¡¡Setup,hidden=TRUE,echo=FALSE,message=FALSE,warning=FALSE¿¿= options(width = 90)
library(mosaic) library(Lock5Data) library(knitr) options(digits = 3) opts$_k$*nit*set(progress =
TRUE) opts$_c$*hunk*set(progress = TRUE, verbose = TRUE, prompt=FALSE,echo=TRUE, fig.align="center",
fig.width=8, fig.height=5, out.width="0.7
linewidth", size="scriptsize") setHook("plot.new", function() par(mgp=c(2,1,0), mar=c(3,3,1.5,0)
) , action="append") @

S-043 R Cookbook
A Handbook of R Snippits
Part A: Genearlized Linear Regression

Joe McIntyre

August 26, 2019

A generalized linear regression model has three components. First, it has a model for some value η, such that

$$\eta = \beta_0 + \beta_1 X_1 + ... + \beta_p X_p$$

Second, it has a function, f, which should be strictly increasing, mapping η onto the mean value of the outcome, Y, conditional on the predictors, $X_1, ..., X_p$.

Finally, it has probability distribution for the outcome, conditional on the predictors.

Generalized linear regression is an extension of linear regression. Like linear regression, it assumes a linear association between the predictors and some function of the outcome. However, a generalized linear model (GLM) also allows us to specify the distribution of the outcome conditional on its mean value, and a link function which maps the mean value of the outcome onto some value, η, which is linear in the predictors. Standard linear regression is generalized linear regression where the conditional outcome distribution is Normal with unknown variance and the link function is the identity function (i.e., the mean of the outcome is linear in the predictors). Similarly, logistic regression is a generalized linear regression model where the conditional outcome is Bernoulli (i.e., 0 or 1) and the link function is the logistic function.

GLMs are appropriate when

1. the variance of the outcome is a function of its mean value, or
2. the mean of the outcome is not linear in the predictors, but some function of the mean is.

We use the glm function, which comes preloaded in R as a part of the stats package, to fit generalized linear models. Here's an example

¡¡LinearRegression, echo=TRUE¿¿= data(HappyPlanetIndex) my.lm = lm(Happiness LifeExpectancy, data=HappyPlanetIndex) my.lm @ We can plot it on top of a scatterplot. ¡¡LinearRegressionPlot, echo=TRUE¿¿= plot(Happiness LifeExpectancy, data=HappyPlanetIndex) abline(a = -1.1037, b = 0.1035) @

There is also a shortcut: ¡¡AblineExample,eval=FALSE¿¿= abline(my.lm) @

You can pull parts of your model out with the `coef()` command: ¡¡GettingLmStuff,echo=TRUE¿¿= coef(my.lm) coef(my.lm)[1] intercept coef(my.lm)[2] slope coef(my.lm)["LifeExpectancy"] alternate way. @

Removing Outliers If you can identify which rows the outliers are on, you can do this (say the rows are 5, 10, 12). ¡¡eval=FALSE¿¿= new.data = old.data[-c(5,10,12),] lm(Y X, data=new.data) @ The "-list" notation means give me my old data, but without rows 5, 10, and 12. Note the comma after the list.

If you notice your points all have X bigger than some value, say 20.5, you could use subset: ¡¡eval=FALSE¿¿= new.data = subset(old.data, X ¡= 20.5) @

0.1 Multiple Regression

We here present an example on the `RestaurantTips` dataset. Let's first regress Tip on Bill. Before doing regression, we should plot the data to make sure using simple linear regression is reasonable. For kicks, we add in the regression line as well:

¡¡RegressionCheck,echo=TRUE¿¿= data(RestaurantTips) plot(Tip Bill, data=RestaurantTips, main="Tip given Bill") mod = lm(Tip Bill, data=RestaurantTips) abline(mod) summary(mod) @

Looking at the plot, the trend appears to be approximately linear. There are a few unusually large tips, but no extreme outliers, and variability appears to be constant as Bill increases, so we proceed. The simple linear regression model is now saved under the name mod (short for model - you can call it anything you want). Once we fit the model, we used summary to see the output.

Results relevant to the intercept are in the (Intercept) row and results relevant to the slope are in the Bill (the explanatory variable) row. The estimate column gives the estimated coefficients, the std. error column gives the standard error for these estimates, the t value is simply estimate/SE, and the p-value is the result of a hypothesis test testing whether that coefficient is significantly different from 0.

We also see the standard error of the error as "Residual standard error" and R2 as "Multiple R-squared". The last line of the regression output gives details relevant to an ANOVA table for testing our model against no model. It has the F-statistic, degrees of freedom, and p-value.

The `abline` line added the regression line to our plot so we can see how the line fits the data.

Multiple Variables. We now include the additional explanatory variables of number in party (Guests) and whether or not they pay with a credit card: ¡¡RegressionMultiple, echo=TRUE¿¿= tip.mod = lm(Tip Bill + Guests + Credit, data=RestaurantTips) summary(tip.mod) @

This output should look very similar to the output for one variable, except now there is a row corresponding to each explanatory variable. Our two-category (y, n) `Credit` variable was automatically converted to a 0-1 dummy variable (with 'y' being 1 and 'n' our baseline).

Missing data. If you have missing data, `lm` will automatically drop those cases because it doesn't know what else to do. It will tell you this, however, with the `summary` command. ¡¡echo=TRUE¿¿= data(AllCountries) dev.lm = lm(BirthRate Rural + Health + ElderlyPop, data=AllCountries) summary(dev.lm) @

Checking conditions. To check the conditions, we need to calculate residuals, make a residual versus fitted values plot, and make a histogram of the residuals:

¡¡ConditionsForRegression, echo=TRUE¿¿= par(mfrow=c(1,2)) plot(tip.modfit, $tip.mod$residuals, main="Residual plot") hist(tip.mod$residuals$, $main = "Hist of Residuals")@On the left we see no real pattern other
checking" plots you might use. Another is the "fitted outcomes vs. actual outcomes" plot of :<< Conditions For Regres

$TRUE >>= predicted = predict(dev.lm)actual = dev.lm$model$BirthRateplot(predicted\,actual, main = "Fitvs.actualBirthRate")@Notethe`dev.lm`variablehasa`model`variableinsideit.Thisisadataframeoftheuseddata

If we tried to skip this, we would get an error:

```
> predicted = predict( dev.lm )
> plot( predicted ~ AllCountries$BirthRate, main="Fit vs. actual Birth Rate" )
Error in (function (formula, data = NULL, subset = NULL, na.action = na.fail,  :
  variable lengths differ (found for 'AllCountries$BirthRate')
```

0.2 Categorical Variables.

You can include any explanatory categorical variable in a multiple regression model, and R will automatically create corresponding 0/1 variables. For example, if you were to include gender coded as male/female, R would create a variable GenderMale that is 1 for males and 0 for females.

Numbers Coding Categories. If you have multiple levels of a category coded with numbers you have to be a bit careful because R will treat this as a quantitative variable. You will now it did this if you only see the variable on one line of your output. You should see $k-1$ lines if you have k different levels for a categorical variable. To make a variable categorical, use `as.factor`: ¡¡MakeFactor, echo=TRUE, fig.keep='none'¿¿= data(USStates) USStates$Region = as.factor(USStates$Region) @

Setting new baselines. We can reorder the levels (the first is our baseline). ¡¡ReorderFactor, echo=TRUE, fig.keep='none'¿¿= levels(USStates$Region)USStates$Region = relevel(USStates$Region, "S")levels( ) @ Now any regression will use the south as our baseline.

Factors R often treats categorical variables as factors. This is often useful, but sometimes annoying. A factor has different *levels* which are the different values it can be. For example: ¡¡CatLevels, echo=TRUE¿¿= data(FishGills3) levels(FishGills3$Calcium)table(FishGills3Calcium) @ Note the weird nameless level. If you make a boxplot, you will get an empty plot in addition to the other three. This is due to a data entry issue. You can drop the unused level: ¡¡DropCatLevels, echo=TRUE¿¿= FishGills3$Calcium = droplevels(FishGills3Calcium) @ You can also turn a categorical variable into a numeric one like so: ¡¡CatToNum, echo=TRUE¿¿= asnum = as.numeric(FishGills3$Calcium)asnum@

Regression on only a categorical variable is fine: ¡¡CatLevelsLM, echo=TRUE¿¿= mylm = lm(GillRate Calcium, data=FishGills3) mylm @ Here "high" is the baseline. We can also use a mean model: ¡¡CatLevelsLM2, echo=TRUE¿¿= mymm = mm(GillRate Calcium, data=FishGills3) mymm @