

# Examples of Re-Organizing Data Using R

Danny Kaplan

February 26, 2009

## Contents

### Re-Organizing Data

The data we receive are not always organized in a manner that supports the analysis that we have to do.

R includes facilities for

- Combining and separating tables.
- Recoding variables.
- Creating new organizations for variables.
- Documenting our re-organization so that we can show it is correct or fix it if it is not.
- Looking at the re-organized data to make sure we have done so right.

### The Osteoporosis Data

Goal of the analysis: See how the various scales for anxiety, depression, and anger are related to one another as the seasons change. Which one's show the greatest seasonal variability? How much of variability is tied to that of other scales? How much variability is typical in a person?

- There are multiple measurements for each person, but these are stored as multiple variables rather than multiple cases.
- For some information (sex, marital status, profession) the natural case is an individual person.

### Organization of Software

- Some software packages (e.g., STATA) provide special operators for particular techniques.
- People often think of these special operators as somehow intrinsic to the calculations.

- Often, the point of the operators is just to re-organize data so that they can be analyzed using a general purpose technique.
- Example: **Panel Data**. This is mainly just ordinary linear modeling (or mixed effects modeling), but the data have to be organized in the right way to see this.

This example is about the re-organization of data, but that's called for because I want to use a general-purpose model fitting approach.

### What do you expect?

- We all want the computer command `do_what_I_want`.
- In fact, all there is is `do_what_I_say`
- Often, what we say to the computer is not what we want.
- We need to be careful to check that what we say is what we want.
- Good systems make it straightforward to say what we want. But often what we want is complicated, and the operators in the system don't already implement it.

RESULT: We have to work. This example will track my mistakes, because a major part of working with the computer is making mistakes, identifying them, and then fixing them.

### A Desired Reorganization into Two Tables

Where the **person** is the case.

ID	Marital	Schooling	Age
8	married	HS grad	34
53	cohab	bachelors	23
57	divorced	grad school	45

Where the **occasion for measurement** is the case

ID	When	Anxiety	Depression	BMI	Hostility
8	Winter	27	14	23	18
8	Spring	18	6	24	11
8	Summer	21	3	24	17
8	Fall	32	19	23	23
53	Winter	11	7	27	19
53	Spring	9	3	28	18

## The Approach

- Verify that the data are organized as we expect.
- Pull out the variables for the first table.
- Recode as appropriate for better readability.
- Pull out the variables for the second table.
- Reshape the second table so that the case is “occasion for measurement.”

## Validating the Data

Is there a unique ID for each person?

```
> nrow(osteo)
```

```
[1] 57
```

```
> length(unique(osteo$id))
```

```
[1] 57
```

Is BMI in the range expected?

```
> summary(osteo$bmi)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
18.85	23.79	25.73	27.52	28.06	51.04	9.00

... and so on.

## Pull out Variables for the First Table

We'll call this new data frame “person” since it has information with the individual person as the case

```
> person = subset(osteo, select = c(id, mstatus,  
  schoolyr, age, sayhlth))
```

```
> head(person)
```

	id	mstatus	schoolyr	age	sayhlth
1	8	2	5	63	3
2	10	6	2	64	4
3	50	2	3	60	NA
4	73	2	5	65	3
5	77	5	5	61	2
6	166	1	3	66	2

## Re-coding Marital Status

- Originally coded as numbers: 1 through 6.
- The codebook tells us what the numbers mean.
- There are actually no 4s in the data, but we want to respect the original code scheme.

```
> marital = factor(person$mstatus, levels = 1:6,
  labels = c("single", "married", "cohabit",
    "sep.", "divorced", "widowed"))
```

Notes: First argument is the variable with the original codes. "levels" argument is the possible values of the levels in the original codes. "labels" argument contains the new codes as character strings.

## Re-coding Marital Status: Verifying the Work

Checking the recoding. We don't want to make a mistake.  
Simple test: Compare the old labels to the new ones:

```
> table(marital, person$mstatus)
```

```
marital      1  2  3  5  6
single      5  0  0  0  0
married     0 34  0  0  0
cohabit     0  0  1  0  0
sep.        0  0  0  0  0
divorced    0  0  0  3  0
widowed     0  0  0  0 14
```

## Exercise: Re-coding the Health

Variable `sayhlth`. Codebook says:

Code	Meaning
1	Excellent
2	Very good
3	Good
4	Fair
5	Poor

These are ordinal data, so we'll tell `factor` that:

```
> health = factor(person$sayhlth, levels = 1:5,
  labels = c("excel", "vgood", "good", "fair",
    "poor"), ordered = TRUE)
```

Exercise: Check whether we got these right.

### Finish the recoding

Put the new variables in place of the old ones

```
> person$sayhlth = health
> person$mstatus = marital
```

### Pull out the second table

To keep our life simple, let's work with one of the variables (anxiety) to sort things out and master the patterns. Then go back and add in the other variables.

```
> measurements = subset(osteo, select = c(id, beck_a1,
      beck_a2, beck_a3, beck_a4))
> head(measurements)
```

	id	beck_a1	beck_a2	beck_a3	beck_a4
1	8	1.000000	1.500000	0.000000	3
2	10	5.000000	5.500000	17.000000	14
3	50	2.000000	3.000000	4.214738	3
4	73	4.194336	3.500000	3.000000	0
5	77	8.000000	2.029088	8.000000	5
6	166	2.000000	1.000000	2.500000	4

### Re-Shaping the Data

- We want there to be one case for each measurement occasion. So, collapse the four `beck_a` variables into a single variable and add another variable that tells WHEN the measurement was made.
- It's easy to make a mistake in telling R what we want to do.
- So do it bit by bit, looking at the results on a small subset of the data to make sure that we are getting what we want. I'll define a tiny data set for development purposes, one small enough that I can see everything

```
> tiny = head(measurements, 2)
> tiny
```

	id	beck_a1	beck_a2	beck_a3	beck_a4
1	8	1	1.5	0	3
2	10	5	5.5	17	14

### The Reshape Operator

R has a large collection of specialized operators. Often these are very sophisticated and the documentation is a challenge.

Trial and error is a good approach to sorting out what the documentation means.

Try one new argument at a time.

```
> reshape(tiny, idvar = "id", direction = "long",
          varying = list(2:5))
```

	id	time	beck_a1
8.1	8	1	1.0
10.1	10	1	5.0
8.2	8	2	1.5
10.2	10	2	5.5
8.3	8	3	0.0
10.3	10	3	17.0
8.4	8	4	3.0
10.4	10	4	14.0

Add in the season names, for readability

```
> reshape(tiny, idvar = "id", direction = "long",
          varying = list(2:5), times = c("winter", "spring",
          "summer", "fall"))
```

	id	time	beck_a1
8.winter	8	winter	1.0
10.winter	10	winter	5.0
8.spring	8	spring	1.5
10.spring	10	spring	5.5
8.summer	8	summer	0.0
10.summer	10	summer	17.0
8.fall	8	fall	3.0
10.fall	10	fall	14.0

Then rename the variable as appropriate

```
> reshape(tiny, idvar = "id", direction = "long",
          varying = list(2:5), times = c("winter", "spring",
          "summer", "fall"), v.names = c("anxiety"))
```

	id	time	anxiety
8.winter	8	winter	1.0
10.winter	10	winter	5.0
8.spring	8	spring	1.5
10.spring	10	spring	5.5
8.summer	8	summer	0.0
10.summer	10	summer	17.0
8.fall	8	fall	3.0
10.fall	10	fall	14.0

**Expand the test case a bit**

```

> measurements = subset(osteo, select = c(id, beck_a1,
      beck_a2, beck_a3, beck_a4, beck_d1, beck_d2,
      beck_d3, beck_d4, bmi1, bmi2, bmi3, bmi4))
> head(measurements)

      id beck_a1 beck_a2  beck_a3 beck_a4  beck_d1
1    8 1.000000 1.500000 0.000000      3 6.313733
2   10 5.000000 5.500000 17.000000     14 9.000000
3   50 2.000000 3.000000 4.214738      3 2.108095
4   73 4.194336 3.500000 3.000000      0 9.458095
5   77 8.000000 2.029088 8.000000      5 11.000000
6  166 2.000000 1.000000 2.500000      4 1.058610
      beck_d2  beck_d3  beck_d4  bmi1  bmi2  bmi3
1 6.500000 8.088270 8.393925 21.37927 21.45102 21.37927
2 12.500000 28.000000 23.000000 36.13845 35.59746 35.48926
3 2.000000 1.000000 1.000000 25.97478 25.68183 25.38888
4 4.079463 4.000000 5.258610 24.03095 23.85930 23.51600
5 6.867023 10.000000 8.408610 25.12771 25.21092 24.46208
6 3.158095 7.092645 5.258722 27.95666 28.28948 28.03987
      bmi4
1 21.37927
2 35.70565
3 25.38888
4 23.85930
5 24.79490
6 27.62385

> tiny = head(measurements, 2)

Try reshaping all three variables

> reshape(tiny, idvar = "id", direction = "long",
      varying = list(2:5, 6:9, 10:13), times = c("winter",
      "spring", "summer", "fall"), v.names = c("anxiety",
      "depression", "bmi"))

      id  time anxiety depression  bmi
8.winter  8 winter     1.0   6.313733 21.37927
10.winter 10 winter     5.0   9.000000 36.13845
8.spring   8 spring     1.5   6.500000 21.45102
10.spring 10 spring     5.5  12.500000 35.59746
8.summer   8 summer     0.0   8.088270 21.37927
10.summer 10 summer    17.0  28.000000 35.48926
8.fall     8  fall      3.0   8.393925 21.37927
10.fall    10  fall    14.0  23.000000 35.70565

```

Good. It seems to be working. Now expand to the whole data set.

```
> occasions = reshape(measurements, idvar = "id",
  direction = "long", varying = list(2:5, 6:9,
    10:13), times = c("winter", "spring",
    "summer", "fall"), v.names = c("beck.anxiety",
    "beck.depression", "bmi"))
```

Let's check more carefully

```
> aggregate(occasions$beck.anxiety, list(season = occasions$time),
  mean, na.rm = TRUE)
```

```
  season      x
1  fall 4.343532
2 spring 4.557675
3 summer 4.764289
4 winter 4.797983
```

```
> mean(osteo$beck_a1, na.rm = TRUE)
```

```
[1] 4.797983
```

```
> mean(osteo$beck_a4, na.rm = TRUE)
```

```
[1] 4.343532
```

### Finally ... The whole thing

```
> measurements = subset(osteo, select = c(id, beck_a1,
  beck_a2, beck_a3, beck_a4, beck_d1, beck_d2,
  beck_d3, beck_d4, bmi1, bmi2, bmi3, bmi4,
  host1, host2, host3, host4, anger1, anger2,
  anger3, anger4, irrit1, irrit2, irrit3, irrit4,
  anxiety1, anxiety2, anxiety3, anxiety4, dirwdc1,
  dirwdc2, dirwdc3, dirwdc4, dirwec1, dirwec2,
  dirwec3, dirwec4))
```

Notice that there are 9 sets of variables. Now convert these to "long" form.

```
> occasions = reshape(measurements, idvar = "id",
  direction = "long", varying = list(2:5, 6:9,
    10:13, 14:17, 18:21, 22:25, 26:29, 30:33),
  times = c("winter", "spring", "summer", "fall"),
  v.names = c("beck.anxiety", "beck.depression",
    "bmi", "hostility", "anger", "anxiety",
    "sun.weekday", "sun.weekend"))
```

Check again

```
> aggregate(occasions$sun.weekend, list(season = occasions$time),
            mean, na.rm = TRUE)
```

```
   season      x
1  fall 0.5633715
2 spring 1.0741815
3 summer 1.6505916
4 winter 0.3589361
```

```
> mean(osteo$dirwec1, na.rm = TRUE)
```

```
[1] 0.3325
```

```
> mean(osteo$dirwec2, na.rm = TRUE)
```

```
[1] 1.135934
```

```
> mean(osteo$dirwec3, na.rm = TRUE)
```

```
[1] 2.449167
```

```
> mean(osteo$dirwec4, na.rm = TRUE)
```

```
[1] 0.748366
```

Something's wrong.

Now go find out what. I went through looking column by column.

```
> head(occasions$beck.depression)
```

```
[1] 6.313733 9.000000 2.108095 9.458095 11.000000
[6] 1.058610
```

```
> head(osteo$beck_d1)
```

```
[1] 6.313733 9.000000 2.108095 9.458095 11.000000
[6] 1.058610
```

```
> head(occasions$bmi)
```

```
[1] 21.37927 36.13845 25.97478 24.03095 25.12771 27.95666
```

```
> head(osteo$bmi1)
```

```
[1] 21.37927 36.13845 25.97478 24.03095 25.12771 27.95666
```

and so on until I discovered that I left the "irritability" out of the names of variables. There were actually only 8 variables that I converted.

```
> occasions = reshape(measurements, idvar = "id",
  direction = "long", varying = list(2:5, 6:9,
    10:13, 14:17, 18:21, 22:25, 26:29, 30:33,
    34:37), times = c("winter", "spring",
    "summer", "fall"), v.names = c("beck.anxiety",
    "beck.depression", "bmi", "hostility",
    "anger", "anxiety", "irritability", "sun.weekday",
    "sun.weekend"))
```

Check again

```
> aggregate(occasions$sun.weekend, list(season = occasions$time),
  mean, na.rm = TRUE)
```

```
  season      x
1  fall 0.748366
2 spring 1.135934
3 summer 2.449167
4 winter 0.332500
```

```
> mean(osteo$dirwec1, na.rm = TRUE)
```

```
[1] 0.3325
```

```
> mean(osteo$dirwec2, na.rm = TRUE)
```

```
[1] 1.135934
```

```
> mean(osteo$dirwec3, na.rm = TRUE)
```

```
[1] 2.449167
```

```
> mean(osteo$dirwec4, na.rm = TRUE)
```

```
[1] 0.748366
```

A Win! I should check some other cases to make sure that I have what I want. I should document that what I have is right.

### Documenting that you have done the right thing

Professional systems provide some certificate of correctness. One way to do this is to provide statements of what should be true if the process is correct. This is often tedious.

Here's the sort of thing to do. Define a set of checks that return TRUE if things are working ...

```

> check.var = function(orig, new) {
  origmeans = aggregate(occasions[[new]], list(season = occasions$time),
    mean, na.rm = TRUE)
  winter = mean(osteo[[paste(orig, "1", sep = "")]],
    na.rm = TRUE)
  spring = mean(osteo[[paste(orig, "2", sep = "")]],
    na.rm = TRUE)
  summer = mean(osteo[[paste(orig, "3", sep = "")]],
    na.rm = TRUE)
  fall = mean(osteo[[paste(orig, "4", sep = "")]],
    na.rm = TRUE)
  return(c(subset(origmeans, season == "winter")$x ==
    winter, subset(origmeans, season == "spring")$x ==
    spring, subset(origmeans, season == "summer")$x ==
    summer, subset(origmeans, season == "fall")$x ==
    fall))
}
> check.var("beck_a", "beck.anxiety")

[1] TRUE TRUE TRUE TRUE

> check.var("beck_d", "beck.depression")

[1] TRUE TRUE TRUE TRUE

```

... and so on.

Hardly anyone does this. It's probably not worthwhile for a single file, but it might be if you routinely deal with repeated files.

### Finishing Up!

Now to study the relationship between the various indices.

- Does anxiety vary with the seasons?

```

> xtable(anova(lm(beck.anxiety ~ time + as.factor(id),
  data = occasions)))

```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
(Intercept)	1	4315.86	4315.86	637.35	0.0000
time	3	6.66	2.22	0.33	0.8053
as.factor(id)	56	7618.06	136.04	20.09	0.0000
Residuals	143	968.33	6.77		

- Does BMI vary with the seasons?

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
(Intercept)	1	3654.58	3654.58	1967.41	0.0000
time	3	12.77	4.26	2.29	0.0810
as.factor(id)	56	383.47	6.85	3.69	0.0000
Residuals	137	254.49	1.86		

```
> xtable(anova(lm(irritability ~ time + as.factor(id),
  data = occasions)))
```

•

... and so on.

### Do we get crabbiier as we get older?

Age is recorded in the “person” data frame. We need to connect that frame to the “occasions” frame. This is done with a `merge` operation:

```
> head(person, 2)
```

```
  id mstatus schoolyr age sayhlth
1  8 married      5  63    good
2 10 widowed      2  64    fair
```

```
> head(occasions, 2)
```

```
      id  time beck.anxiety beck.depression    bmi
8.winter  8 winter          1          6.313733 21.37927
10.winter 10 winter          5          9.000000 36.13845
      hostility anger anxiety irritability sun.weekday
8.winter      NA  NA     NA           NA  0.9166667
10.winter      NA  NA     NA           NA  0.0000000
      sun.weekend
8.winter  0.1666667
10.winter 0.0000000
```

```
> together = merge(person, occasions)
```

```
> head(together, 2)
```

```
  id mstatus schoolyr age sayhlth  time beck.anxiety
1  8 married      5  63    good winter          1
2  8 married      5  63    good  fall          3
  beck.depression    bmi hostility anger anxiety
1          6.313733 21.37927      NA  NA     NA
2          8.393925 21.37927      2   2     2
  irritability sun.weekday sun.weekend
1           NA  0.9166667  0.1666667
2           2  1.8750000  4.5000000
```

Now we are in a position to fit some models:

```
> xtable(anova(lm(irritability ~ age + as.factor(id) +
  time, data = together)), floating = FALSE)
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
(Intercept)	1	3654.58	3654.58	1967.41	0.0000
age	1	2.17	2.17	1.17	0.2813
as.factor(id)	55	379.55	6.90	3.72	0.0000
time	3	14.52	4.84	2.61	0.0543
Residuals	137	254.49	1.86		

### What to learn from this?

- When data are organized in the right format, they can be easy to analyze.
- Getting them into the right format can be hard.

### ANOTHER EXAMPLE

Computing new quantities from the existing variables in the Basic Skills Test data in order to facilitate modeling.

### A Modeling Example

How is the average score of Limited English Proficiency students related to district-wide performance and how to demographics?

#### A NAÏVE APPROACH

We have separate average math scores for non-LEP students. Model these as a function of LEP percentage.

```
> summary(lm(mavnolep ~ lep, data = bst))
```

Call:

```
lm(formula = mavnolep ~ lep, data = bst)
```

Residuals:

```
      Min       1Q   Median       3Q      Max
-48.7339  -7.7436   0.1777  10.7300  38.1661
```

Coefficients:

```
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  634.0339     0.9848  643.793  <2e-16 ***
lep          -0.2925     0.1959  -1.493   0.137
---
```

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 15.32 on 288 degrees of freedom  
 Multiple R-squared: 0.00768, Adjusted R-squared: 0.004235  
 F-statistic: 2.229 on 1 and 288 DF, p-value: 0.1365

Suggests that LEP presence hurts that of non-LEP students. BUT ... what if you control for district-wide properties?

```
> summary(lm(mavnolep ~ mathavg + lep, data = bst))
```

Call:

```
lm(formula = mavnolep ~ mathavg + lep, data = bst)
```

Residuals:

	Min	1Q	Median	3Q	Max
	-10.7017	-0.5212	-0.2802	0.2806	6.3641

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	-8.581783	3.789129	-2.265	0.0243 *
mathavg	1.014201	0.005978	169.651	<2e-16 ***
lep	0.349189	0.019866	17.577	<2e-16 ***

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.525 on 287 degrees of freedom  
 Multiple R-squared: 0.9902, Adjusted R-squared: 0.9901  
 F-statistic: 1.45e+04 on 2 and 287 DF, p-value: < 2.2e-16

### Math Performance and LEP

- Unfortunately, the two quantities we are given — district average scores and average scores for non-LEP students — are mathematically related to the LEP demographics.
- We can separate them using a mathematical relationship:

$$x_{\text{overall}} = x_{\text{LEP}} \times \% \text{ LEP} + x_{\text{NotLEP}} \times (1 - \% \text{ LEP})$$

Or, in R:

```
> bst$mathlep = (bst$mathavg - bst$mavnolep * (1 -
  bst$lep))/bst$lep
```

- This applies only to districts with some LEP students taking the test:

```
> LEP = subset(bst, lep > 0)
```

Now look at the LEP math performance versus LEP percentage:

```

> summary(lm(mathlep ~ lep, data = LEP))

Call:
lm(formula = mathlep ~ lep, data = LEP)

Residuals:
    Min       1Q   Median       3Q      Max
-47.0755  -8.5231  -0.2318   7.7097  28.9769

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 638.8470     1.2976  492.31 < 2e-16 ***
lep         -0.7247     0.1794   -4.04 8.83e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 12.44 on 138 degrees of freedom
Multiple R-squared:  0.1058,    Adjusted R-squared:  0.09929
F-statistic: 16.32 on 1 and 138 DF,  p-value: 8.83e-05

```

and

```

> summary(lm(mathlep ~ mavnolep + lep, data = LEP))

Call:
lm(formula = mathlep ~ mavnolep + lep, data = LEP)

Residuals:
    Min       1Q   Median       3Q      Max
-5.29108 -0.20981  0.02047  0.29095  2.38161

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.21976     3.35235   0.066   0.948
mavnolep     0.99852     0.00524 190.555 <2e-16 ***
lep          0.01552     0.01170   1.327   0.187
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7653 on 137 degrees of freedom
Multiple R-squared:  0.9966,    Adjusted R-squared:  0.9966
F-statistic: 2.031e+04 on 2 and 137 DF,  p-value: < 2.2e-16

```

Looks like the LEP student performance just follows that of the non-LEP students: LEP has little to do with it.

## How about Writing?

```
> LEP$writelep = (LEP$writeavg - LEP$wavnolep *  
  (1 - LEP$lep))/LEP$lep  
> summary(lm(writelep ~ lep, data = LEP))
```

Call:

```
lm(formula = writelep ~ lep, data = LEP)
```

Residuals:

	Min	1Q	Median	3Q	Max
	-1.95509	-0.11237	-0.01454	0.08580	0.38400

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	3.217789	0.022436	143.420	<2e-16 ***
lep	-0.001275	0.003101	-0.411	0.682

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.215 on 138 degrees of freedom  
Multiple R-squared: 0.001224, Adjusted R-squared: -0.006014  
F-statistic: 0.1691 on 1 and 138 DF, p-value: 0.6815

```
> summary(lm(writelep ~ wavnolep + lep, data = LEP))
```

Call:

```
lm(formula = writelep ~ wavnolep + lep, data = LEP)
```

Residuals:

	Min	1Q	Median	3Q	Max
	-0.182858	-0.038116	-0.007095	0.024673	0.236180

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	1.0247614	0.0446417	22.955	<2e-16 ***
wavnolep	0.6824759	0.0137986	49.460	<2e-16 ***
lep	-0.0006231	0.0007169	-0.869	0.386

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.0497 on 137 degrees of freedom  
Multiple R-squared: 0.947, Adjusted R-squared: 0.9463  
F-statistic: 1225 on 2 and 137 DF, p-value: < 2.2e-16