Cross-Validation

James M. Flegal

Agenda

Suppose we have several different models for a particular data set. How should we choose the best one? Naturally, we would want to select the best performing model in order to choose the best. What is performance? How to estimate performance? Thinking about these ideas, we'll consider the following:

- Model assessment and selection
- Prediction error
- Cross-validation
- Smoothing example

Example: Lidar data

- Consider the `lidar` data from `SemiPar` package in R

```
library(SemiPar)
```

```
## Warning: package 'SemiPar' was built under R version 4.0.2
```

```
data(lidar)
```

- Randomly split data into K subsets
- Each observation gets a `foldid` between `1` and `K`

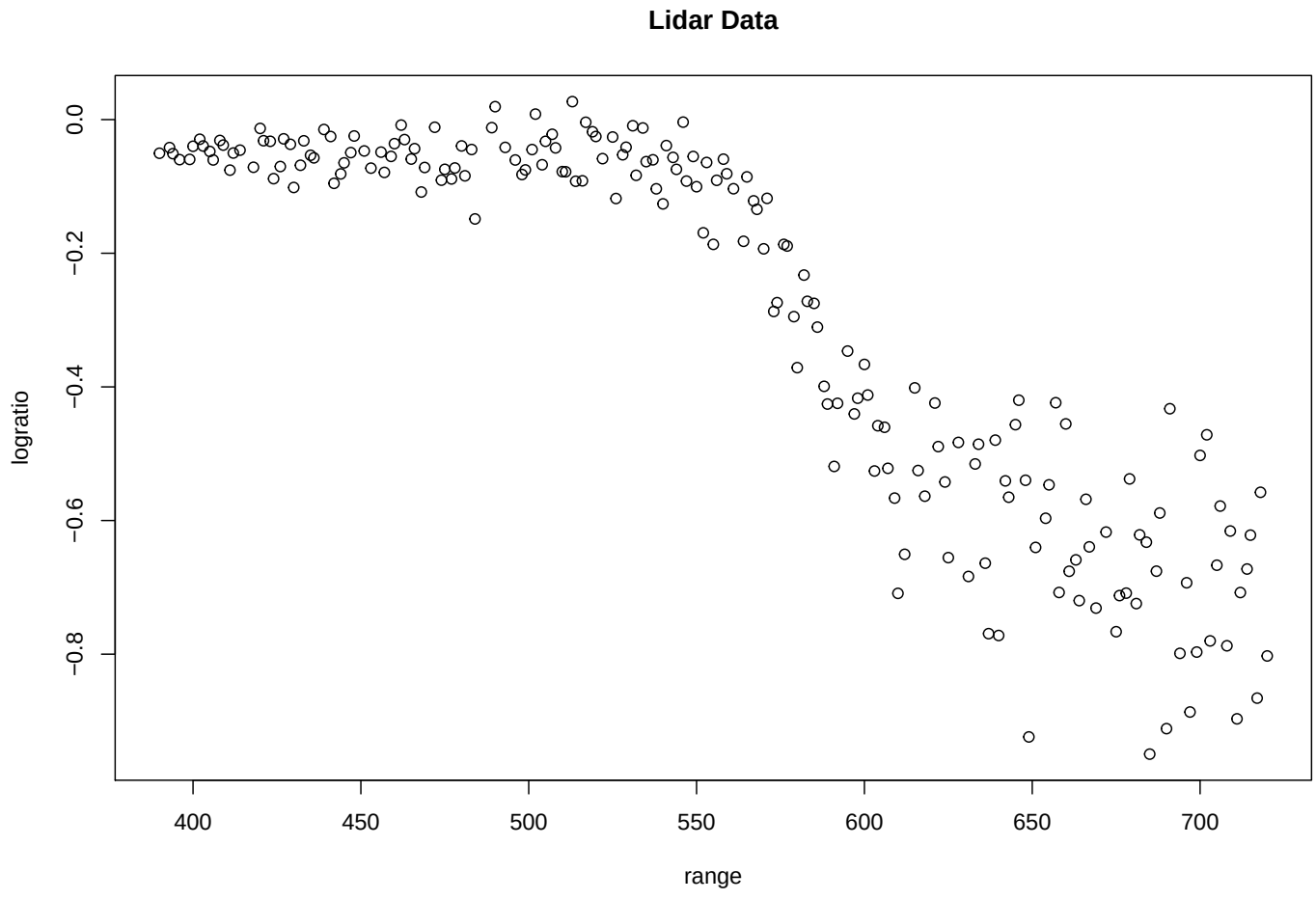
```
K <- 10
n <- nrow(lidar)
foldid <-sample(rep(1:K, length = n))
```

- Split data into training and test data

```
lidar.train <- subset(lidar, foldid != 1)
lidar.test <-  subset(lidar, foldid == 1)
attach(lidar.train)
```

Example: Lidar data

```
plot(range,logratio, main="Lidar Data")
```



loess (locally weighted smoothing)

- loess (locally weighted smoothing) is a popular tool that creates a smooth line through a timeplot or scatter plot to help you to see relationship between variables and foresee trends
- loess is typically to
  - Fitting a line to a scatter plot or time plot where noisy data values, sparse data points or weak interrelationships interfere with your ability to see a line of best fit
  - Linear regression where least squares fitting doesn't create a line of good fit or is too labor-intensive to use
  - Data exploration and analysis

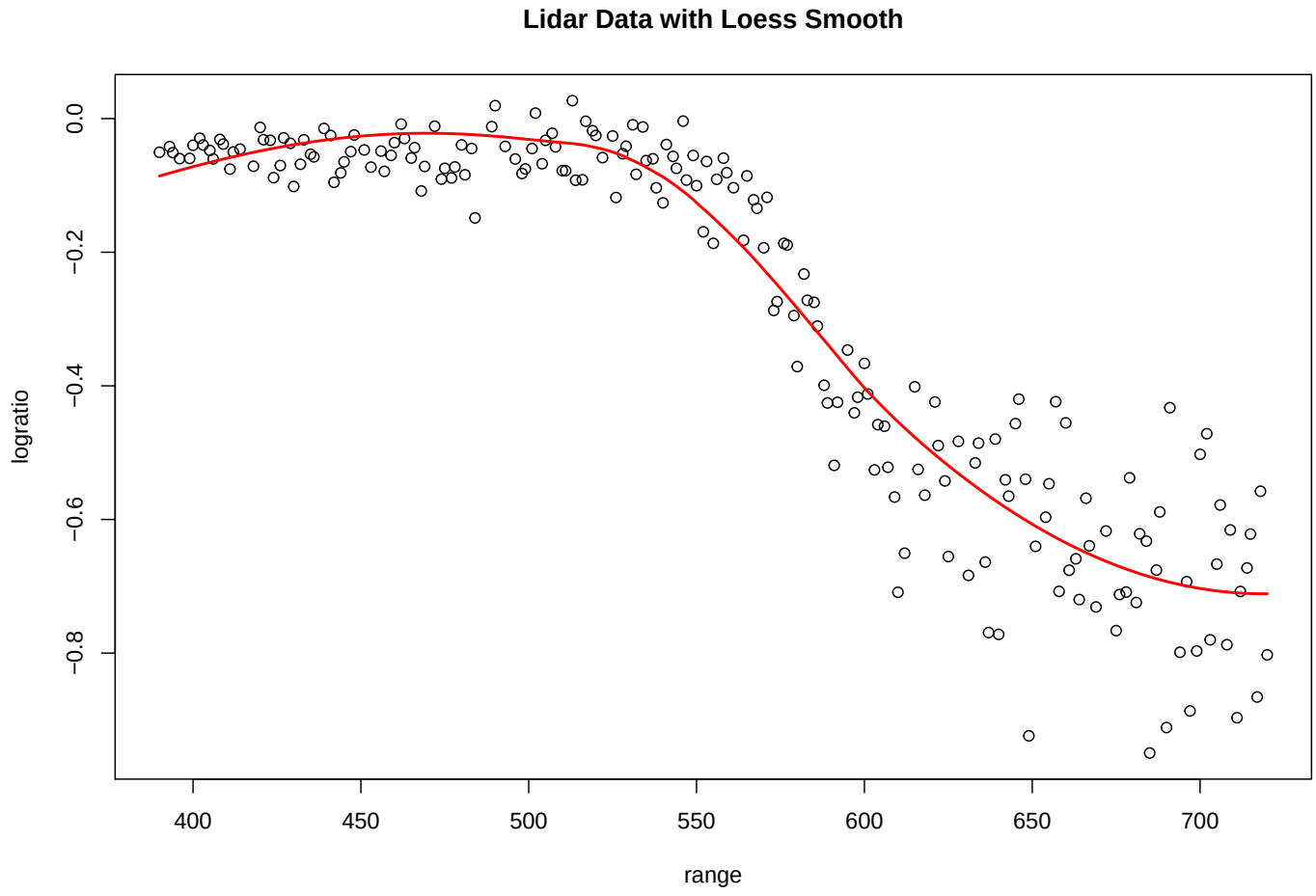
Nonparametric smoothing

- Benefits
  - Provides a flexible approach to representing data
  - Ease of use
  - Computations are relatively easy (sometimes)
- Disadvantages
  - No simple equation for a set of data
  - Less understood than parametric smoothers
  - Depends on a `span` parameter controlling the smoothness

Example: Lidar data

- Can consider fitting a `loess` smooth to the data

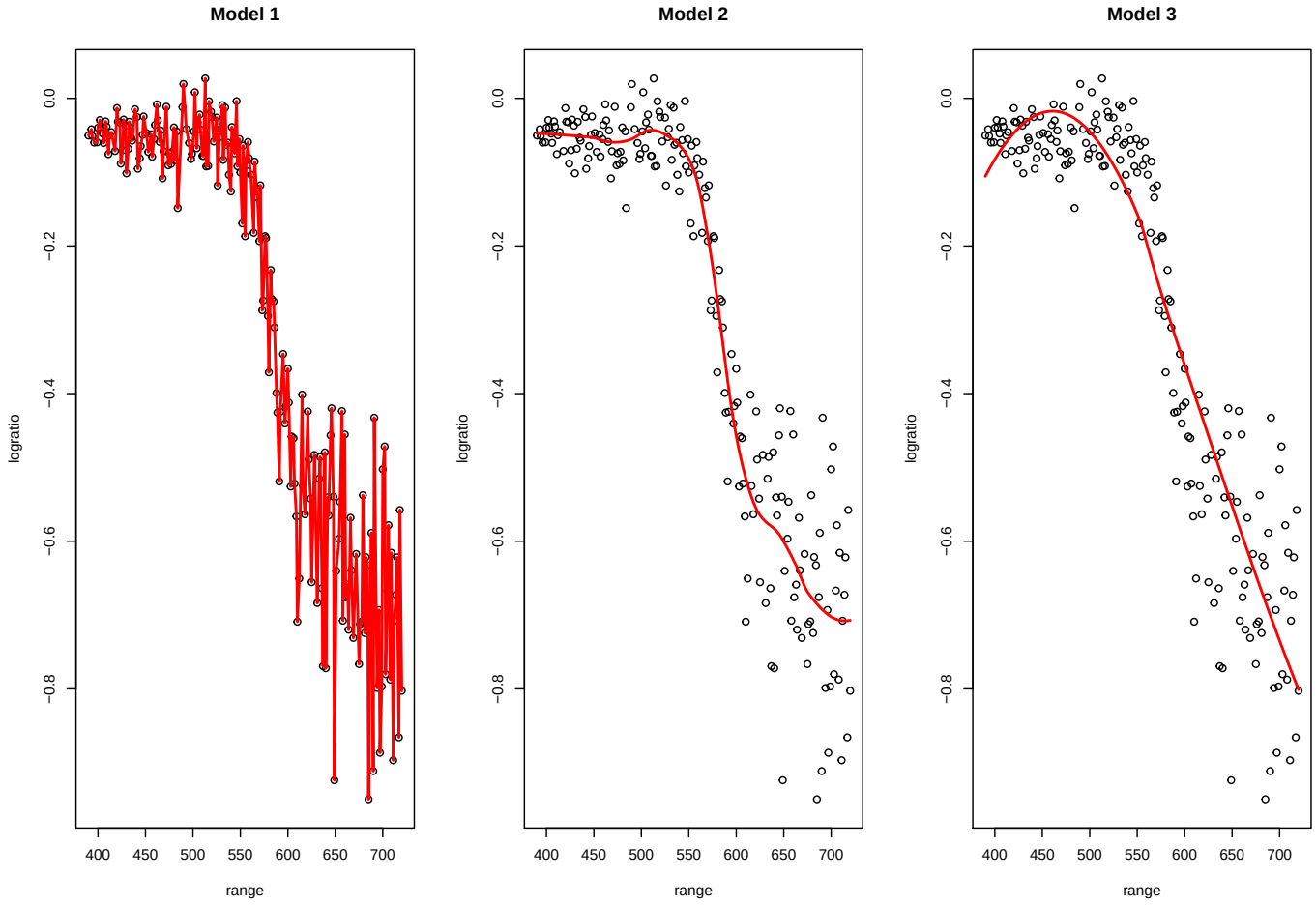
Example: Lidar data



Example: Lidar data

- How to choose the `span` parameter?
- Which model is the best? Why?

Example: Lidar data



Example: Lidar data

- Model 1: $\sum_i \left| Y_i - \hat{f}(X_i) \right|^2 = 0$
- Model 2: $\sum_i \left| Y_i - \hat{f}(X_i) \right|^2 = 1.03$
- Model 3: $\sum_i \left| Y_i - \hat{f}(X_i) \right|^2 = 1.71$

Therefore, Model 1 has the smallest squared error over the data

```
c(sum((logratio - obj1$fitted)^2), sum((logratio - obj2$fitted)^2),
  sum((logratio - obj3$fitted)^2))
```

```
## [1] 1.382878e-30 1.216730e+00 1.827694e+00
```

```
c(mean((logratio - obj1$fitted)^2), mean((logratio - obj2$fitted)^2),
  mean((logratio - obj3$fitted)^2))
```

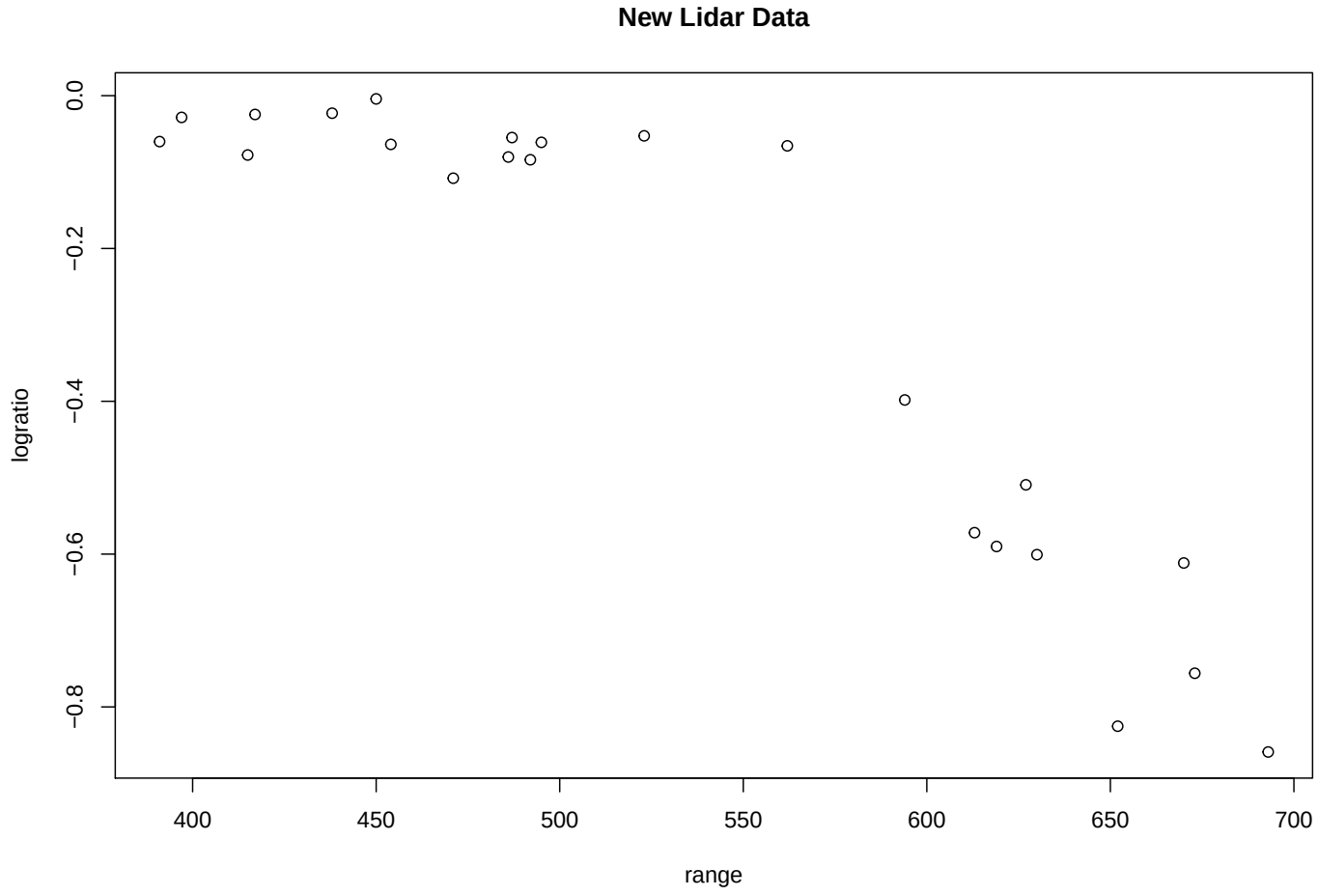
```
## [1] 6.984233e-33 6.145102e-03 9.230777e-03
```

Example: Lidar data

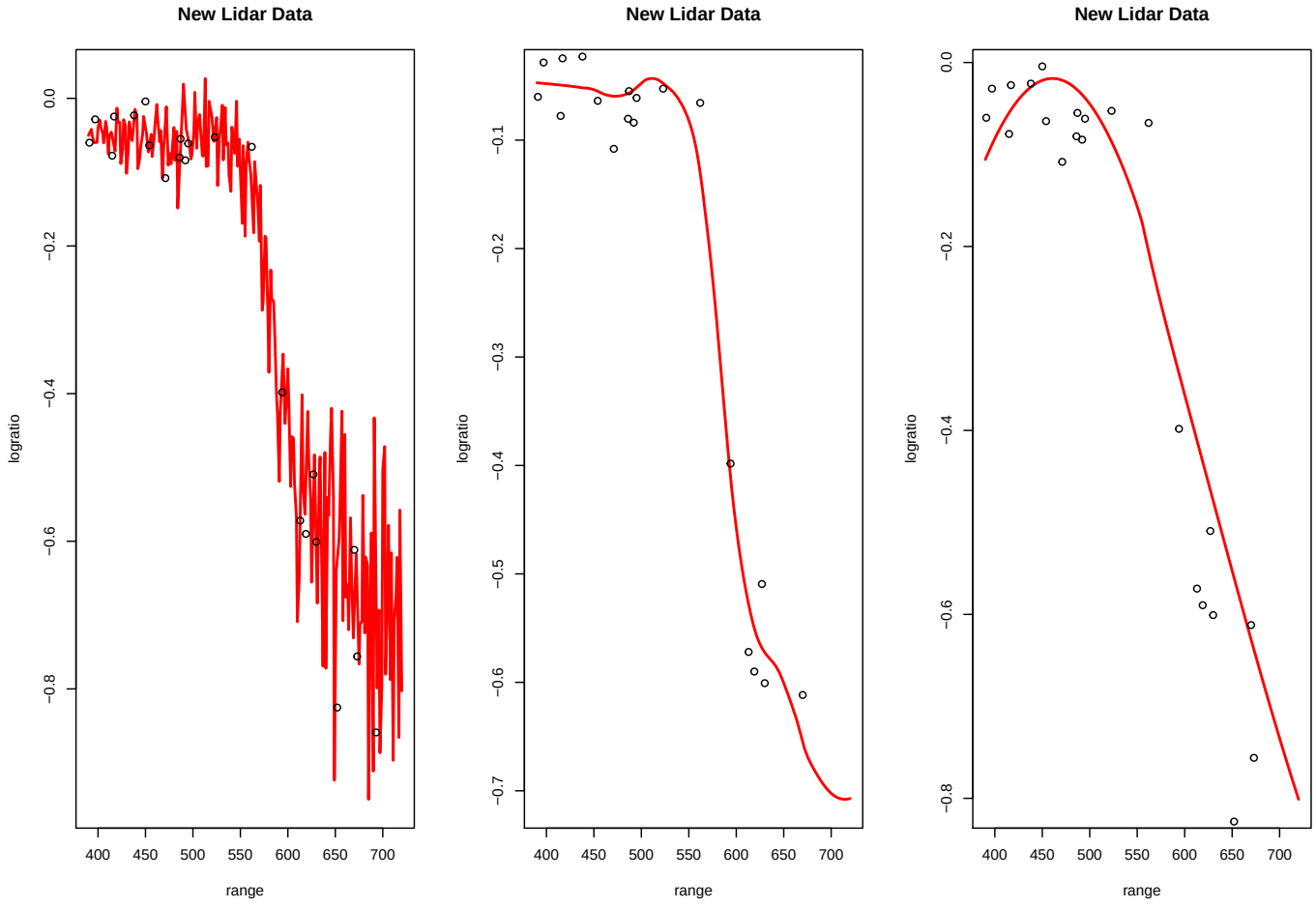
- Suppose now that we have new data from a second experiment, `lidar.test`
- How well do my three models predict the new data? Consider again $\sum_i \left| Y_i^{(new)} - \hat{f}(X_i^{(new)}) \right|^2$

Example: Lidar data

```
plot(lidar.test$range,lidar.test$logratio, xlab="range",
     ylab="logratio", main="New Lidar Data")
```



Example: Lidar data



Example: Lidar data

```
y.hat <- predict(obj1, newdata=lidar.test)
sum((lidar.test$logratio - y.hat)^2)
mean((lidar.test$logratio - y.hat)^2)

y.hat <- predict(obj2, newdata=lidar.test)
sum((lidar.test$logratio - y.hat)^2)
mean((lidar.test$logratio - y.hat)^2)

y.hat <- predict(obj3, newdata=lidar.test)
sum((lidar.test$logratio - y.hat)^2)
mean((lidar.test$logratio - y.hat)^2)
```

Example: Lidar data

- Model 1: $\sum_i \left| Y_i^{(new)} - \hat{f}(X_i^{(new)}) \right|^2 = 1.23$
- Model 2: $\sum_i \left| Y_i^{(new)} - \hat{f}(X_i^{(new)}) \right|^2 = 0.17$
- Model 3: $\sum_i \left| Y_i^{(new)} - \hat{f}(X_i^{(new)}) \right|^2 = 0.24$

Now notice Model 2 has the smallest squared error over the **new** data

Example: Lidar data

We can also consider average squared error since the original and new data contain a different number of points. That is,

$$\frac{1}{n}\sum_{i=1}^{n}\left|Y_i - \hat{f}(X_i)\right|^2 ,$$

computed using (X, Y) belonging to either the original or new data.

ASE on	Model 1	Model 2	Model 3
Fitting Data	0	0.0058	0.0091
New Data	0.0536	0.0075	0.0102

Model Assessment and Selection

Basic problem: We have several different models so how can we choose the best one? We could estimate the performance of each model in order to choose the best one. What is performance? How to estimate performance?

- **Model assessment**: Evaluating how closely a particular model fits the data
- **Model selection**: Choosing the best model among several different ones

Model Assessment and Selection

Model selection is ubiquitous. Where do we use model selection?

- Linear regression (variable selection)
- Smoothing (smoothing parameter selection)
- Kernel density estimation (bandwidth selection)
- ...

Prediction Error

- Prediction error is one measure of performance of a model
- In short, we're interested in how well the model make predictions about new data?
- Exact definition (e.g. squared error or other) depends on problem

Prediction Error

- Consider either a nonparametric (smoothing) or linear regression modeling setting
- In this case, the regression equation is fit using observed data

$$(X_1, Y_1), \ldots, (X_n, Y_n) \ ,$$

  resulting in a regression function $\hat{f}_n(\cdot)$
- The resulting model can be used for prediction at new X values resulting in $\hat{f}_n(X_{n+1})$
- One measure of prediction error for regression is the Mean Squared Prediction Error (MSPE), i.e.

$$\mathrm{MSPE} = \mathrm{E}\left|Y_{n+1} - \hat{f}_n(X_{n+1})\right|^2$$

Prediction Error

Question: How do we estimate prediction error if we only have n observations?

Answer: If we had m additional independent observations $(X_{n+1}, Y_{n+1}), \ldots, (X_{n+m}, Y_{n+m})$, then we could estimate the MSPE by

$$\frac{1}{m} \sum_{i=1}^{m} \left| Y_{n+i} - \hat{f}_n(X_{n+i}) \right|^2$$

Prediction Error

- Notice, if we **reuse** the same data to estimate MPSE we have an in-sample estimate of the MPSE

$$\frac{1}{n}\sum_{i=1}^{n}\left|Y_i - \hat{f}_n(X_i)\right|^2 .$$

- But, as we saw in the `lidar` example this is generally a bad (overly optimistic) estimate of the MSPE. Using the same data for fitting and estimating prediction error generally leads to bad estimates of prediction error that are overly optimistic. This is related to the phenomenon known as **overfitting**.
- How do we estimate prediction error if we only have n observations?

Cross-Validation

- Cross-validation is a method for estimating prediction error
- Main idea is to split the data into two parts
  - **training** portion used for fitting the model
  - **test** portion for validating the model or estimating the prediction error

K-fold Cross-Validation

1. Split the data into K roughly equal-sized parts
2. For the kth part
   - Fit the model using the remaining $K - 1$ parts
   - Calculate the prediction error of the fitted model when predicting for the kth part of the data

Example: $K = 5$

Fold 1	Validation	Train	Train	Train	Train
Fold 2	Train	Validation	Train	Train	Train
Fold 3	Train	Train	Validation	Train	Train
Fold 4	Train	Train	Train	Validation	Train
Fold 5	Train	Train	Train	Train	Validation

Leave-one-out cross validation

- A special case of cross-validation is known as **leave-one-out cross validation**
- Simply K-fold cross-validation where $K = n$

1. For the kth observation
   - Fit the model using the remaining $k - 1$ observations
   - Calculate the prediction error of the fitted model when predicting for the kth observation

Pseudo-code

The following is some pseudo-code for K-fold cross-validation.

```
K <- 10
n <- nrow(mydata)
cv.error <- vector(length = K)

# Randomly split data into K subsets
# - each observation gets a foldid  between 1 and K
foldid <-sample(rep(1:K, length = n))

# Repeat K times
for(i in 1:K) {
  # Fit using training set
  f.hat <- estimator(mydata[foldid != i, ])

  # Calculate prediction error on validation set
  cv.error[i] <- calc_error(f.hat, mydata[foldid == i, ])
}

cv.error.estimate <- mean(cv.error)
```

Example: Lidar Data

Using the `lidar` data again, we will

1. Fit several different `loess` smooths corresponding to different choices of bandwidth (span)
2. Estimate the mean squared prediction error of each smooth
3. Choose the smooth with the smallest estimated MSPE

```
library(ggplot2)
```

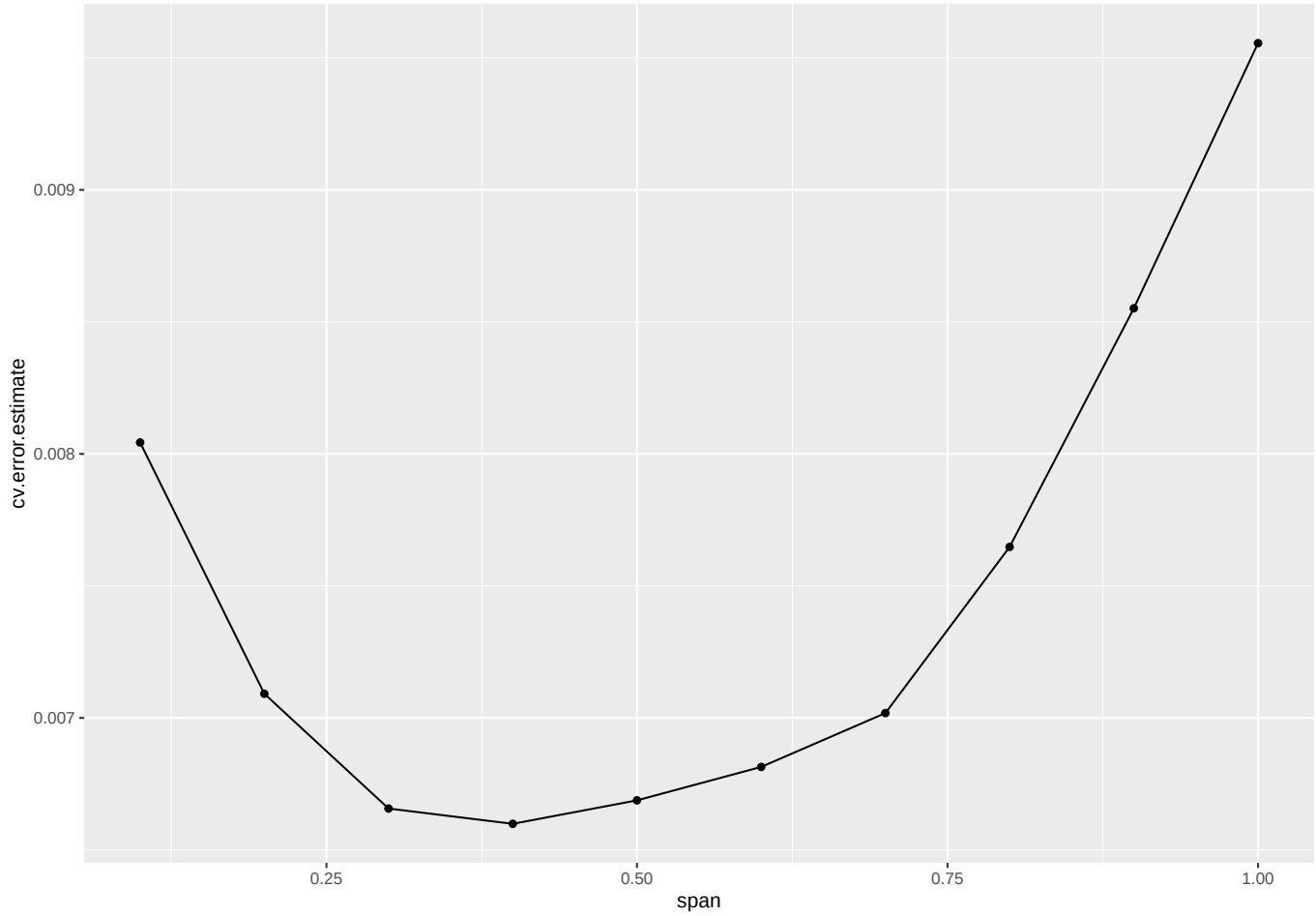
```
## Warning: package 'ggplot2' was built under R version 4.0.2
```

```
library(SemiPar)
data(lidar)
s = seq(from = 0.1, to = 1.0, by = 0.1)
K <- 10
n <- nrow(lidar)
```

Example: Lidar Data

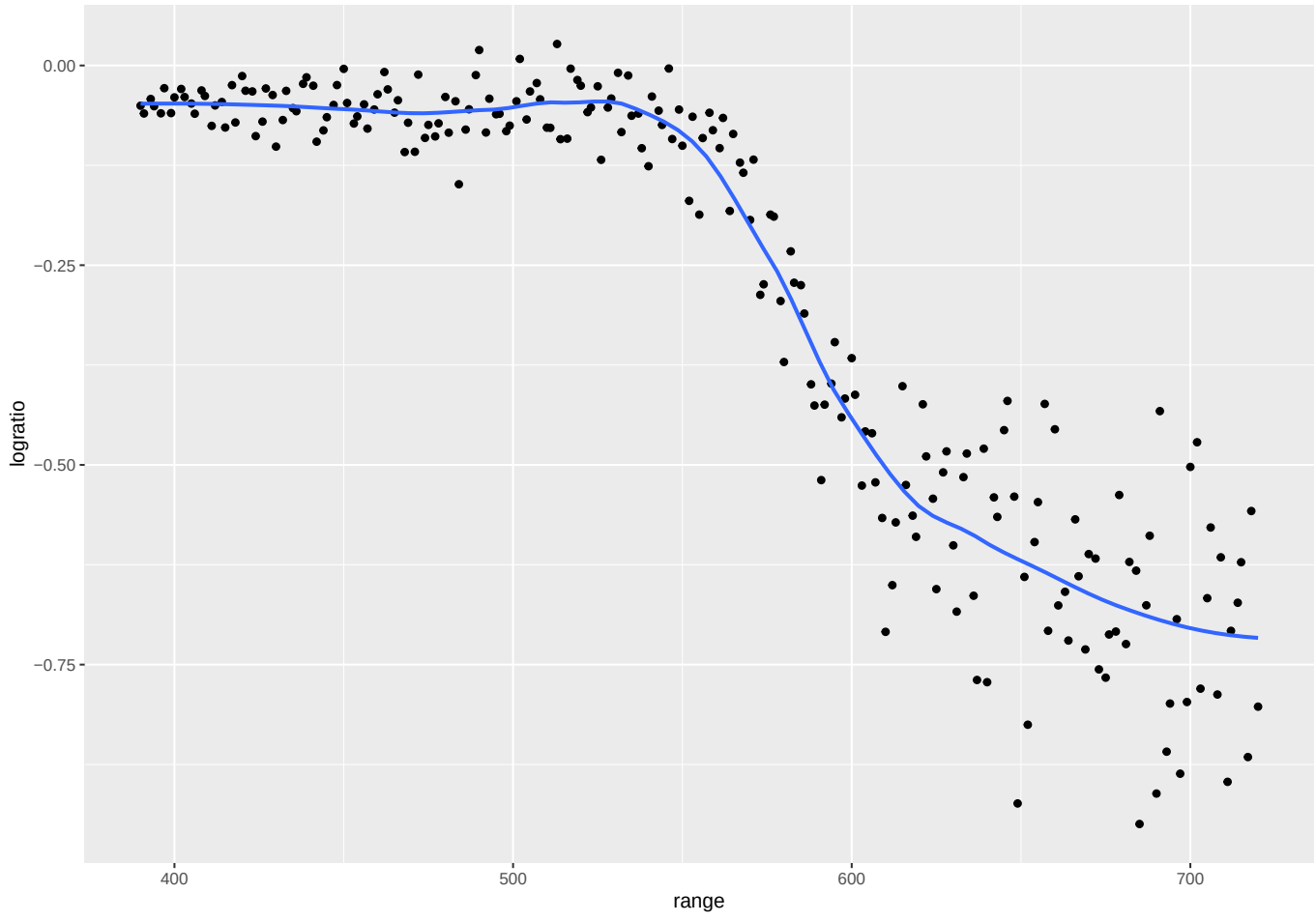
```
cv.error <- matrix(nrow = K, ncol = length(s))
foldid <-sample(rep(1:K, length = n))

for(i in 1:K) {
  # Fit, predict, and calculate error for each bandwidth
  cv.error[i, ] <- sapply(s, function(span) {
    # Fit LOESS model to training set
    obj <- loess(logratio ~ range,
                 data = subset(lidar, foldid != i),
                 span = span,
                 control = loess.control(surface = 'direct'))
    # Predict and calculate error on the validation set
    y.hat <- predict(obj, newdata = subset(lidar, foldid == i))
    pse <- mean((subset(lidar, foldid == i)$logratio - y.hat)^2)
    return(pse)
})
}

# Columns of cv.error correspond to different bandwidths
cv.error.estimate <- colMeans(cv.error)
```

Cross-validation estimates of MSPE



Smoother selected by cross-validation

```
## `geom_smooth()` using formula 'y ~ x'
```



Summary

- Can assess models by prediction error
- Select model with smallest prediction error
- Using same data for fitting and estimating prediction error leads to overfitting
- Cross-validation is a method for estimating prediction error that avoids overfitting