```
Multivariate Imbalance Measure: L1=0.735
Percentage of local common support: LCS=12.4%

Univariate Imbalance Measures:

                statistic   type           L1 min 25%
age           0.179203803 (diff) 4.705882e-03   0   1
education     0.192236086 (diff) 9.811844e-02   1   0
black         0.001346801 (diff) 1.346801e-03   0   0
married       0.010703110 (diff) 1.070311e-02   0   0
nodegree     -0.083477916 (diff) 8.347792e-02   0  -1
re74       -101.486184085 (diff) 5.551115e-17   0   0
re75         39.415450601 (diff) 5.551115e-17   0   0
hispanic     -0.018665082 (diff) 1.866508e-02   0   0
u74          -0.020099030 (diff) 2.009903e-02   0   0
u75          -0.045086156 (diff) 4.508616e-02   0   0
                50%       75%         max
age          0.00000  -1.0000     -6.0000
education    1.00000   1.0000      2.0000
black        0.00000   0.0000      0.0000
married      0.00000   0.0000      0.0000
nodegree     0.00000   0.0000      0.0000
re74        69.73096 584.9160  -2139.0195
re75       294.18457 660.6865    490.3945
hispanic     0.00000   0.0000      0.0000
u74          0.00000   0.0000      0.0000
u75          0.00000   0.0000      0.0000
```

The first line of our output reports $\mathscr{L}_1$, which is a measure of multivariate imbalance created by Iacus et al. (2011). A fuller explanation is available in that article, but in general this statistic ranges from 0 to 1, with lower values indicating better balance. When $\mathscr{L}_1 = 0$ the two distributions perfectly overlap, and when $\mathscr{L}_1 = 1$ the two distributions do not overlap at all. Turning to the table, each row shows several balance statistics for an individual covariate. For all of these statistics, values closer to zero are better. The column labeled `statistic` shows the difference in means between the variables, and the column labeled `L1` computes $\mathscr{L}_1^j$, which is the same measure as $\mathscr{L}_1$ but only calculated for the individual covariate. The remaining columns show quantile differences between the two groups (e.g., the difference in the two groups' respective minima, the difference between the groups' respective 25th percentiles, etc.).

Next, we will actually use the `cem` command to perform **C**oarsened **E**xact **M**atching on our data. Within the `cem` command, we list our treatment variable with the `treatment` argument, our dataset with the `data` argument, and any variables we do not want to match on with the `drop` argument. The `drop` argument should always include our outcome variable, if it is in the same data set, as well as any data indices or irrelevant variables. We could use a vector to list all of the variables we want to be ignored, as we did with the `imbalance` command before, but in this case, only the outcome `re78` needs to be skipped. We type:

```
cem.match.1 <- cem(treatment="treated", data=LL, drop="re78")
cem.match.1
```

Our immediate output from this is simply the following:

```
           G0  G1
All       425 297
Matched   222 163
Unmatched 203 134
```

This tells us that our original data had 425 control observations and 297 treated observations. CEM included 222 of the control and 163 of the treated observations in the matched sample, and the rest are pruned away. To be clear: All observations are still contained in the original `LL` dataset, but now the object `cem.match.1` itemizes which observations are matched or not.

Since CEM proceeds by grouping similar values of covariates into strata, an important feature of this is how we set the ordered intervals of each predictor in coarsening. The `cem` command has reasonable defaults if the user does not set these intervals, but it is important to record what the intervals are. To see what our intervals were for the values of our predictors, we could type: `cem.match.1$breaks`. This would give us the following output:

```
$age
 [1] 17.0 20.8 24.6 28.4 32.2 36.0 39.8 43.6 47.4 51.2 55.0

$education
 [1]  3.0  4.3  5.6  6.9  8.2  9.5 10.8 12.1 13.4 14.7 16.0

$black
 [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0

$married
 [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0

$nodegree
 [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0

$re74
 [1]     0.000  3957.068  7914.136 11871.204 15828.272 19785.340
 [7] 23742.408 27699.476 31656.544 35613.612 39570.680

$re75
 [1]     0.000  3743.166  7486.332 11229.498 14972.664 18715.830
 [7] 22458.996 26202.162 29945.328 33688.494 37431.660

$hispanic
 [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

```
$u74
 [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0

$u75
 [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

To illustrate what this means, consider `age`. The lowest category of coarsened age lumps together everyone aged 17–20.8, the second category lumps everyone aged 20.8–24.6, and so forth. Variables like `black`, `married`, `nodegree`, `hispanic`, `u74`, and `u75` are actually binary, so most of the categories being created are unnecessary, though empty bins will not hurt our analysis. Of course, users are not required to use software defaults, and Iacus et al. urge researchers to use substantive knowledge of each variable's measurement to set the ranges of the coarsening bins (2012, p. 9). Section 8.3.2 offers details on doing this.

Now we can assess imbalance in the new matched sample by typing:

```
 imbalance(LL$treated[cem.match.1$matched],
      LL[cem.match.1$matched,], drop=todrop)
```

Our output from this is as follows:

```
Multivariate Imbalance Measure: L1=0.592
Percentage of local common support: LCS=25.2%

Univariate Imbalance Measures:

              statistic   type          L1 min 25%       50%
age         -0.42486044 (diff) 0.00000000   0  -1   -2.0000
education   -0.10855027 (diff) 0.10902006   0   0   -1.0000
black       -0.01771403 (diff) 0.01771403   0   0    0.0000
married     -0.01630465 (diff) 0.01630465   0   0    0.0000
nodegree     0.09022827 (diff) 0.09022827   0   0    0.0000
re74      -119.33548135 (diff) 0.00000000   0   0    0.0000
re75       -50.01527694 (diff) 0.00000000   0   0  -49.3559
hispanic     0.01561377 (diff) 0.01561377   0   0    0.0000
u74          0.01619411 (diff) 0.01619411   0   0    0.0000
u75          0.02310286 (diff) 0.02310286   0   0    0.0000
              75%      max
age          0.00    1.000
education    0.00    0.000
black        0.00    0.000
married      0.00    0.000
nodegree     0.00    0.000
re74      -492.95  416.416
re75      -136.45 -852.252
hispanic     0.00    0.000
u74          0.00    0.000
u75          0.00    0.000
```

Compare this to the original data. We now have $\mathscr{L}_1 = 0.592$, which is less than our score of 0.735 for the raw data, indicating that multivariate balance is better in the matched sample. Turning to the individual covariates, you can see something of a mixed bag, but overall the balance looks better. For instance, with `age` the difference in means is actually a bit larger in absolute value with the matched sample (0.42) than the raw data (0.18). However, $\mathscr{L}_1^{\text{age}}$ is now minuscule in the matched sample, and less than the 0.0047 value for the raw data. This is likely on account of the fact that the raw data has a larger discrepancy at the high end than the matched sample has. The user now must decide whether the treated and control cases are sufficiently balanced or whether to try other coarsenings to improve balance.

If the user is satisfied with the level of balance, he or she can proceed to estimate the **A**verage **T**reatment effect on the **T**reated (ATT) using the command att. This quantity represents the causal effect on the kind of individual who received the treatment. In this command we specify what our matched sample is using the obj argument, the outcome variable (**re78**) and treatment (**treated**) using the formula argument, and our data using the data argument. This gives us:

```
est.att.1 <- att(obj=cem.match.1, formula=re78~treated, data=LL)
est.att.1
```

Our output from this is:

```
           G0  G1
All       425 297
Matched   222 163
Unmatched 203 134

Linear regression model on CEM matched data:

SATT point estimate: 550.962564 (p.value=0.368242)
95% conf. interval: [-647.777701, 1749.702830]
```

The output recaps the features of our matched sample and then reports our estimate of the sample average treatment effect on the treated (SATT): We estimate in our sample that individuals receiving the treatment earned \$551 more on average than those who did not receive the treatment. However, this effect is not statistically discernible from zero ($p = 0.368$). This is a markedly different conclusion from the one we drew in Chap. 5, when we observed a \$886 difference that was statistically significant. This illustrates the importance of statistical control.

8.3.2 Exploring Different CEM Solutions

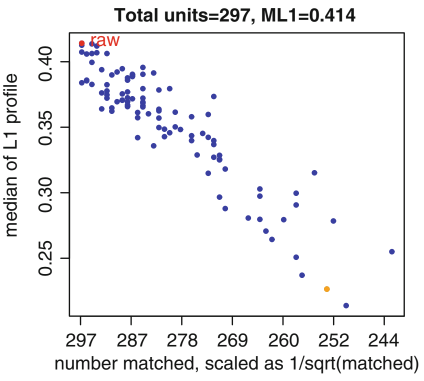
As a final point, if a researcher is not happy with the level of balance or the sample size in the matched sample, then a tool for finding better balance is the cemspace command. This command randomly produces several different coarsenings for the control variables (250 different coarsenings by default). The command then plots the level of balance against the number of treated observations included in the matched sample. The following code calls this command:

```
cem.explore<-cemspace(treatment="treated",data=LL,drop="re78")
```

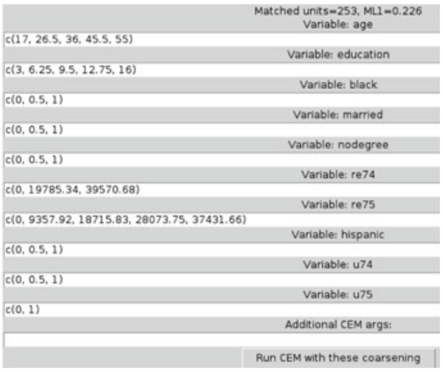
The syntax of `cemspace` is similar to `cem`, though two more options are important: `minimal` and `maximal`. These establish what the minimum and maximum allowed number of coarsened intervals is for the variables. The command above uses the defaults of 1 and 5, which means that no more than five intervals may be included for a variable. Hence, all matched samples from this command will be coarser than what we used in Sect. 8.3.1, and therefore less balanced. The user could, however, increase `maximal` to 12 or even a higher number to create finer intervals and potentially improve the balance over our prior result.

Our output from `cemspace` is shown in Fig. 8.2. *On account of the random element in choosing coarsenings, your results will not exactly match this figure.* Figure 8.2a shows the interactive figure that opens up. The horizontal axis of this figure shows the number of matched treatment units in descending order, while the vertical axis shows the level of imbalance. In general, a matched sample at the bottom-left corner of the graph would be ideal as that would indicate the best balance (reducing bias) and the largest sample (increasing efficiency). Normally, though, we have to make a choice on this tradeoff, generally putting a bit more weight on minimizing imbalance. By clicking on different points on the graph, the second window that `cemspace` creates, shown in Fig. 8.2b will show the intervals used in that particular coarsening. The user can copy the vectors of the interval cutpoints and paste them into his or her own code. *Note:* R will not proceed with new commands until these two windows are closed.

In Fig. 8.2a, one of the potential coarsenings has been chosen, and it is highlighted and yellow. If we want to implement this coarsening, we can copy the vectors shown in the second window illustrated in Fig. 8.2b. Pasting these into our own R script produces the following code:



a) Plot Window



```
Matched units=253, ML1=0.226
Variable: age
c(17, 26.5, 36, 45.5, 55)
Variable: education
c(3, 6.25, 9.5, 12.75, 16)
Variable: black
c(0, 0.5, 1)
Variable: married
c(0, 0.5, 1)
Variable: nodegree
c(0, 0.5, 1)
Variable: re74
c(0, 19785.34, 39570.68)
Variable: re75
c(0, 9357.92, 18715.83, 28073.75, 37431.66)
Variable: hispanic
c(0, 0.5, 1)
Variable: u74
c(0, 0.5, 1)
Variable: u75
c(0, 1)
Additional CEM args:
```

b) X11 Window

Fig. 8.2 Plot of balance statistics for 250 matched samples from random coarsenings against number of treated observations included in the respective matched sample. (**a**) Plot window; (**b**) X11 window

```
age.cut<-c(17, 26.5, 36, 45.5, 55)
education.cut<-c(3, 6.25, 9.5, 12.75, 16)
black.cut<-c(0, 0.5, 1)
married.cut<-c(0, 0.5, 1)
nodegree.cut<-c(0, 0.5, 1)
re74.cut<-c(0, 19785.34, 39570.68)
re75.cut<-c(0, 9357.92, 18715.83, 28073.75, 37431.66)
hispanic.cut<-c(0, 0.5, 1)
u74.cut<-c(0, 0.5, 1)
u75.cut<-c(0, 1)
new.cuts<-list(age=age.cut, education=education.cut,
     black=black.cut, married=married.cut,
     nodegree=nodegree.cut, re74=re74.cut, re75=re75.cut,
     hispanic=hispanic.cut, u74=u74.cut, u75=u75.cut)
```

We end this code by creating a *list* of all of these vectors. While our vectors here have been created using a coarsening created by `cemspace`, this is the procedure a programmer would use to create his or her own cutpoints for the intervals. By substituting the vectors above with user-created cutpoint vectors, a researcher can use his or her own knowledge of the variables' measurement to coarsen.

Once we have defined our own cutpoints, either by using `cemspace` or substantive knowledge, we can now apply CEM with the following code:

```
cem.match.2 <- cem(treatment="treated", data=LL, drop="re78",
     cutpoints=new.cuts)
```

Our key addition here is the use of the `cutpoints` option, where we input our list of intervals. Just as in Sect. 8.3.1, we can now assess the qualities of the matched sample, imbalance levels, and compute the ATT if we wish:

```
cem.match.2
imbalance(LL$treated[cem.match.2$matched],
     LL[cem.match.2$matched,], drop=todrop)
est.att.2 <- att(obj=cem.match.2, formula=re78~treated, data=LL)
est.att.2
```

In this case, in part because of the coarser bins we are using, the balance is worse that what we found in the previous section. Hence, we would be better off in this case sticking with our first result. The reader is encouraged to try to find a coarsening that produces better balance than the software defaults.

8.4 Legislative Roll Call Analysis with `wnominate`

Methodologists in Political Science and other disciplines have developed a wide array of measurement models, several of which are available for user implementation in R. Without a doubt, one of the most prominent measurement models in the discipline is NOMINATE, short for **nomina**l **t**hree-step **e**stimation (McCarty et al. 1997; Poole and Rosenthal 1997; Poole et al. 2011). The NOMINATE model analyzes roll call data from legislative votes, placing legislators, and policy alternatives they vote on in ideological space. The model is particularly prominent because Poole, Rosenthal, and colleagues make DW-NOMINATE scores available for both the US House of Representatives and Senate for every term of Congress. Countless authors have used these data, typically interpreting the first dimension score as a scale of liberal-conservative ideology.

In brief, the basic logic of the model draws from the spatial proximity model of politics, which essentially states that both individuals' ideological preferences and available policy alternatives can be represented in geometric space of one or more dimensions. An individual generally will vote for the policy choice that is closest in space to his or her own ideological ideal point (Black 1958; Downs 1957; Hotelling 1929). The NOMINATE model is based on these assumptions, and places legislators and policy options in ideological space based on how legislators' votes divide over the course of many roll call votes and when legislators behave unpredictably (producing errors in the model). For example, in the US Congress, liberal members typically vote differently from conservative members, and the extreme ideologues are the most likely to be in a small minority whenever there is wide consensus on an issue. Before applying the NOMINATE method to your own data—and even before downloading pre-measured DW-NOMINATE data to include in a model you estimate—be sure to read more about the method and its assumptions because thoroughly understanding how a method works is essential before using it. In particular, Chap. 2 and Appendix A from Poole and Rosenthal (1997) and Appendix A from McCarty et al. (1997) offer detailed, yet intuitive, descriptions of how the method works.

In R, the `wnominate` package implements W-NOMINATE, which is a version of the NOMINATE algorithm that is intended only to be applied to a single legislature. W-NOMINATE scores are internally valid, so it is fair to compare legislators' scores within a single dataset. However, the scores cannot be externally compared to scores when W-NOMINATE is applied to a different term of the legislature or a different body of actors altogether. Hence, it is a good method for trying to make cross-sectional comparisons among legislators of the same body.

While the most common application for W-NOMINATE has been the US Congress, the method could be applied to any legislative body. To that end, the working example in this section focuses on roll call votes cast in the United Nations. This `UN` dataset is available in the `wnominate` package, and it pools 237 roll call votes cast in the first three sessions of the UN (1946–1949) by 59 member nations. The variables are labeled **V1** to **V239**. **V1** is the name of the member nation, and **V2** is a categorical variable either coded as "WP" for a member of the Warsaw Pact, or "Other" for all other nations. The remaining variables sequentially identify roll call votes.

To begin, we clean up, install the `wnominate` package the first time we use it, load the library, and load the `UN` data:[9]

```
rm(list=ls())
install.packages("wnominate")
library(wnominate)
data(UN)
```

[9]The UN data is also available in the file `UN.csv`, available in the Dataverse (see page vii) or the chapter content (see page 125)

Once the data are loaded, they can be viewed with the standard commands such as fix, but for a quick view of what the data look like, we could simply type: head(UN[,1:15]). This will show the structure of the data through the first 13 roll call votes.

Before we can apply W-NOMINATE, we have to reformat the data to an object of class rollcall. To do this, we first need to redefine our UN dataset as a matrix, and split the names of the countries, whether the country was in the Warsaw Pact, and the set of roll calls into three separate parts:

```
UN<-as.matrix(UN)
UN.2<-UN[,-c(1,2)]
UNnames<-UN[,1]
legData<-matrix(UN[,2],length(UN[,2]),1)
```

The first line turned the UN data frame into a matrix. (For more on matrix commands in R, see Chap. 10.) The second line created a new matrix, which we have named UN.2, which has eliminated the first two columns (country name and member of Warsaw Pact) to leave only the roll calls. The third line exported the names of the nations into the vector UNnames. (In many other settings, this would instead be the name of the legislator or an identification variable.) Lastly, our variable of whether a nation was in the Warsaw Pact has been saved as a one-column matrix named legData. (In many other settings, this would be a legislator's political party.) Once we have these components together, we can use the rollcall command to define a rollcall-class object that we name rc as follows:

```
rc<-rollcall(data=UN.2,yea=c(1,2,3),nay=c(4,5,6),
     missing=c(7,8,9),notInLegis=0,legis.names=UNnames,
     legis.data=legData,desc="UN Votes",source="voteview.com")
```

We specify our matrix of roll call votes with the data argument. Based on how the data in the roll call matrix are coded, we use the yea, nay, and missing arguments to translate numeric codes into their substantive meaning. Additionally, notInLegis allows us to specify a code that specifically means that the legislator was not a member at the time of the vote (e.g., a legislator died or resigned). We have no such case in these data, but the default value is notInLegis=9, and 9 means something else to us, so we need to specify an unused code of 0. With legis.names we specify the names of the voters, and with legis.data we specify additional variables about our voters. Lastly, desc and source allow us to record additional information about our data.

With our data formatted properly, we can now apply the W-NOMINATE model. the command is simply called wnominate:

```
result<-wnominate(rcObject=rc,polarity=c(1,1))
```

With the rcObject argument, we simply name our properly formatted data. The polarity argument, by contrast, requires substantive input from the researcher: The user should specify a vector that lists which observation should clearly fall on the positive side of each dimension estimated. Given the politics of the Cold

War, we use observation #1, the USA, as the anchor on both dimensions we estimate. By default, `wnominate` places voters in two-dimensional ideological space (though this could be changed by specifying the `dims` option).

To view the results of our estimation, we can start by typing: `summary(result)`. This prints the following output.

```
SUMMARY OF W-NOMINATE OBJECT
----------------------------

Number of Legislators:   59 (0 legislators deleted)
Number of Votes:   219 (18 votes deleted)
Number of Dimensions:   2
Predicted Yeas:   4693 of 5039 (93.1%) predictions
 correct
Predicted Nays:   4125 of 4488 (91.9%) predictions
 correct
Correct Classification:   89.5% 92.56%
APRE:   0.574 0.698
GMP:   0.783 0.841


The first 10 legislator estimates are:
                coord1D coord2D
United States     0.939   0.344
Canada            0.932   0.362
Cuba              0.520  -0.385
Haiti             0.362  -0.131
Dominican Rep     0.796  -0.223
Mexico            0.459   0.027
Guatemala         0.382   0.364
Honduras          0.588  -0.266
El Salvador       0.888  -0.460
Nicaragua         0.876  -0.301
```

The output begins by recapping several descriptive features of our data and then turns to fit indices. It lists, for instance the correct prediction of yeas and nays, and then under "Correct Classification" lists the percent correctly predicted by the first dimension alone and then both dimensions together. At 89.5 %, the first dimension can explain a lot by itself. The output ends by listing the estimates for our first 10 observations. As can be seen, the USA does take a positive value on both dimensions, per our specification in the model.

We can obtain additional output by typing: `plot(result)`. The result of this command is presented in Fig. 8.3. The top-left panel visualizes the W-NOMINATE scores for the 59 voting nations. The horizontal axis is the first dimension of the score, the vertical axis is the second dimension, and each point represents a nation's position. Hence, in the two-dimensional ideological space defined internally to UN

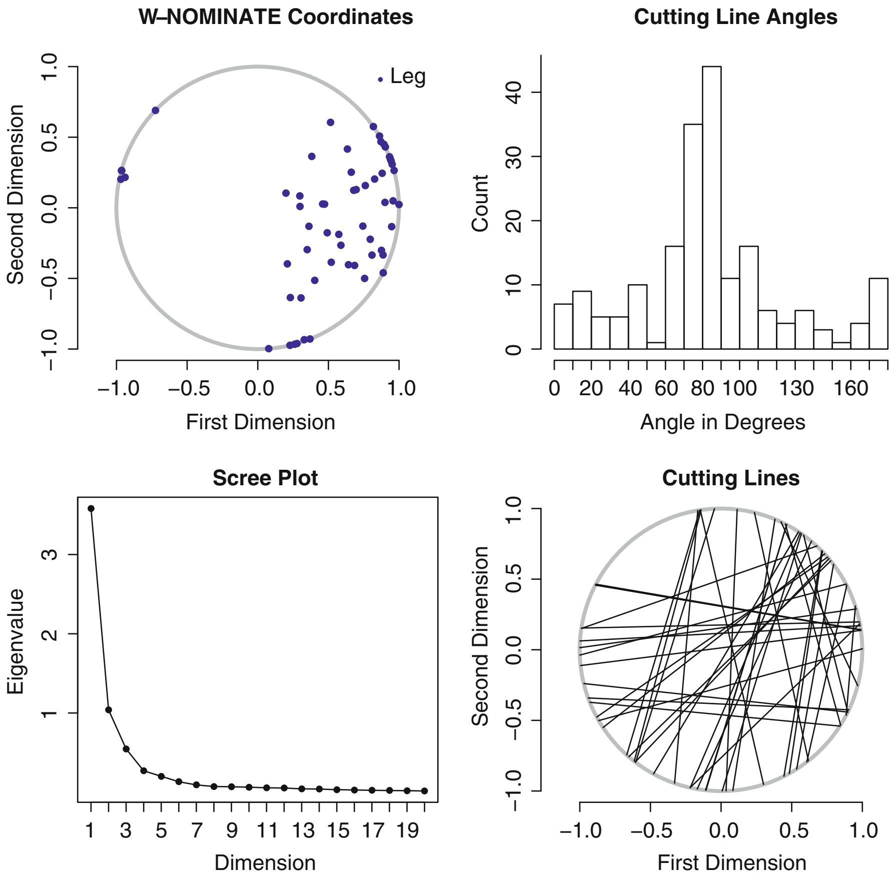


Fig. 8.3 Output plot from estimating W-NOMINATE scores from the first three sessions of the United Nations

proceedings, this is where each nation falls. The bottom-left panel shows a scree plot, which lists the eigenvalue associated with each dimension. Larger eigenvalues indicate that a dimension has more explanatory power. As in all scree plots, each additional dimension has lower explanatory value than the previous one.[10] The top-right panel shows the distribution of the angles of the cutting lines. The cutting lines divide yea from nay votes on a given issue. The fact that so many cutting lines are near the 90° mark indicates that the first dimension is important for many of the

[10]When choosing how many dimensions to include in a measurement model, many scholars use the "elbow rule," meaning they do not include any dimensions past a visual elbow in the scree plot. In this case, a scholar certainly would not include more than three dimensions, and may be content with two. Another common cutoff is to include any dimension for which the eigenvalue exceeds 1, which would have us stop at two dimensions.

votes. Finally, the bottom-right panel shows the Coombs Mesh from this model—a visualization of how all of the cutting lines on the 237 votes come together in a single space.

If the user is satisfied with the results of this measurement model, then it is straightforward to write the scores into a useable data format. Within our `wnominate` output named `result` we can call the attribute named `legislators`, which saves our ideal points for all countries, any non-roll call variables we specified (e.g., Warsaw Pact or not), and a variety of other measures. We save this as a new data frame named `scores` and then write that to a CSV file:

```
scores<-result$legislators
write.csv(scores,"UNscores.csv")
```

Just remember to use the `setwd` command to specify the folder in which you wish to save the output file.

Once we have our W-NOMINATE ideal points in a separate data frame, we can do anything we normally would with data in R, such as draw our own graphs. Suppose we wanted to reproduce our own graph of the ideal points, but we wanted to mark which nations were members of the Warsaw Pact versus those that were not. We could easily do this using our `scores` data. The easiest way to do this might be to use the `subset` command to create separate data frames of our two groups:

```
wp.scores<-subset(scores, V1=="WP")
other.scores<-subset(scores, V1=="Other")
```

Once we have these subsets in hand, we can create the relevant graph in three lines of code.

```
plot(x=other.scores$coord1D, y=other.scores$coord2D,
     xlab="First Dimension", ylab="Second Dimension",
     xlim=c(-1,1), ylim=c(-1,1), asp=1)
points(x=wp.scores$coord1D, y=wp.scores$coord2D,
     pch=3,col='red')
legend(x=-1,y=-.75,legend=c("Other","Warsaw Pact"),
     col=c("black","red"),pch=c(1,3))
```

In the call to `plot`, we graph the 53 nations that were not members of the Warsaw Pact, putting the first dimension on the horizontal axis, and the second on the vertical axis. We label our axes appropriately using `xlab` and `ylab`. We also set the bounds of our graph as running from −1 to 1 on both dimensions, as scores are constrained to fall in these ranges. Importantly, we guarantee that the scale of the two dimensions is the same, as we generally should for this kind of measurement model, by setting the **asp**ect ratio to 1 (`asp=1`). On the second line of code, we use the `points` command to add the six observations that were in the Warsaw Pact, coloring these observations red and using a different plotting character. Lastly, we add a legend.

Figure 8.4 presents the output from our code. This graph immediately conveys that the first dimension is capturing the Cold War cleavage between the USA and its allies versus the Soviet Union and its allies. We specified that the USA would take positive coordinates on both dimensions, so we can see that the Soviet allies

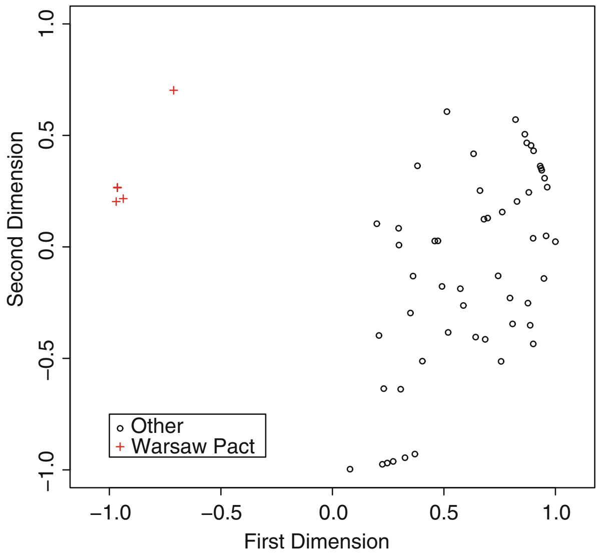


Fig. 8.4 Plot of first and second dimensions of W-NOMINATE scores from the first three sessions of the United Nations. A *red cross* indicates a member of the Warsaw Pact, and a *black circle* indicates all other UN members (Color figure online)

(represented with red crosses) are at the extremes of negative values on the first dimension.

To recap, this chapter has illustrated four examples of how to use R packages to implement advanced methods. The fact that these packages are freely available makes cutting-edge work in political methodology and from a variety of disciplines readily available to any R user. No book could possibly showcase all of the researcher-contributed packages that are available, not least because new packages are being made available on a regular basis. The next time you find yourself facing a taxing methodological problem, you may want to check the CRAN servers to see if someone has already written a program that provides what you need.

8.5 Practice Problems

This set of practice problems considers each of the example libraries in turn, and then suggests you try using a brand new package that has not been discussed in this chapter. Each question calls for a unique data set.

1. Multilevel Logistic Regression: Revisit Singh's (2015) data on voter turnout as a function of compulsory voting rules and several other predictors. If you do not have the file `stdSingh.dta`, please download it from the Dataverse (see page vii) or the chapter content (see page 125). (These data were first introduced in

Sect. 7.4.) Refit this logistic regression model using the `glmer` command, and include a random intercept by country-year (**cntryyear**). Recall that the outcome is turnout (**voted**). The severity of compulsory voting rules (**severity**) is interacted with the first five predictors: 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**). Again, all of the predictor variables have been standardized. What do you learn from this multilevel logistic regression model estimated with `glmer` that you do not learn from a pooled logistic regression model estimated with `glm`?

2. Bayesian Poisson model with MCMC: Determine how to estimate a Poisson model with MCMC using `MCMCpack`. Reload Peake and Eshbaugh-Soha's (2008) data on energy policy news coverage, last discussed in Sect. 7.3. If you do not have the file `PESenergy.csv`, you may download it from the Dataverse (see page vii) or the chapter content (see page 125). Estimate a Bayesian Poisson model in which the outcome is energy coverage (**Energy**), and the inputs are six indicators for presidential speeches (**rmn1173**, **grf0175**, **grf575**, **jec477**, **jec1177**, and **jec479**), an indicator for the Arab oil embargo (**embargo**), an indicator for the Iran hostage crisis (**hostages**), the price of oil (**oilc**), presidential approval (**Approval**), and the unemployment rate (**Unemploy**). Use a Geweke test to determine whether there is any evidence of nonconvergence. How should you change your code in R if nonconvergence is an issue? Summarize your results in a table, and show a density plot of the partial coefficient on Richard Nixon's November 1973 speech (**rmn1173**).
3. Coarsened Exact Matching: In Chap. 5, the practice problems introduced Alvarez et al.'s (2013) data from a field experiment in Salta, Argentina in which some voters cast ballots through e-voting, and others voted in the traditional setting. Load the `foreign` library and open the data in Stata format. If you do not have the file `alpl2013.dta`, you may download it from the Dataverse (see page vii) or the chapter content (see page 125). In this example, the treatment variable is whether the voter used e-voting or traditional voting (**EV**). The covariates are age group (**age_group**), education (**educ**), white collar worker (**white_collar**), not a full-time worker (**not_full_time**), male (**male**), a count variable for number of six possible technological devices used (**tech**), and an ordinal scale for political knowledge (**pol_info**). Use the `cem` library to answer the following:

   a. How balanced are the treatment and control observations in the raw data?
   b. Conduct coarsened exact matching with the `cem` command. How much has the balance improved as a result?
   c. Consider three possible response variables: whether the voter evaluated the voting experience positively (**eval_voting**), whether the voter evaluated the speed of voting as quick (**speed**), and whether the voter is sure his or her vote is being counted (**sure_counted**). What is the average treatment effect on the treated (ATT) on your matched dataset for each of these three responses?

d. How do your estimates of the average treatment effects on the treated differ from simple difference-of-means tests?

4. W-NOMINATE: Back in Sect. 2.1, we introduced Lewis and Poole's roll call data for the 113th US Senate. Consult the code there to read these data, which are in fixed width format. The file name is `sen113kh.ord`, and it is available from the Dataverse (see page vii) and the chapter content (see page 125).

   a. Format the data as a matrix and create the following: a separate matrix just of the 657 roll calls, a vector of the ICPSR identification numbers, and a matrix of the non-roll call variables. Use all of these to create a `rollcall`-class object. The roll call votes are coded as follows: $1 = $ Yea, $6 = $ Nay, 7 & $9 = $ missing, and $0 = $ not a member.
   b. Estimate a two-dimensional W-NOMINATE model for this roll call object. From the summary of your results, report the following: How many legislators were deleted? How many votes were deleted? Was was the overall correct classification?
   c. Examine the output plot of your estimated model, including the W-NOMINATE coordinates and the scree plot. Based on the scree plot, how many dimensions do you believe are sufficient to characterize voting behavior in the 113th Senate? Why?

5. Bonus: Try learning how to use a package you have never used before. Install the `Amelia` package, which conducts multiple imputation for missing data. Have a look at Honaker et al.'s (2011) article in the *Journal of Statistical Software* to get a feel for the logic of multiple imputation and to learn how to do this in R. Fit a linear model on imputed datasets using the `freetrade` data from the `Amelia` library. What do you find?