Statistical Learning

Lecture 04a

ANU - RSFAS

Last Updated: Thu Mar 17 17:30:59 2022

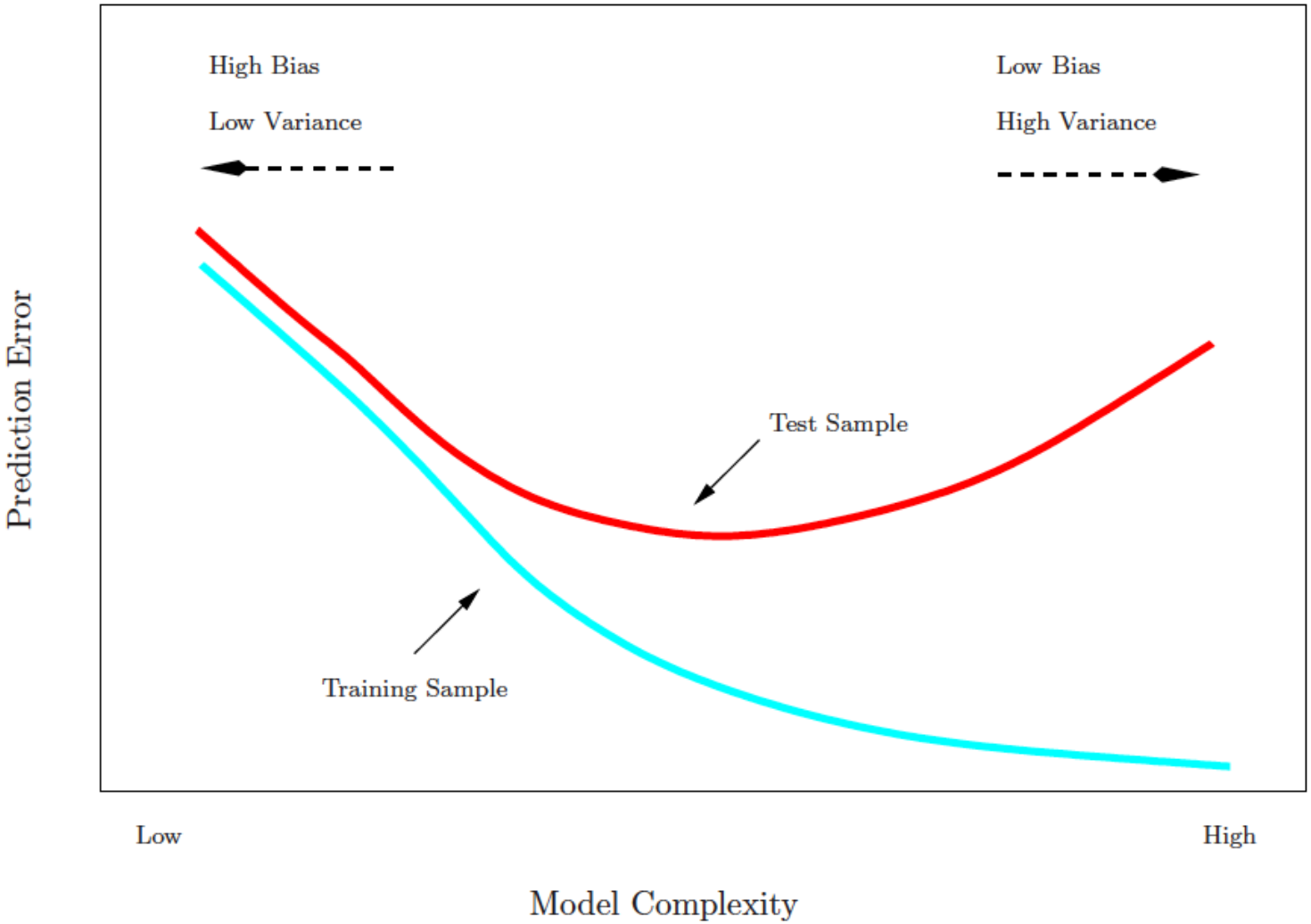
Cross-validation and the Bootstrap

- In the section we discuss two resampling methods: cross-validation and the bootstrap.
- These methods refit a model of interest to samples formed from the training set, in order to obtain additional information about the fitted model.
- For example, they provide estimates of test-set prediction error, and the standard deviation and bias of our parameter estimates.

Training Error versus Test error

- Recall the distinction between the test error and the training error:
- The test error is the average error that results from using a statistical learning method to predict the response on a new observation, one that was not used in training the method.
- In contrast, the training error can be easily calculated by applying the statistical learning method to the observations used in its training.
- But the training error rate often is quite different from the test error rate, and in particular the former can dramatically underestimate the latter.

Training versus Test-Set Performance



More on Prediction-Error Estimates

- Best solution: a large designated test set. Often not available.
- Some methods make a mathematical adjustment to the training error rate in order to estimate the test error rate. These include the Cp statistic, AIC and BIC. They were discussed in your regression course - we may review them.
- Here we instead consider a class of methods that estimate the test error by holding out a subset of the training observations from the fitting process, and then applying the statistical learning method to those held out observations.

Validation-set Approach

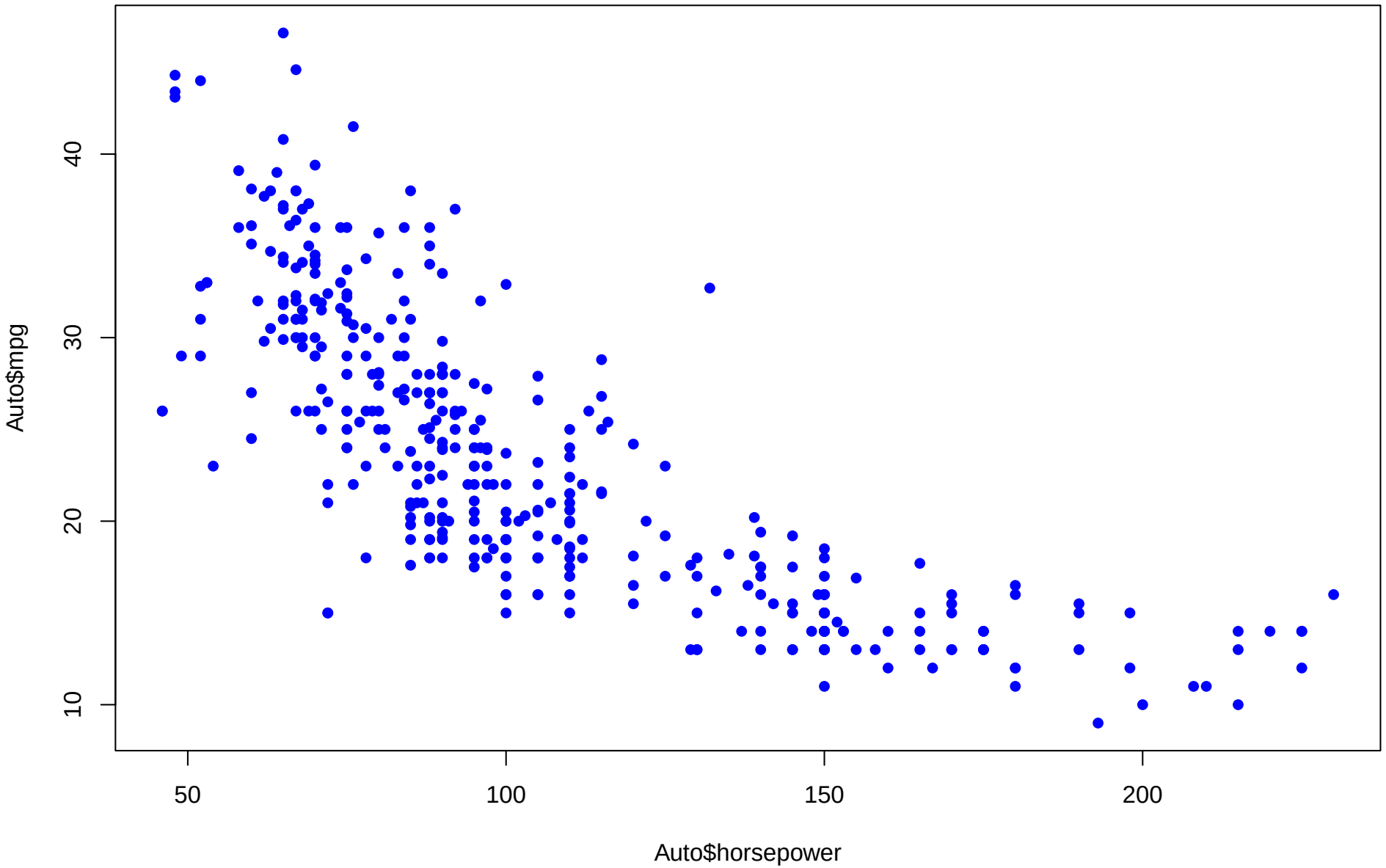
- Here we randomly divide the available set of samples into two parts: a training set and a validation or hold-out set.
- The model is fit on the training set, and the fitted model is used to predict the responses for the observations in the validation set.
- The resulting validation-set error provides an estimate of the test error. This is typically assessed using MSE in the case of a quantitative response and misclassification rate in the case of a qualitative (discrete) response.

Example: Automobile Data

```
library(ISLR)
data(Auto)
head(Auto)
```

```
##   mpg cylinders displacement horsepower weight acceleration year origin
## 1  18         8          307        130   3504         12.0   70      1
## 2  15         8          350        165   3693         11.5   70      1
## 3  18         8          318        150   3436         11.0   70      1
## 4  16         8          304        150   3433         12.0   70      1
## 5  17         8          302        140   3449         10.5   70      1
## 6  15         8          429        198   4341         10.0   70      1
##                       name
## 1 chevrolet chevelle malibu
## 2         buick skylark 320
## 3        plymouth satellite
## 4             amc rebel sst
## 5               ford torino
## 6          ford galaxie 500
```

```
plot(Auto$horsepower, Auto$mpg, pch=16, col="blue")
```



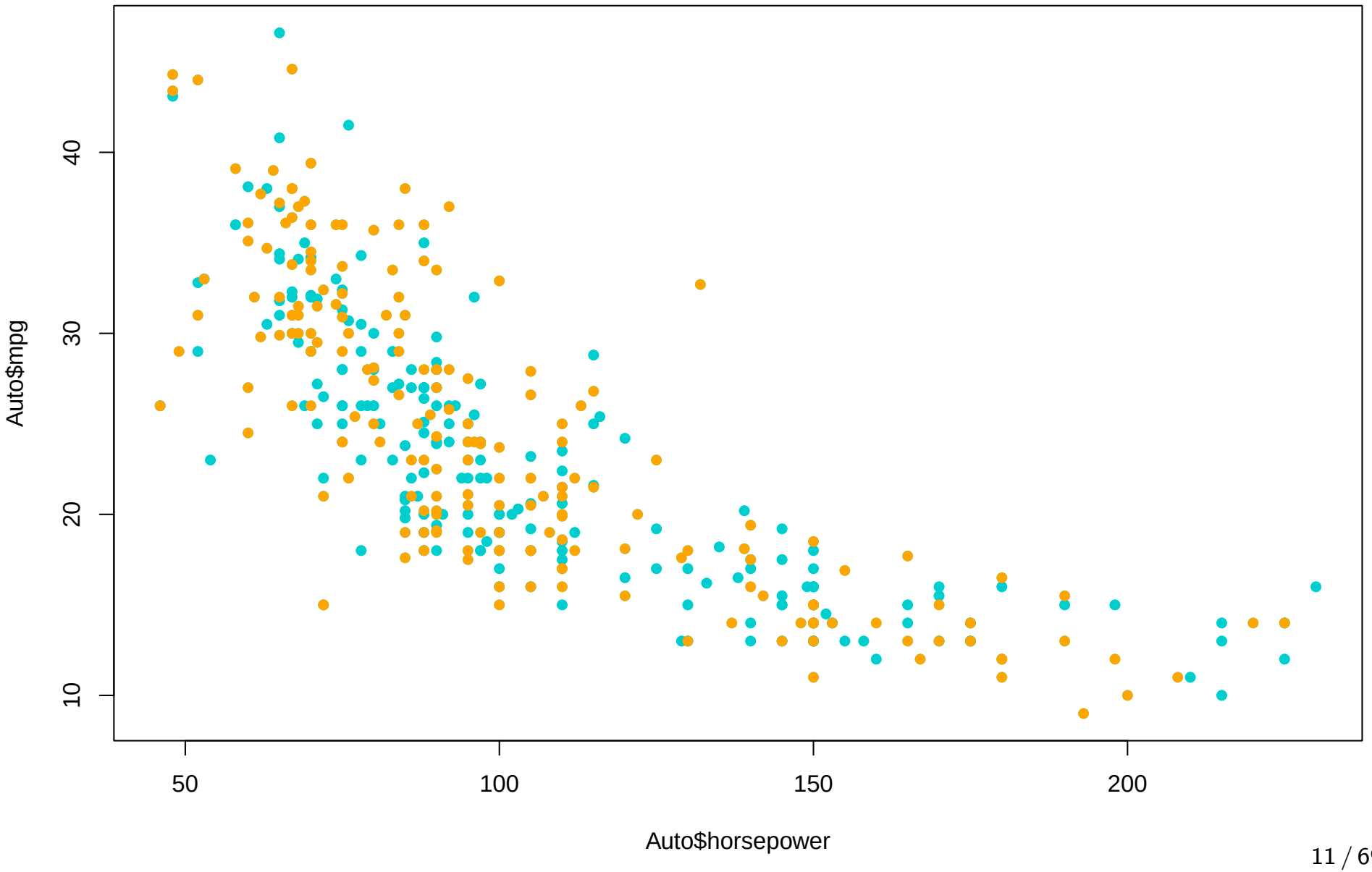
- Want to compare linear vs higher-order polynomial terms in a linear regression

$$\mathrm{mpg} = \beta_0 + \beta_1\ \mathrm{hp} + \beta_2\ \mathrm{hp}^2 + \cdots + \beta_{\boldsymbol{p}}\ \mathrm{hp}^{\boldsymbol{p}} + \epsilon$$

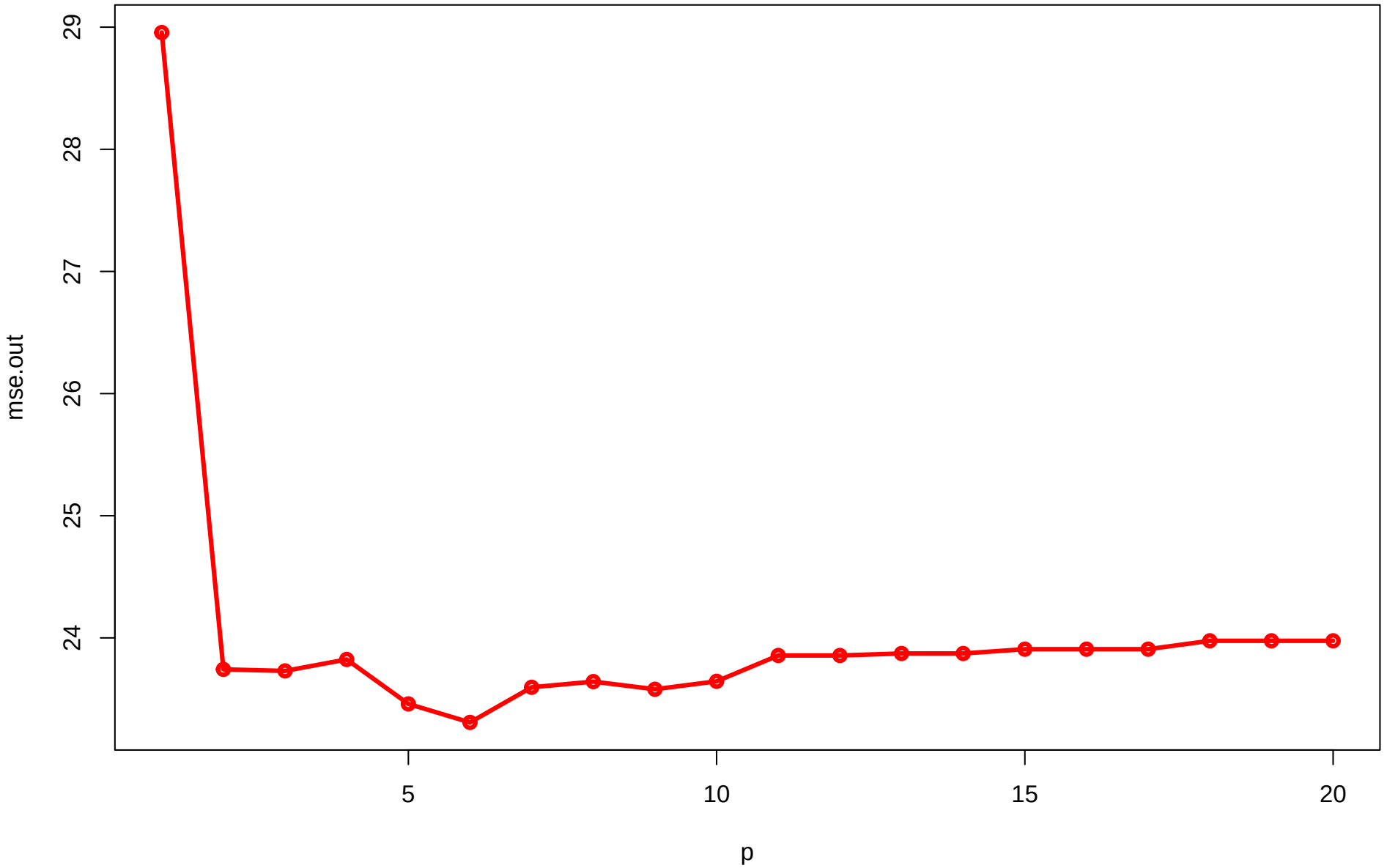
- We randomly split the 392 observations into two sets, a training set containing 196 of the data points, and a validation set containing the remaining 196 observations.

```
set.seed(1001)
samp <- sort(sample(1:392, 196))
train <- Auto[samp,]
test <- Auto[-samp,]
```

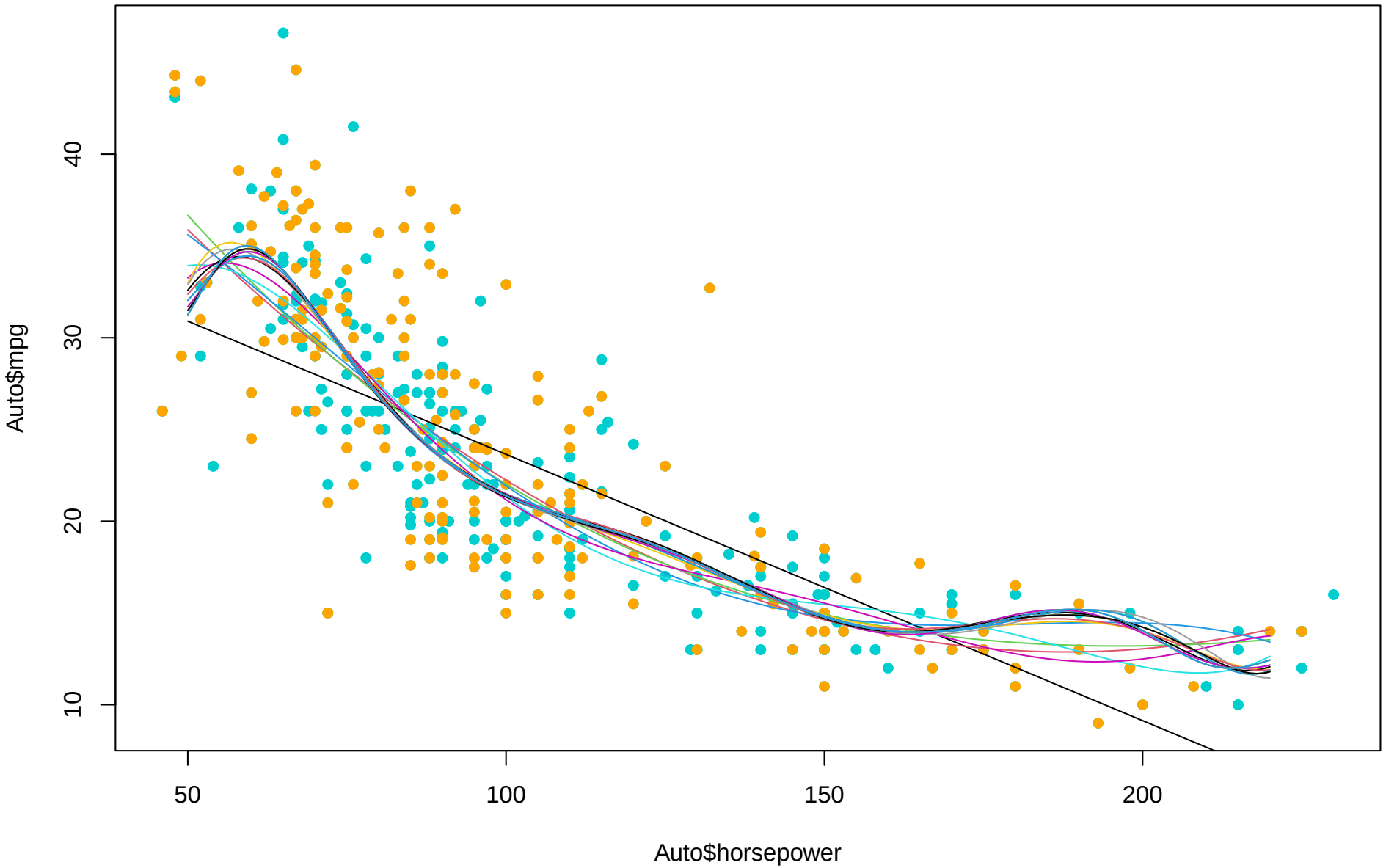
```r
plot(Auto$horsepower, Auto$mpg, pch=16, col="cyan")
points(train$horsepower, train$mpg, col="cyan3", pch=16)
points(test$horsepower, test$mpg, col="orange", pch=16)
```

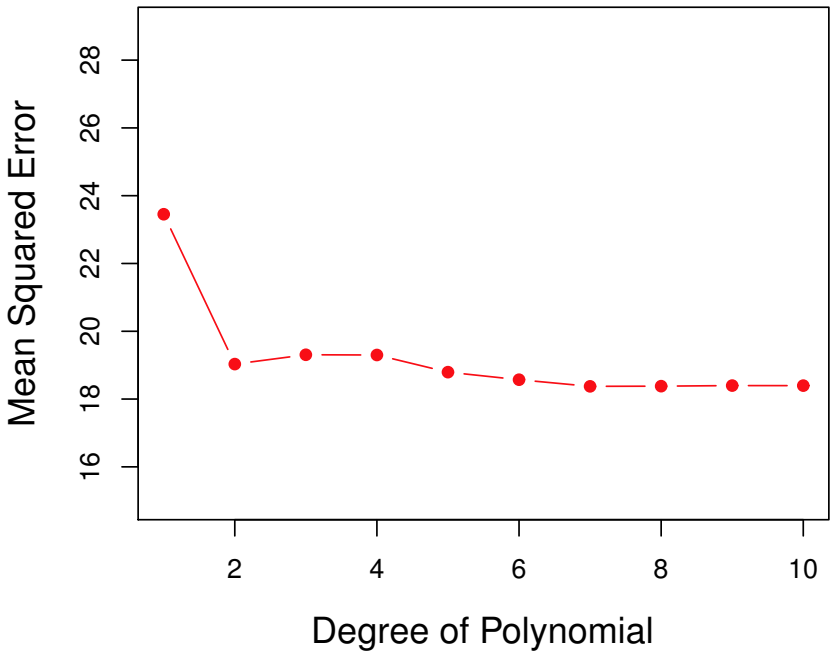


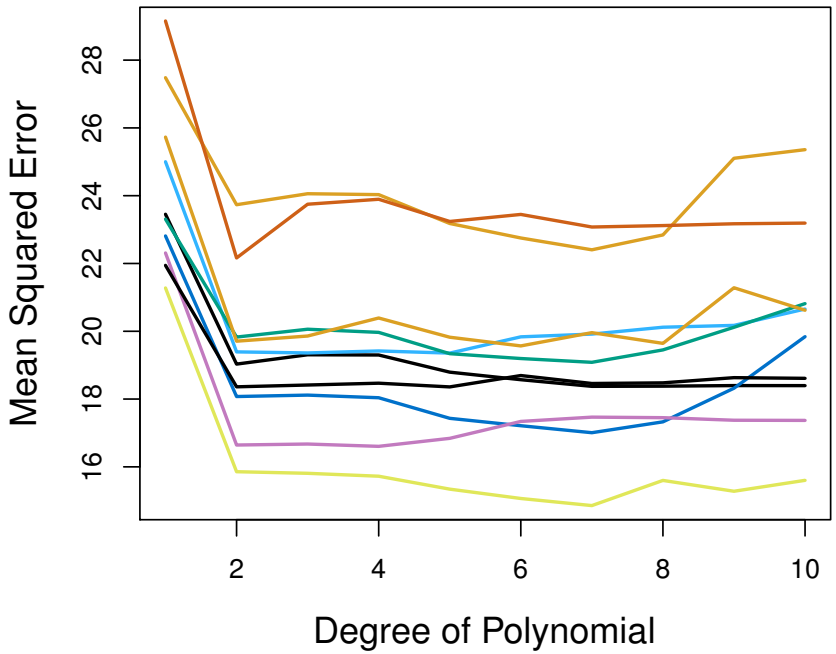
```r
mse.out <- rep(0, 20)

for(p in 1:20){
  mod.train <- lm(mpg ~ poly(horsepower, degree=p,
                             raw=TRUE), data=train)
  pred.test <- predict(mod.train, newdata=test)

  mse.out[p] <- mean( (test$mpg - pred.test)^2)
}

p <- 1:20
plot(p, mse.out, type="o", lwd=3, col="red")
```


mse.out
29
28
27
26
25
24
5
10
15
20
p

```r
plot(Auto$horsepower, Auto$mpg, pch=16, col="cyan")
points(train$horsepower, train$mpg, col="cyan3", pch=16)
points(test$horsepower, test$mpg, col="orange", pch=16)

for(p in 1:20){
  mod.train <- lm(mpg ~ poly(horsepower, degree=p, raw=TRUE), data=train)

  horsepower <- 50:220
  lines(horsepower,  predict(mod.train,
                             newdata=as.data.frame(horsepower)), col=p)
}
```


Auto$mpg
10
20
30
40
50
100
150
200
Auto$horsepower





- Repeat the process again - new random sample (left)
- And a few more times (right)

Drawbacks of the Validation Set Approach

- The validation estimate of the test error can be highly variable, depending on precisely which observations are included in the training set and which observations are included in the validation set.
- In the validation approach, only a subset of the observations – those that are included in the training set rather than in the validation set – are used to fit the model.
- This suggests that the validation set error may tend to overestimate the test error for the model fit on the entire data set.

Another Approach: K-fold Cross-Validation

- Widely used approach for estimating test error.
- Estimates can be used to select best model, and to give an idea of the test error of the final chosen model.
- Idea is to randomly divide the data into K equal-sized parts. We leave out part k, fit the model to the other $K - 1$ parts (combined), and then obtain predictions for the left-out kth part.
- This is done in turn for each part $k = 1, 2, \ldots, K$, and then the results are combined.

- Let the K parts be $C_1, C_2, \ldots, C_K$, where C_k denotes the indices of the observations in part k. There are n_k observations in part k: if N is a multiple of K, then $n_k = n/K$.

- Compute:

$$CV_{(K)} = \sum_{k=1}^{K} \frac{n_k}{n} MSE_k$$

where $MSE_k = \sum_{i \in C_k} (y_i - \hat{y}_i)^2 / n_k$, and $\hat{y}_i$ is the fit for observation i, obtained from the data with part k removed.

- So when we have have $n_k = n/K$ then:

$$CV_{(K)} = \sum_{k=1}^{K} \frac{1}{K} MSE_k$$

- Setting $K = n$ yields n-fold or leave-one out cross-validation (LOOCV).

A Special Case!

- With least-squares linear or polynomial regression, an amazing shortcut makes the cost of LOOCV the same as that of a single model fit! The following formula holds:

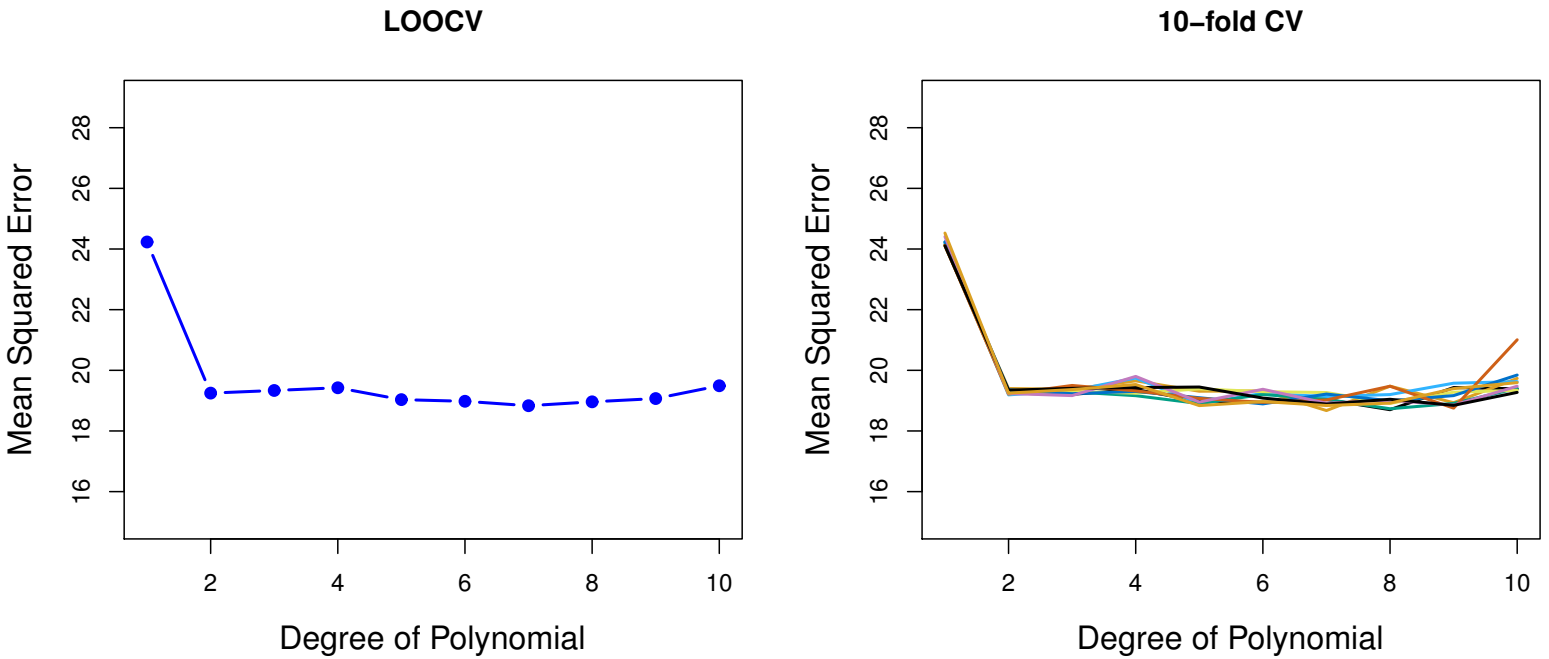
$$CV_{(n)} = \frac{1}{n} \sum_{i=i}^{n} \left(\frac{y_i - \hat{y}_i}{1 - h_i} \right)^2$$

- h_i is the i^{th} entry in the diagonal of the Hat Matrix.

$$\begin{aligned} \hat{\boldsymbol{y}} &= \boldsymbol{H}\boldsymbol{y} \\ &= \boldsymbol{X}\hat{\beta} \\ &= \boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{y} \end{aligned}$$

- LOOCV sometimes useful, but typically doesn't shake up the data enough. The estimates from each fold are highly correlated and hence their average can have high variance.

- A better choice is $K = 5$ or 10.



- Why is there only one line for the left plot? Could we have n lines?

Some Issues with Cross-validation

- Since each training set is only $(K-1)/K$ as big as the original training set, the estimates of prediction error will typically be biased upward.
- This bias is minimized when $K = n$, (LOOCV), but this estimate has high variance.
- $K = 5$ or 10 provides a good compromise for this bias-variance trade-off.

Cross-Validation for Classification Problems

- As before, we divide the data into K roughly equal-sized parts $C_1, C_2, \ldots, C_K$. C_k denotes the indices of the observations in part k.

- Compute:

$$CV_{(K)} = \sum_{k=1}^{K} \frac{n_k}{n} Err_k$$

where $Err_k = \sum_{i \in C_k} \mathbb{I}(y_i \neq \hat{y}_i)/n_k$.

- So when we have have $n_k = n/K$ then:

$$CV_{(K)} = \sum_{k=1}^{K} \frac{1}{K} Err_k$$

Standard Deviation

- To be clear, CV is an estimator. We could and perhaps should put a hat on it!
- We can then get the standard deviation of $MSE = \{MSE_1, MSE_2 \ldots, MSE_K\}$ (or Errs):

$$\widehat{SD}(MSE) = \sqrt{\sum_{k=1}^{K} \left(MSE_k - \overline{MSE}_k\right)^2 / (K-1)}$$

$$\widehat{SD}(Err) = \sqrt{\sum_{k=1}^{K} \left(Err_k - \overline{Err}_k \right)^2 / (K-1)}$$

- This is a useful estimate, but strictly speaking, not quite valid.
- This leads to the standard error for $CV_{(K)}$:

$$SE(CV_{(K)}) = \widehat{SD}(MSE)/\sqrt{K}$$

Cross-validation: Right and Wrong

- Consider a simple classifier applied to some two-class data:
  1. Starting with 5,000 predictors and 50 samples ($n = 50$), find the 100 predictors having the largest correlation with the class labels.
  2. We then apply a classifier such as logistic regression, using only these 100 predictors.
- How do we estimate the test set performance of this classifier?
- Perhaps, we should apply cross-validation directly to step 2?

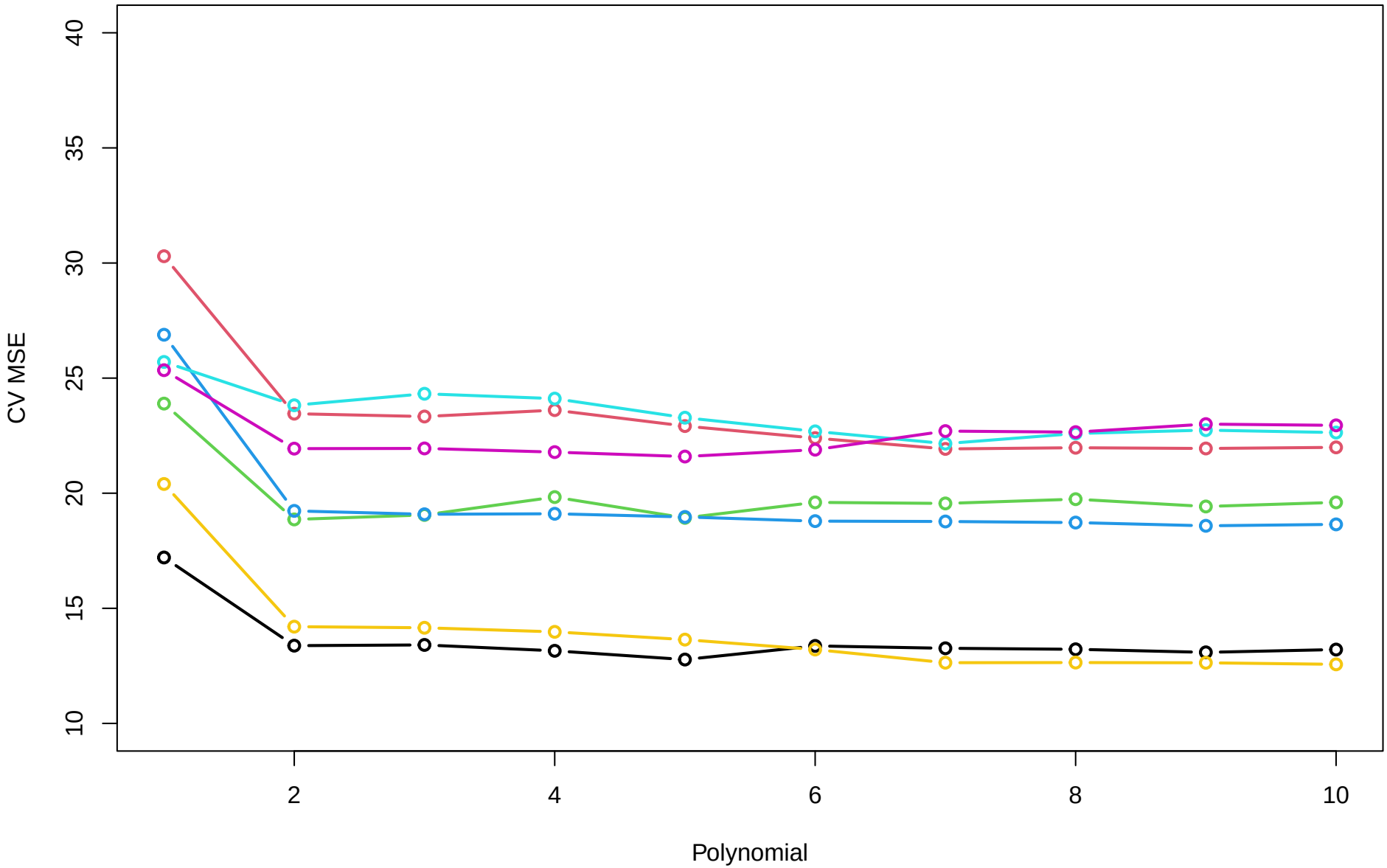
NO!

- This would ignore the fact that in Step 1, the procedure has already seen the labels of the training data, and made use of them. This is a form of training and must be included in the validation process.
- It is easy to simulate realistic data with the class labels independent of the outcome, so that true test error = 50%, but the CV error estimate that ignores Step 1 is zero! (Time to submit your findings!! ??)
- Can you do this simulation?
- According to the book's authors, this error made in many high profile genomics papers.

Back to the Cars

```
library(ISLR)
data(Auto)
data <- Auto
n <- nrow(data)

##
K <- 7
P <- 10
mse.out <- matrix(0, K, P)

##
set.seed(10)
fold <- sample(rep(1:K, each=n/K))

##
for(p in 1:P){
for(k in 1:K){

  data.train <- data[fold!=k,]
  data.test <- data[fold==k,]
  mod.train <- lm(mpg ~ poly(horsepower, degree=p,
                             raw=TRUE), data=data.train)


  pred.test <- predict(mod.train, newdata=data.test)

  mse.out[k,p] <- mean( (data.test$mpg - pred.test)^2)
}}
```

```r
plot(1:P, mse.out[1,], type="b", lwd=2, ylab="CV MSE", xlab="Polynomial",
     ylim=c(10, 40))
for(k in 2:7){ lines(1:P, mse.out[k,], type="b",
                     lwd=2, col=k)}
```



Let's Average the Lines and Calculate the Standard Errors

- We have to average for each model ($p = 1 : P$).

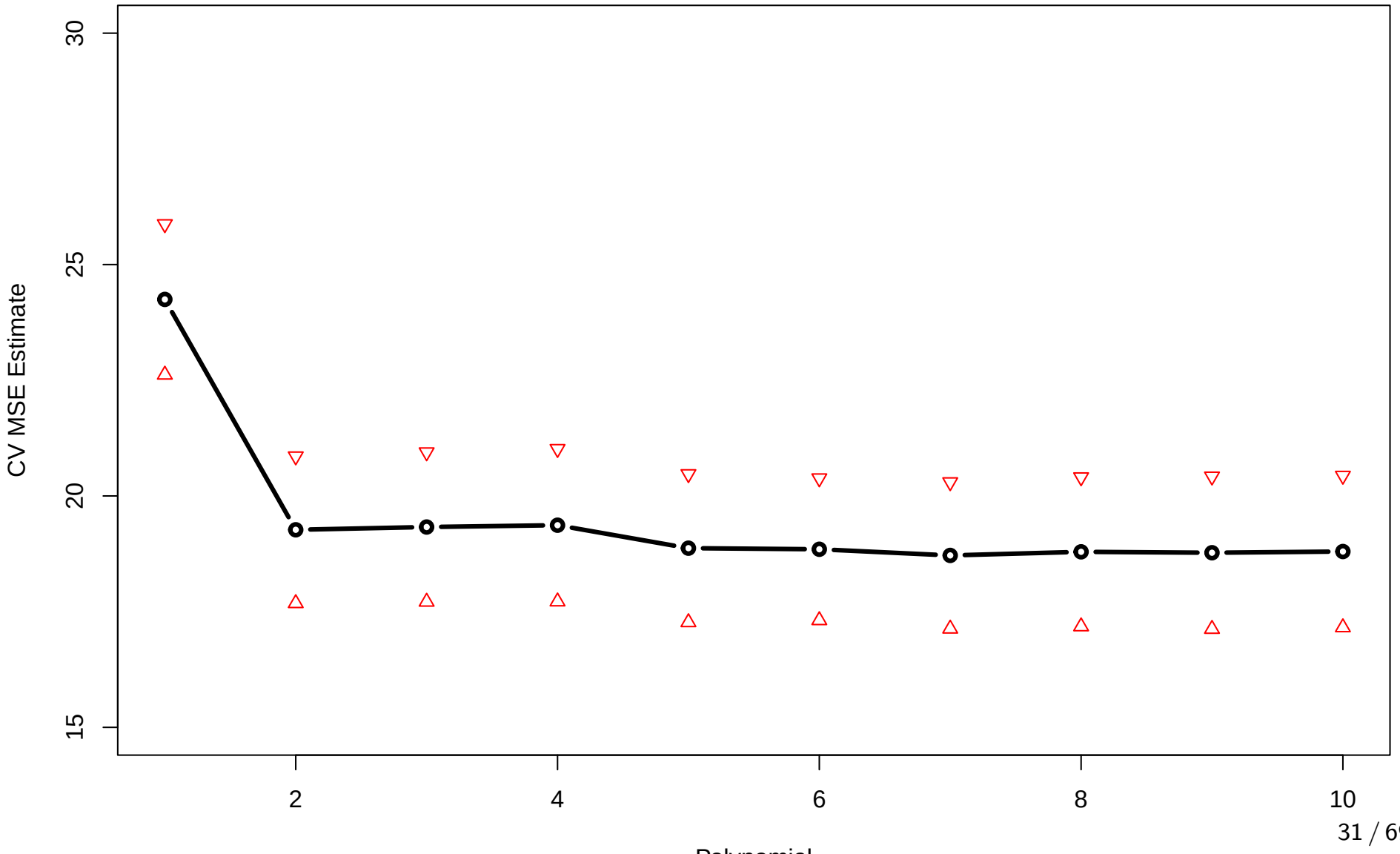
```
mse.est <- apply(mse.out, 2, mean)
mse.est
```

```
##  [1] 24.24651 19.26908 19.32858 19.36780 18.87164 18.84791
##  [9] 18.77325 18.79924
```

```
sd.mse.est <- apply(mse.out, 2, sd)/sqrt(K)
sd.mse.est
```

```
##  [1] 1.628042 1.587856 1.616366 1.649480 1.601857 1.535127
##  [9] 1.648638 1.639547
```

```
plot(1:P, mse.est, lwd=3, type="b", ylim=c(15, 30),
     ylab="CV MSE Estimate", xlab="Polynomial")
points(1:P, mse.est + sd.mse.est, pch=25, col="red")
points(1:P, mse.est - sd.mse.est, pch=24, col="red")
```

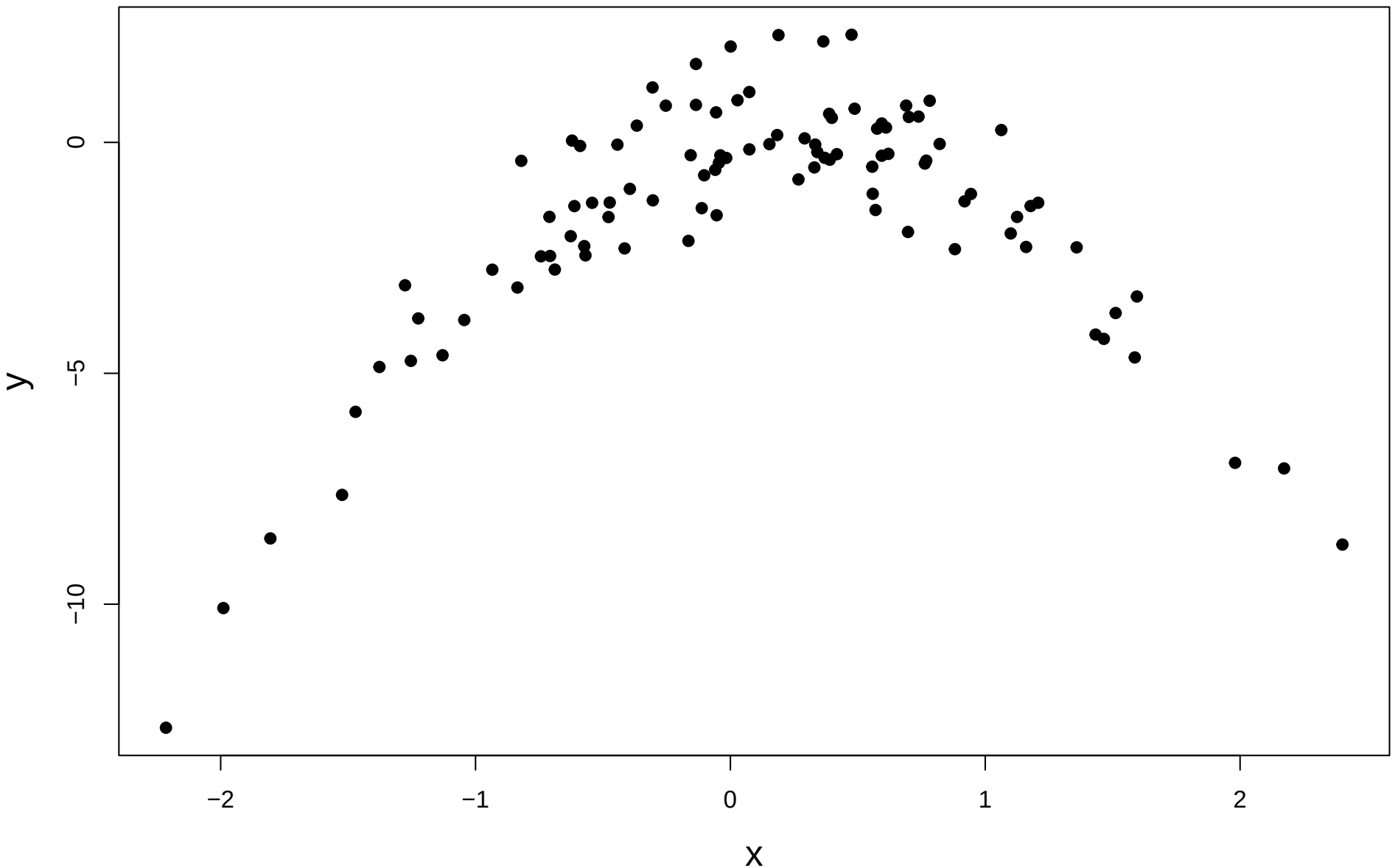


Experiments - Some More Fun with Cross-Validation

- R (and other computer languages/statistical packages) is a laboratory for statisticians!
- Let's generate some data — let's experiment!

```
set.seed(1)
x <- rnorm(100)
y <- x - 2*x^2 + rnorm(100)

D <- as.data.frame(cbind(y, x))
```

```
plot(x, y, pch=19, cex.lab=1.5)
```



- Let's consider the following 4 models for the data:

1. $Y = \beta_0 + \beta_1 X + \epsilon$
2. $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \epsilon$
3. $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \epsilon$
4. $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \beta_4 X^4 + \epsilon$

LOOCV

- Recall the quick magical formula for linear models to calculate the LOOCV. Let's code that.

```
my.loocv <- function(Y, X){

beta.hat <- solve(t(X)%*%X)%*%t(X)%*%Y
y.hat <- X%*%beta.hat
H <-  X%*%solve(t(X)%*%X)%*%t(X)

out <- mean( ((y-y.hat)/(1-diag(H)))^2 )

return(out)

}
```

Let's Use the Code

- I need the design matrix X.
- I will use the `lm()` function to get it.

```
mod1 <- lm(y ~ x)
X <- model.matrix(mod1)
head(X)
```

```
##   (Intercept)          x
## 1           1 -0.6264538
## 2           1  0.1836433
## 3           1 -0.8356286
## 4           1  1.5952808
## 5           1  0.3295078
## 6           1 -0.8204684
```

```
loocv.m1 <- my.loocv(y, X)
loocv.m1
```

```
## [1] 7.288162
```

- It always good if you can so the same thing two different ways. For LOOCV, we should not have any randomness so let's actually leave one out . .

```r
my.loocv2 <- function(Y, X){

  n <- length(y)
  loocv.est.i <- rep(0, n)

  for(i in 1:n){

    train.y <- Y[-i]
    test.y <- Y[i]

    train.X <- X[-i,]
    test.X <- X[i,]

    beta.hat.train <- solve(t(train.X)%*%train.X)%*%t(train.X)%*%train.y
    y.hat.test <-  test.X%*%beta.hat.train

    loocv.est.i[i] <- (test.y - y.hat.test)^2

  }
    out <- mean(loocv.est.i)
    return(out)
}
```

- We get the exact same result! As expected!

```
my.loocv2(y, X)
```

```
## [1] 7.288162
```

- Now let's get the results for the 5 different models.

```
X <- model.matrix(lm(y ~ poly(x, degree=1, raw=TRUE)))
loocv.m1 <- my.loocv(y, X)

X <- model.matrix(lm(y ~ poly(x, degree=2, raw=TRUE)))
loocv.m2 <- my.loocv(y, X)

X <- model.matrix(lm(y ~ poly(x, degree=3, raw=TRUE)))
loocv.m3 <- my.loocv(y, X)

X <- model.matrix(lm(y ~ poly(x, degree=4, raw=TRUE)))
loocv.m4 <- my.loocv(y, X)
```

Results

```
loocv.m1
```

```
## [1] 7.288162
```

```
loocv.m2
```

```
## [1] 0.9374236
```

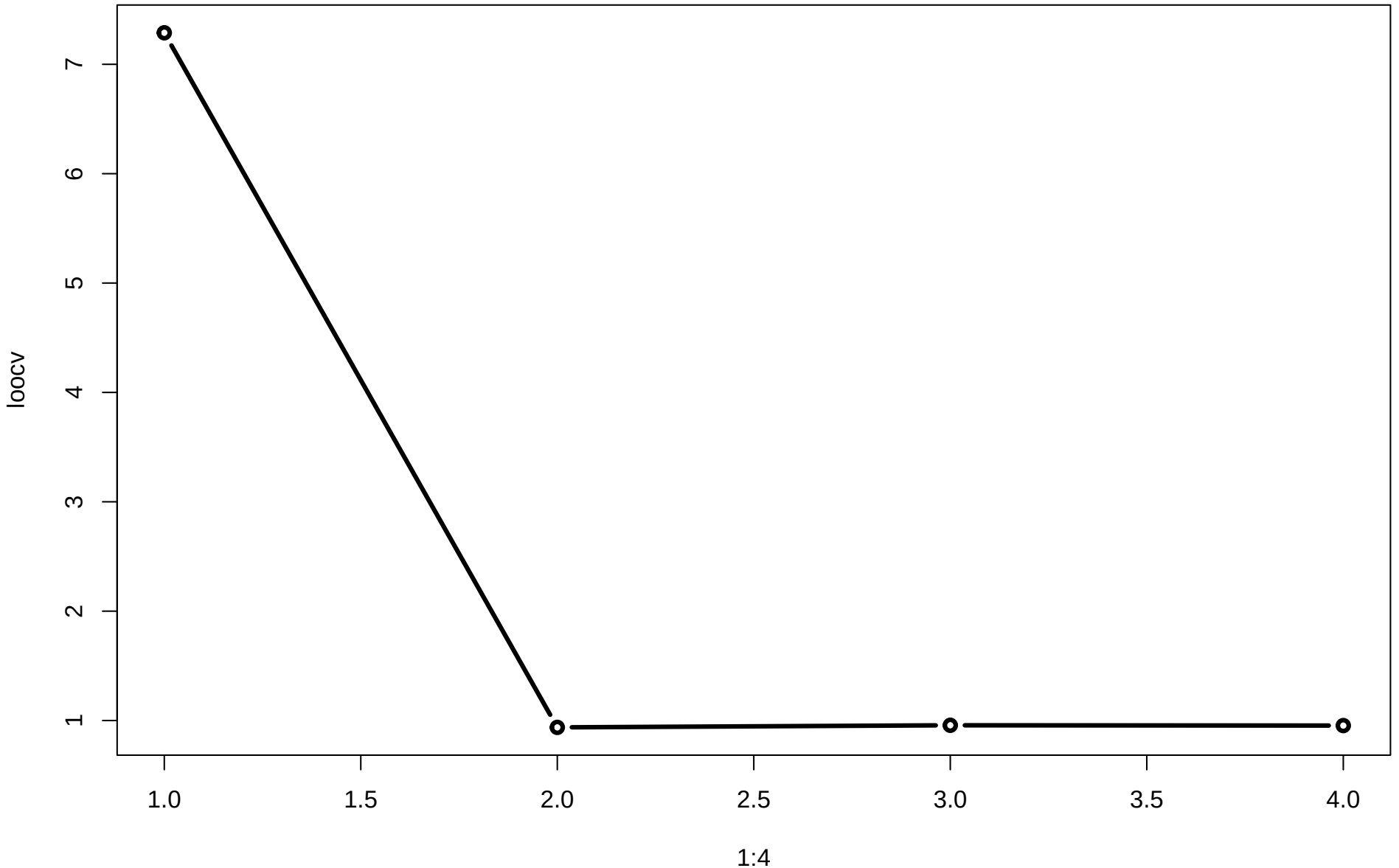
```
loocv.m3
```

```
## [1] 0.9566218
```

```
loocv.m4
```

```
## [1] 0.9539049
```

```r
loocv <- c(loocv.m1, loocv.m2, loocv.m3, loocv.m4)
plot(1:4, loocv, type="b", lwd=3)
```



- This suggests that we should pick model 2.
- Let's look at the standard regression tables for the 4 models.
- Also, let's do an F-test to compare compare models 2 vs 4.
- For these tests we should use the full data sets!

```
mod1 <- lm(y ~ poly(x, degree=1, raw=TRUE))
mod2 <- lm(y ~ poly(x, degree=2, raw=TRUE))
mod3<- lm(y ~ poly(x, degree=3, raw=TRUE))
mod4 <- lm(y ~ poly(x, degree=4, raw=TRUE))
```

```
summary(mod1)
```

```
##
## Call:
## lm(formula = y ~ poly(x, degree = 1, raw = TRUE))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.5161 -0.6800  0.6812  1.5491  3.8183
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     -1.6254     0.2619  -6.205 1.31e-08 ***
## poly(x, degree = 1, raw = TRUE)   0.6925     0.2909   2.380   0.0192 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.6 on 98 degrees of freedom
## Multiple R-squared:  0.05465,    Adjusted R-squared:  0.045
## F-statistic: 5.665 on 1 and 98 DF,  p-value: 0.01924
```

```
summary(mod2)
```

```
##
## Call:
## lm(formula = y ~ poly(x, degree = 2, raw = TRUE))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.9650 -0.6254 -0.1288  0.5803  2.2700
##
## Coefficients:
##                                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)                      0.05672    0.11766   0.482    0.631
## poly(x, degree = 2, raw = TRUE)1  1.01716    0.10798   9.420  2.4e-15 **
## poly(x, degree = 2, raw = TRUE)2 -2.11892    0.08477 -24.997  < 2e-16 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.958 on 97 degrees of freedom
## Multiple R-squared:  0.873,  Adjusted R-squared:  0.8704
## F-statistic: 333.3 on 2 and 97 DF,  p-value: < 2.2e-16
```

```
summary(mod3)
```

```
##
## Call:
## lm(formula = y ~ poly(x, degree = 3, raw = TRUE))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.9765 -0.6302 -0.1227  0.5545  2.2843
##
## Coefficients:
##                                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)                        0.06151    0.11950   0.515    0.608
## poly(x, degree = 3, raw = TRUE)1   0.97528    0.18728   5.208 1.09e-06 **
## poly(x, degree = 3, raw = TRUE)2 -2.12379    0.08700 -24.411  < 2e-16 **
## poly(x, degree = 3, raw = TRUE)3   0.01764    0.06429   0.274    0.784
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9626 on 96 degrees of freedom
## Multiple R-squared:  0.8731, Adjusted R-squared:  0.8691
## F-statistic: 220.1 on 3 and 96 DF,  p-value: < 2.2e-16
```

```
summary(mod4)
```

```
##
## Call:
## lm(formula = y ~ poly(x, degree = 4, raw = TRUE))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.0550 -0.6212 -0.1567  0.5952  2.2267
##
## Coefficients:
##                                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)                        0.156703   0.139462   1.124    0.264
## poly(x, degree = 4, raw = TRUE)1   1.030826   0.191337   5.387 5.17e-07 *
## poly(x, degree = 4, raw = TRUE)2 -2.409898   0.234855 -10.261  < 2e-16 *
## poly(x, degree = 4, raw = TRUE)3 -0.009133   0.067229  -0.136    0.892
## poly(x, degree = 4, raw = TRUE)4  0.069785   0.053240   1.311    0.193
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9591 on 95 degrees of freedom
## Multiple R-squared:  0.8753, Adjusted R-squared:  0.8701
## F-statistic: 166.7 on 4 and 95 DF,  p-value: < 2.2e-16
```

```
anova(mod1, mod2)
```

```
## Analysis of Variance Table
##
## Model 1: y ~ poly(x, degree = 1, raw = TRUE)
## Model 2: y ~ poly(x, degree = 2, raw = TRUE)
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1     98 662.55
## 2     97  89.03  1    573.52 624.87 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- From the t-statistic in table 2 — difference is just due to rounding! This only works with linear models (standard linear regression models [here linear means in terms of the parameters not the covariates]).

```
(-24.99)^2
```

```
## [1] 624.5001
```

```
anova(mod2, mod4)
```

```
## Analysis of Variance Table
##
## Model 1: y ~ poly(x, degree = 2, raw = TRUE)
## Model 2: y ~ poly(x, degree = 4, raw = TRUE)
##   Res.Df    RSS Df Sum of Sq     F Pr(>F)
## 1     97 89.029
## 2     95 87.379  2      1.65 0.897 0.4112
```

- Do not reject the smaller model in favor of the larger model.

The Bootstrap

- The bootstrap is a flexible and powerful statistical tool that can be used to quantify the uncertainty associated with a given estimator or statistical learning method.
- For example, it can provide an estimate of the standard error of a coefficient, or a confidence interval for that coefficient.

Where does the name came from?

- The use of the term bootstrap derives from the phrase to pull oneself up by one's bootstraps, widely thought to be based on one of the eighteenth century "The Surprising Adventures of Baron Munchausen" by Rudolph Erich Raspe:
  The Baron had fallen to the bottom of a deep lake. Just when it looked like all was lost, he thought to pick himself up by his own bootstraps.
- It is not the same as the term "bootstrap" used in computer science meaning to "boot" a computer from a set of core instructions, though the derivation is similar.

A Simple Example

- Suppose that we wish to invest a fixed sum of money in two financial assets that yield returns of X and Y, respectively, where X and Y are random quantities.
- We will invest a fraction α of our money in X, and will invest the remaining $1 - \alpha$ in Y.
- We wish to choose α to minimize the total risk, or variance, of our investment. In other words, we want to:

$$\text{minimize } V(\alpha X + (1-\alpha)Y) = \alpha^2 V(X) + (1-\alpha)^2 V(Y)$$

- Only if X and Y are independent. Let's assume they are dependent, so we have:

$$V(\alpha X + (1-\alpha)Y) = \alpha^2 V(X) + (1-\alpha)^2 V(Y) + 2\alpha(1-\alpha)Cov(X,Y)$$

Minimizing this wrt to α we have:

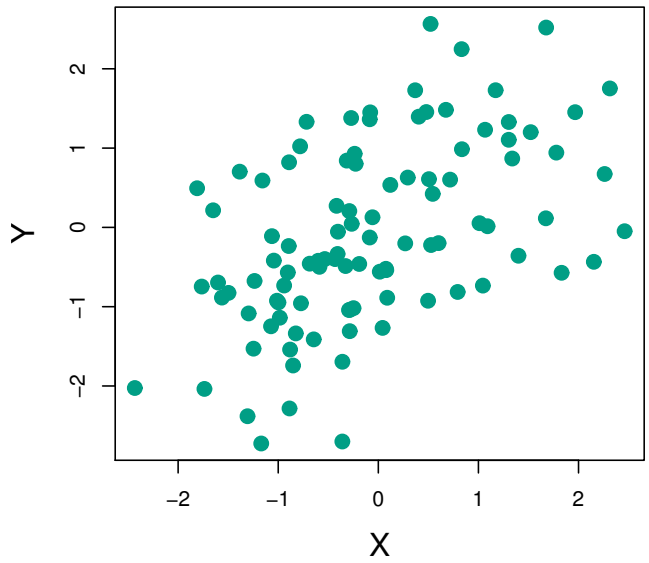
$$\alpha = \frac{\sigma_Y^2 - \sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}}$$

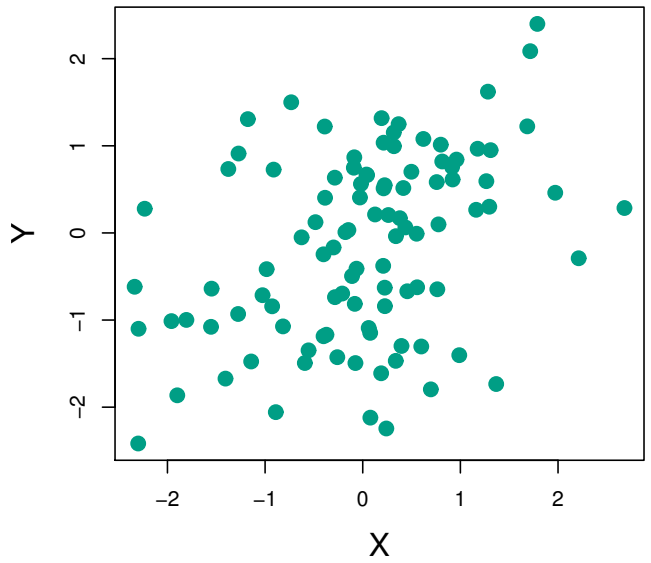
- However, the values of σ_X^2, σ_Y^2, and σ_{XY} are unknown.

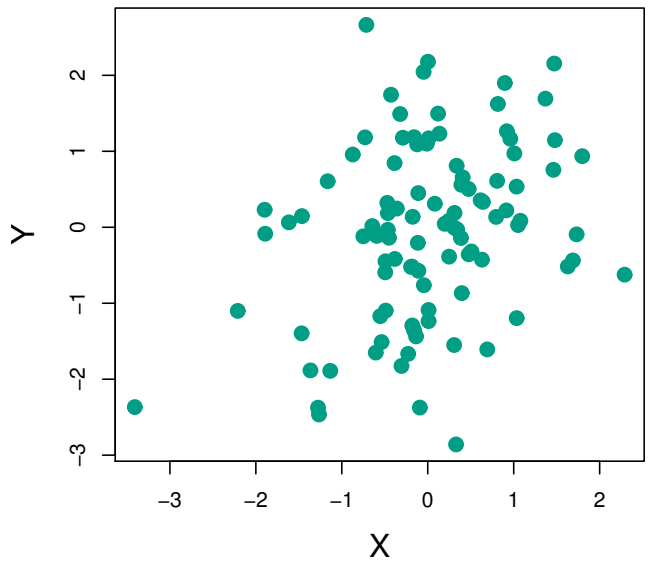
- We can compute estimates for these quantities using a data set that contains measurements for X and Y. Then we can estimate the value of α that minimizes the variance of our investment using

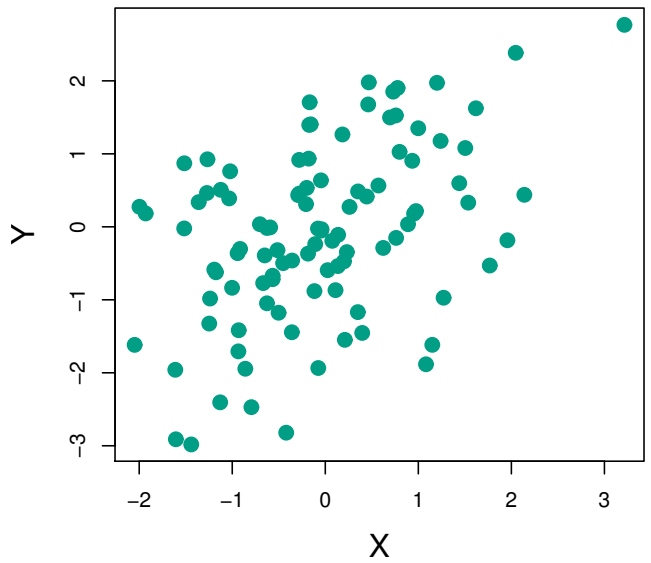
$$\hat{\alpha} = \frac{\hat{\sigma}_Y^2 - \hat{\sigma}_{XY}}{\hat{\sigma}_X^2 + \hat{\sigma}_Y^2 - 2\hat{\sigma}_{XY}}$$

- In the following figure, each panel displays 100 simulated returns for investments X and Y. From left to right and top to bottom, the resulting estimates for α are 0.576, 0.532, 0.657, and 0.651.


Y
X
−2
−1
0
1
2


Y
X
−2
−1
0
1
2


Y
X
−3
−2
−1
0
1
2


Y
X
−3
−2
−1
0
1
2
3

- Of course we are interested in understanding the uncertainty in those estimates.
- To estimate the standard deviation of $\hat{\alpha}$, we repeated the process of simulating 100 paired observations of X and Y, and estimating α 1,000 times.
- We thereby obtained 1,000 estimates for α, which we can call $\hat{\alpha}_1, \hat{\alpha}_1, \ldots, \hat{\alpha}_{1000}$.
- For the simulations, we know the truth. We set the values to:

$$\sigma_X^2 = 1, \sigma_Y^2 = 1.25, \sigma_{XY} = 0.5 \Rightarrow \alpha = 0.6$$

```r
library(MASS)

m <- c(0,0)
S <- matrix( c(1,0.5,0.5,1.25), byrow=TRUE, ncol=2, nrow=2)
S
```

```
##      [,1] [,2]
## [1,]  1.0 0.50
## [2,]  0.5 1.25
```

```r
set.seed(1)
Data <- mvrnorm(n = 100, mu=m, Sigma=S)
head(Data)
```

```
##              [,1]        [,2]
## [1,] -0.11199307 -0.93050025
## [2,]  0.11883004  0.20562733
## [3,] -0.09805471 -1.28127680
## [4,]  1.16015617  1.68639250
## [5,]  0.66255920  0.01811553
## [6,] -1.73430781  0.02090527
```

```r
S.hat <- cov(Data)
S.hat
```

```
##           [,1]      [,2]
## [1,] 0.8495443 0.3708332
## [2,] 0.3708332 1.0331973
```

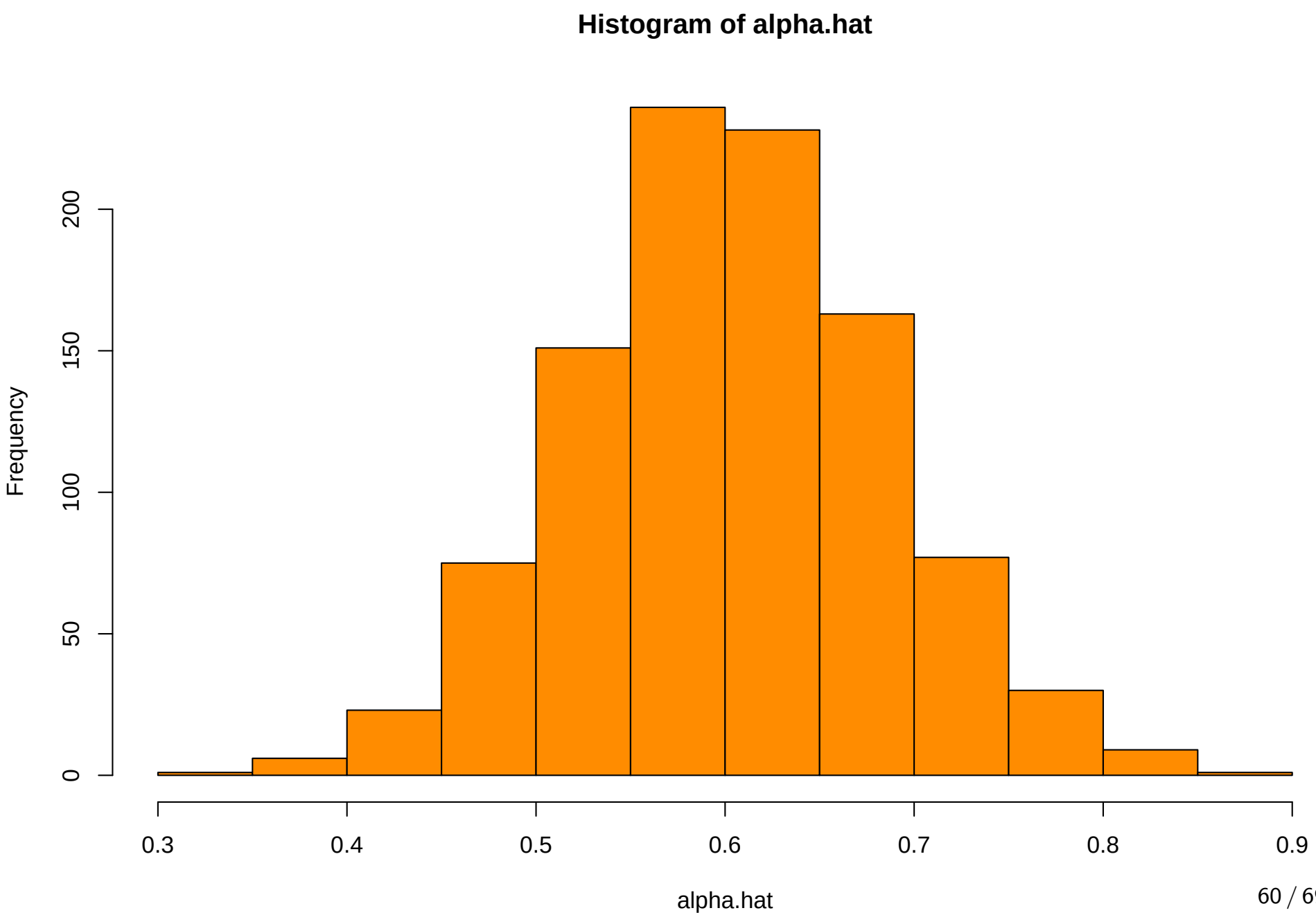
```r
sigma.sq.x.hat <- S.hat[1,1]
sigma.sq.y.hat <- S.hat[2,2]
sigma.xy.hat <- S.hat[1,2]

alpha.hat <- (sigma.sq.y.hat - sigma.xy.hat)/(sigma.sq.x.hat +
                                              sigma.sq.y.hat - 2*sigma.xy.hat)
alpha.hat
```

```
## [1] 0.5804737
```

- Let's repeat the process 1,000 times and store the $\hat{\alpha}$ results.

```
n.sim <- 1000
alpha.hat <- rep(0, n.sim)

m <- c(0,0)
S <- matrix( c(1,0.5,0.5,1.25), byrow=TRUE, ncol=2, nrow=2)
set.seed(1)

for(i in 1:n.sim){
  Data <- mvrnorm(n = 100, mu=m, Sigma=S)
  S.hat <- cov(Data)
  sigma.sq.x.hat <- S.hat[1,1]
  sigma.sq.y.hat <- S.hat[2,2]
  sigma.xy.hat <- S.hat[1,2]

alpha.hat[i] <- (sigma.sq.y.hat - sigma.xy.hat)/(sigma.sq.x.hat +
                                        sigma.sq.y.hat - 2*sigma.xy.hat)
}
```

```r
hist(alpha.hat, col="darkorange")
```



- The sample mean over all 1,000 estimates for $\hat{\alpha}$ is

$$\bar{\alpha} = \sum_{i=1}^{1000} \hat{\alpha}_i / 1000 = 0.603$$

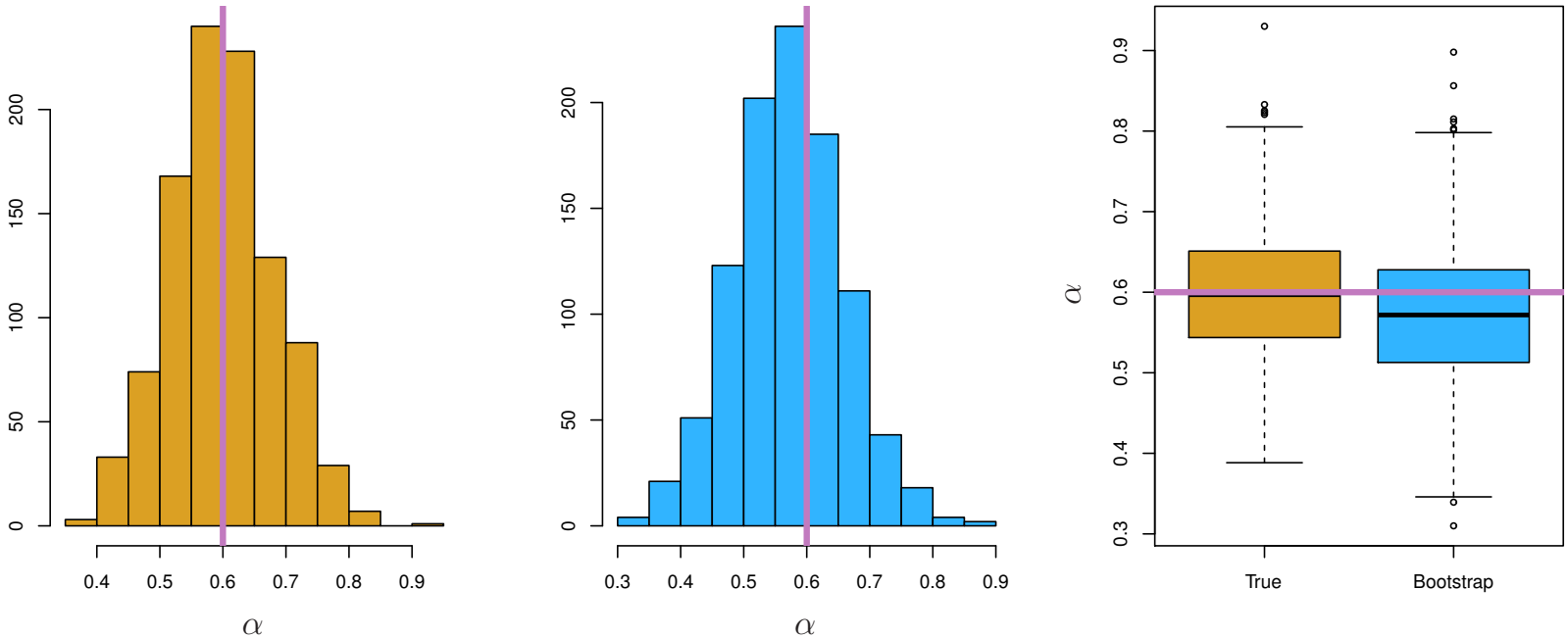
Quite close to the truth of 0.6.

- The standard deviation of the estimates is:

$$\sqrt{\frac{1}{1000-1} \sum_{i=1}^{1000} (\hat{\alpha}_i - \bar{\alpha})^2} = 0.082$$

- This gives us a very good idea of the accuracy of $\hat{\alpha}$: $SE(\hat{\alpha}) = 0.082$.

Results

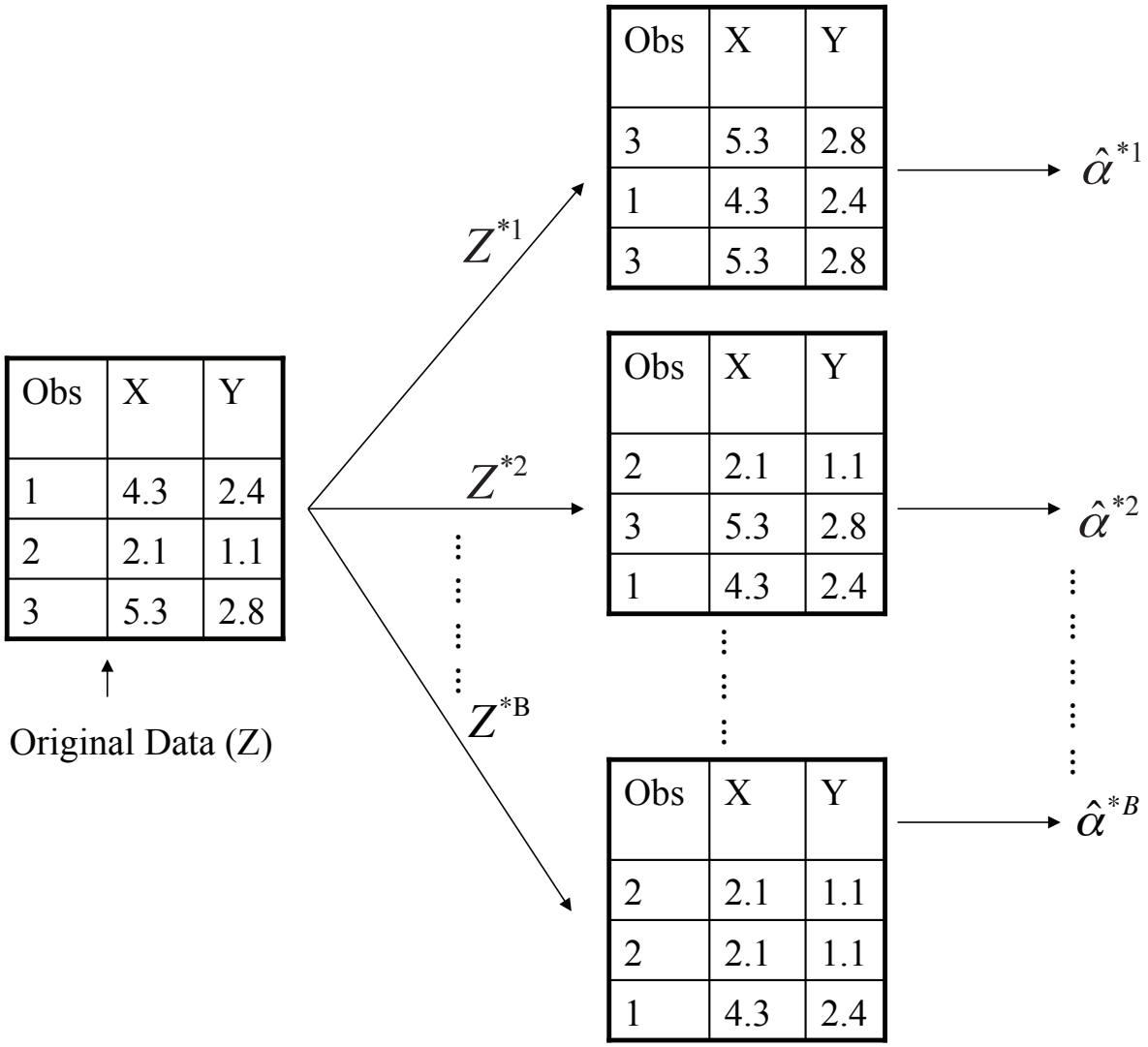


- Left: A histogram of the estimates of $\hat{\alpha}$ obtained by generating 1,000 simulated data sets from the true population (Monte Carlo).
- Center: A histogram of the estimates of $\hat{\alpha}$ obtained from 1,000 bootstrap samples from a single data set.
- Right: The estimates of $\hat{\alpha}$ displayed in the left and center panels are shown as boxplots. In each panel, the pink line indicates the true value of α.

Now Back to the Real World

- The procedure outlined above cannot be applied, because for real data we cannot generate new samples from the original population.
- However, the bootstrap approach allows us to use a computer to mimic the process of obtaining new data sets, so that we can estimate the variability of our estimate without generating additional samples.
- Rather than repeatedly obtaining independent data sets from the population, we instead obtain distinct data sets by repeatedly sampling observations from the original data set with replacement.
- Each of these "bootstrap data sets" is created by sampling with replacement, and is the same size as our original dataset. As a result some observations may appear more than once in a given bootstrap data set and some not at all.

Example with Just 3 Observations



- Each bootstrap data set contains $n = 3$ observations, sampled with

- Denoting the bootstrap data set by Z^{*1}, we use Z^{*1} to produce a new bootstrap estimate for α, which we call $\hat{\alpha}^{*1}$.

- This procedure is repeated B times for some large value of B (say 100 or 1000), in order to produce B different bootstrap data sets:

$$Z^{*1}, \ldots, Z^{*B}$$

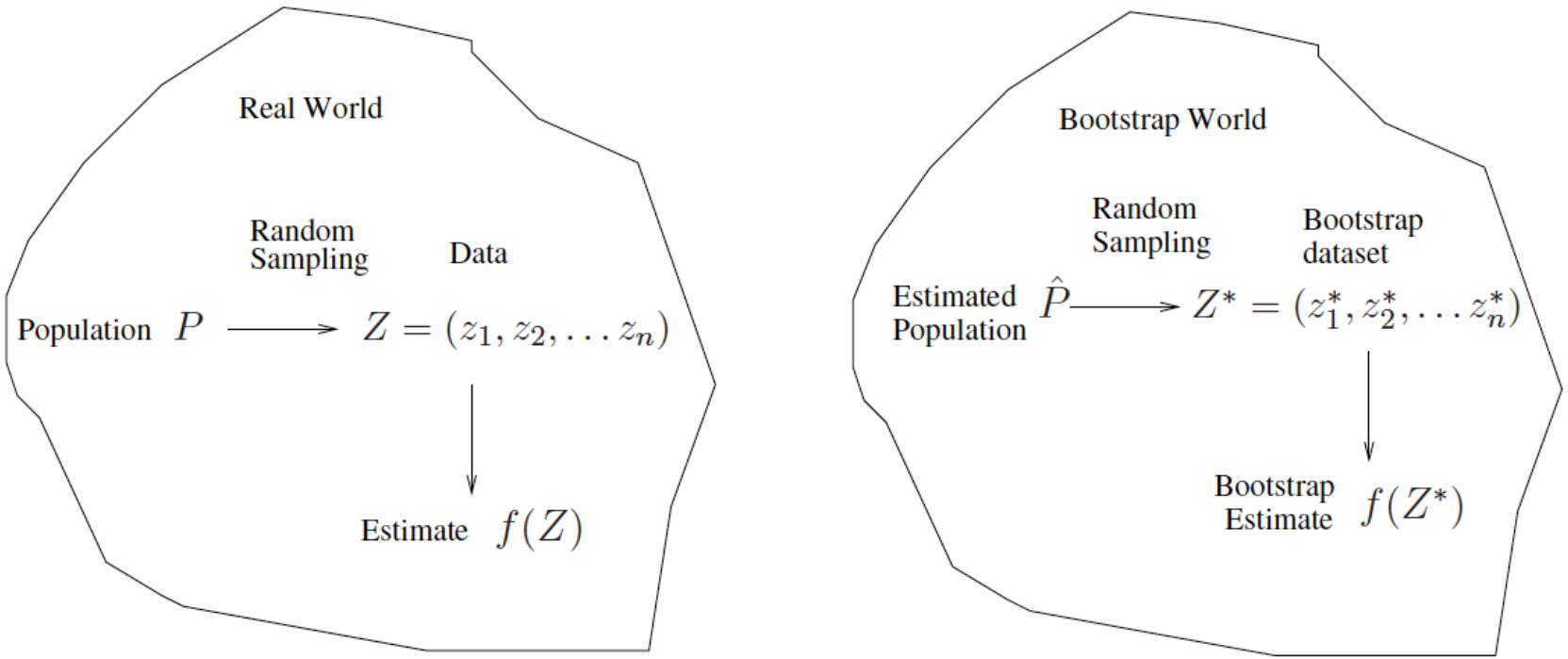
and the corresponding estimates:

$$\hat{\alpha}^{*1}, \ldots, \hat{\alpha}^{*B}$$

- We estimate the standard error of these bootstrap estimates using the formula

$$\sqrt{\frac{1}{B-1}\sum_{i=1}^{B}(\hat{\alpha}^{*i} - \bar{\hat{\alpha}}^{*})^2}$$

A General Picture for the Bootstrap



The Bootstrap in General

- In more complex data situations, figuring out the appropriate way to generate bootstrap samples can require some thought.
- For example, if the data is a time series, we can't simply sample the observations with replacement.
- Primarily used to obtain standard errors of an estimate.
- Also provides approximate confidence intervals for a population parameter. For example, looking at the histogram in the middle panel of previous, the 5% and 95% quantiles of the 1,000 values is (0.43, 0.72). This represents an approximate 90% confidence interval for the true α.
- The above interval is called a Bootstrap Percentile confidence interval. It is the simplest method (among many approaches) for obtaining a confidence interval from the bootstrap.

Can the Bootstrap Estimate Prediction Error?

- In cross-validation, each of the K validation folds is distinct from the other $K-1$ folds used for training: there is no overlap. This is crucial for its success.
- To estimate prediction error using the bootstrap, we could think about using each bootstrap dataset as our training sample, and the original sample as our validation sample.
- But each bootstrap sample has significant overlap with the original data. About two-thirds of the original data points appear in each bootstrap sample.
- This will cause the bootstrap to seriously underestimate the true prediction error.
- The other way around| with original sample = training sample, bootstrap dataset = validation sample — is worse!

Removing the Overlap

- Can partly fix this problem by only using predictions for those observations that did not (by chance) occur in the current bootstrap sample.
- But the method gets complicated, and in the end, cross-validation provides a simpler, more attractive approach for estimating prediction error.