Time Series Analysis

Week 3 – WLS and RLS with forgetting

Pernille Yde Nielsen

February 21, 2025

Outline of the lecture

- **Recap: Ordinary Least Squares (OLS) and its assumptions**
- Weighted Least Squares (WLS)
- Weighted Least Squares for “Local Trend Models”
- Recursive Least Squares with forgetting
- Exponential smoothing in general

Ordinary Least Squares

For all observations the model equations are written as:

$$\begin{bmatrix} Y_1 \\ \vdots \\ Y_n \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_1^T \\ \vdots \\ \boldsymbol{x}_n^T \end{bmatrix} \boldsymbol{\theta} + \begin{bmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{bmatrix} \quad or \quad \boldsymbol{Y} = \boldsymbol{x}\boldsymbol{\theta} + \boldsymbol{\varepsilon}$$

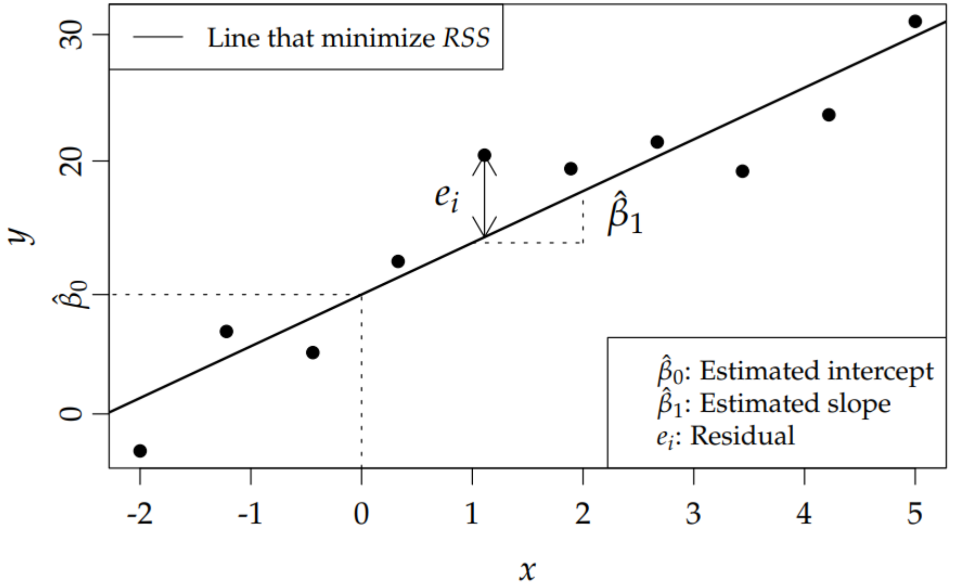
We minimise $S(\boldsymbol{\theta}) = \boldsymbol{\varepsilon}^T\boldsymbol{\varepsilon} = (\boldsymbol{Y} - \boldsymbol{x}\boldsymbol{\theta})^T(\boldsymbol{Y} - \boldsymbol{x}\boldsymbol{\theta})$, by solving $\frac{\partial}{\partial\boldsymbol{\theta}} S(\widehat{\boldsymbol{\theta}}) = 0$.

Ex: Simple linear regression:

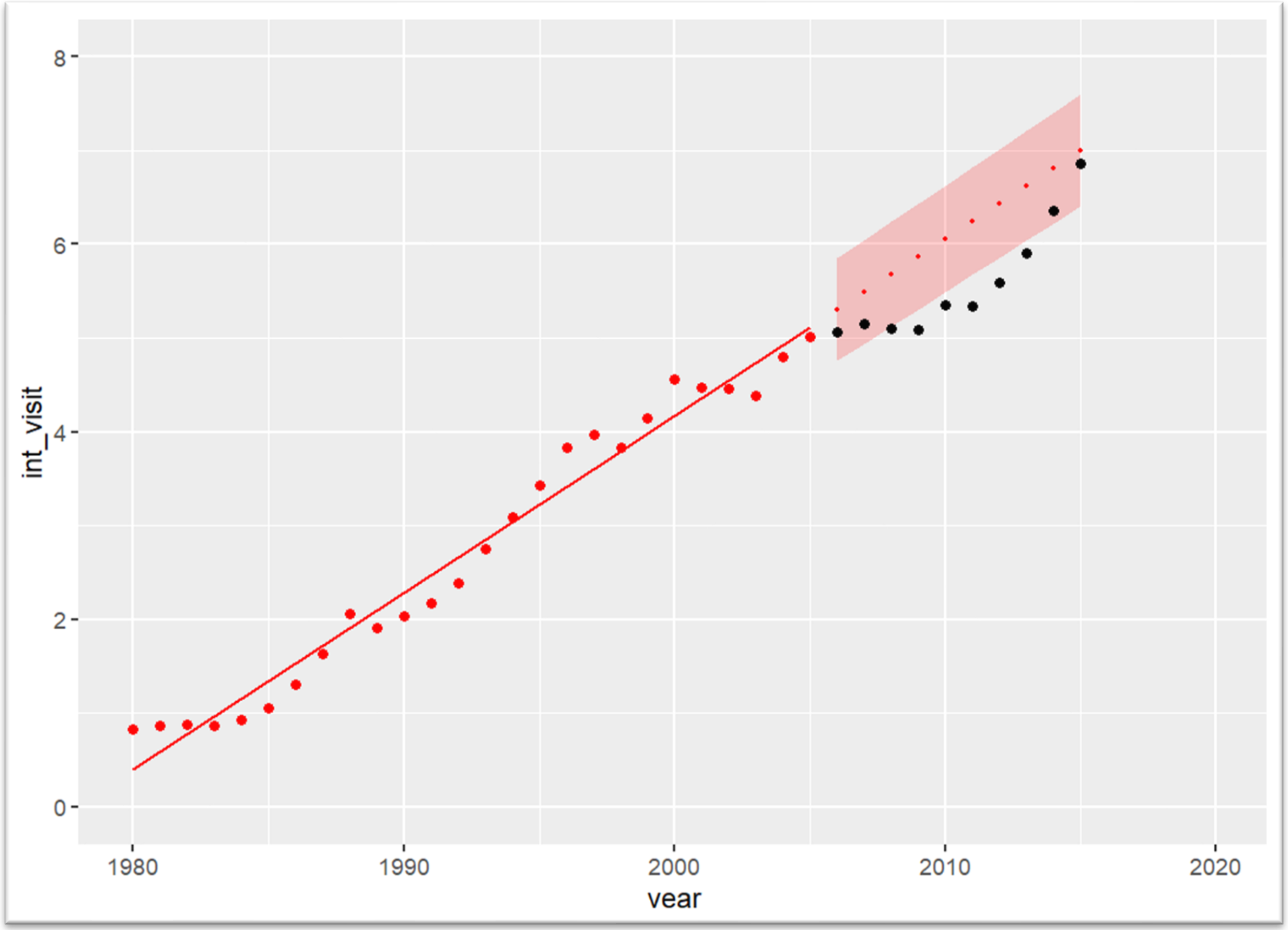
$$Y_i = \beta_0 + \beta_1 x_i + \varepsilon_i, \quad i = \{1, \ldots, n\}$$

$$\begin{bmatrix} Y_1 \\ \vdots \\ Y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{bmatrix}$$

"Design Matrix"



OLS example



(see code in R script)

Data is split into ”train” and ”test” data

We set up a linear model:

$$Y_i = \beta_0 + \beta_1 x_i + \varepsilon_i, \quad i = \{1, \ldots, n\}$$

Fit with OLS:

Intercept: -373 (s.e. 13)

Slope: 0.19 (s.e. 0.006)

We calculate predictions and predictions intervals

We compare predictions to test data

OLS parameter estimates and predictions

Parameters: Point **estimates** and **standard errors**:

$$\widehat{\boldsymbol{\theta}}_t = \arg\min_{\theta} S_t(\boldsymbol{\theta}) = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{Y}$$

```
OLS <- solve(t(X)%*%X)%*%t(X)%*%y
```

$$V[\widehat{\boldsymbol{\theta}}] = \widehat{\sigma}^2(\boldsymbol{X}^T\boldsymbol{X})^{-1}$$

$$\widehat{\sigma}^2 = \boldsymbol{\varepsilon}^T\boldsymbol{\varepsilon}/(N-p)$$

Predictions: Point **estimates** and **standard errors**:

$$\widehat{Y}_t = E_{\widehat{\theta}}[Y_t|\boldsymbol{X}_t = \boldsymbol{x}_t] = \boldsymbol{x}_t^T\widehat{\boldsymbol{\theta}}$$

```
y_pred <- Xtest%*%OLS
```

$$V_{\widehat{\theta}}[Y_t - \widehat{Y}_t] = V_{\widehat{\theta}}[\varepsilon_t + \boldsymbol{x}_t^T(\theta - \widehat{\theta})] = \widehat{\sigma}^2[1 + \boldsymbol{x}_t^T(\boldsymbol{x}^T\boldsymbol{x})^{-1}\boldsymbol{x}_t]$$

```
sigma2_ols*(1+(Xtest%*%solve(t(X)%*%X))%*%t(Xtest))
```

$$\widehat{Y}_t \pm t_{\alpha/2}(n-p)\hat{\sigma}\sqrt{1 + \boldsymbol{x}_t^T(\boldsymbol{x}^T\boldsymbol{x})^{-1}\boldsymbol{x}_t}$$

OLS - model assumptions

Errors must be assumed to all have the same variance and be mutually uncorrelated

Errors are ”**i.i.d.**” (**i**ndependent and **i**dentically **d**istributed)

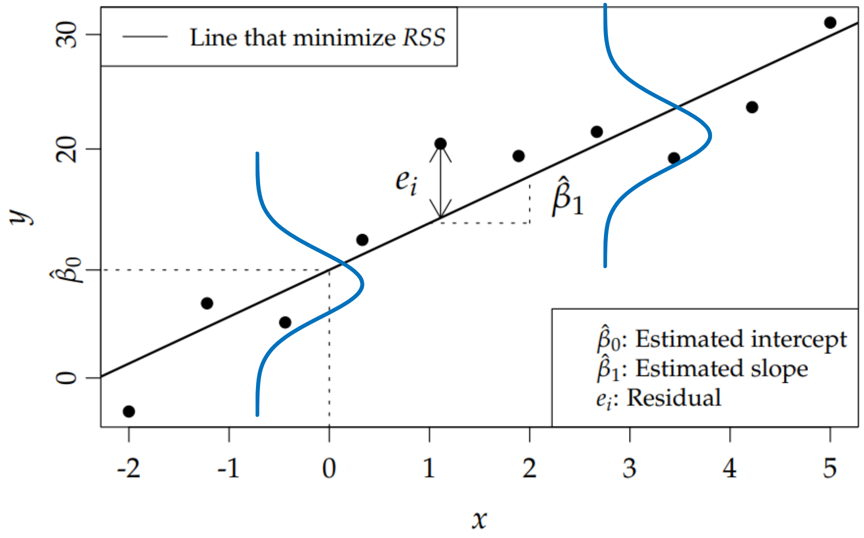
independent:

$$\mathrm{Cov}[\varepsilon_{t_i}, \varepsilon_{t_j}] = \sigma_t^2 \Sigma_{ij}$$

$$\Sigma = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}$$

identically **d**istributed:

$$\varepsilon_t \sim N(0, \sigma^2)$$



Residuals should look like ”white noise” (see plot in R script)

Outline of the lecture

- Recap: Ordinary Least Squares (OLS) and its assumptions
- **Weighted Least Squares (WLS)**
- Weighted Least Squares for “Local Trend Models”
- Recursive Least Squares with forgetting
- Exponential smoothing in general

Weighted Least Squares (WLS)

In WLS we assume the residuals can have different variances and be mutually correlated:

$$V[\boldsymbol{\varepsilon}] = E[\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^T] = \sigma^2\boldsymbol{\Sigma}$$

Weighted Least Squares (WLS)

In WLS we assume the residuals can have different variances and be mutually correlated:

$$V[\boldsymbol{\varepsilon}] = E[\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^T] = \sigma^2\boldsymbol{\Sigma}$$

$$\Sigma = \begin{bmatrix} \rho_{11} & \rho_{12} & \rho_{13} & \cdots & \rho_{1n} \\ \rho_{21} & \rho_{22} & \rho_{21} & \cdots & \rho_{2n} \\ \rho_{31} & \rho_{32} & \rho_{33} & \cdots & \rho_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho_{n1} & \rho_{n2} & \rho_{n3} & \cdots & \rho_{nn} \end{bmatrix}$$

Remember that the diagonal elements have to do with the variances of the individual observations

And the off-diagonal elements have to do with covariance between two observations

Weighted Least Squares (WLS)

In WLS we assume the residuals can have different variances and be mutually correlated:

$$V[\boldsymbol{\varepsilon}] = E[\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^T] = \sigma^2\boldsymbol{\Sigma}$$

We minimize the *weighted* sum of squared residuals:

$$(\boldsymbol{Y} - \boldsymbol{x}\boldsymbol{\theta})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{Y} - \boldsymbol{x}\boldsymbol{\theta})$$

$$\Sigma = \begin{bmatrix} \rho_{11} & \rho_{12} & \rho_{13} & \cdots & \rho_{1n} \\ \rho_{21} & \rho_{22} & \rho_{21} & \cdots & \rho_{2n} \\ \rho_{31} & \rho_{32} & \rho_{33} & \cdots & \rho_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho_{n1} & \rho_{n2} & \rho_{n3} & \cdots & \rho_{nn} \end{bmatrix}$$

Remember that the diagonal elements have to do with the variances of the individual observations

And the off-diagonal elements have to do with covariance between two observations

Weighted Least Squares (WLS)

In WLS we assume the residuals can have different variances and be mutually correlated:

$$V[\boldsymbol{\varepsilon}] = E[\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^T] = \sigma^2\boldsymbol{\Sigma}$$

We minimize the *weighted* sum of squared residuals:

$$(\boldsymbol{Y} - \boldsymbol{x}\boldsymbol{\theta})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{Y} - \boldsymbol{x}\boldsymbol{\theta})$$

Solution: $$\boxed{\widehat{\boldsymbol{\theta}} = (\boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{x})^{-1}\boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{Y}}$$

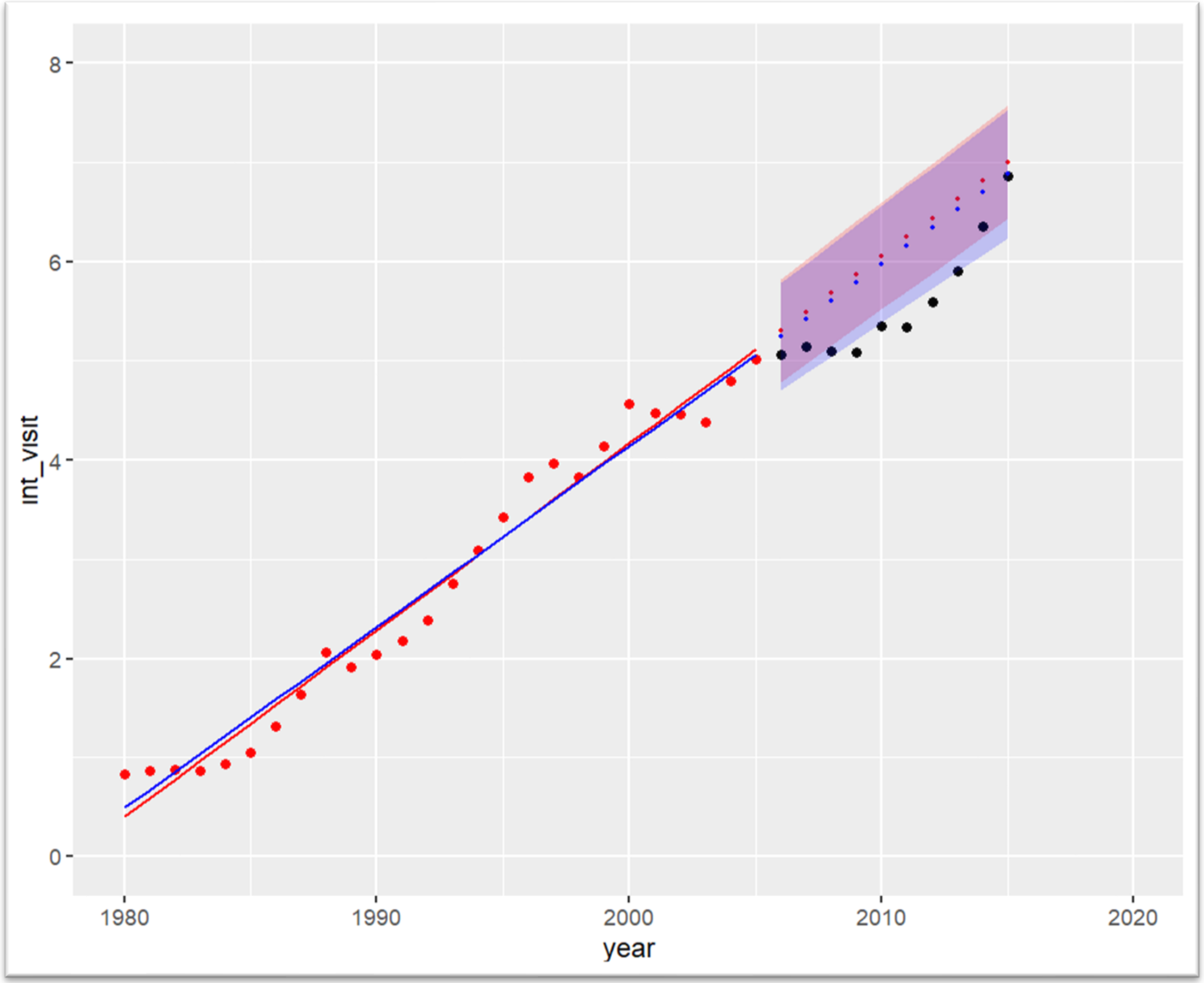
(Equations on blackboard)

(Example in R – next slide)

$$\Sigma = \begin{bmatrix} \rho_{11} & \rho_{12} & \rho_{13} & \cdots & \rho_{1n} \\ \rho_{21} & \rho_{22} & \rho_{21} & \cdots & \rho_{2n} \\ \rho_{31} & \rho_{32} & \rho_{33} & \cdots & \rho_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho_{n1} & \rho_{n2} & \rho_{n3} & \cdots & \rho_{nn} \end{bmatrix}$$

Remember that the diagonal elements have to do with the variances of the individual observations

And the off-diagonal elements have to do with covariance between two observations



WLS example

(see code in R script)

Include correlation-structure:

$$\Sigma = \begin{bmatrix} 1 & \rho^1 & \rho^2 & \cdots & \rho^N \\ \rho^1 & 1 & \rho^1 & \cdots & \rho^{N-1} \\ \rho^2 & \rho^1 & 1 & \cdots & \rho^{N-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho^N & \rho^{N-1} & \rho^{N-2} & \cdots & 1 \end{bmatrix}$$

(rho < 1)

Fit with WLS:

Intercept: -361 (s.e. 20)

Slope: 0.18 (s.e. 0.010)

Parameters and predictions a slightly different to the OLS fit.

WLS with diagonal matrix

In the example above we changed the off-diagonal elements

Now we consider a case were only the diagonal elements are changed.

What does this mean / what situation would this reflect?

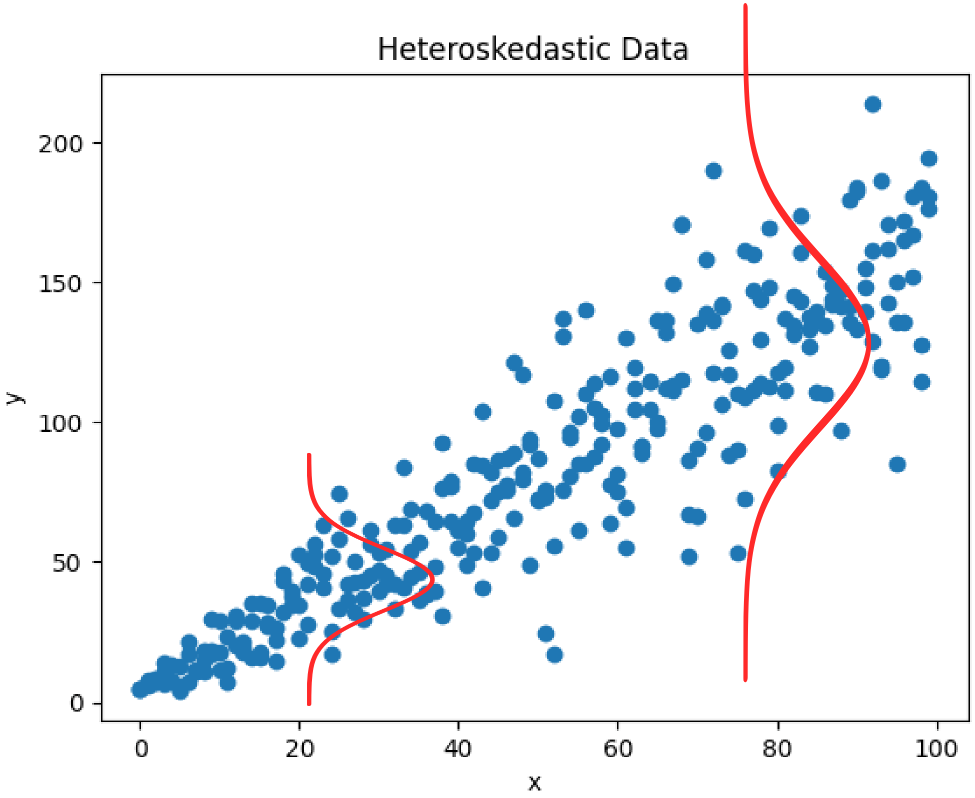
$$\Sigma = \begin{bmatrix} \rho_{11} & 0 & 0 & \dots & 0 \\ 0 & \rho_{22} & 0 & \dots & 0 \\ 0 & 0 & \rho_{33} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \rho_{nn} \end{bmatrix}$$

WLS with diagonal matrix

Off-diagonal elements are zero = no covariance/no correlation

$$\Sigma = \begin{bmatrix} \rho_{11} & 0 & 0 & \dots & 0 \\ 0 & \rho_{22} & 0 & \dots & 0 \\ 0 & 0 & \rho_{33} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \rho_{nn} \end{bmatrix}$$

$$V[\boldsymbol{\varepsilon}] = E[\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^T] = \sigma^2 \boldsymbol{\Sigma}$$



How would Σ look like for this type of data?

Outline of the lecture

- Recap: Ordinary Least Squares (OLS) and its assumptions
- Weighted Least Squares (WLS)
- **Weighted Least Squares for “Local Trend Models”**
- Recursive Least Squares with forgetting
- Exponential smoothing in general

From Global to Local models

Considering our model where x-values are time (”trend models”)

We want to make predictions for future time (”**forecast”**)

How could we improve the model to make a better **forecast**?

From Global to Local models

What if the model had only been based on the 10 most recent observations?

From Global to Local models

What if the model had only been based on the 10 most recent observations?

- Parameters (intercept and slope) are different and hence predicted values are different

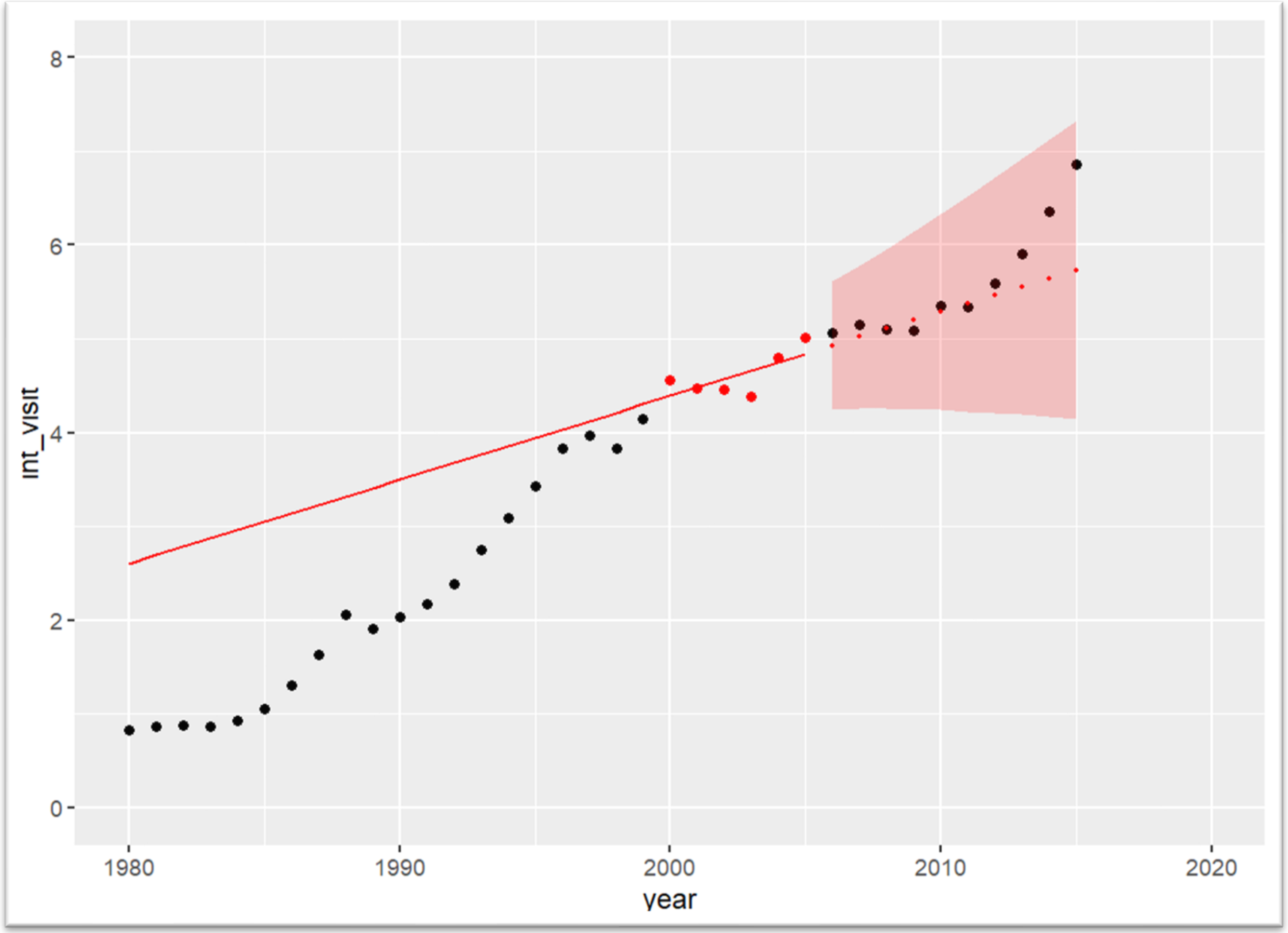
- Prediction intervals are wider

(s.e. on parameter estimates are also larger).

This is because n (number of observations) is smaller.

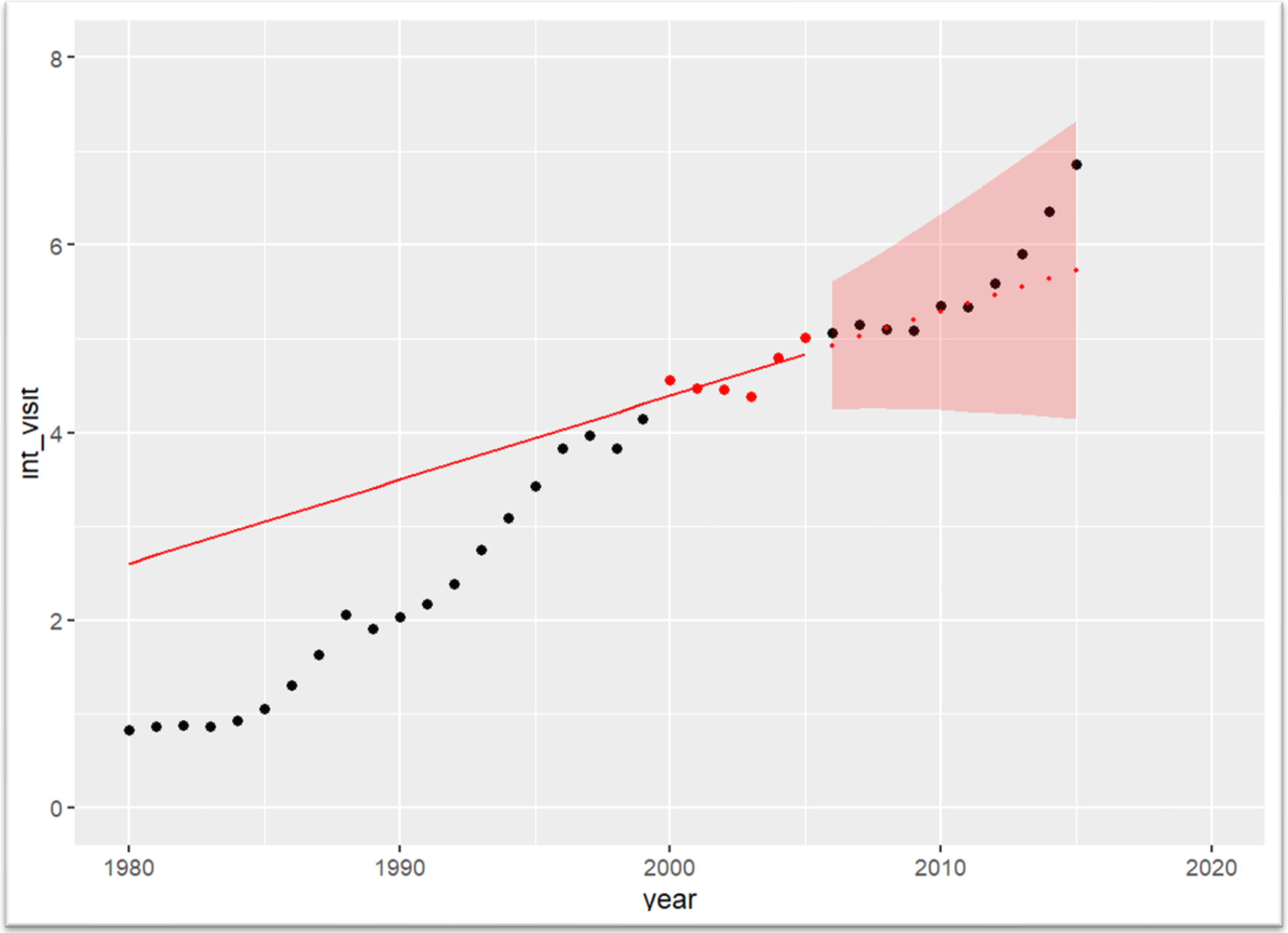
Recall: $\hat{\sigma}^2 = \boldsymbol{\varepsilon}^T \boldsymbol{\varepsilon}/(n-p)$

From Global to Local models



What if the model had only been based on the 5 most recent observations?

From Global to Local models

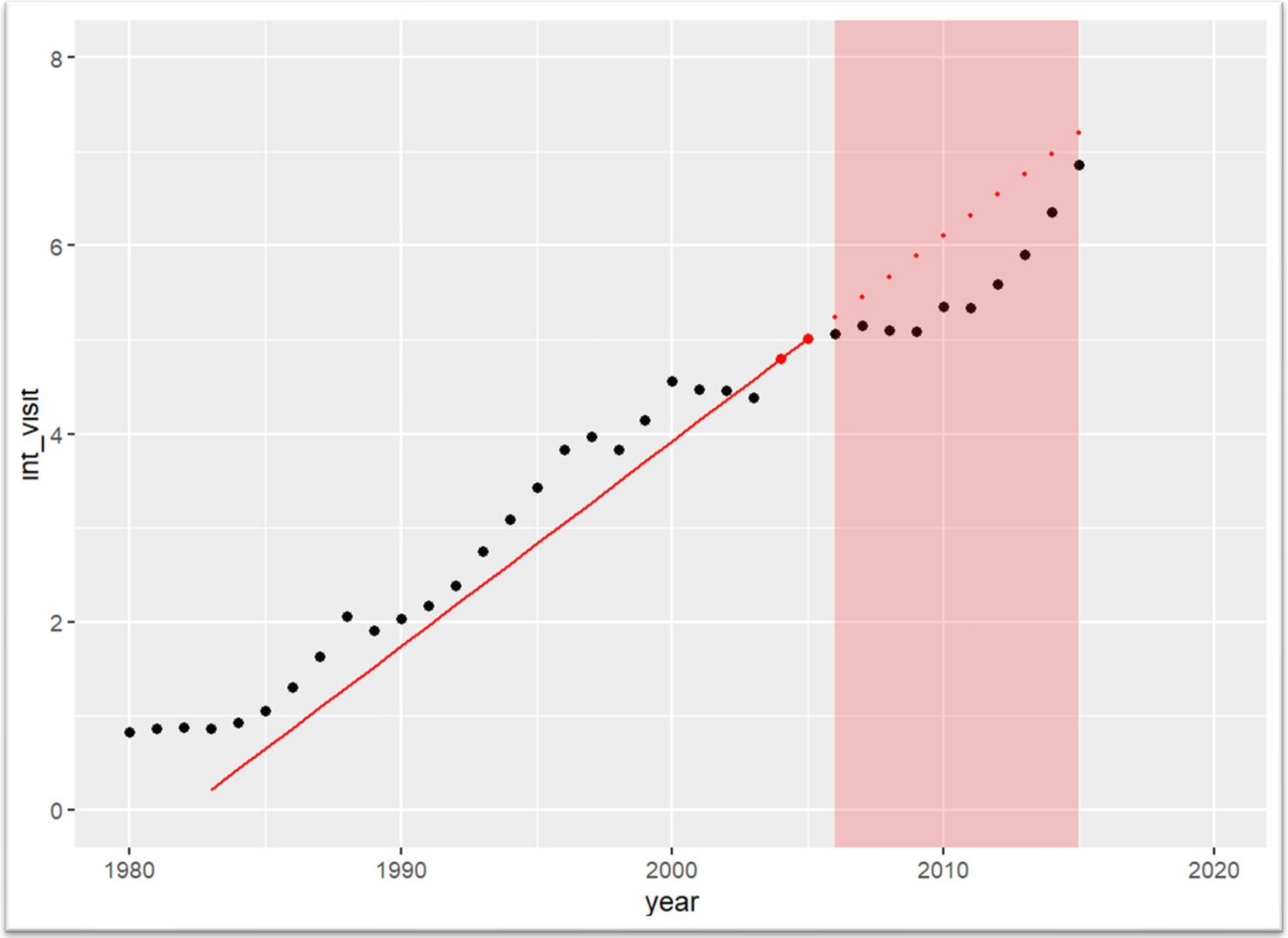


What if the model had only been based on the 5 most recent observations?

- How many observations do we *need*?

- What is optimal?

From Global to Local models



What if the model had only been based on the 2 most recent observations?

He we use only two oberservations to estimate two parameters

The variance estimate ”explodes”

$$\widehat{\sigma}^2 = \boldsymbol{\varepsilon}^T \boldsymbol{\varepsilon} / (n - p)$$

N = number of observations in data

p = number of parameters estimated

From Global to Local models

Questions:

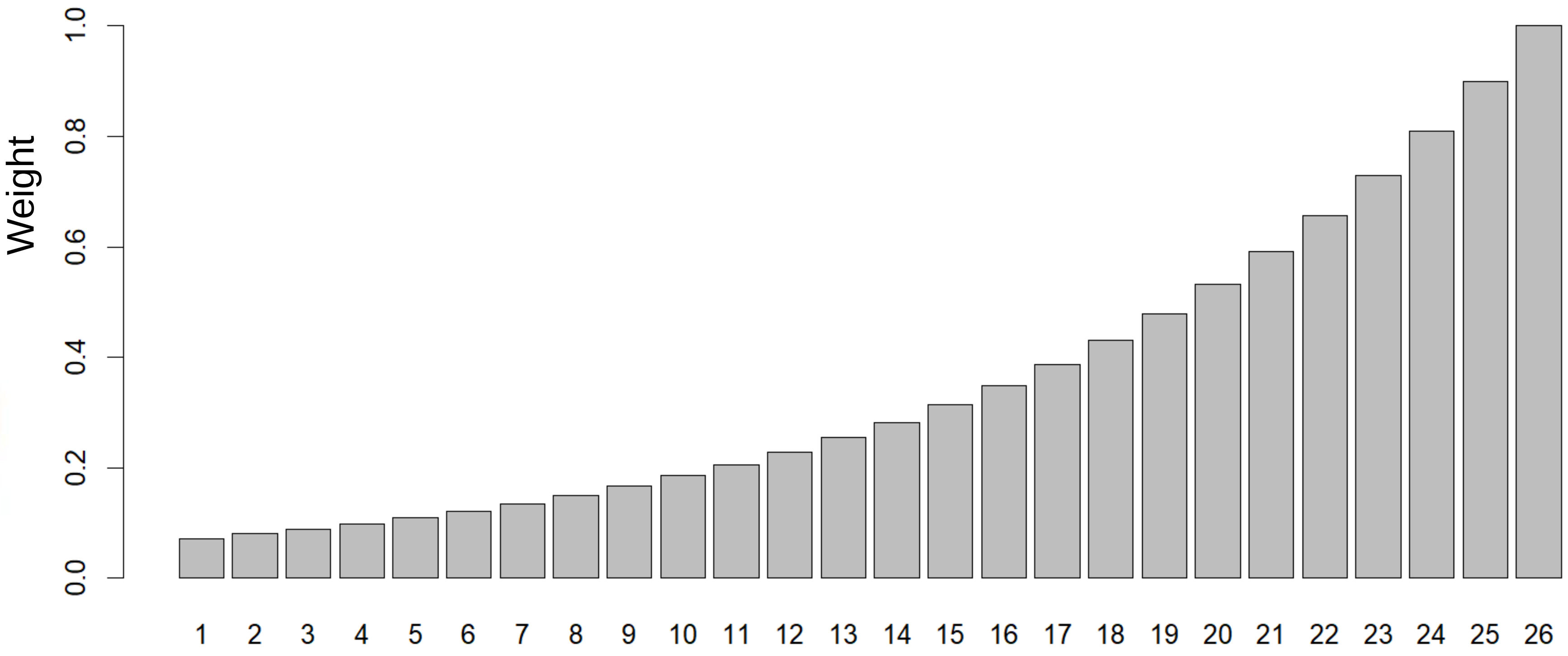
Any good ideas?

How do we choose the best model?

Is there a way to make a ”soft” cut-off of the number of observations included in the training data?

From Global to Local models

We could also use weights and make most recent obs. have highest weight!



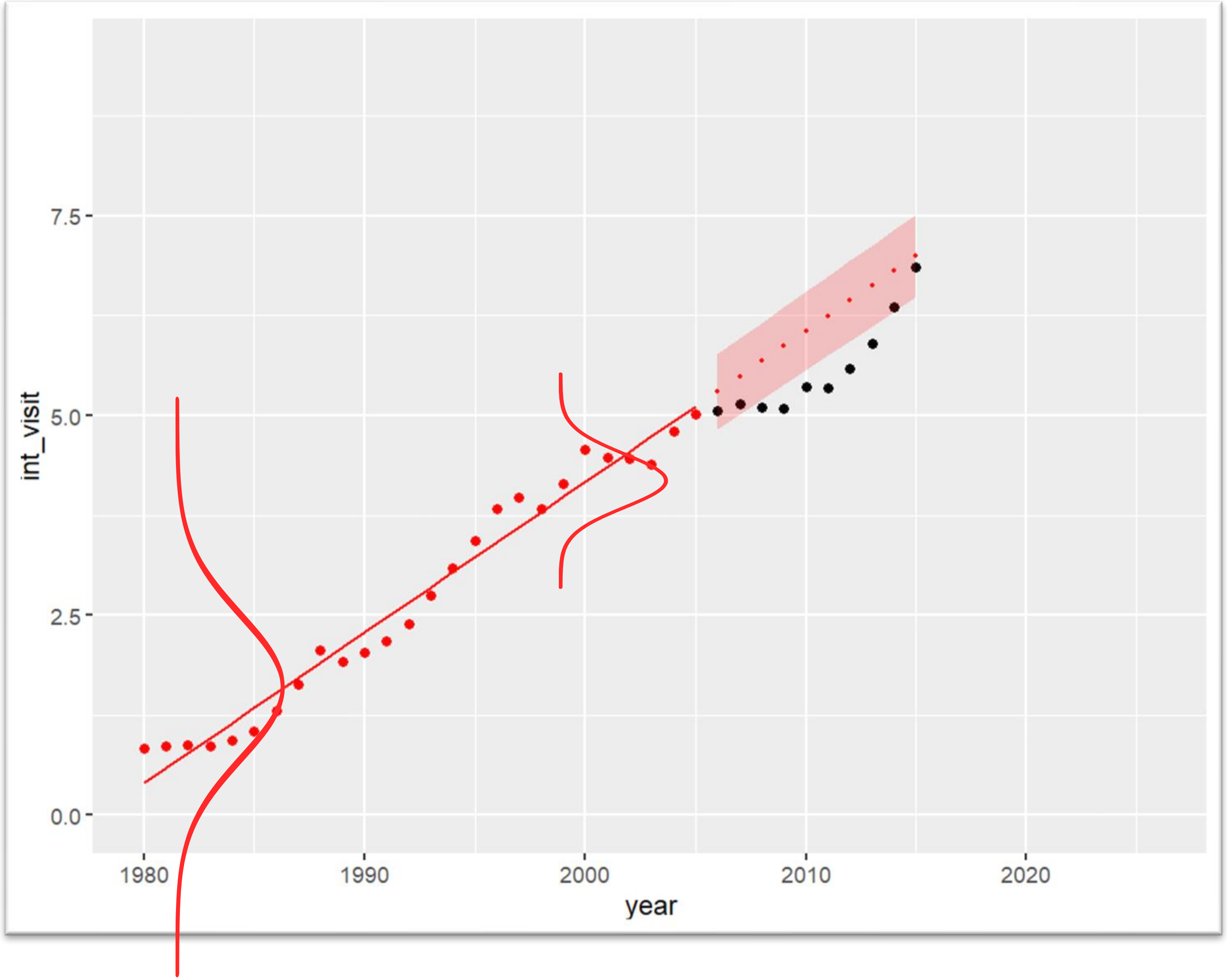
Weights in WLS

$(\boldsymbol{Y} - \boldsymbol{x\theta})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{Y} - \boldsymbol{x\theta})$ Weighted sum of squares, uses Σ^{-1}

Observations with **large variance** will have **low weight**

$$\Sigma = \begin{bmatrix} \rho_{11} & 0 & 0 & \dots & 0 \\ 0 & \rho_{22} & 0 & \dots & 0 \\ 0 & 0 & \rho_{33} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \rho_{nn} \end{bmatrix} = \begin{bmatrix} 1/w_1 & 0 & 0 & \dots & 0 \\ 0 & 1/w_2 & 0 & \dots & 0 \\ 0 & 0 & 1/w_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1/w_n \end{bmatrix}$$

WLS “hack” for down-weighting old observations



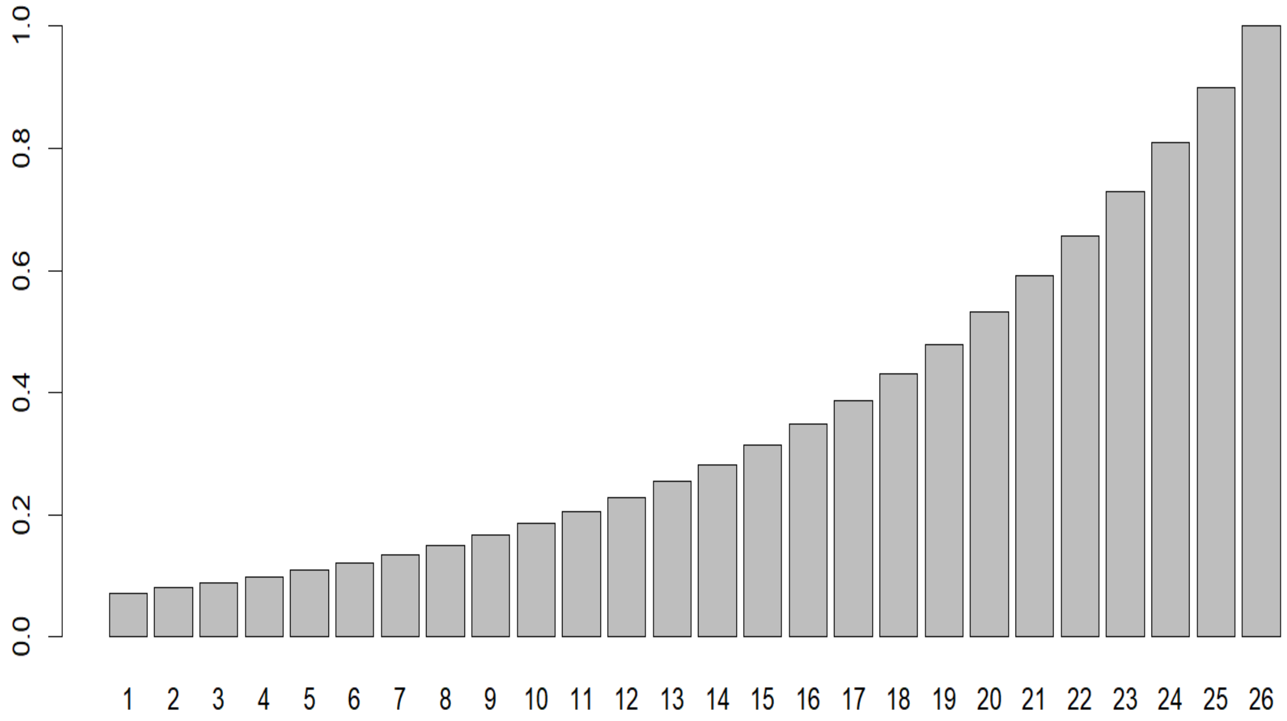
We will assign low weight (large variance) to the old observations and higher weight (small variance) to the more recent observations

This is a “hack”: The old observations were not actually measured with larger uncertainty (larger variance).

However, in the model we want, the old observations should have less weight.

Choice of weights for a "local model"

Exponential weights:



Corresponding WLS Σ-matrix:

$$\Sigma = \begin{bmatrix} 1/\lambda^{n-1} & \dots & 0 & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \dots & 1/\lambda^2 & 0 & 0 \\ 0 & \dots & 0 & 1/\lambda & 0 \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}$$

$$0 < \lambda < 1$$

Example in R

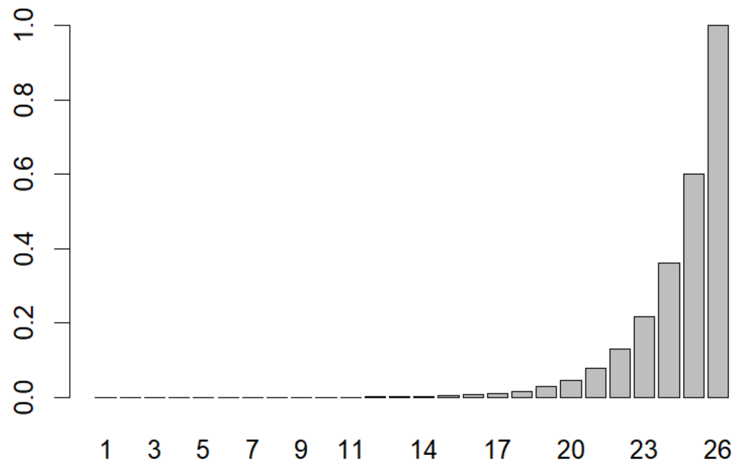
```r
n <- 26
lambda = 0.6
```

```r
SIGMA <- diag(n)
for (i in 1:n) {
  SIGMA[i,i] <- 1/lambda^(n-i)
}
```

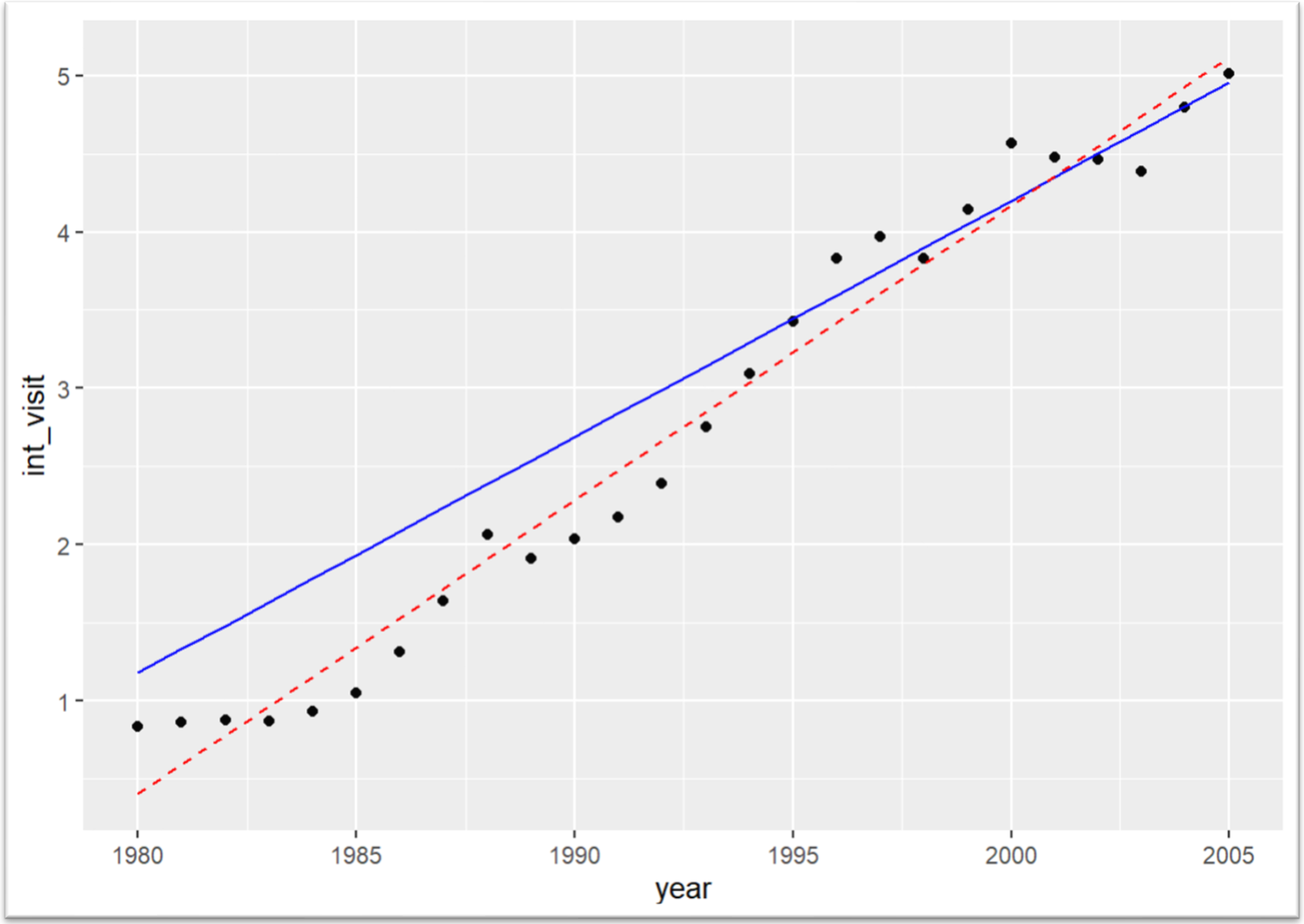
```r
print(SIGMA[20:26,20:26])
```

```
21.43347  0.00000 0.000000 0.00000 0.000000 0.000000    0
 0.00000 12.86008 0.000000 0.00000 0.000000 0.000000    0
 0.00000  0.00000 7.716049 0.00000 0.000000 0.000000    0
 0.00000  0.00000 0.000000 4.62963 0.000000 0.000000    0
 0.00000  0.00000 0.000000 0.00000 2.777778 0.000000    0
 0.00000  0.00000 0.000000 0.00000 0.000000 1.666667    0
 0.00000  0.00000 0.000000 0.00000 0.000000 0.000000    1
```

Plot of weights with lambda = 0.6:



Example in R

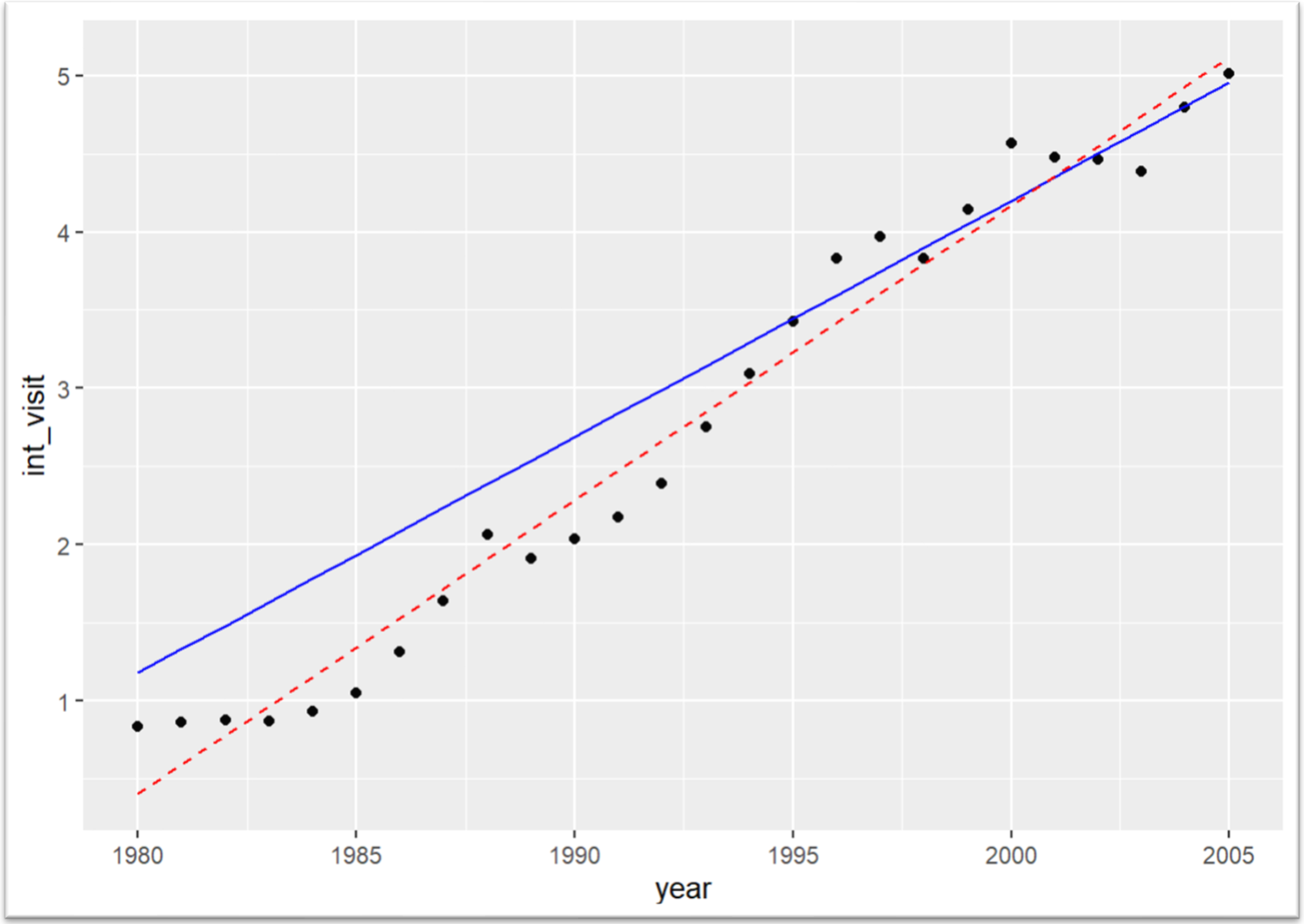


Blue is WLS

(Red is OLS)

The blue line fits the late observations better than the early observations – just as we wanted

Example in R



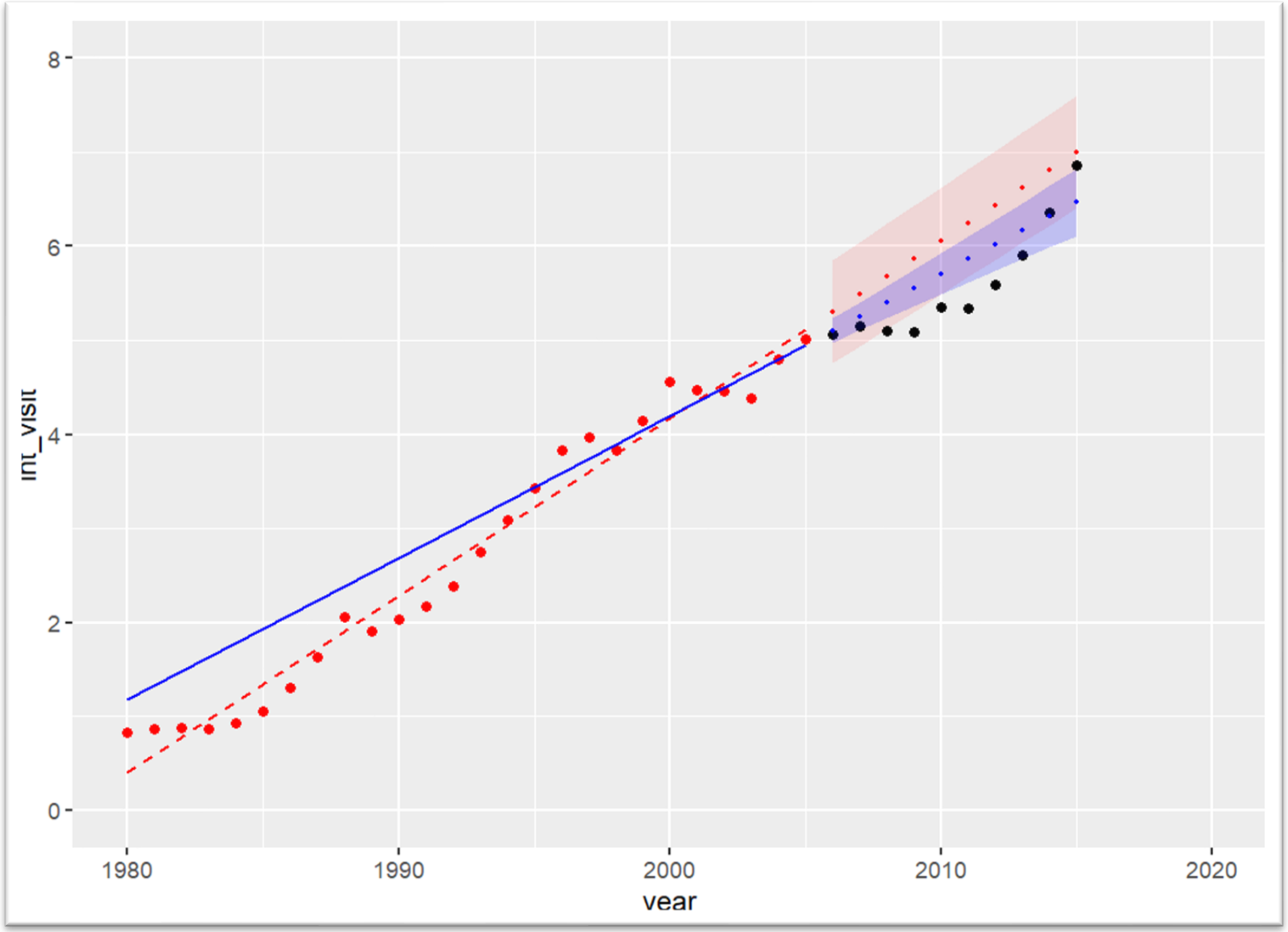
Blue is WLS

(Red is OLS)

The blue line fits the late observations better than the early observations – just as we wanted

What about predictions and prediction intervals?!

Example in R



Prediction intervals using the equations from WLS

The intervals seem over optimistic (too narrow)

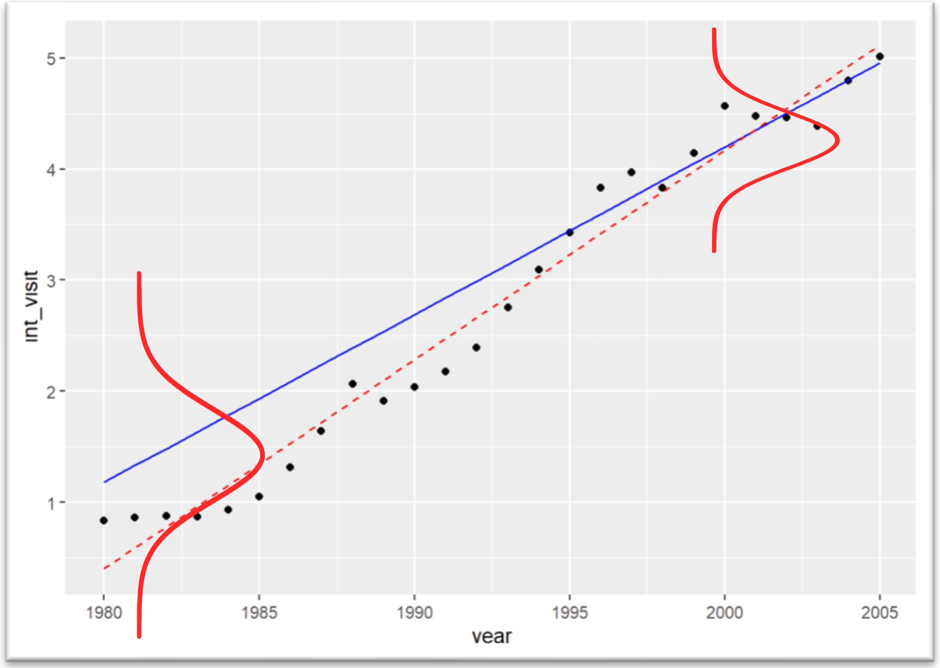
Recal:

$$\widehat{\sigma}^2 = \frac{1}{N-p}(\boldsymbol{Y} - \boldsymbol{x}\widehat{\boldsymbol{\theta}})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{Y} - \boldsymbol{x}\widehat{\boldsymbol{\theta}})$$

Values in Σ^{-1} are small – leading to underestimation of uncertainties?

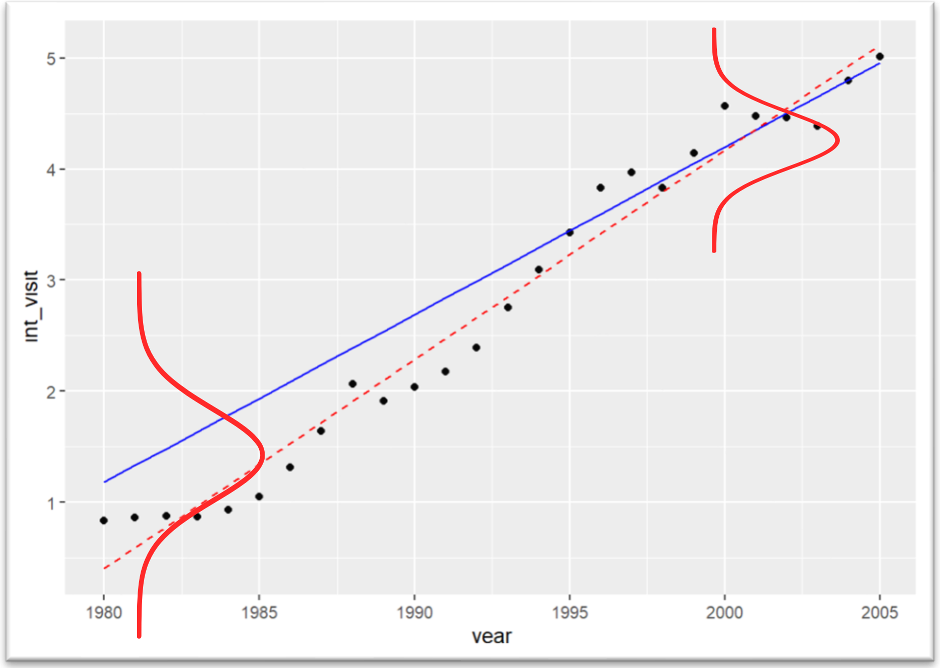
Predictions with Local model

What should we assume about the variance of future observations?



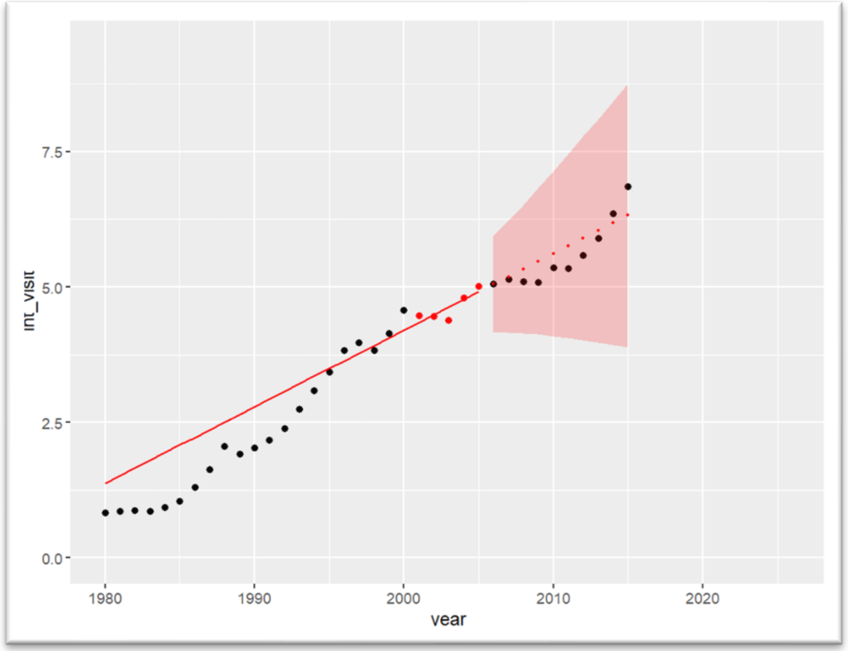
Predictions with Local model

What should we assume about the variance of future observations?



$$\widehat{\sigma}^2 = \frac{1}{N-p}(\boldsymbol{Y} - \boldsymbol{x}\widehat{\boldsymbol{\theta}})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{Y} - \boldsymbol{x}\widehat{\boldsymbol{\theta}})$$

We have seen earlier today that using less data points leads to larger uncertainties



Is it fair to use n = N ?

(when some weights are very close to zero)

Estimating uncertainty in Local model

Define the total memory as the sum of all the weights:

$$T = \sum_{j=0}^{N-1} \lambda^j$$

T is a meassure of the weighted number of observations

(In OLS the sum of weights = N)

Estimating uncertainty in Local model

Define the total memory as the sum of all the weights:

$$T = \sum_{j=0}^{N-1} \lambda^j$$

T is a meassure of the weighted number of observations

(In OLS the sum of weights = N)

Replace N with T, when estimating the variance:

$$\hat{\sigma}^2 = (\boldsymbol{Y} - \boldsymbol{x}_N \widehat{\boldsymbol{\theta}}_N)^T \Sigma^{-1} (\boldsymbol{Y} - \boldsymbol{x}_N \widehat{\boldsymbol{\theta}}_N)/(T - p)$$

Notice we need T > p

(p = number of parameters)

The sum of weights must be larger than the number of parameters estimated.

Estimating uncertainty in Local model

Define the total memory as the sum of all the weights:

$$T = \sum_{j=0}^{N-1} \lambda^j$$

T is a meassure of the weighted number of observations

(In OLS the sum of weights = N)

Replace N with T, when estimating the variance:

$$\hat{\sigma}^2 = (\boldsymbol{Y} - \boldsymbol{x}_N \widehat{\boldsymbol{\theta}}_N)^T \Sigma^{-1} (\boldsymbol{Y} - \boldsymbol{x}_N \widehat{\boldsymbol{\theta}}_N)/(T - p)$$

Notice we need T > p

(p = number of parameters)

The sum of weights must be larger than the number of parameters estimated.

This requirement is a restriction on lambda

(lambda cannot be too small)

Estimating uncertainty in Local model

Define the total memory as the sum of all the weights:

$$T = \sum_{j=0}^{N-1} \lambda^j$$

T is a meassure of the weighted number of observations

(In OLS the sum of weights = N)

Replace N with T, when estimating the variance:

$$\hat{\sigma}^2 = (\boldsymbol{Y} - \boldsymbol{x}_N \widehat{\boldsymbol{\theta}}_N)^T \Sigma^{-1} (\boldsymbol{Y} - \boldsymbol{x}_N \widehat{\boldsymbol{\theta}}_N)/(T - p)$$

Notice we need T > p

(p = number of parameters)

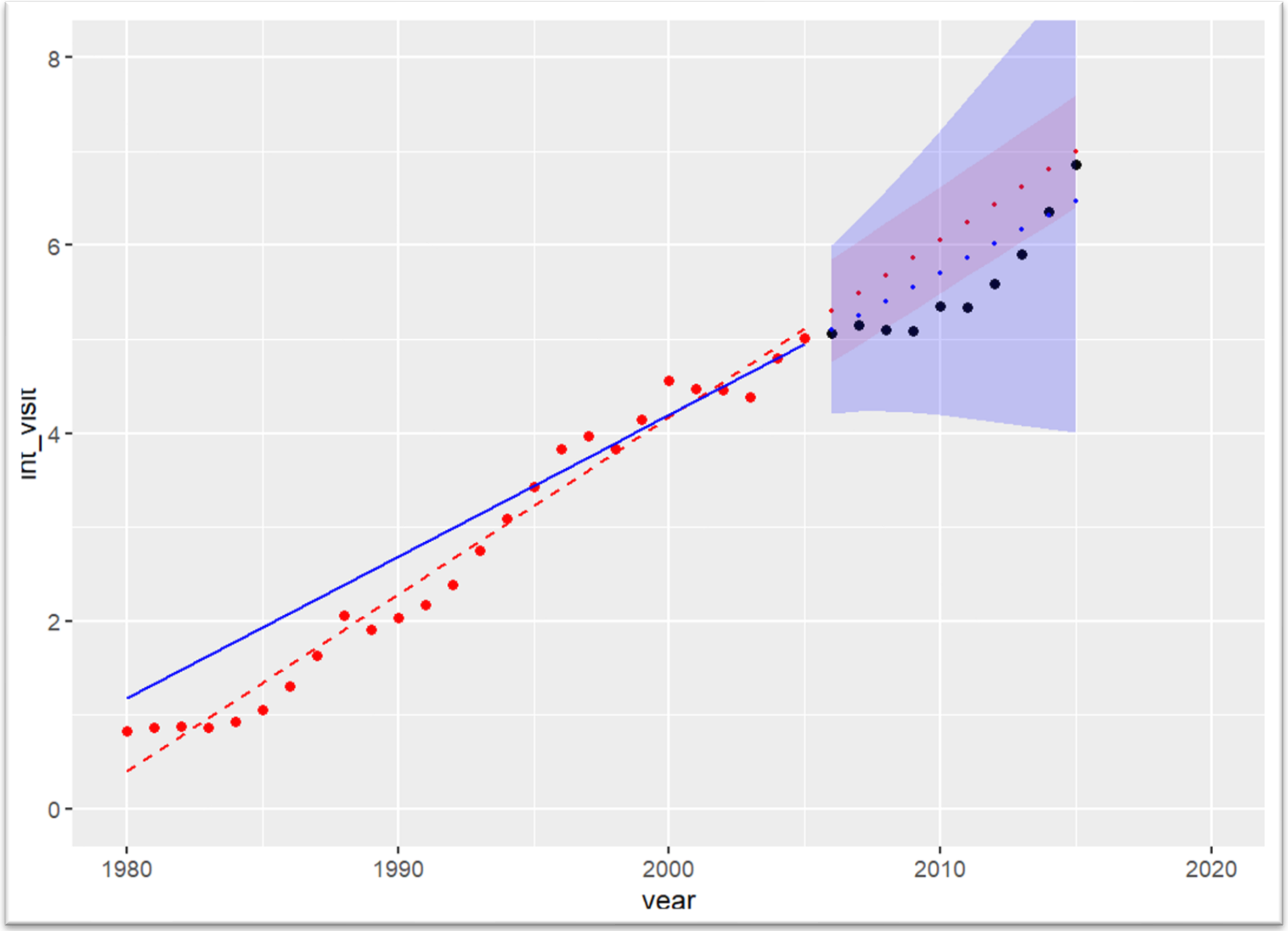
The sum of weights must be larger than the number of parameters estimated.

This requirement is a restriction on lambda

(lambda cannot be too small)

(note: the estimator is not in the book, but this is a ”sneak peak” into chapter 11)

Example in R



Prediction intervals using:

$$T = \sum_{j=0}^{N-1} \lambda^j$$

$$\hat{\sigma}^2 = (\boldsymbol{Y} - \boldsymbol{x}_N \widehat{\boldsymbol{\theta}}_N)^T \Sigma^{-1} (\boldsymbol{Y} - \boldsymbol{x}_N \widehat{\boldsymbol{\theta}}_N)/(T - p)$$

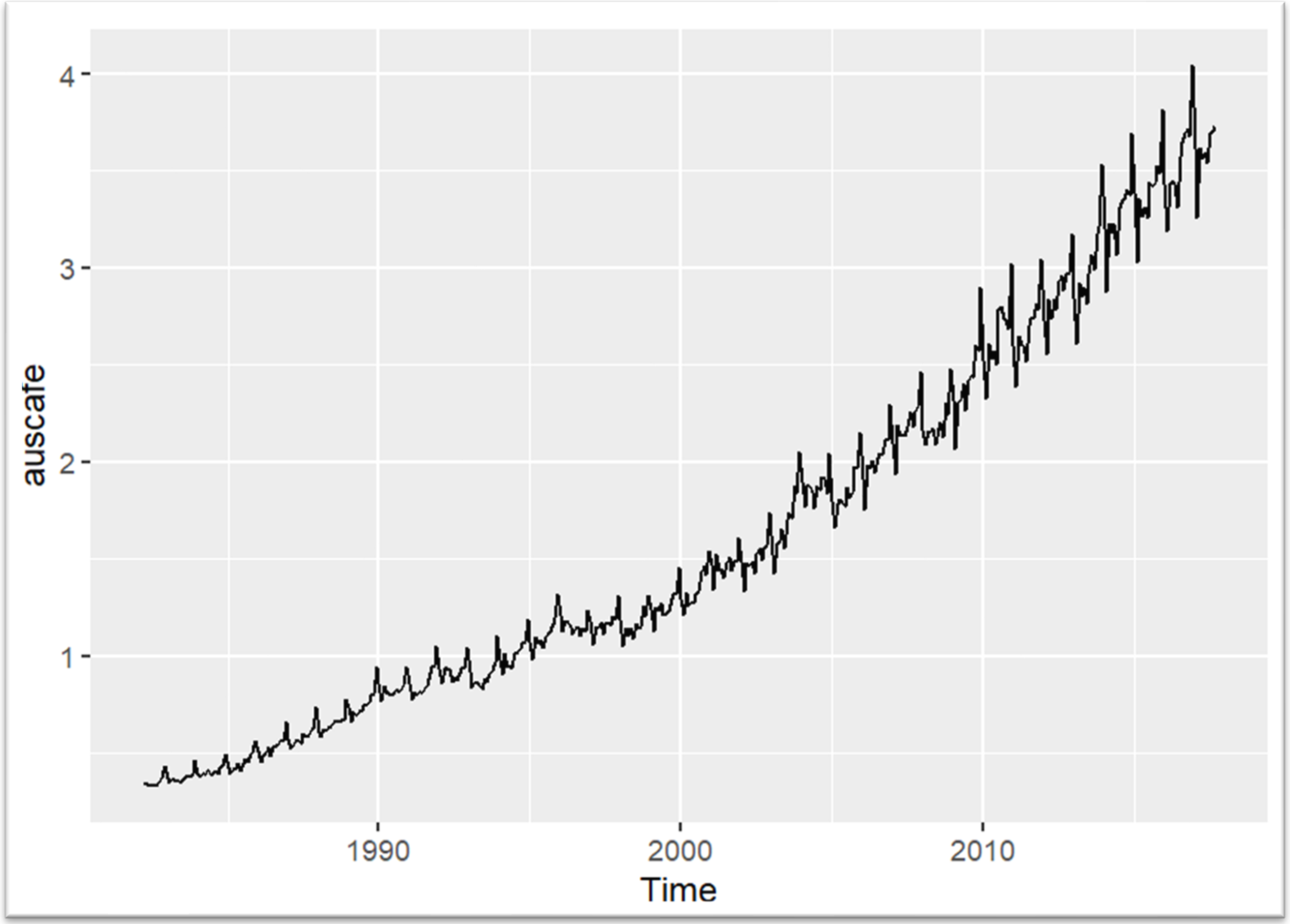
Here T = 2.5

Choice of λ

$$\Sigma = \begin{bmatrix} 1/\lambda^{n-1} & \dots & 0 & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \dots & 1/\lambda^2 & 0 & 0 \\ 0 & \dots & 0 & 1/\lambda & 0 \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}$$

Small λ: short ”memory” (fast ”forgetting”)

λ = 1: OLS (full ”memory”)



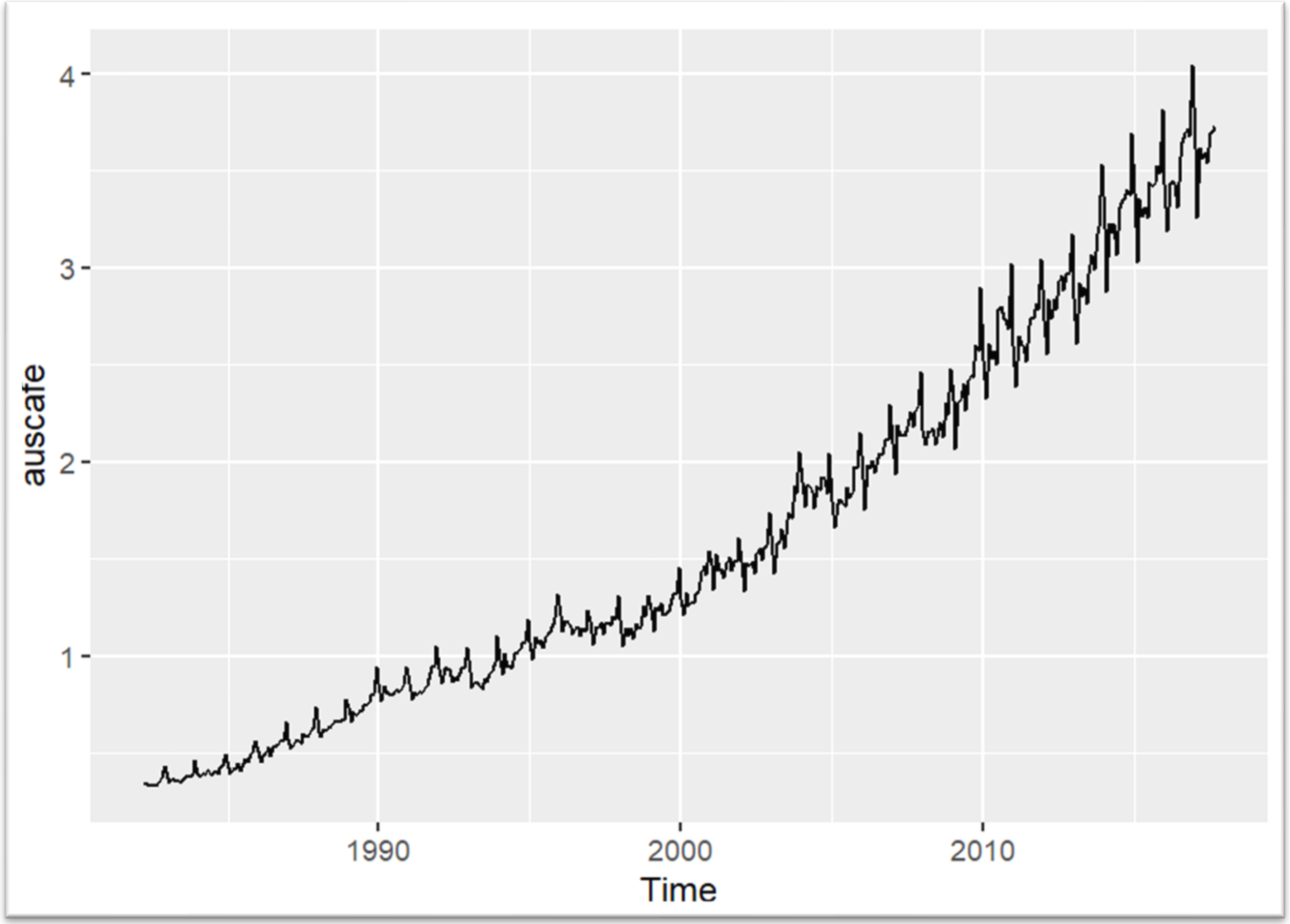
Choice of λ

Optimal choice of λ may depend on the ”prediction horizon”

(e.g., one day ahead, one weak ahead, one year ahead…)

We can use historical data to find optimal λ for a specific prediction horizon:

- Iteratively run through data
- Make prediction for each itteration
- Evaluate prediction accuracies (given specific λ value)



Outline of the lecture

- Recap: Ordinary Least Squares (OLS) and its assumptions
- Weighted Least Squares (WLS)
- Weighted Least Squares for “Local Trend Models”
- **Recursive Least Squares with forgetting**
- Exponential smoothing in general

Recursive Least Squares (recap)

Re-writing OLS in an iterative way:

notation: t = n , ie. the ”latest” observation (t) is also the total number of observations (n)

$$\widehat{\boldsymbol{\theta}}_t = (\boldsymbol{X}_t^T \boldsymbol{X}_t)^{-1} \boldsymbol{X}_t^T \boldsymbol{y}_t = \boldsymbol{R}_t^{-1} \boldsymbol{h}_t$$

$$\boldsymbol{R}_t = \boldsymbol{X}_t^T \boldsymbol{X}_t = \boldsymbol{x}_1 \boldsymbol{x}_1^T + \boldsymbol{x}_2 \boldsymbol{x}_2^T + \ldots + \boldsymbol{x}_t \boldsymbol{x}_t^T = \sum_{s=1}^{t-1} \boldsymbol{x}_s \boldsymbol{x}_s^T + \boldsymbol{x}_t \boldsymbol{x}_t^T = \boldsymbol{R}_{t-1} + \boldsymbol{x}_t \boldsymbol{x}_t^T$$

$$\boldsymbol{h}_t = \boldsymbol{X}_t^T \boldsymbol{y}_t = \boldsymbol{x}_1 Y_1 + \boldsymbol{x}_2 Y_2 + \ldots + \boldsymbol{x}_t Y_t = \sum_{s=1}^{t-1} \boldsymbol{x}_s Y_s + \boldsymbol{x}_t Y_t = \boldsymbol{h}_{t-1} + \boldsymbol{x}_t Y_t$$

$$\boldsymbol{R}_t = \boldsymbol{R}_{t-1} + \boldsymbol{x}_t \boldsymbol{x}_t^T$$

$$\widehat{\boldsymbol{\theta}}_t = \widehat{\boldsymbol{\theta}}_{t-1} + \boldsymbol{R}_t^{-1} \boldsymbol{x}_t (Y_t - \boldsymbol{x}_t^T \widehat{\boldsymbol{\theta}}_{t-1})$$

Making it easy to update the parameter estimates, when new data is available

Recursive Least Squares with “forgetting”

Re-writing **WLS** where Σ^{-1} is a diagonal matrix with elements β(t,s) = $\lambda^{(t-s)}$, (s = 1, 2, 3, …, t):

$$\widehat{\boldsymbol{\theta}} = (\boldsymbol{x}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{x})^{-1} \boldsymbol{x}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{Y}$$

$$\boldsymbol{R}_t = \sum_{s=1}^{t} \beta(t,s) \boldsymbol{X}_s \boldsymbol{X}_s^T, \quad \boldsymbol{h}_t = \sum_{s=1}^{t} \beta(t,s) \boldsymbol{X}_s Y_s$$

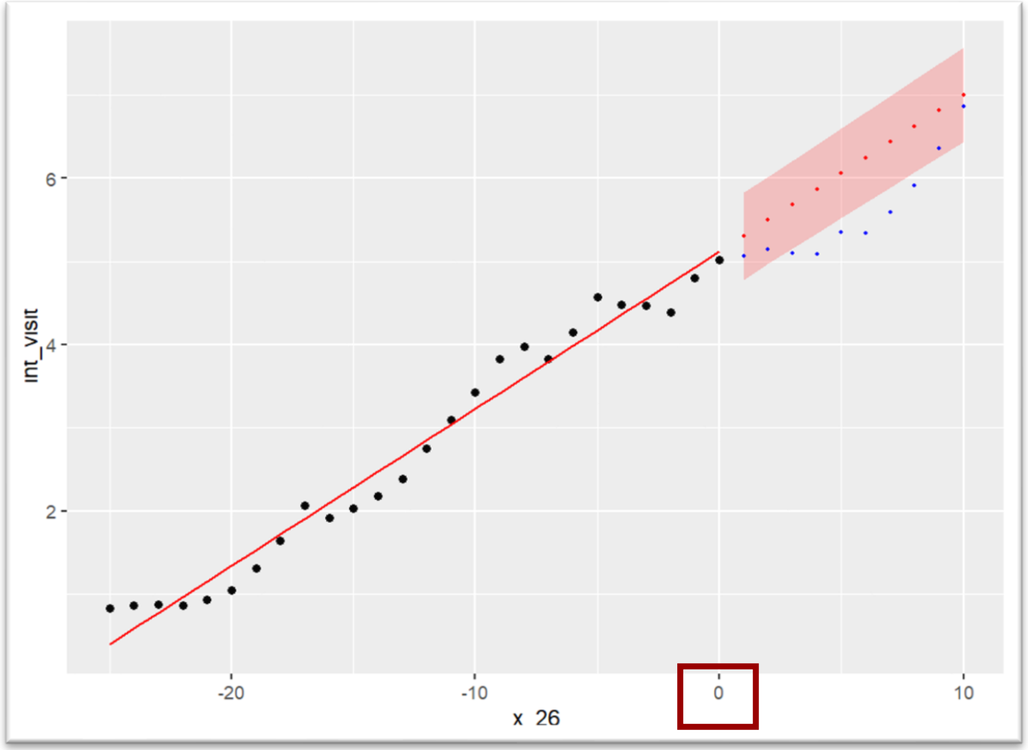
$$\Sigma = \begin{bmatrix} 1/\lambda^{n-1} & \dots & 0 & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \dots & 1/\lambda^2 & 0 & 0 \\ 0 & \dots & 0 & 1/\lambda & 0 \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}$$

$$\widehat{\boldsymbol{\theta}}_t = \widehat{\boldsymbol{\theta}}_{t-1} + \boldsymbol{R}_t^{-1} \boldsymbol{X}_t \left[Y_t - \boldsymbol{X}_t^T \widehat{\boldsymbol{\theta}}_{t-1} \right]$$

$$\boldsymbol{R}_t = \lambda(t) \boldsymbol{R}_{t-1} + \boldsymbol{X}_t \boldsymbol{X}_t^T$$

Local Trend Models in chapter 3.4

$$Y_{N+j} = f^T(j)\boldsymbol{\theta} + \varepsilon_{N+j}$$



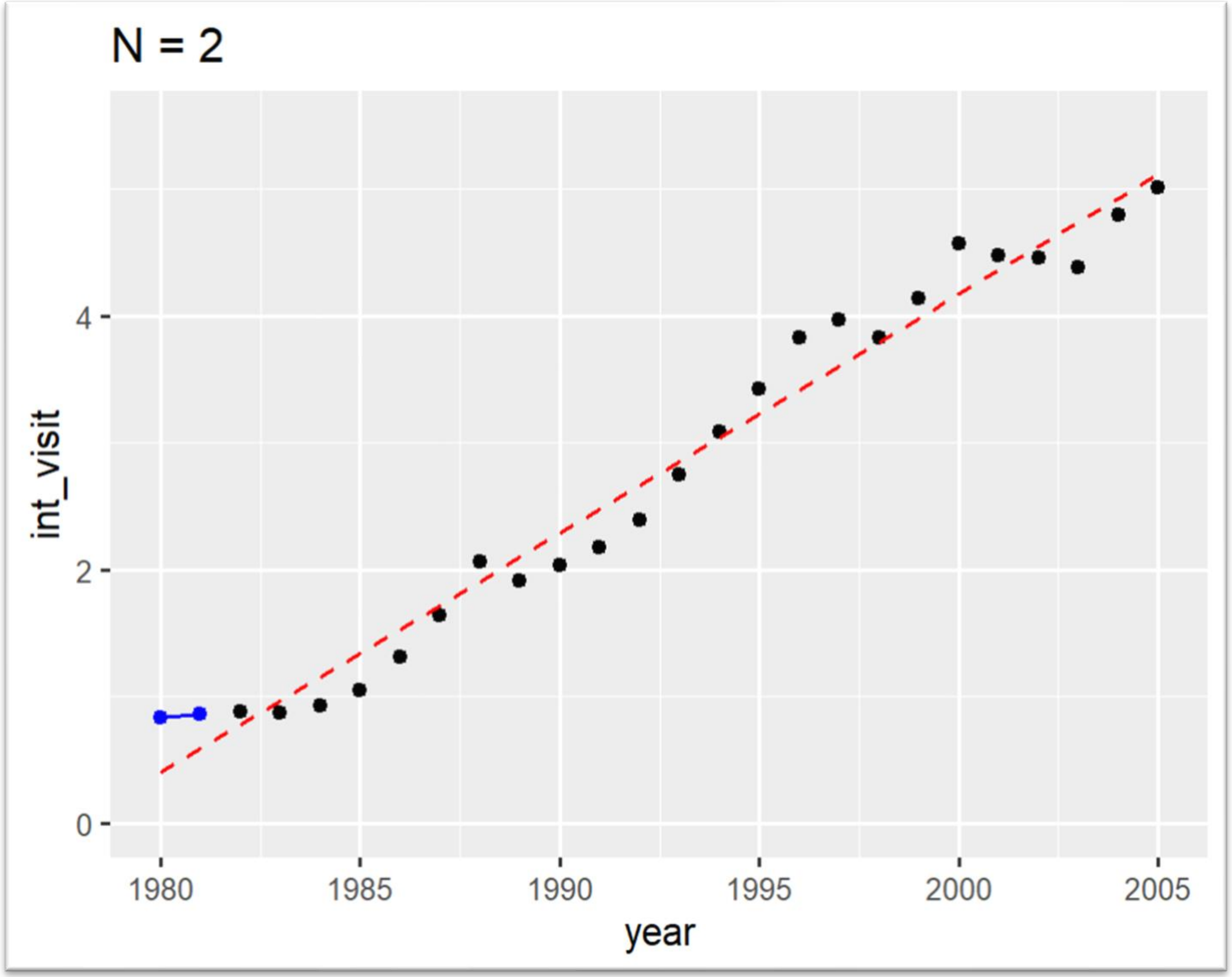
The Trend Model as described in chapter 3.4 is almost the same as RLS, but with the extra detail that the x-axis is updated in every timepoint, such that the current time is equal to time zero.

$$\widehat{\boldsymbol{\theta}}_N = \boldsymbol{F}_N^{-1}\boldsymbol{h}_N$$

$$\boldsymbol{F}_N = \sum_{j=0}^{N-1} \lambda^j \boldsymbol{f}(-j)\boldsymbol{f}^T(-j)$$

$$\boldsymbol{h}_N = \sum_{j=0}^{N-1} \lambda^j \boldsymbol{f}(-j)Y_{N-j}$$

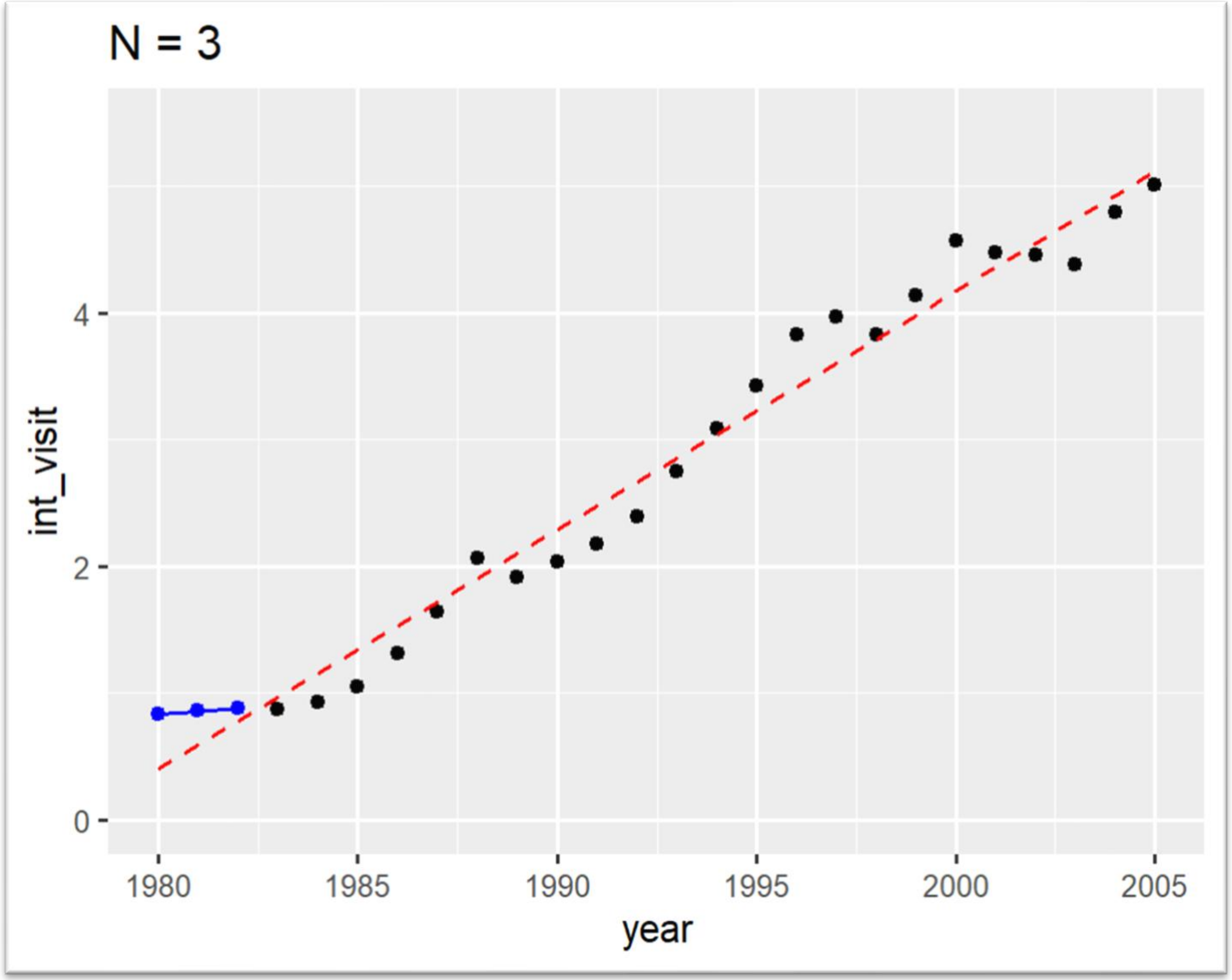
R example – RLS with forgetting



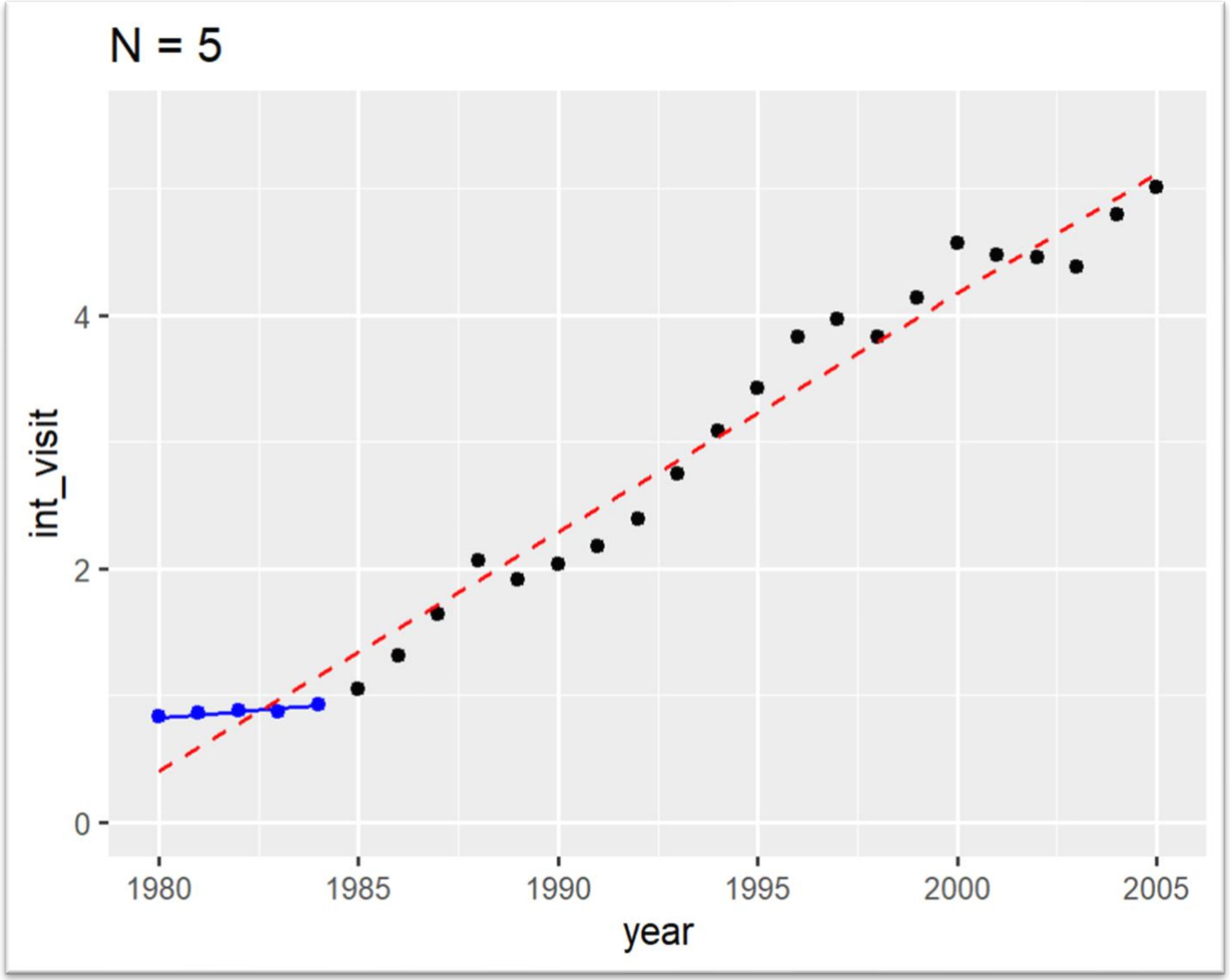
Blue line is local model based on only two observations

Red line is the original OLS based on all 26 observations

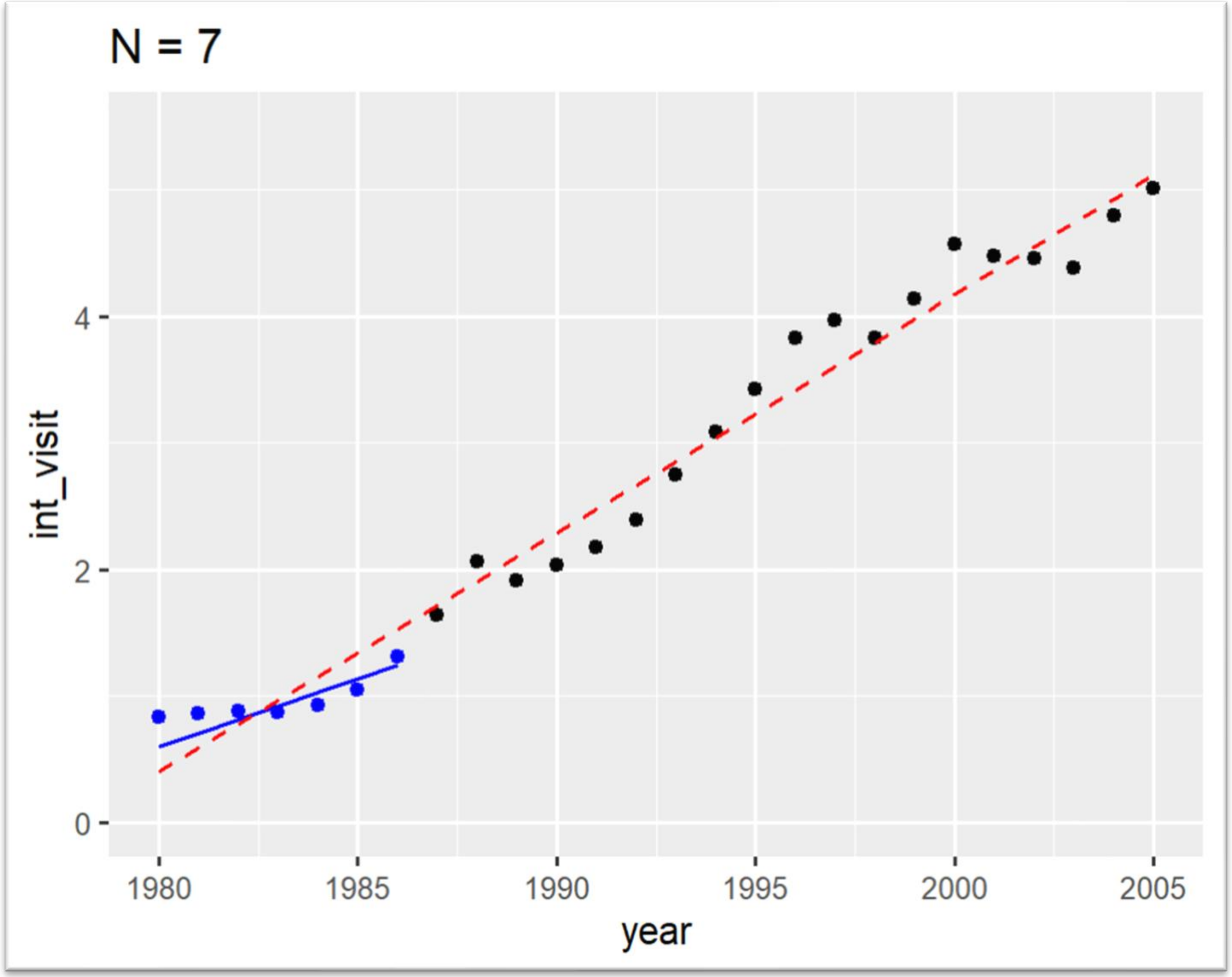
R example – RLS with forgetting



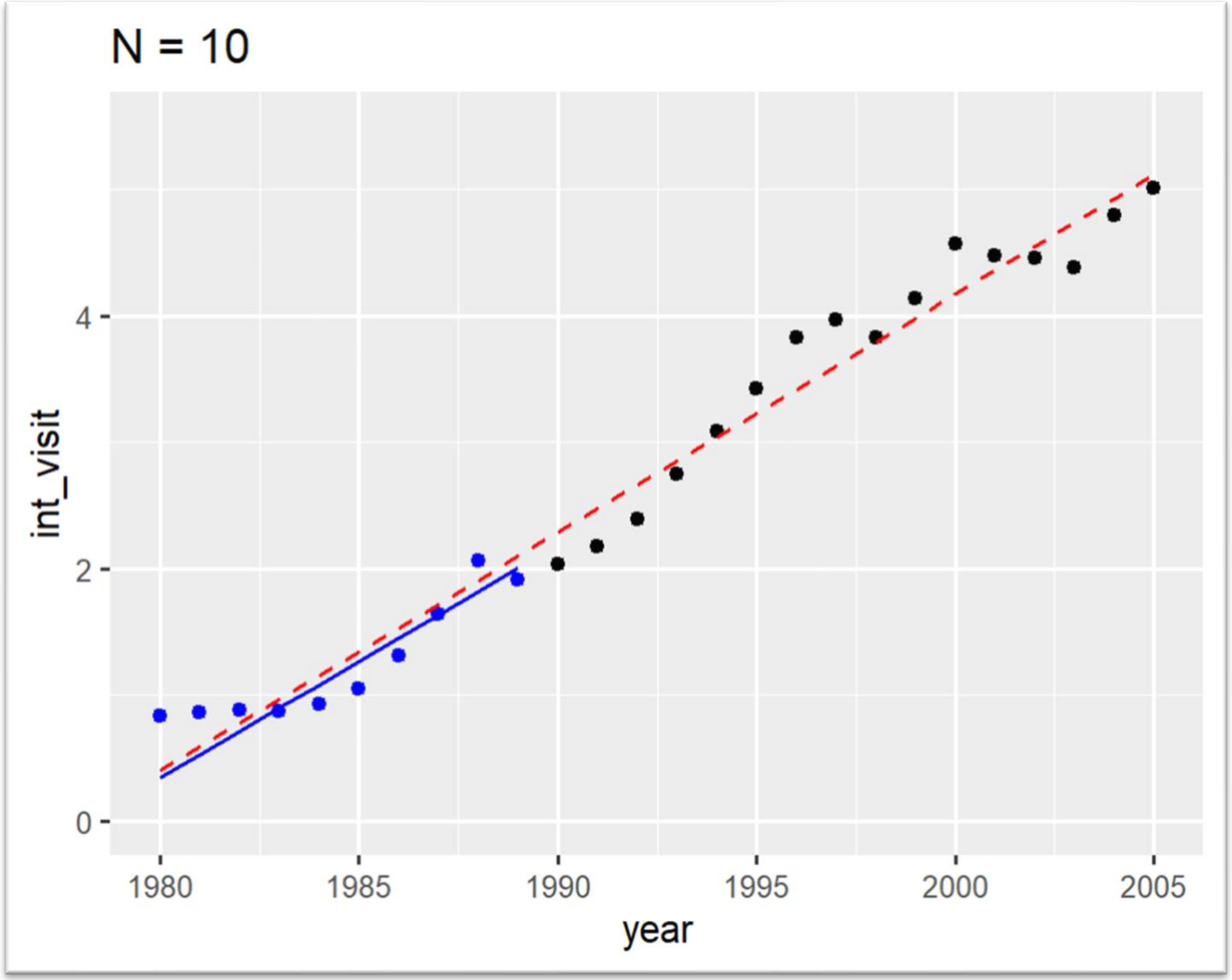
R example – RLS with forgetting



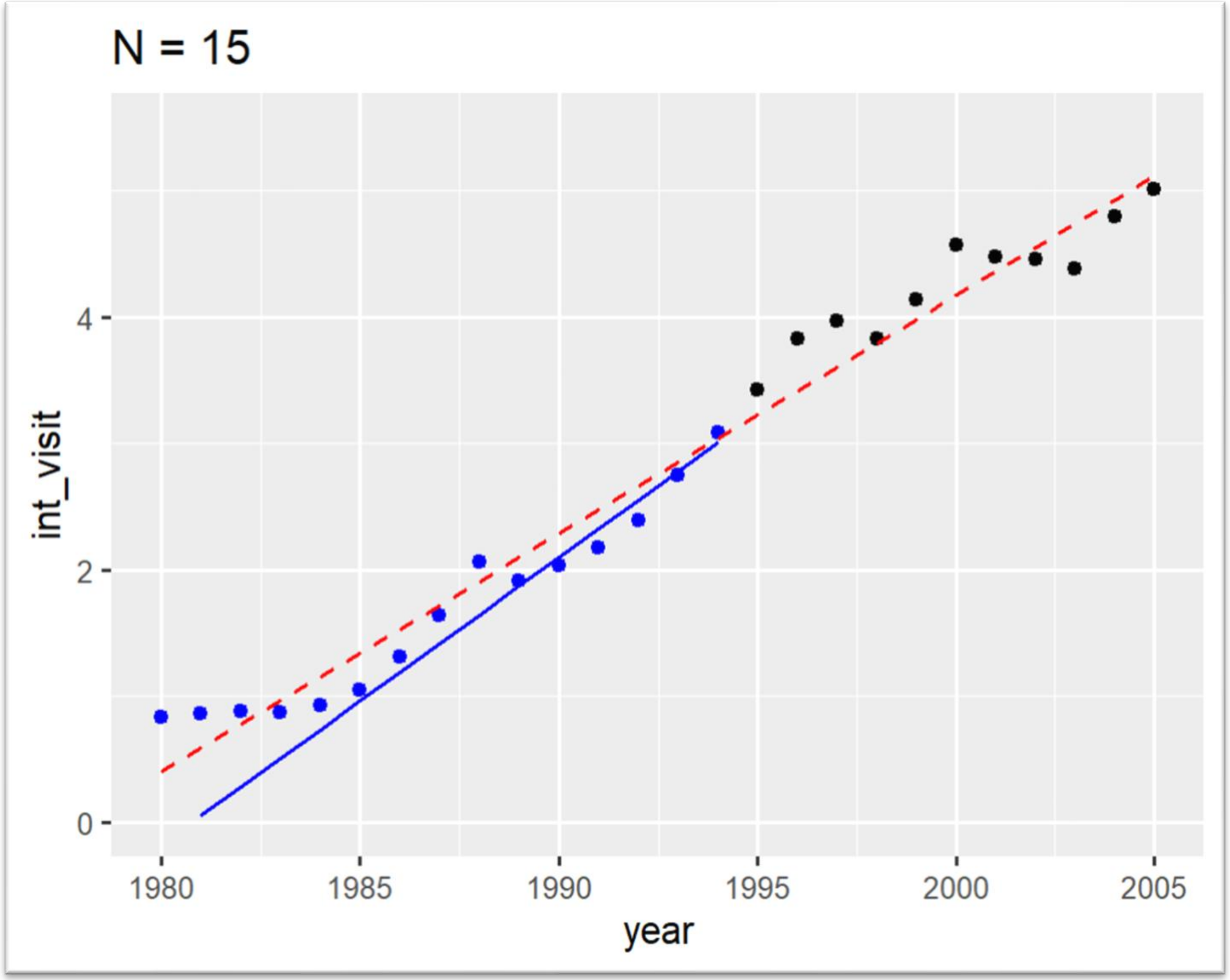
R example – RLS with forgetting



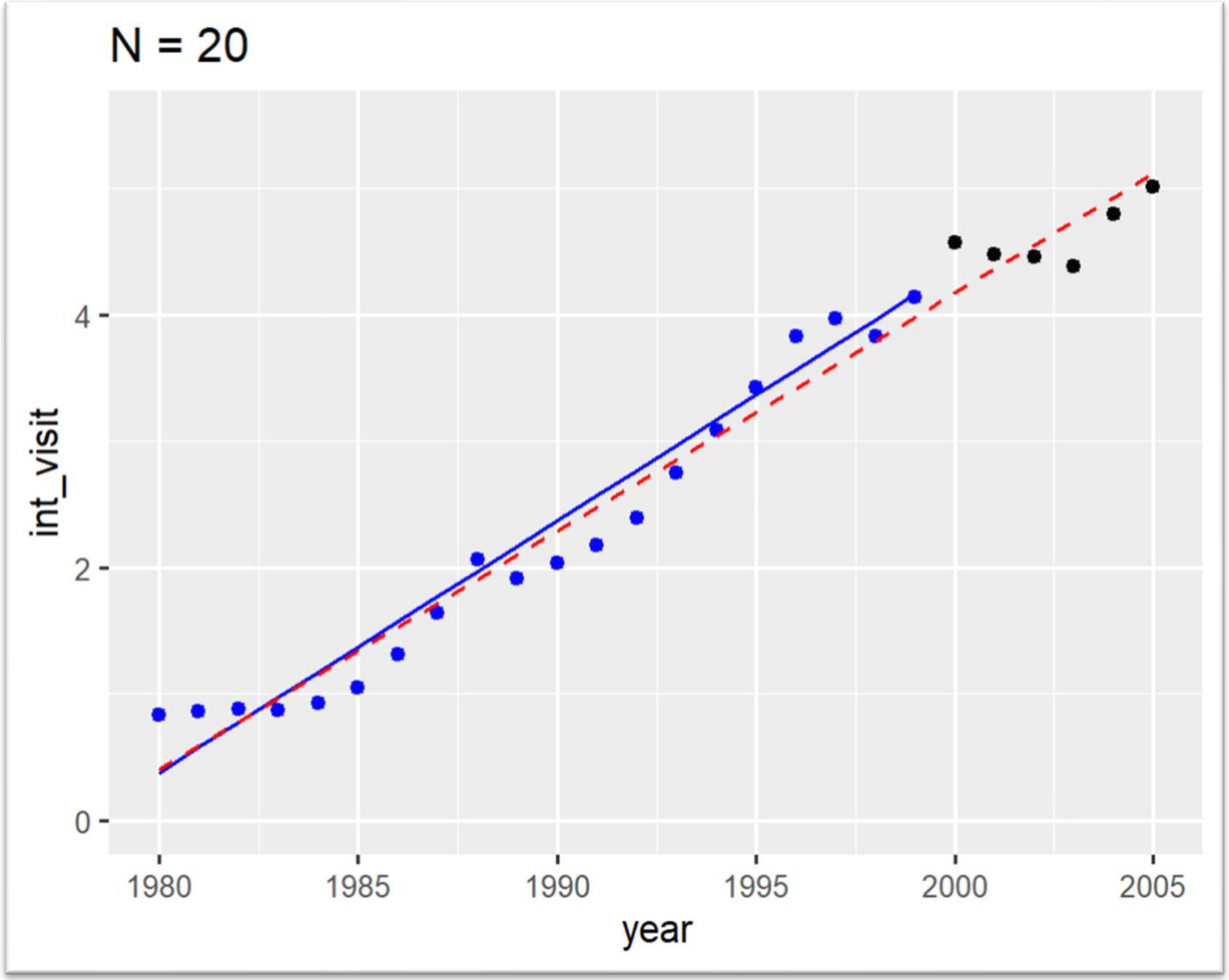
R example – RLS with forgetting



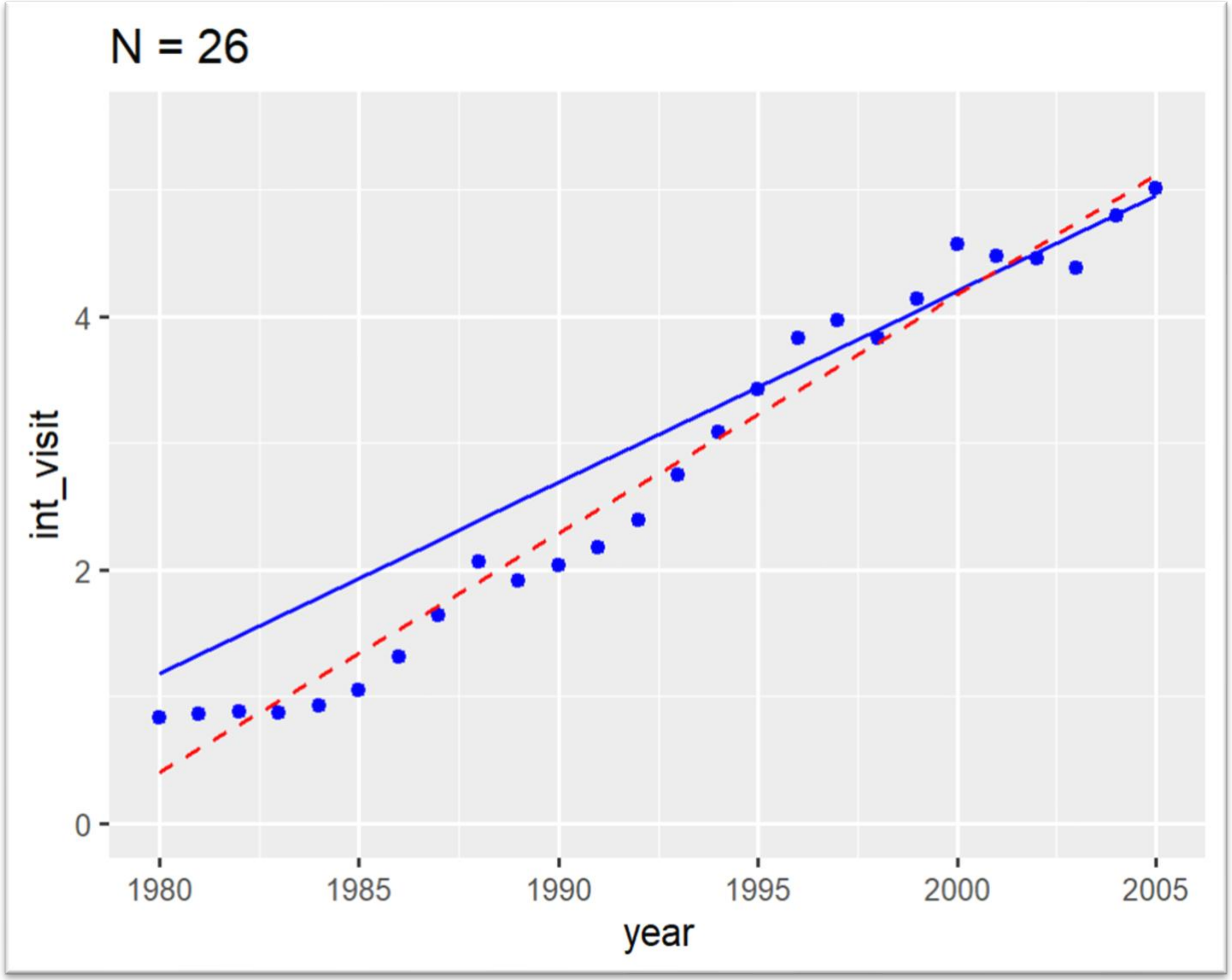
R example – RLS with forgetting



R example – RLS with forgetting



R example – RLS with forgetting



Remember that the blue line is WLS, such that latest timepoints have higher weight.

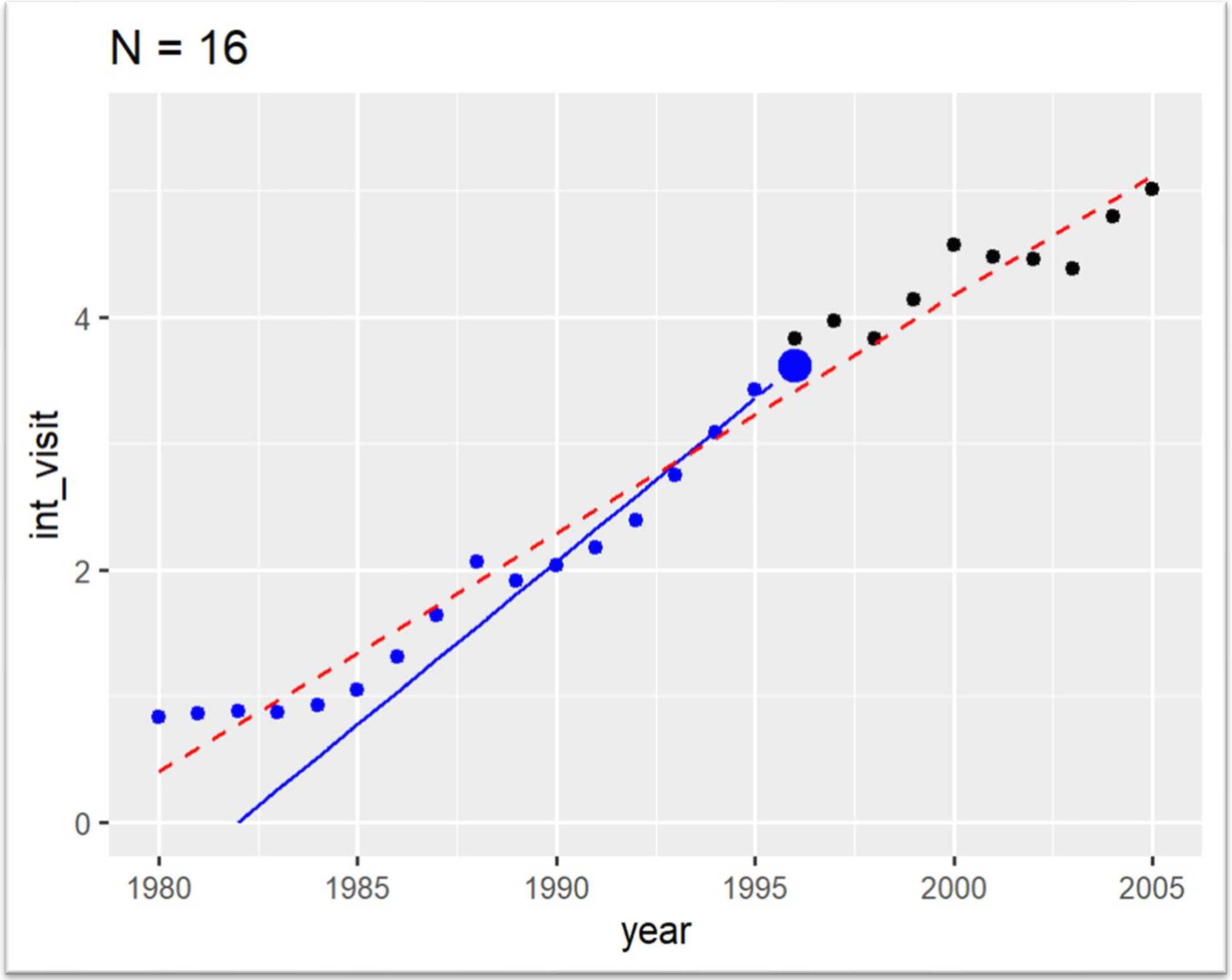
Here we used lambda = 0.6

$$\Sigma = \text{diag}[1/\lambda^{N-1}, \dots, 1/\lambda, 1]$$

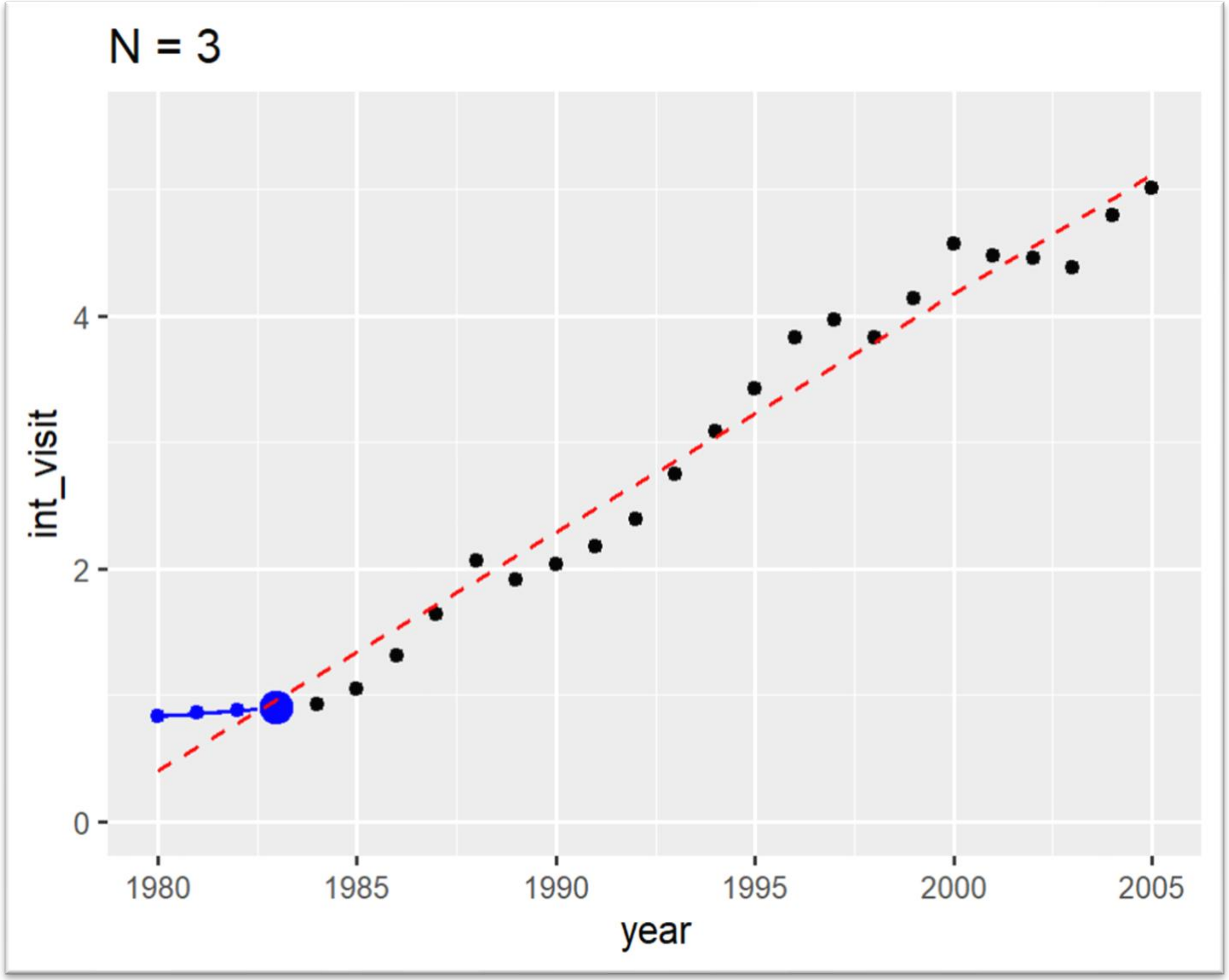
L-step predictions

At each iteration we can make a prediction L steps into the future

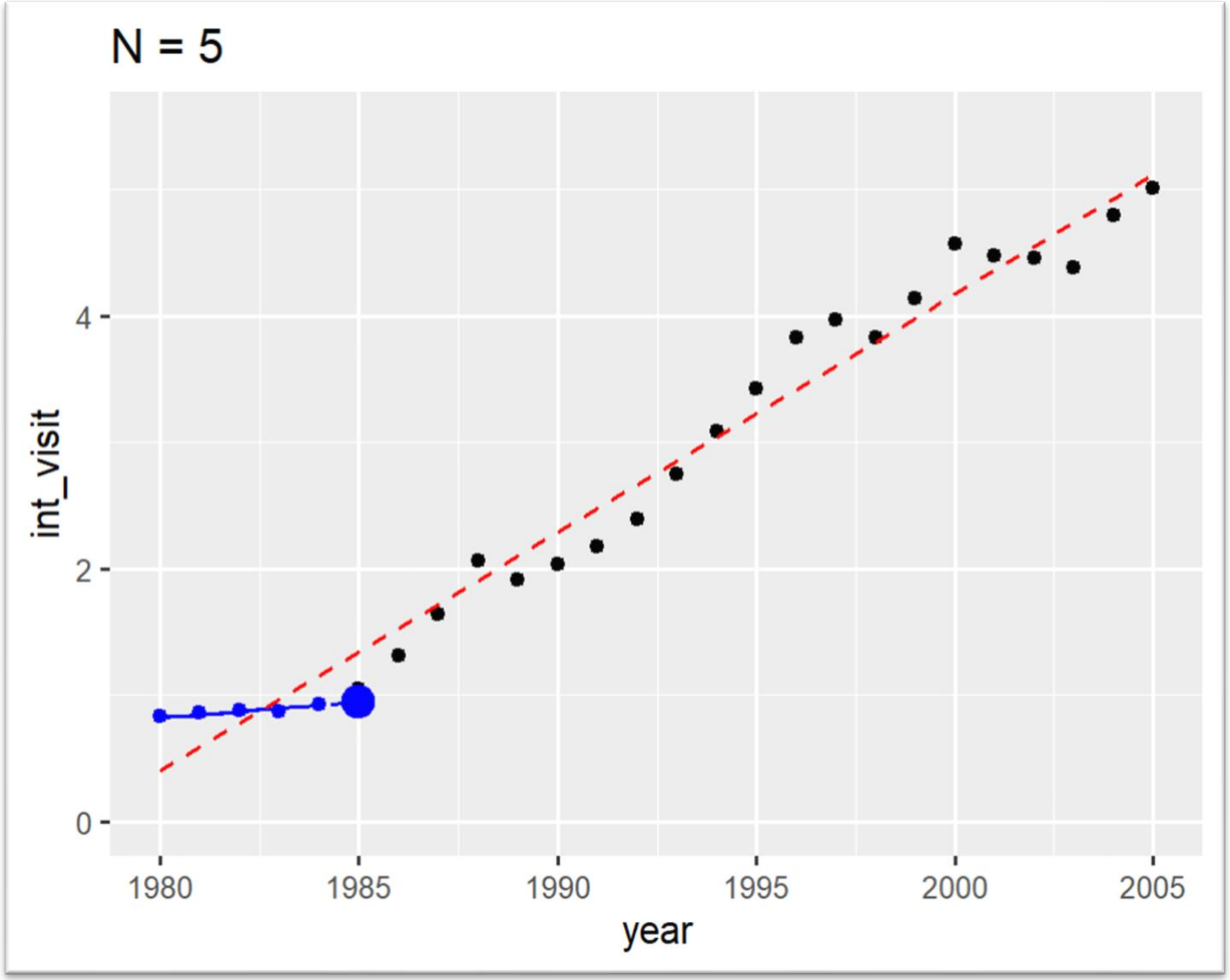
Here we visualise a one-step prediction



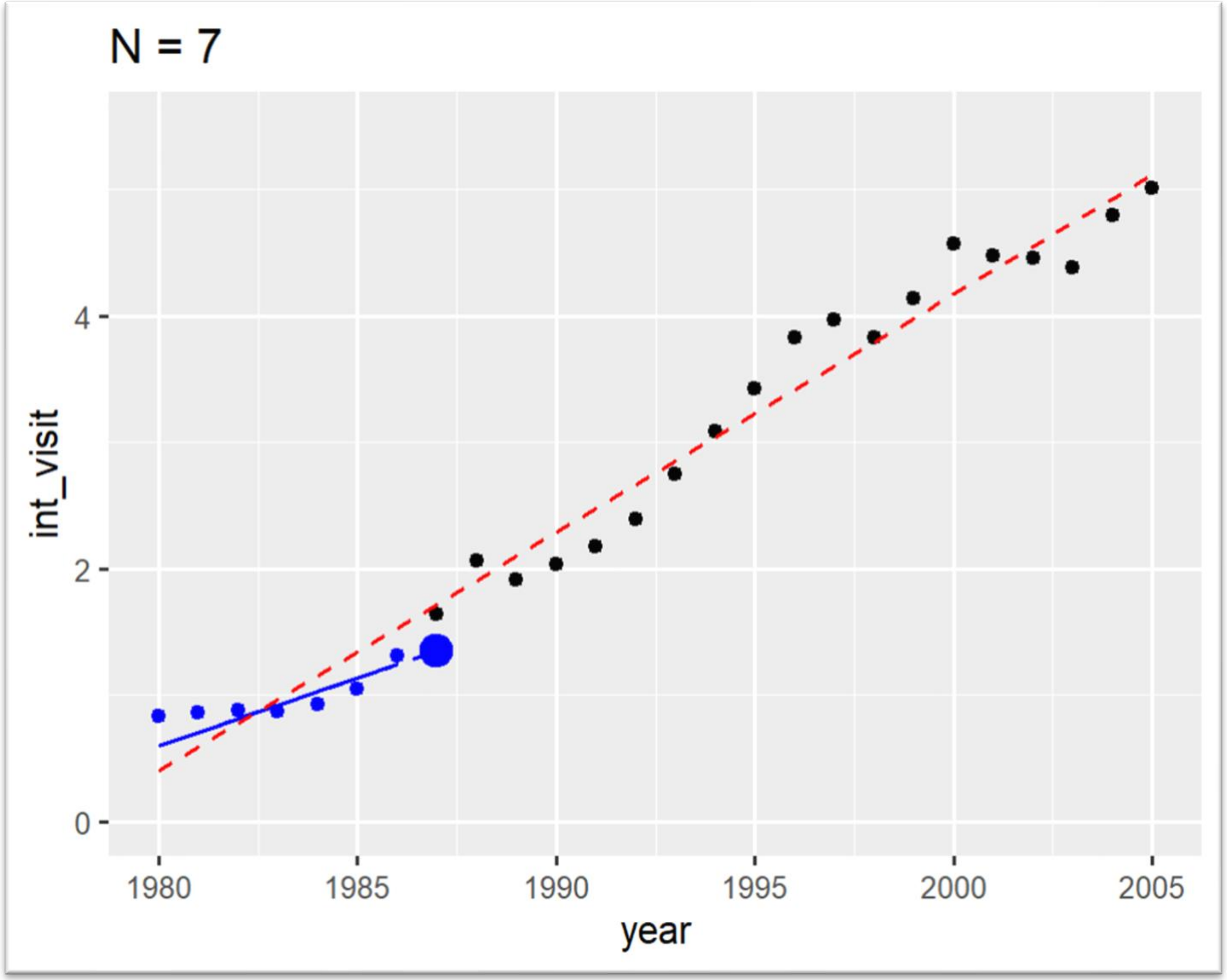
R example – onestep prediction



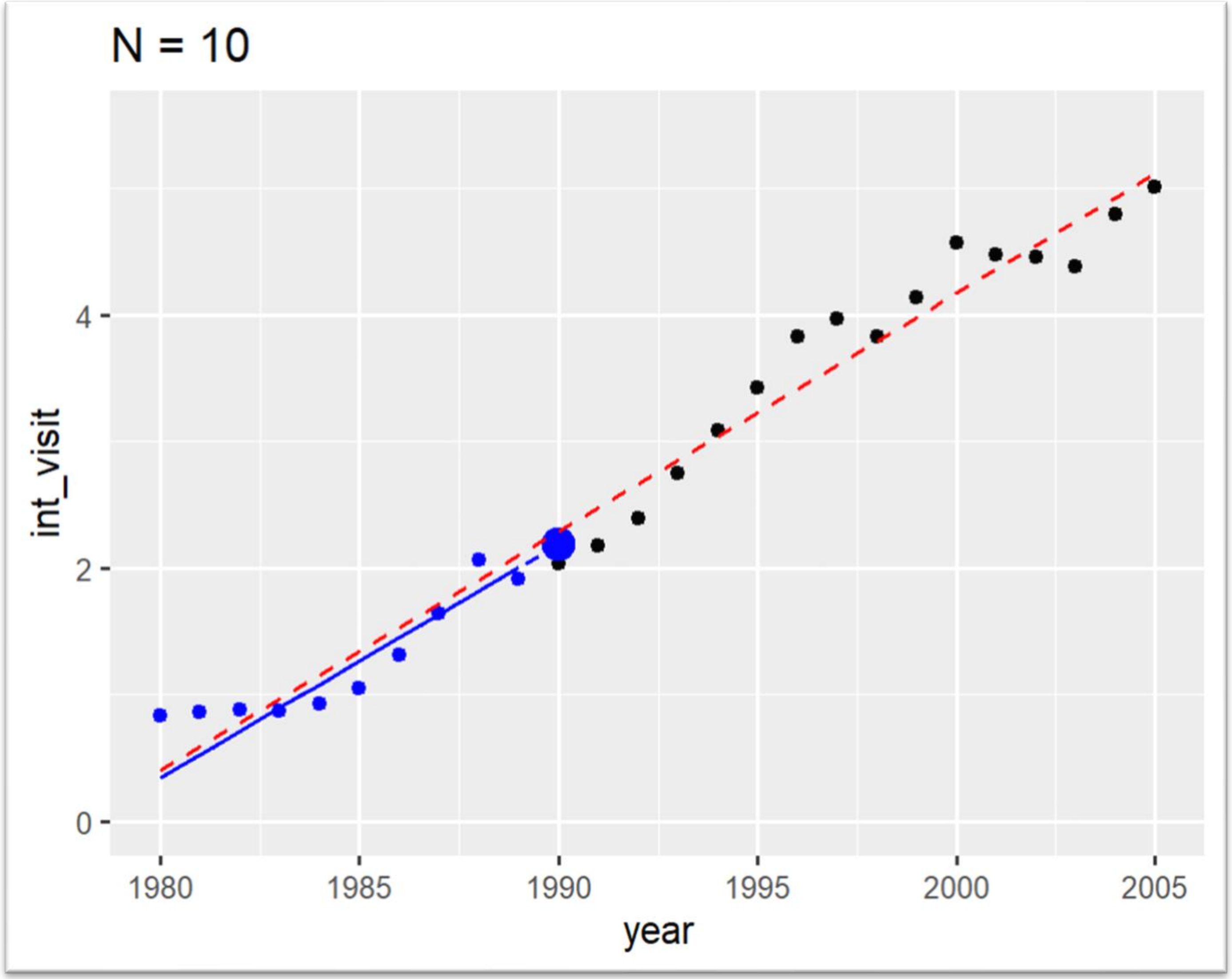
R example – onestep prediction



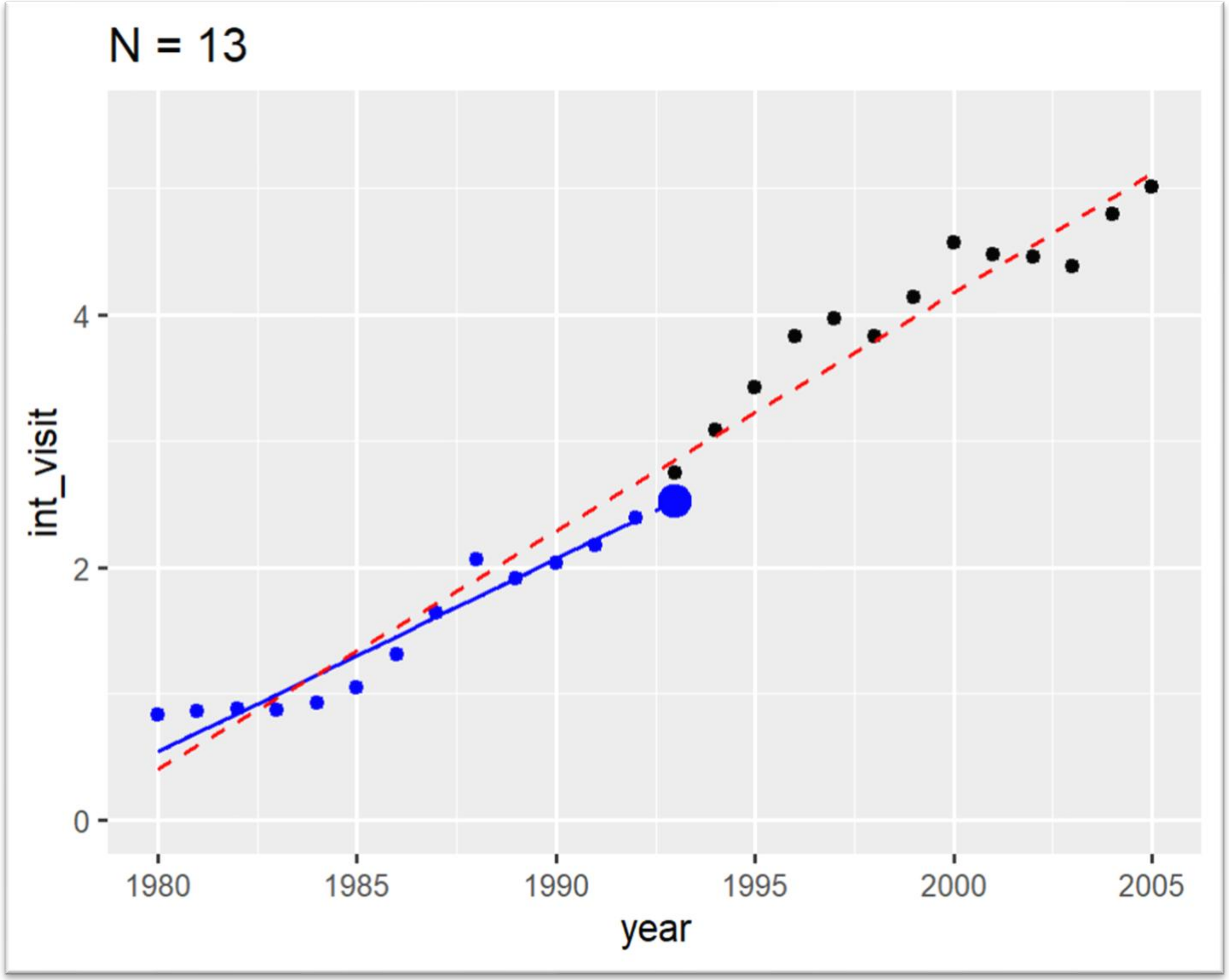
R example – onestep prediction



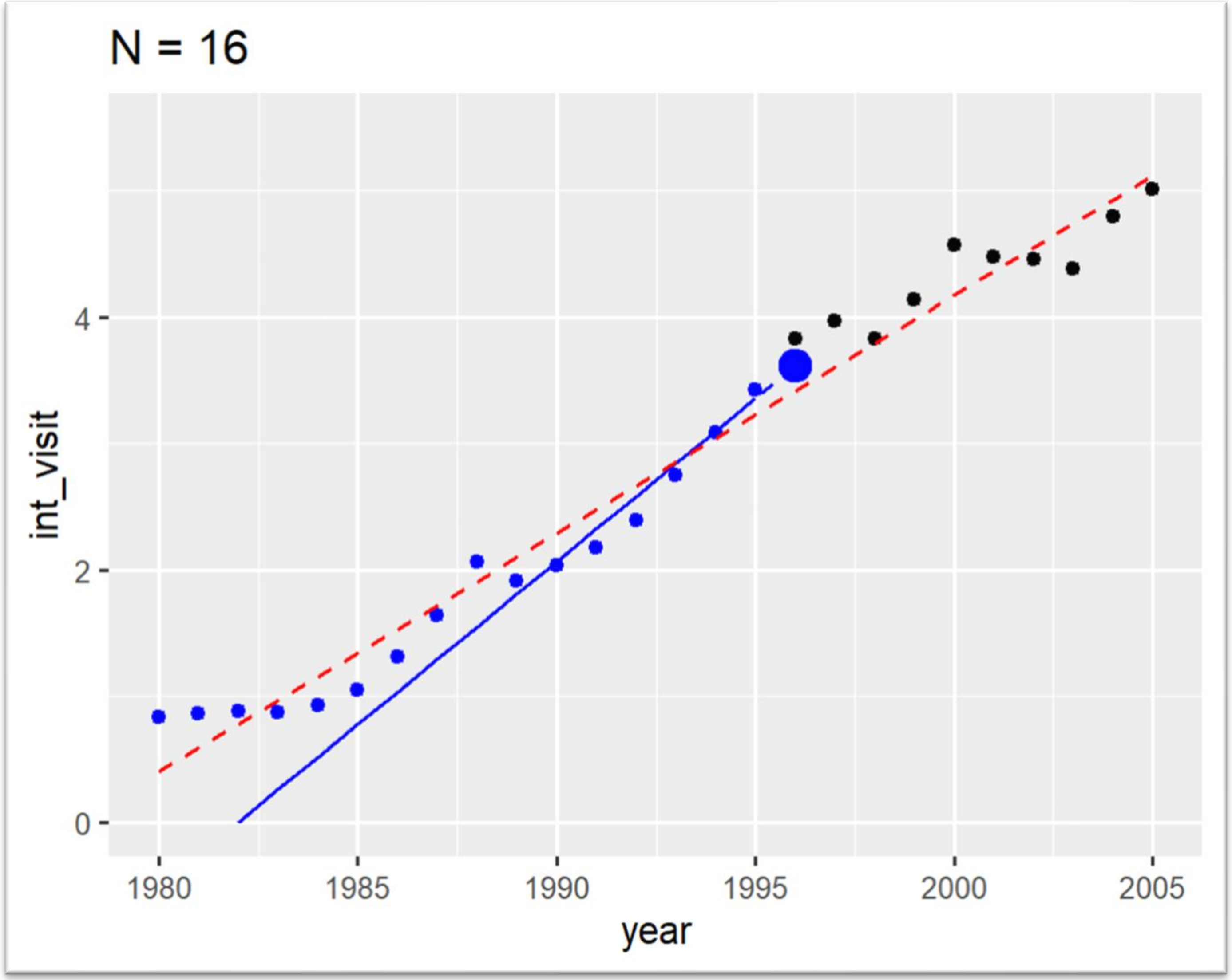
R example – onestep prediction



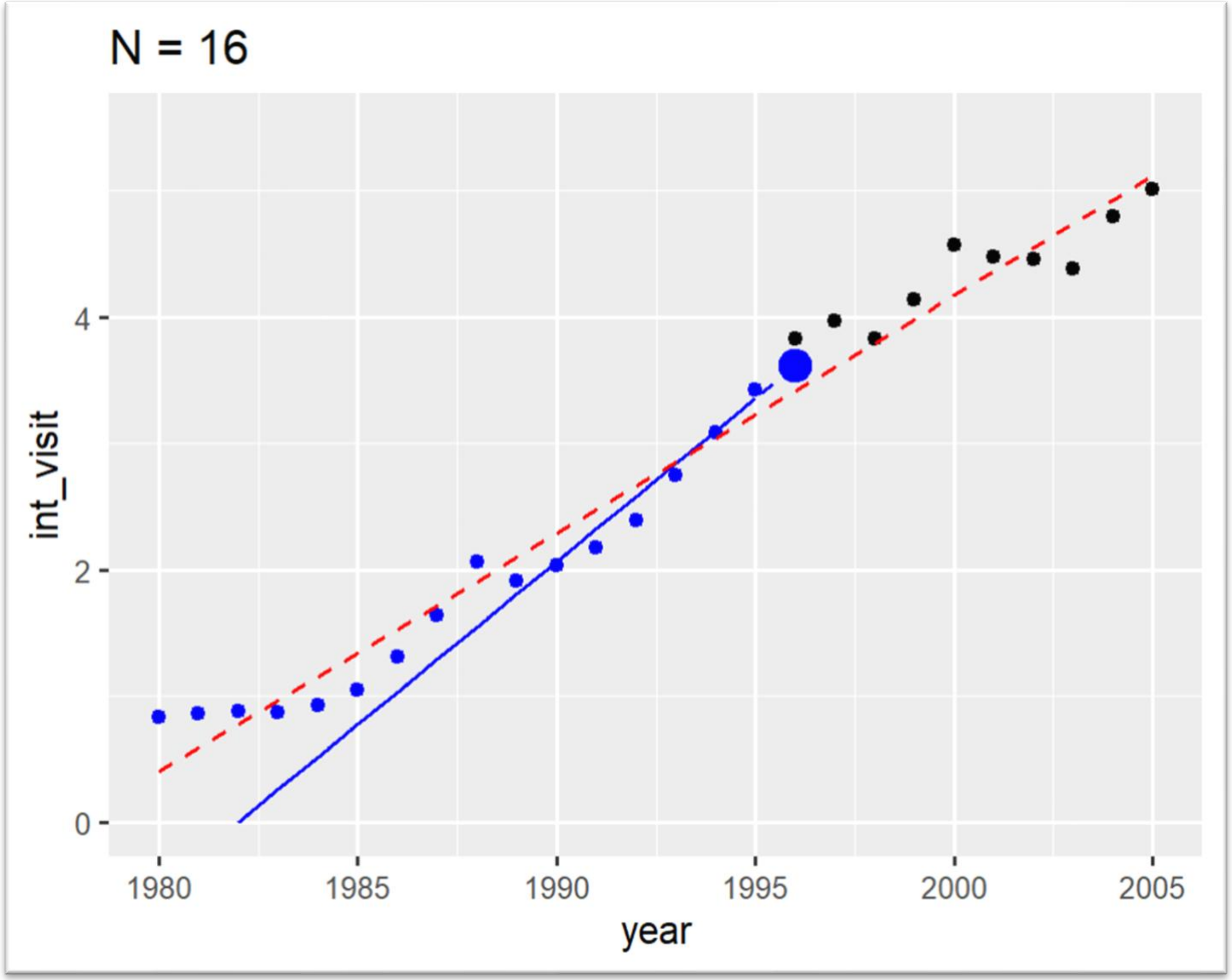
R example – onestep prediction



R example – onestep prediction

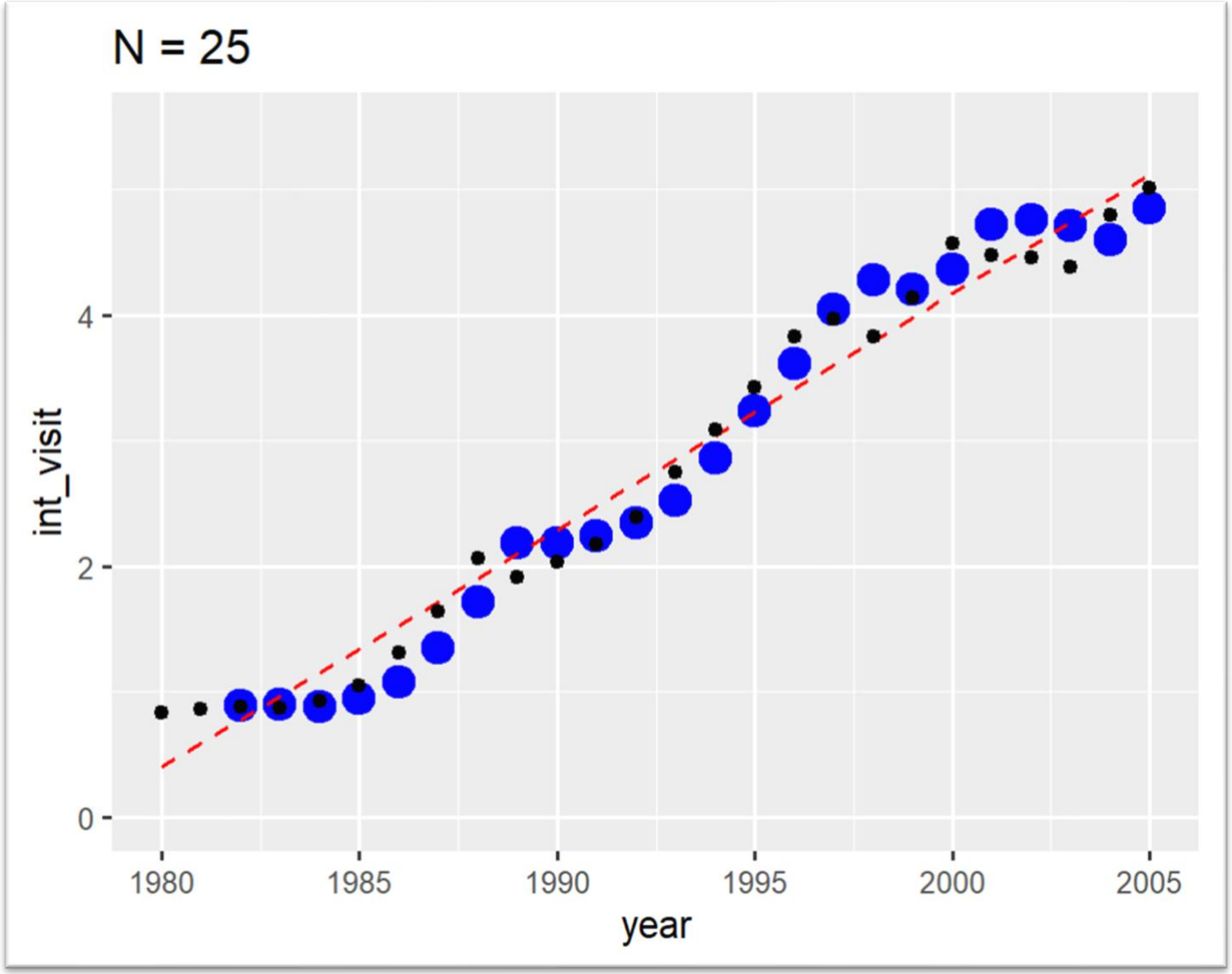


R example – onestep prediction



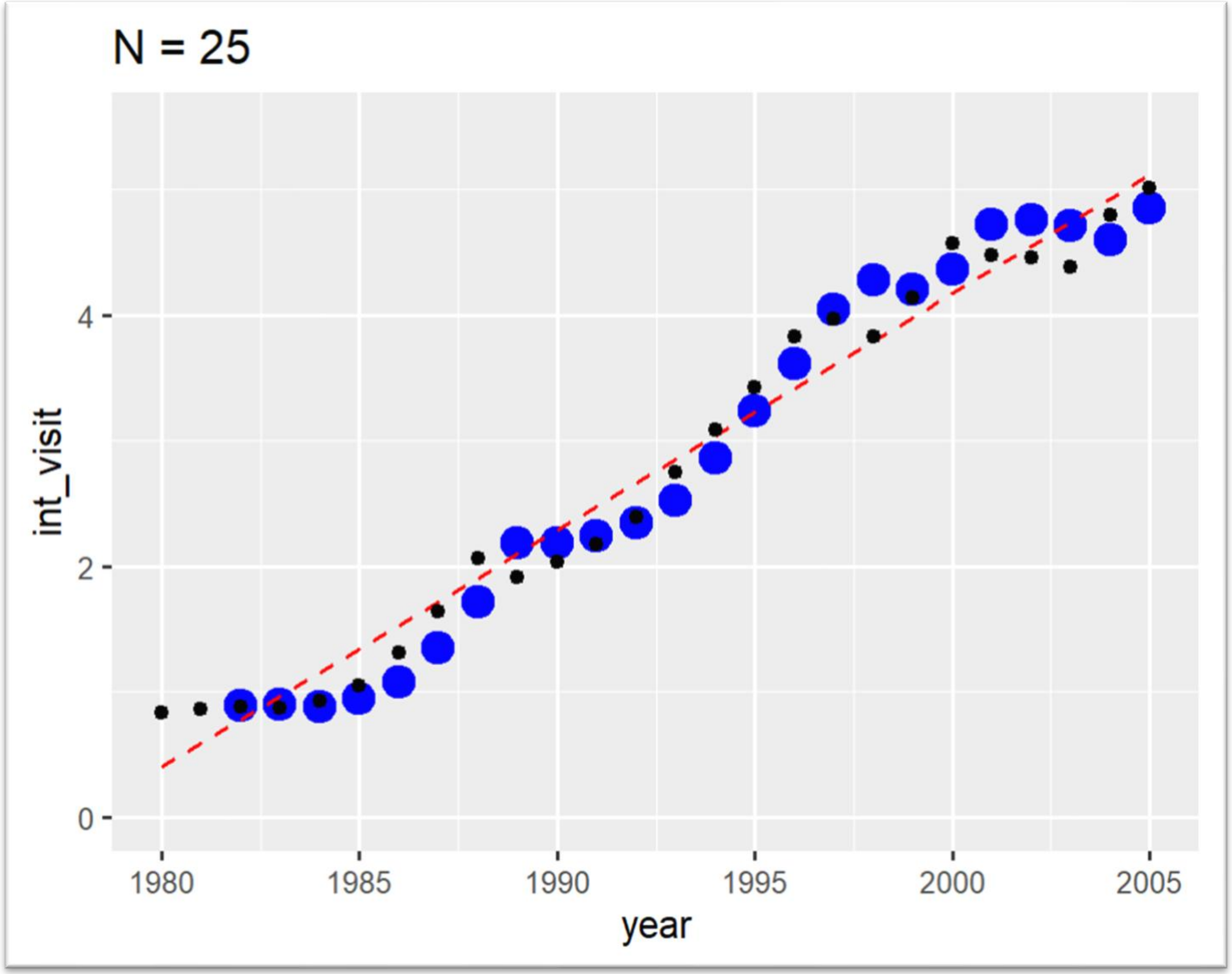
Each onestep prediction is predicted using a new (updated) set of parameters

R example – onestep prediction



Plot of all the onestep predictions

R example – onestep prediction

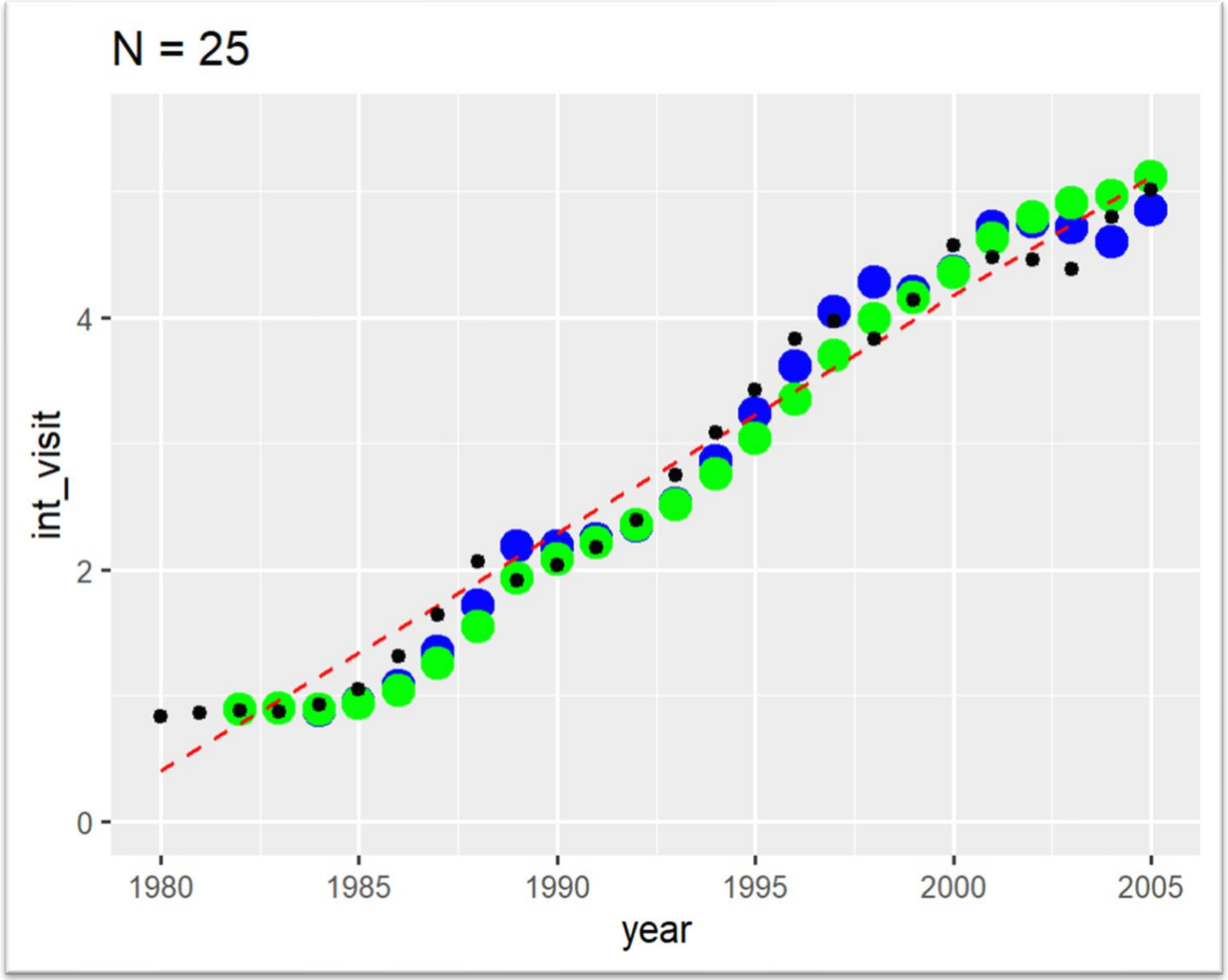


Plot of all the onestep predictions

We can try with different values of lambda

(code in R script)

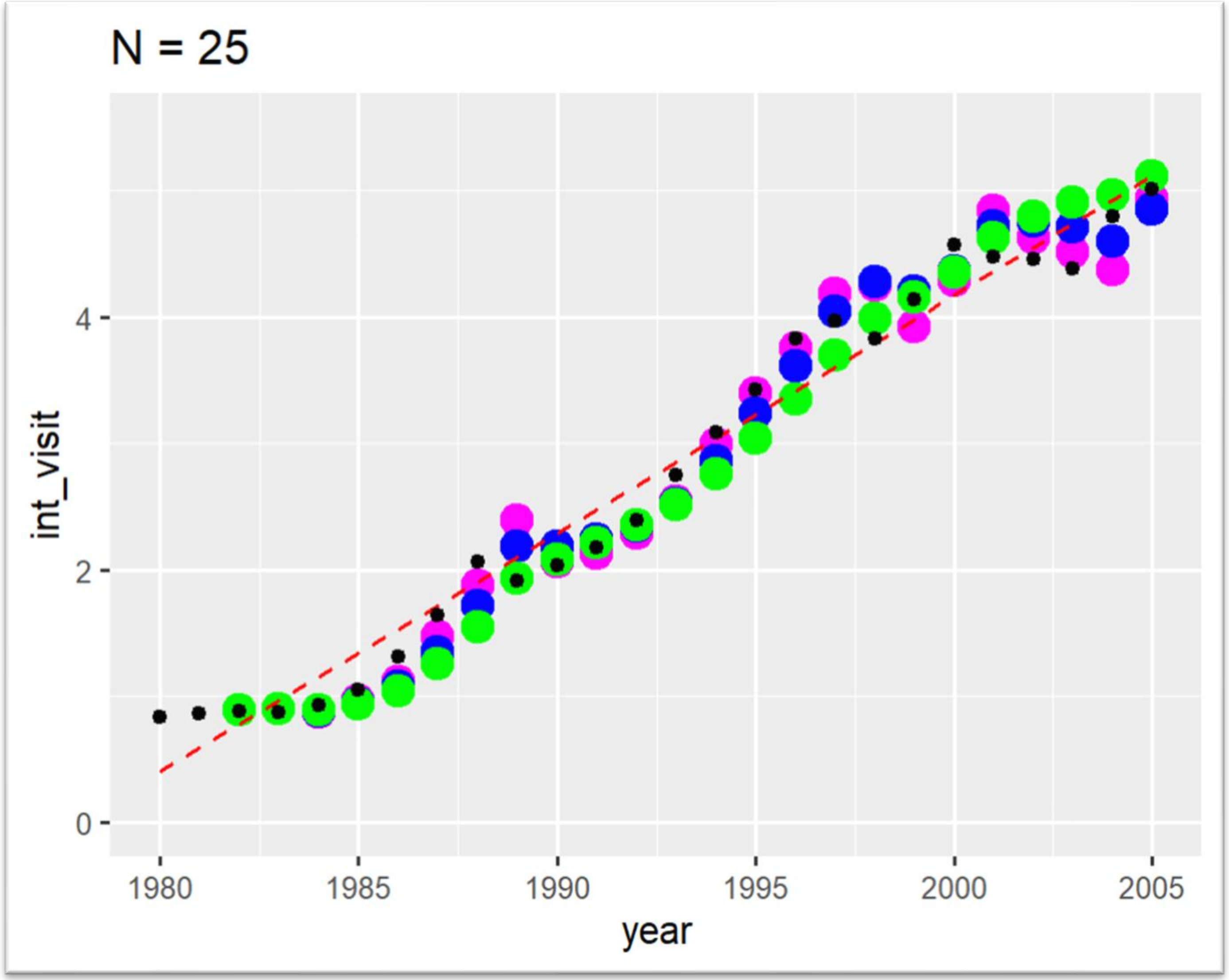
R example – onestep prediction



Blue: lambda = 0.6

Green: lambda = 0.9

R example – onestep prediction

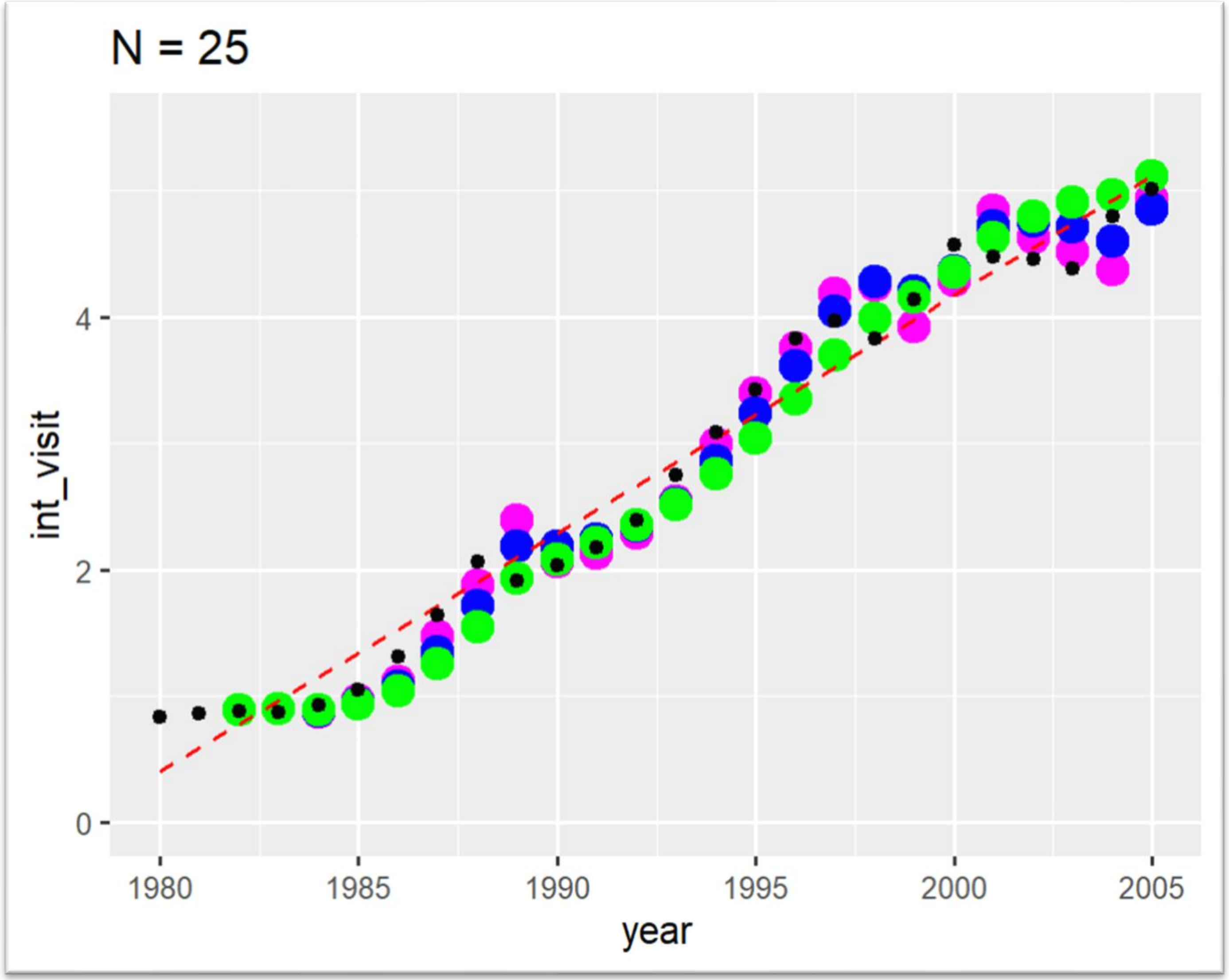


Blue: lambda = 0.6

Green: lambda = 0.9

Pink: lambda = 0.3

R example – onestep prediction



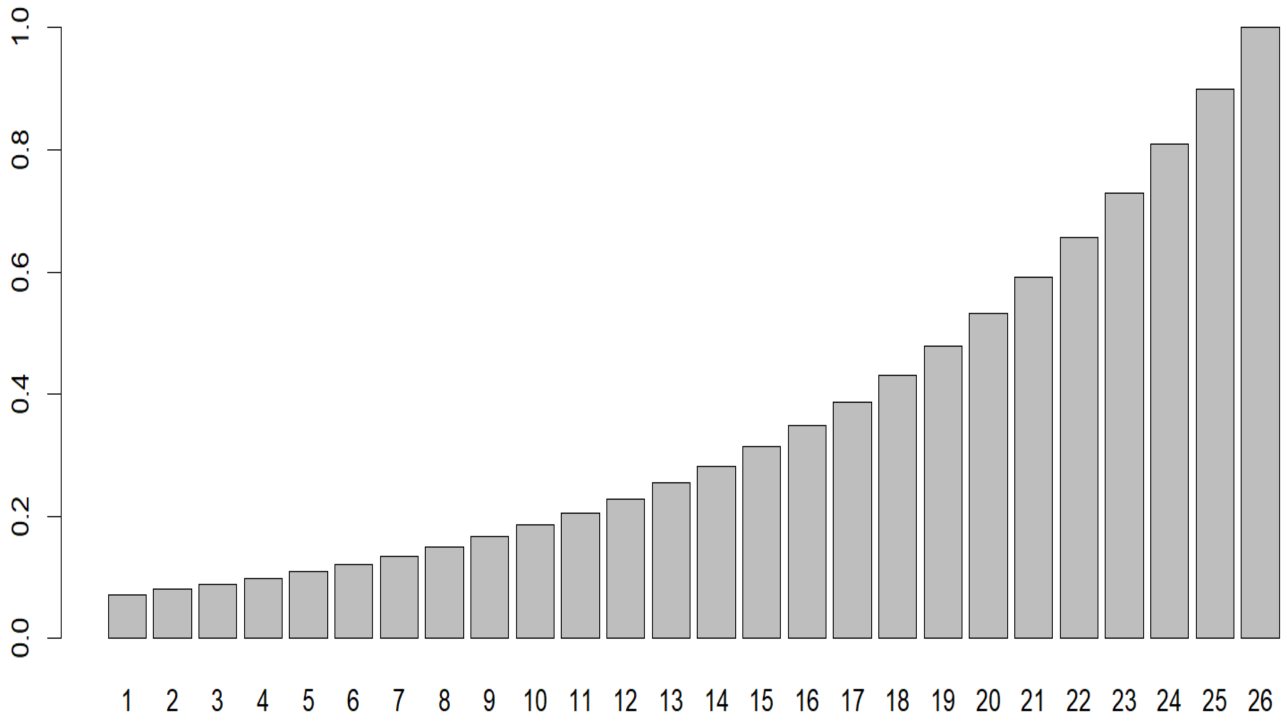
Blue: lambda = 0.6

Green: lambda = 0.9

Pink: lambda = 0.3

What do you think is the general effect of increasing/decreasing lambda?

Choice of λ



$$\Sigma = \begin{bmatrix} 1/\lambda^{n-1} & \dots & 0 & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \dots & 1/\lambda^2 & 0 & 0 \\ 0 & \dots & 0 & 1/\lambda & 0 \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}$$

$$0 < \lambda < 1$$

Larger λ - longer ”memory”

λ = 1 equals OLS

Choice of λ

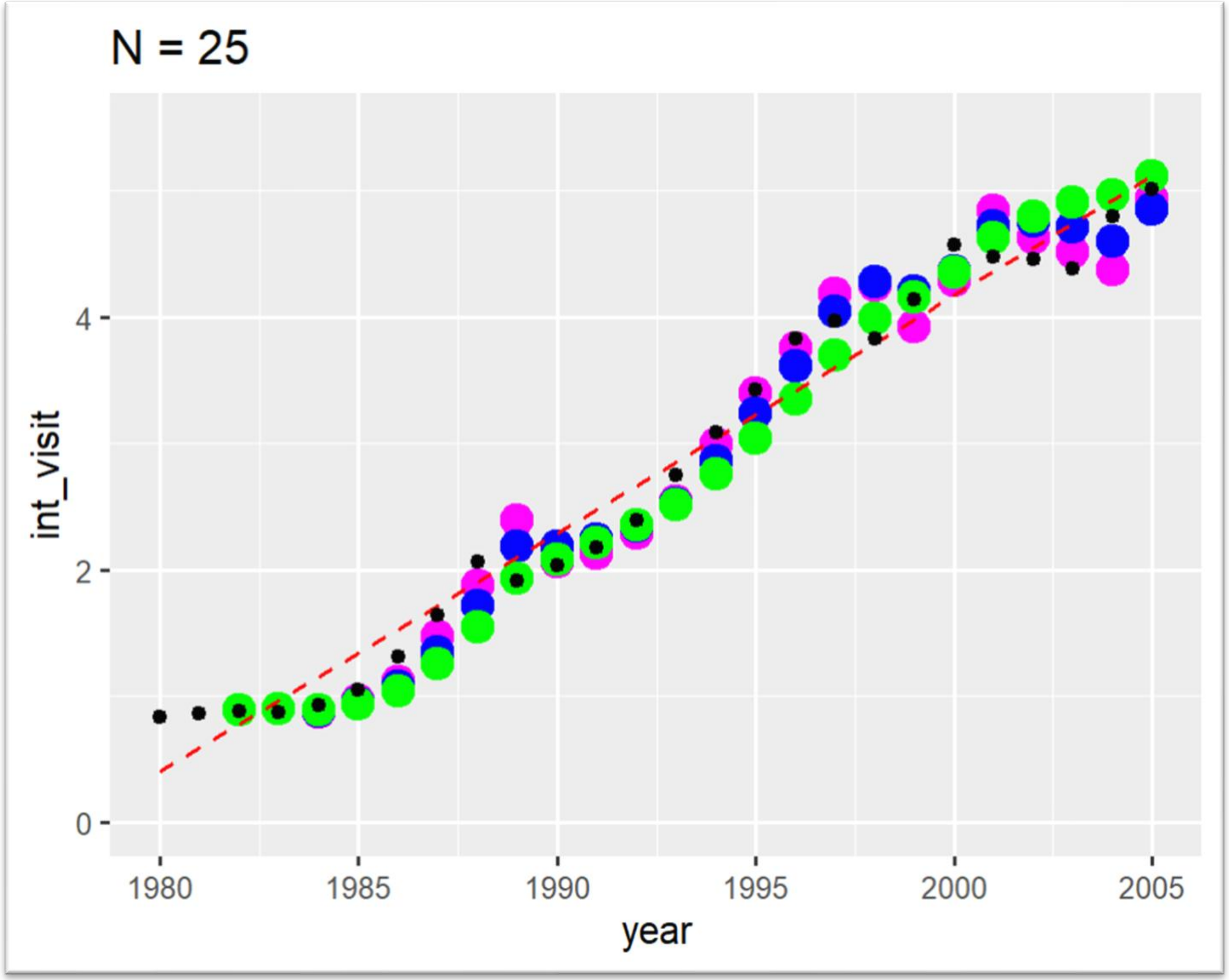
For each step:

- Update model
- Calculate prediction
- Calculate prediction error (when next data is available)

Then calculate the SSE (sum of squared prediction errors)

Repeat for different lambda

Choose optimal lambda (smallest SSE)



Choice of λ

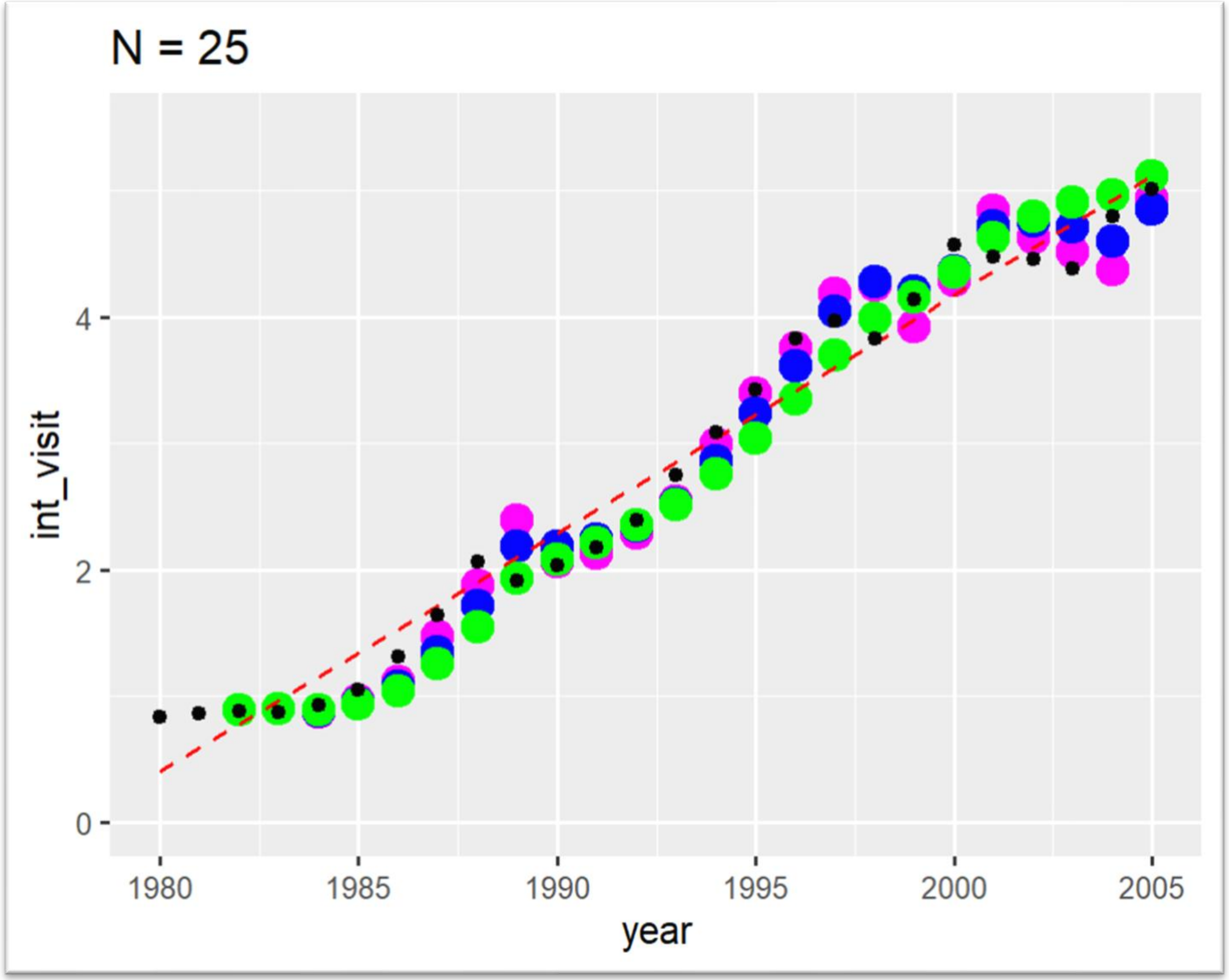
For each step:

- Update model
- Calculate prediction
- Calculate prediction error (when next data is available)

Then calculate the SSE (sum of squared prediction errors)

Repeat for different lambda

Choose optimal lambda (smallest SSE)



But what if we wanted to predict further than one step?

Could we expect the same lambda to be optimal?

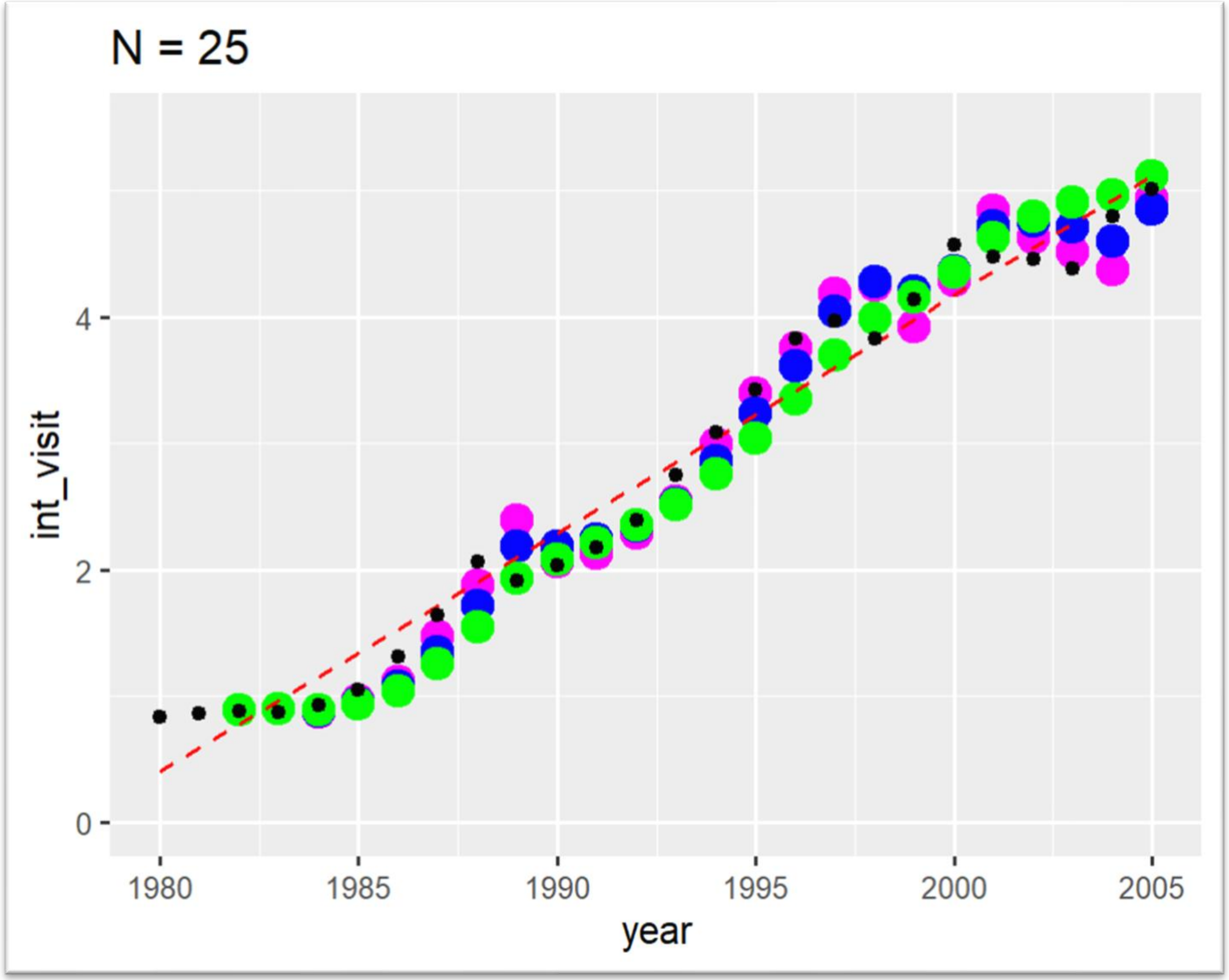
Choice of λ

For each step:
- Update model
- Calculate prediction
- Calculate prediction error (when next data is available)

Then calculate the SSE (sum of squared prediction errors)

Repeat for different lambda

Choose optimal lambda (smallest SSE)



But what if we wanted to predict further than one step?

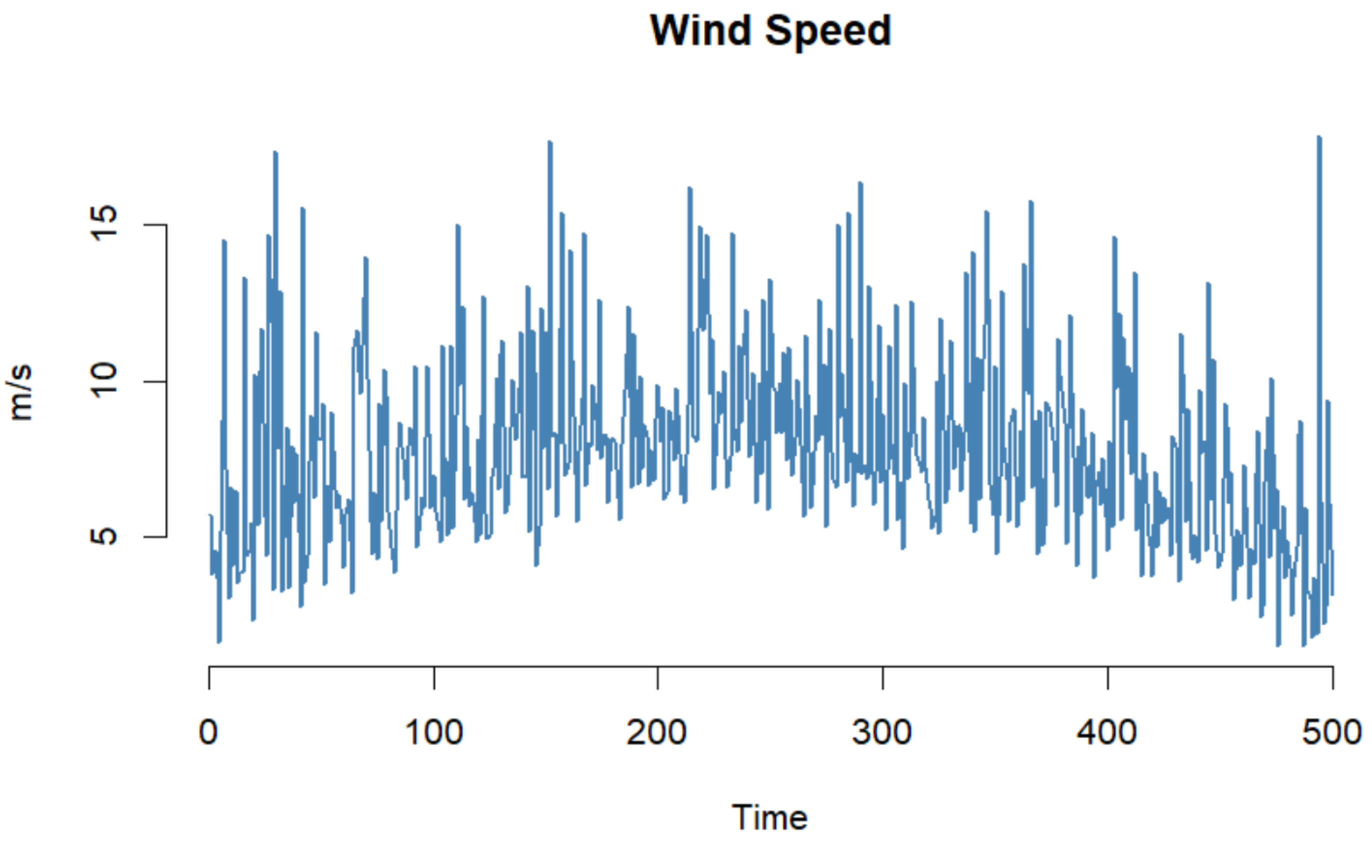
Could we expect the same lambda to be optimal?

Generally larger lambda is better for longer prediction horizons.

Outline of the lecture

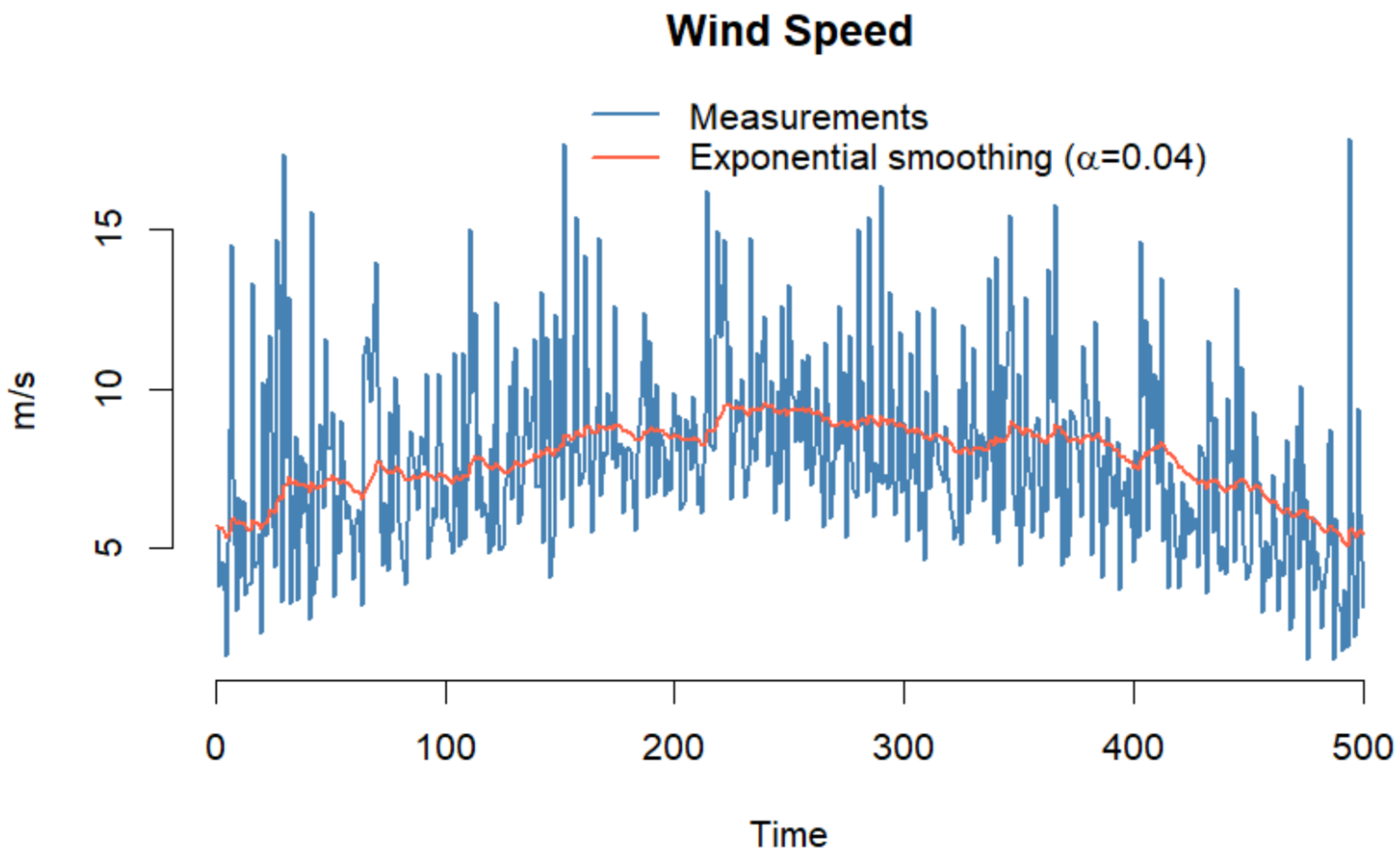
- Recap: Ordinary Least Squares (OLS) and its assumptions
- Weighted Least Squares (WLS)
- Weighted Least Squares for “Local Trend Models”
- Recursive Least Squares with forgetting
- **Exponential smoothing in general**

Smoothing



Let us consider a time series with large fluctuations

Smoothing

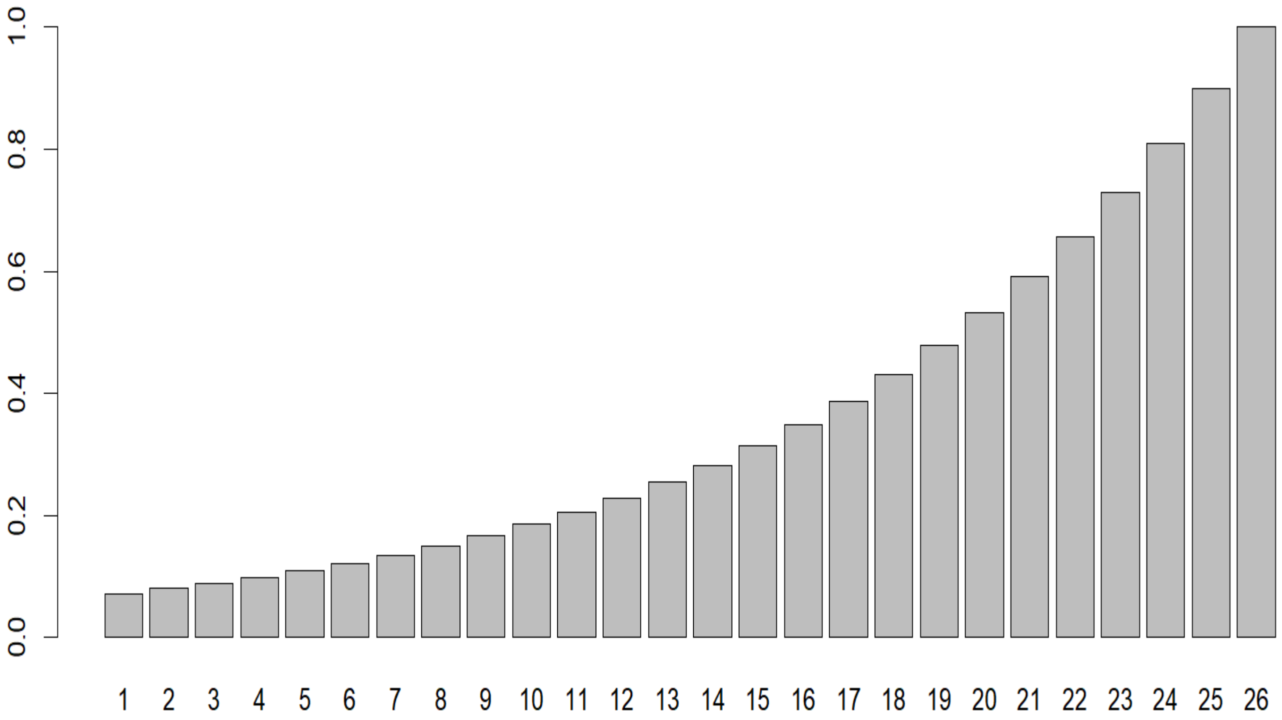


Here is an example of ”*smoothing”* the data

Exponential Smoothing

Given a "memory" factor $\lambda \in]0;1[$

$$\hat{\mu}_N = c \sum_{j=0}^{N-1} \lambda^j Y_{N-j} = c[Y_N + \lambda Y_{N-1} + \cdots + \lambda^{N-1} Y_1]$$



We calculate a weighted average

The weigths decay **exponentially**

Corresponding to estimating **only intercept** (no slope) with WLS and

$$\Sigma = \begin{bmatrix} 1/\lambda^{n-1} & \dots & 0 & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \dots & 1/\lambda^2 & 0 & 0 \\ 0 & \dots & 0 & 1/\lambda & 0 \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}$$

Exponential Smoothing

Given a ”memory” factor $\lambda \in]0;1[$

$$\hat{\mu}_N = c \sum_{j=0}^{N-1} \lambda^j Y_{N-j} = c[Y_N + \lambda Y_{N-1} + \cdots + \lambda^{N-1} Y_1]$$

The constant c is chosen so that the weights sum to one, which implies that $c = (1-\lambda)/(1-\lambda^N)$.

Exponential Smoothing

Given a ”memory” factor $\lambda \in]0;1[$

$$\hat{\mu}_N = c\sum_{j=0}^{N-1} \lambda^j Y_{N-j} = c[Y_N + \lambda Y_{N-1} + \cdots + \lambda^{N-1} Y_1]$$

The constant c is chosen so that the weights sum to one, which implies that $c = (1-\lambda)/(1-\lambda^N)$.
When N is large $c \approx 1-\lambda$:

$$\begin{aligned}
\hat{\mu}_N &= (1-\lambda)[Y_N + \lambda Y_{N-1} + \cdots + \lambda^{N-1} Y_1] \\
&= (1-\lambda) Y_N + (1-\lambda)[\lambda Y_{N-1} + \cdots + \lambda^{N-1} Y_1] \\
&= (1-\lambda) Y_N + \lambda(1-\lambda)[Y_{N-1} + \cdots + \lambda^{N-2} Y_1] \\
&= \boxed{(1-\lambda) Y_N + \lambda \hat{\mu}_{N-1}}
\end{aligned}$$

Recursive formulation

Exponential Smoothing

Given a "memory" factor $\lambda \in]0;1[$

$$\hat{\mu}_N = c\sum_{j=0}^{N-1} \lambda^j Y_{N-j} = c[Y_N + \lambda Y_{N-1} + \cdots + \lambda^{N-1} Y_1]$$

The constant c is chosen so that the weights sum to one, which implies that $c = (1-\lambda)/(1-\lambda^N)$.
When N is large $c \approx 1-\lambda$:

$$\begin{aligned}
\hat{\mu}_N &= (1-\lambda)[Y_N + \lambda Y_{N-1} + \cdots + \lambda^{N-1} Y_1] \\
&= (1-\lambda) Y_N + (1-\lambda)[\lambda Y_{N-1} + \cdots + \lambda^{N-1} Y_1] \\
&= (1-\lambda) Y_N + \lambda(1-\lambda)[Y_{N-1} + \cdots + \lambda^{N-2} Y_1] \\
&= \boxed{(1-\lambda) Y_N + \lambda \hat{\mu}_{N-1}}
\end{aligned}$$

Recursive formulation

Often we specify the forgetting factor, $\alpha = 1-\lambda$ instead.

Simple Exponential Smoothing (SES)

used as a prediction model:

$$\widehat{Y}_{N+\ell|N} = \hat{\mu}_N$$
$$\widehat{Y}_{N+\ell+1|N+1} = (1-\lambda)\, Y_{N+1} + \lambda\, \widehat{Y}_{N+\ell|N}$$

Almost as we did earlier today, but here we only have one parameter –
the ”level” (intercept)

Simple Exponential Smoothing (SES)

used as a prediction model:

$$\widehat{Y}_{N+\ell|N} = \hat{\mu}_N$$
$$\widehat{Y}_{N+\ell+1|N+1} = (1-\lambda)\,Y_{N+1} + \lambda\,\widehat{Y}_{N+\ell|N}$$

Almost as we did earlier today, but here we only have one parameter – the ”level” (intercept)

Given a data set $t = 1, \ldots, N$ we construct

$$S(\alpha) = \sum_{t=1}^{N} (Y_t - \widehat{Y}_{t|t-l}(\alpha))^2 = \sum_{t=1}^{N} (Y_t - \hat{\mu}_{t-l}(\alpha))^2$$

Sum of squared ℓ-step prediction errors

Simple Exponential Smoothing (SES)

used as a prediction model:

$$\widehat{Y}_{N+\ell|N} = \hat{\mu}_N$$
$$\widehat{Y}_{N+\ell+1|N+1} = (1-\lambda)\, Y_{N+1} + \lambda\, \widehat{Y}_{N+\ell|N}$$

Almost as we did earlier today, but here we only have one parameter – the ”level” (intercept)

Given a data set $t = 1, \ldots, N$ we construct

$$S(\alpha) = \sum_{t=1}^{N} (Y_t - \widehat{Y}_{t|t-l}(\alpha))^2 = \sum_{t=1}^{N} (Y_t - \hat{\mu}_{t-l}(\alpha))^2$$

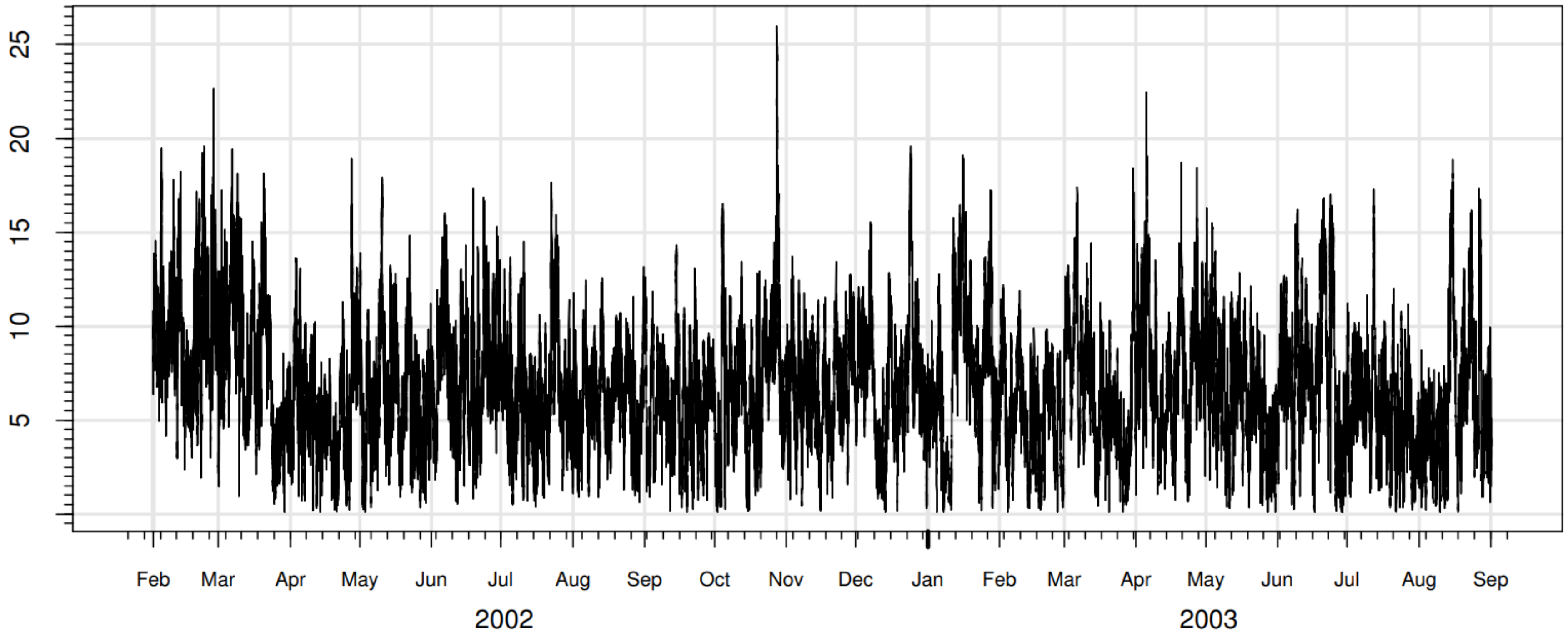
Sum of squared
ℓ-step prediction errors
is used to find optimal lambda
(or optimal alpha)

The value minimizing $S(\alpha)$ is chosen.

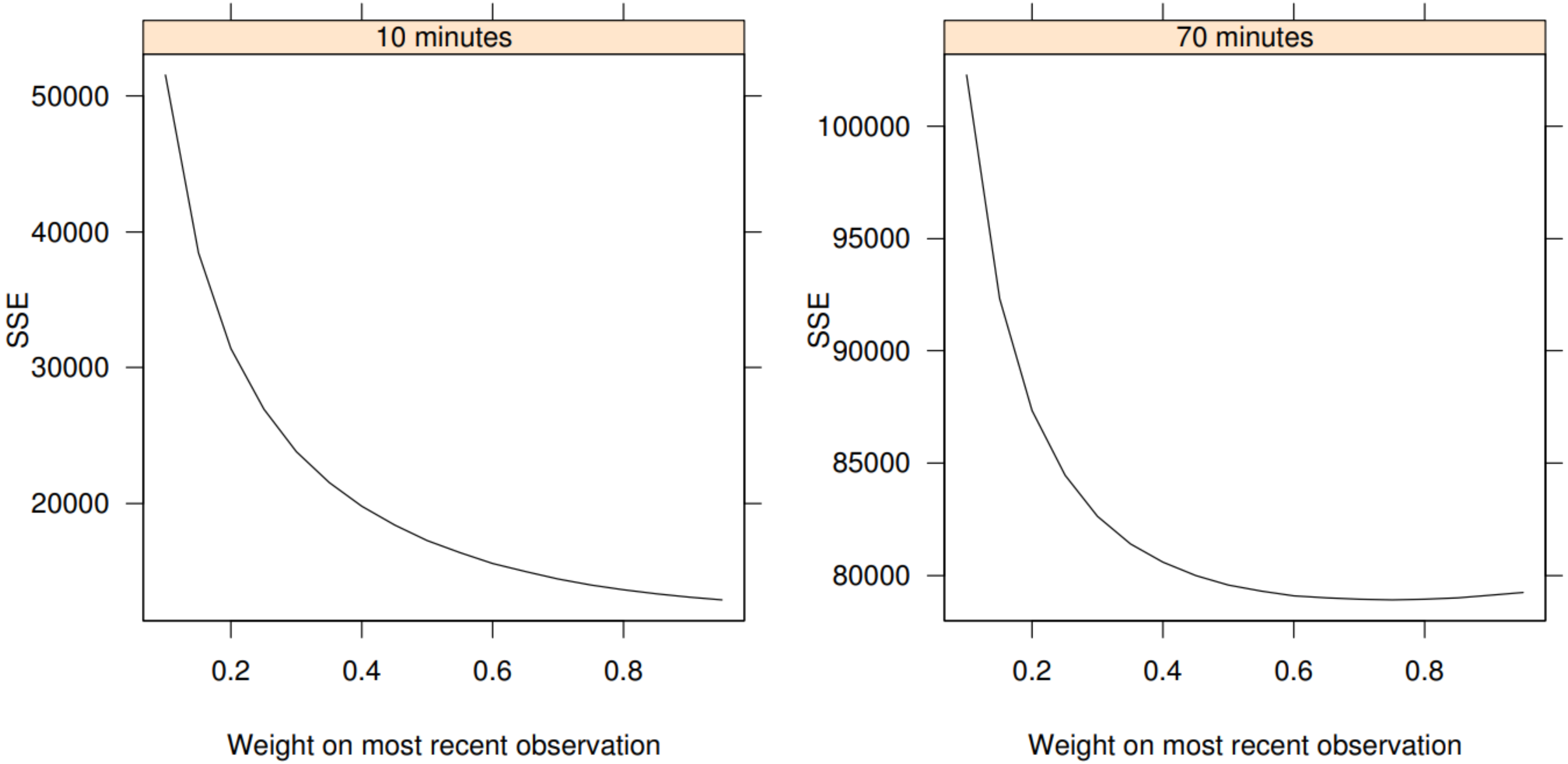
$$\alpha = 1 - \lambda$$

Example – wind speed 76m a.g.l. at Risø

- Measurements of wind speed every 10th minute
- Task: Forecast up to approximately 3 hours ahead using exponential smoothing



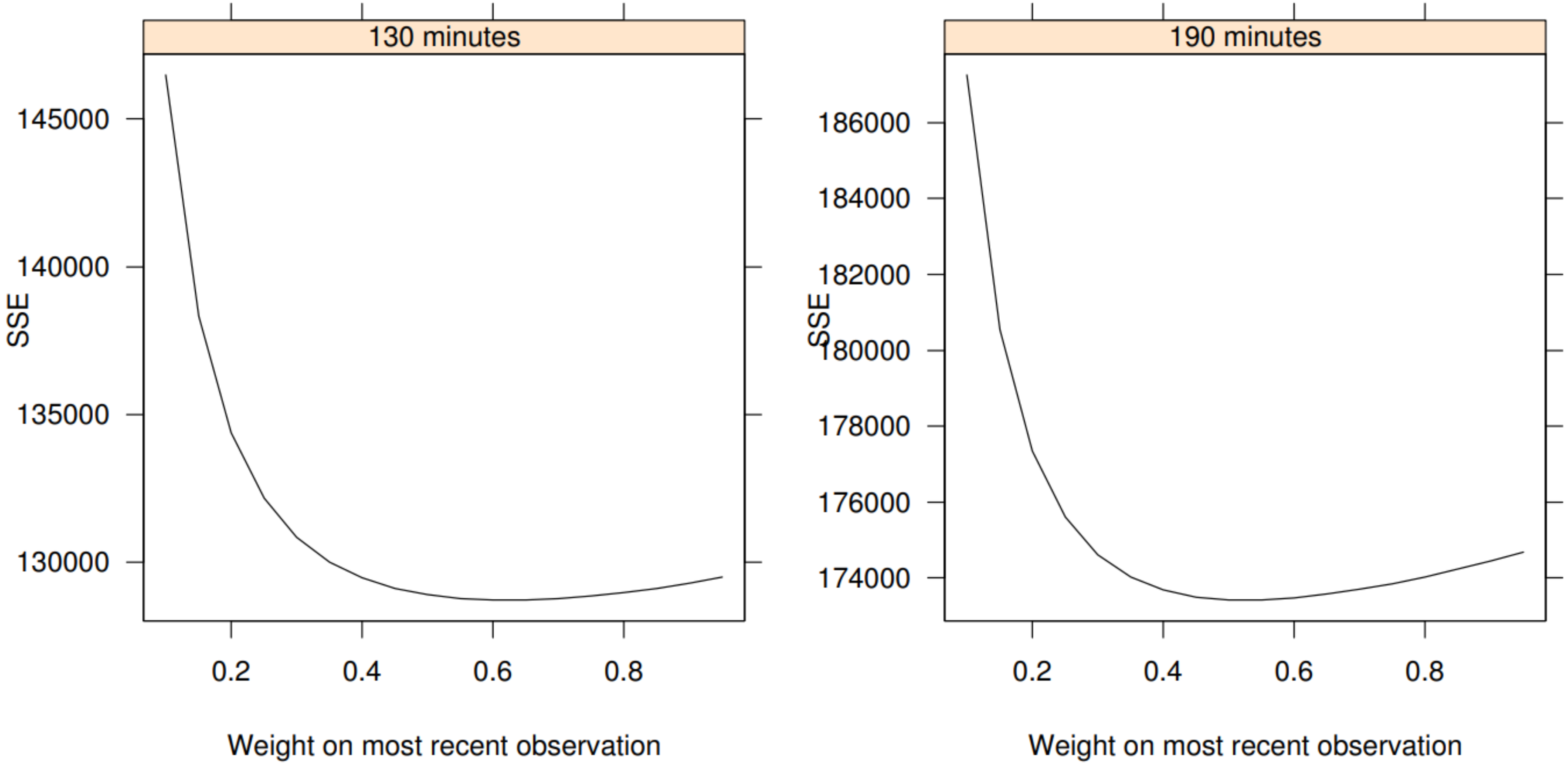
Example – wind speed 76m a.g.l. at Risø



- 10 minutes (1-step): Use $\alpha = 0.95$ or higher
- 70 minutes (7-step): Use $\alpha \approx 0.7$

$$\alpha = 1 - \lambda$$

Example – wind speed 76m a.g.l. at Risø



- 130 minutes (13-step): Use $\alpha \approx 0.6$
- 190 minutes (19-step): Use $\alpha \approx 0.5$

$$\alpha = 1 - \lambda$$

Simple, double and triple exponential smoothing

- Simple Exponential smoothing
  - The model includes a **level** / constant mean (intercept)
  - All future predictions have the same value (constant)
  - The predicted level is an exponentially weighted sum of past observations

Simple, double and triple exponential smoothing

- Simple Exponential smoothing
  - The model includes a **level** / constant mean (intercept)
  - All future predictions have the same value (constant)
  - The predicted level is an exponentially weighted sum of past observations

- Holt's Linear Trend model (= ”double exponential smoothing”)
  - Includes both **level** (intercept at time = N) and **trend** (slope)
  - Both the level and trend have *individual* smoothing parameters (individual lambda's) (this is different from the local model we have made – here we used same lambda for both parameters)
  - In damped trend models a damping is included to the

Simple, double and triple exponential smoothing

- Simple Exponential smoothing
  - The model includes a **level** / constant mean (intercept)
  - All future predictions have the same value (constant)
  - The predicted level is an exponentially weighted sum of past observations

- Holt's Linear Trend model (= ”double exponential smoothing”)
  - Includes both **level** (intercept at time = N) and **trend** (slope)
  - Both the level and trend have *individual* smoothing parameters (individual lambda's) (this is different from the local model we have made – here we used same lambda for both parameters)
  - In damped trend models a damping is included to the

- Holt-Winters' model (= ”triple exponential smoothing”)
  - Includes **level** and **trend** and **seasonal** component
  - 3 *individual* smoothing parameters

Simple, double and triple exponential smoothing

- Simple Exponential smoothing
  - The model includes a **level** / constant mean (intercept)
  - All future predictions have the same value (constant)
  - The predicted level is an exponentially weighted sum of past observations

- Holt’s Linear Trend model (= ”double exponential smoothing”)
  - Includes both **level** (intercept at time = N) and **trend** (slope)
  - Both the level and trend have *individual* smoothing parameters (individual lambda’s) (this is different from the local model we have made – here we used same lambda for both parameters)
  - In damped trend models a damping is included to the

- Holt-Winters’ model (= ”triple exponential smoothing”)
  - Includes **level** and **trend** and **seasonal** component
  - 3 *individual* smoothing parameters

- ”ETS models” in general = Error-Trend-Season
  - Both additive and multiplicative forms

Questions ?

Next time:

- Stochastic Processes w. Peder ☺