Data Partitioning

CROSS-VALIDATION; JACKKNIFE AND, BRIEFLY, LINEAR REGRESSION

DEFINITIONS

“**Data partitioning** refers to **procedures** where **some observations from the sample** are removed as part of the analysis”

These techniques are used for the following purposes:

- To **evaluate** the accuracy of the model or classification scheme
- To **decide** what is a reasonable model for the data
- To **find a smoothing parameter** in density estimation
- To **estimate the bias and error** in parameter estimation
- And many others…

EXAMPLE – for motivation

- We have a sample where we measured the **average atmospheric temperature** and the **corresponding amount of steam** used per month [Draper and Smith, 1981]
- **Our goal** → model the relationship between these variables

Once we have a model:

1. We *can use it to **predict** how much steam is needed for a given average monthly temperature*
2. To ***gain understanding*** about the structure of the relationship between the two variables

Then, the problem is "deciding what model to use"

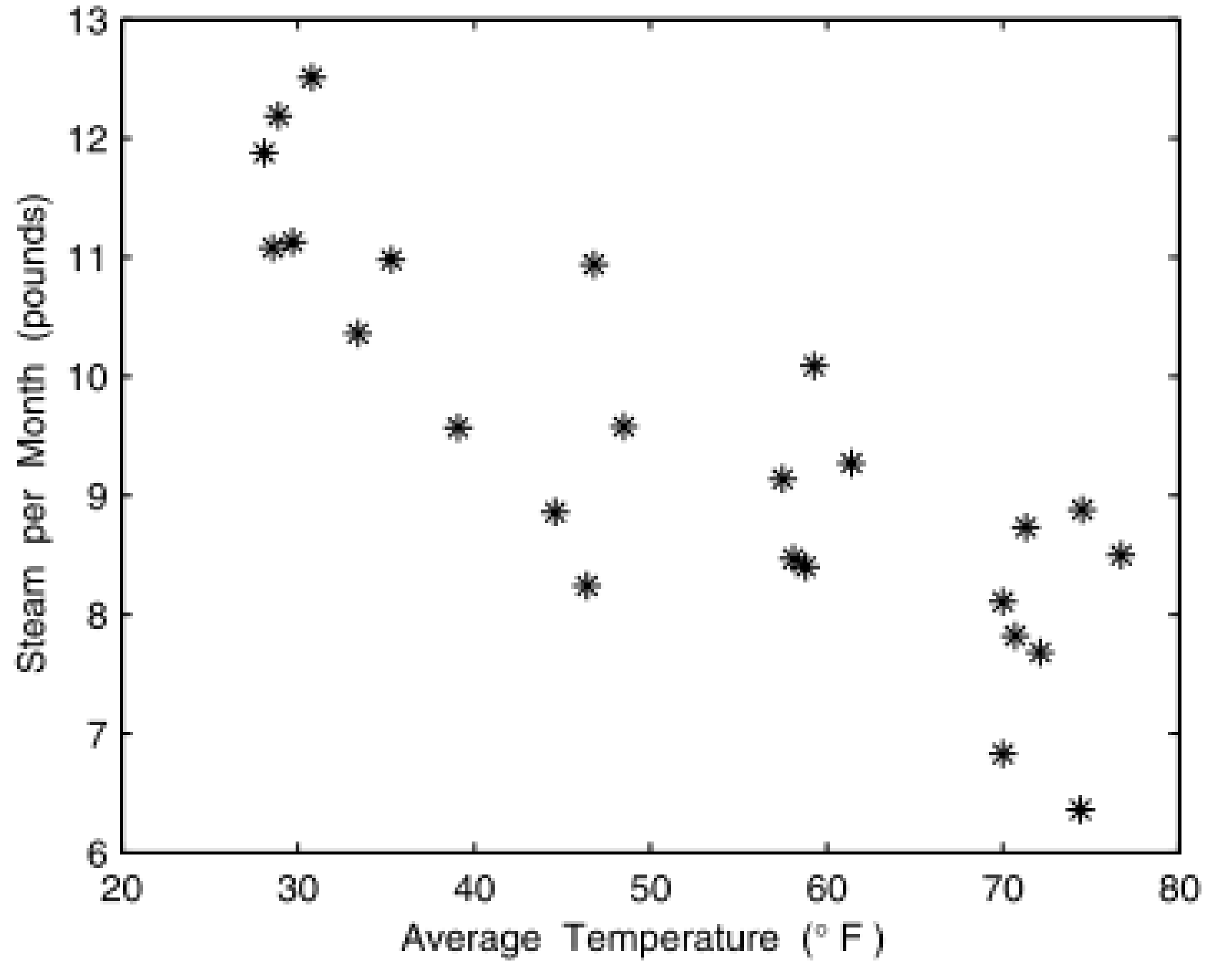


FIGURE 8.1
Scatterplot of a data set where we are interested in modeling the relationship between average temperature (the predictor variable) and the amount of steam used per month (the response variable). The scatterplot indicates that modeling the relationship with a straight line is reasonable.

- We see from the plot that: as the temperature increases, the amount of steam used per month decreases

- It appears that using a **line** (i.e., a first degree polynomial) to **model** the relationship between the variables is **reasonable**

- *Other models might provide a better fit*: e.g., a cubic or some higher degree polynomial might be a better model for the relationship between average temperature and steam usage

Which model to use?

To make that decision, we need to assess the accuracy of the various models

- Next, we could choose the model that has the best accuracy or lowest error

"We can use the **prediction error** to **measure the accuracy**"

GENERAL APPROACH:

1. **One way to assess the error** would be to **observe** new data (average temperature and corresponding monthly steam usage)
2. Then, **determine** what is the predicted monthly steam usage for the new observed average temperatures
3. Next, **compare this prediction** with the **true steam** used and **calculate the error**
4. We **do this for all of the proposed models**
5. **Pick the model with the smallest error**

What is the problem with this approach?

It is sometimes impossible to obtain new data, so all we have available to evaluate our models (or our statistics) is the original data set

- There are 2 methods that allow us to use the data already in hand for the evaluation of the models: **cross-validation** and **jackknife**
- **Cross-validation** is typically used to determine the classification error rate for pattern recognition applications or the prediction error when building models.
- Bootstrap method can be used for *estimating the bias and standard error of statistics* - the **jackknife procedure** (more a data partitioning method than a simulation method) *has a similar purpose* and was developed prior to the bootstrap [Quenouille,1949]. The connection between the methods is well known and is discussed in the literature [Efron and Tibshirani, 1993; Efron, 1982; Hall, 1992].

Cross-Validation

Often, one of the jobs of a statistician or engineer is to <u>create models using sample data</u> (usually for the purpose of making predictions)

E.g. **Given a data set** that contains the **drying time** and the **tensile strength** of batches of cement, "can we model the relationship between these two variables"?

- We would like to be able to **predict** the <u>tensile strength</u> of the cement <u>for a given drying time that we will observe in the future</u>
- Then, decide what *model best describes the relationship* between the variables and *estimate its accuracy*

THE PROBLEM

The "naive researcher" will build a model based on the data set and <u>then use that same data</u> to **assess the performance of the model**

THE PROBLEM
The "naive researcher" will build a model based on the data set and then use that same data to **assess the performance of the model**

Cross-Validation

What not to do: have a model being evaluated or tested with data it has already seen

If this is done: an overly optimistic (i.e., low) prediction error may be yielded

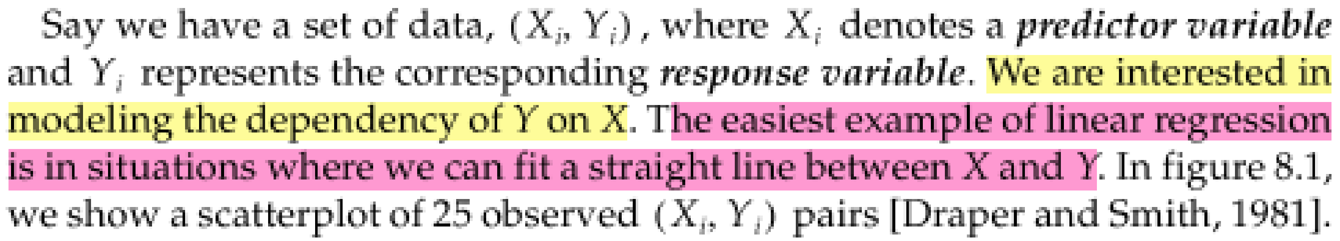
Cross-validation: (addresses this problem) by iteratively partitioning the sample into two sets of data:

- **Set 1**: used for building the model
- **Set 2**: used to test it

What we will see:

- **Cross-validation** in a linear regression application - where we are interested in estimating the expected prediction error
- We use linear regression to illustrate the **cross-validation** concept

Linear Regression Briefly…



Say we have a set of data, (X_i, Y_i), where X_i denotes a ***predictor variable*** and Y_i represents the corresponding ***response variable***. We are interested in modeling the dependency of Y on X. The easiest example of linear regression is in situations where we can fit a straight line between X and Y. In figure 8.1, we show a scatterplot of 25 observed (X_i, Y_i) pairs [Draper and Smith, 1981].

- X: average atmospheric temperature measured in degrees F
- Y: variable corresponds to the pounds of steam used per month

From the scatterplot, we see that a straight line is a reasonable model for the relationship between these variables

We will use these data to illustrate linear regression.

The linear, first-order model is given by:

$$Y = \beta_0 + \beta_1 X + \varepsilon, \qquad (8.1)$$

ε = error in measurements

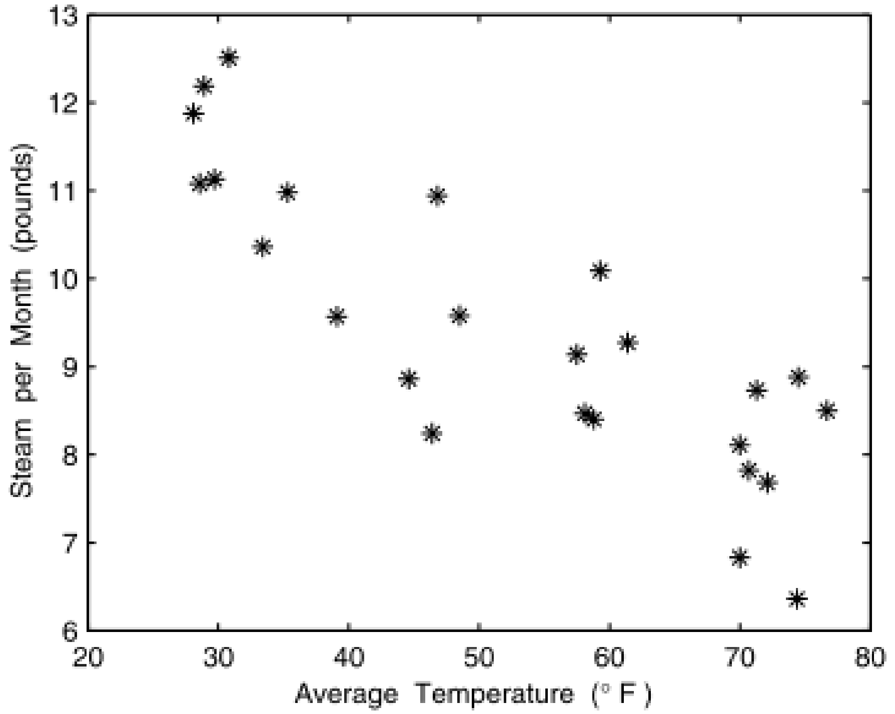


FIGURE 8.1
Scatterplot of a data set where we are interested in modeling the relationship between average temperature (the predictor variable) and the amount of steam used per month (the response variable). The scatterplot indicates that modeling the relationship with a straight line is reasonable.

Linear Regression Briefly…

The linear, first-order model is given by:

$$Y = \beta_0 + \beta_1 X + \varepsilon, \qquad (8.1)$$

- **β0** and **β1** are <u>parameters to be estimated</u> from the data
- **ε** represents the <u>error in the measurements</u>

- **Linear**: it refers to the linearity of the parameters βi
  - *We know from elementary algebra that β1 is the slope and β0 is the y-intercept (where y crosses the y-axis)*
- **Order (or degree) of the model**: it refers to the highest power of the predictor variable X
  - *e.g.,* $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \varepsilon$ *, is a linear, 2nd order model*

To get the model in equation 8.1, we need to estimate the parameters β_0 and β_1. Thus, the estimate of our model is

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 X, \qquad (8.3)$$

where $\hat{Y}$ denotes the predicted value of Y for some value of X, and $\hat{\beta}_0$, $\hat{\beta}_1$ are the estimated parameters.

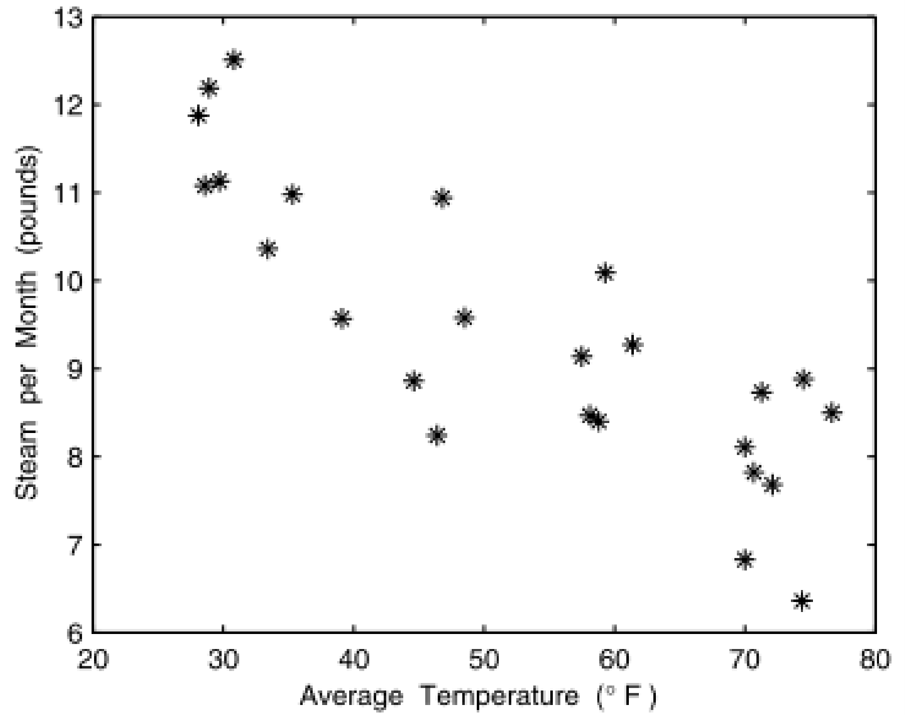


FIGURE 8.1
Scatterplot of a data set where we are interested in modeling the relationship between average temperature (the predictor variable) and the amount of steam used per month (the response variable). The scatterplot indicates that modeling the relationship with a straight line is reasonable.

Linear Regression Briefly…

Assume that we have a sample of observed predictor variables with corresponding responses. We denote these by (X_i, Y_i), $i = 1, \ldots, n$. The ***least squares*** fit is obtained by finding the values of the parameters that minimize the sum of the squared errors

$$RSE = \sum_{i=1}^{n} \varepsilon^2 = \sum_{i=1}^{n} (Y_i - (\beta_0 + \beta_1 X_i))^2, \qquad (8.4)$$

where *RSE* denotes the ***residual squared error***.

Estimates of the parameters $\hat{\beta}_0$ and $\hat{\beta}_1$ are easily obtained in MATLAB® using the function **`polyfit`**

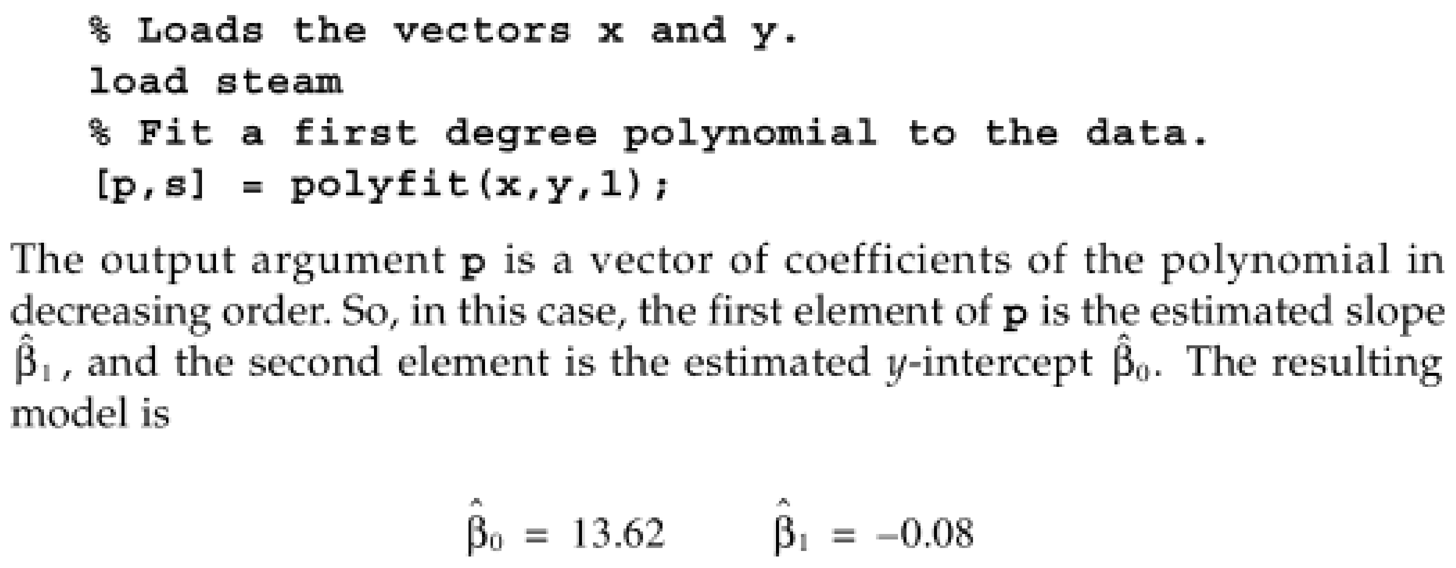
We use the function **`polyfit`** in example 8.1 to model the linear relationship between the atmospheric temperature and the amount of steam used per month (see figure 8.1).

Linear Regression Briefly…

The ***polyfit*** function takes 3 arguments:

- **Observed x** values
- **Observed y** values
- The **degree of the polynomial** that we want to fit to the data

The following commands fit a polynomial of degree one to the steam data.



```
% Loads the vectors x and y.
load steam
% Fit a first degree polynomial to the data.
[p,s] = polyfit(x,y,1);
```

The output argument `p` is a vector of coefficients of the polynomial in decreasing order. So, in this case, the first element of `p` is the estimated slope $\hat{\beta}_1$, and the second element is the estimated *y*-intercept $\hat{\beta}_0$. The resulting model is

$$\hat{\beta}_0 = 13.62 \qquad \hat{\beta}_1 = -0.08$$

Linear Regression Briefly…

We can use the **polyval** function to get predictions using this model. First, we have to get a domain of interest **xf**, and then we evaluate the estimated function over that domain.

```
% We can evaluate the polynomial over a
% domain of interest.
xf = linspace(min(x),max(x));
yf = polyval(p,xf);
```

Next, we produce a scatterplot of the original data with the fitted line superimposed on it.

```
% Now produce the plot for figure 8.2
plot(x,y,'o',xf,yf,'-')
axis equal
xlabel('Average Temperature (\circ F)')
ylabel('Steam per Month (pounds)')
```

The predictions that would be obtained from the model (i.e., points on the line given by the estimated parameters) are shown in figure 8.2, and we see that it seems to be a reasonable fit.

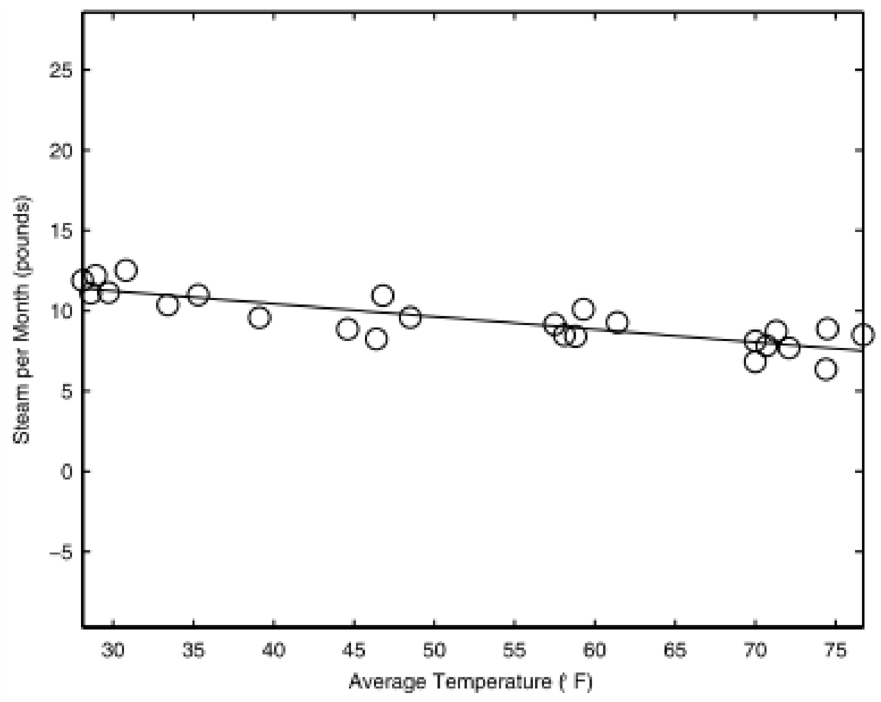


FIGURE 8.2
This figure shows a scatterplot of the **steam** data along with the line obtained using **polyfit**. The estimate of the slope is $\hat{\beta}_1 = -0.08$, and the estimate of the y-intercept is $\hat{\beta}_0 = 13.62$.

The ***prediction error*** is defined as

$$PE = E[(Y - \hat{Y})^2] , \tag{8.5}$$

where the expectation is with respect to the true population. To estimate the error given by equation 8.5, we need to test our model (obtained from **poly-fit**) using an independent set of data that we denote by (x_i', y_i'). This means that we would take an observed (x_i', y_i') and obtain the estimate of $\hat{y}_i'$ using our model:

$$\hat{y}_i' = \hat{\beta}_0 + \hat{\beta}_1 x_i' . \tag{8.6}$$

We then compare $\hat{y}_i'$ with the true value of y_i'. Obtaining the outputs or $\hat{y}_i'$ from the model is easily done in MATLAB using the **polyval** function as shown in example 8.2.

Say we have m independent observations (x_i', y_i') that we can use to test the model. We estimate the prediction error (equation 8.5) using

$$\hat{PE} = \frac{1}{m}\sum_{i=1}^{m}(y_i' - \hat{y}_i')^2 . \tag{8.7}$$

Equation 8.7 measures the average squared error between the predicted response obtained from the model and the true measured response. It should be noted that other measures of error can be used, such as the absolute difference between the observed and predicted responses.

Linear Regression Briefly…

Question: How to estimate the prediction error using $$\hat{PE} = \frac{1}{m}\sum_{i=1}^{m}(y_i' - \hat{y}_i')^2$$

- We first <u>choose some points from the steam data set</u> and put them aside **to use as an independent test sample**. *The rest of the observations are then used to obtain the model.*

```
load steam
% Get the set that will be used to
% estimate the line.
indtest = 2:2:20; % Just pick some points.
xtest = x(indtest);
ytest = y(indtest);
% Now get the observations that will be
% used to fit the model.
xtrain = x;
ytrain = y;
% Remove the test observations.
xtrain(indtest) = [];
ytrain(indtest) = [];
```

The next step is to fit a first degree polynomial:

```
% Fit a first degree polynomial (the model)
% to the data.
[p,s] = polyfit(xtrain,ytrain,1);
```

We can use the MATLAB function **polyval** to get the predictions at the x values in the testing set and compare these to the observed y values in the testing set.

```
% Now get the predictions using the model and the
% testing data that was set aside.
yhat = polyval(p,xtest);
% The residuals are the difference between the true
% and the predicted values.
r = (ytest - yhat);
```

Finally, the estimate of the prediction error (equation 8.7) is obtained as follows:

```
pe = mean(r.^2);
```

The estimated prediction error is $\hat{PE} = 0.91$

Toward Cross-Validation

- This was a situation where we partitioned the data into one set for building the model and another for estimating the prediction error
- This is perhaps not the best use of the data, because we have all of the data available for evaluating the error in the model
- We could repeat the above procedure, repeatedly partitioning the data into many training and testing sets
- **This is the fundamental idea underlying cross-validation**

K-Fold Cross-Validation

- The most general form of this procedure is called K-fold cross-validation
- The basic concept is to split the data into K partitions of approximately equalize
- One partition is reserved for testing, and the rest of the data are used for fitting the model
- The test set is used to calculate the squared error (yi – yˆi)2.
- Note that the prediction **yˆi** is from the model obtained using the current training set (one without the i-th observation in it)
- This **procedure is repeated until all K partitions have been used as a test set**
- **Note:** we have n squared errors (because each observation will be a member of one testing set) $\rightarrow$ **The average of these errors is the estimated expected prediction error**

K-Fold Cross-Validation

- In most situations, where the size of the data set is relatively small, the analyst can set K = n , so the size of the testing set is one
- Since this requires fit- ting the model n times, this can be computationally expensive if n is large
- We note, however, that there are efficient ways of doing this [Gentle 1998; Hjorth, 1994]
- We outline the steps for cross-validation below and demonstrate this approach in the following example

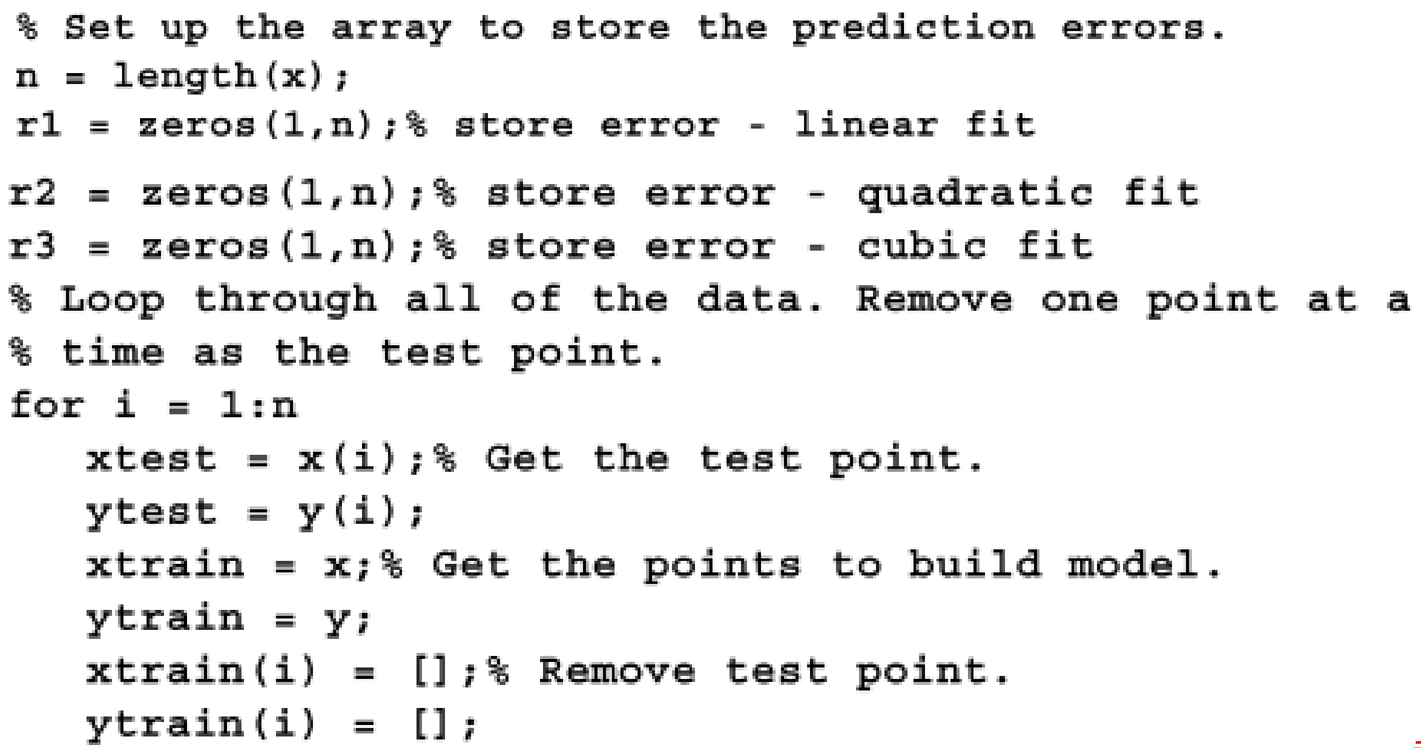
PROCEDURE – CROSS-VALIDATION

1. Partition the data set into K partitions. For simplicity, we assume that $n = r \cdot K$, so there are r observations in each set.
2. Leave out one of the partitions for testing purposes.
3. Use the remaining $n - r$ data points for training (e.g., fit the model, build the classifier, estimate the probability density function, etc.).
4. Use the test set with the model and determine the error between the observed and predicted response.
5. Repeat steps 2 through 4 until all K partitions have been used as a test set.
6. Determine the average of the n errors.

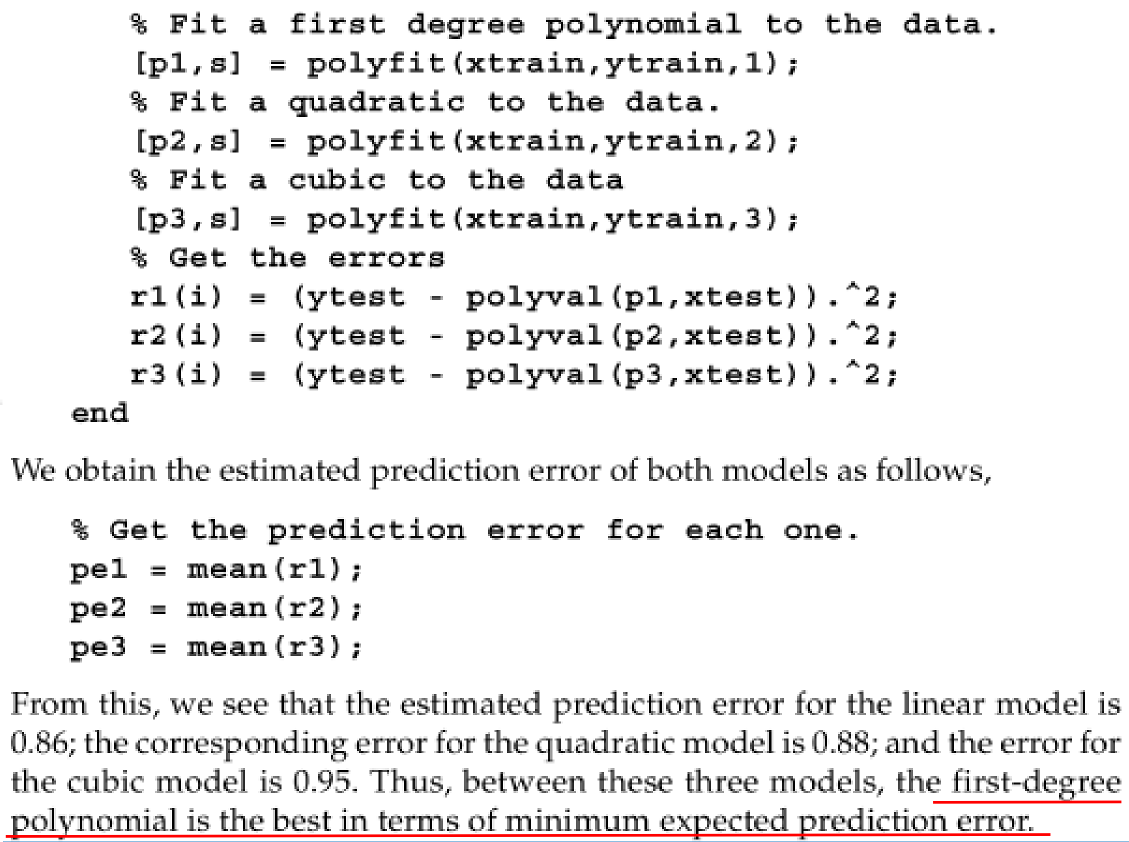
Note that the error mentioned in step 4 depends on the application and the goal of the analysis [Hjorth, 1994]. For example, in pattern recognition applications, this might be the cost of misclassifying a case. In the following example, we apply the cross-validation technique to help decide what type of model should be used for the **`steam`** data.

EXAMPLE

- In this example, we apply cross-validation to the modeling problem of example 8.1
- We fit **linear**, **quadratic** (degree 2) and **cubic** (degree 3) **models** to the data
- We compare their accuracy using the estimates of prediction error obtained from cross-validation



```
% Set up the array to store the prediction errors.
n = length(x);
r1 = zeros(1,n);% store error - linear fit
r2 = zeros(1,n);% store error - quadratic fit
r3 = zeros(1,n);% store error - cubic fit
% Loop through all of the data. Remove one point at a
% time as the test point.
for i = 1:n
   xtest = x(i);% Get the test point.
   ytest = y(i);
   xtrain = x;% Get the points to build model.
   ytrain = y;
   xtrain(i) = [];% Remove test point.
   ytrain(i) = [];
```



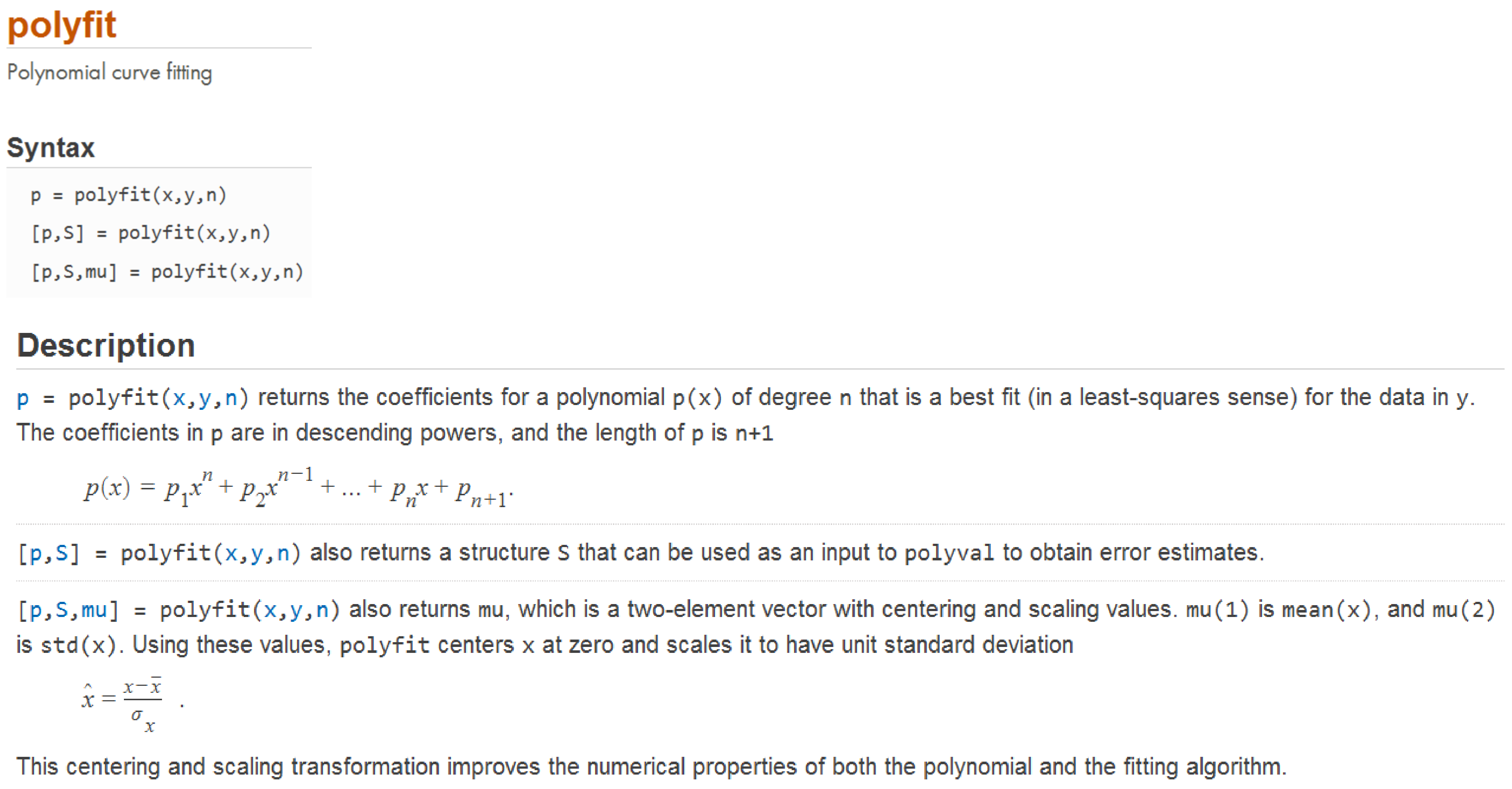
```
   % Fit a first degree polynomial to the data.
   [p1,s] = polyfit(xtrain,ytrain,1);
   % Fit a quadratic to the data.
   [p2,s] = polyfit(xtrain,ytrain,2);
   % Fit a cubic to the data
   [p3,s] = polyfit(xtrain,ytrain,3);
   % Get the errors
   r1(i) = (ytest - polyval(p1,xtest)).^2;
   r2(i) = (ytest - polyval(p2,xtest)).^2;
   r3(i) = (ytest - polyval(p3,xtest)).^2;
end
```

We obtain the estimated prediction error of both models as follows,

```
% Get the prediction error for each one.
pe1 = mean(r1);
pe2 = mean(r2);
pe3 = mean(r3);
```

From this, we see that the estimated prediction error for the linear model is 0.86; the corresponding error for the quadratic model is 0.88; and the error for the cubic model is 0.95. Thus, between these three models, the first-degree polynomial is the best in terms of minimum expected prediction error.

HELP with **polyfit** - Description HERE



polyfit

Polynomial curve fitting

Syntax

```
p = polyfit(x,y,n)
[p,S] = polyfit(x,y,n)
[p,S,mu] = polyfit(x,y,n)
```

Description

`p = polyfit(x,y,n)` returns the coefficients for a polynomial `p(x)` of degree `n` that is a best fit (in a least-squares sense) for the data in `y`. The coefficients in `p` are in descending powers, and the length of `p` is `n+1`

$$p(x) = p_1 x^n + p_2 x^{n-1} + \ldots + p_n x + p_{n+1}.$$

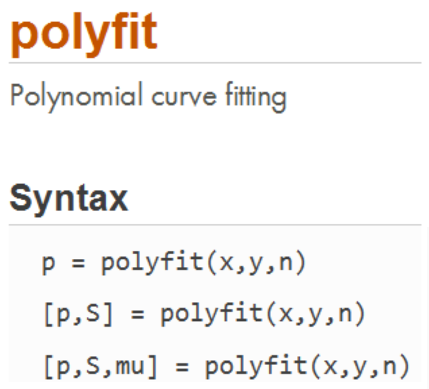
`[p,S] = polyfit(x,y,n)` also returns a structure `S` that can be used as an input to `polyval` to obtain error estimates.

`[p,S,mu] = polyfit(x,y,n)` also returns `mu`, which is a two-element vector with centering and scaling values. `mu(1)` is `mean(x)`, and `mu(2)` is `std(x)`. Using these values, `polyfit` centers `x` at zero and scales it to have unit standard deviation

$$\hat{x} = \frac{x - \bar{x}}{\sigma_x} \quad .$$

This centering and scaling transformation improves the numerical properties of both the polynomial and the fitting algorithm.

HELP with **polyfit** - Output Arguments HERE

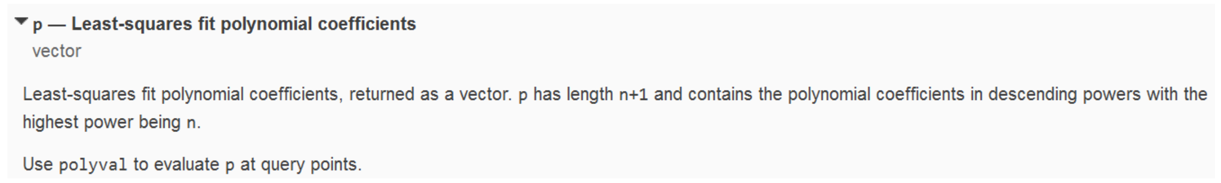


polyfit

Polynomial curve fitting

Syntax

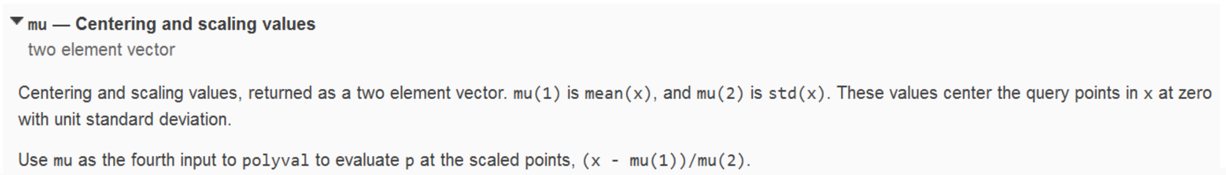
```
p = polyfit(x,y,n)
[p,S] = polyfit(x,y,n)
[p,S,mu] = polyfit(x,y,n)
```



p — Least-squares fit polynomial coefficients
vector

Least-squares fit polynomial coefficients, returned as a vector. `p` has length `n+1` and contains the polynomial coefficients in descending powers with the highest power being `n`.

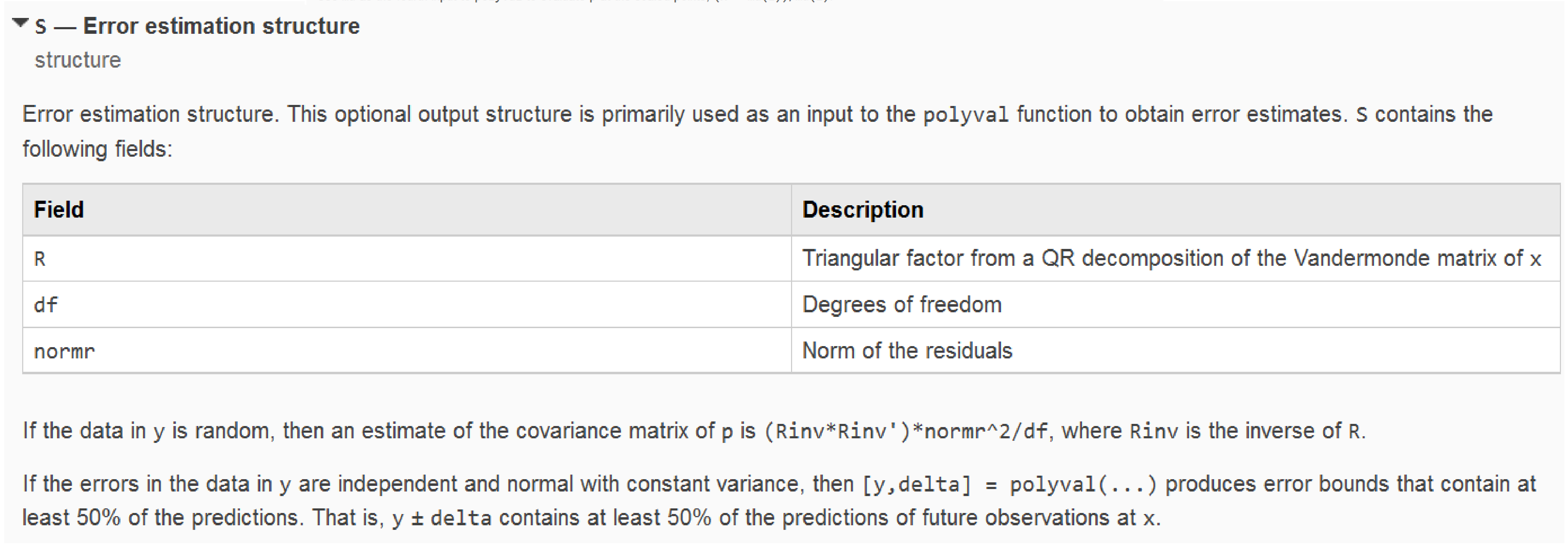
Use `polyval` to evaluate `p` at query points.



mu — Centering and scaling values
two element vector

Centering and scaling values, returned as a two element vector. `mu(1)` is `mean(x)`, and `mu(2)` is `std(x)`. These values center the query points in `x` at zero with unit standard deviation.

Use `mu` as the fourth input to `polyval` to evaluate `p` at the scaled points, `(x - mu(1))/mu(2)`.



S — Error estimation structure
structure

Error estimation structure. This optional output structure is primarily used as an input to the `polyval` function to obtain error estimates. `S` contains the following fields:

Field	Description
`R`	Triangular factor from a QR decomposition of the Vandermonde matrix of `x`
`df`	Degrees of freedom
`normr`	Norm of the residuals

If the data in `y` is random, then an estimate of the covariance matrix of `p` is `(Rinv*Rinv')*normr^2/df`, where `Rinv` is the inverse of `R`.

If the errors in the data in `y` are independent and normal with constant variance, then `[y,delta] = polyval(...)` produces error bounds that contain at least 50% of the predictions. That is, `y` ± `delta` contains at least 50% of the predictions of future observations at `x`.

Jackknife - Data Partitioning Method

- It is like cross-validation, another data partitioning method
- However, the **jackknife method** is <u>used to estimate the bias and the standard error</u> of statistics

Let's say that we have a random sample of size n, and we denote our estimator of a parameter θ as:

$$\hat{\theta} = T = t(x_1, x_2, \ldots, x_n) \qquad (8.8)$$

θ might be the mean, the variance, the correlation coefficient, or some other statistic of interest

- **T** is also a random variable (has some error associated with it)

<u>Goal</u>: *get an estimate of the bias and the standard error of the estimate T to assess the accuracy of the results*

Jackknife - Data Partitioning Method

$$\hat{\theta} = T = t(x_1, x_2, \ldots, x_n) \qquad (8.8)$$

When we **cannot determine** the bias and the standard error **using analytical techniques**, then methods such as the **jackknife** may be used

Jackknife:

- No parametric assumptions are made about the *underlying population that generated the data*
- The variation *in the estimate* is investigated by looking at the sample data

Jackknife - Data Partitioning Method

$$\hat{\theta} = T = t(x_1, x_2, \dots, x_n) \qquad (8.8)$$

The jackknife method is similar to cross-validation in that we leave out one observation x_i from our sample to form a ***jackknife sample*** as follows

$$x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n .$$

This says that the *i*-th jackknife sample is the original sample with the *i*-th data point removed.

Jackknife - Data Partitioning Method

$$\hat{\theta} = T = t(x_1, x_2, \ldots, x_n) \tag{8.8}$$

We calculate the value of the estimate using this reduced jackknife sample to obtain the i-th ***jackknife replicate***. This is given by

$$T^{(-i)} = t(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n).$$

This means that we leave out one point at a time and use the rest of the sample to calculate our statistic. We continue to do this for the entire sample, leaving out one observation at a time, and the end result is a sequence of n jackknife replications of the statistic.

Jackknife - Data Partitioning Method

$$\hat{\theta} = T = t(x_1, x_2, \ldots, x_n) \qquad (8.8)$$

$$T^{(-i)} = t(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$$

The estimate of the bias of T obtained from the jackknife technique is given by [Efron and Tibshirani, 1993]

$$\hat{\text{Bias}}_{Jack}(T) = (n-1)(\overline{T^{(J)}} - T), \qquad (8.9)$$

where

$$\overline{T^{(J)}} = \frac{\sum_{i=1}^{n} T^{(-i)}}{n}. \qquad (8.10)$$

We see from equation 8.10 that $\overline{T^{(J)}}$ is simply the average of the jackknife replications of T

Jackknife - Data Partitioning Method

The **estimated standard error** using the jackknife is defined as follows:

$$\hat{SE}_{Jack}(T) = \left[\frac{n-1}{n}\sum_{i=1}^{n}\left(T^{(-i)} - \overline{T^{(J)}}\right)^2\right]^{1/2}. \qquad (8.11)$$

Equation 8.11 is essentially the sample standard deviation of the jackknife replications with a factor $(n-1)/n$ in front of the summation instead of $1/(n-1)$. Efron and Tibshirani [1993] show that this factor ensures that the jackknife estimate of the standard error of the sample mean, $\hat{SE}_{Jack}(\bar{x})$, is an unbiased estimate.

Jackknife - Data Partitioning Method

PROCEDURE – JACKKNIFE

1. Leave out an observation.
2. Calculate the value of the statistic using the remaining sample points to obtain $T^{(-i)}$.
3. Repeat steps 1 and 2, leaving out one point at a time, until all n $T^{(-i)}$ are recorded.
4. Calculate the jackknife estimate of the bias of T using equation 8.9.
5. Calculate the jackknife estimate of the standard error of T using equation 8.11.

$$\widehat{\text{Bias}}_{Jack}(T) = (n-1)(\overline{T^{(J)}} - T), \tag{8.9}$$

$$\widehat{SE}_{Jack}(T) = \left[\frac{n-1}{n}\sum_{i=1}^{n}(T^{(-i)} - \overline{T^{(J)}})^2\right]^{1/2}. \tag{8.11}$$

Jackknife - Data Partitioning Method

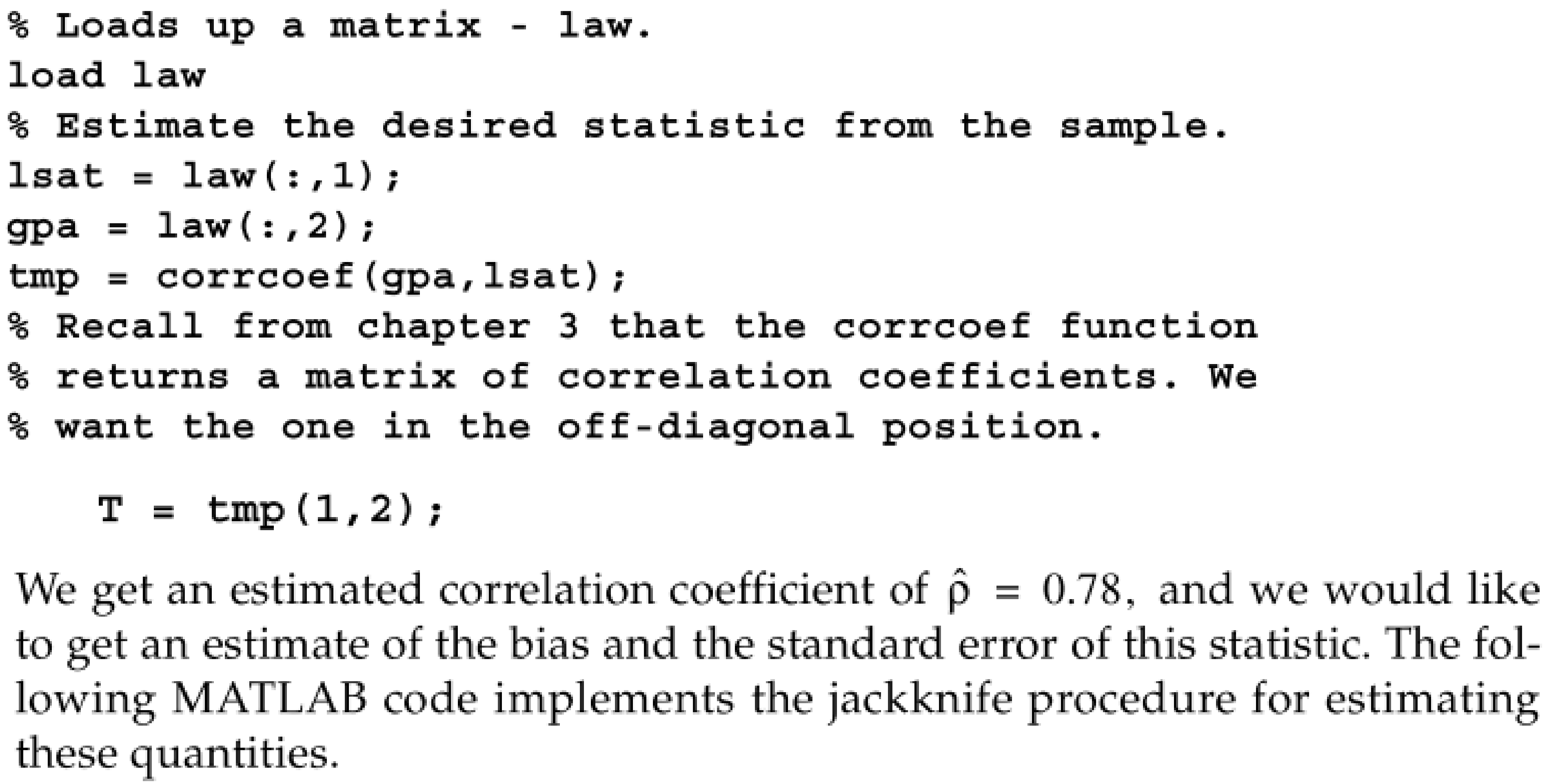
The following two examples show how this is used to obtain jackknife estimates of the bias and standard error for an estimate of the correlation coefficient:

Example 1

- We have a data set that has been examined in Efron and Tibshirani [1993]
- These data consist of measurements collected on the freshman class of 82 law schools in 1973
- The **average score** for the entering class on a national law test (**lsat**) and the **undergraduate grade point average** (**gpa**) were recorded
- A random sample of size n = 15 was taken from the population
- We would like to use these sample data to estimate the correlation coefficient ρ between the test scores (**lsat**) and the grade point average (**gpa**)

Jackknife - Data Partitioning Method

We start off by finding the statistic of interest:



```
% Loads up a matrix - law.
load law
% Estimate the desired statistic from the sample.
lsat = law(:,1);
gpa = law(:,2);
tmp = corrcoef(gpa,lsat);
% Recall from chapter 3 that the corrcoef function
% returns a matrix of correlation coefficients. We
% want the one in the off-diagonal position.

   T = tmp(1,2);
```

We get an estimated correlation coefficient of $\hat{\rho} = 0.78$, and we would like to get an estimate of the bias and the standard error of this statistic. The following MATLAB code implements the jackknife procedure for estimating these quantities.

Jackknife - Data Partitioning Method

We start off by finding the statistic of interest:

```
% Set up memory for jackknife replicates.
n = length(gpa);
reps = zeros(1,n);
for i = 1:n
   % Store as temporary vector:
   gpat = gpa;
   lsatt = lsat;
   % Leave i-th point out:
   gpat(i) = [];
   lsatt(i) = [];
   % Get correlation coefficient:
   % In this example, we want off-diagonal element.
   tmp = corrcoef(gpat,lsatt);
   reps(i) = tmp(1,2);
end
```

```
mureps = mean(reps);
sehat = sqrt((n-1)/n*sum((reps-mureps).^2));
% Get the estimate of the bias:
biashat = (n-1)*(mureps-T);
```

Our estimate of the standard error of the sample correlation coefficient is

$$\hat{\mathrm{SE}}_{Jack}(\hat{\rho}) = 0.14\,,$$

and our estimate of the bias is

$$\hat{\mathrm{Bias}}_{Jack}(\hat{\rho}) = -0.0065\,.$$

Example 2

- There is a MATLAB function called **csjack** that implements the jack- knife procedure
- **csjack** will work with any MATLAB function that takes the random sample as the argument and returns a statistic.
- **csjack** can be one that comes with MATLAB, such as **mean** or **var**, or it can be one written by the user.

We illustrate its use with a user-written function called **corr** that *returns the single correlation coefficient between two univariate random variables*

```
function r = cscorr(data)
% This function returns the single correlation
% coefficient between two variables.
tmp = corrcoef(data);
r = tmp(1,2);
```

The data used in this example are taken from Hand, et al. [1994]. They were originally from Anscombe [1973], where they were created to illustrate the point that even though an observed value of a statistic is the same for data sets ($\hat{\rho} = 0.82$), that does not tell the entire story. He also used them to show the importance of looking at scatterplots, because it is obvious from the plots that the relationships between the variables are not similar. The scatterplots are shown in figure 8.3.

```
% Here is another example.
% We have 4 data sets with essentially the same
% correlation coefficient.
% The scatterplots look very different.
% When this file is loaded, you get four sets
% of x and y variables.
load anscombe
% Do the scatterplots.
subplot(2,2,1),plot(x1,y1,'k*');
subplot(2,2,2),plot(x2,y2,'k*');
subplot(2,2,3),plot(x3,y3,'k*');
subplot(2,2,4),plot(x4,y4,'k*');
```

We now determine the jackknife estimate of bias and standard error for $\hat{\rho}$ using **csjack**.

```
% Note that 'corr' is something we wrote.
[b1,se1,jv1] = csjack([x1,y1],'cscorr');
[b2,se2,jv2] = csjack([x2,y2],'cscorr');
[b3,se3,jv3] = csjack([x3,y3],'cscorr');
[b4,se4,jv4] = csjack([x4,y4],'cscorr');
```

The jackknife estimates of bias are:

```
b1 = -0.0052
b2 =  0.0008
b3 =  0.1514
b4 =  NaN
```

The jackknife estimates of the standard error are:

```
se1 = 0.1054
se2 = 0.1026
se3 = 0.1730
se4 = NaN
```

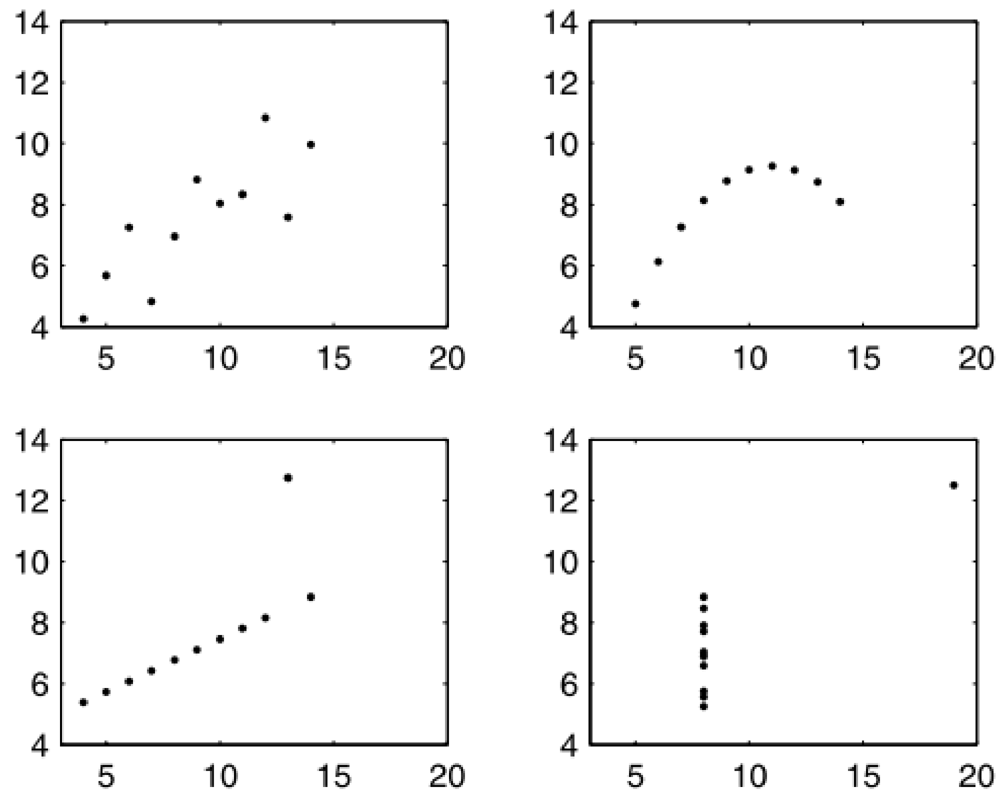


FIGURE 8.3
This shows the scatterplots of the four data sets discussed in example 8.5. These data were created to show the importance of looking at scatterplots [Anscombe, 1973]. All data sets have the same estimated correlation coefficient of $\hat{\rho} = 0.82$, but it is obvious that the relationship between the variables is very different.

Note that the jackknife procedure does not work for the fourth data set, because when we leave out the last data point, the correlation coefficient is undefined for the remaining points.

The jackknife method is also described in the literature using pseudo-values. The ***jackknife pseudo-values*** are given by

$$\widehat{T}_i = nT - (n-1)T^{(-i)} \qquad i = 1, \ldots, n, \tag{8.12}$$

where $T^{(-i)}$ is the value of the statistic computed on the sample with the *i*-th data point removed.

We take the average of the pseudo-values given by

$$J(T) = \frac{\sum_{i=1}^{n} \widehat{T}_i}{n}, \tag{8.13}$$

and use this to get the jackknife estimate of the standard error, as follows

$$\hat{SE}_{JackP}(T) = \left[\frac{1}{n(n-1)}\sum_{i=1}^{n}(\widetilde{T}_i - J(T))^2\right]^{1/2} . \tag{8.14}$$

PROCEDURE – PSEUDO-VALUE JACKKNIFE

1. Leave out an observation.
2. Calculate the value of the statistic using the remaining sample points to obtain $T^{(-i)}$.
3. Calculate the pseudo-value $\widetilde{T}_i$ using equation 8.12.
4. Repeat steps 2 and 3 for the remaining data points, yielding n values of $\widetilde{T}_i$.
5. Determine the jackknife estimate of the standard error of T using equation 8.14.

Example 8.6

We now repeat example 8.4 using the jackknife pseudo-value approach and compare estimates of the standard error of the correlation coefficient for these data. The following MATLAB code implements the pseudo-value procedure.

```
% Loads up a matrix.
load law
lsat = law(:,1);
gpa = law(:,2);
% Get the statistic from the original sample
tmp = corrcoef(gpa,lsat);
T = tmp(1,2);
% Set up memory for jackknife replicates
n = length(gpa);
reps = zeros(1,n);
for i = 1:n
   % store as temporary vector
   gpat = gpa;
   lsatt = lsat;
   % leave i-th point out
   gpat(i) = [];
   lsatt(i) = [];
   % get correlation coefficient
   tmp = corrcoef(gpat,lsatt);
   % In this example, is off-diagonal element.
   % Get the jackknife pseudo-value for the i-th point.
   reps(i) = n*T-(n-1)*tmp(1,2);
end
JT = mean(reps);
sehatpv = sqrt(1/(n*(n-1))*sum((reps - JT).^2));
```

We obtain an estimated standard error of $\hat{SE}_{JackP}(\hat{\rho}) = 0.14$, which is the same result we had before.
❑

Efron and Tibshirani [1993] describe a situation where the jackknife procedure does not work and suggest that the bootstrap be used instead. These are applications where the statistic is not smooth. An example of this type of statistic is the median. Here *smoothness* refers to statistics where small changes in the data set produce small changes in the value of the statistic. We illustrate this situation in the next example.

Very Important

When it does not work

The Computational Statistics Toolbox that you can use to run

some code/functions of this lecture can be downloaded from:

- http://lib.stat.cmu.edu
- http://www.crcpress.com/e_products/downloads/
- http://www.pi-sigma.info/