Topic 1 - Smoothing

Assignment 2: Bivariate Smoothing

Isin Altinkaya

09.11.22

Objectives

- Implement a spline smoother; use LOOCV for choosing the tuning parameter λ.
- Use Simpson's rule with breakpoints in the knots
- Use bsplinepen from fda
- Use matrix decomposition SVD
- Test with real data as well as simulated data

Theoretical background

$$SSE(s) = \sum_{i=1}^{n} (y_i - s(x_i))^2 + \lambda \int s''(t)^2 dt$$

Minimizing the SSE, we can define a piecewise 3rd-order polynomial $\sum_i \beta_i \phi_i$ with knots in x_i and basis functions ϕ_i, namely, a cubic spline.

We can define the smoother as $\hat{s} = \Phi \hat{\beta}$ where $\phi_{ij} = \phi_j(x_i)$.

$$SSE(s, \lambda) = (Y - \phi\beta)^T (Y - \phi\beta) + \lambda \beta^T \beta \Omega$$

$$\Omega_{ij} = \int \phi_i^{''}(t) \phi_j^{''}(t) dt$$

Theoretical background

This SSE is minimized by $\hat{\beta}$,

$$\hat{\beta} = (\Phi^T \Phi + \lambda \Omega)^{-1} \Phi^T Y$$

Inserting into the definition of smoother $\hat{s}$, we get the resulting smoother in the form of

$$\hat{s} = \Phi(\Phi^T \Phi + \lambda \Omega)^{-1} \Phi^T Y$$

Following the linear smoother definition, we can define the Leave-One-Out Cross-Validation as,

$$\mathrm{LOOCV} = \sum_{i=1}^{n} \left(\frac{y_i - \hat{f}_i}{1 - S_{ii}} \right)^2$$

Simpson's rule

With quadratic polynomials Simpson's rule leads to exact computation of Ω_{ij}.

$b - a$ is the vector of knot differences `diff(inner_knots)`

$$\int_a^b g_{ij}(z)\mathrm{d}z = \frac{b-a}{6}\left(g_{ij}(a) + 4g_{ij}\left(\frac{a+b}{2}\right) + g_{ij}(b)\right).$$

Data simulation

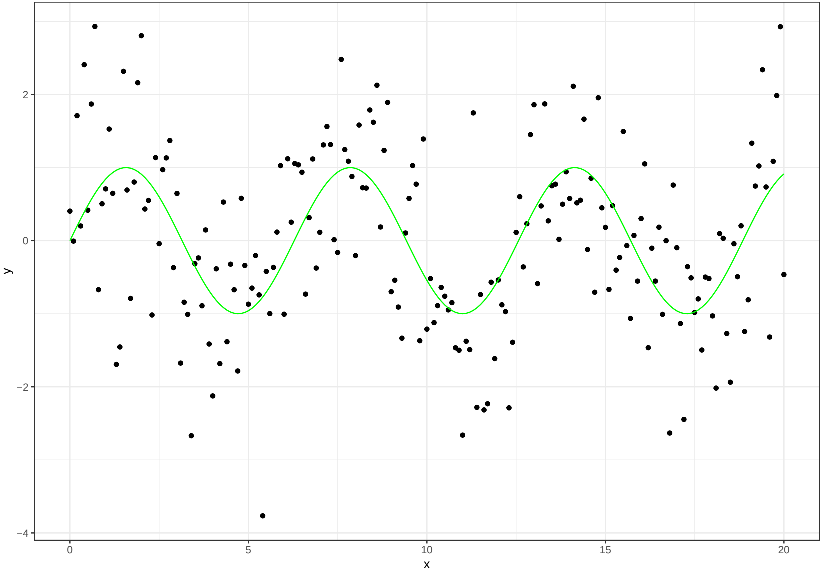
- Signal + Noise
- S3 Object-Oriented Programming

```r
simulate_data <- function(from = 0, to = 20,
    step = 0.1, signal = sin, noise = rnorm) {
    x <- seq(from, to, step)
    data_y <- signal(x)
    y <- data_y + noise(x)
    structure(list(df = data.frame(x = x,
        y = y), signal = data_y), class = "test_data")
}
```

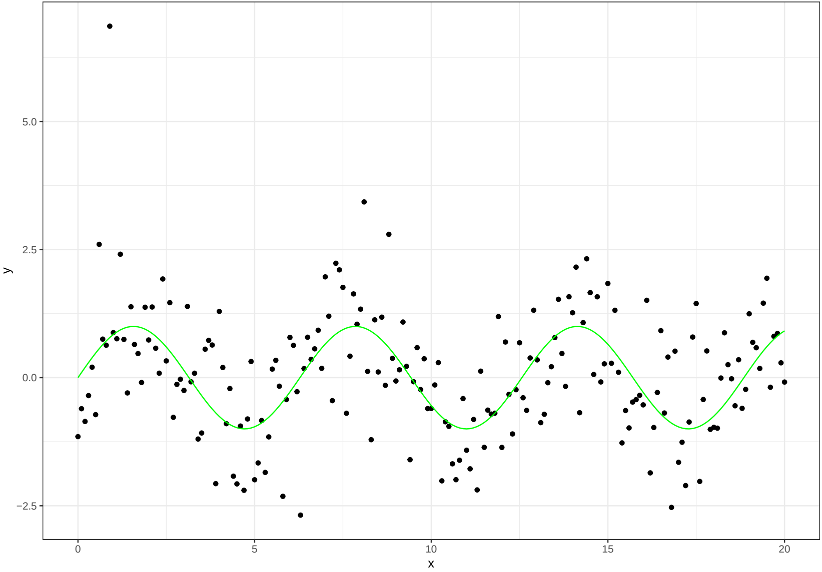
Data simulation

```
$df
  x         y
1 0 1.3709584
2 1 0.2767728
3 2 1.2724258
4 3 0.7739826

$signal
[1] 0.0000000 0.8414710 0.9092974
[4] 0.1411200

attr(,"class")
[1] "test_data"
```

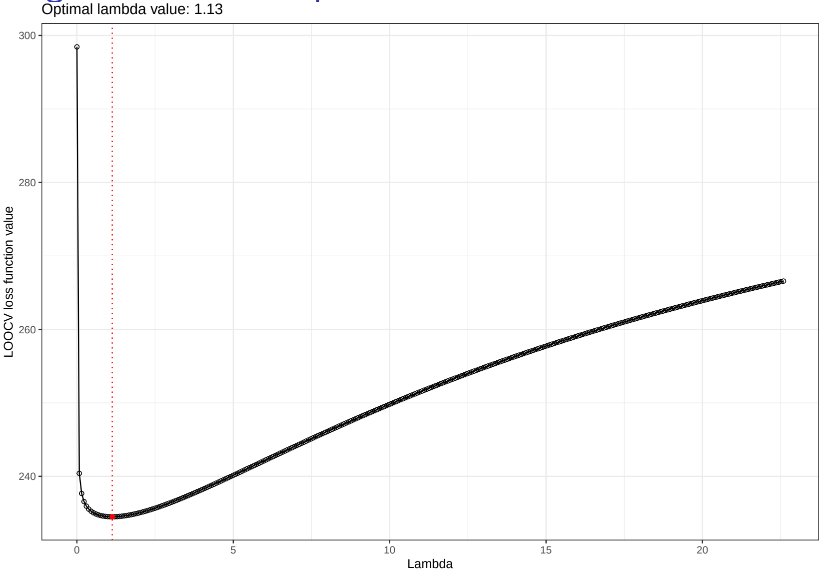
Data simulation: The sine wave (Sim1)



Data simulation: The sine wave with outlier (Sim2)

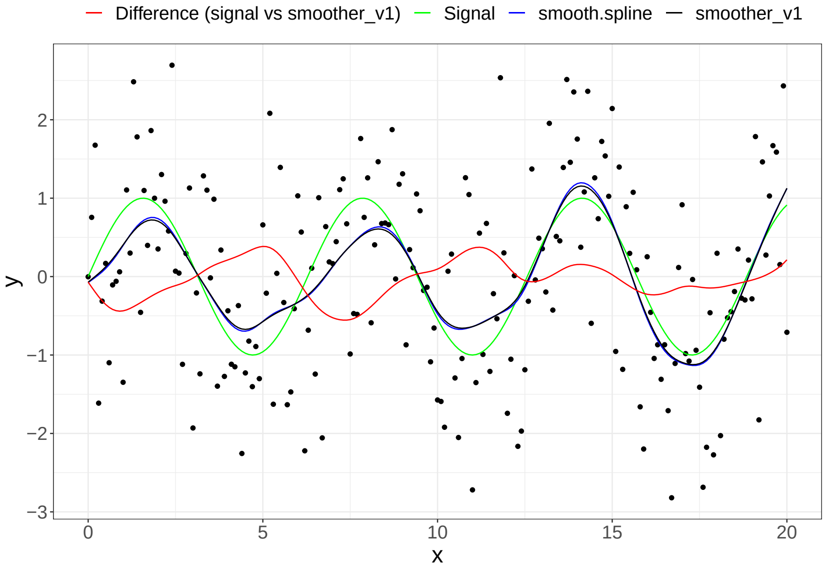


Testing the LOOCV implementation

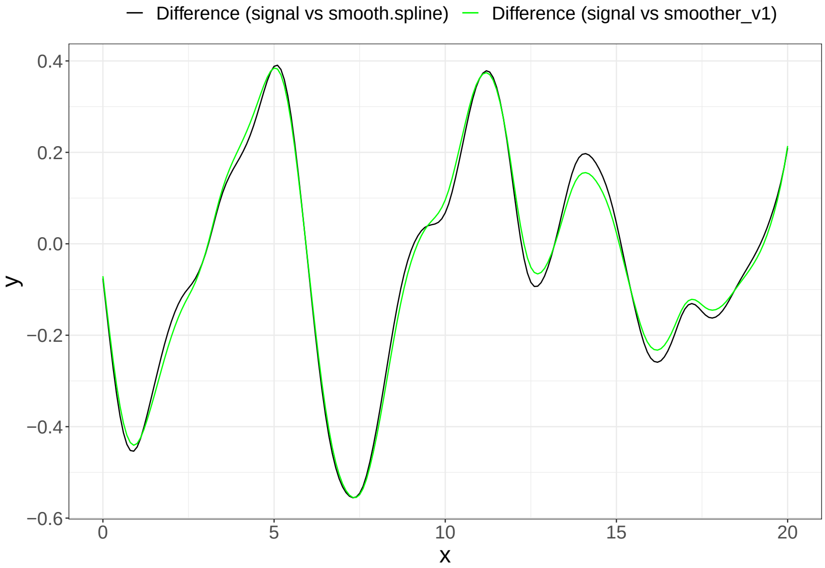


mapping: x = ~x, y = ~y
geom_label: na.rm = FALSE

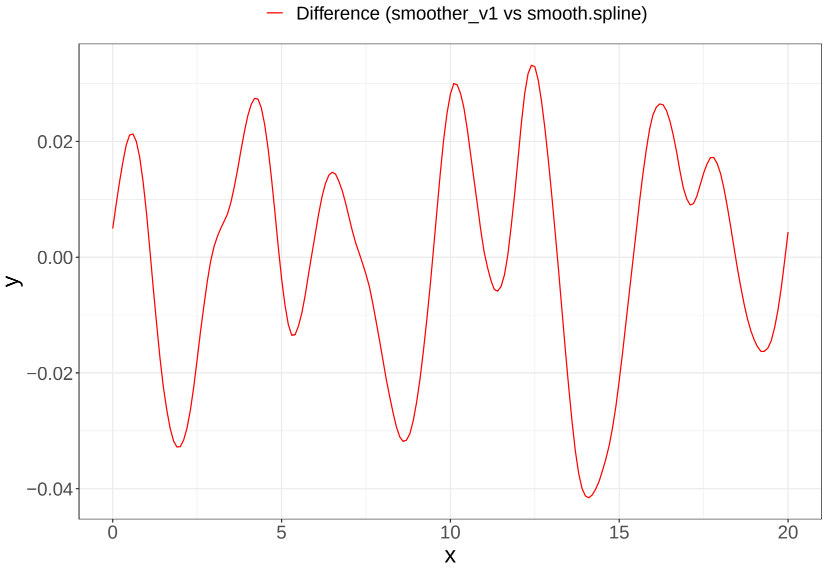
Comparing the smoother with R's smooth.spline



Distance between smoother and the signal



Differences in distances between smoother and the signal



Speed profiling using profvis

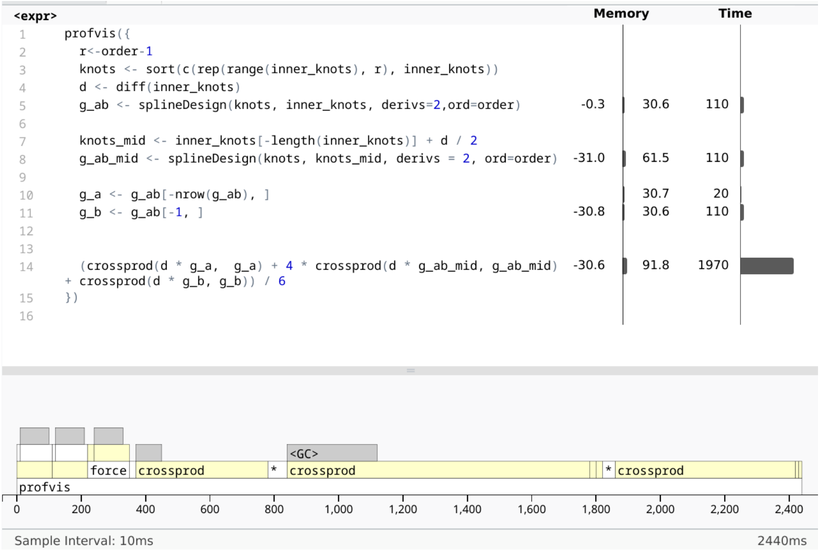
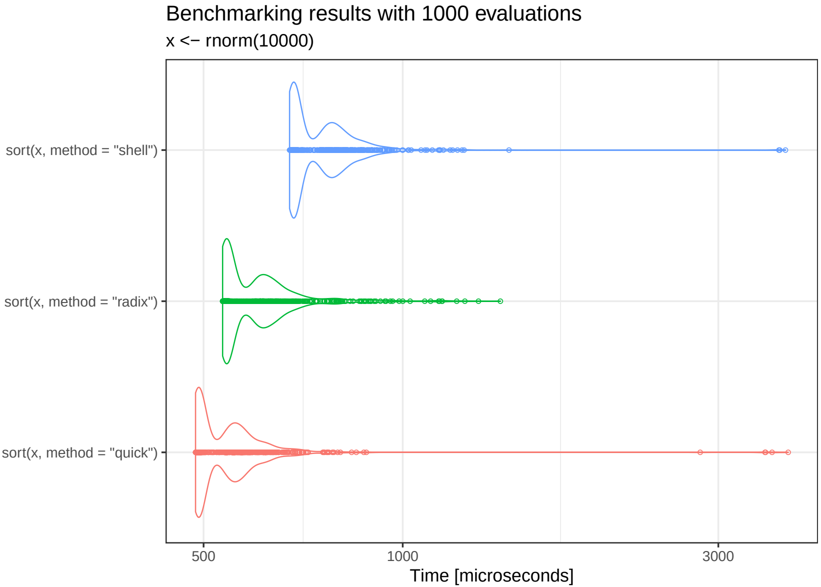


Figure 1: Speed profiling for penalty_matrix_simpson_v1 function

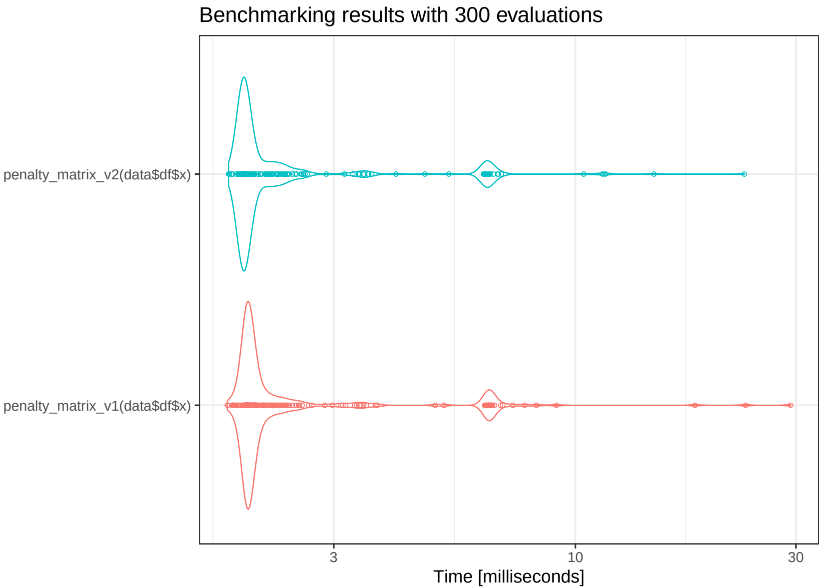
Sorting operation (used in knot definitions)

- quick: Hoare's Quicksort method, default: radix



Speed-up the sorting operation

How impactful is this improvement?



Alternatives to diff function

We used diff function for computing the vector of knot differences (b-a) to construct the penalty matrix.

```
diff_v2 <- function(v) {
    v[2:length(v)] - v[1:(length(v) - 1L)]
}

# Use byte compiling for faster diff()
diff_v3 <- compiler::cmpfun(function(v) {
    v[2:length(v)] - v[1:(length(v) - 1L)]
})

# Save length to a variable instead of
# two calls
diff_v4 <- compiler::cmpfun(function(v) {
    l <- length(v)
    v[2:l] - v[1:(l - 1L)]
})
```

Test accuracy of alternatives

```
all(diff(x10) == diff_v2(x10))
```

```
[1] TRUE
```

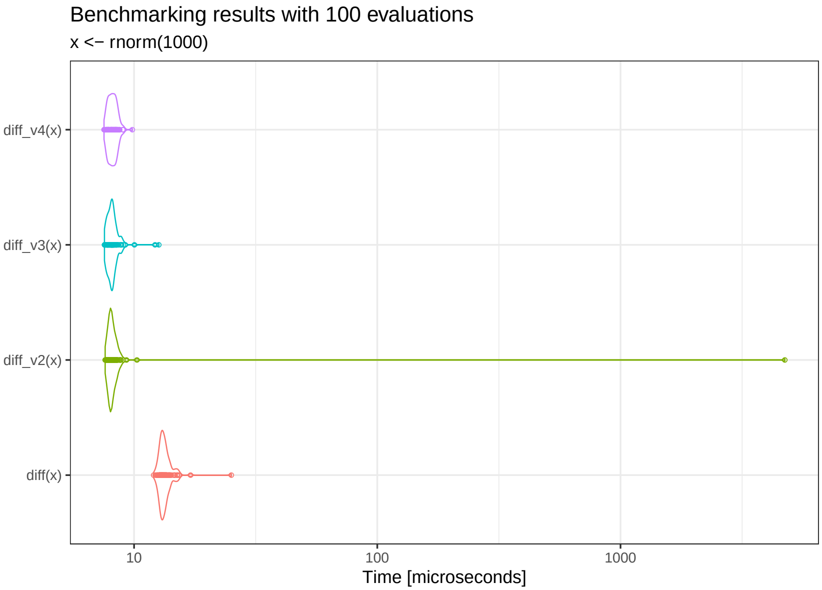
```
all(diff(x10) == diff_v3(x10))
```

```
[1] TRUE
```

```
all(diff(x10) == diff_v4(x10))
```

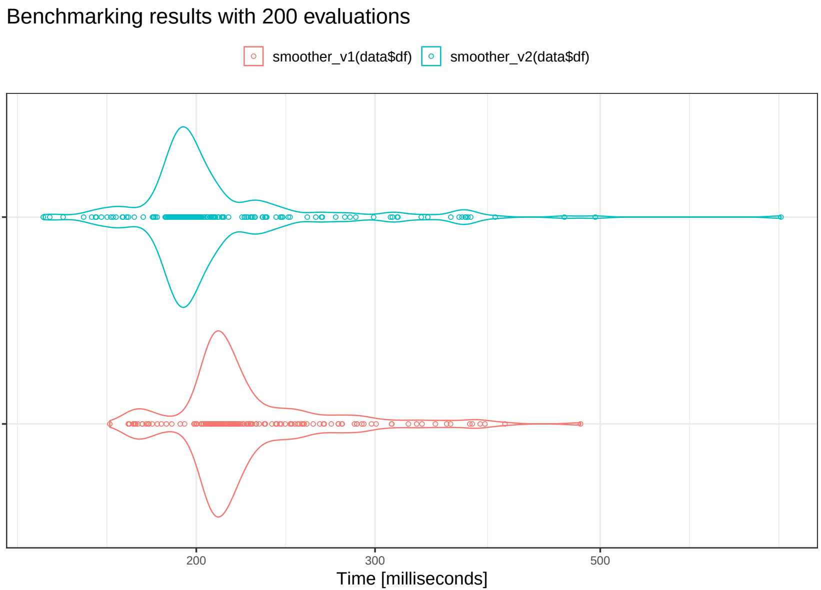
```
[1] TRUE
```

Compare the speed of diff alternatives



Saving and re-using shared matrix operations

Idea: We can precompute the $\Phi_M = \Phi\Phi^T$



Singular value decomposition (SVD)

Singular value decomposition $\Phi = UDV'$.

Diagonalize the matrix $D^{-1}V'\Phi V D^{-1} = W\Gamma W'$.

$$S_\lambda = \widetilde{U}(I + \lambda\Gamma)^{-1}\widetilde{U}'$$

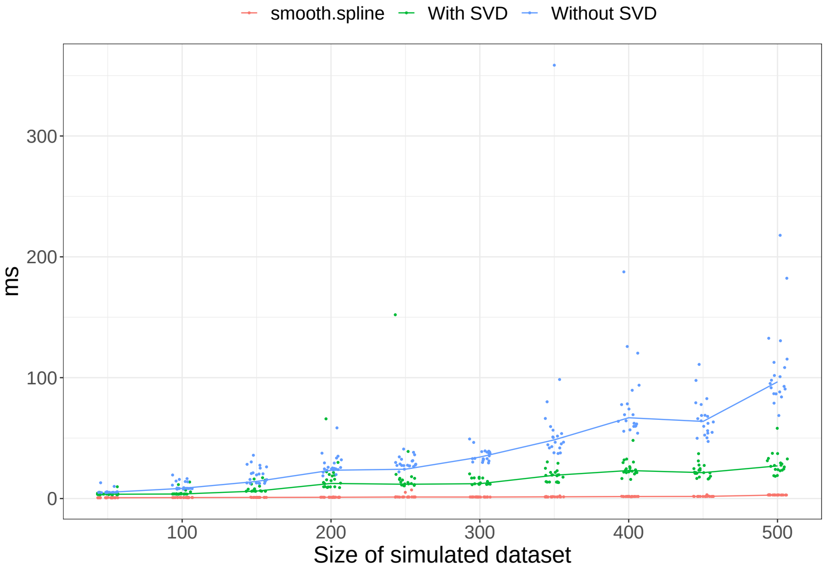
for $\widetilde{U} = UW$.

1. Compute the coefficients $\hat{\beta} = \widetilde{U}'y$ by expanding y in the basis given by the columns of $\widetilde{U}$.
2. The i-th coefficient is shrunk towards 0,

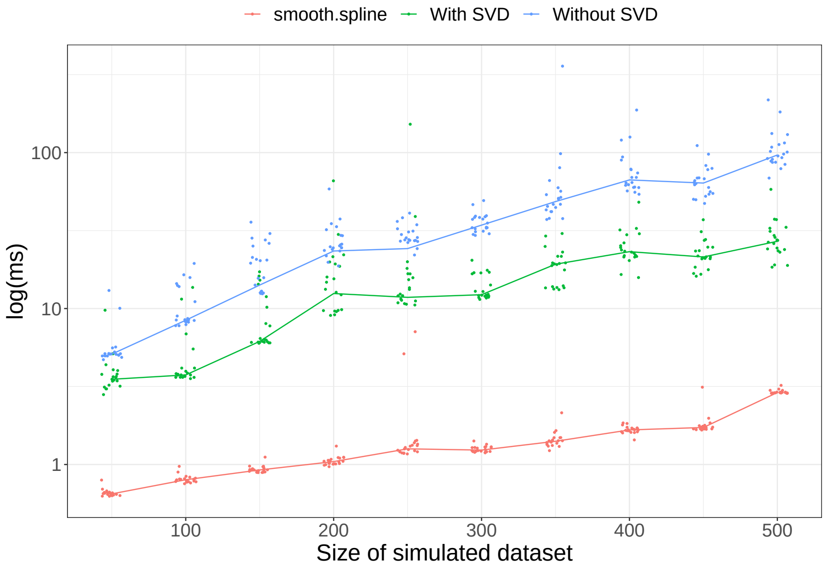
$$\hat{\beta}_i(\lambda) = \frac{\hat{\beta}_i}{1 + \lambda\gamma_i}.$$

3. The smoothed values $\widetilde{U}\hat{\beta}_i(\lambda)$ are computed as an expansion using the shrunken coefficients.

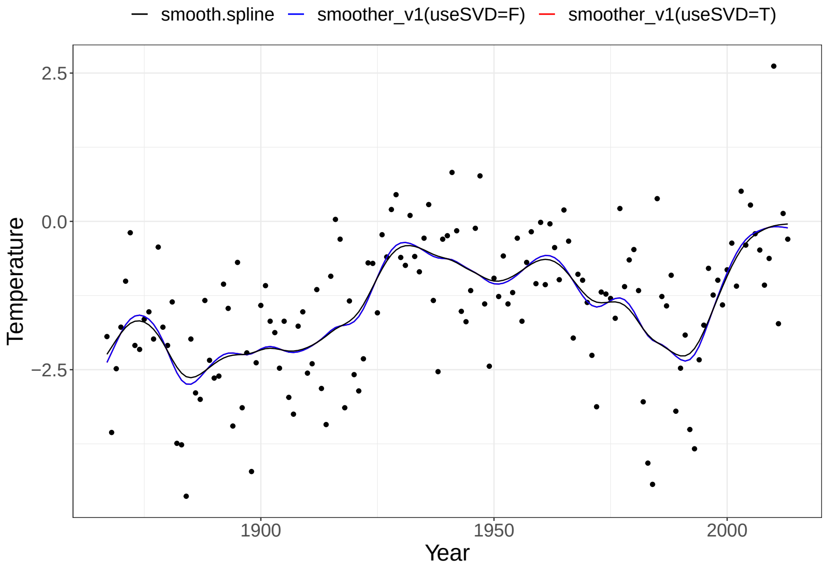
Singular value decomposition (SVD)



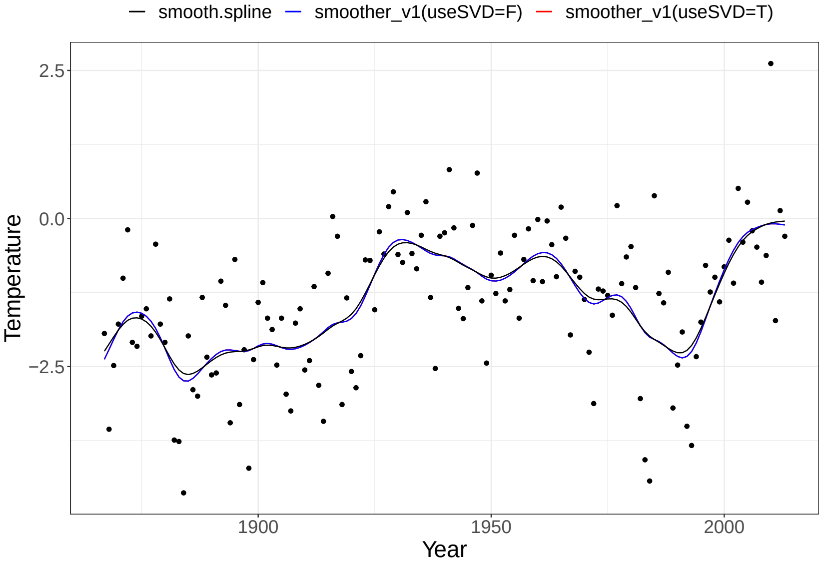
Singular value decomposition (SVD)



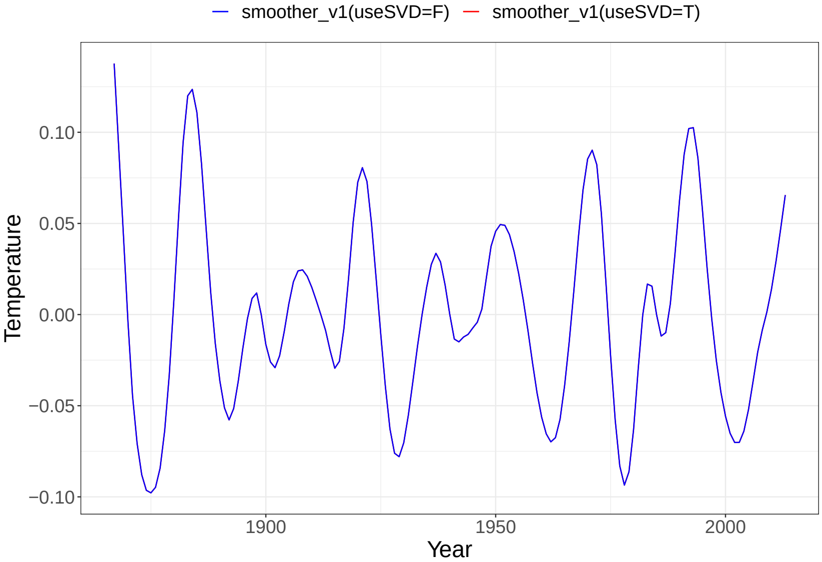
Testing with real data



Testing with real data



Testing with real data



Testing with real data