Time Series Analysis

Spring 2024

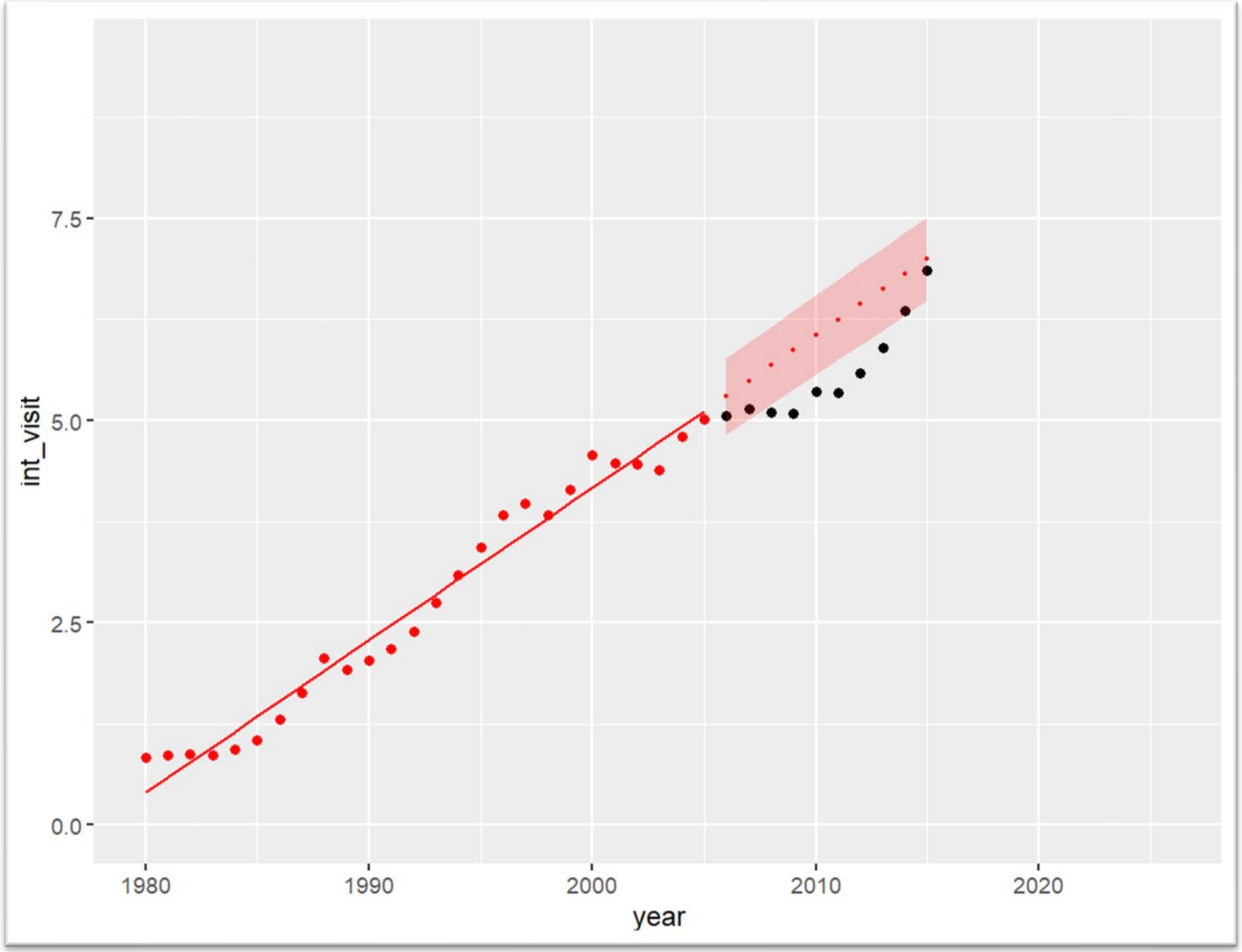
Peder Bacher and Pernille Yde Nielsen

February 16, 2024

Outline of the lecture

- Regression based methods, 2^{nd} part:
  - From Global models to Local models
  - WLS
  - Local Trend Model
  - Exponential smoothing in general

Global model from last week



Model from last week:

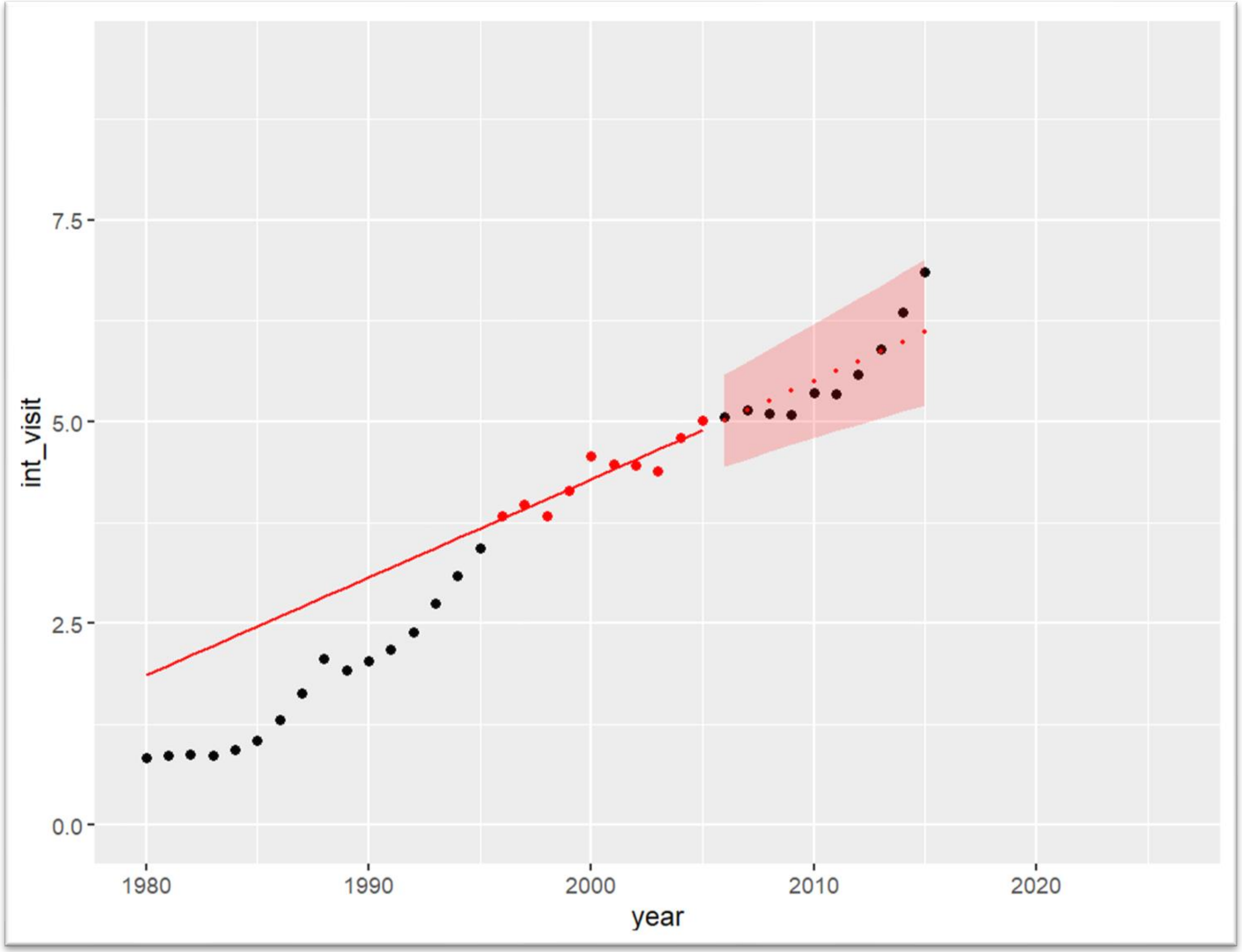
The ”training data” consists of 26 observations (1980 to 2005)

We predict values for the next 10 observations (2006 to 2015)

Outline of the lecture

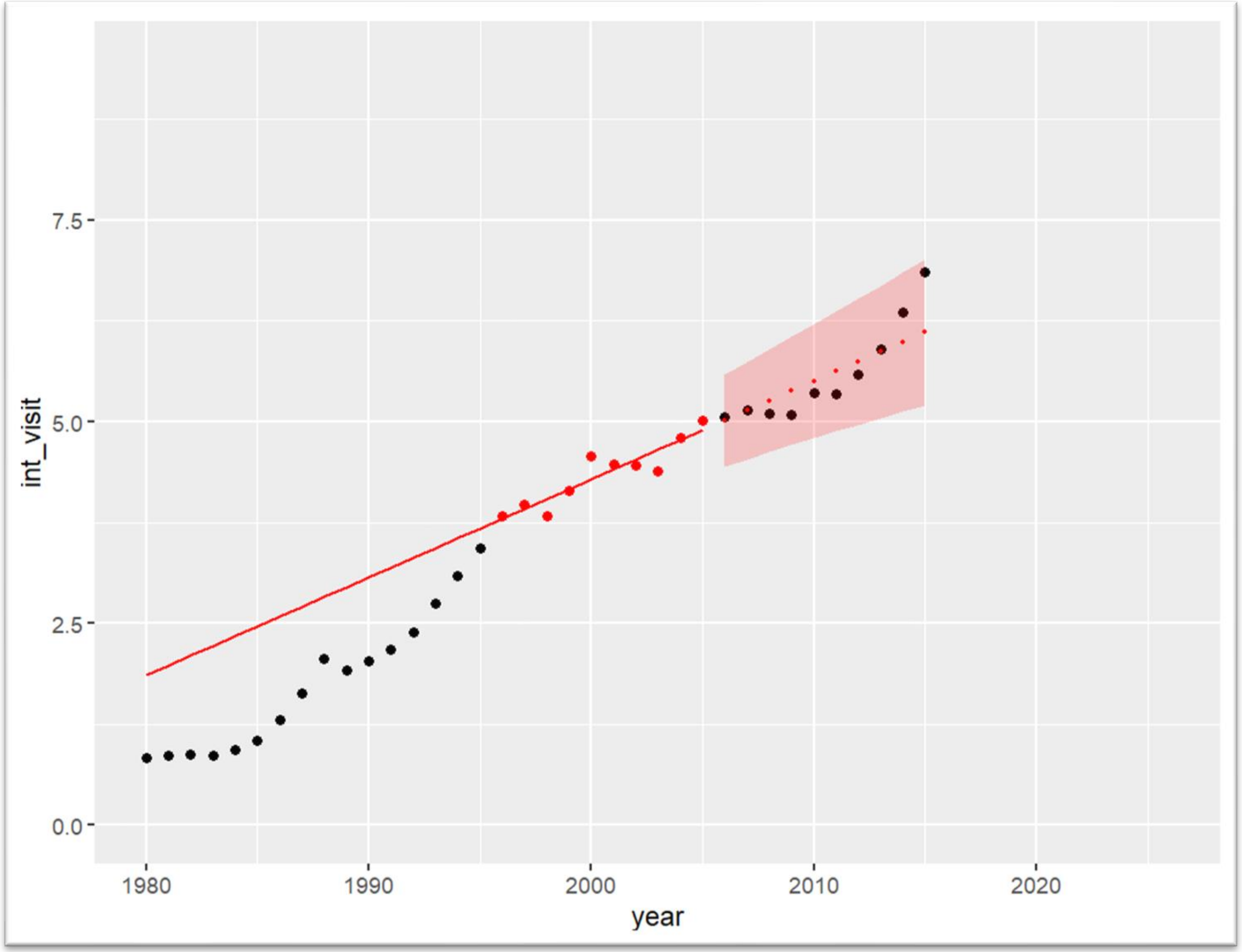
- Regression based methods, 2nd part:
  - **From Global models to Local models**
  - WLS
  - Local Trend Model
  - Exponential smoothing in general

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

From Global to Local models

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

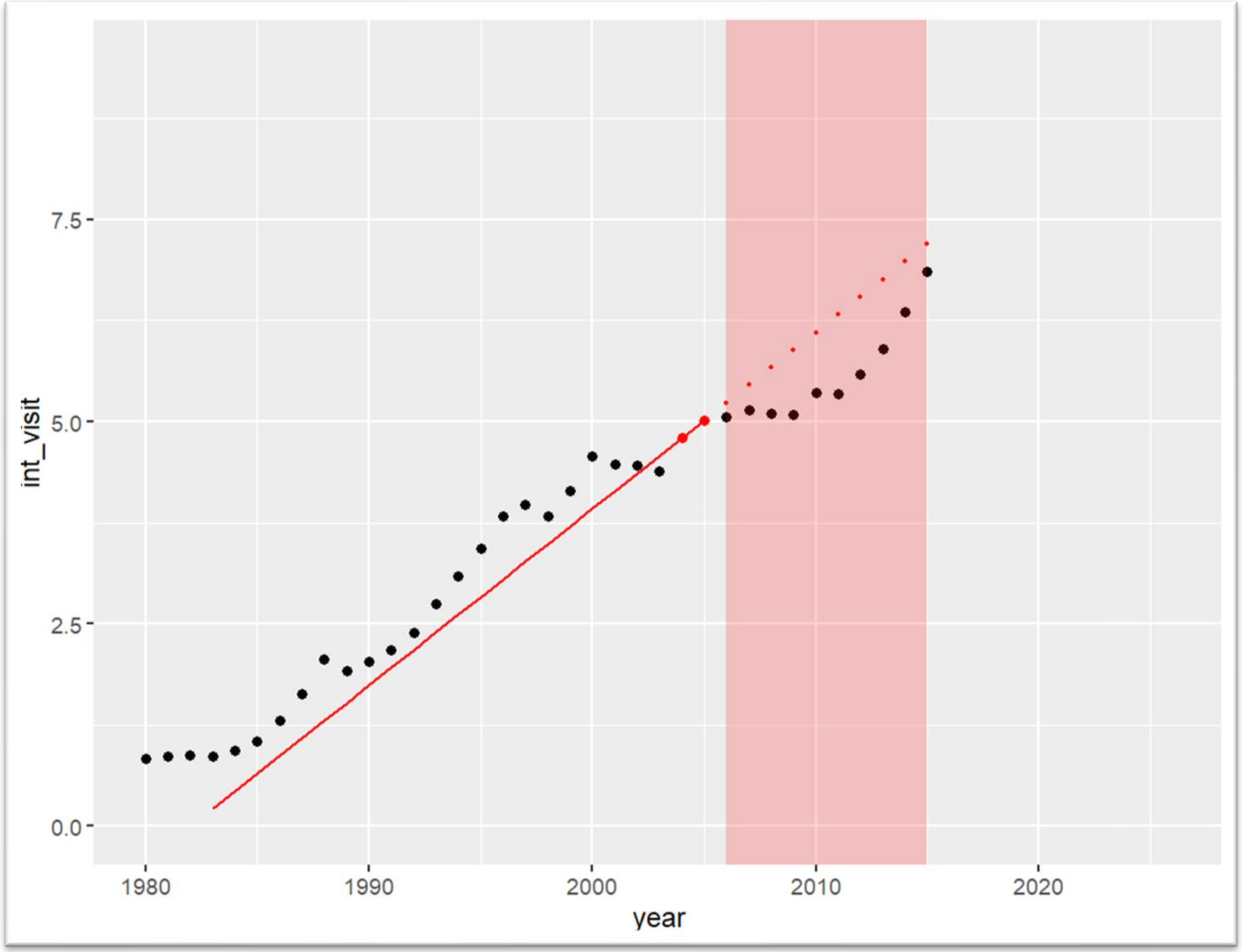
From Global to Local models

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”

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

N = number of observations in data

p = number of parameters estimated

From Global to Local models

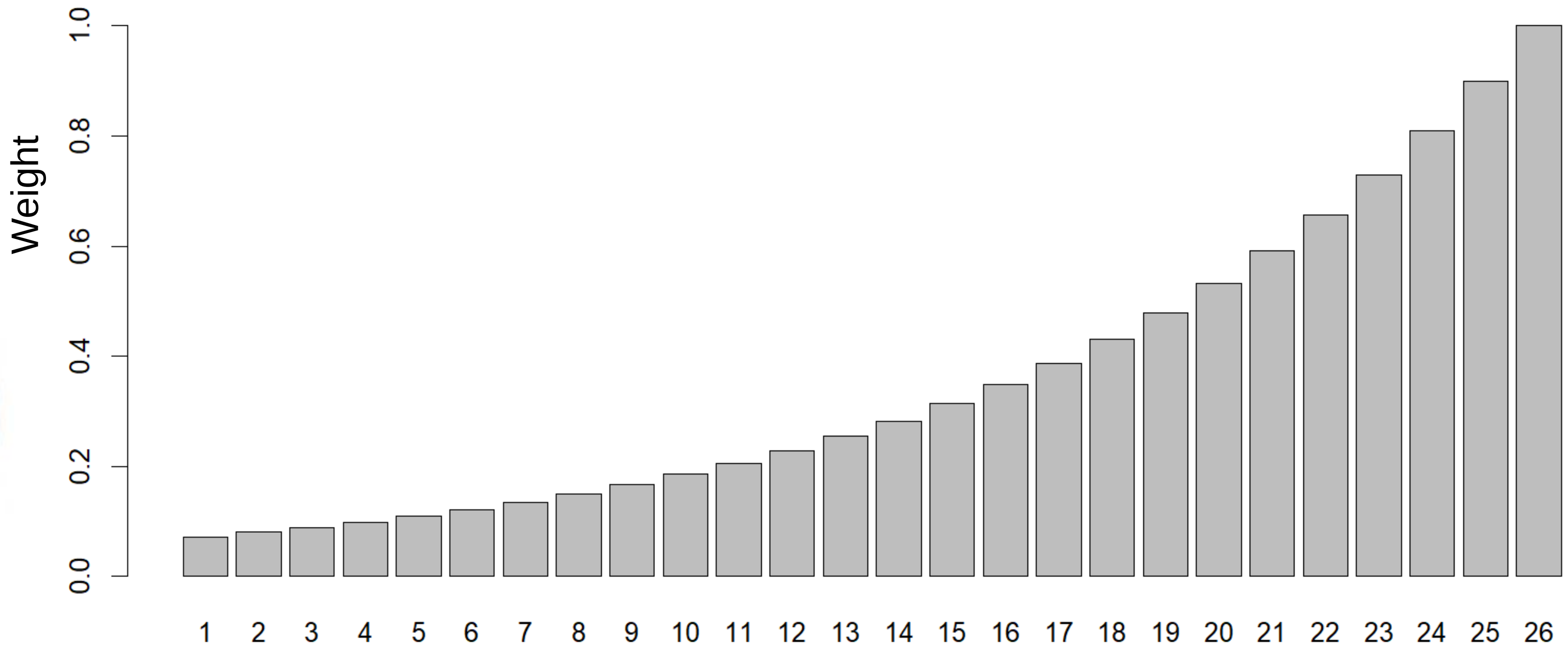
Questions:

How do we choose how many observations to use?

Is it fair to make a sharp ”cut-off” like that?
 - could we make a more *soft* cut-off?

From Global to Local models

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



Outline of the lecture

- Regression based methods, 2nd part:
  - From Global models to Local models
  - **WLS**
  - Local Trend Model
  - Exponential smoothing in general

Weighted Least Squares (WLS)

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

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

We minimize the *weighted* sum of squared residuals:

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

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:

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

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

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

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:

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

Solution:

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

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

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

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

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

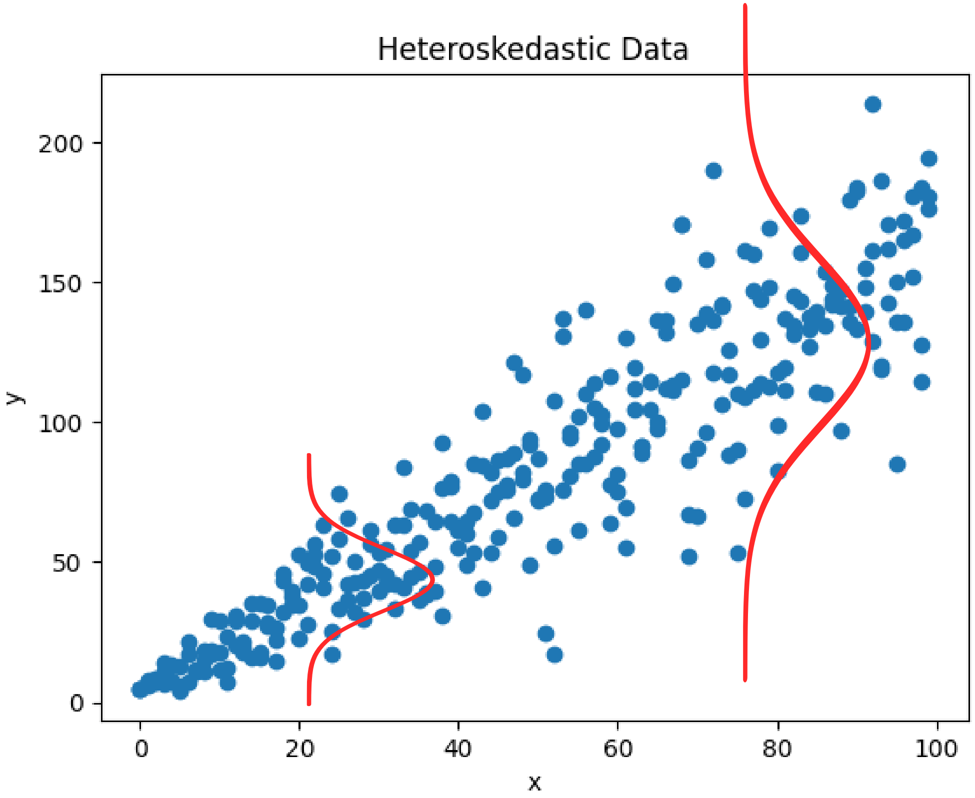
And the off-diagonal elements have to do with covariance between two observations

WLS with diagonal matrix

Today we consider the case where only variances are different
(no covariances = off-diagonal elements are zero)

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

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



Weights in WLS

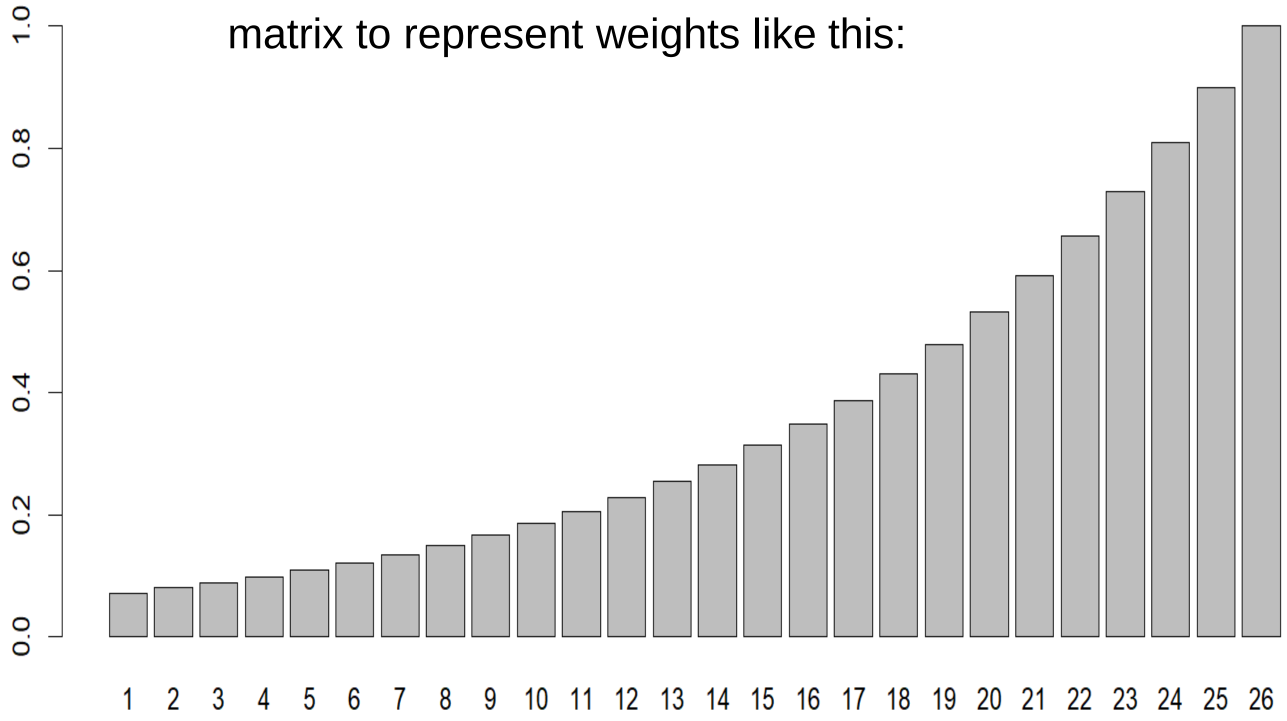
Recall: Large variance = Low weight

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

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

Choice of weights for a "local model"

Now we want the variance-covariance matrix to represent weights like this:



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

$$0 < \lambda < 1$$

Example in R

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

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

```
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
```

Example in R

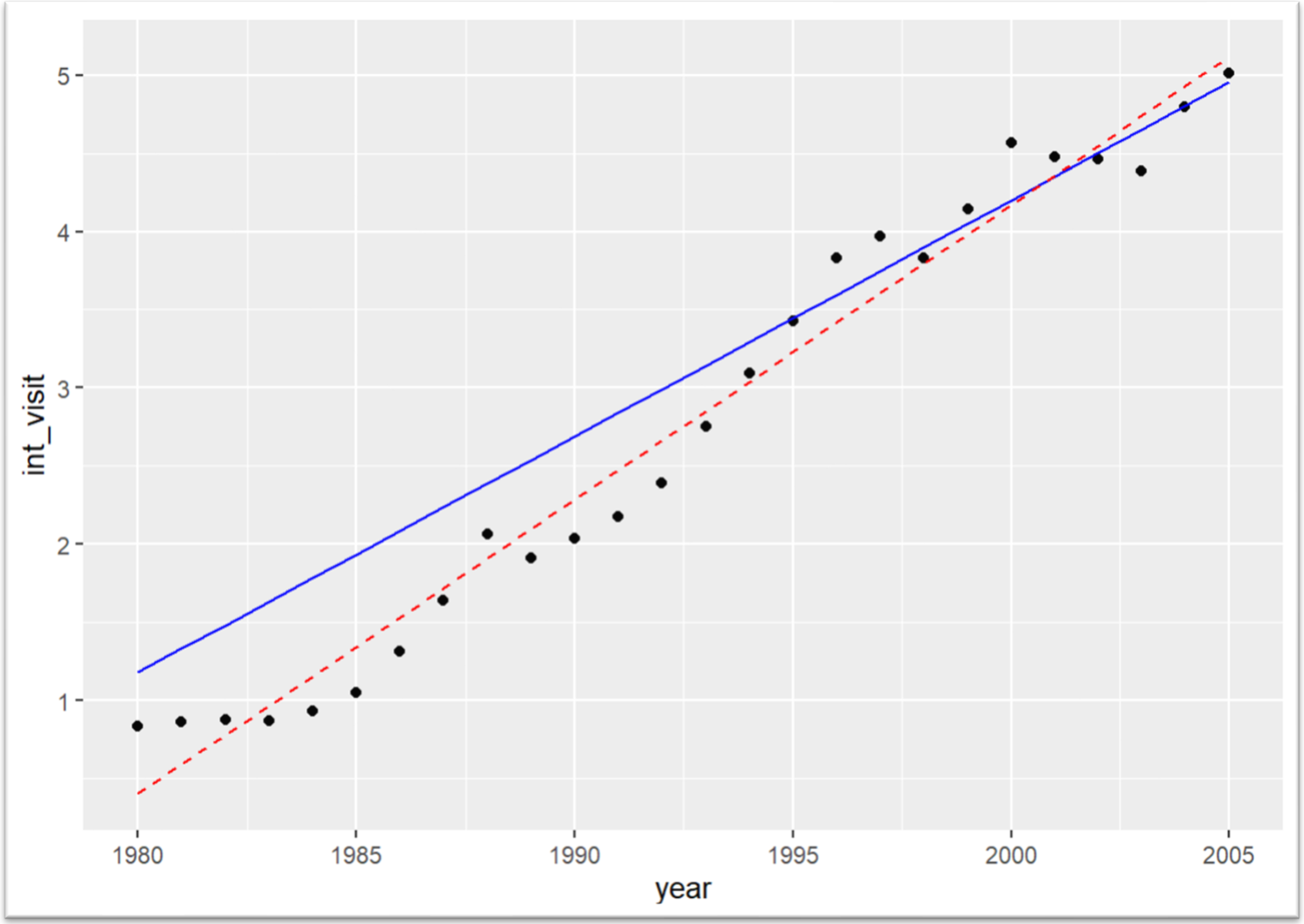
$$\widehat{\boldsymbol{\theta}} = (\boldsymbol{x}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{x})^{-1} \boldsymbol{x}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{Y}$$

```
WLS <- solve(t(X[1:26,])%*%solve(SIGMA)%*%X[1:26,])%*%(t(X[1:26,])%*%solve(SIGMA)%*%y[1:26])
```

$$\hat{Y} = x\hat{\theta}$$

```
yhat_wls <- X[1:26,]%*%WLS
```

Example in R

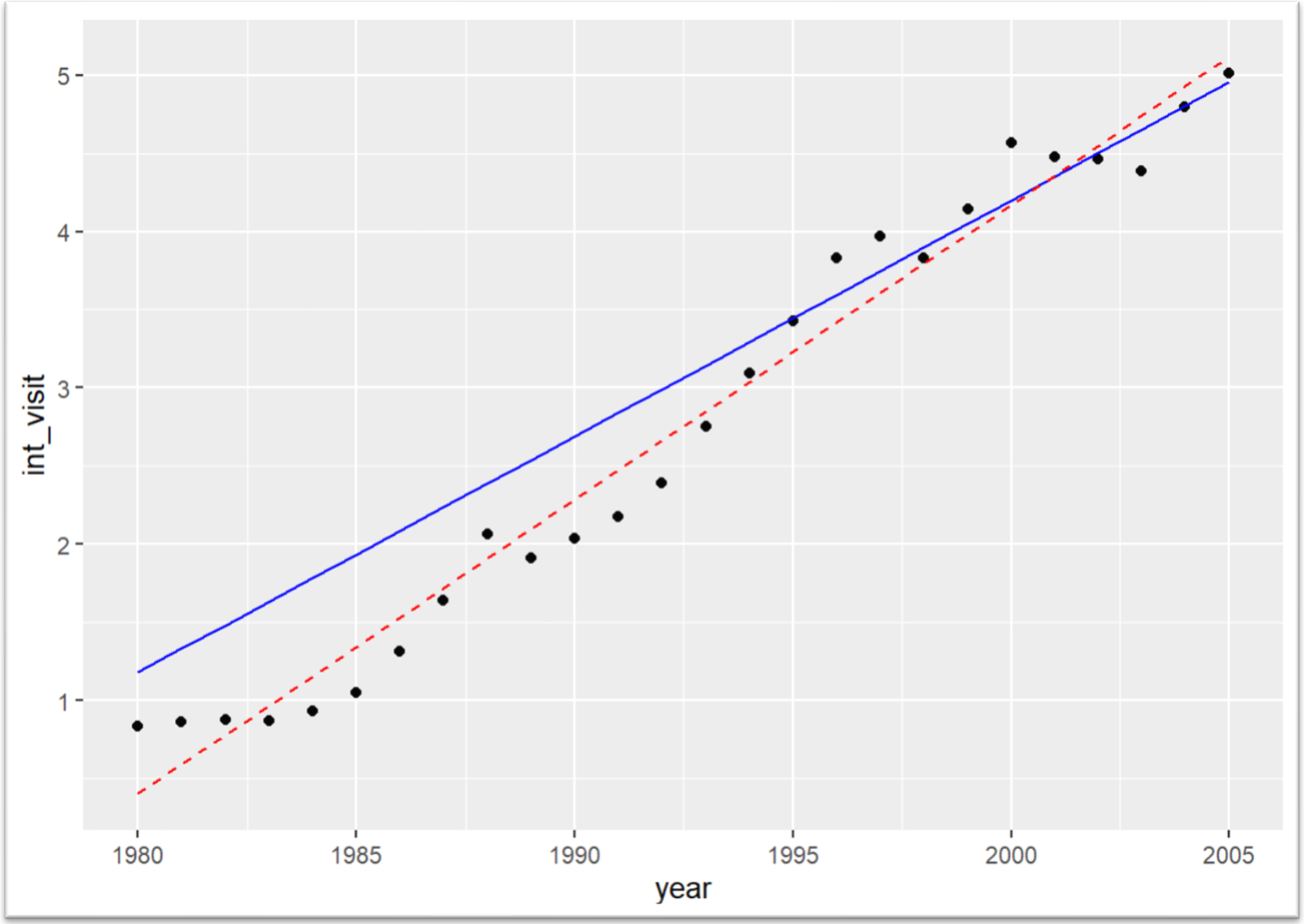


Blue is WLS

Red is OLS from last week

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

Example in R



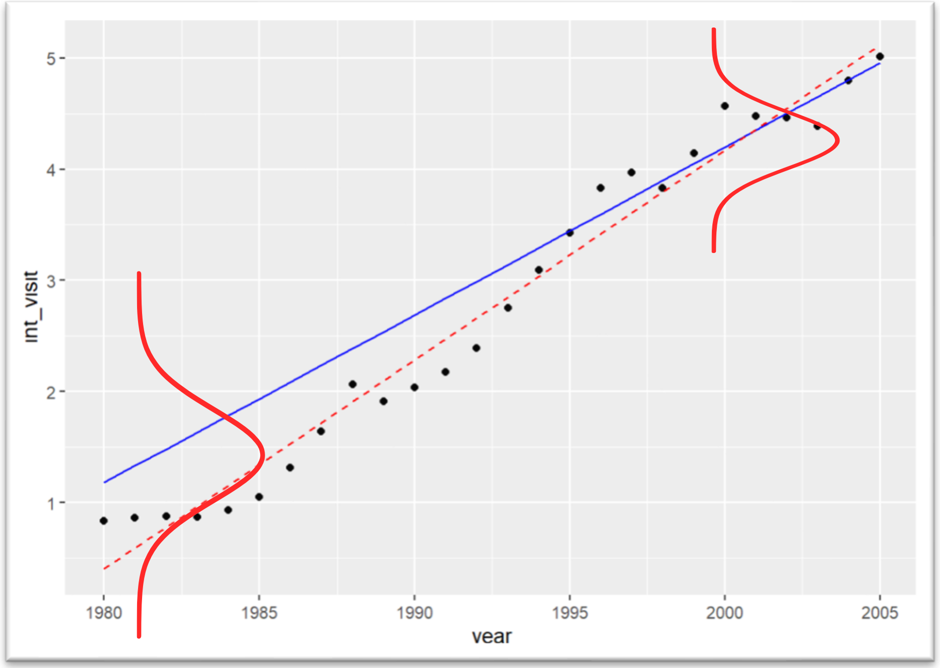
Blue is WLS

Red is OLS from last week

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

What about predictions and prediction intervals?!

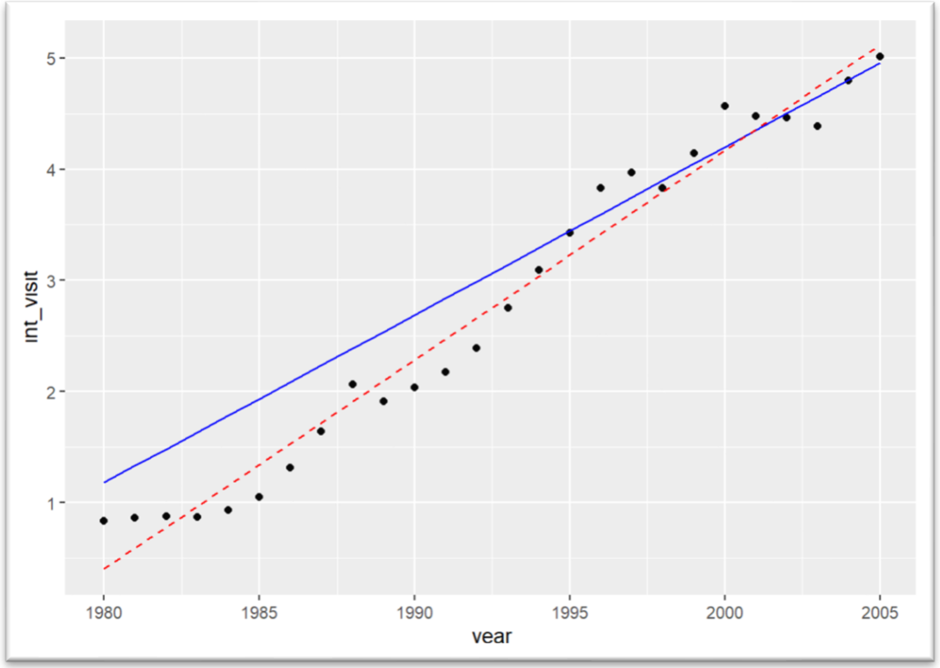
Predictions with WLS



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

What should we assume about the variance of future observations?

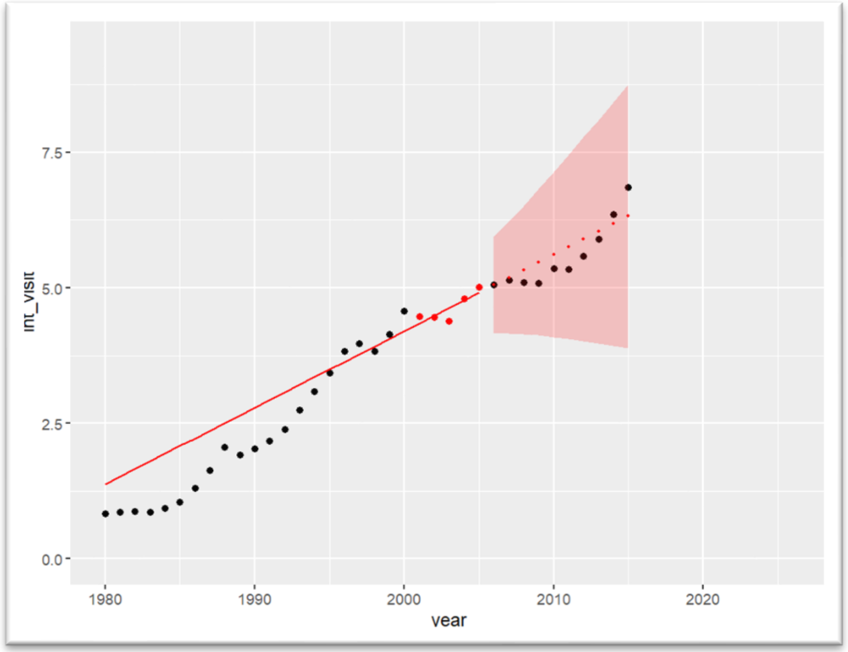
Predictions with WLS



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

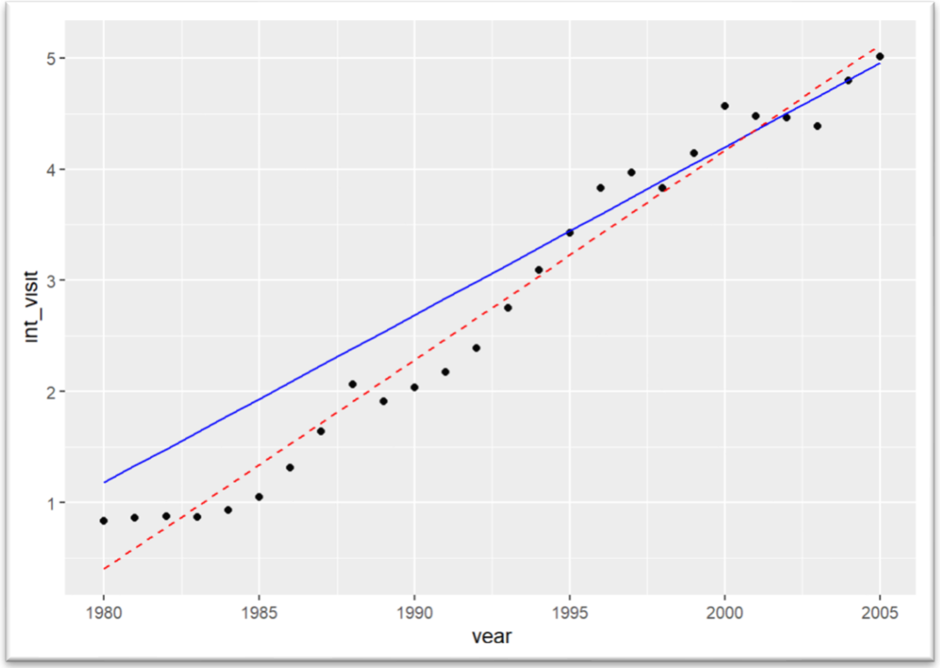
What should we assume about the variance of future observations?

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



So is it fair to use n = N (n = 26 in this example), when the observations are weighted?

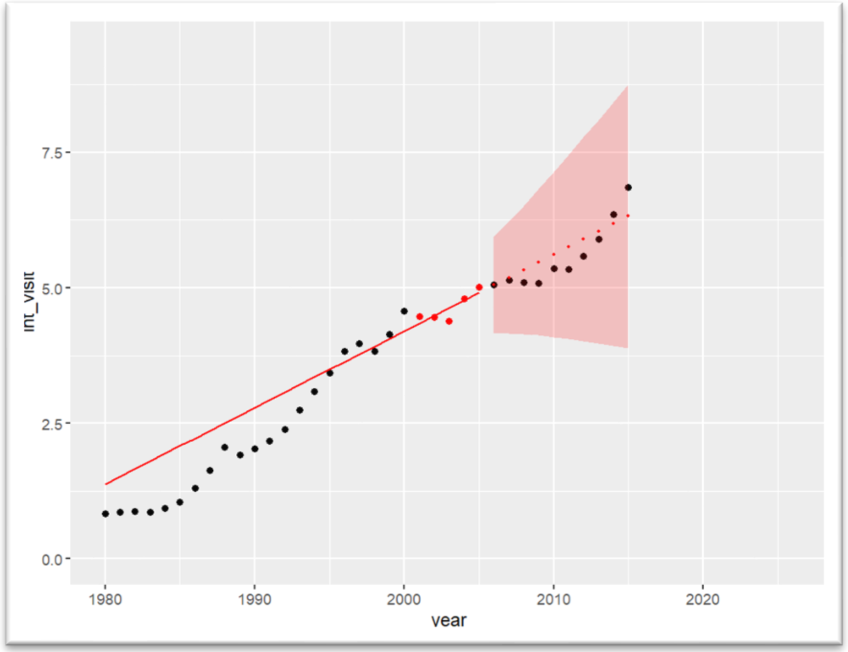
Prediction with WLS



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

What should we assume about the variance of future observations?

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



So is it fair to use n = N (n = 26 in this example), when the observations are weighted?

We will get back to this issue later

Outline of the lecture

- Regression based methods, 2nd part:
  - From Global models to Local models
  - WLS
  - **Local Trend Model**
  - Exponential smoothing in general

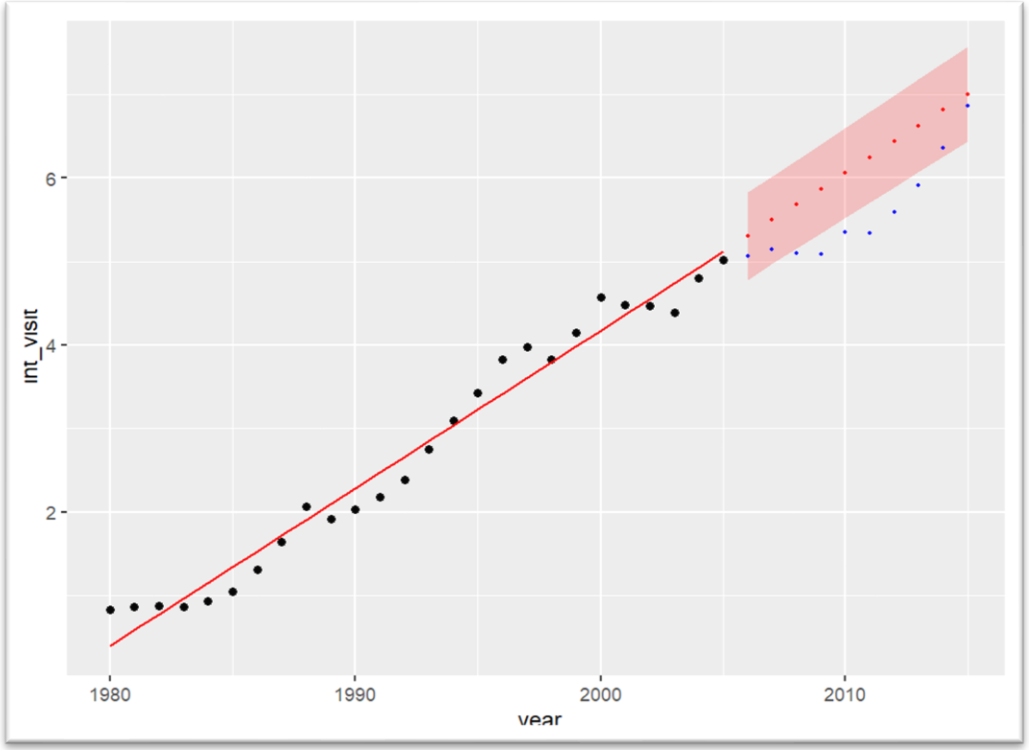
Trend Models – change of notation and reference point

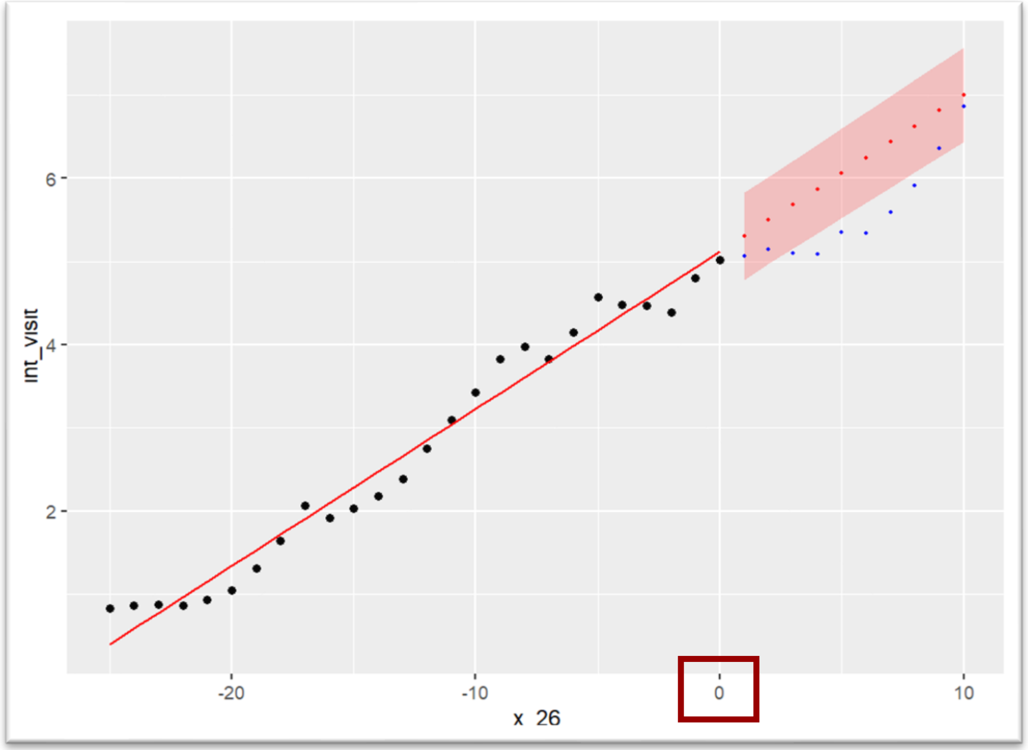
The General Linear Model

$$Y_t = \boldsymbol{x}_t^T \boldsymbol{\theta} + \varepsilon_t$$

The Trend Model:

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





Trend Models recap

Trend models are:

- Linear regression models
- The independent variables are functions of time
- The reference time is often the latest timepoint instead of the ”origin”
- Notation is a bit different – the principle is the same

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

N refers to the ”latest” timepoint in the data – this is our reference timepoint (=”now”)

If we put j=0, we get the estimate ”now”

j = 1,2,3,… refers to future timepoints

The data available to estimate the model is the data from j = 0, -1, -2, -3, …

Trend Model Notation

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

$$\begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_N \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}^T(-N+1) \\ \boldsymbol{f}^T(-N+2) \\ \vdots \\ \boldsymbol{f}^T(0) \end{bmatrix} \boldsymbol{\theta}_N + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_N \end{bmatrix}$$

Global Trend model, parameter estimates

$$\begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_N \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}^T(-N+1) \\ \boldsymbol{f}^T(-N+2) \\ \vdots \\ \boldsymbol{f}^T(0) \end{bmatrix} \boldsymbol{\theta}_N + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_N \end{bmatrix}$$

OLS solution to the GLOBAL trend model:

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

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

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

Global Trend model, parameter estimates

$$\begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_N \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}^T(-N+1) \\ \boldsymbol{f}^T(-N+2) \\ \vdots \\ \boldsymbol{f}^T(0) \end{bmatrix} \boldsymbol{\theta}_N + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_N \end{bmatrix}$$

OLS solution to the GLOBAL trend model:

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

The solution is found by minimizing the sum of squared residuals ("Least Squares")

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

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

Global Trend model, parameter estimates

$$\begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_N \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}^T(-N+1) \\ \boldsymbol{f}^T(-N+2) \\ \vdots \\ \boldsymbol{f}^T(0) \end{bmatrix} \boldsymbol{\theta}_N + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_N \end{bmatrix}$$

OLS solution to the GLOBAL trend model:

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

The solution is found by minimizing the sum of squared residuals (”Least Squares”)

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

We can change this to minimizing the **weighted** sum of squared residuals (WLS) making a **Local Trend model**

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

Local Trend model with WLS

The criterion:

$$S(\boldsymbol{\theta}; N) = \sum_{j=0}^{N-1} \lambda^j [Y_{N-j} - \boldsymbol{f}^T(-j)\boldsymbol{\theta}]^2$$

can be written as:

$$\begin{bmatrix} Y_1 - \boldsymbol{f}^T(N-1)\boldsymbol{\theta} \\ Y_2 - \boldsymbol{f}^T(N-2)\boldsymbol{\theta} \\ \vdots \\ Y_N - \boldsymbol{f}^T(0)\boldsymbol{\theta} \end{bmatrix}^T \begin{bmatrix} \lambda^{N-1} & 0 & \cdots & 0 \\ 0 & \lambda^{N-2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Y_1 - \boldsymbol{f}^T(N-1)\boldsymbol{\theta} \\ Y_2 - \boldsymbol{f}^T(N-2)\boldsymbol{\theta} \\ \vdots \\ Y_N - \boldsymbol{f}^T(0)\boldsymbol{\theta} \end{bmatrix}$$

which is a WLS criterion with $\boldsymbol{\Sigma} = \text{diag}[1/\lambda^{N-1}, \ldots, 1/\lambda, 1]$

Local Trend model, parameter estimates

$$\begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_N \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}^T(-N+1) \\ \boldsymbol{f}^T(-N+2) \\ \vdots \\ \boldsymbol{f}^T(0) \end{bmatrix} \boldsymbol{\theta}_N + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_N \end{bmatrix}$$

WLS solution to the LOCAL trend model:

$$\begin{aligned} \widehat{\boldsymbol{\theta}}_N &= \boldsymbol{F}_N^{-1} \boldsymbol{h}_N \\ \boldsymbol{F}_N &= \sum_{j=0}^{N-1} \underline{\lambda^j} \boldsymbol{f}(-j) \boldsymbol{f}^T(-j) \\ \boldsymbol{h}_N &= \sum_{j=0}^{N-1} \underline{\lambda^j} \boldsymbol{f}(-j) Y_{N-j} \end{aligned}$$

Local Trend model, iterative updates

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

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

$$\boldsymbol{F}_{N+1} = \boldsymbol{F}_N + \lambda^N \boldsymbol{f}(-N)\boldsymbol{f}^T(-N)$$

$$\boldsymbol{h}_{N+1} = \lambda \boldsymbol{L}^{-1}\boldsymbol{h}_N + \boldsymbol{f}(0)\, Y_{N+1}$$

$$\widehat{\boldsymbol{\theta}}_{N+1} = \boldsymbol{F}_{N+1}^{-1}\boldsymbol{h}_{N+1}$$

1) Calculate F_1 and h_1
2) Use update equations to calculate new values of F and h
3) Use F and h to calculate parameter estimates
4) Maybe disregard first few parameter estimates as transient

Local Trend model, iterative updates

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

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

$$\boldsymbol{F}_{N+1} = \boldsymbol{F}_N + \lambda^N \boldsymbol{f}(-N)\boldsymbol{f}^T(-N)$$

$$\boldsymbol{h}_{N+1} = \lambda \boldsymbol{L}^{-1} \boldsymbol{h}_N + \boldsymbol{f}(0) Y_{N+1}$$

$$\widehat{\boldsymbol{\theta}}_{N+1} = \boldsymbol{F}_{N+1}^{-1} \boldsymbol{h}_{N+1}$$

1) Calculate F_1 and h_1
2) Use update equations to calculate new values of F and h
3) Use F and h to calculate parameter estimates
4) Maybe disregard first few parameter estimates as transient

How many observation do you *need* to estimate p parameters?

Local Trend model, iterative updates

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

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

$$\begin{aligned}
\boldsymbol{F}_{N+1} &= \boldsymbol{F}_N + \lambda^N \boldsymbol{f}(-N)\boldsymbol{f}^T(-N) \\
\boldsymbol{h}_{N+1} &= \lambda \boldsymbol{L}^{-1}\boldsymbol{h}_N + \boldsymbol{f}(0)\,Y_{N+1} \\
\widehat{\boldsymbol{\theta}}_{N+1} &= \boldsymbol{F}_{N+1}^{-1}\boldsymbol{h}_{N+1}
\end{aligned}$$

1) Calculate F_1 and h_1
2) Use update equations to calculate new values of F and h
3) Use F and h to calculate parameter estimates
4) Maybe disregard first few parameter estimates as transient

How many observation do you *need* to estimate p parameters?

- F_N need to be invertible in order to estimate parameters
- start by updating F and h a few times
- then estimate parameters once you have enough observations

R example

```
f <- function(j) rbind(1, j)
L <- matrix(c(1.,0., 1.,1.),
            byrow=TRUE, nrow=2)
Linv <- solve(L)
```

$$Y_{N+j} = \theta_0 + \theta_1 j + \varepsilon_{N+j}$$

$$\boldsymbol{L} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, \qquad \boldsymbol{f}(0) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

```
i <- 1
(F_N <-  (lambda^0) * f(0)%*%t(f(0)))
(h_N <-  (lambda^0) * f(0)  * y[i])
```

Calculate F_1 and h_1

Using only one observation: y[1]

```
theta_N <- solve(F_N)%*%h_N
```

```
Fejl i solve.default(F_N) :
  Lapack routine dgesv: system is exactly singular: U[2,2] = 0
```

Too soon to estimate parameters!

R example

```r
i <- 2
F_N <- F_N + lambda^(i-1) * f(-(i-1)) %*% t(f(-(i-1)))
h_N <- lambda * Linv %*% h_N + f(0)*y[i]
```

First update

Now we have F_2 and h_2

```r
theta_N <- solve(F_N)%*%h_N
```

This time estimation of parameters works

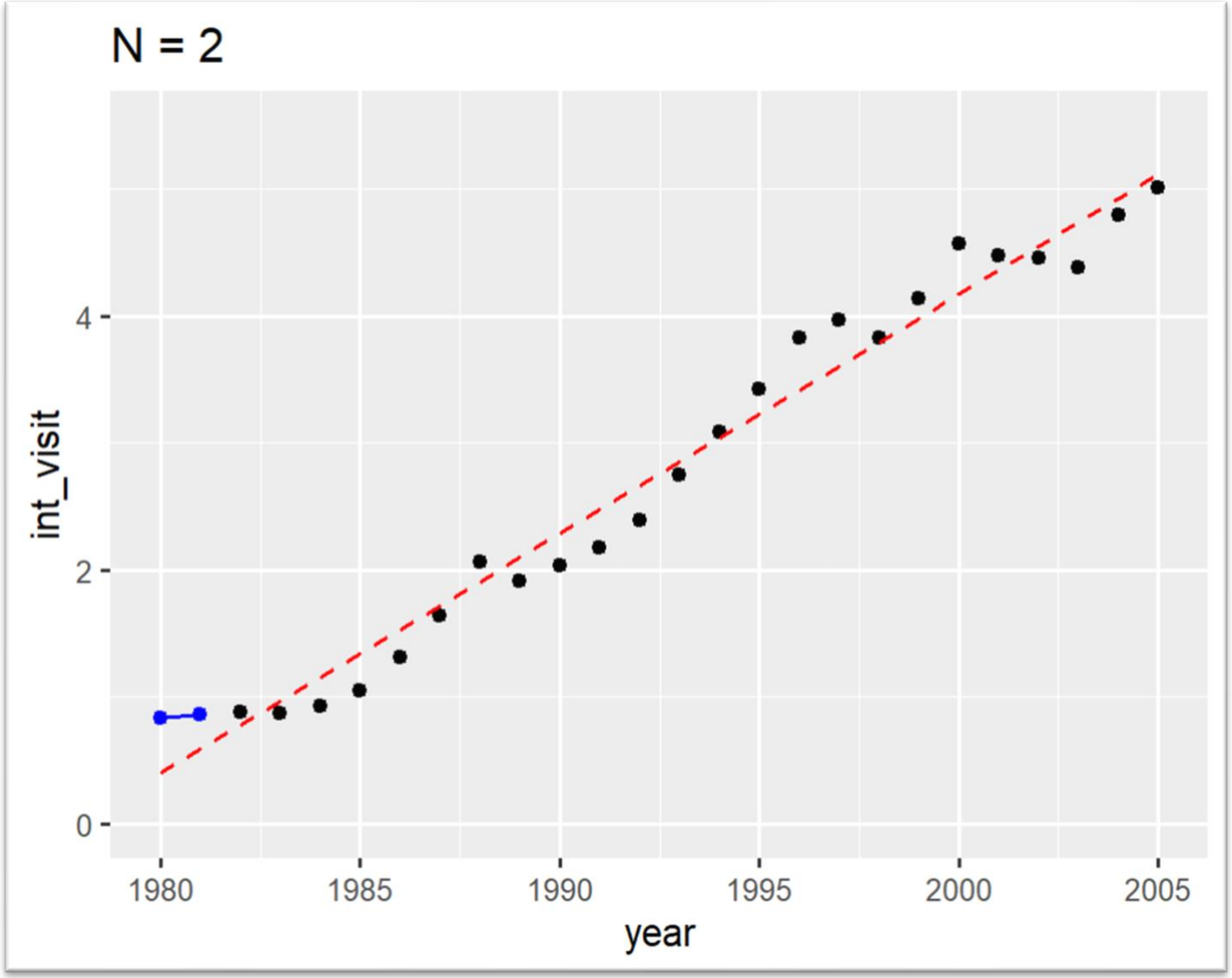
```
> theta_N
       [,1]
  4.9551683
j 0.1510241
```

Intercept at N = 2

Slope calculated from only two observations

(and weighted by lambda)

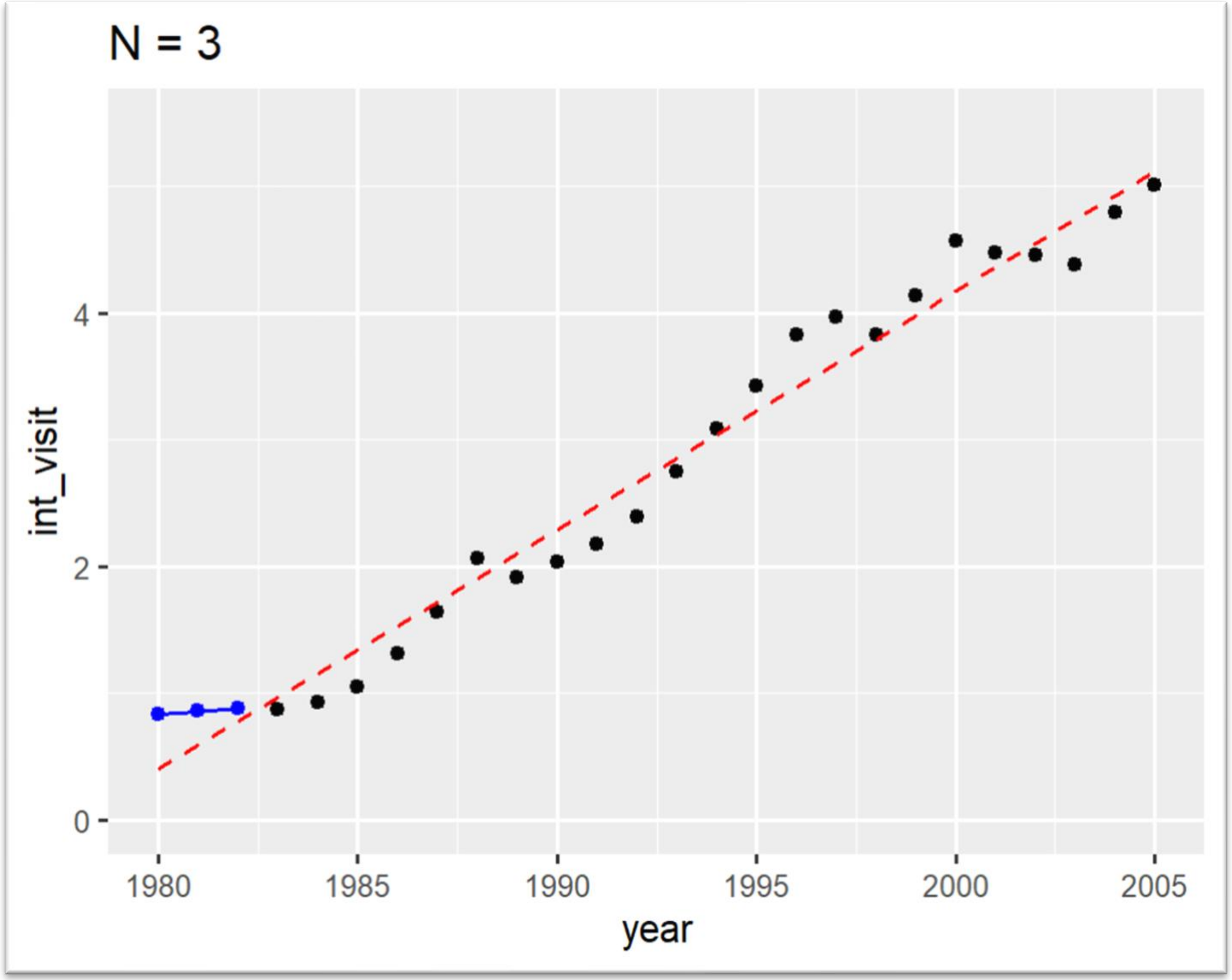
R example



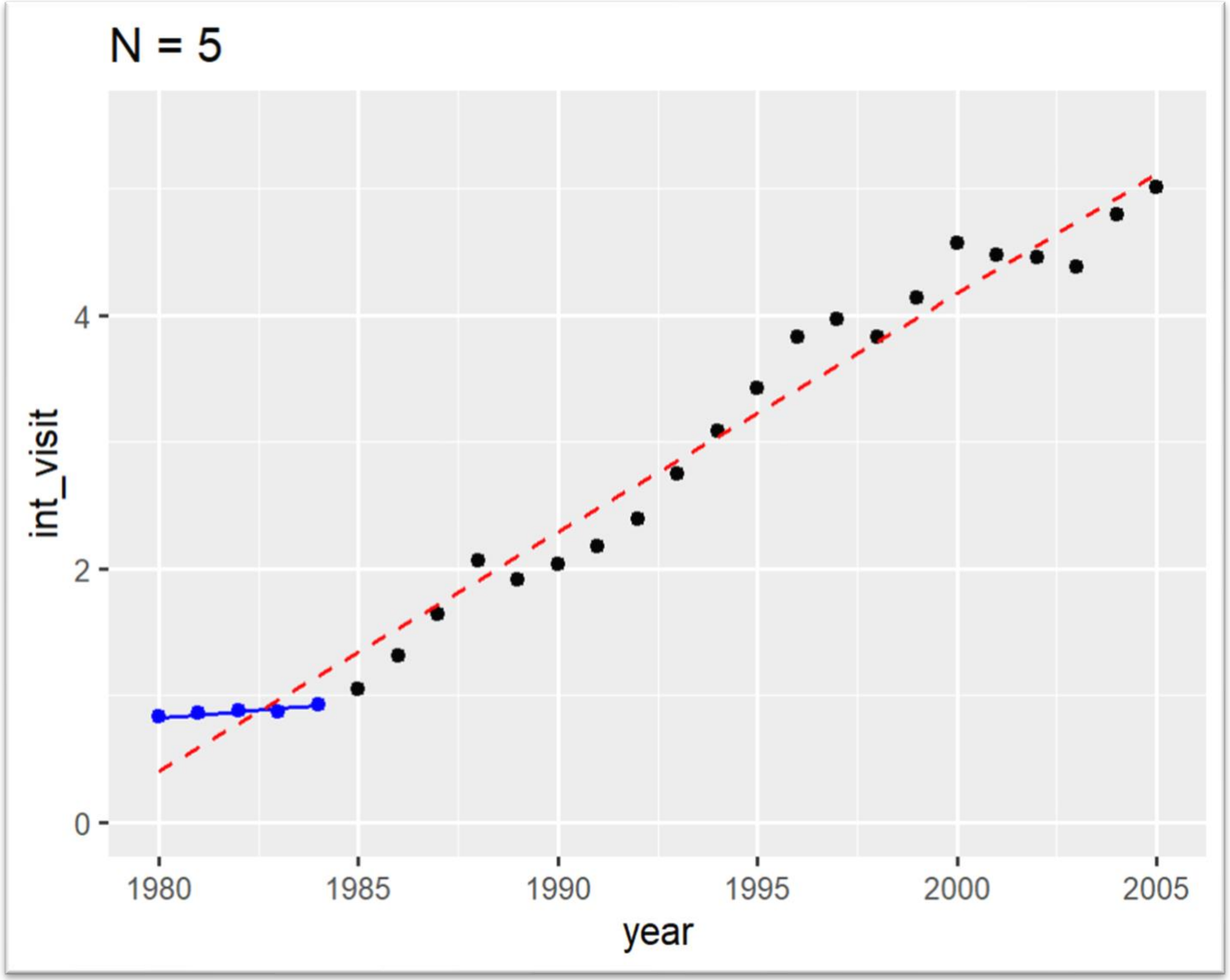
Blue line is local model based on only two observations

Red line is the original OLS based on all 26 observations

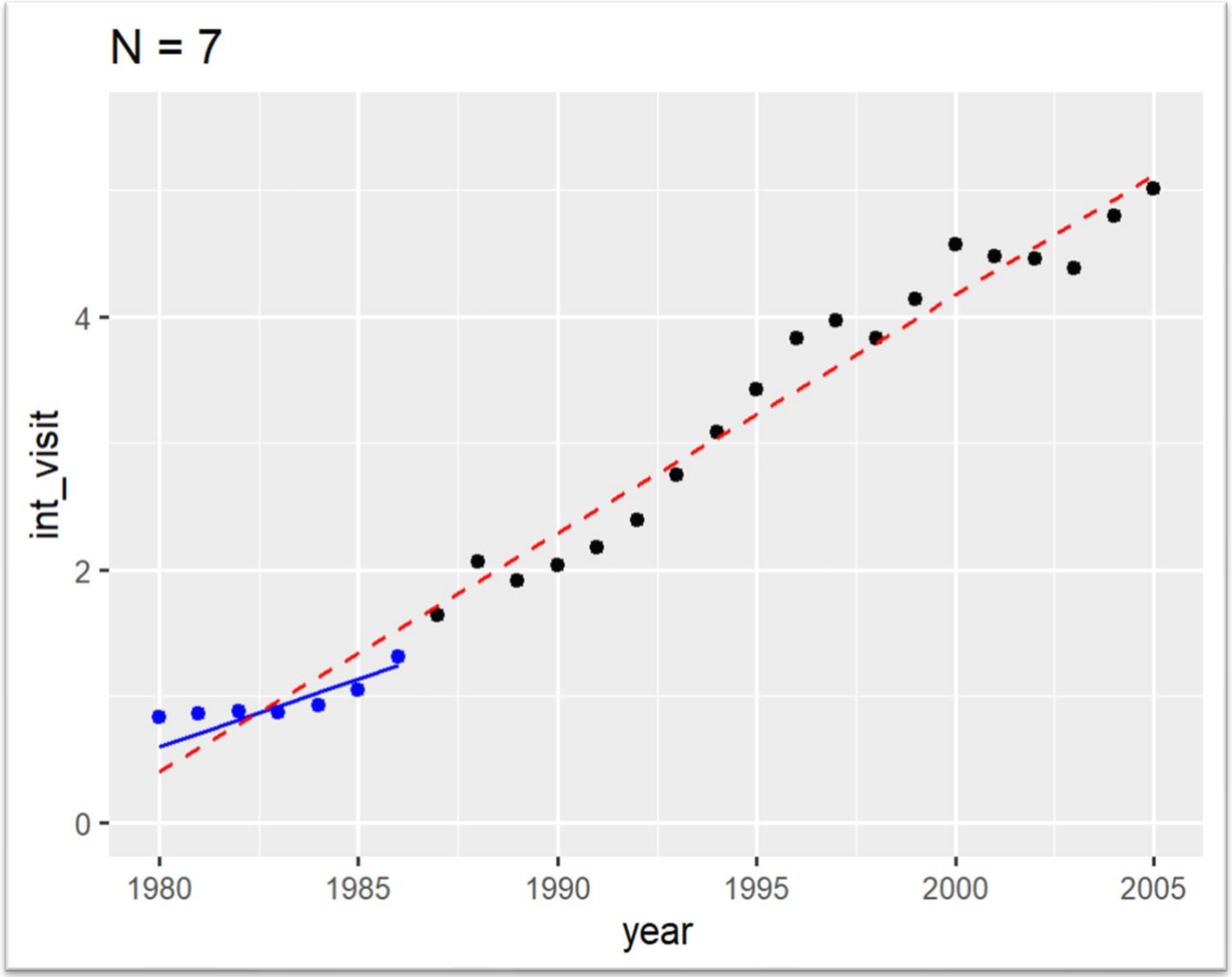
R example



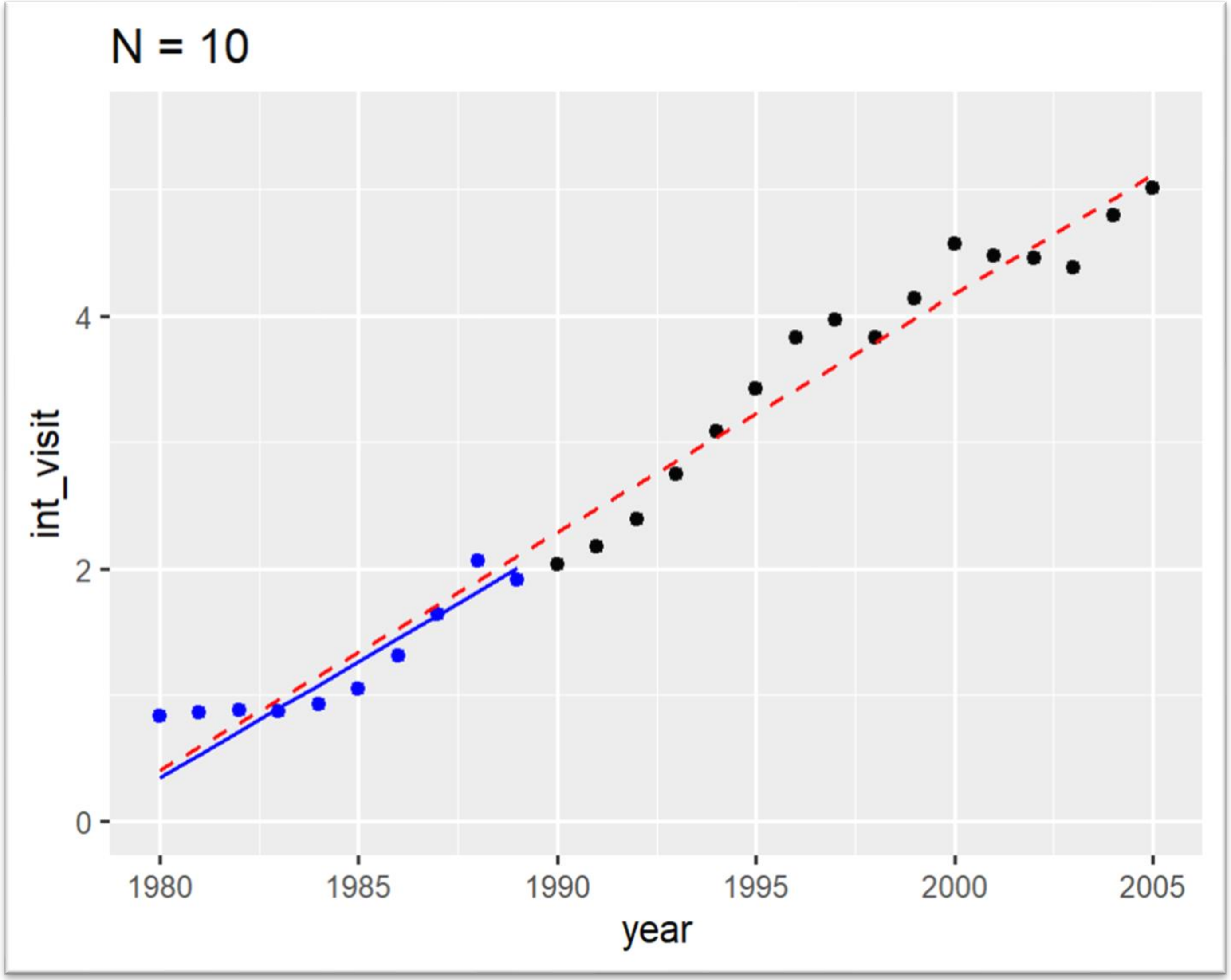
R example



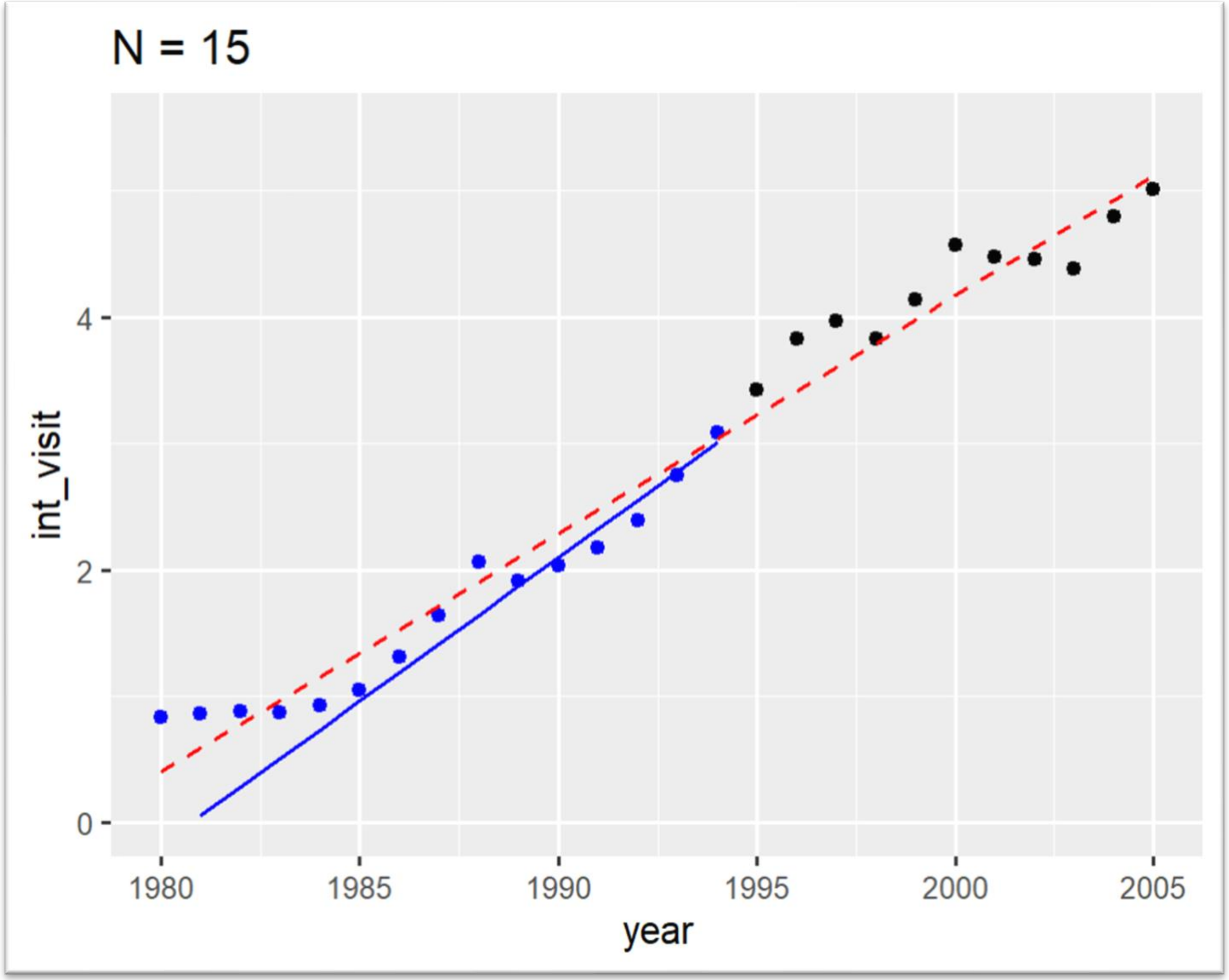
R example



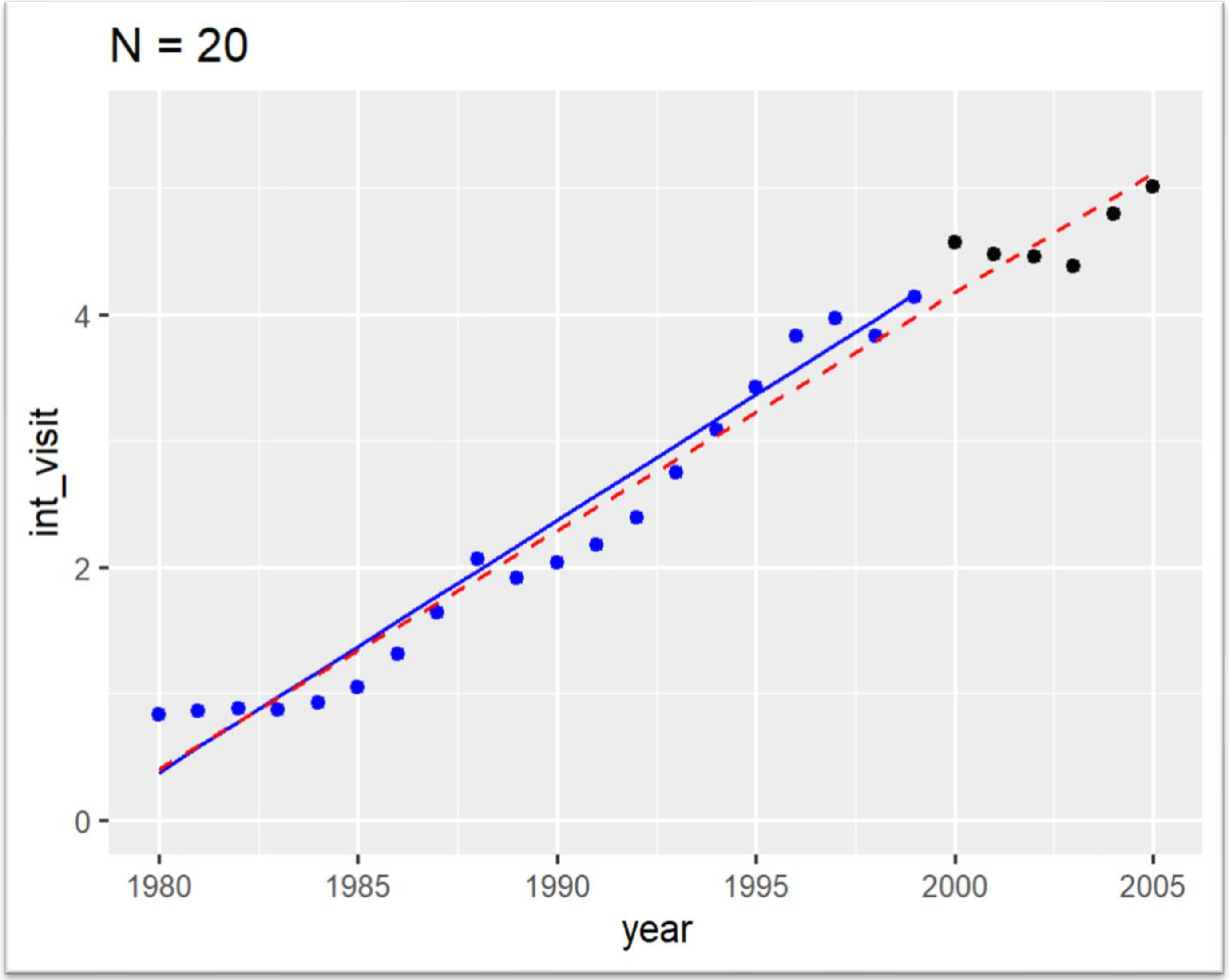
R example



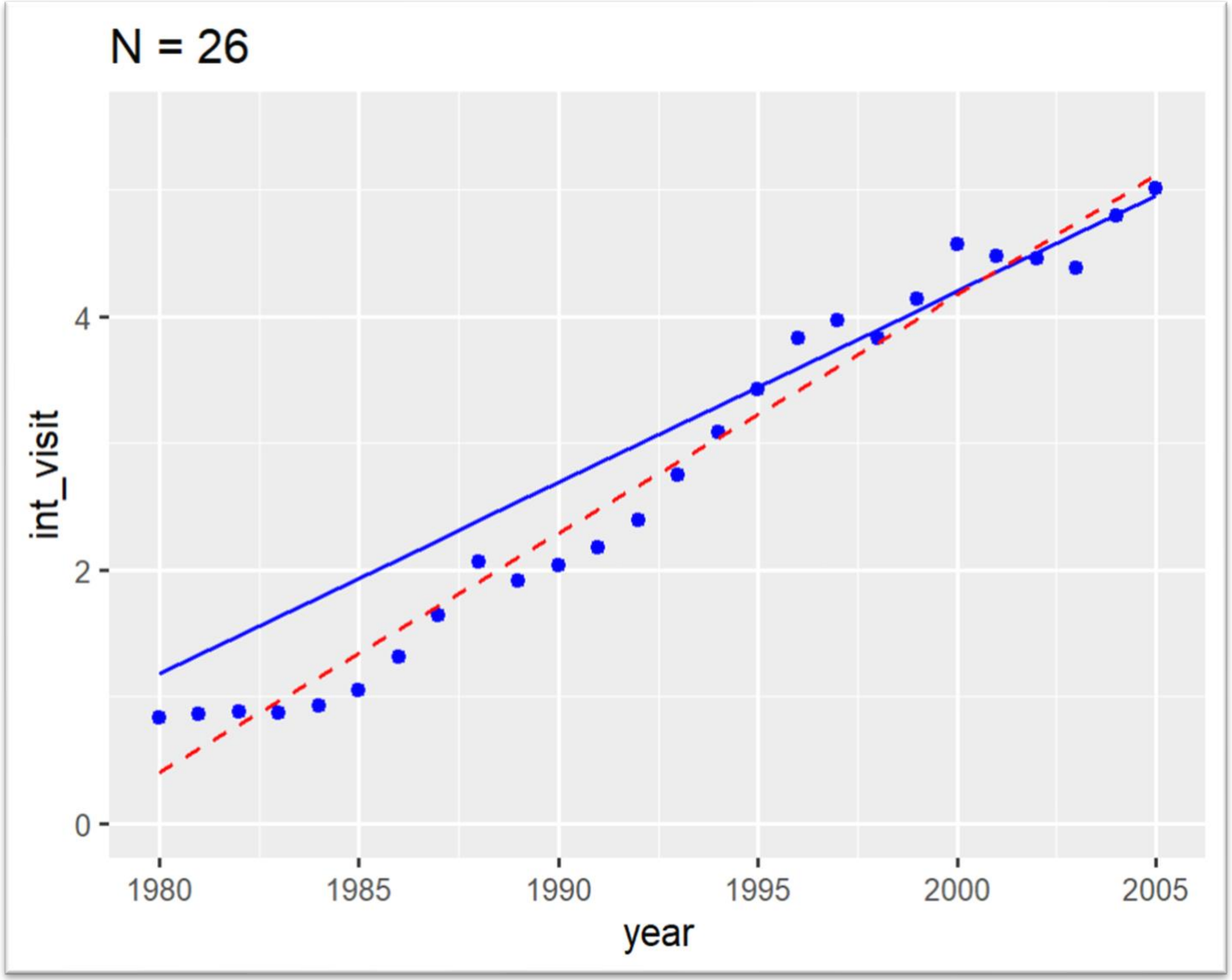
R example



R example



R example



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

Here we used lambda = 0.6

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

Why do all this ??

Why do all this ??

Imagine new observations are available as time passes
We want to make best prediction based on the available data at every timepoint

We also want to evaluate the model performance for different values of lambda

Local Trend model forecasting ("ℓ-step predictions")

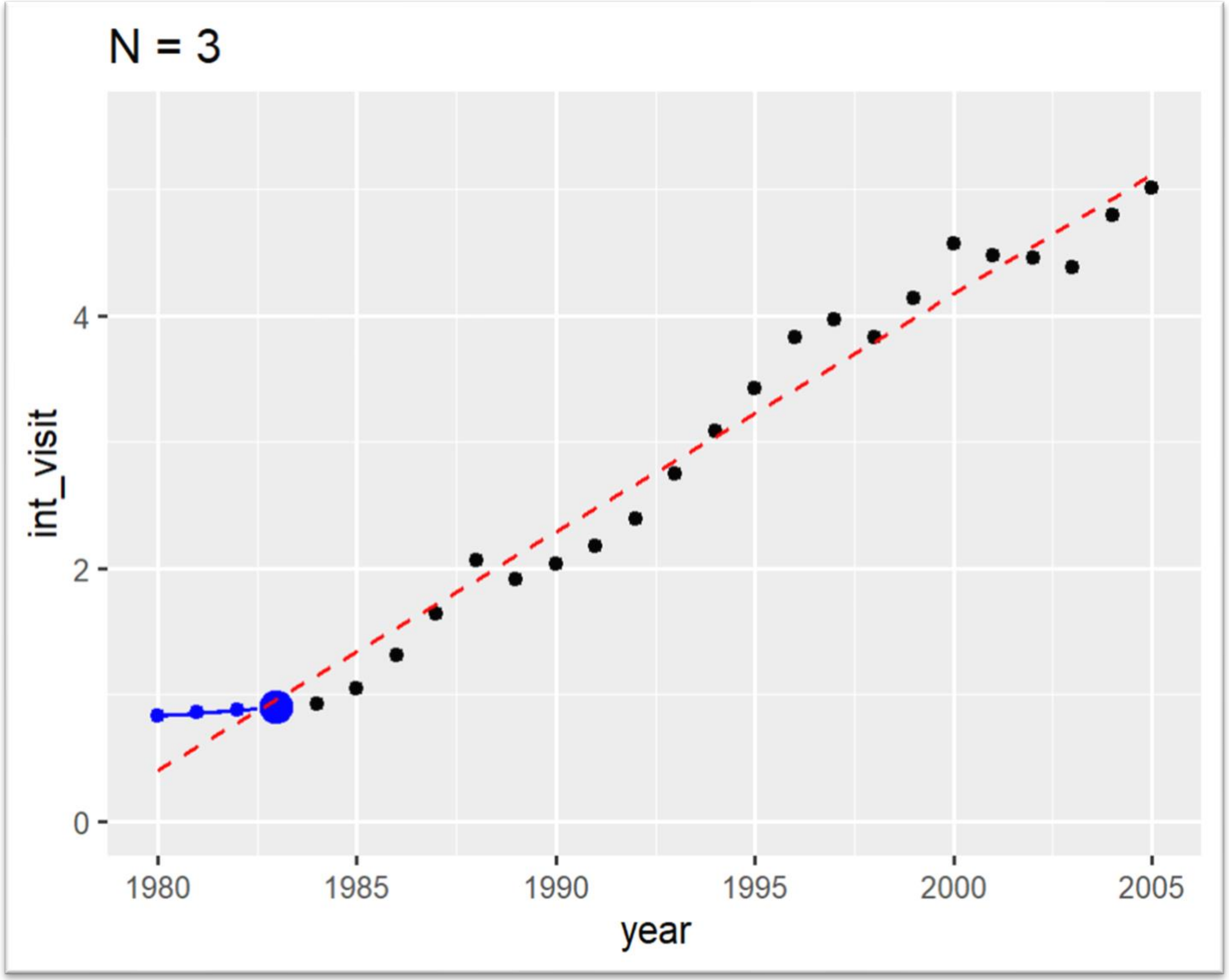
At every value of N in our iteration we could forecast future values using the most recent values of parameter estimates.

$$\widehat{Y}_{N+\ell|N} = \boldsymbol{f}^T(\ell)\widehat{\boldsymbol{\theta}}_N$$

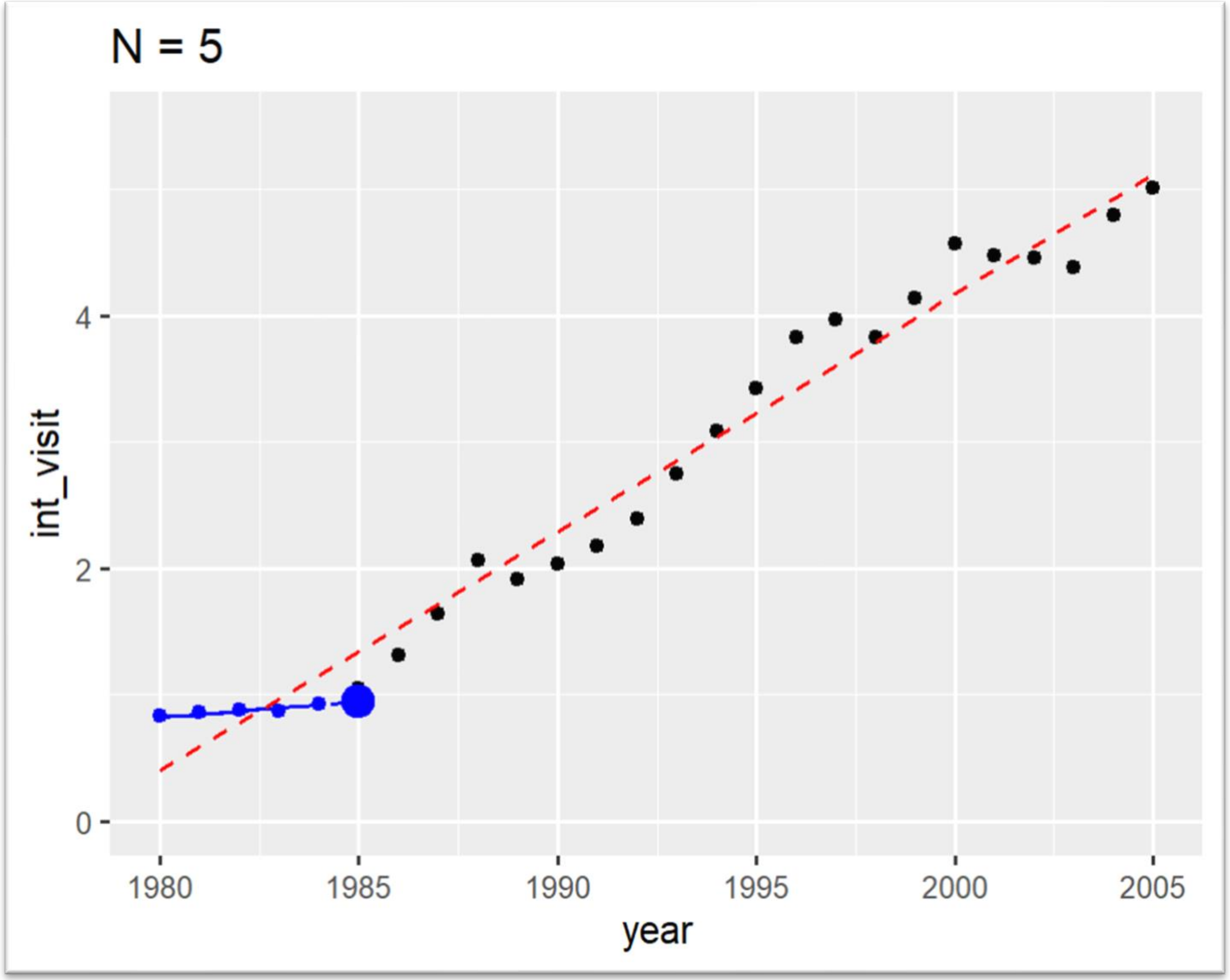
Recall "ℓ-step predictions" from last time?

Forecasting only one time-step ahead is called one-step predictions (ℓ = 1)

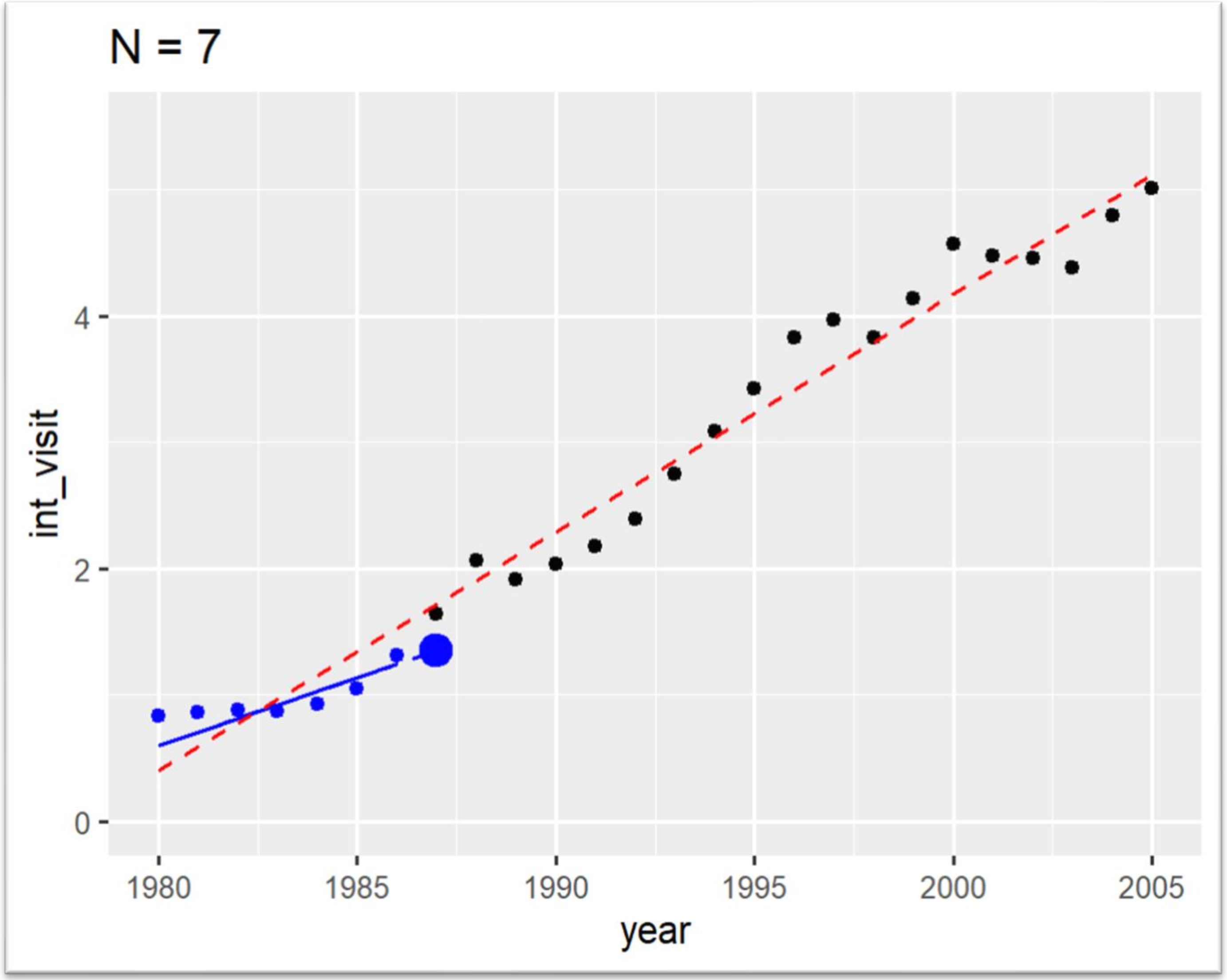
R example – onestep prediction



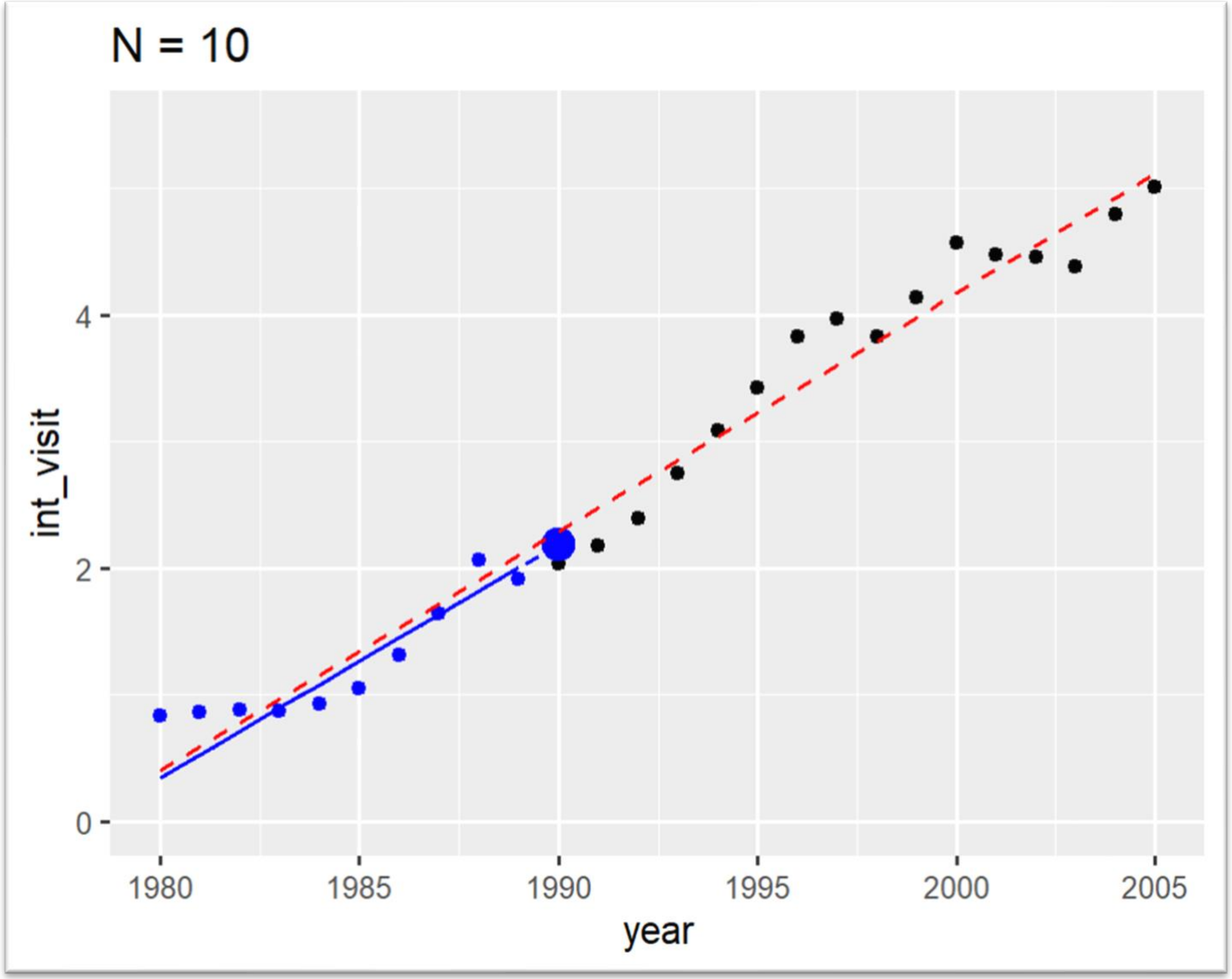
R example – onestep prediction



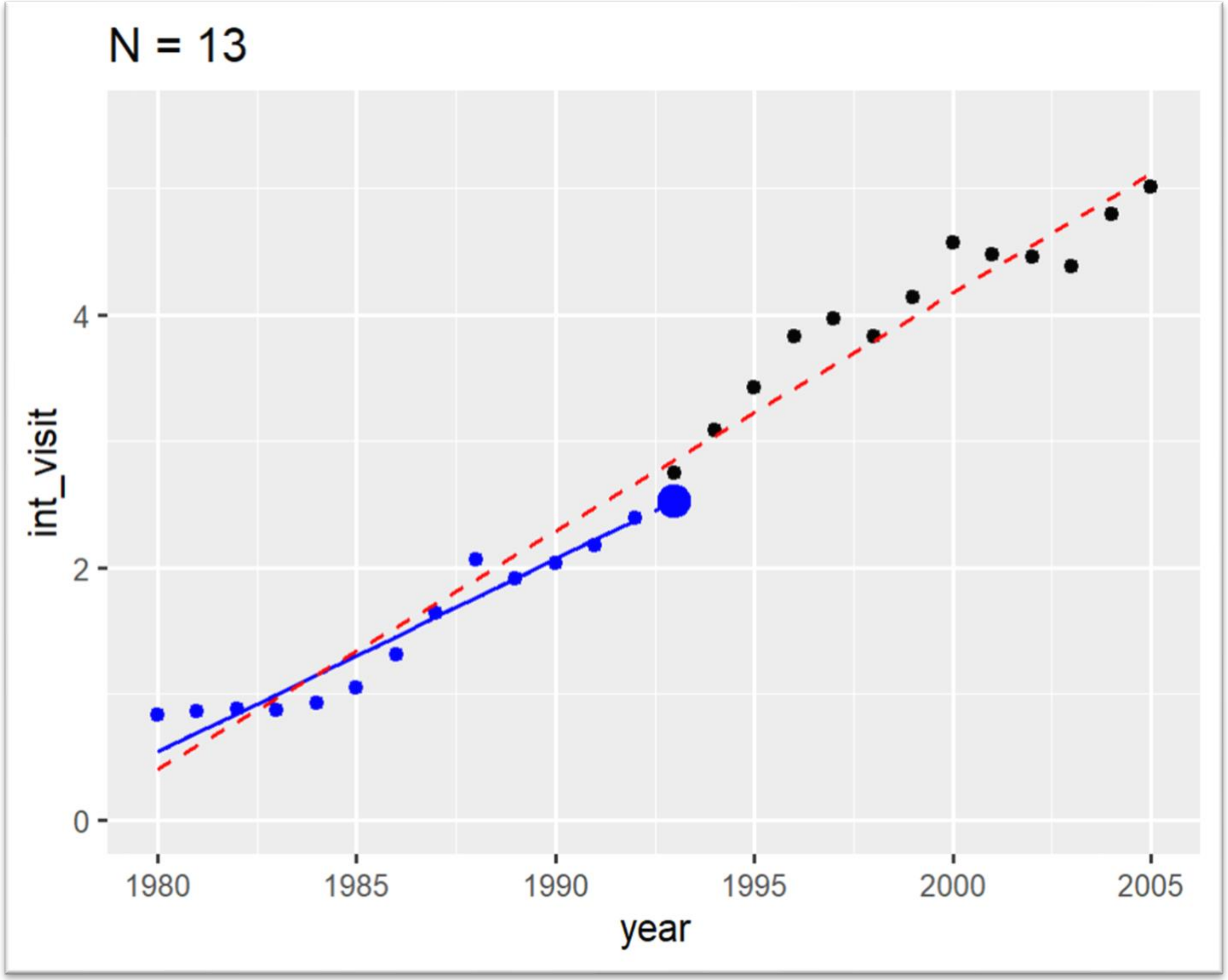
R example – onestep prediction



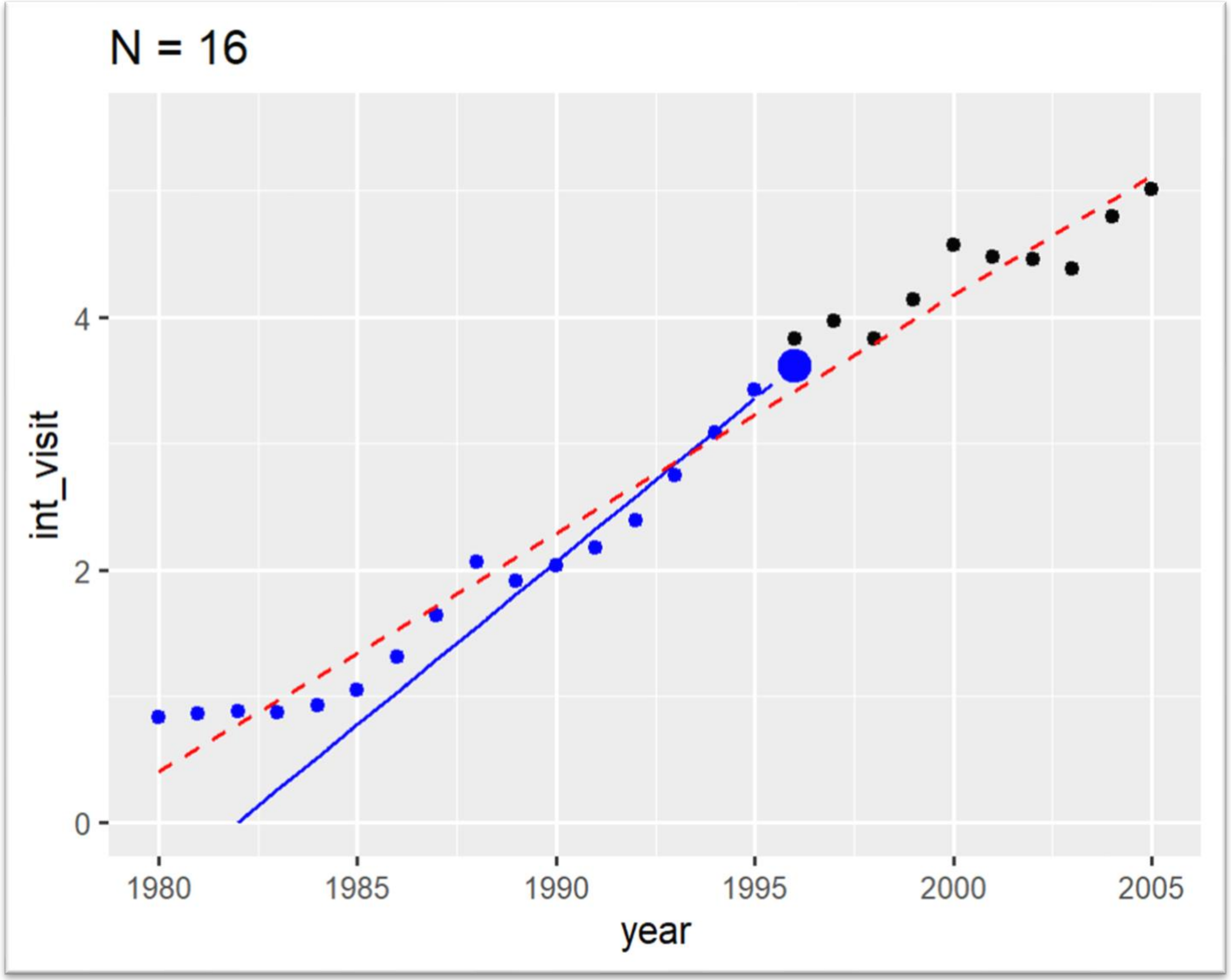
R example – onestep prediction



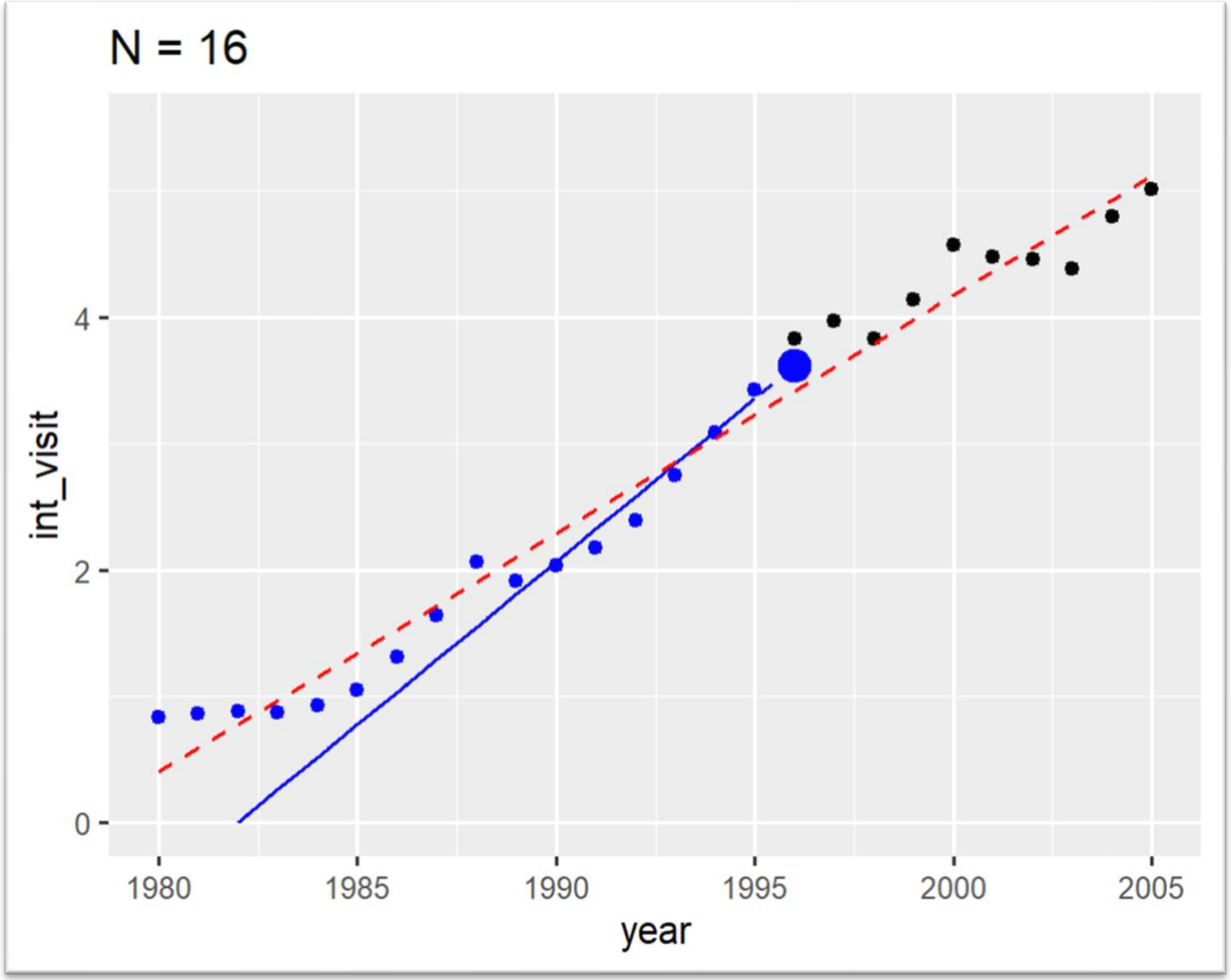
R example – onestep prediction



R example – onestep prediction

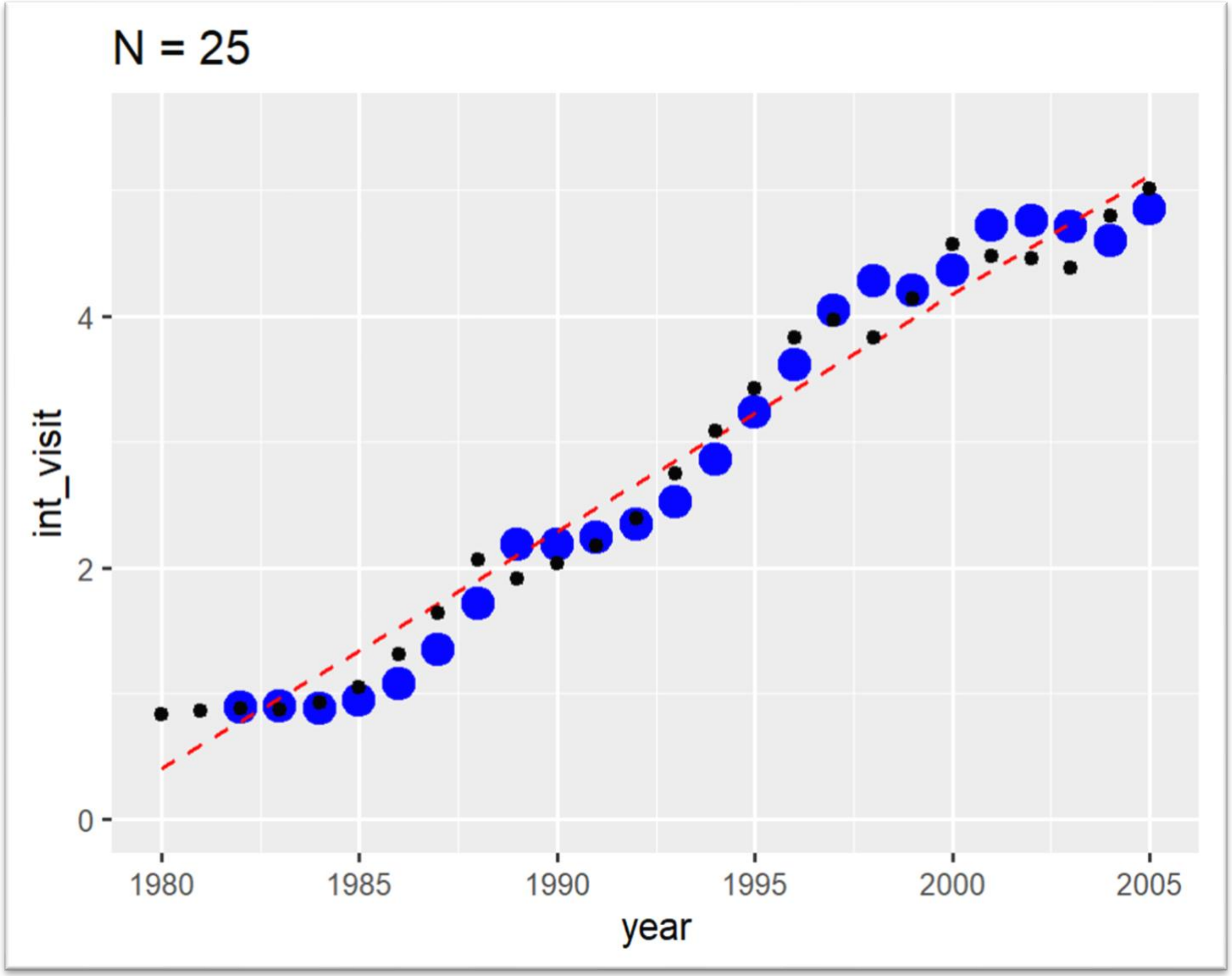


R example – onestep prediction



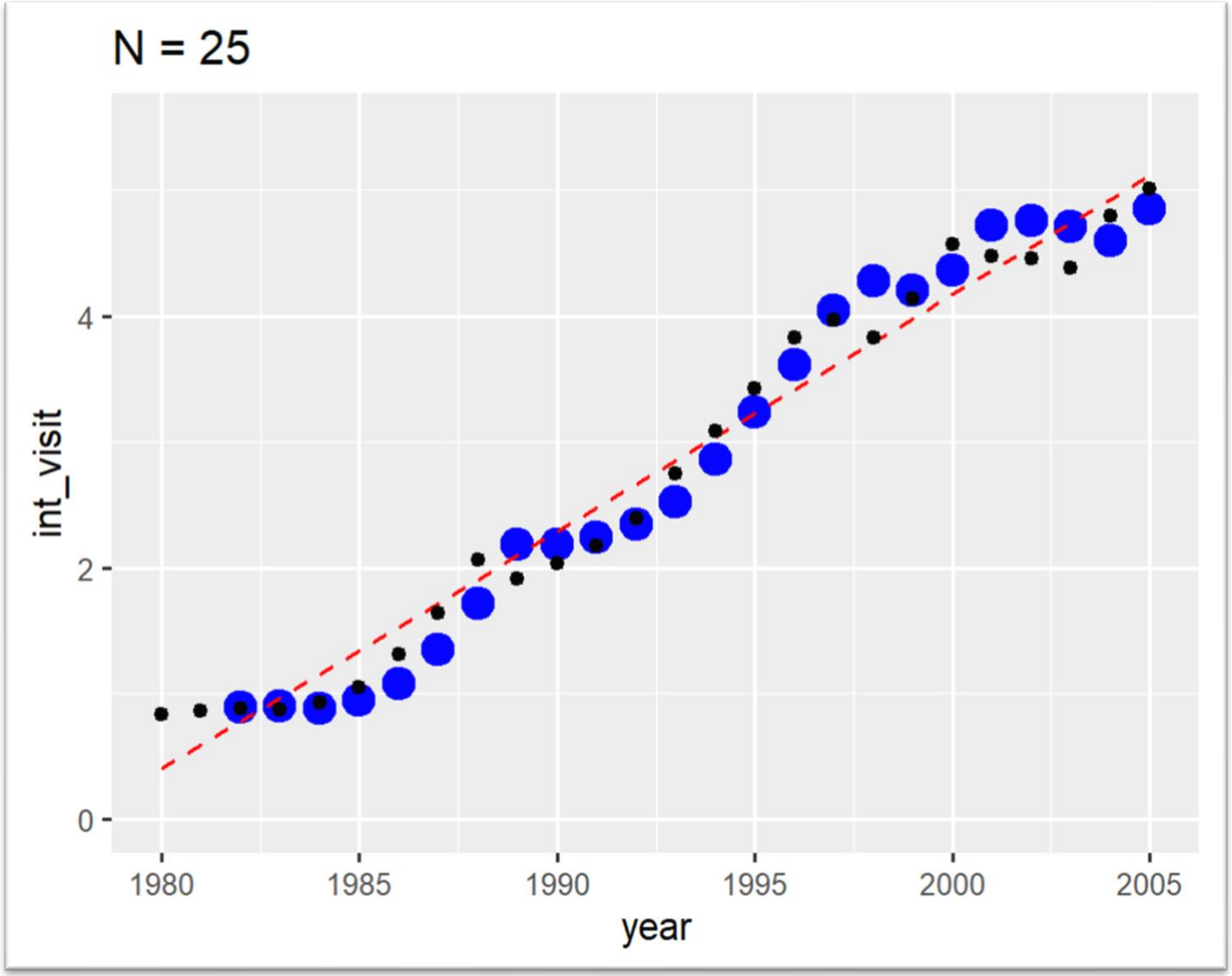
Each onestep prediction is predicted using a new (updated) set of parameters

R example – onestep prediction



Plot of all the onestep predictions

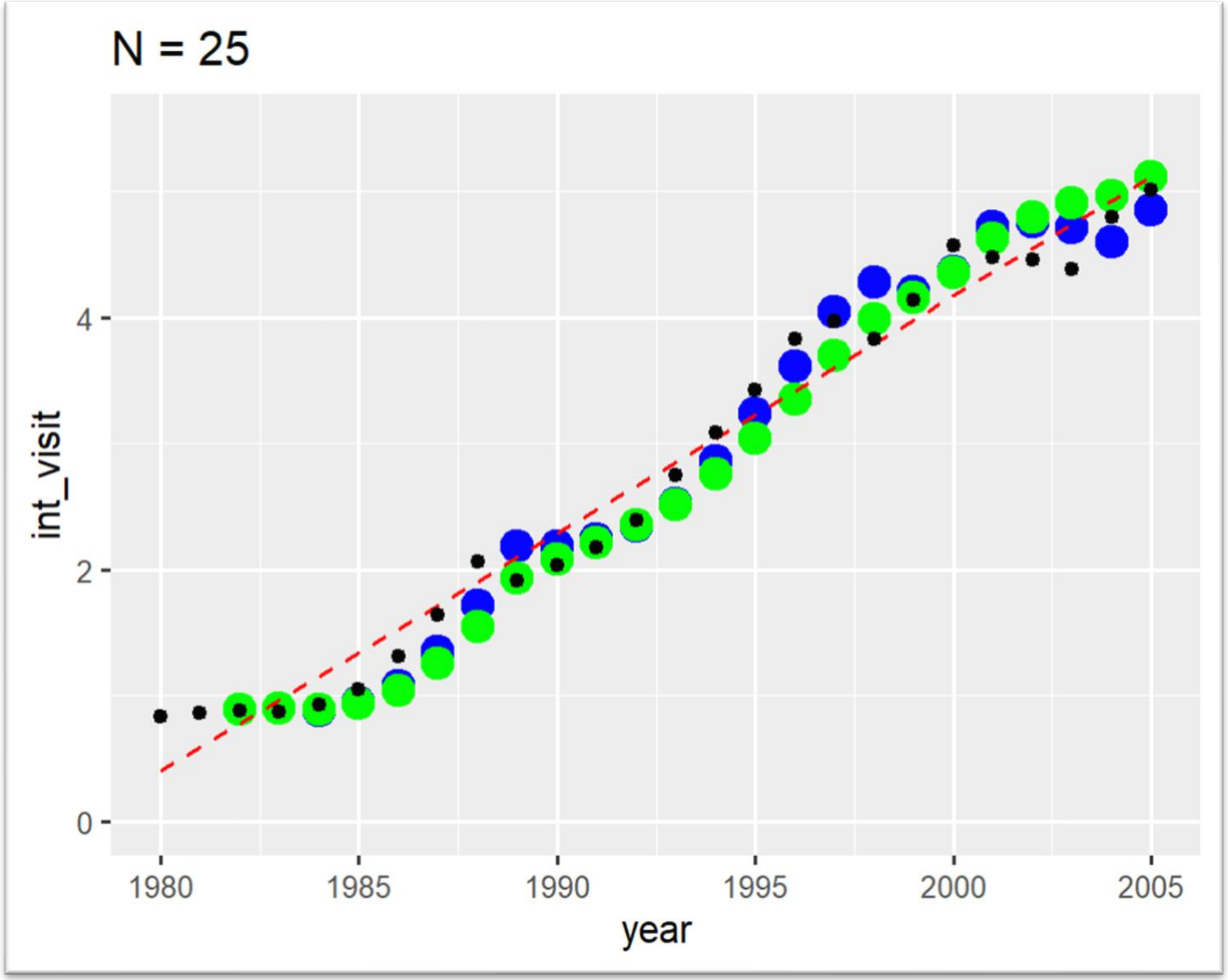
R example – onestep prediction



Plot of all the onestep predictions

We can try with different values of lambda

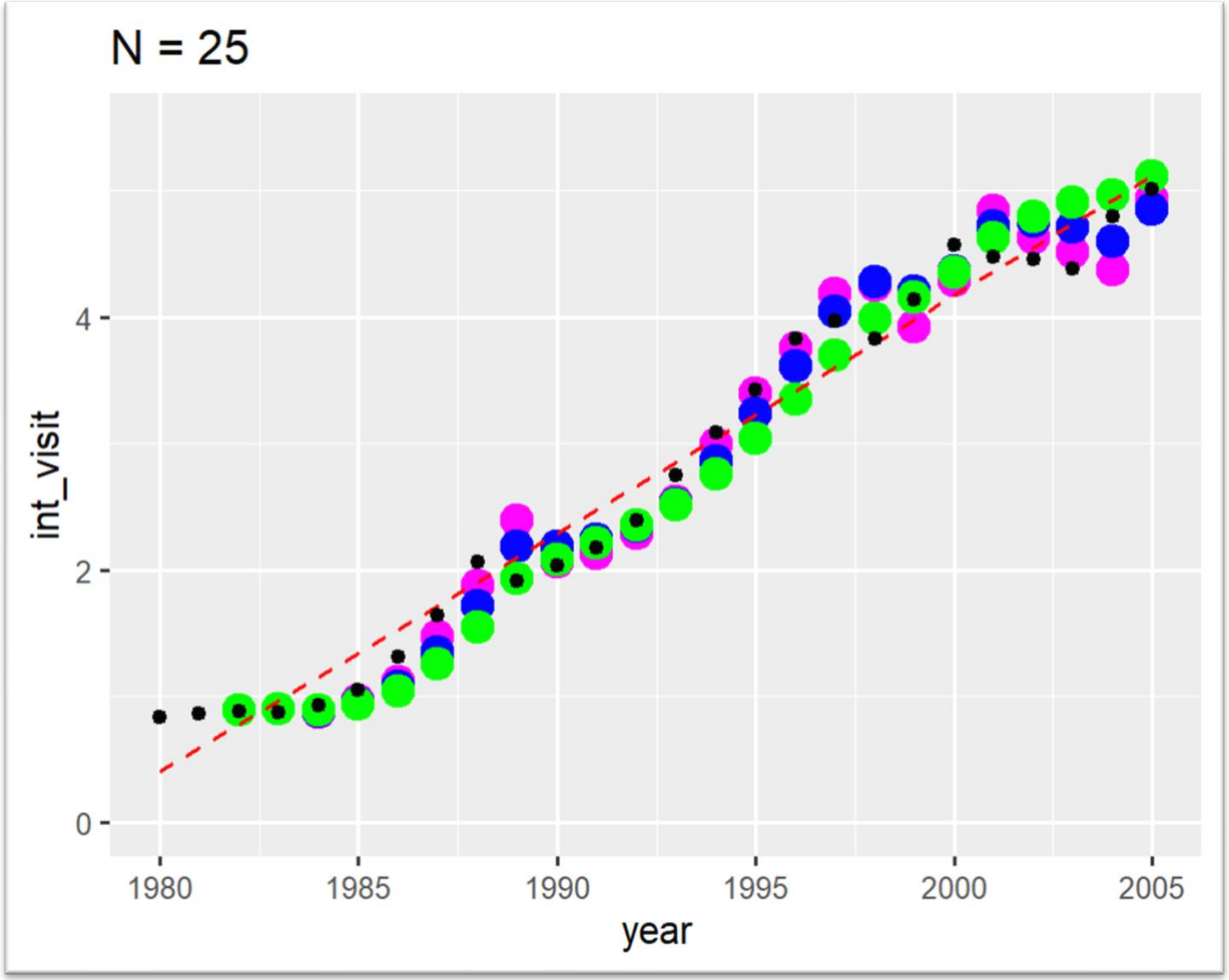
R example – onestep prediction



Blue: lambda = 0.6

Green: lambda = 0.9

R example – onestep prediction

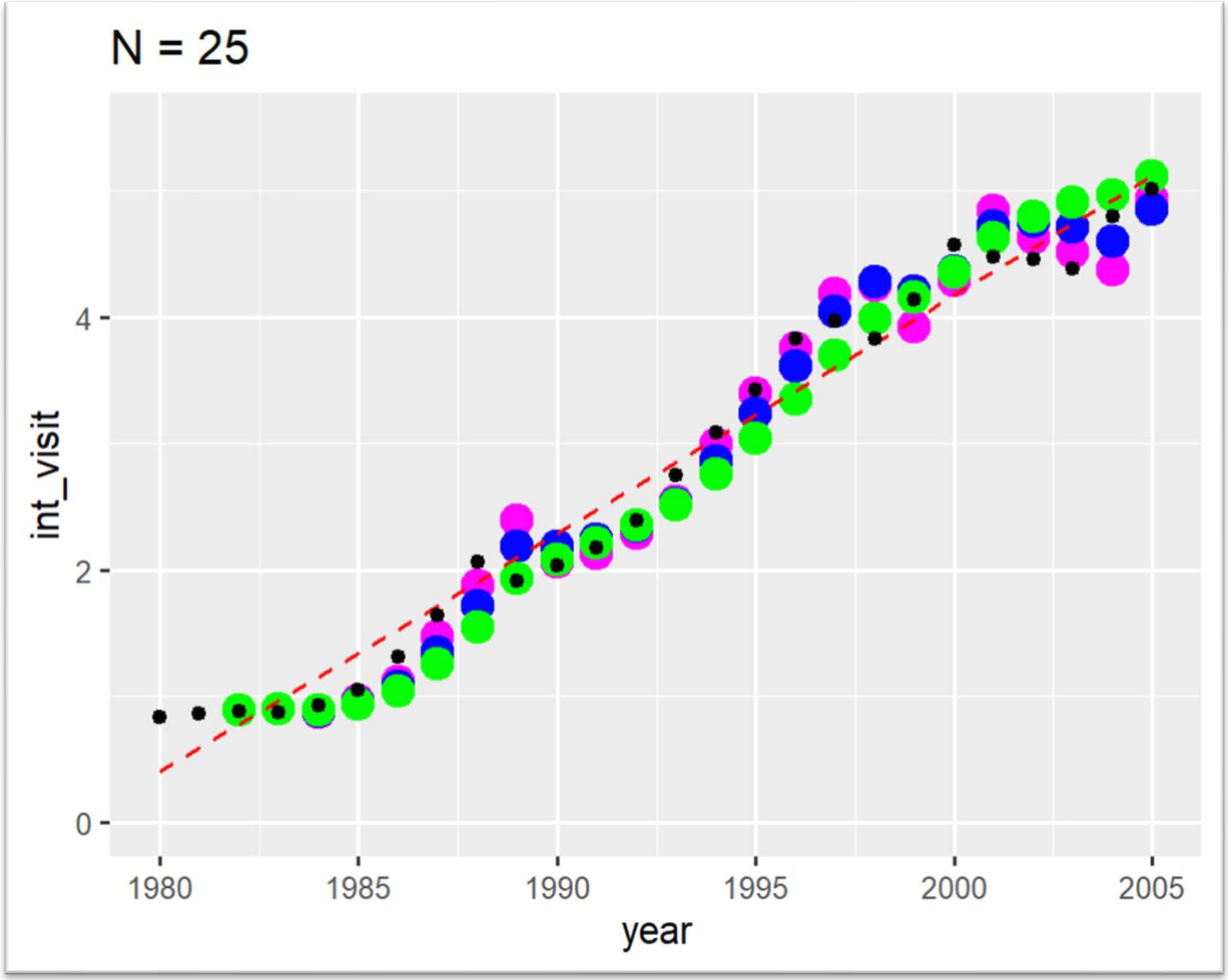


Blue: lambda = 0.6

Green: lambda = 0.9

Pink: lambda = 0.3

R example – onestep prediction



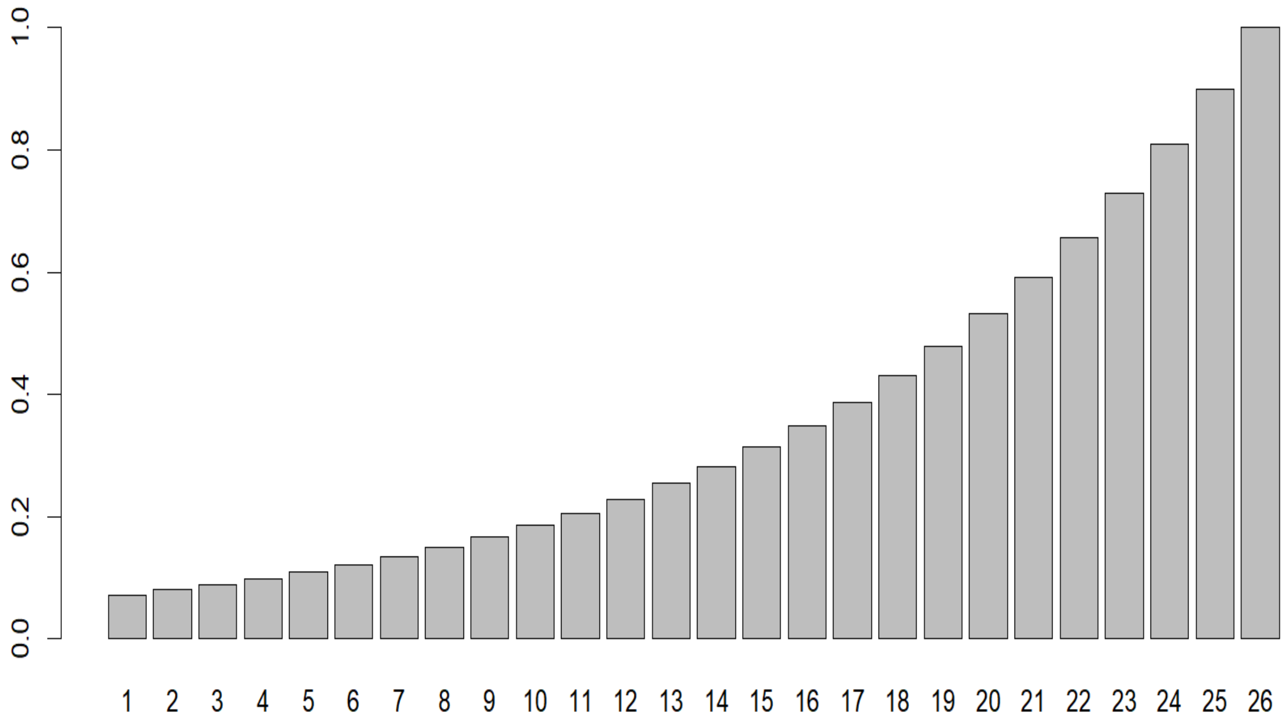
Blue: lambda = 0.6

Green: lambda = 0.9

Pink: lambda = 0.3

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

Choice of λ



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

$$0 < \lambda < 1$$

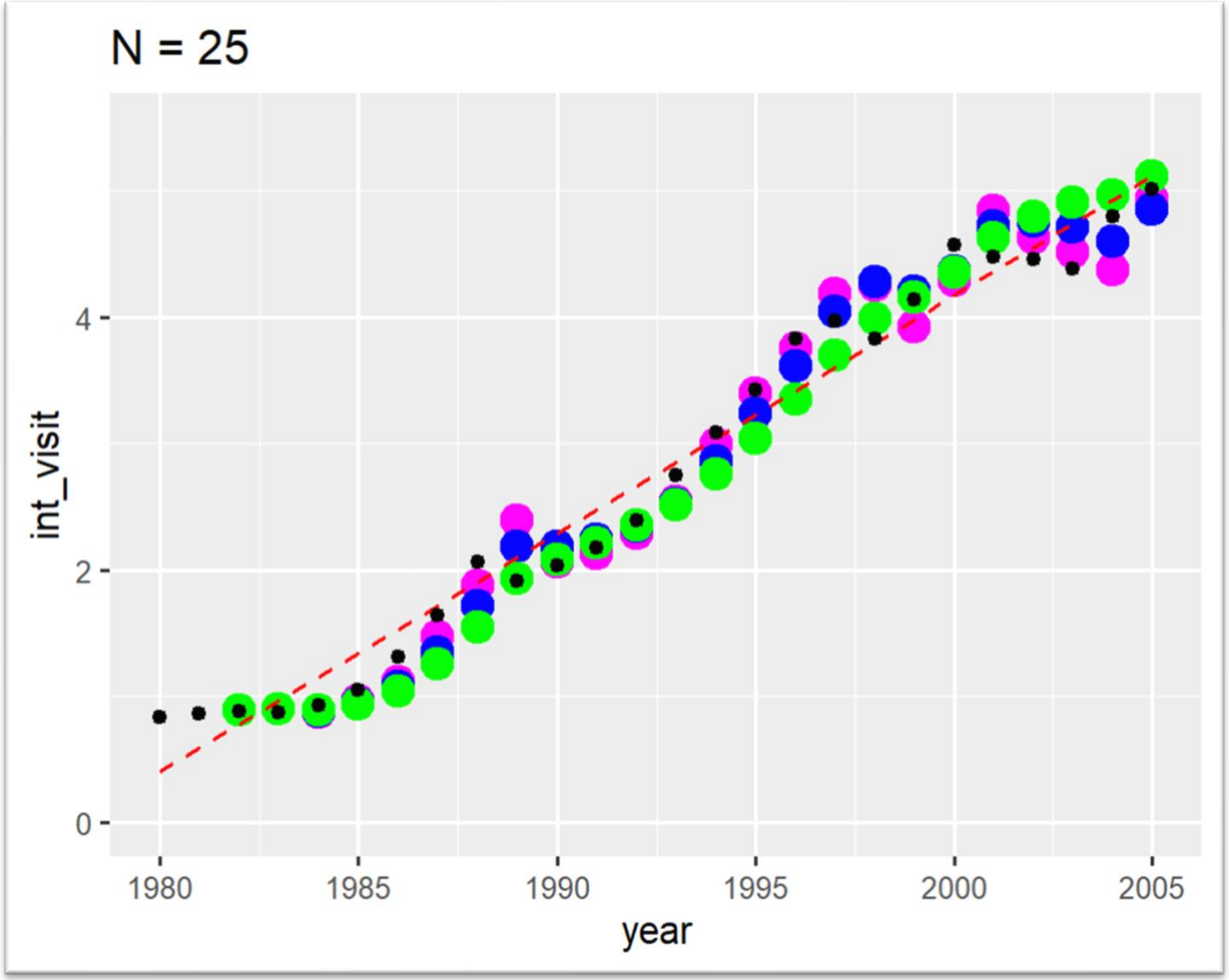
Larger λ - longer ”memory”

λ = 1 equals OLS

Choice of λ

For each onestep prediction we could calculate the prediction error once the next observation (y-value) is available

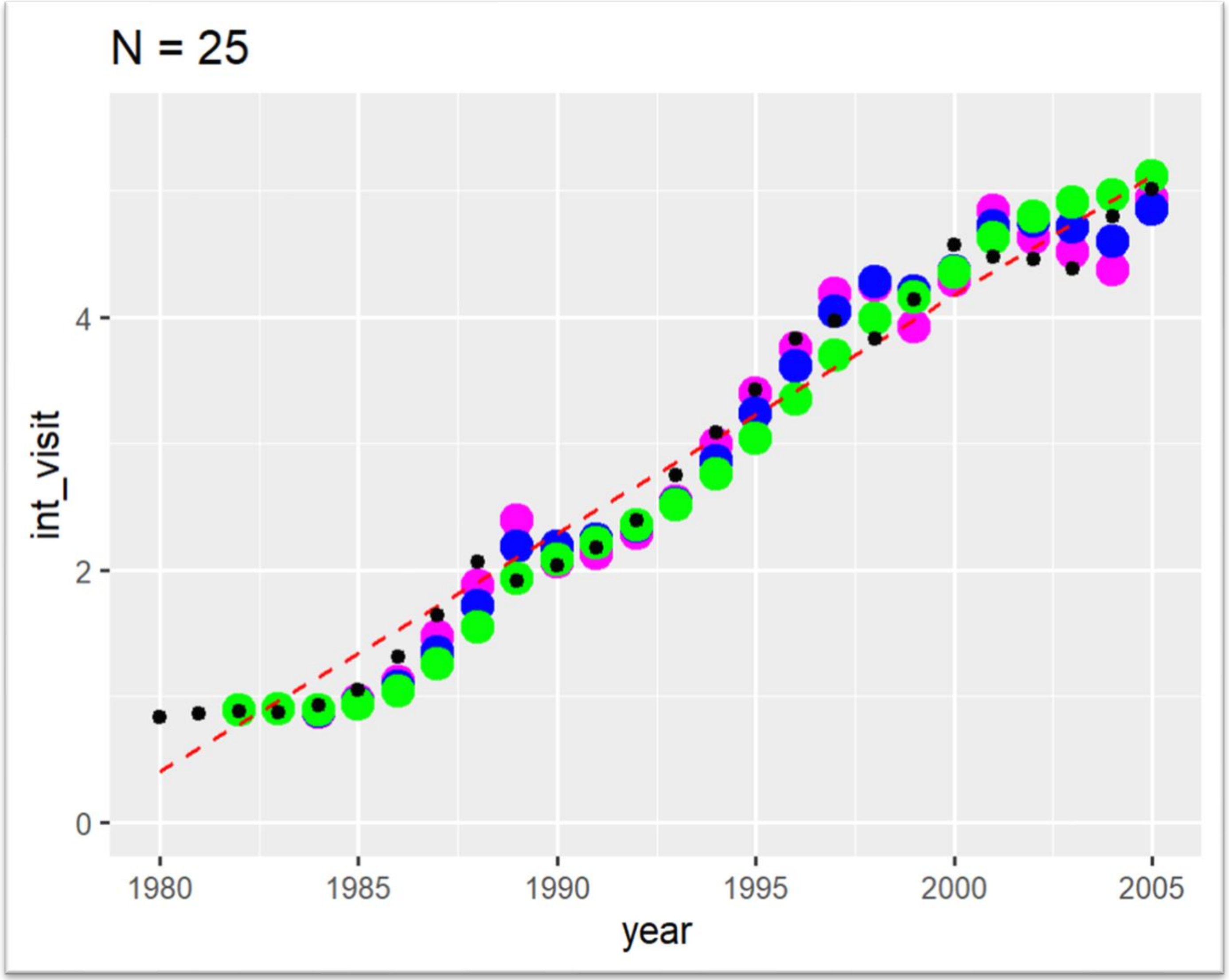
And then we could calculate which model gives results that are the closest to the actual observatione (smallest prediction errors)



Choice of λ

For each onestep prediction we could calculate the prediction error once the next observation (y-value) is available

And then we could calculate which model gives results that are the closest to the actual observatione (smallest prediction errors)



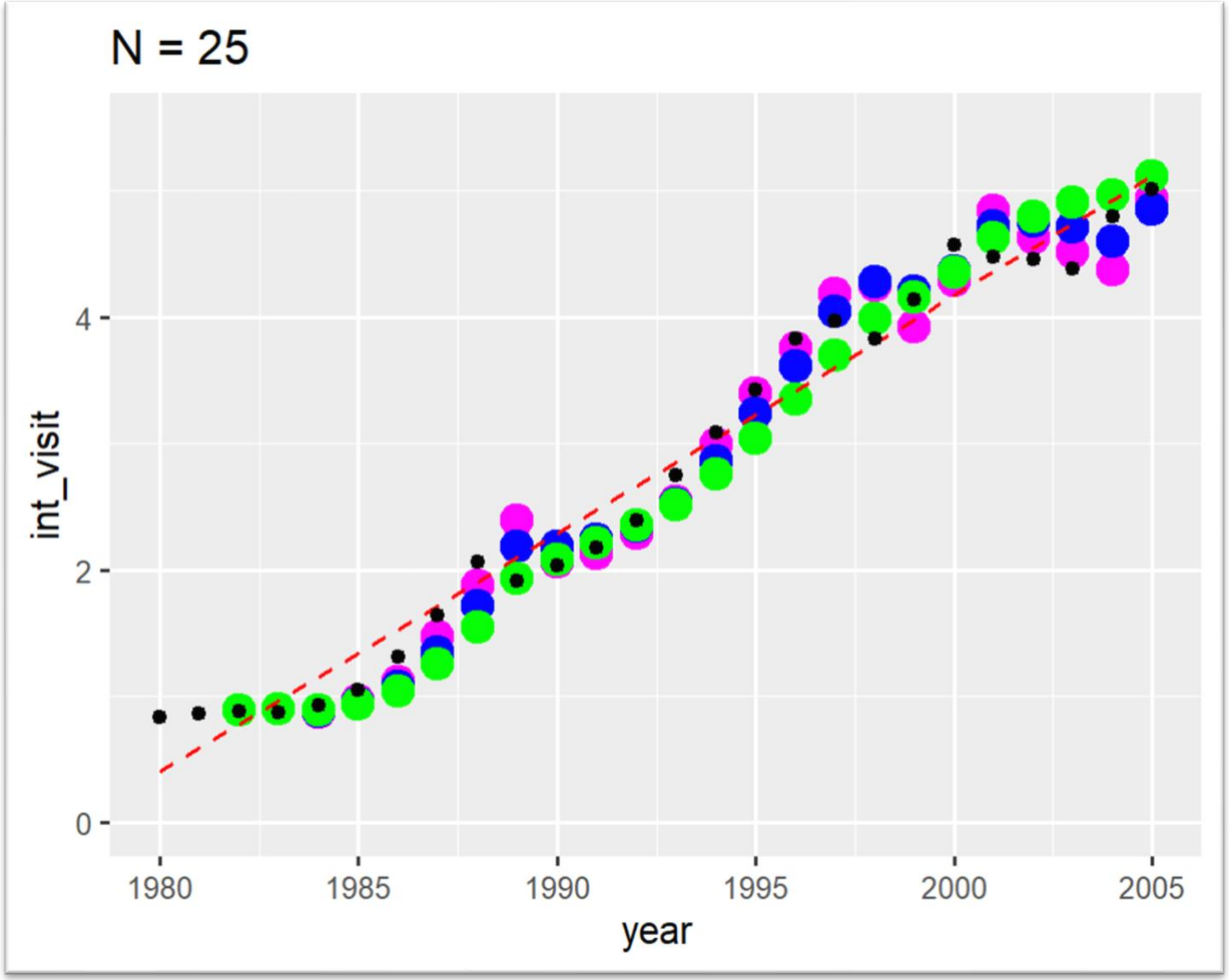
But what if we wanted to predict further than one step?

Could we expect the same lambda to be optimal?

Choice of λ

For each onestep prediction we could calculate the prediction error once the next observation (y-value) is available

And then we could calculate which model gives results that are the closest to the actual observatione (smallest prediction errors)



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

Could we expect the same lambda to be optimal?

And what about uncertainties (prediction intervals)?

Estimating uncertainty in Local Linear Trend model

<u>Recall from OLS</u>

smaller n = larger prediction intervals

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

$$\mathrm{Var}[\widehat{\boldsymbol{\theta}}] = \mathrm{E}\left[(\widehat{\boldsymbol{\theta}} - \boldsymbol{\theta})(\widehat{\boldsymbol{\theta}} - \boldsymbol{\theta})^T\right] = \sigma^2 (\boldsymbol{x}^T \boldsymbol{x})^{-1}$$

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

Estimating uncertainty in Local Linear Trend model

Recall from OLS

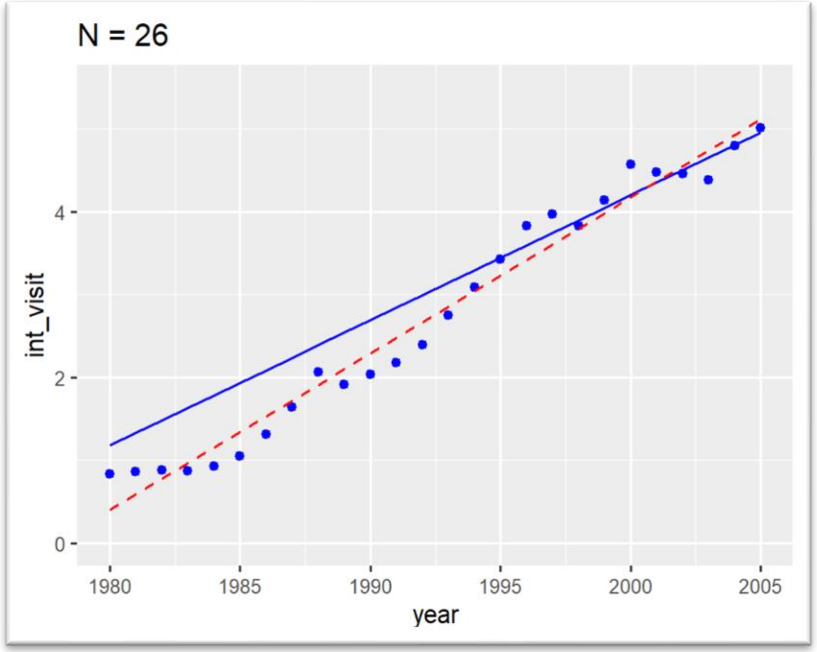
smaller n = larger prediction intervals

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

$$\mathrm{Var}[\widehat{\boldsymbol{\theta}}] = \mathrm{E}\left[(\widehat{\boldsymbol{\theta}} - \boldsymbol{\theta})(\widehat{\boldsymbol{\theta}} - \boldsymbol{\theta})^T\right] = \sigma^2(\boldsymbol{x}^T\boldsymbol{x})^{-1}$$

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

With WLS



Estimating uncertainty in Local Linear Trend model

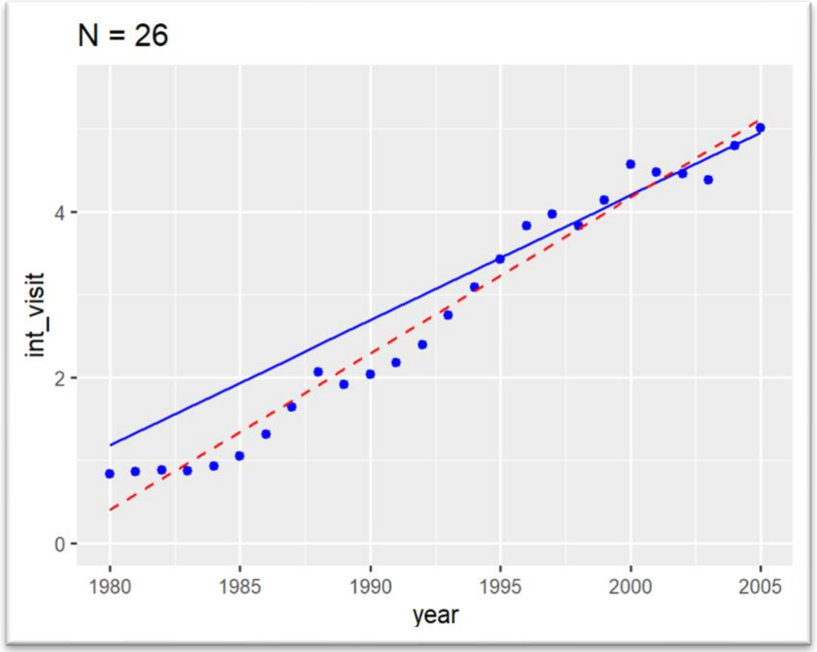
Recall from OLS
smaller n = larger prediction intervals

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

$$\mathrm{Var}[\widehat{\boldsymbol{\theta}}] = \mathrm{E}\left[(\widehat{\boldsymbol{\theta}}-\boldsymbol{\theta})(\widehat{\boldsymbol{\theta}}-\boldsymbol{\theta})^T\right] = \sigma^2(\boldsymbol{x}^T\boldsymbol{x})^{-1}$$

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

With WLS



Larger residuals

But is model *really* based on N observations?
If lambda is small some observations have close to zero weight

Estimating uncertainty in Local Linear Trend model (2)

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

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

T is a meassure of how many observations the estimation is essentially based upon

(In OLS the sum of weights = N)

Estimating uncertainty in Local Linear Trend model (2)

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

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

T is a meassure of how many observations the estimation is essentially based upon

(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 Linear Trend model (2)

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

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

T is a meassure of how many observations the estimation is essentially based upon

(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 Linear Trend model (2)

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

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

T is a meassure of how many observations the estimation is essentially based upon

(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

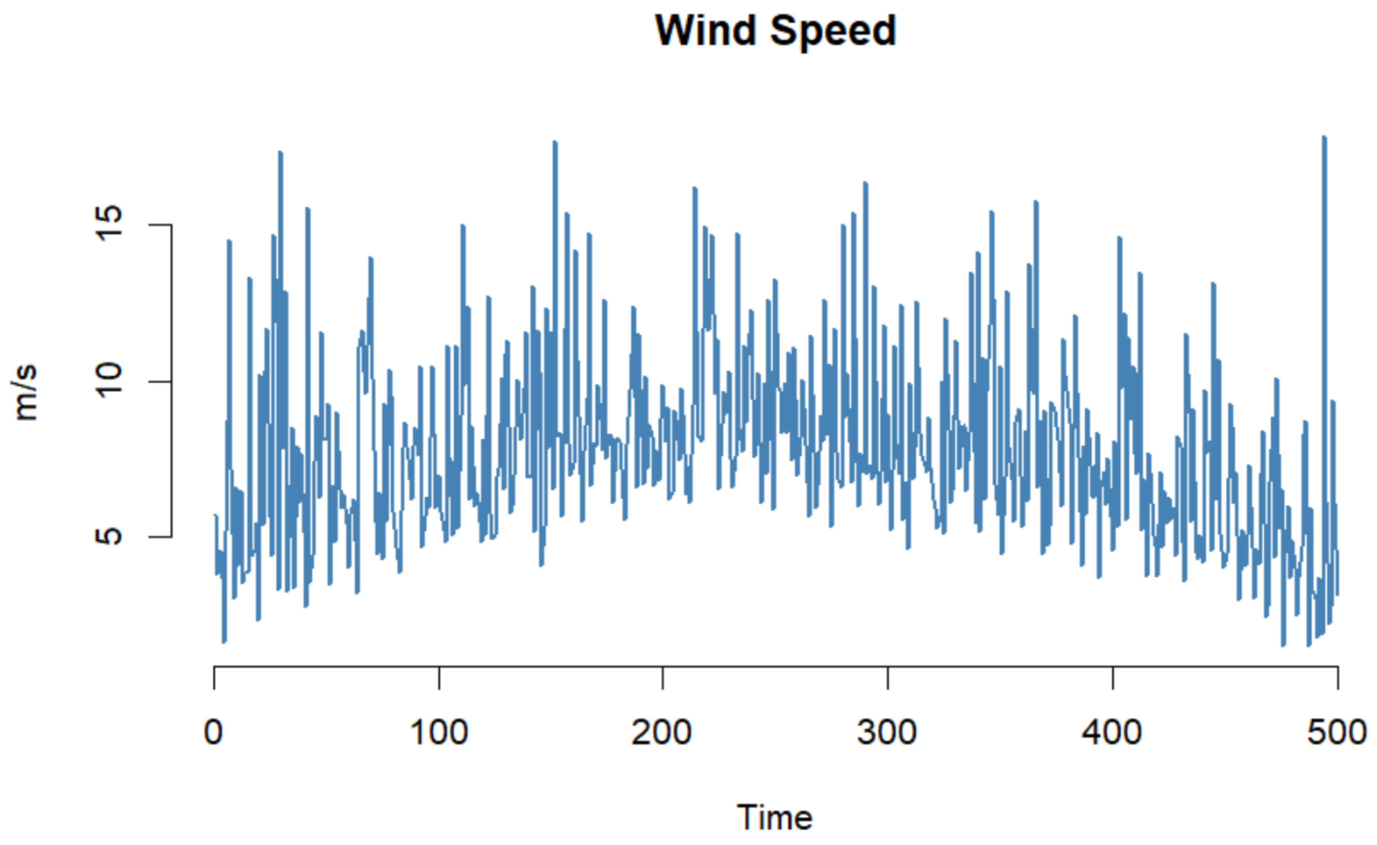
(lambda cannot be too small)

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

Outline of the lecture

- Regression based methods, 2nd part:
  - From Global models to Local models
  - WLS
  - Local Trend Model
  - **Exponential smoothing in general**

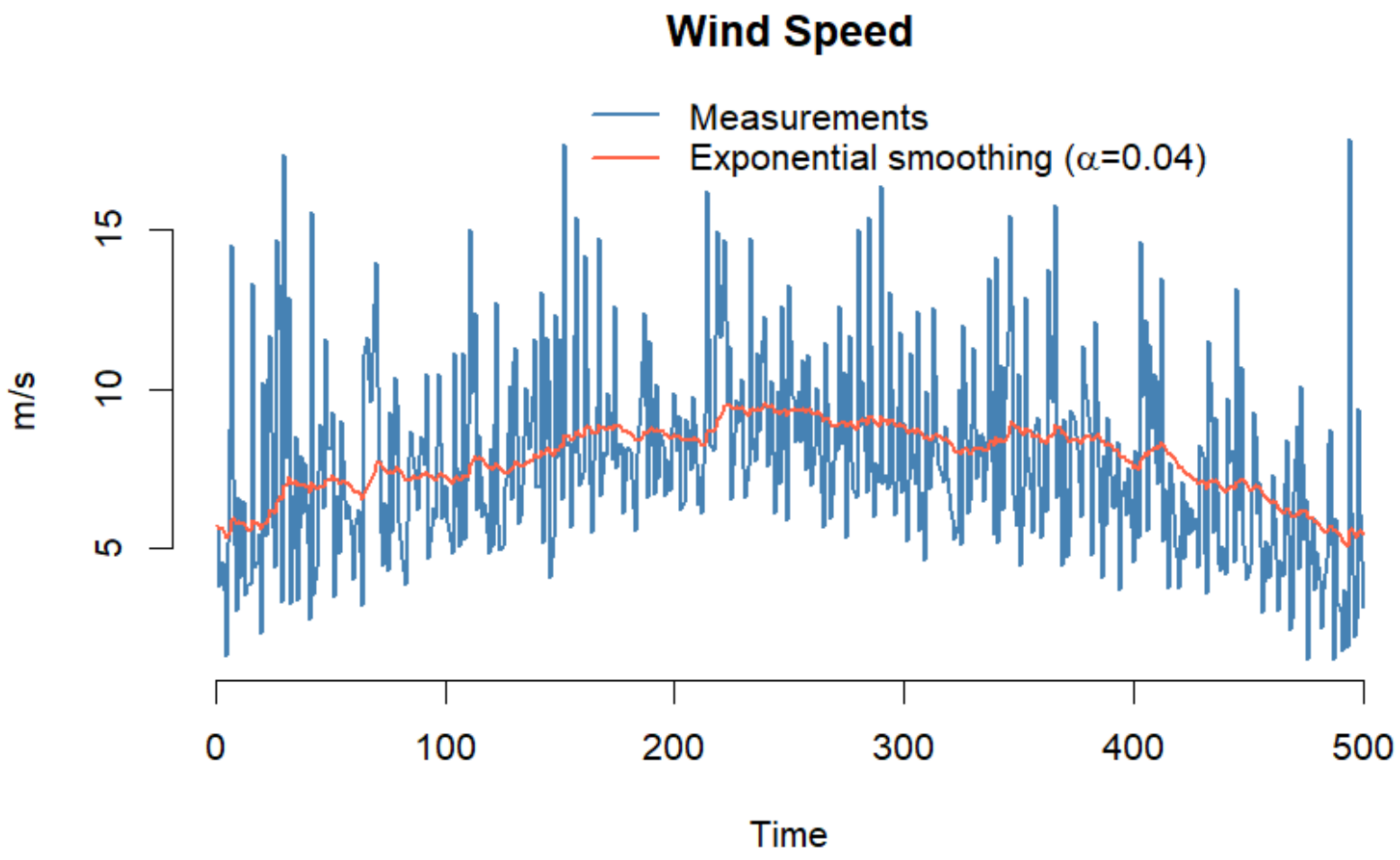
Smoothing



Let us consider a time series which does not have very strong trend

But there are large fluctuations!

Smoothing



Here is an example of *smoothing* the data

Smoothing

- What is smoothing?
- What is the underlying assumption when doing smoothing?
- How can smoothing be used for prediction?

Exponential Smoothing

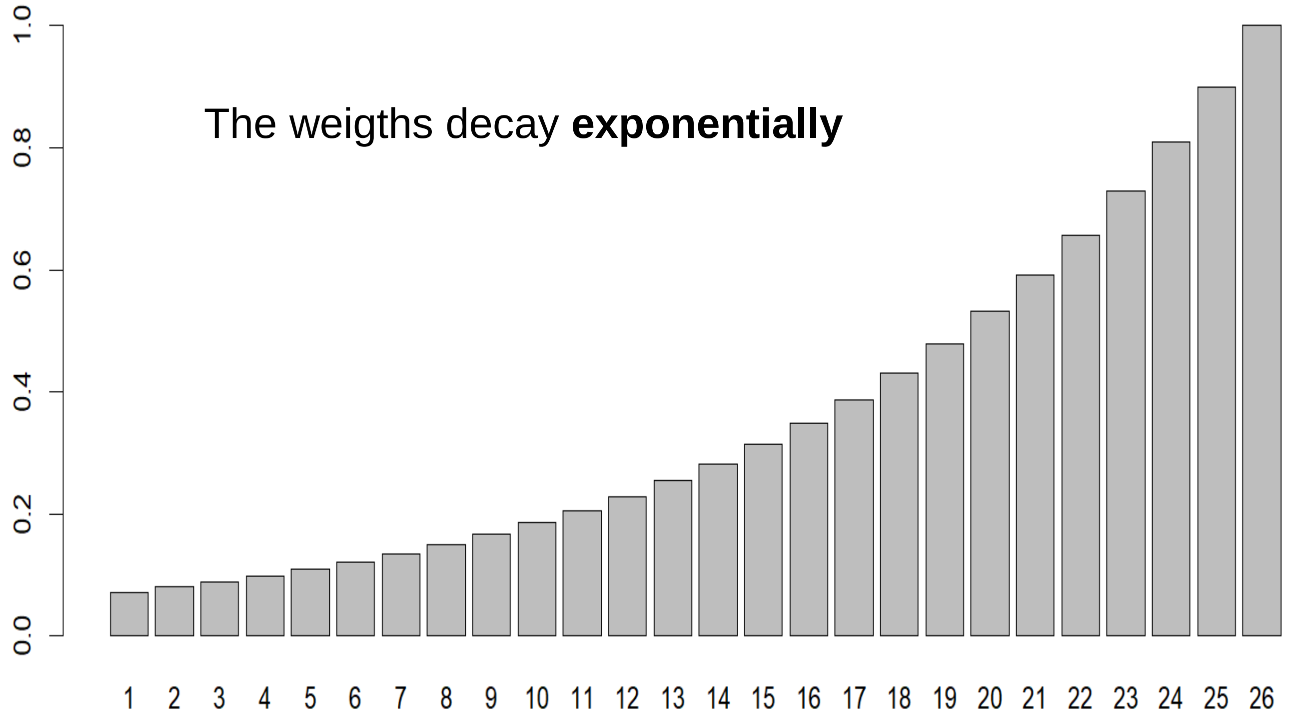
Given a forgetting factor $\lambda \in]0;1[$

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

Exponential Smoothing

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

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



Exponential Smoothing

Given a forgetting 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 forgetting 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 forgetting 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)

For large N:

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

Recursive formulation

Definition (Simple Exponential Smoothing):

The sequence S_N defined by

$$S_N = (1-\lambda)\, Y_N + \lambda S_{N-1}$$

is called the *simple exponential smoothing* or first order exponential smoothing of the time series Y.

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

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

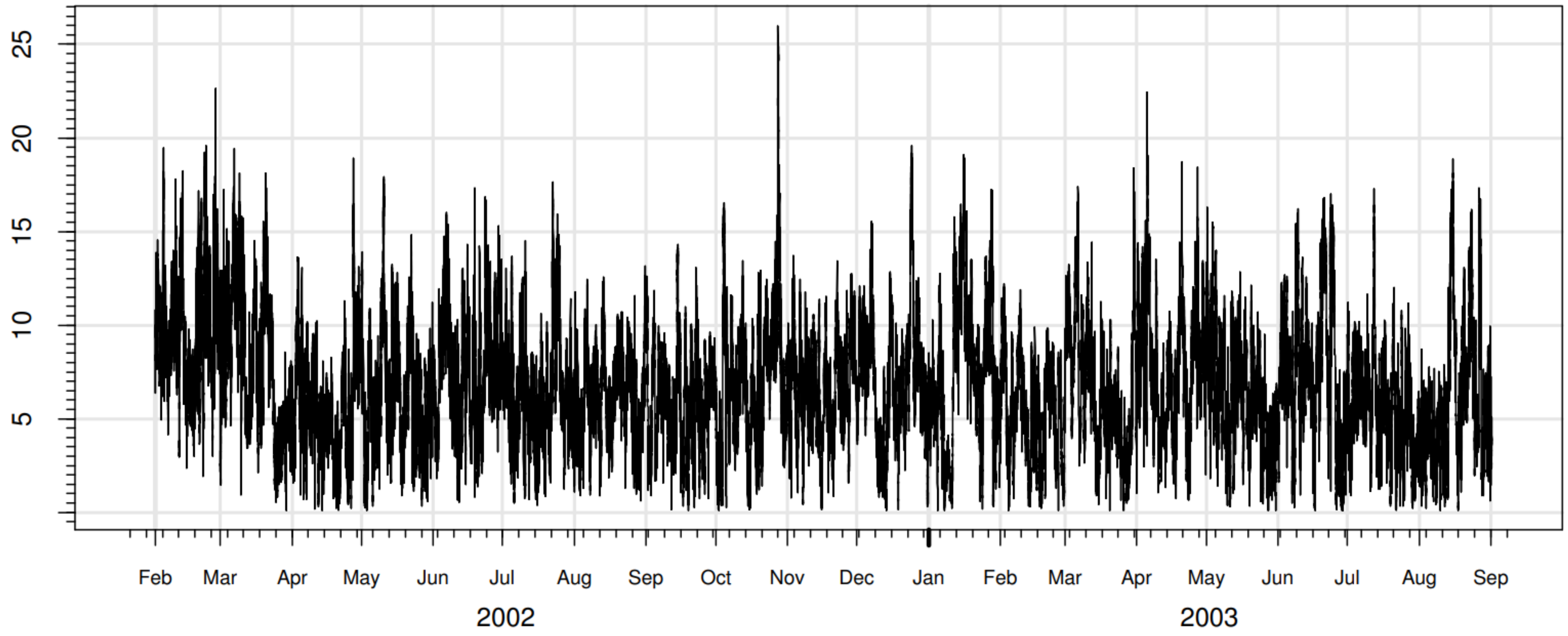
Sum of squared ℓ-step prediction errors
is used to find optimal lambda (or optimal alpha)

$$\alpha = 1 - \lambda$$

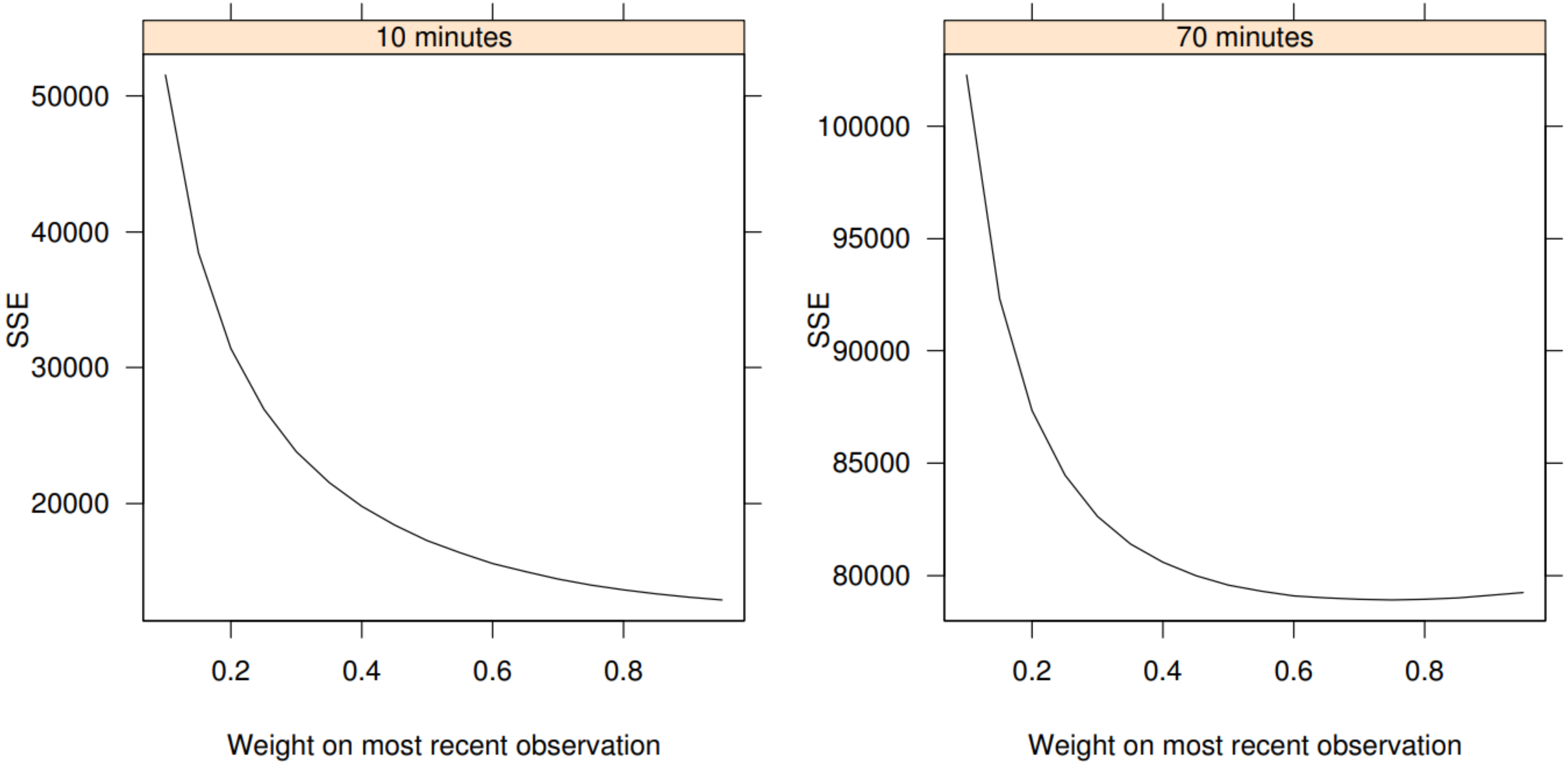
The value minimizing $S(\alpha)$ is chosen.

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



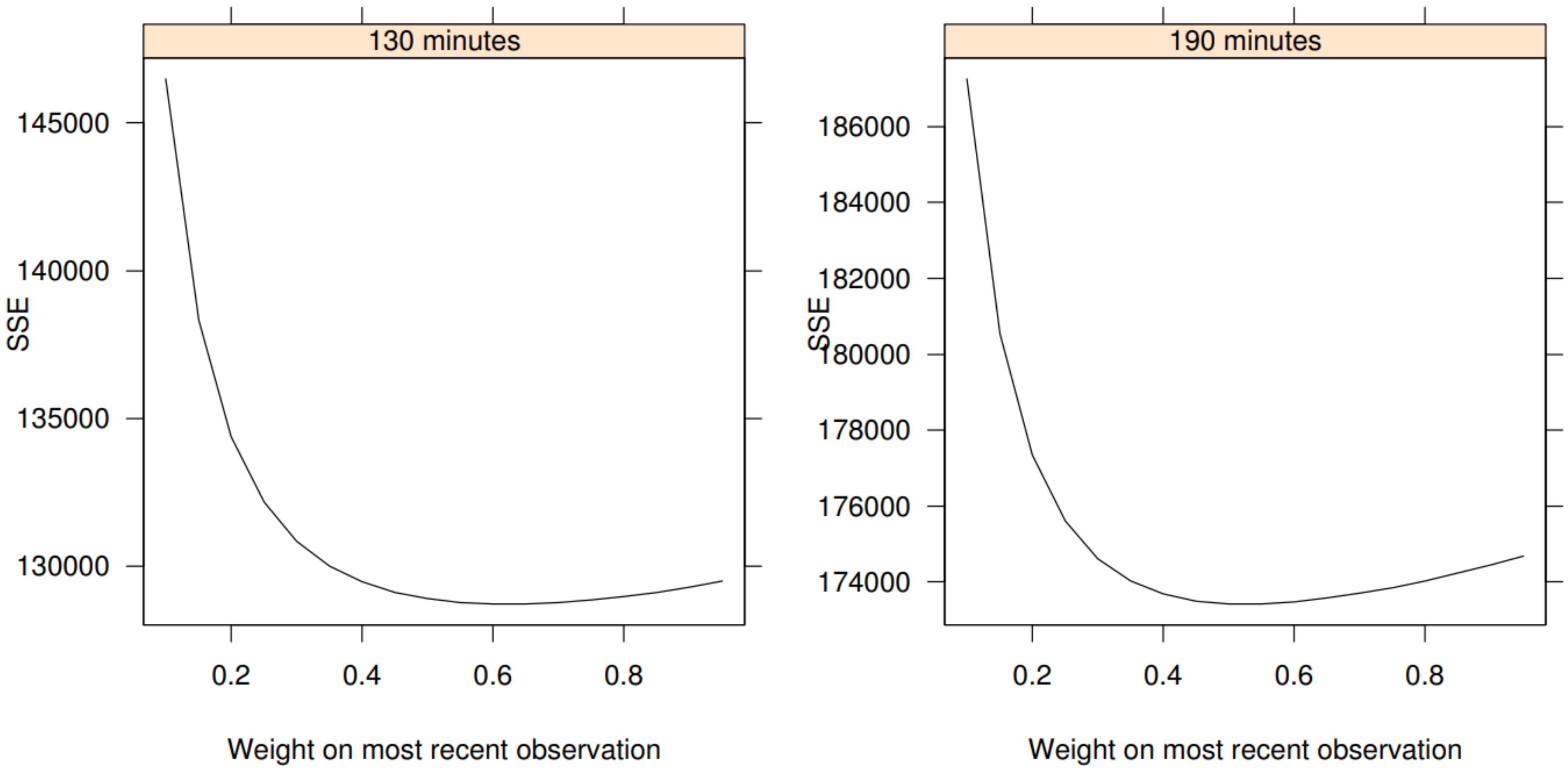
Example – wind speed 76m a.g.l. at Risø



- 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 $\alpha \approx 0.5$

$$\alpha = 1 - \lambda$$

Optimal value of $\alpha = 1 - \lambda$

- Minimize the sum of squared ℓ-step prediction errors
- Optimal values depend on ℓ = the ”prediction horizon”
- The same can be done for more complicated trend models

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 (from final year of observations)
  - 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 (from final year of observations)
  - 3 *individual* smoothing parameters

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

Next time

- Stochastic Processes (and ARIMA models) w. Peder ☺