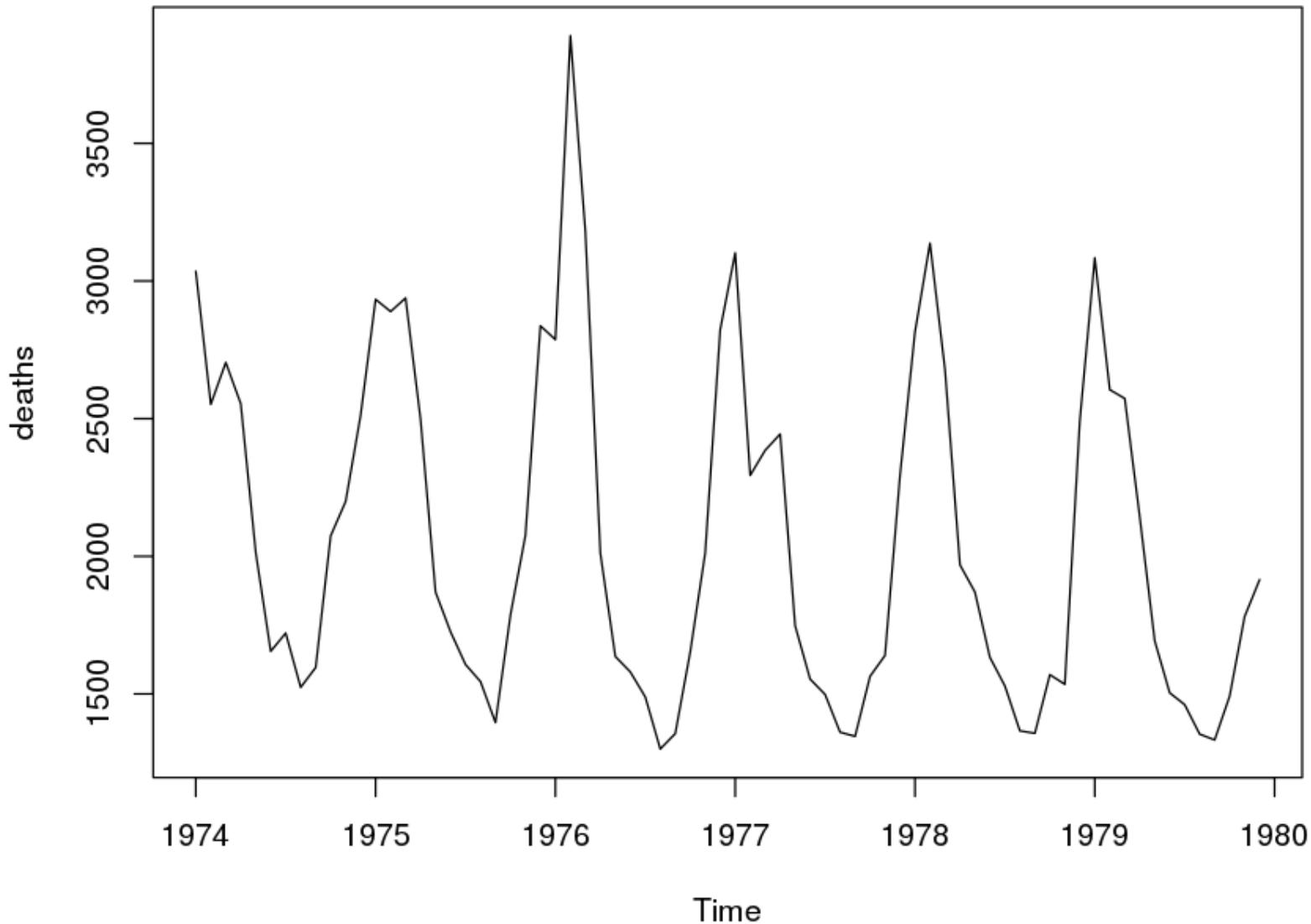
  where $z_{1-\alpha/2}$ is the $(1-\alpha/2)$-quantile of the standard normal distribution.
- This construction is based on the assumption that

$$\hat{f}(z) - f(z) \overset{a}{\sim} N\left(0, \sigma^2 \mathbf{s}(z)^\top \mathbf{s}(z)\right).$$

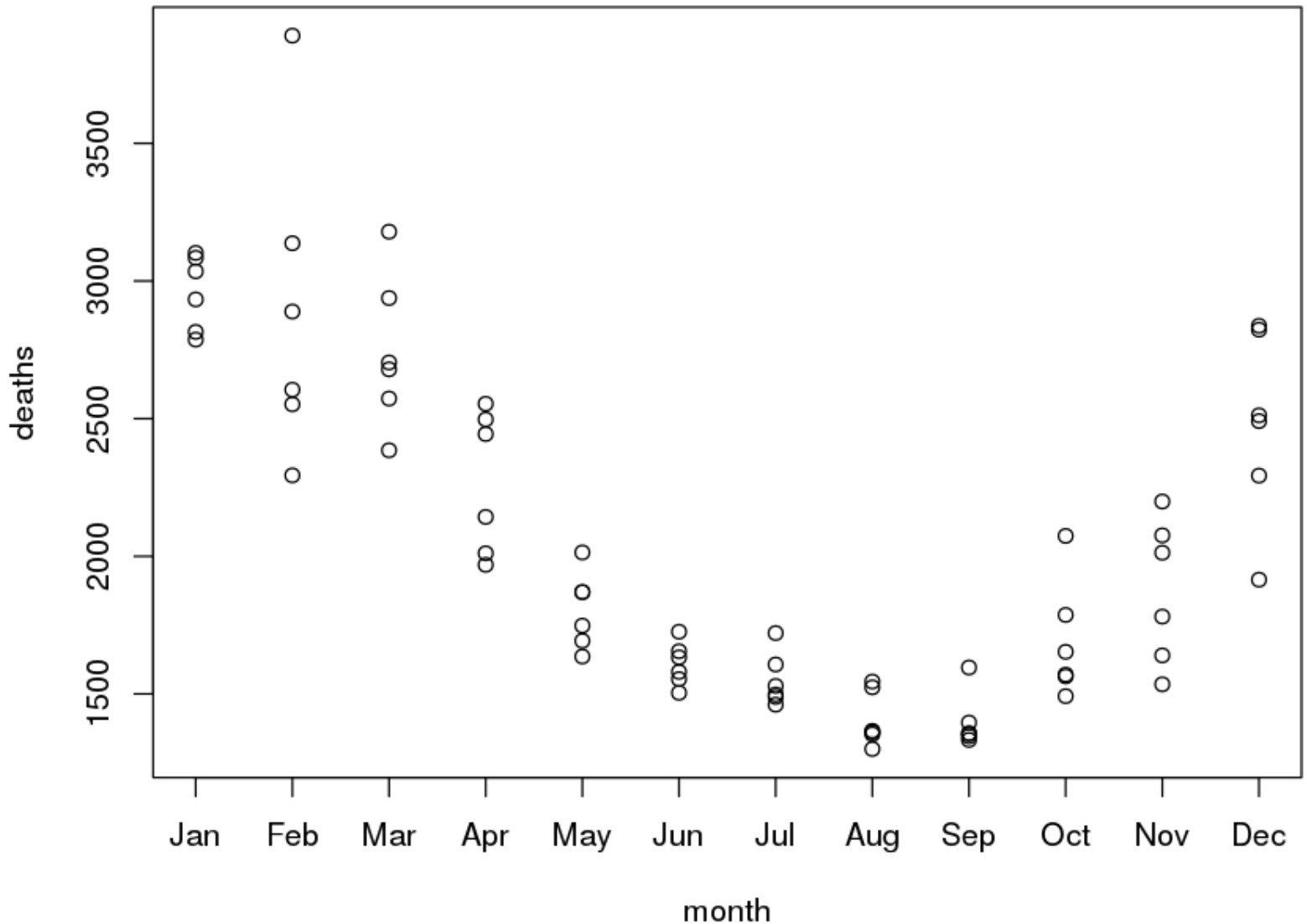
Univariate Smoothing

- While both the form of the variance of the estimate $\hat{f}(z)$ and the normal distribution can be derived from the distributional assumption for the error terms, we also have to assume that $\hat{f}(z)$ is (approximately) unbiased.
- In general, this property will only be fulfilled asymptotically and therefore the coverage probability of the constructed confidence intervals is also only asymptotic in nature.
- If the error terms are not assumed to be normally distributed, the normal distribution for the estimates will also hold only asymptotically.

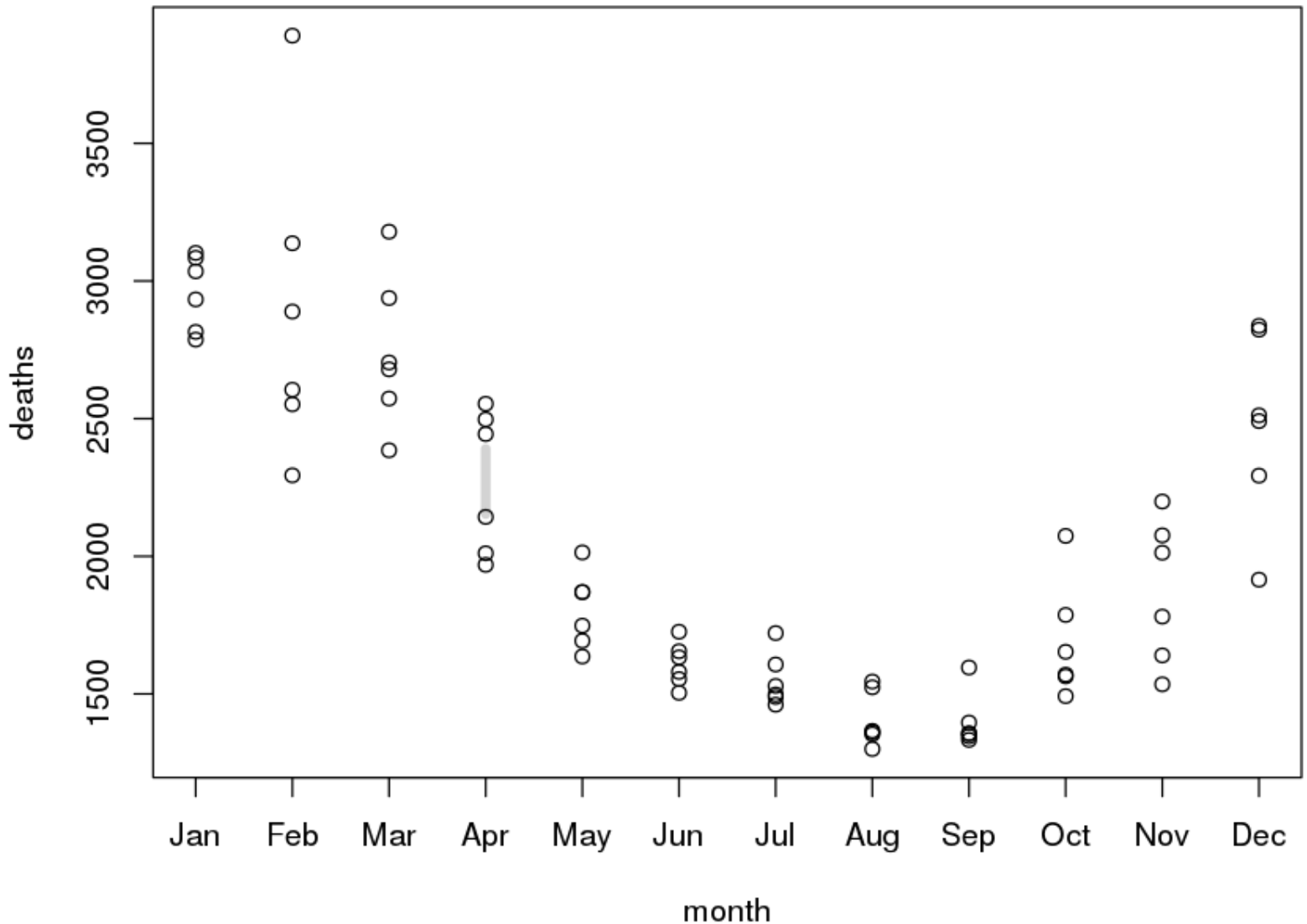
Univariate Smoothing



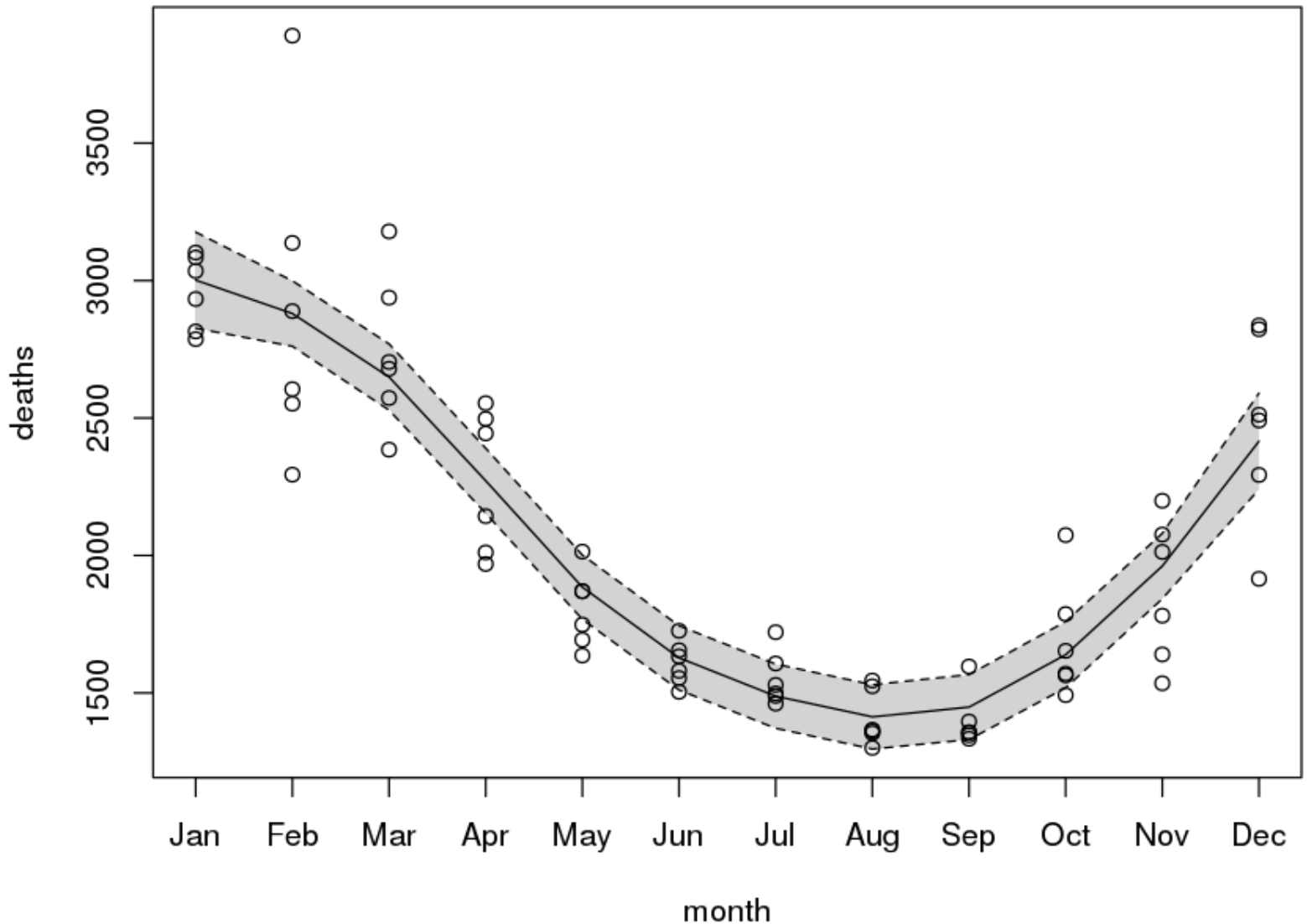
Univariate Smoothing



Univariate Smoothing



Univariate Smoothing



Univariate Smoothing

- We need to keep in mind that the confidence intervals have been constructed for a single covariate value and therefore the coverage probability is only valid pointwise and not simultaneously for the complete function.
- More specifically, for a pointwise level $(1 - \alpha)$ confidence interval $[L(z), U(z)]$, we have

$$P\left(L(z) \leq f(z) \leq U(z)\right) \geq 1 - \alpha \text{ for a single given } z,$$

  i.e. the probability that the (random) interval covers the (fixed) true function value is at least $1 - \alpha$.
- It is critical that the expression only applies for a chosen, typical value of z, and not for several covariate values at the same time.

Univariate Smoothing

- However, often such simultaneous expressions are of interest, and thus the construction of (simultaneous) *confidence bands* is necessary.
- Such a simultaneous confidence band $[L(z), U(z)]$ of level $1 - \alpha$ should fulfill

$$P\left(L(z) \leq f(z) \leq U(z) \text{ for all } z \in \{z_1, \ldots, z_n\}\right) \geq 1 - \alpha.$$

- In this case, the coverage probability does not apply pointwise, but rather simultaneously for all covariate values in $\{z_1, \ldots, z_n\}$.

Univariate Smoothing

- An option is to derive simulation based simultaneous confidence bands.
- First, we determine the covariance matrix of the vector $\hat{\mathbf{f}} = (\hat{f}(z_1), \ldots, \hat{f}(z_n))^\top$, which can be written as a linear smoother $\hat{\mathbf{f}} = \mathbf{S}\mathbf{y}$, where $\mathbf{S}$ consists of the vectors $\mathbf{s}(z_1), \ldots, \mathbf{s}(z_n)$.
- The covariance matrix can then be expressed as

$$Cov(\hat{\mathbf{f}}) = \sigma^2 \mathbf{S}\mathbf{S}^\top$$

and therefore

$$\hat{\mathbf{f}} - \mathbf{f} \overset{a}{\sim} N\left(\mathbf{0}, \sigma^2 \mathbf{S}\mathbf{S}^\top\right). \tag{1}$$

Univariate Smoothing

- From this joint distribution we obtain a simultaneous confidence band with level $1 - \alpha$ as

$$\hat{f}(z_i) \pm m_{1-\alpha} \sigma \sqrt{\mathbf{s}(z_i)^\top \mathbf{s}(z_i)},$$

  where $m_{1-\alpha}$ defines the $(1 - \alpha)$-quantile of the distribution of the random variable

$$\max_{1 \leq i \leq n} \left| \frac{\hat{f}(z_i) - f(z_i)}{\sigma \sqrt{\mathbf{s}(z_i)^\top \mathbf{s}(z_i)}} \right| . \tag{2}$$

- Since the distribution of this random variable and corresponding quantiles are difficult to obtain analytically, we approximate the distribution using simulation.
- To do so, we first draw N random vectors from the asymptotic distribution (1) and then calculate the respective N realizations from (2). If N is chosen large enough, we obtain an estimate for $m_{1-\alpha}$ based on the corresponding empirical $(1 - \alpha)$-quantile.

Univariate Smoothing

```
R> smooth.fit <- function(y, z, bsFun, alpha = 0.05,
+    type = "sim", N = 10000, ...) {
+    yn <- deparse(substitute(y), backtick = TRUE,
+      width.cutoff = 500)
+    zn <- deparse(substitute(z), backtick = TRUE,
+      width.cutoff = 500)
+
+    D <- bsFun(z, ...)
+    Z <- D$B
+    K <- D$K
+    lambda <- find.hyper(y, Z, K, ...)
+    S <- Smat(Z, K, lambda)
+    fit <- S %*% y
+    n <- length(y)
+    StS <- S %*% t(S)
+    e <- y - fit
+    sigma <- drop(sqrt(t(e) %*% e / (n - sum(diag(S)))))
+    sigmaStS <- sigma * sqrt(diag(StS))
+
+  ## continued on next page
```

Univariate Smoothing

```
+  ## continued from previous page
+
+    if(!is.na(pmatch(type, "point"))) {
+      se.fit <- qnorm(1 - alpha / 2) * sigmaStS
+    }
+    if(!is.na(pmatch(type, "sim"))) {
+      fs <- mvrnorm(N, mu = rep(0, n), Sigma = sigma^2 * StS)
+      m <- rep(0, N)
+      for(i in 1:N) {
+        m[i] <- max(abs(fs[i, ] / sigmaStS))
+      }
+      m <- quantile(m, probs = 1 - alpha)
+      se.fit <- m * sigmaStS
+    }
+
+    rval <- data.frame("z" = z, "y" = y,
+      "fit" = fit, "se.fit" = se.fit)
+    attr(rval, "labels") <- c(yn, zn)
+    class(rval) <- c("smooth.fit", "data.frame")
+    rval
+  }
```

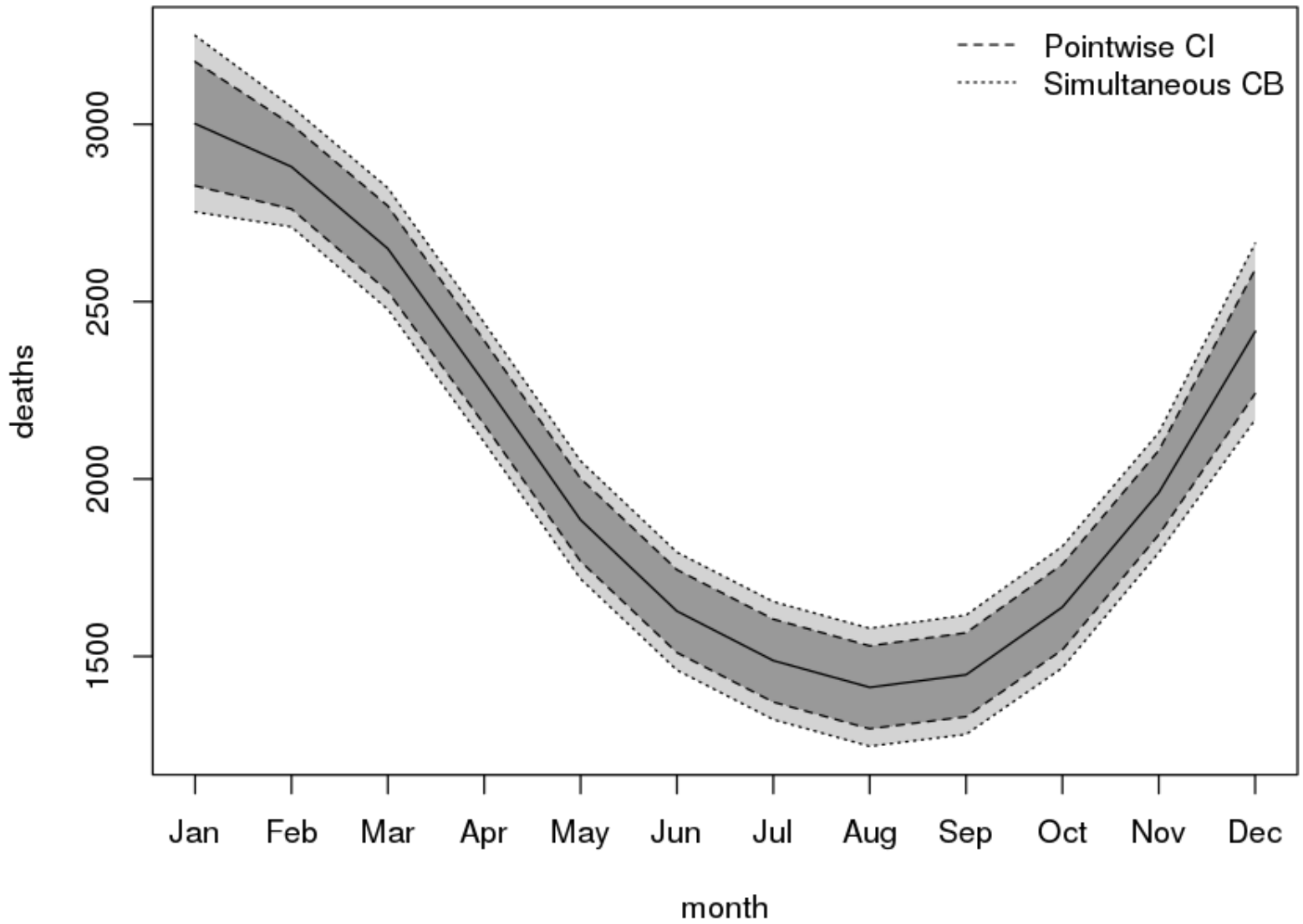
Univariate Smoothing

```
R> plot.smooth.fit <- function(x, residuals = TRUE,
+    se.lwd = 1, se.lty = 2, se.bg = "lightgray", add = FALSE, ylim = NULL, ...) {
+    x <- x[order(x[["z"]]), ]
+    if(!add) {
+      if(is.null(ylim)) {
+        ylim <- with(x, range(c(fit - se.fit, fit + se.fit)))
+        if(residuals)
+          ylim <- with(x, range(c(ylim, y)))
+      }
+      with(x, plot(z, y, type = "n",
+        ylab = attr(x, "labels")[1], xlab = attr(x, "labels")[2],
+        ylim = ylim, ...))
+    }
+    if(!is.null(se.bg)) {
+      se.poly.x <- with(x, c(z, rev(z)))
+      se.poly.y <- with(x, c((fit - se.fit), rev((fit + se.fit))))
+      polygon(se.poly.x, se.poly.y, col = se.bg, border = NA)
+    }
+    if(residuals)
+      with(x, points(z, y))
+    with(x, lines(fit ~ z))
+    with(x, lines((fit - se.fit) ~ z, lwd = se.lwd, lty = se.lty))
+    with(x, lines((fit + se.fit) ~ z, lwd = se.lwd, lty = se.lty))
+    invisible(NULL)
+  }
```

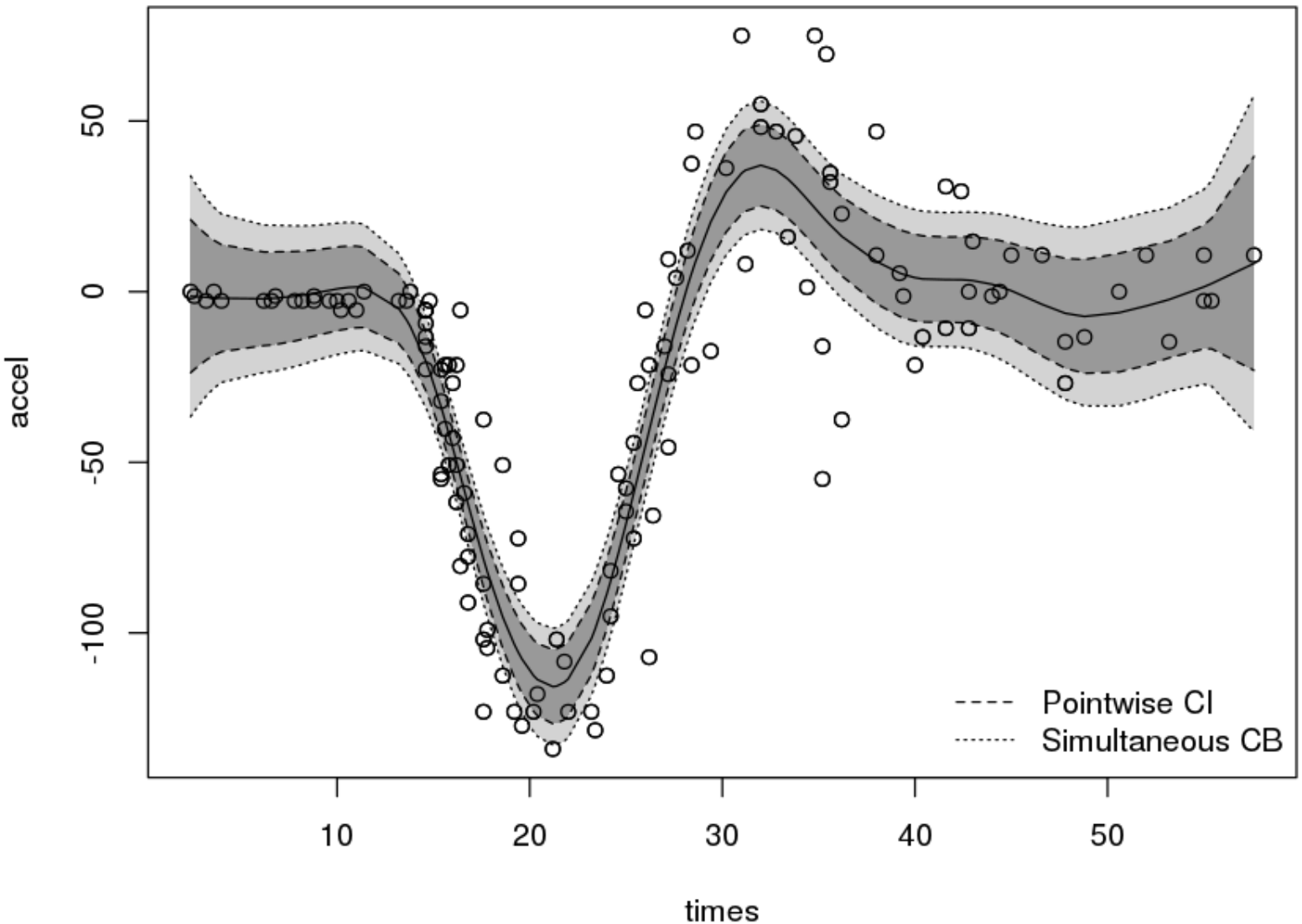
Univariate Smoothing

```
R> fit.ps.point <- with(dat,
+    smooth.fit(deaths, month, psDesign, type = "point"))
R> fit.ps.sim <- with(dat,
+    smooth.fit(deaths, month, psDesign, type = "sim"))

R> plot(fit.ps.sim, residuals = FALSE,
+    se.lty = 3, se.bg = NULL, se.bg = "lightgray",
+    axes = FALSE)
R> plot(fit.ps.point, residuals = FALSE,
+    se.lty = 2, se.bg = NULL, se.bg = "gray60",
+    add = TRUE)

R> axis(1, at = 1:12, labels = month.abb)
R> axis(2)
R> box()
R> legend("topright", c("Pointwise CI", "Simultaneous CB"),
+    lwd = 1, lty = 2:3, box.col = NA, bg = NA)
```

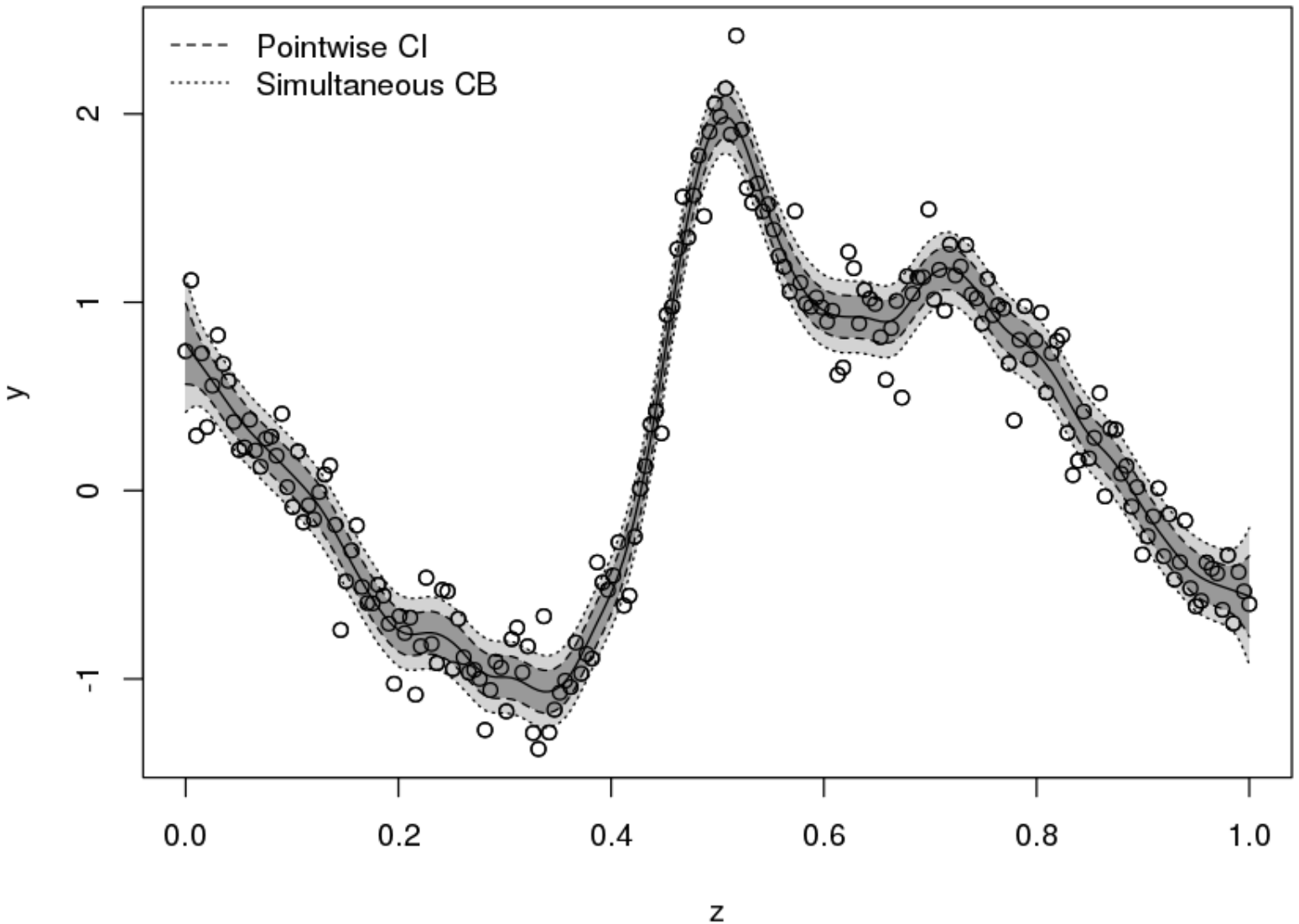
Univariate Smoothing



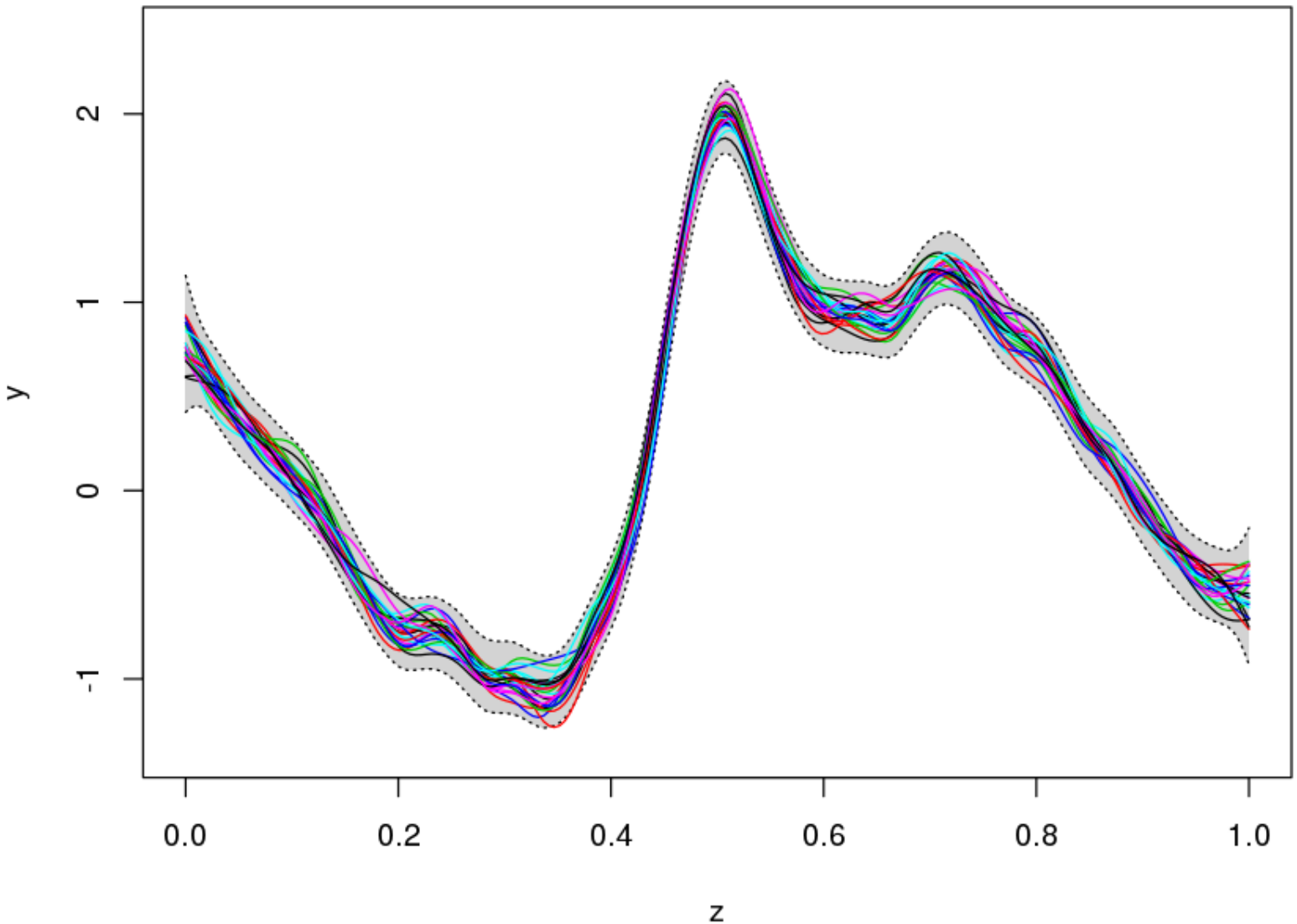
Univariate Smoothing



Univariate Smoothing



Univariate Smoothing



Univariate Smoothing

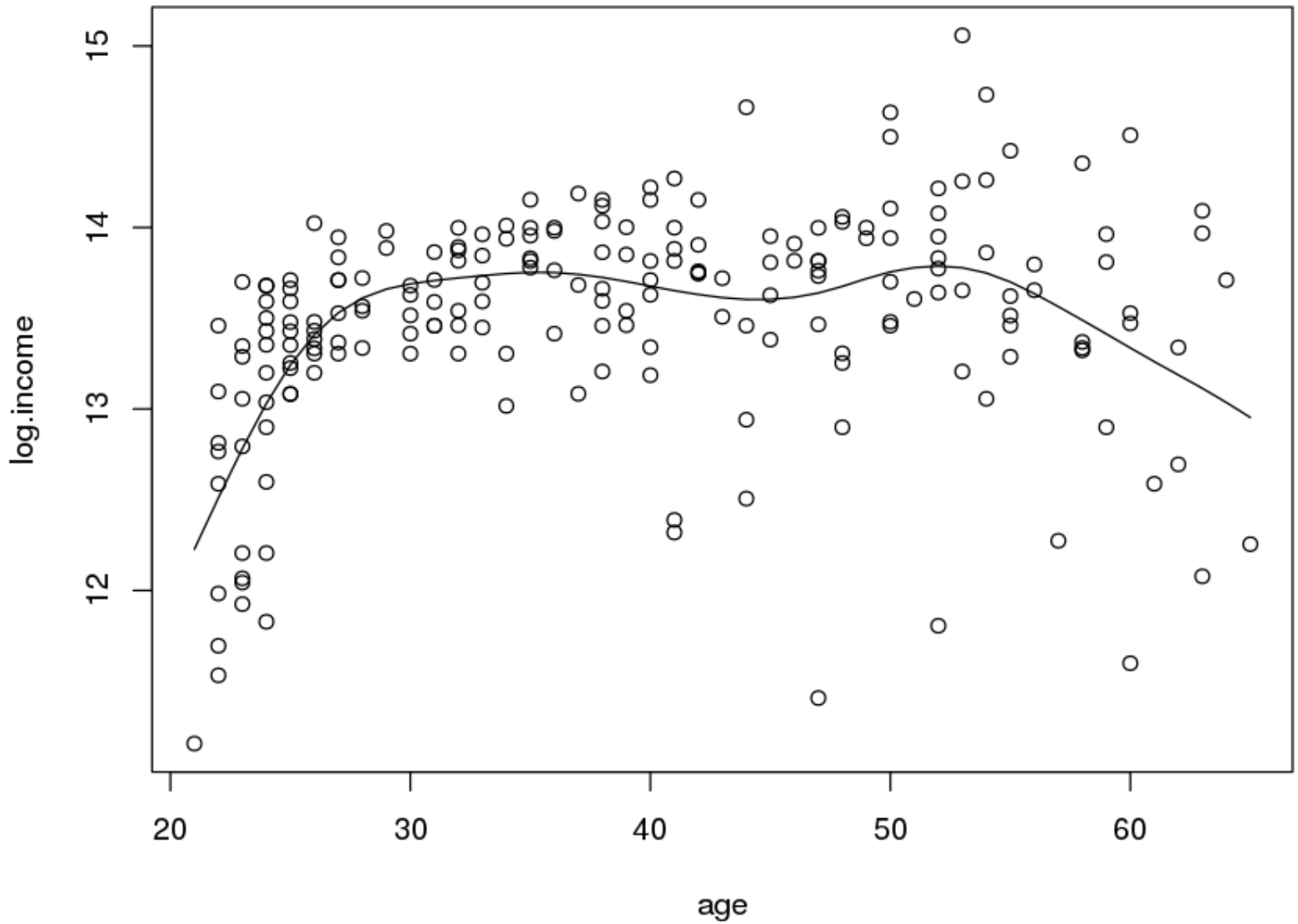
Inference using first derivatives

- The derivatives of the regression function f can be of interest as well as f itself. Consider the simple linear model

$$E(y|x) = \beta_0 + \beta_1 x.$$

- Generally, there is less interest in β_0 than in β_1, because β_1 is the effect of x on y as measured by a rate of change – that is, it is the derivative of $E(y)$ with respect to x.
- In contrast, the additive constant β_0 tells us nothing about the effect of x on y.
- When we plot (nonlinear) $\hat{f}(z)$, our eye often looks at its slope and how it changes with z. In our mind we note where the slope is positive, where it is negative, and where it is essentially zero.

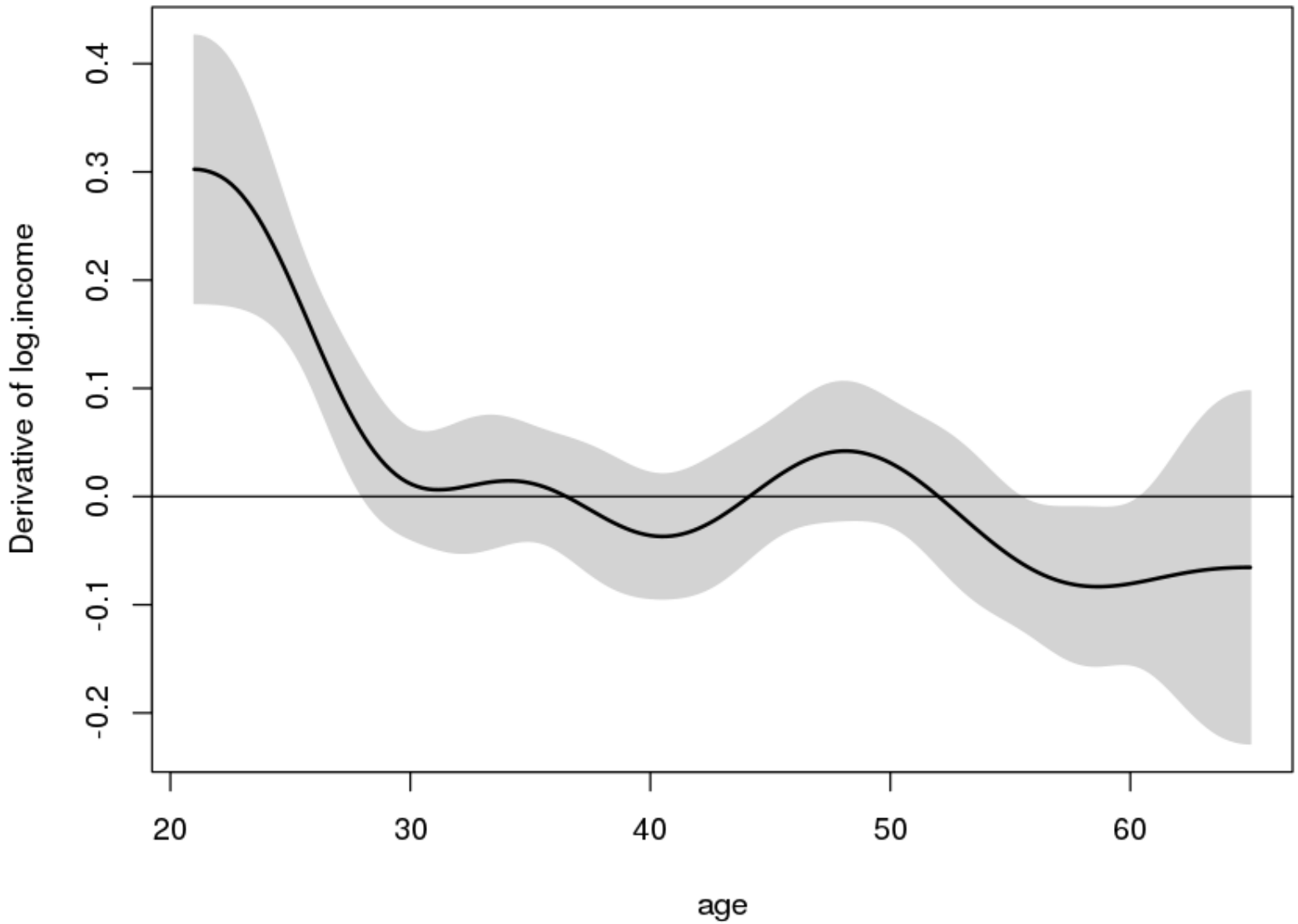
Univariate Smoothing



Univariate Smoothing

- The interesting features of this plot can be expressed in terms of the first derivative $f'(\texttt{age})$.
- It appears that f' is large and positive at young ages, meaning that young workers see their incomes rise rapidly. But eventually f reaches a plateau, suggesting that middle-age workers do not experience much (if any) rise in income.
- There is some suggestion in the data that f' is negative at older ages, meaning that older workers actually may see a drop in income.

Univariate Smoothing



Univariate Smoothing

- The graphic helps us answer some basic questions:
  - How fast is income rising at young ages?
  - When does income start to plateau?
  - Does income really decline at older ages?
- Note that the confidence intervals are based on a homoscedasticity assumption although the data suggests higher variance of `log.income` at higher ages.
- The plot is also helpful in testing the existance of certain features such as bumbs and dips.
- For example, the dip for workers in their mid-40s corresponds to a mid-career decline in income?

Univariate Smoothing

Prediction intervals

- Suppose we wish to predict the value of y at a new data point, z_0.
- That is, we wish to predict $\mathbf{z}_0^\top \gamma + \varepsilon$ when z_0 is known.
- The predictor is $\hat{f}(z_0) = \mathbf{z}_0^\top \hat{\gamma}$, which predicts $\mathbf{z}_0^\top \gamma$ by $\mathbf{z}_0^\top \hat{\gamma}$ and ε by 0 – that is, both quantities are predicted by estimates of their expected values.
- Uncertainty in the prediction has two causes: γ will differ from $\hat{\gamma}$, and ε will not equal 0.
- Since the new ε is independent of $\hat{\gamma}$, it follows that

$$Var(f(z_0) - \hat{f}(z_0)) = Var(\varepsilon) + Var(\hat{f}(z_0)).$$

Univariate Smoothing

- From the previous we get

$$Var(\hat{f}(z_0)) = \sigma^2 \mathbf{s}(z_0)^\top \mathbf{s}(z_0).$$

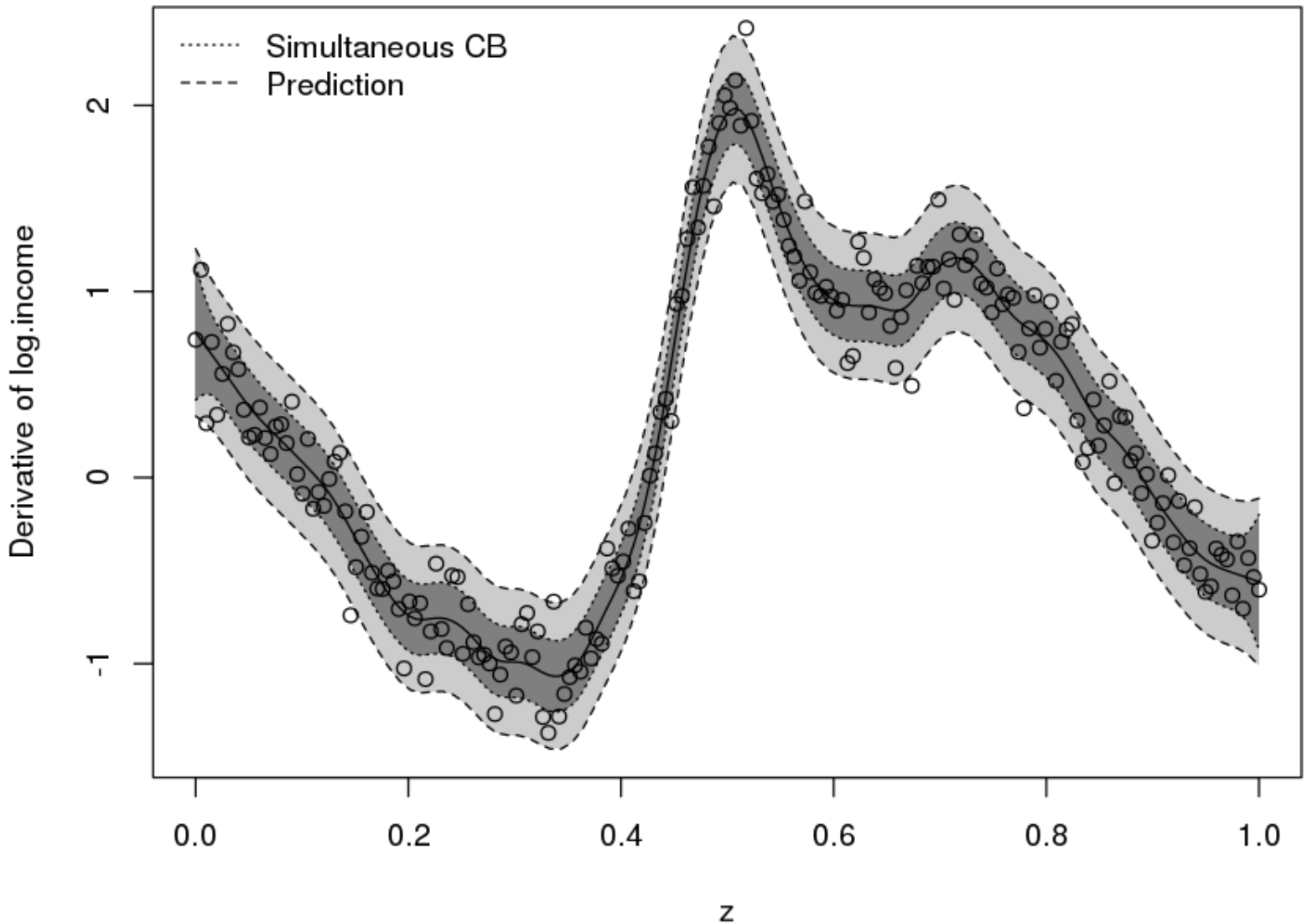
- Hence, we have

$$\sigma \sqrt{1 + \mathbf{s}(z_0)^\top \mathbf{s}(z_0)}.$$

- A $(1 - \alpha)$-confidence interval is then given by

$$\hat{f}(z_0) \pm z_{1-\alpha/2} \sigma \sqrt{1 + \mathbf{s}(z_0)^\top \mathbf{s}(z_0)}.$$

Univariate Smoothing



Univariate Smoothing

P-values

- Consider the problem of testing whether γ is identically zero.
- The frequentist covariance matrix for γ is given by

$$\boldsymbol{\Sigma}_\gamma = \sigma^2 \left(\mathbf{Z}^\top \mathbf{Z} + \lambda \mathbf{K} \right)^{-1}.$$

- Under the null hypothesis we have $\gamma = \mathbf{0}$ and so

$$\hat{\gamma} \overset{a}{\sim} N(\mathbf{0}, \boldsymbol{\Sigma}_\gamma).$$

- From this it follows that, if $\boldsymbol{\Sigma}_\gamma$ is of full rank, then under the null hypothesis

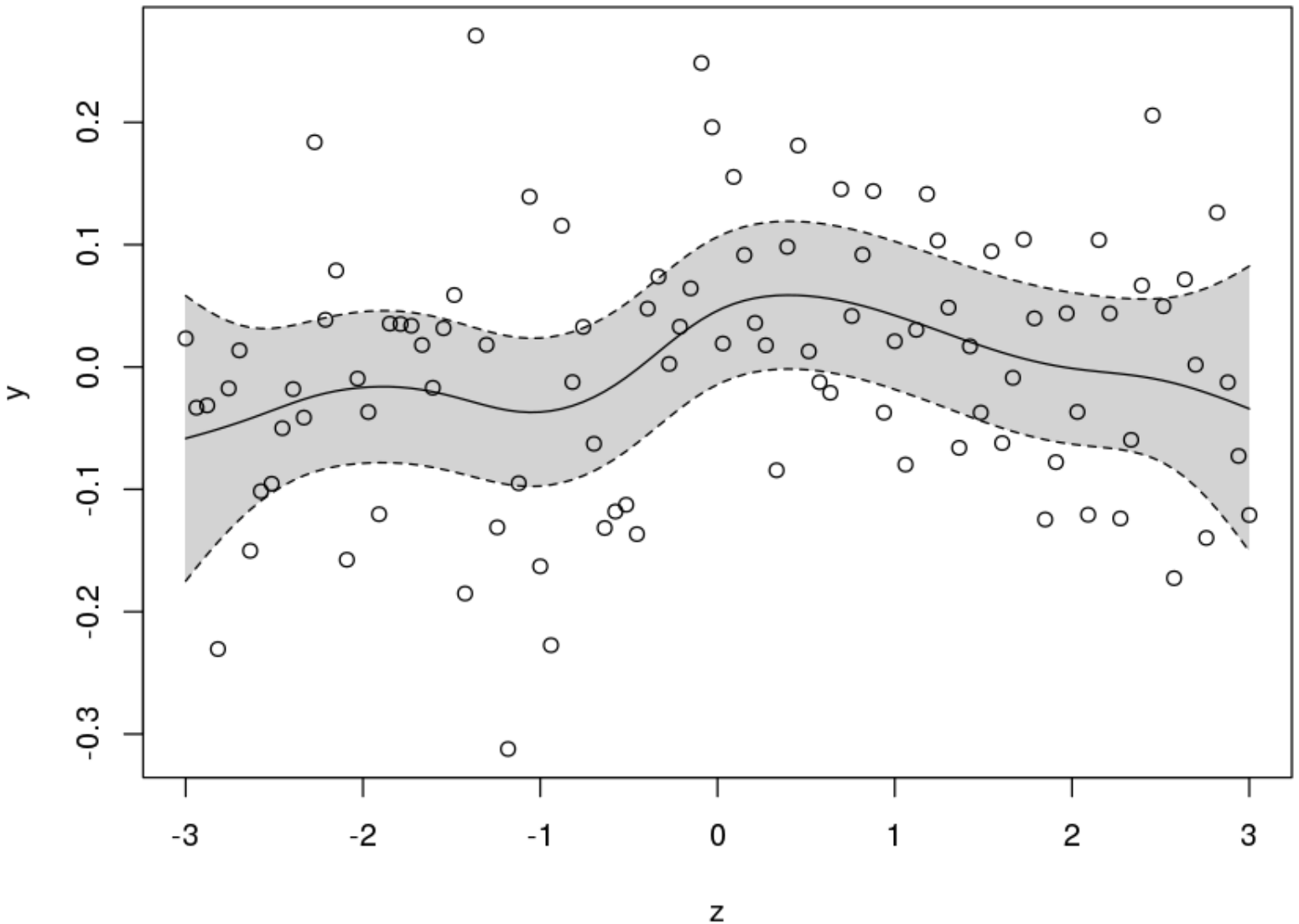
$$\hat{\gamma}^\top \boldsymbol{\Sigma}_\gamma^{-1} \hat{\gamma} \overset{a}{\sim} \chi^2_p.$$

- Specifically the p-value for the test that $\gamma = \mathbf{0}$ is

$$P\left(X > \hat{\gamma}^\top \boldsymbol{\Sigma}_\gamma^{-1} \hat{\gamma} \right),$$

where $X \sim \chi^2_p$.

Univariate Smoothing



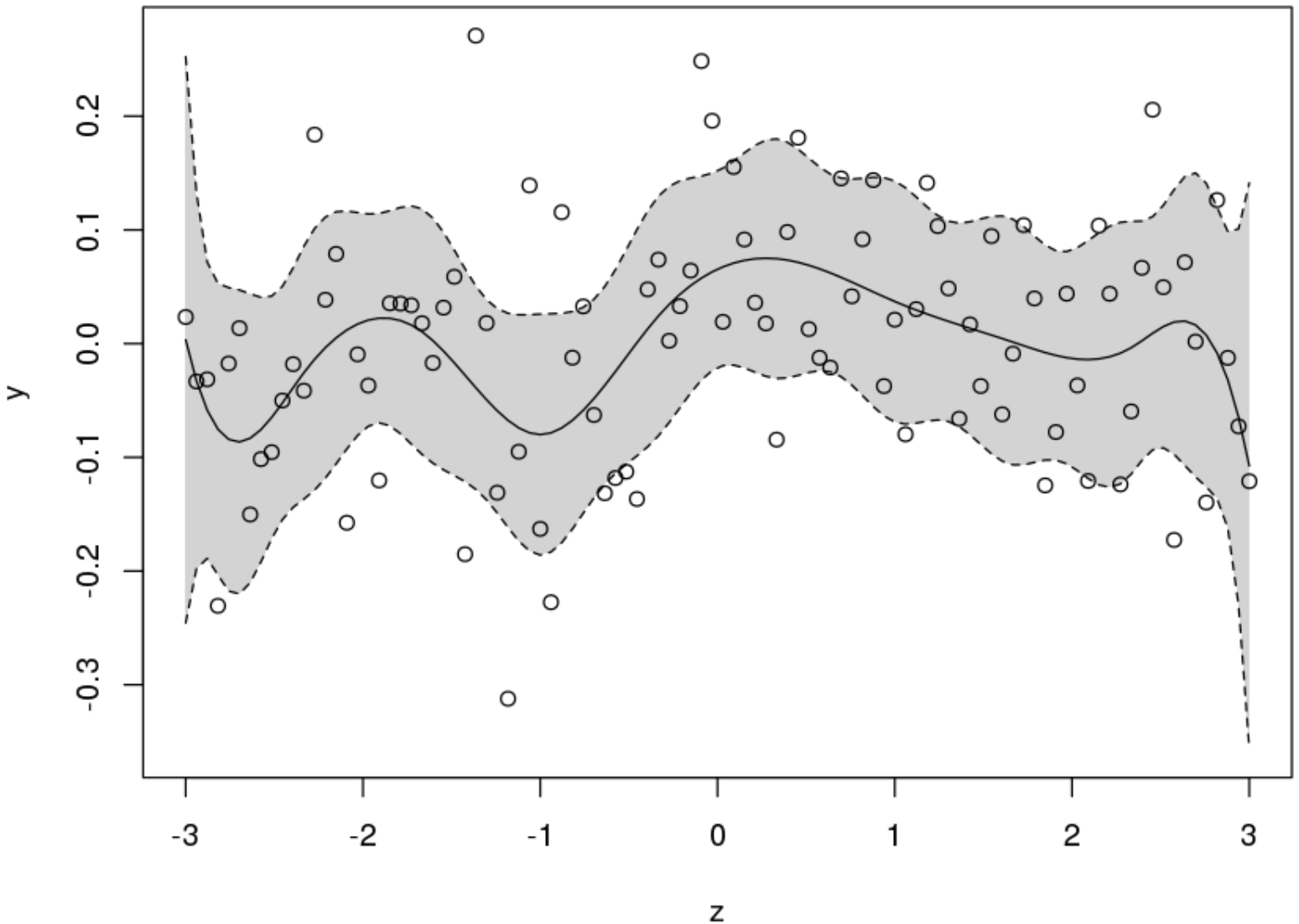
Univariate Smoothing

```
R> set.seed(111)
R> n <- 100
R> z <- seq(-3, 3, length = n)
R> y <- sin(z) * 0.001 + rnorm(n, sd = 0.1)

R> fit <- smooth.fit(y, z, psDesign, type = "sim")

R> lambda <- attr(fit, "lambda")
R> gamma <- attr(fit, "coefficients")
R> sigma <- attr(fit, "sigma")
R> p <- attr(fit, "p")
R> D <- psDesign(z)
R> Z <- D$B
R> K <- D$K

R> Sigma <- 1/sigma^2 * (t(Z) %*% Z + lambda * K)
R> X <- t(gamma) %*% Sigma %*% gamma
R> pvalue <- drop(1 - pchisq(X, p))
R> pvalue

[1] 0.09314607
```

Univariate Smoothing

- The p-values, calculated in this manner, behave correctly for un-penalized models, and reasonably well for models with known smoothing parameters.
- When smoothing paramaters have been estimated, the p-values are typically lower than they should be when the null is true, meaning that the tests reject the null too readily.
- This is because smoothing parameter uncertainty has been neclected for testing.
- In practical terms, if these p-values give a clear cut result, then it is usually safe to rely on it, but p-values around a threshold for accepting or rejecting must be treated with caution.
- If hypothesis testing is a key aim of an analysis, then it may sometimes be preferable to base tests on overspecified un-penalized models, so that although the fits may be excessively wiggly, the p-values will be correct.

Univariate Smoothing



Univariate Smoothing

```
R> Z <- bsDesign(z, knots = 10)
R> gamma <- solve(t(Z) %*% Z) %*% t(Z) %*% y
R> e <- y - Z %*% gamma
R> edf <- ncol(Z)
R> sigma <- sqrt(drop(t(e) %*% e / (n - edf)))

R> Sigma <- 1/sigma^2 * (t(Z) %*% Z)
R> X <- t(gamma) %*% Sigma %*% gamma
R> pvalue <- drop(1 - pchisq(X, edf + 1))
R> pvalue

[1] 0.09470576
```

Univariate Smoothing

The R package mgcv

- All (and a lot more features) of the discussed are implemented in the recommended R package **mgcv**.
- Within **mgcv**, users may utilize a number of different smoother, but also implement their own basis functions and penalties!
- The main model fitting function is called `gam()` and has arguments

```
gam(formula, family = gaussian(), data = list(),
  weights = NULL, subset = NULL, ...)
```

Univariate Smoothing

- Smooth terms are included into the model formula by the constructor function `s()`

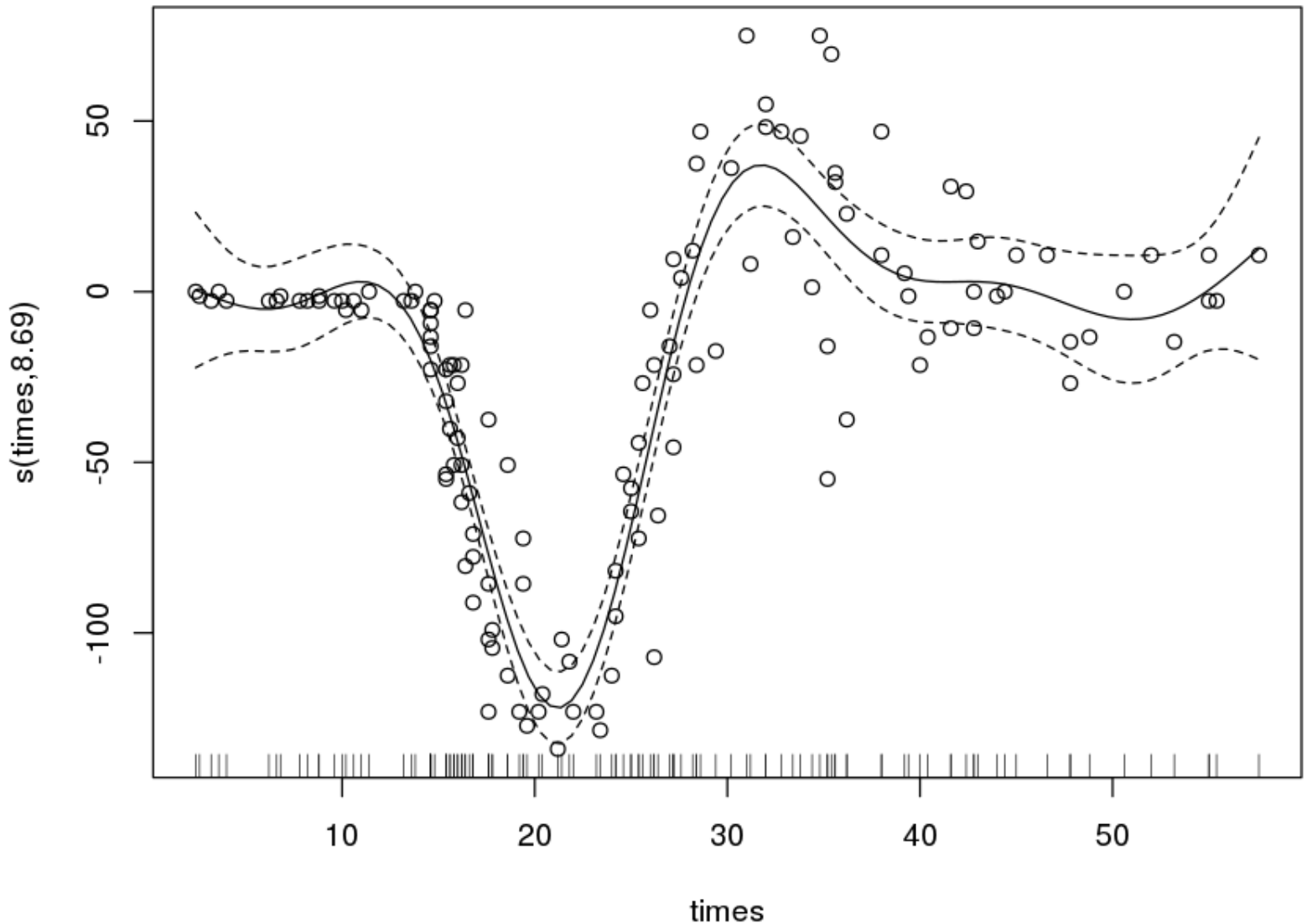
```
s(..., k = -1, fx = FALSE, bs = "tp", m = NA,
  by = NA, xt = NULL, id = NULL, sp = NULL)
```

- The function does not return an evaluated design or penalty matrix, it simply returns a smooth specification object of class "`xx.smooth.spec`", where "`xx`" is the basis type specified in argument `bs`, e.g. `bs = "ps"` is a P-spline term.
- This object is then further processed by the generic function `smooth.construct()`, for which a method that is chosen by the `bs` type will construct the necessary matrices for estimation.

Univariate Smoothing

```
R> library("mgcv")
R> data("mcycle")

R> b <- gam(accel ~ s(times), data = mcycle)

R> AIC(b)

[1] 1218.889

R> BIC(b)

[1] 1249.797

R> plot(b, residuals = TRUE, pch = 1)
```

Univariate Smoothing



Univariate Smoothing

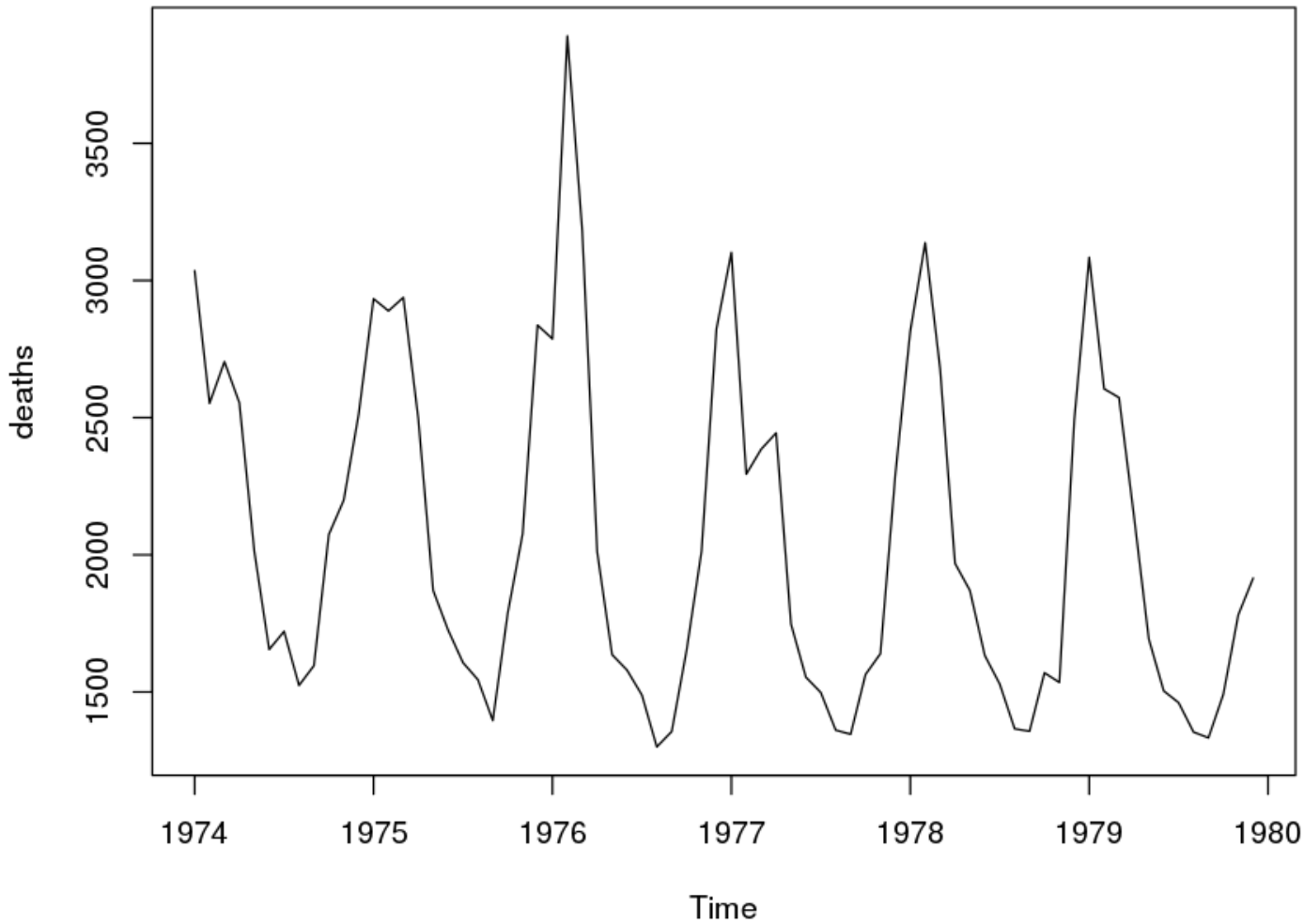
```
R> summary(b)

Family: gaussian
Link function: identity

Formula:
accel ~ s(times)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -25.546      1.951   -13.1   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Univariate Smoothing

```
Approximate significance of smooth terms:
            edf Ref.df    F p-value
s(times) 8.693  8.972 53.4  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =  0.783   Deviance explained = 79.8%
GCV score = 545.78  Scale est. = 506       n = 133
```

Univariate Smoothing

Example: Time series decomposition



Univariate Smoothing

```
R> library("MASS")
R> data("deaths")
R> class(deaths)

[1] "ts"

R> frequency(deaths)

[1] 12

R> d <- data.frame(
+    "deaths" = as.numeric(deaths),
+    "time" = as.numeric(time(deaths)),
+    "month" = as.integer(cycle(deaths))
+  )
```

Univariate Smoothing

```
R> head(d, n = 15)
```

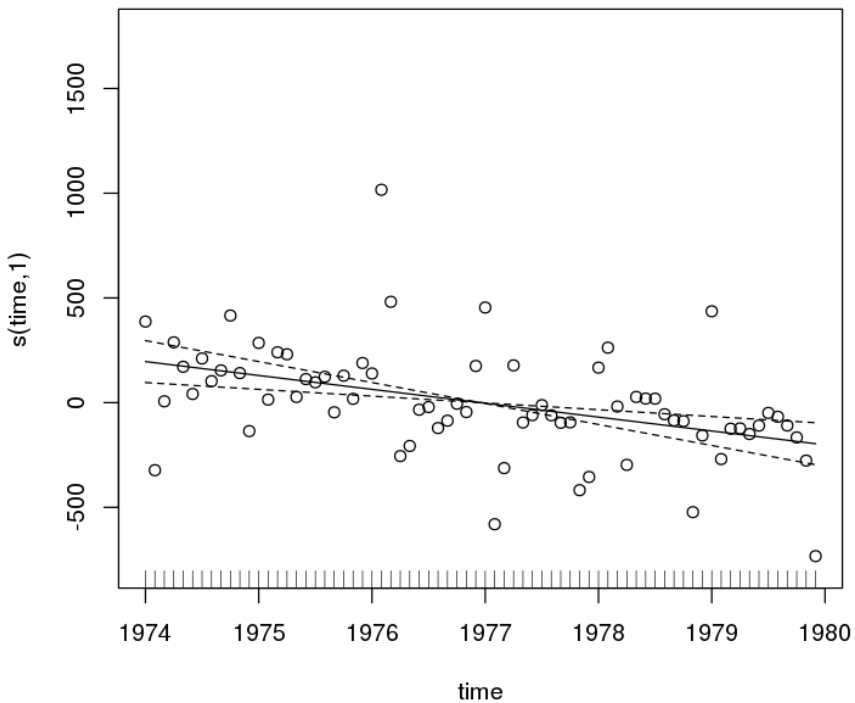
	deaths	time	month
1	3035	1974.000	1
2	2552	1974.083	2
3	2704	1974.167	3
4	2554	1974.250	4
5	2014	1974.333	5
6	1655	1974.417	6
7	1721	1974.500	7
8	1524	1974.583	8
9	1596	1974.667	9
10	2074	1974.750	10
11	2199	1974.833	11
12	2512	1974.917	12
13	2933	1975.000	1
14	2889	1975.083	2
15	2938	1975.167	3

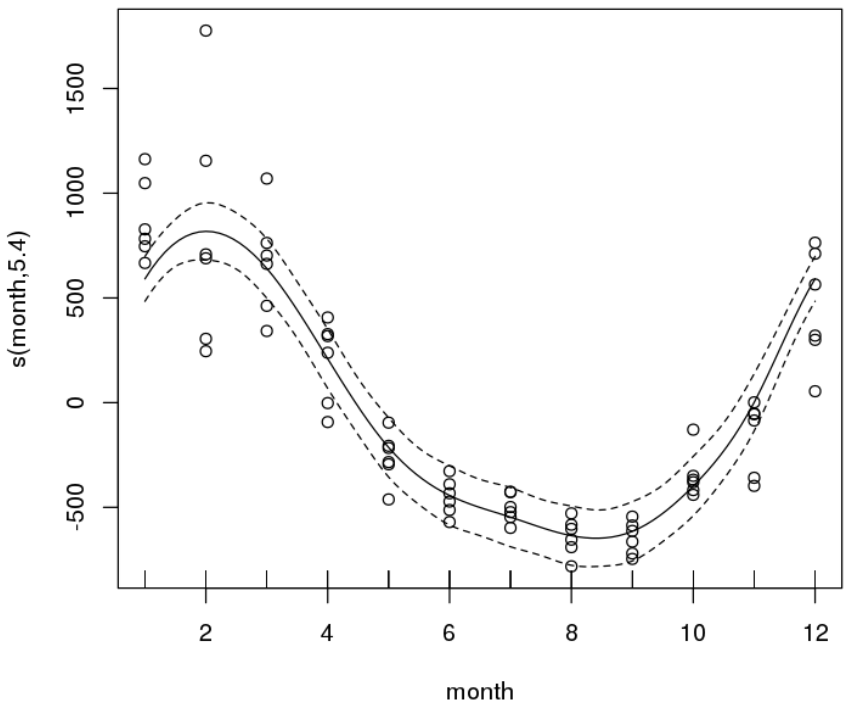
Univariate Smoothing

```
R> library("mgcv")

R> b <- gam(deaths ~ s(time, k = 40) + s(month, bs = "cc", k = 12),
+    data = d)

R> summary(b)

Family: gaussian
Link function: identity

Formula:
deaths ~ s(time, k = 40) + s(month, bs = "cc", k = 12)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2056.63      29.12   70.64   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
           edf Ref.df     F  p-value
s(time)  1.000      1 15.31 0.000215 ***
s(month) 5.405     10 33.12  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =  0.836   Deviance explained = 85.1%
GCV score =  68030  Scale est. = 61034     n = 72
```

Univariate Smoothing

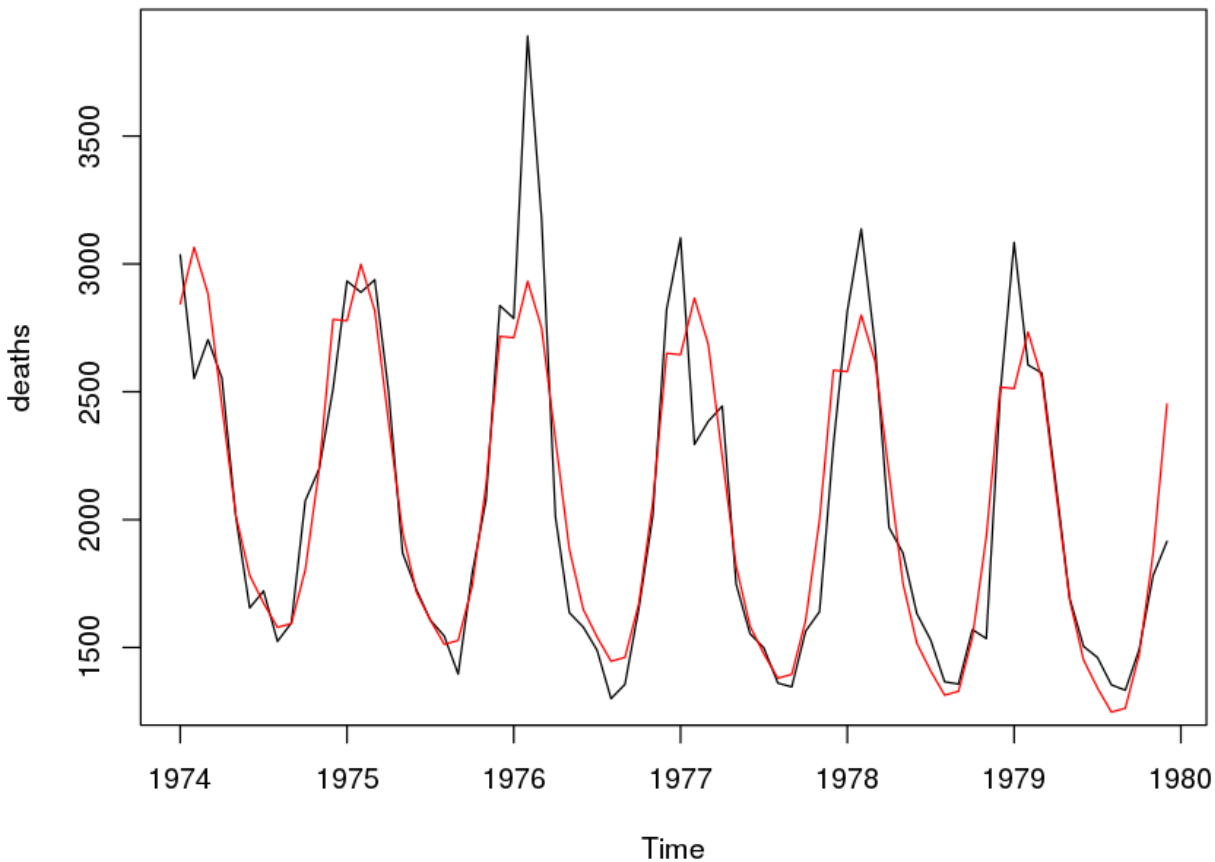
```
R> par(mfrow = c(1, 2), mar = c(4.1, 4.1, 1.1, 1.1))
R> plot(b, select = 1, residuals = TRUE, pch = 1)
R> plot(b, select = 2, residuals = TRUE, pch = 1)
```





Univariate Smoothing

```
R> f <- fitted(b)
R> f <- ts(f, start = start(deaths), frequency = frequency(deaths))
R> plot(deaths)
R> lines(f, col = "red")
```



Univariate Smoothing

```
R> p <- predict(b, type = "response", se.fit = TRUE)
R> f <- ts(p$fit,
+    start = start(deaths), frequency = frequency(deaths))
R> fup <- ts(p$fit + qnorm(0.975) * p$se.fit,
+    start = start(deaths), frequency = frequency(deaths))
R> flo <- ts(p$fit - qnorm(0.975) * p$se.fit,
+    start = start(deaths), frequency = frequency(deaths))
```

Univariate Smoothing

```
R> plot(deaths, ylim = range(c(deaths, f, fup, flo)))
R> lines(f, col = "red")
R> lines(fup, col = "red", lty = 2)
R> lines(flo, col = "red", lty = 2)
```