Chapter 7
Generalized Linear Models

While the linear regression model is common to Political Science, many of the outcome measures researchers wish to study are binary, ordinal, nominal, or count variables. When we study these limited dependent variables, we turn to techniques such as logistic regression, probit regression, ordered logit (and probit) regression, multinomial logit (and probit) regression, Poisson regression, and negative binomial regression. A review of these and several other methods can be seen in volumes such as King (1989) and Long (1997).

In fact, all of these techniques can be thought of as special cases of the *generalized linear model*, or GLM (Gill 2001). The GLM approach in brief is to transform the mean of our outcome in some way so that we can apply the usual logic of linear regression modeling to the transformed mean. This way, we can model a broad class of dependent variables for which the distribution of the disturbance terms violates the normality assumption from the Gauss–Markov theorem. Further, in many cases, the outcome is bounded, so the *link function* we use to transform the mean of the outcome may reflect a more realistic functional form (Gill 2001, pp. 31–32).

The `glm` command in R is flexible enough to allow the user to specify many of the most commonly used GLMs, such as logistic and Poisson regression. A handful of models that get somewhat regular usage, such as ordered logit and negative binomial regression, actually require unique commands that we also will cover. In general, though, the `glm` command is a good place to look first when a researcher has a limited dependent variable. In fact, the `glm` command takes an argument called `family` that allows the user to specify what kind of model he or she wishes to estimate. By typing `?family` into the R console, the user can get a quick overview of which models the `glm` command can estimate.

Electronic supplementary material: The online version of this chapter (doi: 10.1007/978-3-319-23446-5_7) contains supplementary material, which is available to authorized users.

J.E. Monogan III, *Political Analysis Using R*, Use R!,
DOI 10.1007/978-3-319-23446-5_7

This chapter proceeds by illustrating examples with binary outcomes, ordinal outcomes, and count outcomes. Each of these sections uses a different example data set in order to consider dependent variables of each type. Each section will introduce its example data in turn.

7.1 Binary Outcomes

We first consider binary outcome variables, or variables that take only two possible values. Usually these outcomes are coded 0 or 1 for simplicity of interpretation. As our example in this section, we use survey data from the Comparative Study of Electoral Systems (CSES). Singh (2014a) studies a subset of these data that consists of 44,897 survey respondents from 30 elections. These elections occurred between the years of 1996–2006 in the countries of Canada, the Czech Republic, Denmark, Finland, Germany, Hungary, Iceland, Ireland, Israel, Italy, the Netherlands, New Zealand, Norway, Poland, Portugal, Slovenia, Spain, Sweden, Switzerland, and the United Kingdom.

Singh uses these data to assess how ideological distance shapes individuals' vote choice and willingness to vote in the first place. Building on the spatial model of politics advanced by Hotelling (1929), Black (1948), Downs (1957), and others, the article shows that linear differences in ideology do a better job of explaining voter behavior than squared differences. The variables in the data set are as follows:

voted: Indicator coded 1 if the respondent voted, 0 if not.
votedinc: Indicator coded 1 if the respondent voted for the incumbent party, 0 if he or she voted for another party. (Non-voters are missing.)
cntryyear: A character variable listing the country and year of the election.
cntryyearnum: A numeric index identifying the country-year of the election.
distanceinc: Distance between the survey respondent and the incumbent party on a 0–10 ideology scale.
distanceincsq: Squared distance between the voter and incumbent party.
distanceweighted: Distance between the survey respondent and the most similar political party on a 0–10 ideology scale, weighted by the competitiveness of the election.
distancesqweighted: Squared weighted distance between the voter and most similar ideological party.

The data are saved in Stata format, so we will need to load the `foreign` library. Download the file `SinghJTP.dta` from the Dataverse linked on page vii or the chapter content linked on page 97. Then open the data as follows:

```
library(foreign)
voting<-read.dta("SinghJTP.dta",convert.factors=FALSE)
```

A good immediate step here would be to use commands such as `summary` as well as graphs to get a feel for the descriptive attributes of the data. This is left for the reader.

7.1.1 Logit Models

As a first model from these data, we will model the probability that a respondent voted for the incumbent party, rather than another party. We will use only one predictor in this case, and that is the distance between the voter and the incumbent party. We craft this as a *logistic regression* model. The syntax for this model is:

```
inc.linear<-glm(votedinc~distanceinc,
     family=binomial(link="logit"),data=voting)
```

The syntax of `glm` (**g**eneralized **l**inear **m**odel) is nearly identical to `lm`: We still start with a functional specification that puts the dependent variable to the left of the tilde (~) and predictors on the right separated with plus signs. Again, we reference our dataset with the `data` option. Now, however, we *must* use the `family` option to specify which GLM we want to estimate. By specifying `binomial(link="logit")`, we declare a binary outcome and that we are estimating a logit, rather than probit, model. After estimation, by typing `summary(inc.linear)` we get the output from our logistic regression model, which is as follows:

```
Call:
glm(formula = votedinc ~ distanceinc, family = binomial
  (link = "logit"),
    data = voting)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.2608  -0.8632  -0.5570   1.0962   2.7519

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.19396    0.01880   10.32   <2e-16 ***
distanceinc -0.49469    0.00847  -58.41   <2e-16 ***
---
Signif. codes:  0  ***  0.001  **  0.01  *  0.05 . 0.1
 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 47335  on 38210  degrees of freedom
Residual deviance: 42910  on 38209  degrees of freedom
  (6686 observations deleted due to missingness)
AIC: 42914

Number of Fisher Scoring iterations: 4
```

Table 7.1 Logit model of probability of voting for incumbent party, 30 cross-national elections

Predictor	Estimate	Std. error	z value	Pr(> |z|)
Intercept	0.1940	0.0188	10.32	0.0000
Distance	−0.4947	0.0085	−58.41	0.0000

Notes: $N = 38{,}211$. AIC = 42,914. 69 % correctly predicted. Data from Singh (2014a)

The printout looks similar to the printout of the linear model we estimated in Chap. 6.[1] A more formal presentation of our results can be found in Table 7.1.[2] Comparing these results to the linear model, however, a few differences are important to note. First, the coefficient estimates themselves are not as meaningful as those of the linear model. A logit model transforms our outcome of interest, the probability of voting for the incumbent party, because it is bounded between 0 and 1. The logit transform is worthwhile because it allows us to use a linear prediction framework, but it calls for an added step of effort for interpretation. (See Sect. 7.1.3 for more on this.) A second difference in the output is that it reports z ratios instead of t ratios: Just as before, these are computed around the null hypothesis that the coefficient is zero, and the formula for the ratio uses the estimate and standard error in the same way. Yet, we now need to assume these statistics follow a normal distribution, rather than a t distribution.[3] Third, different fit statistics are presented: deviance scores and the Akaike information criterion (AIC).[4]

In Table 7.1, we report the coefficients, standard errors, and inferential information. We also report the AIC, which is a good fit index and has the feature of penalizing for the number of parameters. Unlike R^2 in linear regression, though, the AIC has no natural metric that gives an absolute sense of model fit. Rather, it works better as a means of comparing models, with *lower* values indicating a better penalized fit. To include a measure of fit that does have a natural scale to it, we also report what percentage of responses our model correctly predicts. To compute this, all we need to do is determine whether the model would predict a vote for the incumbent party and compare this to how the respondent actually voted. In R, we can roll our own computation:

[1] In this case, the coefficient estimates we obtain are similar to those reported by Singh (2014a). However, our standard errors are smaller (and hence z and p values are bigger) because Singh clusters the standard errors. This is a useful idea because the respondents are nested within elections, though multilevel models (which Singh also reports) address this issue as well—see Sect. 8.1.

[2] LaTeX users can create a table similar to this quickly by typing: `library(xtable); xtable(inc.linear)`.

[3] An explanation of how the inferential properties of this model are derived can be found in Eliason (1993, pp. 26–27).

[4] Deviance is calculated as −2 times the logged ratio of the fitted likelihood to the saturated likelihood. Formally, $-2 \log \frac{L_1}{L_2}$, where L_1 is the fitted likelihood and L_2 is the saturated likelihood. R reports two quantities: the null deviance computes this for an intercept-only model that always predicts the modal value, and the residual deviance calculates this for the reported model.

```
predicted<-as.numeric(
     predict.glm(inc.linear,type="response")>.5)
true<-voting$votedinc[voting$voted==1]
correct<-as.numeric(predicted==true)
100*table(correct)/sum(table(correct))
```

On the first line, we create a vector of the predictions from the model. The code uses the `predict.glm` command, which usefully can forecast from any model estimated with the `glm` command. By specifying `type="response"` we clarify that we want our predictions to be on the probability scale (instead of the default scale of latent utility). We then ask if each probability is greater than 0.5. By wrapping all of this in the `as.numeric` command, we count all probabilities above 0.5 as predicted values of 1 (for the incumbent) and all that are less than 0.5 as predicted values of 0 (against the incumbent). On the second line, we simply subset the original vector of the outcome from the original data to those that voted and hence were included in the model. This subsetting step is essential because the `glm` command automatically deletes missing data from estimation. Hence, without subsetting, our predicted and true values would not properly link together. On the third line, we create a vector coded 1 if the predicted value matches the true value, and on the fourth line we create a table of this vector. The printout is:

```
correct
       0        1
30.99108 69.00892
```

Hence, we know that the model correctly predicts 69 % of the outcome values, which we report in Table 7.1.

As one more example of logistic regression, Singh (2014a) compares a model with linear ideological distances to one with squared ideological distances. To fit this alternative model, we type:

```
inc.squared<-glm(votedinc~distanceincsq,
     family=binomial(link="logit"),data=voting)
summary(inc.squared)
```

The output of the summary command on the second line is:

```
Call:
glm(formula = votedinc ~ distanceincsq, family = binomi
 al(link = "logit"),
    data = voting)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.1020  -0.9407  -0.5519   1.2547   3.6552

Coefficients:
               Estimate Std. Error z value Pr(>|z|)
(Intercept)   -0.179971   0.014803  -12.16   <2e-16 ***
```

```
distanceincsq -0.101549   0.002075  -48.94   <2e-16 ***
---
Signif. codes:  0  ***  0.001  **  0.01  *  0.05  .
 0.1  1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 47335  on 38210  degrees of freedom
Residual deviance: 43087  on 38209  degrees of freedom
  (6686 observations deleted due to missingness)
AIC: 43091

Number of Fisher Scoring iterations:}5
```

With this second model we can see how the AIC can be useful: With a higher value of 43,091 in the quadratic model, we conclude that the model with linear ideological distance fits better with an AIC of 42,914. This corresponds to the original article's conclusion that the linear form of the variable fits better.

7.1.2 Probit Models

Logit models have gained traction over the years for the sake of simplicity in computation and interpretation. (For instance, logit models can be interpreted with *odds ratios*.) However, a key assumption of logit models is that the error term in the latent variable model (or the latent utility) has a logistic distribution. We may be more content to assume that the error term of our model is normally distributed, given the prevalence of this distribution in nature and in asymptotic results.[5] *Probit regression* allows us to fit a model with a binary outcome variable with a normally distributed error term in the latent variable model.

To show how this alternative model of a binary outcome works, we turn to a model of the probability a survey respondent voted at all. Singh (2014a) models this as a function of the ideological proximity to the nearest party weighted by the competitiveness of the election. The theory here is that individuals with a relatively proximate alternative in a competitive election are more likely to find it worthwhile to vote. We fit this model as follows:

```
turnout.linear<-glm(voted~distanceweighted,
     family=binomial(link="probit"),data=voting)
summary(turnout.linear)
```

[5] Additionally, in advanced settings for which we need to develop a multivariate distribution for multiple outcome variables, the normal distribution is relatively easy to work with.

The output of our summary command is:

```
Call:
glm(formula = voted ~ distanceweighted, family = binomi
 al(link = "probit"),
    data = voting)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.9732   0.5550   0.5550   0.5776   0.6644

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)       1.068134   0.009293 114.942  < 2e-16
  ***
distanceweighted -0.055074   0.011724  -4.698 2.63e-06
  ***
---
Signif. codes:  0  ***  0.001  **  0.01  *  0.05  .
0.1  1

(Dispersion parameter for binomial family taken to
  be 1)

    Null deviance: 37788  on 44896  degrees of freedom
Residual deviance: 37766  on 44895  degrees of freedom
AIC: 37770

Number of Fisher Scoring iterations: 4
```

The layout of these probit model results look similar to the results of the logit model. Note, though, that changing the distribution of the latent error term to a normal distribution changes the scale of the coefficients, so the values will be different between logit and probit models. The substantive implications typically are similar between the models, so the user must decide which model works best in terms of assumptions and interpretation for the data at hand.

7.1.3 Interpreting Logit and Probit Results

An important feature of GLMs is that the use of a link function makes the coefficients more difficult to interpret. With a linear regression model, as estimated in Chap. 6, we could simply interpret the coefficient in terms of change in the expected value of the outcome itself, holding the other variables equal. With a GLM, though, the mean of the outcome has been transformed, and the coefficient speaks to change in the transformed mean. Hence, for analyses like logit and probit models,

we need to take additional steps in order to interpret the effect an input has on the outcome of interest.

For a logistic regression model, the analyst can quickly calculate the *odds ratio* for each coefficient simply by taking the exponential of the coefficient.[6] Recall that the *odds* of an event is the ratio of the probability the event occurs to the probability it does not occur: $\frac{p}{1-p}$. The odds ratio tells us the multiplicative factor by which the odds will change for a unit increase in the predictor. Within R, if we want the odds ratio for our distance coefficient in Table 7.1, we simply type:

```
exp(inc.linear$coefficients[-1])
```

This syntax will take the exponential of every coefficient estimate from our model, no matter the number of covariates. The `[-1]` omits the intercept, for which an odds ratio would be meaningless. Having only one predictor, the printout in this case is:

```
distanceinc
  0.6097611
```

We need to be careful when interpreting the meaning of odds ratios. In this case, for a one point increase in distance from the incumbent party on the ideology scale, the odds that a respondent will vote for the incumbent party diminish by a factor of 0.61. (With multiple predictors, we would need to add the *ceteris paribus* caveat.) If, instead of interpreting as a multiplicative factor, the analyst preferred to discuss change in percentage terms, type:

```
100*(exp(inc.linear$coefficients[-1])-1)
```

In this case a value of −39.02389 is returned. Hence, we can say: for a one point increase in distance from the incumbent party on the ideology scale, the odds that a respondent will vote for the incumbent party diminish by 39 %. Remember, though, that all of these statements relate specifically to *odds*, so in this case we are referring to a 39 % decrease in the ratio of the probability of voting for the incumbent to the probability of voting for any other party.

An alternative interpretation that often is easier to explain in text is to report *predicted probabilities* from a model. For a logistic regression model, inputting the predictions from the linear function (the latent utilities) into the logistic cumulative distribution function produces the predicted probability that the outcome takes a value of 1. A simple approach to intuitively illustrate the effect of a predictor is to plot the predicted probabilities at every value that a predictor can take, which shows how the probability changes in a nonlinear way as the predictor changes. We proceed first by creating our predicted probabilities:

```
distances<-seq(0,10,by=.1)
inputs<-cbind(1,distances)
colnames(inputs)<-c("constant","distanceinc")
inputs<-as.data.frame(inputs)
```

[6] This is because the logit link function is the log-odds, or logarithm of the odds of the event.

```
forecast.linear<-predict(inc.linear,newdata=inputs,
     type="response")
```

On the first line, we create a **seq**uence of possible distances from the incumbent party, ranging from the minimum (0) to the maximum (10) in small increments (0.1). We then create a matrix named `inputs` that stores predictor values of interest for all predictors in our model (using the **c**olumn **bind**, `cbind`, command to combine two vectors as columns in a matrix). Subsequently, we name the columns to match our variable names and recategorize this matrix **as** a **data frame**. On the final line, we use the `predict` command, which saves the predicted probabilities to a vector. Observe the use of the `newdata` option to specify our data frame of predictor values and the `type` option to specify that we want our predicted values on the `response` scale. By setting this to the response scale, the command returns predicted probabilities of voting for the incumbent party at each hypothetical distance.

As an alternative to the model in which voting for the incumbent is a function of the linear ideological distance between the voter and the party, we also fitted a model using the squared distance. We easily can compute the predicted probability from this alternative model against the value of distance on its original scale. Again, the predicted probabilities are computed by typing:

```
inputs2<-cbind(1,distances^2)
colnames(inputs2)<-c("constant","distanceincsq")
inputs2<-as.data.frame(inputs2)
forecast.squared<-predict(inc.squared,newdata=inputs2,
     type="response")
```

In this case, we use the original vector `distances` that captured hypothetical predictor values, and square them. By using these squared values, we save our predicted probabilities from the alternative model into the vector `forecast.squared`.

To plot the predicted probabilities from each model on the same space, we type:

```
plot(y=forecast.linear,x=distances,ylim=c(0,.6),type="l",
     lwd=2,xlab="",ylab="")
lines(y=forecast.squared,x=distances,lty=2,col="blue",lwd=2)
legend(x=6,y=.5,legend=c("linear","squared"),lty=c(1,2),
     col=c("black","blue"),lwd=2)
mtext("Ideological Distance",side=1,line=2.75,cex=1.2)
mtext("Probability of Voting for Incumbent",side=2,
     line=2.5,cex=1.2)
```

On the first line, we plot the predicted probabilities from the model with linear distance. On the vertical axis (`y`) are the probabilities, and on the horizontal axis (`x`) are the values of distance. We bound the probabilities between 0 and 0.6 for a closer look at the changes, set `type="l"` to produce a line plot, and use the option `lwd=2` to increase the line thickness. We also set the x- and y-axis labels to be empty (`xlab="",ylab=""`) so that we can fill in the labels later with a more precise command. On the second line, we add another line to the open figure of the

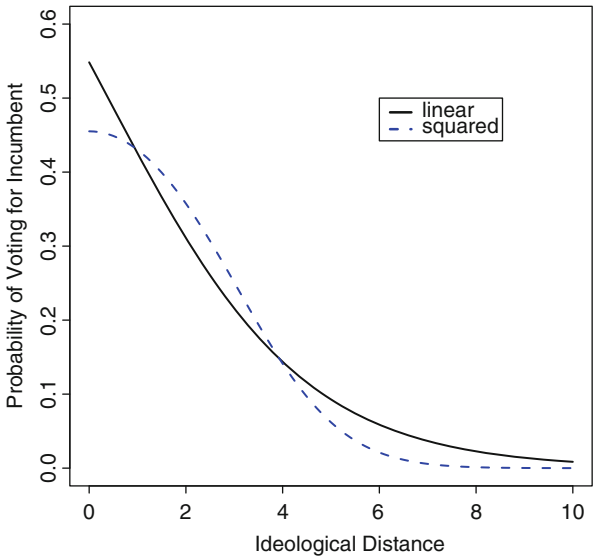

Fig. 7.1 Predicted probability of voting for the incumbent party as a function of ideological distance from the incumbents, based on a linear and a quadratic functional form

predicted probabilities from the model with squared distance. This time, we color the line blue and make it dashed (`lty=2`) to distinguish it from the other model's predicted probabilities. On the third line, we add a legend to the plot, located at the coordinates where `x=6` and `y=0.5`, that distinguishes the lines based on the linear and squared distances. Finally, on the last two lines we add axis labels using the `mtext` command: The `side` option lets us declare which axis we are writing on, the `line` command determines how far away from the axis the label is printed, and the `cex` command allows us to expand the font size (to 120 % in this case). The full results are presented in Fig. 7.1. As the figure shows, the model with squared distance is more responsive at middle values, with a flatter response at extremes. Hence, Singh's (2014a) conclusion that linear distance fits better has substantive implications for voter behavior.

As one final example of reporting predicted probabilities, we turn to an example from the probit model we estimated of turnout. Predicted probabilities are computed in a similar way for probit models, except that the linear predictions (or utilities) are now input into a normal cumulative distribution function. In this example, we will add to our presentation of predicted probabilities by including confidence intervals around our predictions, which convey to the reader the level of uncertainty in our forecast. We begin as we did in the last example, by creating a data frame of hypothetical data values and producing predicted probabilities with them:

```
wght.dist<-seq(0,4,by=.1)
inputs.3<-cbind(1,wght.dist)
colnames(inputs.3)<-c("constant","distanceweighted")
inputs.3<-as.data.frame(inputs.3)
forecast.probit<-predict(turnout.linear,newdata=inputs.3,
     type="link",se.fit=TRUE)
```

In this case, weighted ideological distance from the nearest ideological party is our one predictor. This predictor ranges from approximately 0–4, so we create a vector spanning those values. On the last line of the above code, we have changed two features: First, we have specified `type="link"`. This means that our predictions are now linear predictions of the latent utility, and not the probabilities in which we are interested. (This will be corrected in a moment.) Second, we have added the option `se.fit=TRUE`, which provides us with a standard error of each linear prediction. Our output object, `forecast.probit` now contains both the linear forecasts and the standard errors.

The reason we saved the linear utilities instead of the probabilities is that doing so will make it easier for us to compute confidence intervals that stay within the probability limits of 0 and 1. To do this, we first compute the confidence intervals of the linear predictions. For the 95 % confidence level, we type:

```
lower.ci<-forecast.probit$fit-1.95996399*forecast.probit$se.fit
upper.ci<-forecast.probit$fit+1.95996399*forecast.probit$se.fit
```

Notice that by calling `forecast.probit$fit` we obtain the linear predictions of utilities, and by calling `forecast.probit$se.fit` we call the standard errors of our forecast. 1.95996399 is the two-tailed 95 % critical value from a normal distribution. Now that we have vectors of the lower and upper bounds of the confidence interval, we can insert these into the normal cumulative distribution function to put the bounds on the predicted probability scale.

We can now plot the predicted probabilities with confidence intervals as follows:

```
plot(y=pnorm(forecast.probit$fit),x=wght.dist,ylim=c(.7,.9),
     type="l",lwd=2,xlab="Weighted Ideological Distance",
     ylab="Probability of Turnout")
lines(y=pnorm(lower.ci),x=wght.dist,lty=3,col="red",lwd=2)
lines(y=pnorm(upper.ci),x=wght.dist,lty=3,col="red",lwd=2)
```

In the first line, we plot the predicted probabilities themselves. To obtain the probabilities for the vertical axis, we type `y=pnorm(forecast.probit$fit)`. The `pnorm` function is the normal cumulative distribution function, so this converts our linear utility predictions into actual probabilities. Meanwhile, `x=wght.dist` places the possible values of weighted distance to the nearest party on the horizontal axis. On the second line, we plot the lower bound of the 95 % confidence interval of predicted probabilities. Here, `pnorm(lower.ci)` converts the confidence interval forecast onto the probability scale. Finally, we repeat the process on line three to plot the upper bound of the confidence interval. The full output can be seen in Fig. 7.2. A noteworthy feature of this plot is that the confidence interval becomes noticeably wide for the larger values of weighted distance. This is because the mean of the variable is low and there are few observations at these higher values.

The predicted probabilities in both of these cases were simple because they included only one predictor. For any GLM, including a logit or probit model, predicted probabilities and their level of responsiveness depend on the value of all covariates. Whenever a researcher has multiple predictors in a GLM model,

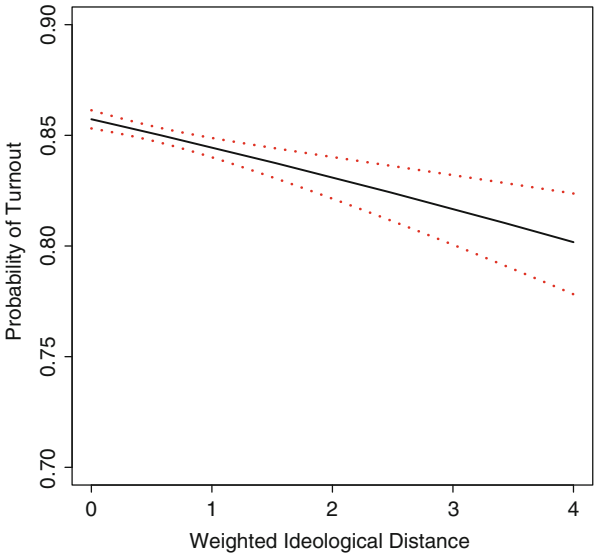

Fig. 7.2 Predicted probability of turning out to vote as a function of weighted ideological distance from the nearest party, with 95 % confidence intervals

reasonable values of the control variables must be included in the forecasts. See Sect. 7.3.3 for an example of using the `predict` function for a GLM that includes multiple predictors.

7.2 Ordinal Outcomes

We now turn to ordinal outcome measures. Ordinal variables have multiple categories as responses that can be ranked from lowest to highest, but other than the rankings the numeric values have no inherent meaning. As an example of an ordinal outcome variable, we again use survey data from the CSES, this time from Singh's (2014b) study of satisfaction with democracy. Relative to the previous example, these data have a wider scope, including 66,908 respondents from 62 elections. The variables in this data set are as follows:

satisfaction: Respondent's level of satisfaction with democracy. Ordinal scale coded 1 (not at all satisfied), 2 (not very satisfied), 3 (fairly satisfied), or 4 (very satisfied).
cntryyear: A character variable listing the country and year of the election.
cntryyearnum: A numeric index identifying the country-year of the election.
freedom: Freedom House scores for a country's level of freedom. Scores range from −5.5 (least free) to −1 (most free).
gdpgrowth: Percentage growth in Gross Domestic Product (GDP).

gdppercapPPP: GDP per capita, computed using purchasing price parity (PPP), chained to 2000 international dollars, in thousands of dollars.
CPI: Corruption Perceptions Index. Scores range from 0 (least corrupt) to 7.6 (most corrupt).
efficacy: Respondent thinks that voting can make a difference. Ordinal scale from 1 (disagree) to 5 (agree).
educ: Indicator coded 1 if the respondent graduated from college, 0 if not.
abstained: Indicator coded 1 if the respondent abstained from voting, 0 if the respondent voted.
prez: Indicator coded 1 if the country has a presidential system, 0 if not.
majoritarian_prez: Indicator coded 1 if the country has a majoritarian system, 0 if not.
winner: Indicator coded 1 if the respondent voted for the winning party, 0 if not.
voted_ID: Indicator coded 1 if the respondent voted for the party he or she identifies with, 0 if not.
voted_affect: Indicator coded 1 if the respondent voted for the party he or she rated highest, 0 if not.
voted_ideo: Indicator coded 1 if the respondent voted for the most similar party on ideology, 0 if not.
optimality: Vote optimality scale ranging from 0 to 3, coded by adding **voted_ID**, **voted_affect**, and **voted_ideo**.
winnerXvoted_ID: Interaction term between voting for the winner and voting by party identification.
winnerXvoted_affect: Interaction term between voting for the winner and voting for the highest-rated party.
winnerXvoted_ideo: Interaction term between voting for the winner and voting by ideological similarity.
winnerXoptimality: Interaction term between voting for the winner and the vote optimality scale.

These data are also in Stata format, so if the `foreign` library is not already loaded, it will need to be called. To load our data, download the file `SinghEJPR.dta` from the Dataverse linked on page vii or the chapter content linked on page 97. Then type:

```
library(foreign)
satisfaction<-read.dta("SinghEJPR.dta")
```

In this example, we wish to model each respondent's level of satisfaction with democracy. This variable takes on the values 1, 2, 3, and 4, and we can only make statements about which values reflect higher levels of satisfaction. In other words, we can say that a value of 4 ("very satisfied") reflects higher satisfaction than a value of 3 ("fairly satisfied"), a value of 3 is more than 2 ("not very satisfied"), and therefore a value of 4 is higher than a value of 2. We cannot, however, say *how much* more satisfaction one value reflects relative to another, as the numbers have no inherent meaning other than providing a rank order of satisfaction. To do so would

require us to quantify adjectives such as "very" and "fairly." Hence, an ordered logit or ordered probit model is going to be appropriate for this analysis.

As our first example, Singh (2014b, Table SM2) fits a model in which satisfaction in democracy is based on whether the respondent voted for the candidate most ideologically proximate, whether the candidate voted for the winner, and the interaction between these two variables.[7] The most important of these terms is the interaction, as it tests the hypothesis that individuals who were on the winning side *and* voted for the most similar party ideologically will express the greatest satisfaction.

Turning to specifics, for ordinal regression models we actually must use the special command `polr` (short for **p**roportional **o**dds **l**ogistic **r**egression), which is part of the `MASS` package. Most R distributions automatically install `MASS`, though we still need to load it with the `library` command.[8] In order to load the `MASS` package and then estimate an *ordered logit* model with these data, we would type:

```
library(MASS)
satisfaction$satisfaction<-ordered(as.factor(
     satisfaction$satisfaction))
ideol.satisfaction<-polr(satisfaction~voted_ideo*winner+
     abstained+educ+efficacy+majoritarian_prez+
     freedom+gdppercapPPP+gdpgrowth+CPI+prez,
     method="logistic",data=satisfaction)
summary(ideol.satisfaction)
```

Observe that we recode our dependent variable, `satisfaction`, using the `as.factor` command that was introduced in Sect. 2.4.2. We further embed this within the `ordered` command to convey that the factor can be ordered numerically. We had to recode in this way because the model command `polr` requires the outcome to be saved as a vector of class `factor`. Meanwhile, the right-hand side of the model resembles all other models we have estimated so far by separating variable names by plus signs. Notice that we use interactive notation with `voted_ideo*winner`, in order to include both terms plus the product.[9] All together, we have modeled satisfaction as a function of whether the voter voted for the most ideologically similar party, voted for the winner, an interaction term between the two, and several other individual-level and country-level control variables. Within the command, the option `method="logistic"` specifies that we wish to estimate an ordered *logit* model rather than use another link function. At the end of the line, we specify our `data` option as usual to point to our data set of interest.

[7] This and the next example do not exactly replicate the original results, which also include random effects by country-year. Also, the next example illustrates ordered probit regression, instead of the ordered logistic model from the original article. Both of the examples are based on models found in the online supporting material at the *European Journal of Political Research* website.

[8] If a user does need to install the package, `install.packages("MASS")` will do the job.

[9] An equivalent specification would have been to include `voted_ideo+winner+winnerXvoted_ideo` as three separate terms from the data.

After estimating this model, we type `summary(ideol.satisfaction)` into the console. The output looks like this:

```
Call:
polr(formula=satisfaction~voted_ideo*winner+abstained+
    educ+efficacy+majoritarian_prez+freedom+gdppercap
      PPP+
    gdpgrowth+CPI+prez,data=satisfaction,method=
      "logistic")

Coefficients:
                      Value Std. Error  t value
voted_ideo         -0.02170   0.023596  -0.9198
winner              0.21813   0.020638  10.5694
abstained          -0.25425   0.020868 -12.1838
educ                0.08238   0.020180   4.0824
efficacy            0.16246   0.006211  26.1569
majoritarian_prez   0.05705   0.018049   3.1609
freedom             0.04770   0.014087   3.3863
gdppercapPPP        0.01975   0.001385  14.2578
gdpgrowth           0.06653   0.003188  20.8673
CPI                -0.23153   0.005810 -39.8537
prez               -0.11503   0.026185  -4.3930
voted_ideo:winner   0.19004   0.037294   5.0957

Intercepts:
    Value    Std. Error t value
1|2  -2.0501   0.0584   -35.1284
2|3  -0.0588   0.0575    -1.0228
3|4   2.7315   0.0586    46.6423

Residual Deviance: 146397.33
AIC: 146427.33
```

The output shows the estimate of each coefficient, the standard error, and the z value. (Though R calls it a t value, maximum likelihood methods typically call for z ratios, as mentioned previously.) After the coefficient presentation, three *cut points* are presented under the label of `Intercepts`. These cut points identify the model by finding where on a latent utility scale the cutoff is between a respondent's choosing 1 versus 2 as a response, 2 versus 3, and 3 versus 4. Although the cut points often are often not of substantive interest, they are important for the sake of forecasting predicted outcomes. The default fit statistics in the output are the residual deviance and the AIC.

The results are presented more formally in Table 7.2. Although R's base output omits the p value, the user can easily draw inferences based on the available information: Either through computing confidence intervals, comparing the z value to a critical value, or computing p values oneself. For instance, the key hypothesis

Table 7.2 Ordered logit model of satisfaction with democracy, 62 cross-national elections

Predictor	Estimate	Std. error	z value	Pr(> |z|)
Voted for proximate party	−0.0217	0.0236	−0.9198	0.3577
Voted for winner	0.2181	0.0206	10.5694	0.0000
Voted proximate×winner	0.1900	0.0373	5.0957	0.0000
Abstained	−0.2542	0.0209	−12.1838	0.0000
College graduate	0.0824	0.0202	4.0824	0.0000
Efficacy	0.1625	0.0062	26.1569	0.0000
Majoritarian system	0.0571	0.0180	3.1609	0.0016
Freedom	0.0477	0.0141	3.3863	0.0007
Economic development	0.0197	0.0014	14.2578	0.0000
Economic growth	0.0665	0.0032	20.8673	0.0000
Corruption	−0.2315	0.0058	−39.8537	0.0000
Presidential system	−0.1150	0.0262	−4.3930	0.0000
τ_1	−2.0501	0.0584	−35.1284	0.0000
τ_2	−0.0588	0.0575	−1.0228	0.3064
τ_3	2.7315	0.0586	46.6423	0.0000

Notes: $N = 66{,}908$. AIC = 146,427. Data from Singh (2014b)

here is for the interaction term to be positive. Hence, we can obtain our one-tailed p value by typing:

```
1-pnorm(5.0957)
```

R prints out `1.737275e-07`, which in scientific notation means $p = 0.00000017$. Therefore, we will conclude with 99.9 % confidence that the coefficient on the interaction term is discernibly greater than zero. In Table 7.2, we have opted to report the two-tailed p values.[10]

A nice feature of using the logit link function, is that the results can be interpreted in terms of odds ratios. Odds ratios need to be computed and interpreted a bit differently for an ordinal model, though. See Long (1997, pp. 138–140) for a full explanation. For ordered logit models, we must exponentiate the *negative* value of a coefficient and interpret the odds of being in lower groups relative to higher groups. By way of example, the odds ratios for our coefficients from Table 7.2, along with the percentage changes in odds, can be produced at once:

[10] Unfortunately the `xtable` command does not produce ready-made LaTeX tables for results from `polr`. By creating a matrix with the relevant results, though, LaTeX users can produce a table faster than hand coding, though some revisions of the final product are necessary. Try the following:

```
coef<-c(ideol.satisfaction$coefficients,ideol.satisfaction$zeta)
se<-sqrt(diag(vcov(ideol.satisfaction)))
z<-coef/se
p<-2*(1-pnorm(abs(z)))
xtable(cbind(coef,se,z,p),digits=4)
```

```
exp(-ideol.satisfaction$coefficients)
100*(exp(-ideol.satisfaction$coefficients)-1)
```

The resulting printout from the second line is:

```
      voted_ideo            winner         abstained
        2.194186       -19.597657         28.949139
            educ          efficacy majoritarian_prez
       -7.908003       -14.994659         -5.545347
         freedom      gdppercapPPP         gdpgrowth
       -4.658313         -1.955471         -6.436961
              CPI              prez voted_ideo:winner
       26.053177         12.190773        -17.307376
```

If we wanted to interpret the effect of efficacy, then, we could say for a one point increase on a five-point efficacy scale, the odds a respondent will report that they are "not at all satisfied" with democracy relative to any of the three higher categories decrease by 15 %, *ceteris paribus*. Also, the odds that a respondent will report "not at all satisfied" or "not very satisfied" relative to the two higher categories also decrease by 15 %, all else equal. Further still, the odds that a respondent will report one of the bottom three satisfaction levels relative to the highest category of "very satisfied" diminish by 15 %, holding the other predictors constant. In general, then, we can interpret an odds ratio for an ordered logit as shaping the odds of all options below a threshold relative to all options above a threshold.

As a second example, we turn now to a model of voter satisfaction that focuses not on the role of ideological proximity in vote choice, but instead on which party the voter evaluated the most highly when asked to rate the parties. Again, the interaction between voting for the highest-rated party and also voting for the winning party is the primary coefficient of interest. This time, we will also try a different link function and estimate an *ordered probit* model instead of an ordered logit model. In this case we will type:

```
affect.satisfaction<-polr(satisfaction~voted_affect*winner+
     abstained+educ+efficacy+majoritarian_prez+
     freedom+gdppercapPPP+gdpgrowth+CPI+prez,
     method="probit",data=satisfaction)
```

Besides changing one of the interacted variables, the only difference between this code and the prior command for the ordered logit command is the specification of `method="probit"`. This changes the scale of our coefficients somewhat, but the substantive implications of results are generally similar regardless of this choice. By typing `summary(affect.satisfaction)`, we obtain the output:

```
Call:
polr(formula = satisfaction ~ voted_affect * winner +
 abstained +
    educ + efficacy + majoritarian_prez + freedom +
     gdppercapPPP +
```

```
    gdpgrowth + CPI + prez, data = satisfaction, method
     = "probit")

Coefficients:
                        Value Std. Error t value
voted_affect          0.03543  0.0158421   2.237
winner                0.04531  0.0245471   1.846
abstained            -0.11307  0.0170080  -6.648
educ                  0.05168  0.0115189   4.487
efficacy              0.09014  0.0035177  25.625
majoritarian_prez     0.03359  0.0101787   3.300
freedom               0.03648  0.0082013   4.448
gdppercapPPP          0.01071  0.0007906  13.546
gdpgrowth             0.04007  0.0018376  21.803
CPI                  -0.12897  0.0033005 -39.075
prez                 -0.03751  0.0147650  -2.540
voted_affect:winner   0.14278  0.0267728   5.333

Intercepts:
    Value    Std. Error t value
1|2  -1.1559   0.0342   -33.7515
2|3  -0.0326   0.0340    -0.9586
3|4   1.6041   0.0344    46.6565

Residual Deviance: 146698.25
AIC: 146728.25
```

Once again, our hypothesis of interest is supported by the results, with a positive and discernible effect on the interaction between voting for the winning party and voting for the highest-rated party.

7.3 Event Counts

As a third type of GLM, we turn to models of event counts. Whenever our dependent variable is the number of events that occurs within a defined period of time, the variable will have the feature that it can never be negative and must take on a discrete value (e.g., 0,1,2,3,4, . . .). Count outcomes therefore tend to have a strong right skew and a discrete probability distribution such as the Poisson or negative binomial distribution.

As an example of count data, we now return to Peake and Eshbaugh-Soha's (2008) data that were previously discussed in Chap. 3. Recall that the outcome variable in this case is the number of television news stories related to energy policy in a given month. (See Chap. 3 for additional details on the data.) The number of news stories in a month certainly is an event count. However, note that because

these are monthly data, they are *time dependent*, which is a feature we ignore at this time. In Chap. 9 we revisit this issue and consider models that account for time. For now, though, this illustrates how we usually fit count models in R.

First, we load the data again:[11]

```
pres.energy<-read.csv("PESenergy.csv")
```

After viewing the descriptive statistics on our variables and visualizing the data as we did in Chap. 3, we can now turn to fitting a model.

7.3.1 Poisson Regression

The simplest count model we can fit is a Poisson model. If we were to type:

```
energy.poisson<-glm(Energy~rmn1173+grf0175+grf575+jec477+
     jec1177+jec479+embargo+hostages+oilc+Approval+Unemploy,
     family=poisson(link=log),data=pres.energy)
```

This will fit a Poisson regression model in which TV coverage of energy policy is a function of six terms for presidential speeches, an indicator for the Arab oil embargo, an indicator for the Iran hostage crisis, the price of oil, presidential approval, and the unemployment rate.[12] Notice that this time, we set `family=poisson(link=log)`, declaring the distribution of the outcome as Poisson and specifying our log link function. If we type `summary(energy.poisson)` into the console, R returns the following output:

```
Call:
glm(formula = Energy ~ rmn1173 + grf0175 + grf575 +
 jec477 +
    jec1177 + jec479 + embargo + hostages + oilc +
      Approval +
    Unemploy, family = poisson(link = log), data = pres.
      energy)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-8.383  -2.994  -1.054   1.536  11.399

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)     13.250093   0.329121    40.259       < 2e-16      ***
rmn1173          0.694714   0.077009    9.021        < 2e-16      ***
```

[11]The footnote should read: "For users who do not have the file handy from Chapter 3, please download the file from the Dataverse linked on page vii or the chapter content linked on page 97.

[12]Note that the presidential speech terms are coded 1 only in the month of the speech, and 0 in all other months. The terms for the oil embargo and hostage crisis were coded 1 while these events were ongoing and 0 otherwise.

```
grf0175          0.468294   0.096169    4.870       1.12e-06    ***
grf575          -0.130568   0.162191    -0.805      0.420806
jec477           1.108520   0.122211    9.071       < 2e-16     ***
jec1177         0.576779    0.155511    3.709       0.000208    ***
jec479          1.076455    0.095066    11.323      < 2e-16     ***
embargo         0.937796    0.051110    18.349      < 2e-16     ***
hostages        -0.094507   0.046166    -2.047      0.040647    *
oilc            -0.213498   0.008052    -26.515     < 2e-16     ***
Approval        -0.034096   0.001386    -24.599     < 2e-16     ***
Unemploy        -0.090204   0.009678    -9.321      < 2e-16     ***
---
Signif. codes:   0  ***     0.001  **   0.01  *      0.05  .  0.1   1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 6009.0  on 179  degrees of freedom
Residual deviance: 2598.8  on 168  degrees of freedom
AIC: 3488.3

Number of Fisher Scoring iterations: 5
```

The format of the output—coefficient estimates, standard errors, z, and p—should be very familiar by now, as are the deviance and AIC scores. In this case the link function is simply a logarithm, so although the coefficients themselves are not very meaningful, interpretation is still simple. As one option, if we take the exponential of a coefficient, this offers us a *count ratio*, which allows us to deduce the percentage change in the expected count for a change in the input variable. For instance, if we wanted to interpret the effect of presidential approval, we could type:

```
exp(-.034096)
```

Here, we simply inserted the estimated coefficient from the printed output. The result gives us a count ratio of 0.9664787. We could interpret this as meaning for a percentage point increase in the president's approval rating, coverage of energy policy diminishes by 3.4 % on average and holding all other predictors equal. As a quick way to get the count ratio and percentage change for every coefficient, we could type:

```
exp(energy.poisson$coefficients[-1])
100*(exp(energy.poisson$coefficients[-1])-1)
```

In both lines the `[-1]` index for the coefficient vector throws away the intercept term, for which we do not want a count ratio. The printout from the second line reads:

```
   rmn1173    grf0175     grf575     jec477    jec1177
100.313518  59.726654 -12.240295 202.987027  78.029428
    jec479    embargo   hostages       oilc   Approval
193.425875 155.434606  -9.017887 -19.224639  -3.352127
  Unemploy
 -8.625516
```

From this list, we can simply read-off the percentage changes for a one-unit increase in the input, holding the other inputs equal. For a graphical means of interpretation, see Sect. 7.3.3.

7.3.2 Negative Binomial Regression

An intriguing feature of the Poisson distribution is that the variance is the same as the mean. Hence, when we model the logarithm of the mean, our model is simultaneously modeling the variance. Often, however, we find that the variance of our count variable is wider than we would expect given the covariates—a phenomenon called *overdispersion*. Negative binomial regression offers a solution to this problem by estimating an extra dispersion parameter that allows the conditional variance to differ from the conditional mean.

In R, **n**egative **b**inomial regression models actually require a special command from the `MASS` library called `glm.nb`. If the `MASS` library is not loaded, be sure to type `library(MASS)` first. Then, we can fit the negative binomial model by typing:

```
energy.nb<-glm.nb(Energy~rmn1173+grf0175+grf575+jec477+
    jec1177+jec479+embargo+hostages+oilc+Approval+Unemploy,
    data=pres.energy)
```

Notice that the syntax is similar to the `glm` command, but there is no `family` option since the command itself specifies that. By typing `summary(energy.nb)` the following results print:

```
Call:
glm.nb(formula = Energy ~ rmn1173 + grf0175 + grf575 +
  jec477 +
    jec1177 + jec479 + embargo + hostages + oilc +
      Approval +
    Unemploy, data = pres.energy, init.theta =
      2.149960724, link = log)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.7702  -0.9635  -0.2624   0.3569   2.2034

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) 15.299318   1.291013  11.851  < 2e-16 ***
rmn1173      0.722292   0.752005   0.960  0.33681
grf0175      0.288242   0.700429   0.412  0.68069
grf575      -0.227584   0.707969  -0.321  0.74786
jec477       0.965964   0.703611   1.373  0.16979
```

```
jec1177       0.573210   0.702534   0.816  0.41455
jec479        1.141528   0.694927   1.643  0.10045
embargo       1.140854   0.350077   3.259  0.00112 **
hostages      0.089438   0.197520   0.453  0.65069
oilc         -0.276592   0.030104  -9.188  < 2e-16 ***
Approval     -0.032082   0.005796  -5.536  3.1e-08 ***
Unemploy     -0.077013   0.037630  -2.047  0.04070 *
---
Signif. codes: 0  ***  0.001  **  0.01  *  0.05  . 0.1 1

(Dispersion parameter for Negative Binomial(2.15) family
  taken
to be 1)

    Null deviance: 393.02  on 179  degrees of freedom
Residual deviance: 194.74  on 168  degrees of freedom
AIC: 1526.4

Number of Fisher Scoring iterations: 1


              Theta:  2.150
          Std. Err.:  0.242

 2 x log-likelihood:  -1500.427
```

The coefficients reported in this output can be interpreted in the same way that coefficients from a Poisson model are interpreted because both model the logarithm of the mean. The key addition, reported at the end of the printout, is the dispersion parameter θ. In this case, our estimate is $\hat{\theta} = 2.15$, and with a standard error of 0.242 the result is discernible. This indicates that overdispersion is present in this model. In fact, many of the inferences drawn vary between the Poisson and negative binomial models. The two models are presented side by side in Table 7.3. As the results show, many of the discernible results from the Poisson model are not discernible in the negative binomial model. Further, the AIC is substantially lower for the negative binomial model, indicating a better fit even when penalizing for the extra overdispersion parameter.

7.3.3 Plotting Predicted Counts

While count ratios are certainly a simple way to interpret coefficients from count models, we have the option of graphing our results as well. In this case, we model the logarithm of our mean parameter, so we must exponentiate our linear prediction to predict the expected count given our covariates. As with logit and probit models, for count outcomes the `predict` command makes forecasting easy.

Suppose we wanted to plot the effect of presidential approval on the number of TV news stories on energy, based on the two models of Table 7.3. This situation contrasts a bit from the graphs we created in Sect. 7.1.3. In all of the logit and probit examples, we only had one predictor. By contrast, in this case we have several other predictors, so we have to set those to plausible alternative values. For this example, we will set the value of all dummy variable predictors to their modal value of zero, while the price of oil and unemployment are set to their mean. If we fail to insert reasonable values for the covariates, the predicted counts will not resemble the actual mean and the size of the effect will not be reasonable.[13] In this example, the way in which we use the `predict` command to forecast average counts with multiple predictors can be used in exactly the same way for a logit or probit model to forecast predicted probabilities with multiple predictors.

Table 7.3 Two count models of monthly TV new stories on energy policy, 1969–1983

	Poisson			Negative binomial		
Parameter	Estimate	Std. error	Pr(> |z|)	Estimate	Std. error	Pr(> |z|)
Intercept	13.2501	0.3291	0.0000	15.2993	1.2910	0.0000
Nixon 11/73	0.6947	0.0770	0.0000	0.7223	0.7520	0.3368
Ford 1/75	0.4683	0.0962	0.0000	0.2882	0.7004	0.6807
Ford 5/75	−0.1306	0.1622	0.4208	−0.2276	0.7080	0.7479
Carter 4/77	1.1085	0.1222	0.0000	0.9660	0.7036	0.1698
Carter 11/77	0.5768	0.1555	0.0002	0.5732	0.7025	0.4145
Carter 4/79	1.0765	0.0951	0.0000	1.1415	0.6949	0.1005
Arab oil embargo	0.9378	0.0511	0.0000	1.1409	0.3501	0.0011
Iran hostage crisis	−0.0945	0.0462	0.0406	0.0894	0.1975	0.6507
Price of oil	−0.2135	0.0081	0.0000	−0.2766	0.0301	0.0000
Presidential approval	−0.0341	0.0014	0.0000	−0.0321	0.0058	0.0000
Unemployment	−0.0902	0.0097	0.0000	−0.0770	0.0376	0.0407
θ	—			2.1500	0.2419	0.0000
AIC	3488.2830			1526.4272		

Notes: $N = 180$. Data from Peake and Eshbaugh-Soha (2008)

[13] Besides this approach of making predictions using central values of control variables, Hanmer and Kalkan (2013) make the case that forecasting outcomes based on the observed values of control variables in the data set is preferable. Readers are encouraged to consult their article for further advice on this issue.

Turning to specifics, in our data the variable `approve` ranges from 24% approval to 72.3%. Thus, we construct a vector that includes the full range of approval as well as plausible values of all of the other predictors:

```
approval<-seq(24,72.3,by=.1)
inputs.4<-cbind(1,0,0,0,0,0,0,0,0,mean(pres.energy$oilc),
     approval,mean(pres.energy$Unemploy))
colnames(inputs.4)<-c("constant","rmn1173","grf0175",
     "grf575","jec477","jec1177","jec479","embargo","hostages",
     "oilc","Approval","Unemploy")
inputs.4<-as.data.frame(inputs.4)
```

The first line above creates the vector of hypothetical values of our predictor of interest. The second line creates a matrix of hypothetical data values—setting the indicator variables to zero, the continuous variables to their means, and approval to its range of hypothetical values. The third line names the columns of the matrix after the variables in our model. On the last line, the matrix of predictor values is converted to a data frame.

Once we have the data frame of predictors in place, we can use the `predict` command to forecast the expected counts for the Poisson and negative binomial models:

```
forecast.poisson<-predict(energy.poisson,newdata=inputs.4,
     type="response")
forecast.nb<-predict(energy.nb,newdata=inputs.4,type="response")
```

These two lines only differ in the model from which they draw coefficient estimates for the forecast.[14] In both cases, we specify `type="response"` to obtain predictions on the count scale.

To graph our forecasts from each model, we can type:

```
plot(y=forecast.poisson,x=approval,type="l",lwd=2,
     ylim=c(0,60),xlab="Presidential Approval",
     ylab="Predicted Count of Energy Policy Stories")
lines(y=forecast.nb,x=approval,lty=2,col="blue",lwd=2)
legend(x=50,y=50,legend=c("Poisson","Negative Binomial"),
     lty=c(1,2),col=c("black","blue"),lwd=2)
```

The first line plots the Poisson predictions as a line with the `type="l"` option. The second line adds the negative binomial predictions, coloring the line blue and dashing it with `lty=2`. Finally, the `legend` command allows us to quickly distinguish which line represents which model. The full output is presented in Fig. 7.3. The predictions from the two models are similar and show a similar negative effect of approval. The negative binomial model has a slightly lower forecast at low values of approval and a slightly shallower effect of approval, such that the predicted counts overlap at high values of approval.

[14] As a side note, by using R's matrix algebra commands, described further in Chap. 10, the user can compute predicted counts easily with alternate syntax. For instance, for the negative binomial model, we could have typed: `forecast.nb<-exp(as.matrix(inputs.4)%*%energy.nb$coefficients)`.

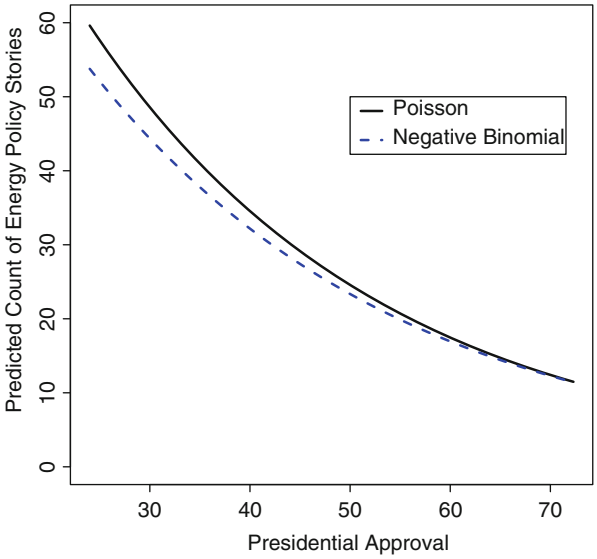

Fig. 7.3 Predicted count of energy policy stories on TV news as a function of presidential approval, holding continuous predictors at their mean and nominal predictors at their mode. Predictions based on Poisson and negative binomial model results

After the first seven chapters of this volume, users should now be able to perform most of the basic tasks that statistical software is designed to do: manage data, compute simple statistics, and estimate common models. In the remaining four chapters of this book, we now turn to the unique features of R that allow the user greater flexibility to apply advanced methods with packages developed by other users and tools for programming in R.

7.4 Practice Problems

1. Logistic regression: Load the `foreign` library, and download a subset of Singh's (2015) cross-national survey data on voter turnout, the file `stdSingh.dta`, available from the Dataverse listed on page vii or the chapter content listed on page 97. The outcome variable is whether the survey respondent voted (**voted**). A key predictor, with which several variables are interacted, is the degree to which a citizen is subject to mandatory voting rules. This is measured with a scale of how severe the compulsory voting rules are (**severity**). Five predictors should be interacted with **severity**: age (**age**), political knowledge (**polinfrel**), income (**income**), efficacy (**efficacy**), and partisanship (**partyID**). Five more predictors should be included only for additive effects: district magnitude (**dist_magnitude**), number of parties (**enep**), victory margin

(**vicmarg_dist**), parliamentary system (**parliamentary**), and per capita GDP (**development**). All of the predictor variables have been standardized.

a. Estimate a logistic regression model with these data, including the five interaction terms.
b. What is the odds ratio for number of parties? How would you interpret this term?
c. Plot the effect of per capita GDP on the probability of turning out. Hold all predictors other than development at their mean. *Hint:* Build on the code starting on page 107. If you used R's interaction notation (e.g., if `age*severity` is a term in the model), then when you create a new dataset of predictor values, you need only define your values for the original variables, and not for the products. In other words, you would need a column for **age**, for **severity**, and for any other predictors, but not for **age** × **severity**.
d. Bonus: Plot the effect of age on probability of turnout for three circumstances: When severity of compulsory voting rules is at its minimum, at its mean, and at its maximum. Hold all other predictors besides age and severity at their mean. Your final result should show three different predicted probability lines.

2. Ordered logit: The practice problems from Chapter 2 introduced Hanmer and Kalkan's (2013) subset of the 2004 American National Election Study. If you do not have these data already, the file `hanmerKalkanANES.dta` can be downloaded from the Dataverse linked on page vii or from the chapter content linked on page 97. Load the `foreign` library and open these data. (Again, be sure to specify the `convert.factors=F` option.) Consider two outcome variables: retrospective economic evaluations (**retecon**, taking on ordinal values coded −1, −0.5, 0, 0.5, and 1) and assessment of George W. Bush's handling of the war in Iraq (**bushiraq**, taking on ordinal values coded 0, 0.33, 0.67, and 1). There are seven predictor variables: partisanship on a seven-point scale (**partyid**), ideology on a seven-point scale (**ideol7b**), an indicator of whether the respondent is white (**white**), an indicator of whether the respondent is female (**female**), age of the respondent (**age**), level of education on a seven-point scale (**educ1_7**), and income on a 23-point scale (**income**).

a. Estimate an ordered logistic model of retrospective economic evaluations as a function of the seven predictors.
b. What is the odds ratio on the coefficient for female? How would you interpret this term?
c. Estimate an ordered logistic model of evaluation of Bush's handling of the war in Iraq as a function of the seven predictors.
d. What is the odds ratio on the coefficient for the seven-point partisanship scale? How would you interpret this term?
e. Bonus: The results of a model are biased if there is *reciprocal causation*, meaning that one of the independent variables not only influences the dependent variable, but also the dependent variable influences the independent

variable. Suppose you were worried about reciprocal causation bias in the model of retrospective economic evaluations. Which independent variable or variables would be most suspect to this criticism?

3. Count model: In the practice problems from Chapters 3 and 4, we introduced Peake and Eshbaugh-Soha's (2008) analysis of drug policy coverage. If you do not have their data from before, download `drugCoverage.csv` from the Dataverse linked on page vii or the chapter content linked on page 97. The outcome variable is news coverage of drugs (**drugsmedia**), and the four inputs are an indicator for a speech on drugs that Ronald Reagan gave in September 1986 (**rwr86**), an indicator for a speech George H.W. Bush gave in September 1989 (**ghwb89**), the president's approval rating (**approval**), and the unemployment rate (**unemploy**).[15]

   a. Estimate a Poisson regression model of drug policy coverage as a function of the four predictors.
   b. Estimate a negative binomial regression model of drug policy coverage as a function of the four predictors. Based on your models' results, which model is more appropriate, Poisson or negative binomial? Why?
   c. Compute the count ratio for the presidential approval predictor for each model. How would you interpret each quantity?
   d. Plot the predicted counts from each model contingent on the unemployment level, ranging from the minimum to maximum observed values. Hold the two presidential speech variables at zero, and hold presidential approval at its mean. Based on this figure, what can you say about the effect of unemployment in each model?

[15] Just as in the example from the chapter, these are time series data, so methods from Chap. 9 are more appropriate.