

Applied Regression Modeling: A Business Approach

Computer software help: R and S-PLUS

R is a free software environment for statistical computing and graphics—further information is available at www.r-project.org. Although graphical user-interfaces do exist for R, they are still at an early stage of development at the time of writing, so the instructions here relate to its programming interface. The commercial package S-PLUS (available at www.insightful.com/products/splus/) has many features in common with R, and so many of these instructions will also work in S-PLUS. S-PLUS also has a good graphical user-interface, but that is not addressed here. More extensive guidance on the use of R and S-PLUS for regression modeling can be found in Fox (2002) and Venables and Ripley (2002). The following instructions are based on “R 2.2.1 for Windows.” The book website contains supplementary material for other versions of R and S-PLUS.

Getting started and summarizing univariate data

- 1 Change R’s default **options** by selecting Edit > GUI Preferences.

For example, you may find it easier to select SDI for Single or multiple windows and multiple windows for Pager style.

To open a **R data file**, type `mydata <- read.table("file", header=TRUE, sep="\t")`, where `file` is the name of the data file, `header` is a logical value indicating whether the file contains the names of the variables as its first line, and `sep` is the field separator character (e.g., `"\t"` for tab-separated files).

One way to then make the dataset available for analysis is to type `attach(mydata)`. You will now be able to analyze each of the variables in the `mydata` dataset. To see what these variables are, type `names(mydata)`. When you are finished with this dataset, type `detach(mydata)`. We do not go into the other methods for working with R datasets here—see an R companion book such as Fox (2002) for further discussion of this.

To **recall** a previously entered command, hit the “up” arrow (repeatedly) on your keyboard.

Output appears in the main R Console Window and can be copied and pasted from R to a word processor like Microsoft Word. Graphs appear in separate windows and can also easily be copied and pasted to other applications.

- 2 You can access **help** by selecting Help > Html help.

For example, to find out about “boxplots” click Search Engine & Keywords, type `boxplots` in the search box, and click on one of the resulting hits.

- 3 To **transform data** or compute a **new variable**, type, for example, `logX <- log(X)` for the natural logarithm of X and `Xsq <- X**2` for X^2 . If you get the error message “syntax error in . . .,” this means there is a syntax error in your expression—a common mistake is to forget the multiplication symbol (`*`) between a number and a variable (e.g., `2*X` represents $2X$).

Created variables like this will be placed in R’s “global environment.” To see a list of the variables in the global environment, type `ls()`. Variables in the global environment

take precedence over identically named variables in attached datasets. To avoid any confusion, avoid creating a variable (or other R “object”) that has the same name as a variable in an attached dataset.

To create **indicator (dummy) variables** from a qualitative variable, type, for example, `D1 <- ifelse(X=="level", yes=1, no=0)`, where `X` is the qualitative variable and “level” is the name of one of the categories in `X`. Repeat for other indicator variables (if necessary).

- 4 Calculate **descriptive statistics** for quantitative variables by typing `summary(Y)`, where `Y` is the quantitative variable.

Other, more specific commands include `mean(Y)`, `median(Y)`, `sd(Y)`, `min(Y)`, `max(Y)`, and `quantile(Y, probs=c(.25,.5,.75), type=7)`, where `probs` is a numeric vector of probabilities with values in `[0, 1]`, and `type` is an integer between 1 and 9 selecting one of nine quantile (percentile) algorithms (the default algorithm 7 gives slightly different results than algorithm 6 used by SPSS and Minitab).

- 5 Create **contingency tables** or **cross-tabulations** for qualitative variables by typing `table(X1, X2)`, where `X1` and `X2` are the qualitative variables.

Calculate row sums by typing `rowsum <- apply(table(X1, X2), 1, sum)`. Then the row percentages are `100*table(X1, X2)/rowsum`.

Calculate column sums by typing `colsum <- apply(table(X1, X2), 2, sum)`. Then column percentages are `100*t(t(table(X1, X2))/colsum)` (the function `t()` stands for “transpose”).

- 6 If you have a quantitative variable and a qualitative variable, you can calculate **descriptive statistics** for cases grouped in different categories by typing, for example, `aggregate(Y, by, FUN)`, where `Y` is the quantitative variable, `by` is a list of the qualitative variables (e.g., `list(X1=X1, X2=X2)`), and `FUN` is the name of the function for calculating the descriptive statistic (e.g., `mean`).

- 7 To make a **stem-and-leaf plot** for a quantitative variable, type `stem(Y, scale=1)`, where `Y` is the quantitative variable and `scale` controls the plot length.

To make a **histogram** for a quantitative variable, type `hist(Y, freq=FALSE, breaks=c(1,2,3,4))`, where `Y` is the quantitative variable, `freq` is a logical value (if `TRUE`, the histogram represents frequencies, if `FALSE`, it represents probability densities), and `breaks` specifies how to construct the breakpoints.

- 8 To make a **scatterplot** with two quantitative variables, type `plot(X, Y)`, where `X` is the horizontal axis variable and `Y` is the vertical axis variable.

All possible scatterplots for more than two variables can be drawn simultaneously (called a **scatterplot matrix**) by typing `pairs(cbind(Y, X1, X2))`, where `Y`, `X1`, and `X2` are quantitative variables.

- 9 You can **mark or label cases** in a scatterplot with different colors/symbols according to categories in a qualitative variable by using the `points()` function. For example, in a dataset of 20 observations, suppose `X2` contains values 1–4 to represent four categories, and `Y` and `X1` are two quantitative variables. Then the following code produces a scatterplot with numbers (representing the value of `X2`) marking the points:

```
plot(X1, Y, type="n")
for (i in 1:20) points(X1[i], Y[i], pch=as.character(X2[i])).
```

You can also **identify individual cases** after drawing a scatterplot by typing `identify(X, Y, labels=row.names(mydata))`, where `X` is the horizontal axis variable, `Y` is the vertical axis variable, and `labels` is an optional variable giving labels for the points. After typing this command, left-click on points on the scatterplot to make labels appear; when you're finished, right-click and select Stop.

- 10 To make a **bar chart** for cases in different categories, use the `barplot()` function. For frequency bar charts of one qualitative variable, type `barplot(table(X1))`, where `X1` is a qualitative variable. For frequency bar charts of two qualitative variables, type `barplot(table(X1, X2), beside=TRUE)`, where `X1` and `X2` are qualitative variables, and `beside` is a logical value for whether the chart is clustered (TRUE) or stacked (FALSE).

The bars can also represent various summary functions for a quantitative variable. For example, to produce a bar chart of means, type:
`means <- aggregate(Y, by=list(X1=X1, X2=X2), mean)`
`barplot(tapply(means$X, list(means$X1, means$X2), sum), beside=TRUE)`,
 where `X1` and `X2` are the qualitative variables and `Y` is a quantitative variable.

- 11 To make **boxplots** for cases in different categories, use the `boxplot()` function. For just one qualitative variable, type `boxplot(Y ~ X1)`, and for two qualitative variables, type `boxplot(Y ~ X1 + X2)`, where `Y` is a quantitative variable, and `X1` and `X2` are qualitative variables.
- 12 To make a **QQ-plot** (also known as a **normal probability plot**) for a quantitative variable, type `qqnorm(Y)`, where `Y` is a quantitative variable. To add a diagonal line to aid interpretation, type `qqline(Y)`.
- 13 To compute a **confidence interval** for a univariate population mean, type `t.test(Y, conf.level=0.95)`, where `Y` is the variable for which you want to calculate the confidence interval, and `conf.level` is the confidence level of the interval.
- 14 To do a **hypothesis test** for a univariate population mean, type `t.test(Y, mu=value, alternative="two.sided")`, where `Y` is the variable for which you want to do the test, `mu` is the (null) hypothesized value, and `alternative` is "two.sided" for two tailed, "less" for lower tailed, or "greater" for upper tailed.

Simple linear regression

- 15 To fit a **simple linear regression model** (i.e., find a least squares line), type
`model <- lm(Y ~ X)`
`summary(model)`,
 where `Y` is the response variable and `X` is the predictor variable.
- 16 To add a **regression line** or **least squares line** to a scatterplot, type `plot(X, Y)`, where `X` is the predictor variable and `Y` is the response variable, and then type `lines(sort(X), fitted(model)[order(X)])`, where `model` is the name of the fitted model object (see computer help #15).

- 17 To find 95% **confidence intervals for the regression parameters** in a simple linear regression model, type `confint(model)`, where `model` is the name of the fitted model object.

This applies more generally to multiple linear regression also.

- 18 To find a **confidence interval for the mean of Y** at a particular value of X in a simple linear regression model, type `predict(model, interval="confidence")`, where `model` is the name of the fitted model object.

The confidence intervals for the mean of Y at each of the X-values in the dataset are displayed as two columns headed `lwr` and `upr`.

You can also obtain a confidence interval for the mean of Y at an X-value that is not in the dataset by typing `predict(model, newdata=data.frame(X=a), interval="confidence")`, where `newdata` contains a data frame with variables that have the same names as the predictors in the model (X in this case), and `a` is the particular predictor value we are interested in.

This applies more generally to multiple linear regression also.

- 19 To find a **prediction interval** for an individual value of Y at a particular value of X in a simple linear regression model, type `predict(model, interval="prediction")`, where `model` is the name of the fitted model object.

The prediction intervals for the individual value of Y at each of the X-values in the dataset are displayed as two columns headed `lwr` and `upr`.

You can also obtain a prediction interval at an X-value that is not in the dataset by typing `predict(model, newdata=data.frame(X=a), interval="prediction")`, where `newdata` contains a data frame with variables that have the same names as the predictors in the model (X in this case), and `a` is the particular predictor value we are interested in.

This applies more generally to multiple linear regression also.

Multiple linear regression

- 20 To fit a **multiple linear regression model**, type
`model <- lm(Y ~ X1 + X2)`
`summary(model)`,
 where Y is the response variable and X1 and X2 are the predictor variables.
- 21 To add a **quadratic regression line** to a scatterplot, first fit the quadratic regression model, `model <- lm(Y ~ X + Xsq)`, where Y is the response variable, X is the predictor variable X, and Xsq is X^2 . To produce a smooth quadratic regression line, it might be necessary to create a “more even” version of X for use in the graph, for example, `newX <- seq(min(X), max(X), length=100)` and `newXsq <- newX**2`. Next, type `plot(X, Y)` and then type `lines(newX, predict(model, newdata=data.frame(X=newX, Xsq=newXsq)))`.
- 22 Categories of a qualitative variable can be thought of as defining **subsets** of the sample. If there are also quantitative response and predictor variables in the dataset, a regression model can be fit to the data to represent separate regression lines for each subset. For example, in a dataset of 20 observations, suppose X2 contains the values 1–4 to represent four categories, and Y and X1 are two quantitative variables. Create an interaction term, `X1X2 <- X1*X2`, and fit the model, `model <- lm(Y ~ X1 + X2 + X1X2)`.

Then the following code produces a scatterplot with numbers (representing the value of X2) instead of circles marking the points, and four separate regression lines:

```
plot(X1, Y, type="n")
for (i in 1:20) points(X1[i], Y[i], pch=as.character(X2[i]))
for (i in 1:4) lines(X1[X2==i], fitted(model)[X2==i]).
```

- 23 To find the F-statistic and associated p-value for a **nested model F-test** in multiple linear regression, first fit the reduced model, for example, `model1 <- lm(Y ~ X1)`, and the complete model, for example, `model2 <- lm(Y ~ X1 + X2 + X3)`.

Then type `anova(model1, model2)`. The F-statistic is in the second row of the “Analysis of Variance Table” in the column headed F, while the associated p-value is in the column headed `Pr(>F)`.

- 24 To save **studentized residuals** in a multiple linear regression model (e.g., `model <- lm(Y ~ X1 + X2)`), type `sre <- rstandard(model)`. They can now be used just like any other variable, for example, to construct residual plots. (R also has a `rstudent()` function, but that calculates “deleted studentized residuals.”)

- 25 To add a **loess fitted line** to a scatterplot (useful for checking the zero mean regression assumption in a residual plot), type, for example, `scatter.smooth(fitted(model), rstandard(model), span=0.75)`, where `model` is the name of the fitted model object.

- 26 To save **leverages** in a multiple linear regression model, type `lev <- hatvalues(model)`, where `model` is the name of the fitted model object. They can now be used just like any other variable, for example, to construct scatterplots.

- 27 To save **Cook’s distances** in a multiple linear regression model, type `cook <- cooks.distance(model)`, where `model` is the name of the fitted model object. They can now be used just like any other variable, for example, to construct scatterplots.

- 28 To create some **residual plots** automatically in a multiple linear regression model, type `plot(model)`, where `model` is the name of the fitted model object. This produces a plot of residuals versus fitted values, a QQ-plot of the studentized residuals, a plot of the square root of the studentized residuals versus fitted values, and a plot of studentized residuals versus leverages (with Cook’s distance thresholds of 0.5 and 1 marked)—click or hit Enter to cycle through the plots.

To create residual plots manually, first create studentized residuals (see computer help #24), and then construct scatterplots with these studentized residuals on the vertical axis.

- 29 To create a **correlation matrix** of quantitative variables (useful for checking potential **multicollinearity** problems), type `cor(cbind(Y, X1, X2))`, where Y, X1, and X2 are quantitative variables.

- 30 To find **variance inflation factors** in multiple linear regression, first install and load the `car` package of Fox (2002). Then type `vif(model)`, where `model` is the name of the fitted model object.

- 31 To draw a **predictor effect plot** for graphically displaying the effects of transformed quantitative predictors and/or interactions between quantitative and qualitative predictors in multiple linear regression, first create a variable representing the effect, say, “X1effect” (see computer help #3).

If the “X1effect” variable just involves X1 (e.g., $1+3X1+4X1^2$), type `plot(sort(X1), X1effect[order(X1)], type="l")`.

If the “X1effect” variable involves a qualitative variable (e.g., $1-2X1+3D2X1$, where D2 is an indicator variable), type

```
X1effect0 <- X1effect[D2==0]
X1effect1 <- X1effect[D2==1]
plot(X1, X1effect, type="n")
lines(sort(X1[D2==0]), X1effect0[order(X1[D2==0])])
lines(sort(X1[D2==1]), X1effect1[order(X1[D2==1])]).
```

See Section 5.4 for an example.