Chapter 8
Using Packages to Apply Advanced Models

In the first seven chapters of this book, we have treated R like a traditional statistical software program and reviewed how it can perform data management, report simple statistics, and estimate a variety of regression models. In the remainder of this book, though, we turn to the added flexibility that R offers—both in terms of programming capacity that is available for the user as well as providing additional applied tools through *packages*. In this chapter, we focus on how loading additional batches of code from user-written packages can add functionality that many software programs will not allow. Although we have used packages for a variety of purposes in the previous seven chapters (including `car`, `gmodels`, and `lattice`, to name a few), here we will highlight packages that enable unique methods. While the CRAN website lists numerous packages that users may install at any given time, we will focus on four particular packages to illustrate the kinds of functionality that can be added.

The first package we will discuss, `lme4`, allows users to estimate multilevel models, thereby offering an extension to the regression models discussed in Chaps. 6 and 7 (Bates et al. 2014). The other three were all developed specifically by Political Scientists to address data analytic problems that they encountered in their research: `MCMCpack` allows users to estimate a variety of models in a Bayesian framework using Markov chain Monte Carlo (MCMC) (Martin et al. 2011). `cem` allows the user to conduct coarsened exact matching—a method for causal inference with field data (Iacus et al. 2009, 2011). Lastly, `wnominate` allows the user to scale choice data, such as legislative roll call data, to estimate ideological ideal points of legislators or respondents (Poole and Rosenthal 1997; Poole et al. 2011). The following four sections will consider each package separately, so each section will introduce its data example in turn. These sections are designed to offer a brief overview of the

Electronic supplementary material: The online version of this chapter (doi: 10.1007/978-3-319-23446-5_8) 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_8

kinds of capacities R packages offer, though some readers may be unfamiliar with the background behind some of these methods. The interested reader is encouraged to consult with some of the cited resources to learn more about the theory behind these methods.

8.1 Multilevel Models with `lme4`

Having discussed linear models in Chap. 6 and several examples of generalized linear models in Chap. 7, we now turn to an extension of these kinds of models: multilevel models. Multilevel models, or hierarchical models, are appropriate whenever data of interest have either a nested or longitudinal structure. A nested structure occurs when observations can be thought of as being within or part of an upper-level unit: A common policy example is to study learning outcomes for students, but the students are nested within classrooms. In such a case, the researcher would need to account for the fact that the students in the sample are not independent of each other, but are likely to be similar if in the same classroom. Similarly, whenever a researcher studies individuals that have repeated observations over time, it is reasonable to think of the time-referenced observations as being embedded within the individual's observations. For example, in the LaLonde (1986) data first introduced in Chap. 4, the income of participants in his study are observed in 1974, 1975, and 1978. Some policy analyses might opt to consider the three temporal observations for each individual as being nested within each individual's case.[1] More comprehensive explanations of multilevel models can be found in Scott et al. (2013) and Gelman and Hill (2007). We proceed by extending two of our prior examples to illustrate a multilevel linear model and a multilevel logit model.

8.1.1 Multilevel Linear Regression

In this example, we return to our example from Chap. 6 on the number of hours teachers spend in the classroom teaching evolution. Originally, we fitted a linear model using ordinary least squares (OLS) as our estimator. However, Berkman and Plutzer (2010) make the point that teachers in the same state are likely to share similar features. These features could be similarities in training, in the local culture, or in state law. To account for these unobserved similarities, we can think of teachers as being *nested* within states. For this reason, we will add a *random effect* for each state. Random effects account for *intraclass correlation*, or error correlation among

[1]Note that this multilevel approach to panel data is most sensible for short panels such as these where there are many individuals relative to the number of time points. For long panels in which there are many time points relative to the number of individuals, more appropriate models are described as pooled time series cross-section methods. For more on the study of short panels, see Monogan (2011) and Fitzmaurice et al. (2004).

observations within the same group. In the presence of intraclass correlation, OLS estimates are inefficient because the disturbance terms are not independent, so a random effects model accounts for this problem.

First, we reload the data from the National Survey of High School Biology Teachers as follows:[2]

```
rm(list=ls())
library(foreign)
evolution<-read.dta("BPchap7.dta")
evolution$female[evolution$female==9]<-NA
evolution<-subset(evolution,!is.na(female))
```

Recall that we had a handful of observations of `female` that needed to be recoded as missing. As before, we subset our data to omit these missing observations.

In order to fit a multilevel model, there are actually a few commands available. We will opt to use a command from the `lme4` (**l**inear **m**ixed **e**effect) library.[3] On our first use, we will install the package, and then on every use we will load the library:

```
install.packages("lme4")
library(lme4)
```

Once we have loaded the library, we fit our multilevel linear model using the `lmer` (**l**inear **m**ixed **e**ffects **r**egression) command:

```
hours.ml<-lmer(hrs_allev~phase1+senior_c+ph_senior+notest_p+
     ph_notest_p+female+biocred3+degr3+evol_course+certified+
     idsci_trans+confident+(1|st_fip),data=evolution)
```

The syntax to the `lmer` command is nearly identical to the code we used when fitting a model with OLS using `lm`. In fact, the only attribute we added is the additional term `(1|st_fip)` on the right-hand side of the model. This adds a random intercept by state. On any occasion for which we wish to include a random effect, in parentheses we place the term for which to include the effect followed by a vertical pipe and the variable that identifies the upper-level units. So in this case we wanted a random intercept (hence the use of `1`), and we wanted these to be assigned by state (hence the use of `st_fip`).

We obtain our results by typing:

```
summary(hours.ml)
```

In our result output, R prints the correlation between all of the *fixed effects*, or estimated regression parameters. This part of the printout is omitted below:

```
Linear mixed model fit by REML ['lmerMod']
Formula:
hrs_allev~phase1+senior_c+ph_senior+notest_p+ph_notest
 _p+
```

[2] If you do not have these data from before, you can download the file `BPchap7.dta` from the Dataverse on page vii or the chapter content on page 125.

[3] See also the `nlme` library, which was a predecessor to `lme4`.

```
    female+biocred3+degr3+evol_course+certified+idsci_
      trans+
    confident+(1|st_fip)
   Data: evolution

REML criterion at convergence: 5940

Scaled residuals:
    Min      1Q  Median      3Q     Max
-2.3478 -0.7142 -0.1754  0.5566  3.8846

Random effects:
 Groups   Name        Variance Std.Dev.
 st_fip   (Intercept)  3.089   1.758
 Residual             67.873   8.239
Number of obs: 841, groups:  st_fip, 49

Fixed effects:
            Estimate Std. Error t value
(Intercept)  10.5676     1.2138   8.706
phase1        0.7577     0.4431   1.710
senior_c     -0.5291     0.3098  -1.708
ph_senior    -0.5273     0.2699  -1.953
notest_p      0.1134     0.7490   0.151
ph_notest_p  -0.5274     0.6598  -0.799
female       -0.9702     0.6032  -1.608
biocred3      0.5157     0.5044   1.022
degr3        -0.4434     0.3887  -1.141
evol_course   2.3894     0.6270   3.811
certified    -0.5335     0.7188  -0.742
idsci_trans   1.7277     1.1161   1.548
confident     2.6739     0.4468   5.984
```

The output first prints a variety of fit statistics: AIC, BIC, log-likelihood, deviance, and restricted maximum likelihood deviance. Second, it prints the variance and standard deviation of the random effects. In this case, the variance for the `st_fip` term is the variance of our state-level random effects. The residual variance corresponds to the error variance of regression that we would normally compute for our residuals. Last, the fixed effects that are reported are synonymous with linear regression coefficients that we normally are interested in, albeit now our estimates have accounted for the intraclass correlation among teachers within the same state. Table 8.1 compares our OLS and multilevel estimates side-by-side. As can be seen, the multilevel model now divides the unexplained variance into two components (state and individual-level), and coefficient estimates have changed somewhat.

Table 8.1 Two models of hours of class time spent teaching evolution by high school biology teachers

	OLS			Multilevel		
Parameter	Estimate	Std. error	Pr(> |z|)	Estimate	Std. error	Pr(> |z|)
Intercept	10.2313	1.1905	0.0000	10.5675	1.2138	0.0000
Standards index 2007	0.6285	0.3331	0.0596	0.7576	0.4431	0.0873
Seniority (centered)	−0.5813	0.3130	0.0636	−0.5291	0.3098	0.0876
Standards × seniority	−0.5112	0.2717	0.0603	−0.5273	0.2699	0.0508
Believes there is no test	0.4852	0.7222	0.5019	0.1135	0.7490	0.8795
Standards × believes no test	−0.5362	0.6233	0.3899	−0.5273	0.6598	0.4241
Teacher is female	−1.3546	0.6016	0.0246	−0.9703	0.6032	0.1077
Credits earned in biology (0–2)	0.5559	0.5072	0.2734	0.5157	0.5044	0.3067
Science degrees (0–2)	−0.4003	0.3922	0.3077	−0.4434	0.3887	0.2540
Completed evolution class	2.5108	0.6300	0.0001	2.3894	0.6270	0.0001
Has normal certification	−0.4446	0.7212	0.5377	−0.5335	0.7188	0.4580
Identifies as scientist	1.8549	1.1255	0.0997	1.7277	1.1161	0.1216
Self- rated expertise (−1 to +2)	2.6262	0.4501	0.0000	2.6738	0.4468	0.0000
State-level variance	—			3.0892		
Individual-level variance	69.5046			67.8732		

Notes: $N = 841$. Data from Berkman and Plutzer (2010)

8.1.2 Multilevel Logistic Regression

While somewhat more complex, the logic of multilevel modeling can also be applied when studying limited dependent variables. There are two broad approaches to extending GLMs into a multilevel framework: marginal models, which have a population-averaged interpretation, and generalized linear mixed models (GLMMs), which have an individual-level interpretation (Laird and Fitzmaurice 2013, pp. 149–156). While readers are encouraged to read further on the kinds of models available, their estimation, and their interpretation, for now we focus on the process of estimating a GLMM.

In this example, we return to our example from Sect. 7.1.1 from the last chapter, on whether a survey respondent reported voting for the incumbent party as a function of the ideological distance from the party. As Singh (2014a) observes, voters making their choice in the same country-year are going to face many features of the choice that are unique to that election. Hence, intraclass correlation among voters within the same election is likely. Further, the effect of ideology itself may be stronger in some elections than it is in others: Multilevel methods including GLMMs allow us to evaluate whether there is variation in the effect of a predictor across groups, which is a feature that we will use.

Turning to the specifics of code, if the `lme4` library is not loaded, we need that again. Also, if the data from are not loaded, then we need to load the `foreign` library and the data set itself. All of this is accomplished as follows:[4]

```
library(lme4)
library(foreign)
voting<-read.dta("SinghJTP.dta")
```

Building on the model from Table 7.1, we first simply add a random intercept to our model. The syntax for estimating the model and printing the results is:

```
inc.linear.ml<-glmer(votedinc~distanceinc+(1|cntryyear),
     family=binomial(link="logit"),data=voting)
summary(inc.linear.ml)
```

Notice that we now use the `glmer` command (**g**eneralized **l**inear **m**ixed **e**ffects **r**egression). By using the `family` option, we can use any of the common link functions available for the `glm` command. A glance at the output shows that, in addition to the traditional fixed effects that reflect logistic regression coefficients, we also are presented with the variance of the random intercept for country and year of the election:

```
Generalized linear mixed model fit by the Laplace
  approximation
Formula: votedinc ~ distanceinc + (1 | cntryyear)
   Data: voting
      AIC      BIC    logLik deviance
 41998.96 42024.62 -20996.48 41992.96
Random effects:
 Groups    Name        Variance Std.Dev.
 cntryyear (Intercept) 0.20663  0.45457
Number of obs: 38211, groups: cntryyear, 30

Fixed effects:
               Estimate   Std. Error   z value Pr(>|z|)
(Intercept)  0.161788717  0.085578393   1.89053 0.058687 .
distanceinc -0.501250136  0.008875997 -56.47254  < 2e-16  ***
---
Signif. codes:  0  ***  0.001  **  0.01  *  0.05  .  0.1  1

Correlation of Fixed Effects:
            (Intr)
distanceinc -0.185
```

To replicate a model more in line with Singh's (2014a) results, we now fit a model that includes a random intercept and a random coefficient on ideological distance, both contingent on the country and year of the election. The syntax for estimating this model and printing the output is:

[4]If you do not have these data from before, you can download the file `SinghJTP.dta` from the Dataverse on page vii or the chapter content on page 125.

```
inc.linear.ml.2<-glmer(votedinc~distanceinc+
     (distanceinc|cntryyear),family=binomial(link="logit"),
     data=voting)
summary(inc.linear.ml.2)
```

Notice that we now have conditioned the variable `distanceinc` by `cntryyear`. This adds a random coefficient for ideological distance. Also, by default, adding this random effect also adds a random intercept as well. Our output in this case is:

```
Generalized linear mixed model fit by the Laplace
  approximation
Formula: votedinc ~ distanceinc + (distanceinc |
  cntryyear)
   Data: voting
   AIC   BIC logLik deviance
 41074 41117 -20532    41064
Random effects:
 Groups    Name        Variance Std.Dev. Corr
 cntryyear (Intercept) 0.616658 0.78528
           distanceinc 0.098081 0.31318  -0.808
Number of obs: 38211, groups: cntryyear, 30

Fixed effects:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.26223    0.14531   1.805   0.0711 .
distanceinc -0.53061    0.05816  -9.124   <2e-16 ***
---
Signif. codes: 0  ***  0.001  **  0.01  *  0.05 . 0.1 1

Correlation of Fixed Effects:
            (Intr)
distanceinc -0.808
```

Under random effects, we first see the variance for the election-referenced random intercept, and then variance for the election-referenced coefficient for ideological distance. The fixed effects for the logistic regression coefficients are also presented in the usual way. The AIC indicates that this version of the model fits better than either the model with only a random intercept or the model from Table 7.1 that included no random effects, as the score of 41,074 is lower than the AIC of either of those models. In sum, this discussion should offer a sense of the kinds of hierarchical models that R can estimate using `lme4` (Bates et al. 2014).

8.2 Bayesian Methods Using `MCMCpack`

The `MCMCpack` package allows users to perform Bayesian inference on a variety of common regression models and measurement models. The package even has a command, `MCMCmetrop1R`, that will construct a MCMC sample from a user-defined distribution using a Metropolis algorithm. Readers who wish to learn more about Bayesian methods are encouraged to consult resources such as: Carlin and Louis (2009), Gelman et al. (2004), Gill (2008), and Robert (2001).

As a simple illustration of how the package functions, we focus in this section on some of the common regression models that are programmed into the package. This is powerful for the R user in that researchers who prefer to report Bayesian models can easily do so if their model specification conforms to a common structure. In illustrating these techniques, we will revisit one more time the linear model of evolution hours by Berkman and Plutzer (2010) and the logistic regression model of incumbent party support by Singh (2014a).

8.2.1 Bayesian Linear Regression

To estimate our Bayesian linear regression model, we must reload the data from the National Survey of High School Biology Teachers, if they are not already loaded:

```
rm(list=ls())
library(foreign)
evolution<-read.dta("BPchap7.dta")
evolution$female[evolution$female==9]<-NA
evolution<-subset(evolution,!is.na(female))
```

With the data loaded, we must install `MCMCpack` if this is the first use of the package on the computer. Once the program is installed, we then must load the library:

```
install.packages("MCMCpack")
library(MCMCpack)
```

Now we can use **MCMC** to fit our Bayesian linear **regress**ion model with the `MCMCregress` command:

```
mcmc.hours<-MCMCregress(hrs_allev~phase1+senior_c+ph_senior+
     notest_p+ph_notest_p+female+biocred3+degr3+
     evol_course+certified+idsci_trans+confident,data=evolution)
```

Be prepared that estimation with MCMC usually takes a longer time computationally, though simple models such as this one usually finish fairly quickly. Also, because MCMC is a simulation-based technique, it is normal for summaries of the results to differ slightly across replications. To this end, if you find differences between your results and those printed here after using the same code, you need not worry unless the results are markedly different.

While the above code relies on the defaults of the MCMCregress command, a few of this command's options are essential to highlight: A central feature of Bayesian methods is that the user must specify *priors* for all of the parameters that are estimated. The defaults for MCMCregress are vague conjugate priors for the coefficients and the variance of the disturbances. However, the user has the option of specifying his or her own priors on these quantities.[5] Users are encouraged to review these options and other resources about how to set priors (Carlin and Louis 2009; Gelman et al. 2004; Gill 2008; Robert 2001). Users also have the option to change the number of iterations in the MCMC sample with the mcmc option and the burn-in period (i.e., number of starting iterations that are discarded) with the burnin option. Users should always assess model *convergence* after estimating a model with MCMC (to be discussed shortly) and consider if either the burn-in or number of iterations should be changed if there is evidence of nonconvergence.

After estimating the model, typing `summary(mcmc.hours)` will offer a quick summary of the *posterior* sample:

```
Iterations = 1001:11000
Thinning interval = 1
Number of chains = 1
Sample size per chain = 10000

1. Empirical mean and standard deviation for each
   variable,plus standard error of the mean:

                Mean     SD Naive SE Time-series SE
(Intercept) 10.2353 1.1922 0.011922       0.011922
phase1       0.6346 0.3382 0.003382       0.003382
senior_c    -0.5894 0.3203 0.003203       0.003266
ph_senior   -0.5121 0.2713 0.002713       0.002713
notest_p     0.4828 0.7214 0.007214       0.007214
ph_notest_p -0.5483 0.6182 0.006182       0.006182
female      -1.3613 0.5997 0.005997       0.006354
biocred3     0.5612 0.5100 0.005100       0.005100
degr3       -0.4071 0.3973 0.003973       0.003973
evol_course  2.5014 0.6299 0.006299       0.005870
certified   -0.4525 0.7194 0.007194       0.007194
idsci_trans  1.8658 1.1230 0.011230       0.010938
confident    2.6302 0.4523 0.004523       0.004590
sigma2      70.6874 3.5029 0.035029       0.035619
```

[5]For priors on the coefficients, the option b0 sets the vector of means of a multivariate Gaussian prior, and B0 sets the variance-covariance matrix of the multivariate Gaussian prior. The prior distribution of the error variance of regression is inverse Gamma, and this distribution can be manipulated by setting its shape parameter with option c0 and scale parameter with option d0. Alternatively, the inverse Gamma distribution can be manipulated by changing its mean with the option sigma.mu and its variance with the option sigma.var.

```
2. Quantiles for each variable:

                2.5%       25%      50%       75%     97.5%
(Intercept)  7.92359  9.438567 10.2273 11.03072 12.59214
phase1      -0.02787  0.405026  0.6384  0.86569  1.30085
senior_c    -1.22527 -0.808038 -0.5885 -0.37351  0.04247
ph_senior   -1.04393 -0.694228 -0.5105 -0.32981  0.03152
notest_p    -0.92717 -0.006441  0.4863  0.97734  1.88868
ph_notest_p -1.75051 -0.972112 -0.5462 -0.13138  0.63228
female      -2.52310 -1.771210 -1.3595 -0.96109 -0.18044
biocred3    -0.42823  0.212168  0.5558  0.90768  1.55887
degr3       -1.19563 -0.671725 -0.4048 -0.14536  0.38277
evol_course  1.26171  2.073478  2.5064  2.92601  3.73503
certified   -1.84830 -0.942671 -0.4477  0.03113  0.95064
idsci_trans -0.33203  1.107771  1.8667  2.63507  4.09024
confident    1.73568  2.324713  2.6338  2.94032  3.48944
sigma2      64.12749 68.277726 70.5889 72.95921 77.84095
```

Since MCMC produces a sample of simulated parameter values, all of the reported information is based on simple descriptive statistics of the simulated output (which is 10,000 sets of parameters, in this case). Part 1 of the summary above shows the mean of the sample for each parameter, standard deviation of each parameter's sample, and two versions of the standard error of the mean. Part 2 of the summary shows percentiles of each parameter's sample. Table 8.2 shows one common format for presenting results from a model like this: reporting the mean and standard

Table 8.2 Linear model of hours of class time spent teaching evolution by high school biology teachers (MCMC Estimates)

Predictor	Mean	Std. Dev.	[95 % Cred. Int.]	
Intercept	10.2353	1.1922	[7.9236:	12.5921]
Standards index 2007	0.6346	0.3382	[−0.0279:	1.3008]
Seniority (centered)	−0.5894	0.3203	[−1.2253:	0.0425]
Standards × seniority	−0.5121	0.2713	[−1.0439:	0.0315]
Believes there is no test	0.4828	0.7214	[−0.9272:	1.8887]
Standards × believes no test	−0.5483	0.6182	[−1.7505:	0.6323]
Teacher is female	−1.3613	0.5997	[−2.5231:	−0.1804]
Credits earned in biology (0–2)	0.5612	0.5100	[−0.4282:	1.5589]
Science degrees (0–2)	−0.4071	0.3973	[−1.1956:	0.3828]
Completed evolution class	2.5014	0.6299	[1.2617:	3.7350]
Has normal certification	−0.4525	0.7194	[−1.8483:	0.9506]
Identifies as scientist	1.8658	1.1230	[−0.3320:	4.0902]
Self-rated expertise (−1 to +2)	2.6302	0.4523	[1.7357:	3.4894]
Error variance of regression	70.6874	3.5029	[64.1275:	77.8410]

Notes: $N = 841$. Data from Berkman and Plutzer (2010)

deviation of each parameter's marginal posterior distribution, and a 95 % *credible interval* based on the percentiles.[6]

When a user loads MCMCpack in R, the coda library will also load.[7] coda is particularly useful because it allows the user to assess convergence of models estimated with MCMC and report additional quantities of interest. As was mentioned earlier, whenever a researcher estimates a model using MCMC, he or she should assess whether there is any evidence of nonconvergence. In the event that the Markov chains have not converged, the model should be sampled for more iterations. MCMCregress estimates the model using a single chain. Hence, for our model of the number of hours of evolution taught in high school classrooms, we can assess convergence using **Geweke**'s convergence **diag**nostic, which simply asks whether the means of the first and last parts of the chain are the same. To compute this diagnostic, we type:

```
geweke.diag(mcmc.hours, frac1=0.1, frac2=0.5)
```

Here we have specified that we want to compare the mean of the first 10 % of the chain (frac1=0.1) to the mean of the last 50 % of the chain (frac2=0.5). The resulting output presents a z-ratio for this difference of means test for each parameter:

```
Fraction in 1st window = 0.1
Fraction in 2nd window = 0.5

(Intercept)      phase1    senior_c   ph_senior     notest_p
   -1.34891    -1.29015    -1.10934    -0.16417     0.95397
ph_notest_p       female    biocred3       degr3  evol_course
    1.13720    -0.57006     0.52718     1.25779     0.62082
  certified idsci_trans   confident      sigma2
    1.51121    -0.87436    -0.54549    -0.06741
```

In this case, none of the test statistics surpass any common significance thresholds for a normally distributed test statistic, so we find no evidence of nonconvergence. Based on this, we may be content with our original MCMC sample of 10,000.

One more thing we may wish to do with our MCMC output is **plot** the overall estimated **dens**ity function of our marginal posterior distributions. We can plot these one at a time using the densplot function, though the analyst will need to reference the parameter of interest based on its numeric order of appearance in the summary table. For example, if we wanted to plot the coefficient for the indicator of whether a teacher completed an evolution class (**evol_course**), that is the tenth

[6] To write out a similar table to Table 8.2 in LaTeX, load the xtable library in R and type the following into the console:

```
xtable(cbind(summary(mcmc.hours)$statistics[,1:2],
summary(mcmc.hours)$quantiles[,c(1,5)]),digits=4)
```

[7] This frequently occurs when one package depends on code from another.

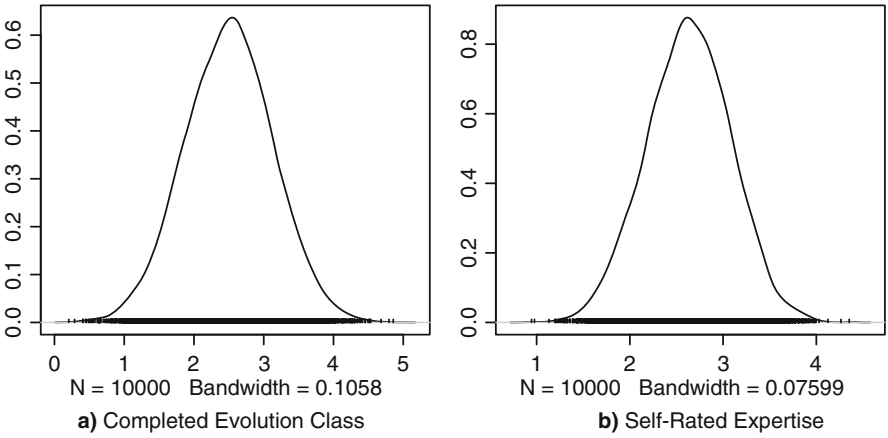

Fig. 8.1 Density plots of marginal posterior distribution of coefficients for whether the teacher completed an evolution class and the teacher's self-rated expertise. Based on an MCMC sample of 10,000 iterations (1000 iteration burn-in). (**a**) Completed evolution class; (**b**) Self-rated expertise

parameter estimate reported in the table. Similarly, if we wanted to report the density plot for the coefficient of the teacher's self-rated expertise (**confident**), that is the thirteenth parameter reported in the summary table. Hence we could plot each of these by typing:

```
densplot(mcmc.hours[,10])
densplot(mcmc.hours[,13])
```

The resulting density plots are presented in Fig. 8.1. As the figures show, both of these marginal posterior distributions have an approximate normal distribution, and the mode is located near the mean and median reported in our summary output.

8.2.2 Bayesian Logistic Regression

As one additional illustration that `MCMCpack` estimates a variety of models, we illustrate Bayesian logistic regression, using Singh's (2014a) data on voting for incumbent parties one last time. If you do not have these data or the library loaded, be sure to do so:

```
library(MCMCpack)
library(foreign)
voting<-read.dta("SinghJTP.dta")
```

To estimate the Bayesian **logit** model using **MCMC**, we type:

```
inc.linear.mcmc<-MCMClogit(votedinc~distanceinc,data=voting)
```

Just as with the MCMCregress command, we have chosen to use the defaults in this case, but users are encouraged to consider setting their own priors to suit their needs. As a matter of fact, this is one case where we will need to raise the number of iterations in our model. We can check convergence of our model using the Geweke diagnostic:

```
geweke.diag(inc.linear.mcmc, frac1=0.1, frac2=0.5)
```

Our output in this case actually shows a significant difference between the means at the beginning and end of the chain for each parameter:

```
Fraction in 1st window = 0.1
Fraction in 2nd window = 0.5

(Intercept) distanceinc
      2.680      -1.717
```

The absolute value of both z-ratios exceeds 1.645, so we can say the mean is significantly different for each parameter at the 90 % confidence level, which is evidence of nonconvergence.

As a response, we can double both our burn-in period and number of iterations to 2,000 and 20,000, respectively. The code is:

```
inc.linear.mcmc.v2<-MCMClogit(votedinc~distanceinc,
     data=voting,burnin=2000,mcmc=20000)
```

We can now check for the convergence of this new sample by typing:

```
geweke.diag(inc.linear.mcmc.v2, frac1=0.1, frac2=0.5)
```

Our output now shows nonsignificant z-ratios for each parameter, indicating that there is no longer evidence of nonconvergence:

```
Fraction in 1st window = 0.1
Fraction in 2nd window = 0.5

(Intercept) distanceinc
    -1.0975      0.2128
```

Proceeding with this sample of 20,000, then, if we type `summary(inc.linear.mcmc.v2)` into the console, the output is:

```
Iterations = 2001:22000
Thinning interval = 1
Number of chains = 1
Sample size per chain = 20000

1. Empirical mean and standard deviation for each
   variable, plus standard error of the mean:
```

```
                Mean       SD  Naive SE Time-series SE
(Intercept)   0.1940 0.01846 1.305e-04      0.0003857
distanceinc -0.4946 0.00829 5.862e-05      0.0001715

2. Quantiles for each variable:

                2.5%     25%     50%     75%   97.5%
(Intercept)   0.1573  0.1817  0.1944  0.2063  0.2298
distanceinc -0.5105 -0.5003 -0.4946 -0.4890 -0.4783
```

These results are similar to those that we reported last chapter in Table 7.1, though now we have the opportunity to interpret the findings as a Bayesian. As with Bayesian linear regression, if we wished to report density plots of each parameter, we could apply the `densplot` command just as before. Overall, this brief illustration should show researchers how easy it is to use Bayesian methods in R with `MCMCpack`. Readers who wish to use more advanced Bayesian methods are encouraged to consult the `MCMCpack` reference manual and Martin et al. (2011).

8.3 Causal Inference with `cem`

A prominent innovation in political methodology has been the development of several new matching methods. In brief, matching is a technique designed to select a subset of field data to make for a fair comparison of individuals who receive a treatment to control individuals who did not receive the treatment. With matching, some observations are thrown away so that the remaining control and treatment observations are similar on all covariates that are known to shape the outcome of interest. In the absence of experimental data, matching methods serve to allow the researcher to isolate how the treatment variable affects responses (see Rubin 2006 for a thorough treatment of causal inference with matching).

Political Scientists have developed several new matching methods (Imai and van Dyk 2004; Sekhon and Grieve 2012). As an illustration of one of these, and how the novel technique is implemented in R, we turn to the method developed by Iacus et al. (2009, 2011, 2012), **C**oarsened **E**xact **M**atching (CEM). In short, CEM proceeds by temporarily recoding each covariate into an ordered variable that lumps similar values of the covariate together. The data are then sorted into strata based on their profiles of the coarsened variables, and any observations in a stratum that does not contain at least one treatment and one control unit are thrown away. The resulting sample should show much more balance in the control variables between the treated and control observations. In R, this technique is implemented with the `cem` command within the `cem` package.

8.3.1 Covariate Imbalance, Implementing CEM, and the ATT

As our example data, we return to LaLonde's (1986) study of the National Supported Work Demonstration that we previously examined in Chaps. 4 and 5. Our treatment variable (**treated**) indicates if the individual received the treatment from the National Supported Work Demonstration, meaning the person was placed in a private sector job for a year with public funds covering the labor costs. We would like to know the causal effect of this treatment on the individual's earnings in 1978, after the treatment was completed (**re78**). In Sect. 5.1.1 we did a naïve test of this hypothesis by simply using a difference of means test between the treated individuals and the control group without controlling for any of the other variables that are likely to affect annual income. In our naïve test, we found that income was higher for the treated group, but now we can ask what the estimated effect is when we account for other important covariates.

As was noted in earlier chapters, LaLonde's data are already included in the `cem` package, so we can load these data easily if we have already loaded the `cem` library. The following lines of code clean up, install `cem` (in case you have not already), opens `cem`, and loads LaLonde's data (named `LL`):[8]

```
rm(list=ls())
install.packages("cem")
library(cem)
data(LL)
```

An important task when using matching methods is assessing the degree to which the data are *balanced*, or the degree to which treated cases have a similar distribution of covariate values relative to the control group. We can assess the degree to which our treatment and control groups have differing distributions with the `imbalance` command. In the code below, we first increase the penalty for scientific notation (an option if you prefer decimal notation). Then, we create a vector naming the variables we do not want to assess balance for—the treatment variable (**treated**) and the outcome of interest (**re78**). All other variables in the dataset are covariates that we believe can shape income in 1978, so we would like to have balance on them. In the last line, we actually call the `imbalance` command.

```
options(scipen=8)
todrop <- c("treated","re78")
imbalance(group=LL$treated, data=LL, drop=todrop)
```

Within the `imbalance` command, the `group` argument is our treatment variable that defines the two groups for which we want to compare the covariate distributions. The `data` argument names the dataset we are using, and the `drop` option allows us omit certain variables from the dataset when assessing covariate balance. Our output from this command is as follows:

[8] LaLonde's data is also available in the file `LL.csv`, available in the Dataverse (see page vii) or the chapter content (see page 125).