PLS 300

Recoding Variables in R

The following handout describes how to recode variables from the 2008 NES dataset in R. Why would you want to do this? Several potential reasons:

Variables from raw survey data are rarely ever in usable form. The author of your research methods textbook make this point in his description of analyzing survey data. Changes may need to be made to fit our assumptions regarding level of measurement, to ensure the data are measured at the intended nominal, ordinal, or interval level. Or, missing data may concern us. Unless we are specifically interested in specifically studying people who "don't know" their opinion, we may wish to drop these individuals from the analysis or assign them another score. (There are many other, more sophisticated ways of dealing with the "don't know" responses, but these go way beyond the scope of our class.) We may need to re-scale variables to a different level of measurement (from interval to ordinal, for example), or we may wish to create new, more descriptive verbal labels for existing variables.

Another reason is that we may need to transform the data — with respect to any of these changes — in order to analyze the data with inferential statistics. We may need to make necessary adjustments to the level of measurement or store a particular variable in a different way, as a numeric variable versus a factor, in order to conduct inferential tests, such as a test of a mean difference.

Moreover, all of these purposes are necessary for doing real, professional analysis of data. While time consuming (at least the first time), you can minimize the time you spend doing this by planning out which variables you plan to study, before you sit down to recode variables.

The goal of the following exercise is for you to see how to recode variables for three different situations. From these three, you should be able to apply each to your own individual data analysis needs, using two particular R functions: `recode()` and `cut()`. Follow along with the examples, running each of the R commands as shown. *This step will be critical to your paper. You have to follow through these examples in order to understand how you will do your analysis.*

To follow along with these examples, you will need to keep in mind and do the following:

1. Set up R to load the 2008 ANES data. *Either* use your own NES data from the computer lab assignment, or load these commands:

   ```
   site="http://faculty.gvsu.edu/kilburnw/nes2008.RData"
   load(file=url(site))
   ls()
   ```

2. Next, you will find it useful to have any of the codebooks open in a text file, for reference.

3. Use these commands to load to required packages, `gmodels` and `car`, useful for recoding the data:

   ```
   lab.packages <- c('lattice', 'gmodels', 'car')
   install.packages(pkgs=lab.packages)
   # loads packages into memory
   library(lattice)
   ```

```
library(gmodels)
library(car)
```

4. We will work through a few examples. In your own work, you should think about which of the following examples apply to your own data analysis needs.

5. Copy and paste the R commands in the highlighted font. You should copy and paste it directly into R, beginning with the ones above. Do this for the examples below to follow along.

6. Recognize that there are two variable types that we use in R, called "factors" for qualitative (ordinal or numeric) variables, and "numeric" for quantitative (interval or ratio) variables.

Example A: Recoding a Factor Variable with Different Verbal Labels

First we will look at the question asking individuals for their party identification, "Generally speaking, do you usually think of yourself as a Republican, Democrat, and Independent, or what?" This is variable `V083097`. Take a look:

```
table(nes08$V083097)
```

```
   -9. Refused           -8. Don't know              1. Democrat            2. Republican
             0                        0                      515                      387
3. Independent 4. Other party (SPECIFY)   5. No preference {VOL}
           491                       13                       72
```

Figure 1: The levels are the verbal labels for each recorded survey response. We want to change these levels into more useful labels.

You should see that some people volunteered a response "no preference", while others chose "another party" and specified it to the interviewer. (As users of the survey, however, we do not have those specific responses. All we know is they volunteered a response.)

The variable is stored as a factor. Enter:

```
str(nes08$V083097)
```

which reports that it is a "Factor w/7 levels". As a "Factor", R treats the variable as an ordinal or nominal level measurement. It will not allow you to perform mathematical operations on it until it is converted or treated as a "Numeric" variable. For present purposes, we continue to treat it as a Factor, and inspect the levels:

```
levels(nes08$V083097)
```

It is up to the judgment of the researcher how to proceed with the raw data, but in this case, I think it would be a good idea to create a new variable with the following 'levels' or verbal labels: "Democrat", "Republican", and "Independent". While you may disagree, I think we'll place the

```
> levels(nes08$V083097)
[1] "-9. Refused"             "-8. Don't know"          "1. Democrat"             "2. Republican"
[5] "3. Independent"          "4. Other party (SPECIFY)" "5. No preference {VOL}"
```

Figure 2: The levels are the verbal labels for each recorded survey response. We want to change these levels into more useful labels.

people who said they have "No preference" with the self-described "Independents". The "Other party" people will need to be excluded from the analysis.

The easiest way to do this is with the `recode()` function.

Key steps:

Identify the OLD levels in the variable Use the `levels()` function to identify the levels of the old variable. It is easiest to copy and paste these into the recode function, below. When typing out the new labels, copy and paste the old levels into the recode function. Remember, only Factor variables have levels. Numeric variables do not, so this recoding example only applies to variables stored as Factors.

Imagine the NEW levels for the variable Decide what the new levels will be. In this case, they are "Democrat", "Republican", and "Independent".

Know the syntax for `recode` In the `recode()` function, factors are in single or double quotes, while numeric scores appear without quotes.

Always create a new variable In all cases, create a new variable with a name you can remember. Assign it the contents of the old variable. Here, for example, we would create a new variable called 'pid' and assign it the contents of 'V083097': `nes08$pid<-nes08$V083097`

Here's the basic structure of the recode function:

```
nes08$new variable to create<-recode(name of old variable to recode,"coding rules
(separated by a semicolon);" other options)
```

We will create a new variable, `nes08$pid` as shorthand for "Party Identification", which we assign the contents of the recode function.

Specifically, here's how to create the new variable. We enter the recode command to produce the resulting recoded variable.

```
> nes08$pid<-recode(nes08$V083097, "'1. Democrat'='Democrat'; '2. Republican'='Republican'; '3.
Independent'='Independent'; '5. No preference {VOL}'='Independent'; else=NA")
> table(nes08$pid)

   Democrat Independent  Republican
        515         563         387
```

Figure 3: The recode function for creating a new party identification variable with the desired factor levels.

```
nes08$pid<-recode(nes08$V083097, "'1. Democrat'='Democrat';
'2. Republican'='Republican'; '3. Independent'='Independent';
'5. No preference {VOL}'='Independent'; else=NA")
```

While it may look complicated, when broken down into specific parts, it's easier to see the pattern. The first part, `nes08$pid<-recode(nes08$V083097, )` simply identifies the new variable we want to create, `nes08$pid` and tells R that we would like to recode its values with recoded levels of `nes08$V083097`. The old factor levels appear in single quotes on the left side of each equal sign. First, we take the old coding '1. Democrat' and assign it a new coding in the new pid variable, 'Democrat'. Notice how we assigned *two* of the old codings to the 'Independent' coding in the new variable. The last part, `"else=NA"` converts all other categories (such as the Other Party people) to a missing value. If you have not already, attempt to recode the variable and produce results that match the coding for `nes08$pid`

```
> table(nes08$pid)
```

```
> table(nes08$pid)

   Democrat Independent  Republican
        515         563         387
>
```

Figure 4: Results of the recoded variable for party identification. Yours should match it.

Important Part of Example A: Changing the order of the factor levels

One additional command in the `recode()` function changes the order of the factor levels, as displayed. Notice the difference between the two versions of the pid variable below. The first example below shows the order of the factors as we recoded it. The factor levels 'Democrat', 'Independent', and 'Republican' follow each other alphabetically, and so they just happen to be ordered in a meaningful way. To change this order, we have to use an option at the end of the function, the `levels=c()` option. The order of the levels are collected in this statement at the end. To change the order of the factor levels, you would just change the order of the factors listed in `levels=c()`.

In the example for pid (party identification), the factor levels 'Democrat', 'Independent', and 'Republican' are already in alphabetical order, so it makes little difference here. But to switch the order around to have 'Republicans' first:

```
nes08$pid<-recode(nes08$V083097, "'1. Democrat'='Democrat';
'2. Republican'='Republican'; '3. Independent'='Independent';
'5. No preference {VOL}'='Independent'; else=NA",
levels=c('Republican', 'Independent', 'Democrat'))
```

Another example appears in the following figure.

```
> nes08$pid<-recode(nes08$V083097, "'1. Democrat'='Democrat'; '2. Republican'='Republican'; '3.
Independent'='Independent'; '5. No preference {VOL}'='Independent'; else=NA")
> table(nes08$pid)

   Democrat Independent  Republican
        515         563         387
> nes08$pid<-recode(nes08$V083097, "'1. Democrat'='Democrat'; '2. Republican'='Republican'; '3.
Independent'='Independent'; '5. No preference {VOL}'='Independent'; else=NA", levels=c('Independent',
'Republican', 'Democrat'))
> table(nes08$pid)

Independent  Republican    Democrat
        563         387         515
>
```

Figure 5: Two versions of the pid variable, with different sorted orders of the factor levels.

Problem 1 for Example A: Recoding "angry" feelings toward the federal government

Take the variable, `nes08$V085204`. Use the example from above to create a new variable, `nes08$fedanger`, where the only codings are the following: 'No' responses and 'Yes' responses. Use the example above. After you have created your new variable, compare it to my example in the script file.

Problem 2 for Example A: Recoding "happy" or "sad" feelings toward a Republican presidential victory

Next, take variable `nes08$V083017`. Use `table(nes08$V083017)` to observe how the factor levels are out of order. (In addition, note that while `table()` lists the factor levels with no observations, for "Refused", and "Don't know", the other tabling function we used, `CrossTable(nes08$V083017)`, does not. — `CrossTable()`requires first loading the 'gmodels' package. Add it to your package statement prior to using it.

Create a new variable, called 'happyR', stored as a factor, where the three levels are in correct order — from sad, to neither, to happy.

You should be able to do this one by yourself, without the help of an example in a script file.

Example B: Recoding a Factor Variable into a Numeric Variable – and Vice Versa

Other times, we may wish to treat a factor variable as a numeric variable. – Perhaps we wish to analyze the variable with interval assumptions, and calculate means. To do this, we must first convert it into a numeric variable.

First, notice that some variables are already stored as numeric, such as the feeling thermometers: `str(nes08$V085063t)`. Most variables are stored as factors, however. The questions measuring party identification are factors, and run across three different variables, beginning with nes08$V083097. Look at the codebook entries for these three variables, from this one to nes08$V083098b. The three variables measure different parts of party identification.

We can create a party identification variable on a seven point scale:

```
# Creating a 7 point party identification scale from the following
# three questions about party identification and strength of identification.
nes08$partyid<-NA
nes08$partyid[nes08$V083097=='1. Democrat' & nes08$V083098a=='1. Strong']<-1
nes08$partyid[nes08$V083097=='1. Democrat' & nes08$V083098a=='5. Not very strong' ]<-2
nes08$partyid[nes08$V083098b=='5. Closer to Democratic']<-3
nes08$partyid[nes08$V083098b=='3. Neither [VOL]']<-4
nes08$partyid[nes08$V083097=='3. Independent']<-4
nes08$partyid[nes08$V083098b=='1. Closer to Republican' ]<-5
nes08$partyid[nes08$V083097=='2. Republican' & nes08$V083098a=='5. Not very strong']<-6
nes08$partyid[nes08$V083097=='2. Republican' & nes08$V083098a=='1. Strong']<-7
```

```
> nes08$partyid<-NA
> nes08$partyid[nes08$V083097=='1. Democrat' & nes08$V083098a=='1. Strong']<-1
> nes08$partyid[nes08$V083097=='1. Democrat' & nes08$V083098a=='5. Not very strong' ]<-2
> nes08$partyid[nes08$V083098b=='5. Closer to Democratic']<-3
> nes08$partyid[nes08$V083098b=='3. Neither [VOL]']<-4
> nes08$partyid[nes08$V083097=='3. Independent']<-4
> nes08$partyid[nes08$V083098b=='1. Closer to Republican' ]<-5
> nes08$partyid[nes08$V083097=='2. Republican' & nes08$V083098a=='5. Not very strong']<-6
> nes08$partyid[nes08$V083097=='2. Republican' & nes08$V083098a=='1. Strong']<-7
>
> table(nes08$partyid)

  1   2   3   4   5   6   7
291 224  22 383 181 207 179
```

Figure 6: Results of the recoded variable for a 7-point party identification scale. Yours should match it.

Notice the subsetting commands in each statement. For example, in the second line, we tell R to assign the 'partyid' variable a 1 only if the respondent identified as a 'Democrat' in one variable and then 'Strong' in the second variable. The variable is stored as a numeric variable. This storage type is useful, because it allows us to perform various mathematical operations on it. The only drawback is that we have to mentally remember that the scale ranges from Strong Democrat to Strong Republican. After creating the 7-point scale, try producing some basic summary statistics: `summary(nes08$partyid)`. Further, looking ahead to inferential statistics, calculate a t-test: `t.test(nes08$partyid)`. Attempting either of these commands with a factor variable would result in an error message.

We can recode the variable to a factor, with the labels — factor levels — associated with the With this variable, we can assign each of the points on the scale, ranging from 'Strong Democrat' to 'Strong Republican' a factor label. This too, is easy to do with the `recode()` function.

```
nes08$partyid2<-recode(nes08$partyid, "1='Strong Democrat' ; 2='Weak Democrat';
3='Ind. Lean Dem.'; 4='Independent'; 5='Ind. Lean Rep.'; 6='Weak Republican';
7='Strong Republican'",
as.factor.result=TRUE, levels=c('Strong Democrat','Weak Democrat','Ind. Lean Dem.',
'Independent','Ind. Lean Rep.','Weak Republican','Strong Republican') )
```

```
> nes08$partyid2<-recode(nes08$partyid, "1='Strong Democrat' ; 2='Weak Democrat'; 3='Ind. Lean Dem.'; 4='Independent';
5='Ind. Lean Rep.'; 6='Weak Republican'; 7='Strong Republican'", as.factor.result=TRUE, levels=c('Strong Democrat','Weak
Democrat','Ind. Lean Dem.','Independent','Ind. Lean Rep.','Weak Republican','Strong Republican') )
>
> str(nes08$partyid2)
 Factor w/ 7 levels "Strong Democrat",..: 6 2 1 4 2 4 1 7 5 5 ...
> table(nes08$partyid2)

  Strong Democrat     Weak Democrat    Ind. Lean Dem.       Independent    Ind. Lean Rep.   Weak Republican
              291               224                22               383               181               207
Strong Republican
              179
>
```

Figure 7: Recoding the 7 point scale into a factor stored variable, with labels. Notice the as.factor.result=TRUE option. This command ensures the result is stored as a factor variable.

Once we have a factor variable, it is easy to convert it back into a numeric variable:

```
nes08$partyid3<-as.numeric(nes08$partyid2)
```

The `as.numeric()` command tells R to store the variable as a numeric variable.

Other examples with the 2008 ANES dataset

recoding variables for missing values Let's first create a new variable for Condaleeza Rice's feeling thermometer score. `nes08$condift<-nes08$V083042`.

I cleaned up these variables a bit for you. So this is not an issue here, but if any respondents refused to answer the question, one easy way to get rid of these responses is to use `>nes08$condift[nes08$condift>100]<-NA`. The condition inside the brackets `[]` tells you what values of the variable should be recoded, in this case, all responses greater than 100 should be given a missing value, indicated by the `<-NA`. Any other numeric value would valid as a replacement for NA.

For preparing your cross-tabs, this command is very useful. For any variable for which you would like to have 'don't know' or 'refusal' responses coded to missing, you can just tell R to make values such as those greater than 7 or 100, or whatever it is, to be assigned missing values.

Transforming a numerical, continuous (interval or ratio) scale variable into a factor This is accomplished with the `cut()` function and the `recode()` function. The `cut()` function as two arguments.

Example C

converting a numeric, continuous variable into discrete groups You may find it useful to convert variables that are continuous, such as feeling thermometers, into variables that contain only a few levels, such as 'cool', 'lukewarm', or 'hot'. This could be useful in a cross-tabulation. To do so, we use the `cut` function to create a new variable that is stored as a factor with the specified labels.

The option `labels=c(Low,Medium,High)` adds specific labels to each of the factor levels. Try first creating a new variable for any of the continuous variables, such as a feeling thermometer score for Bush, and then convert it to a factor with three levels of equal length along the feeling thermometer scale.

```
nes08$condift2<-cut(nes08$condift, 3, labels=c('cool', 'medium', 'hot'))
```

In this example above, we told R to create a new variable, 'condift', which contains 3 categories. The cut command takes *roughly equal length widths along the 0-100 scale* and places respondents in each of three groups. In this case, respondents from 0-30 were grouped into the 'cool' category.

Then check `table(nes08$condift2)`. Or cross-tabulate it with the original variable to see which numeric points on the thermometer scale were included with which of the three categories.

An alternative is to assign *equal groups of respondents along the scale*, we use the `quantile()` function.

```
nes08$condift3<-cut(nes08$condift, quantile(nes08$condift,
(0:3)/3, na.rm=T), c('cool', 'lukewarm', 'hot'))
```

In this example, the ,`(0:3)/3` tells R to construct 3 approximately equal numbers of observations along 3 cut points, 1/3, 2/3, and 3/3. The quantile function requires that you exclude missing observations with the na.rm=T option. Tabulate this variable and compare it with the original `condift`.

removing a variable Our NES dataset is stored as a data frame in R; to remove a variable created within it, we use the following command, assigning it NULL. Just using the 2004 data as an example:

`nes2004$varname<-NULL` , such as `$>$nes2004$kerryft<-NULL`

Problem 3 for Example C

Take any of the feeling thermometer variables, with the exception of Rice, and create a categorical variable with three or four factor levels, using the examples from above. You should be able to do this without aid of an additional script file.

One last example of a potential obstacle in recoding variables

```
# One other complication in recoding variables may occur when you
# have single quotes in the factor levels.

# Here's one example:
table(nes08$V083007)

# notice the quote in the factor levels after the variable is tabled
# To recode it we will first create a numeric version of the variable.
```

```
nes08$vote04<-as.numeric(nes08$V083007)

table(nes08$vote04)

# Then we will reassign it the appropriate factor labels
nes08$vote04<-recode(nes08$vote04, "3='Yes, voted'; 4='No, didnt vote';
else=NA", as.factor.result=TRUE)?
table(nes08$vote04)
```

Review: Tabulations and Cross-Tabulations

To tabulate variables, or to do cross-tabulations like in class, we first need to install the `gmodels` package. Enter the commands: `<library(gmodels)`. We will use the function `CrossTable`. (After the library is loaded, you can type `>help(CrossTable)` to bring up the various options for this function.)

The function is useful for producing frequency tabulations. Try this command:

`>CrossTable(nes2004$v043025, max.width=1)` or max.width=2 for double column display

The `max.width` command is not always necessary, but without it CrossTable will make some assumptions you may not like about how to display the results. Note that CrossTable provided proportions for each category, but if you prefer percentages, try the 'SPSS' formatting option: `format=c("SPSS")`

`>CrossTable(nes2004$v043025, max.width=2, format=c("SPSS"))`

Or to produce a cross–tabulation of variables named 'bushjob' and 'pid':
`>CrossTable(nes2004$bushjob, nes2004$pid, digits=2, format=c("SPSS")`

If one variable has longer value labels than the other, list it first in the command so that it appears as the row variable. You'll see that in addition to frequency counts and row and column percentages, the table also includes a measure of each cell's contribution to a Chi-Square test of independence. To turn off this option, add `prop.chisq=F` to the `CrossTable()` function. The `digits=2` limits the display to two decimal places.

So with this brief overview of recoding and doing cross-tabulations, you should be prepared to do some of your own cross-tabs for the homework assignment. In addition, you'll want to look through the codebooks for the NES to identify any potential variables you might want to use in your research paper.