Tutorial: A Module for Linear Regression

Chapter Table of Contents

OVERVIEW ........................................... 23
   Solving a System of Equations .......................... 23
A MODULE FOR LINEAR REGRESSION ..................... 25
PLOTTING REGRESSION RESULTS .......................... 31
SUMMARY ........................................... 34
Chapter 3
Tutorial: A Module for Linear Regression

Overview

SAS/IML software makes it possible for you to solve mathematical problems or implement new statistical techniques and algorithms. The language is patterned after linear algebra notation. For example, the least-squares formula familiar to statisticians

\[ B = (X'X)^{-1}X'Y \]

can be easily translated into the Interactive Matrix Language statement

```plaintext
b=inv(x'x)*x'y;
```

This is an example of an assignment statement that uses a built-in function (INV) and operators (transpose and matrix multiplication).

If a statistical method has not been implemented directly in a SAS procedure, you may be able to program it using IML. Because the operations in IML deal with arrays of numbers rather than with one number at a time, and the most commonly used mathematical and matrix operations are built directly into the language, programs that take hundreds of lines of code in other languages often take only a few lines in IML.

Solving a System of Equations

Because IML is built around traditional matrix algebra notation, it is often possible to directly translate mathematical methods from matrix algebraic expressions into executable IML statements. For example, consider the problem of solving three simultaneous equations:

\[
\begin{align*}
3x_1 - x_2 + 2x_3 &= 8 \\
2x_1 - 2x_2 + 3x_3 &= 2 \\
4x_1 + x_2 - 4x_3 &= 9
\end{align*}
\]

These equations can be written in matrix form as

\[
\begin{bmatrix}
3 & -1 & 2 \\
2 & -2 & 3 \\
4 & 1 & -4
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3
\end{bmatrix}
=
\begin{bmatrix}
8 \\
2 \\
9
\end{bmatrix}
\]
and can be expressed symbolically as

\[ Ax = c \]

Because \( A \) is nonsingular, the system has a solution given by

\[ x = A^{-1}c \]

In the following example, you solve this system of equations using an interactive session. Submit the PROC IML statement to begin the procedure. Throughout this chapter, the right angle brackets (\( > \)) indicate statements you submit; responses from IML follow:

```
proc iml;
  IML Ready
Enter
  reset print;
```

The PRINT option of the RESET command causes automatic printing of results. Notice that as you submit each statement, it is executed and the results are displayed. While you are learning IML or developing modules, it is a good idea to have all results printed automatically. Once you are familiar with SAS/IML software, you will not need to use automatic printing.

Next, set up the matrices \( A \) and \( c \). Both of these matrices are input as matrix literals; that is, input the row and column values as discussed in Chapter 2, “Understanding the Language.”

```
> a={3  -1  2,
   >    2  -2  3,
   >    4   1 -4};

A 3 rows 3 cols (numeric)
   3   -1   2
   2   -2   3
   4    1  -4

> c={8, 2, 9};

C 3 rows 1 col (numeric)
   8
   2
   9
```
Now write the solution equation, \( x = A^{-1}c \), as an IML statement. The appropriate statement is an assignment statement that uses a built-in function and an operator (INV is a built-in function that takes the inverse of a square matrix, and * is the operator for matrix multiplication).

```plaintext
> x=inv(a)*c;
```

After IML executes the statement, the first row of matrix \( X \) contains the \( x_1 \) value for which you are solving, the second row contains the \( x_2 \) value, and the third row contains the \( x_3 \) value.

Now end the session by entering the QUIT command.

```plaintext
> quit;
```

**A Module for Linear Regression**

The previous method may be more familiar to statisticians when different notation is used. A linear model is usually written

\[
y = Xb + e
\]

where \( y \) is the vector of responses, \( X \) is the design matrix, and \( b \) is a vector of unknown parameters estimated by minimizing the sum of squares of \( e \), the error or residual.

The following example illustrates the programming techniques involved in performing linear regression. It is not meant to replace regression procedures such as the REG procedure, which are more efficient for regressions and offer a multitude of diagnostic options.

Suppose that you have response data \( y \) measured at five values of the independent variable \( x \) and you want to perform a quadratic regression.

Submit the PROC IML statement to begin the procedure.

```plaintext
> proc iml;
```

Input the design matrix \( X \) and the data vector \( y \) as matrix literals.
Chapter 3. Tutorial: A Module for Linear Regression

```r
> x = c(1, 1, 1, 2, 4, 3, 9, 4, 16, 5, 25);
> y = c(1, 5, 9, 23, 36);

> b = solve(x %*% x) %*% x %*% y;

> yhat = x %*% b;

> r = y - yhat;
```

X 5 rows 3 cols (numeric)

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>25</td>
</tr>
</tbody>
</table>

Y 5 rows 1 col (numeric)

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>23</td>
</tr>
<tr>
<td>36</td>
</tr>
</tbody>
</table>

Compute the least-squares estimate of $b$ using the traditional formula.

The predicted values are simply the $X$ matrix multiplied by the parameter estimates, and the residuals are the difference between actual and predicted $y$.
To calculate the estimate of the variance of the responses, calculate the sum of squared errors (SSE), its degrees of freedom (DFE), and the mean squared error (MSE). Note that in computing the degrees, you use the function NCOL to return the number of columns of $X$.

```plaintext
> sse=ssq(r);

SSE 1 row 1 col (numeric)

6.4

> dfe=nrow(x)-ncol(x);

DFE 1 row 1 col (numeric)

2

> mse=sse/dfe;

MSE 1 row 1 col (numeric)

3.2
```

Notice that each calculation has required one simple line of code.

Now suppose you want to solve the problem repeatedly on new data sets without reentering the code. To do this, define a module (or subroutine). Modules begin with a START statement and end with a FINISH statement, with the program statements in between. The following statements define a module named REGRESS to perform linear regression.

```plaintext
> start regress; /* begin module */
> xpxi=inv(t(x)*x); /* inverse of X'X */
> beta=xpxi*(t(x)*y); /* parameter estimate */
> yhat=x*beta; /* predicted values */
> resid=y-yhat; /* residuals */
> sse=ssq(resid); /* SSE */
> n=nrow(x); /* sample size */
> dfe=nrow(x)-ncol(x); /* error DF */
> mse=sse/dfe; /* MSE */
> cssy=ssq(y-sum(y)/n); /* corrected total SS */
> rsquare=(cssy-sse)/cssy; /* RSQUARE */
> print,"Regression Results",
    sse dfe mse rsquare;
> stdb=sqrt(vecdiag(xpxi)*mse); /* std of estimates */
```
Chapter 3. Tutorial: A Module for Linear Regression

```sas
> t=beta/stdb; /* parameter t-tests */
> prob=1-probf(t#t,1,dfe); /* p-values */
> print,"Parameter Estimates",
>     beta stdb t prob;
> print,y yhat resid;
> finish regress; /* end module */
```

Submit the module REGRESS for execution.

```sas
> reset noprint;
> run regress; /* execute module */
```

Regression Results

<table>
<thead>
<tr>
<th>SSE</th>
<th>DFE</th>
<th>MSE</th>
<th>RSQUARE</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.4</td>
<td>2</td>
<td>3.2</td>
<td>0.9923518</td>
</tr>
</tbody>
</table>

Parameter Estimates

<table>
<thead>
<tr>
<th>BETA</th>
<th>STDB</th>
<th>T</th>
<th>PROB</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.4</td>
<td>3.8366652</td>
<td>0.6255432</td>
<td>0.5954801</td>
</tr>
<tr>
<td>-3.2</td>
<td>2.9237940</td>
<td>-1.094468</td>
<td>0.3879690</td>
</tr>
<tr>
<td>2</td>
<td>0.4780914</td>
<td>4.1833001</td>
<td>0.0526691</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Y</th>
<th>YHAT</th>
<th>RESID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.2</td>
<td>-0.2</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>10.8</td>
<td>-1.8</td>
</tr>
<tr>
<td>23</td>
<td>21.6</td>
<td>1.4</td>
</tr>
<tr>
<td>36</td>
<td>36.4</td>
<td>-0.4</td>
</tr>
</tbody>
</table>

At this point, you still have all of the matrices defined if you want to continue calculations. Suppose that you want to correlate the estimates. First, calculate the covariance estimate of the estimates; then, scale the covariance into a correlation matrix with values of 1 on the diagonal.

```sas
> reset print; /* turn on auto printing */
> covb=xpxi*mse; /* covariance of estimates */
> s=1/sqrt(vecdiag(covb));
```

S

<table>
<thead>
<tr>
<th>COVB</th>
<th>3 rows</th>
<th>3 cols</th>
<th>(numeric)</th>
</tr>
</thead>
<tbody>
<tr>
<td>14.72</td>
<td>-10.56</td>
<td>1.6</td>
<td></td>
</tr>
<tr>
<td>-10.56</td>
<td>8.5485714</td>
<td>-1.371429</td>
<td></td>
</tr>
<tr>
<td>1.6</td>
<td>-1.371429</td>
<td>0.2285714</td>
<td></td>
</tr>
</tbody>
</table>

SAS OnlineDoc™: Version 8
Your module REGRESS remains available to do another regression, in this case, an orthogonalized version of the last polynomial example. In general, the columns of \( X \) will not be orthogonal. You can use the ORPOL function to generate orthogonal polynomials for the regression. Using them provides greater computing accuracy and reduced computing times. When using orthogonal polynomial regression, you expect the statistics of fit to be the same and the estimates to be more stable and uncorrelated.

To perform an orthogonal regression on the data, you must first create a vector containing the values of the independent variable \( x \), which is the second column of the design matrix \( X \). Then, use the ORPOL function to generate orthogonal second-degree polynomials.

\[
\texttt{x1=\{1,2,3,4,5\};} \quad \text{/* second column of } X \text{ */}
\]

\[
\texttt{x=orpol(x1,2);} \quad \text{/* generates orthogonal polynomials */}
\]

> reset noprint; /* turns off auto printing */
> run regress; /* run REGRESS */
Regression Results

<table>
<thead>
<tr>
<th></th>
<th>SSE</th>
<th>DFE</th>
<th>MSE</th>
<th>RSQUARE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>6.4</td>
<td>2</td>
<td>3.2</td>
<td>0.9923518</td>
</tr>
</tbody>
</table>

Parameter Estimates

<table>
<thead>
<tr>
<th></th>
<th>BETA</th>
<th>STDB</th>
<th>T</th>
<th>PROB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>33.093806</td>
<td>1.7888544</td>
<td>18.5</td>
<td>0.0029091</td>
</tr>
<tr>
<td></td>
<td>27.828043</td>
<td>1.7888544</td>
<td>15.556349</td>
<td>0.0041068</td>
</tr>
<tr>
<td></td>
<td>7.4833148</td>
<td>1.7888544</td>
<td>4.1833001</td>
<td>0.0526691</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Y</th>
<th>YHAT</th>
<th>RESID</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>1.2</td>
<td>-0.2</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>10.8</td>
<td>-1.8</td>
</tr>
<tr>
<td></td>
<td>23</td>
<td>21.6</td>
<td>1.4</td>
</tr>
<tr>
<td></td>
<td>36</td>
<td>36.4</td>
<td>-0.4</td>
</tr>
</tbody>
</table>

> reset print;
> covb=xpxi*mse;

COVB 3 rows 3 cols (numeric)

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2</td>
<td>-2.73E-17</td>
<td>4.693E-16</td>
<td></td>
</tr>
<tr>
<td>-2.73E-17</td>
<td>3.2</td>
<td>-2.18E-15</td>
<td></td>
</tr>
<tr>
<td>4.693E-16</td>
<td>-2.18E-15</td>
<td>3.2</td>
<td></td>
</tr>
</tbody>
</table>

> s=1/sqrt(vecdiag(covb));

S 3 rows 1 col (numeric)

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.559017</td>
</tr>
<tr>
<td>0.559017</td>
</tr>
<tr>
<td>0.559017</td>
</tr>
</tbody>
</table>

> corrb=diag(s)*covb*diag(s);

CORRB 3 rows 3 cols (numeric)

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-8.54E-18</td>
<td>1.467E-16</td>
</tr>
<tr>
<td>-8.54E-18</td>
<td>1</td>
<td>-6.8E-16</td>
</tr>
<tr>
<td>1.467E-16</td>
<td>-6.8E-16</td>
<td>1</td>
</tr>
</tbody>
</table>

Note that the values on the off-diagonal are displayed in scientific notation; the values are close to zero but not exactly zero because of the imprecision of floating-point
arithmetic. To clean up the appearance of the correlation matrix, use the FUZZ option.

```sas
> reset fuzz;
> corrb=diag(s)*covb*diag(s);
```

<table>
<thead>
<tr>
<th>CORRB</th>
<th>3 rows</th>
<th>3 cols</th>
<th>(numeric)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

## Plotting Regression Results

You can create some simple plots by using the PGRAF subroutine. The PGRAF subroutine produces scatter plots suitable for printing on a line printer. If you want to produce better quality graphics using color, you can use the graphics capabilities of IML (see Chapter 12, “Graphics Examples,” for more information).

Here is how you can plot the residuals against $x$. First, create a matrix containing the pairs of points by concatenating $X_1$ with RESID using the horizontal concatenation operator ($||$).

```sas
> xy=x1||resid;
```

<table>
<thead>
<tr>
<th>XY</th>
<th>5 rows</th>
<th>2 cols</th>
<th>(numeric)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-0.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>-1.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>-0.4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Next, use a CALL statement to call the PGRAF subroutine to produce the desired plot. The arguments to PGRAF are, in order,

- the matrix containing the pairs of points
- a plotting symbol
- a label for the X-axis
- a label for the Y-axis
- a title for the plot
You can also plot the predicted values $\hat{y}$ against $x$. You must first create a matrix, say XYH, containing the points. Do this by concatenating X1 with YHAT. Next, call the PGRAF subroutine to plot the points.

```
> xyh=x1||yhat;
```

```
XYH  5 rows 2 cols (numeric)
1   1.2
2   4
3  10.8
4  21.6
5  36.4
```

```
> call pgraf(xyh,'*','x','Predicted','Plot of Predicted Values');
```
Plotting Regression Results

You can get a more detailed plot, denoting the observed values with a “y” and the predicted values with a “p” using the following statements. Create a matrix `NEWXY` containing the pairs of points to overlay. You need to use both the horizontal concatenation operator (||) and the vertical concatenation operator (//). The NROW function returns the number of observations, that is, the number of rows of `X1`. The matrix `LABEL` contains the character label for each point, plotting a “y” for each observed point and a “p” for each predicted point.

```sas
> newxy=(x1//x1)||(y//yhat);
```

<table>
<thead>
<tr>
<th>NEWXY</th>
<th>10 rows</th>
<th>2 cols</th>
<th>(numeric)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>23</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>10.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>21.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>36.4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```sas
> n=nrow(x1);
```

<table>
<thead>
<tr>
<th>N</th>
<th>1 row</th>
<th>1 col</th>
<th>(numeric)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```sas
> label=repeat('y',n,1)//repeat('p',n,1);
```
> call pgraf(newxy,label,’x’,’y’,’Scatter Plot with Regression Line’);

As you can see, the observed and predicted values are too close together to be able to
distinguish them at all values of \( x \).

**Summary**

In this chapter, you have seen the programming techniques necessary for solving
systems of equations. You have seen how to define a module for performing
linear regression and obtaining covariance and correlation matrices, and how to obtain
some simple diagnostic plots. Many of the ideas presented in Chapter 2, “Under-
standing the Language,” such as the use of assignment statements, functions, CALL
statements, and subscripting have been demonstrated.


Copyright © 1999 by SAS Institute Inc., Cary, NC, USA.

ISBN 1–58025–553–1

All rights reserved. Printed in the United States of America. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

**U.S. Government Restricted Rights Notice.** Use, duplication, or disclosure of the software by the government is subject to restrictions as set forth in FAR 52.227–19 Commercial Computer Software-Restricted Rights (June 1987).


1st printing, October 1999

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries.® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

The Institute is a private company devoted to the support and further development of its software and related services.