Recoding

Paul E. Johnson[1] [2]

[1]Department of Political Science

[2]Center for Research Methods and Data Analysis, University of Kansas

2018

Outline

What's the big idea?

"They" give us variables that need to be *cleaned up*

- Recoding:
  - change variable names
  - alter numeric values
  - assign labels to values
- Difficult record-keeping process, requires teamwork

Outline

numbers, strings, and factors (oh my!)

- Today focus on R (R Core Team, 2017) variable classes
  1. integers
  2. numeric floating point numbers (AKA "doubles"),
  3. character strings, and
  4. factors
- Not discussing "Date" or "POSIXct" for date/time information

Be Careful, Check your work

- Always check effect of recodes.
  - Don't erase old variables & values
  - Do create new variables or new data frame
- One way is to rename every variable
- Sometimes I'll do the following
  1. Make a backup copy

     ```
     dat.orig <- dat
     ```

  2. Do recodes in dat
  3. Can compare variable across data.frames

     ```
     table(dat$x123, dat.orig$x123)
     ```

     Numeric variables: Scatterplot

     ```
     plot.default(dat$x456, dat.orig$x456)
     ```

     compare side by side

     ```
     cbind(dat$x456, dat.orig$x456)
     ```

Variable modes in R

- A **vector** is a collection of scores, all of which are stored in the same storage "mode".
- Mode examples:
  - integer
  - numeric (floating point number, AKA "double" precision floating number)
  - character (letters, non-number characters, or numbers that are quoted ("3"))
  - logical (legal values TRUE and FALSE represent 1 and 0)
- Most common methods to create vectors are
  - The `c()` function (c = concatenate) will guess the storage mode from your input
  - The `vector()` function will explicitly ask for a vector using a storage mode.
- Character vector

```
x <- c("alpha", "beta", "gamma", "omega", "psi")
is.character(x)
```

Variable modes in R ...

```
[1] TRUE
```

- If you create a vector with one character-value and some numbers, guess what happens?

```
x <- c("alpha", 2, 3, 3, 4)
```

R converts (demotes?) the numbers to characters.

```
x
```

```
[1] "alpha" "2"     "3"     "3"     "4"
```

Why? All elements in a vector must be of the same type.

Ways to check:

```
is.character(x)
```

```
[1] TRUE
```

Variable modes in R ...

```
mode(x)
```

```
[1] "character"
```

- Conversion back to numeric will replace the character with the missing value symbol (because there is no number for the character “alpha”)

```
as.numeric(x)
```

```
[1] NA  2  3  3  4
```

- If you enter data that appears to be integers, R guesses you wanted floating point numbers (double-precision real-valued numbers)

```
x <- c(55, 2, 3, 3, 4)
x
```

```
[1] 55  2  3  3  4
```

Variable modes in R ...

```
is.double(x)
```

```
[1] TRUE
```

```
is.integer(x)
```

```
[1] FALSE
```

Variable modes in R ...

- But if you *really do want integer data*, you can signal R about that by the letter “L” (short for “long integer” storage format) with your integers:

```
x <- c(55L, 2L, 3L, 3L, 4L)
x
```

```
[1] 55  2  3  3  4
```

```
is.integer(x)
```

```
[1] TRUE
```

KU

Variable modes in R ...

- What if you combine integers and floating point numbers?

```
x <- c(2L, 3L, 3L, 4L, 5.5323)
is.integer(x)
```

```
[1] FALSE
```

```
x
```

```
[1] 2.0000 3.0000 3.0000 4.0000 5.5323
```

```
is.double(x)
```

```
[1] TRUE
```

R has "promoted" the integers to floating point numbers in order to store them along with the floating value.

Variable modes in R ...

- Does not help if you explicitly create a vector by declaring its storage mode:

```
x <- vector(mode = "integer", length = 5)
is.integer(x)
```

```
[1] TRUE
```

```
x <- c(1, 2, 3, 4, 5)
is.integer(x)
```

```
[1] FALSE
```

```
## R is hiding the decimals from you
x
```

```
[1] 1 2 3 4 5
```

```
mode(x)
```

```
[1] "numeric"
```

Recoding Examples

- Replace 999 with NA (in an age variable, perhaps)
- Create new columns, such as “xlog” or “xsquared”.
- Change “Male” to “M” in a string variable
- Correct the misspelling of “Cincinnati”
- Re-group observations, to combine “aged” “elderly” “old” and “senior” in a character variable or a factor.
- Numeric and string recodes are comparatively easy
  - “atomic” data types (they have no R “attributes”)
- Factor variables require more effort, more internal components have to be fixed properly.

Outline

Recoding Numeric Variables is easy

We are interested in 3 particular problems

1. Setting some values as missings
2. Re-scaling and transforming values
3. Re-grouping values

Small test data frame

```
dat <- readRDS(file = "data/smtest.rds")
str(dat)
```

```
'data.frame': 48 obs. of  4 variables:
 $ x: num  235.7 35.3 52 415.7 412.2 ...
 $ y: int  11 14 15 10 17 10 15 14 10 8 ...
 $ w: chr  "g" "f" "i" "h" ...
 $ z: Factor w/ 4 levels "eenie","meanie",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
dat.orig <- dat ## spare copy
```

Two Styles for Setting missing values

Suppose every score for y greater than 11 is bogus, must be reset as missing

1. The index approach

```
dat$y[dat$y > 11] <- NA
```

   Find all values for which $y > 11$ and change them to symbol NA

2. The ifelse function

```
dat$y <- ifelse(dat$y > 11, NA, dat$y)
```

   If the value of y exceeds 11, return NA, but return y otherwise

- Either way, all values above 11 become missing.
- To understand the detail here, I suggest you look at the TRUE-FALSE vector $dat\$y > 11$. I'll show first 10 values:

```
head(dat$y > 11, 10)
```

```
 [1] FALSE  TRUE  TRUE FALSE  TRUE FALSE  TRUE  TRUE FALSE FALSE
```

Always double-check recodes

Always Double-check

I've got the original data set, so I can compare easily

```
table(is.na(dat$y), dat.orig$y, exclude = NULL)
```

```
        6 8 9 10 11 12 13 14 15 16 17 18 <NA>
  FALSE 2 4 5  4  7  0  0  0  0  0  0  0    0
  TRUE  0 0 0  0  0  1  6  5  5  4  3  2    0
  <NA>  0 0 0  0  0  0  0  0  0  0  0  0    0
```

"exclude = NULL" means "show me everything, missings and all"

Otherwise, create new variables with new names.

Always double-check recodes ...

This creates a new variable $y2$

```
dat$y2 <- dat$y
dat$y2[dat$y2 > 11] <- NA
## I run this to save space in output
table(is.na(dat$y2), dat$y, exclude = NULL)
```

```
        6 8 9 10 11 12 13 14 15 16 17 18
 FALSE 2 4 5  4  7  0  0  0  0  0  0  0
 TRUE  0 0 0  0  0  1  6  5  5  4  3  2
```

```
## Could as well do
## table(dat$y2, dat$y, exclude = NULL)
```

"exclude = NULL" means "show me everything, missings and all"

Any Logical Vector can be applied

- I don't know why this might happen, but suppose a co-author reports that all odd numbers between 11 and 19 are invalid
  - Did I show you the `seq()` function yet?

```
seq(11, 19, by = 2)
```

```
[1] 11 13 15 17 19
```

  - Did I show you about `%in%` yet?

```
c(10, 11, 12, 13, 14, 15, 16) %in% seq(11, 19, by = 2)
```

```
[1] FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE
```

  - Put those together

```
dat$y[dat$y %in% seq(11, 19, by = 2)] <- NA
```

Any y in the sequence $11, 13, \ldots, 19$ are set to NA

Rescaling and transforming

- R has math functions like +, -, /, log(), sqrt(), exp(), and so forth.
- Run "help("+")", or "?log", etc.
- It is as simple as column in, column out

```
dat$x2 <- 0.01 * dat$x
dat$xexp <- exp(dat$x)
dat$xlog <- log(dat$x)
dat$xsqrt <- sqrt(dat$x)
```

- Note
  1. I created new variables, but you are allowed to destroy/replace `x` itself if you want to
  2. I **strongly prefer** to name new variables by appending a suffix(" `xlog` " yes!, " `logx` " no!)

Alternative declaration approach

- I often use this method instead:

```
dat[ , "xlog.2"] <- log(dat$x)
```

- Because
  1. I can use a calculated value in the newly named column

     ```
     newname <- paste0("xlog", ".3")
     dat[ , newname] <- exp(dat$x)
     ```

  2. It works with matrices (with which $ does not)

Use cut to create categorical ranges

- This converts a numeric variable into a factor variable
- Divide a numeric range into groupings, use the `cut` function

```
dat$xcut <-cut(dat$x,
     breaks = c(-5, 60, 100, 1000000),
     labels = c("Minimal", "Medium", "Huge"))
table(dat$xcut, exclude = NULL)
```

```
Minimal  Medium    Huge
      6       7      35
```

Conditional Recodes are Easy As Well

- The function `ifelse()` is convenient
- arguments are a "logical condition", and a value if the condition is true, and one if it is false.

```
y <- c(31, 33, 41, 61)
ifelse(x < 3, y, x)
```

```
[1] 31 33  3  4  5
```

The only dangers are ...

1. You don't understand the function you apply:

```
z <- c(-2, -0.4, 0, 1, 2, 3)
log(z)
```

```
[1]       NaN       NaN      -Inf 0.0000000 0.6931472 1.0986123
```

2. Floating point numbers deserve caution: digitial computers are vulnerable to "rounding error".
   - Comparison of a numeric variable against a particular numeric value is hazardous/fatal.
   - Math: 2/3 is not exactly equal to 0.666667, but it may look like it.

```
2/3
```

```
[1] 0.6666667
```

```
print(2/3, digits = 20)
```

```
[1] 0.66666666666666662966
```

The only dangers are . . . ...

- The R-FAQ has a section explicitly devoted to this question: "7.31 Why doesn't R think these numbers are equal?" Conclusion: This is not an R problem, it is a digital computing problem.
- Because computers have finite, discrete storage, it is technically impossible to represent the continuum of the real number line in a computer variable

Example: Ambiguous Subtraction

Until 2015, I thought >=and <= were safe from trouble, but here's a counter-example

```
a <- 0.58; b <- 0.08
```

```
(a-b) >= 0.5
```

```
[1] FALSE
```

- WTF?
- First look here:

```
a
```

```
[1] 0.58
```

```
b
```

```
[1] 0.08
```

Example: Ambiguous Subtraction ...

```
a-b
```

```
[1] 0.5
```

- Turn up the precision of the display, the problem is easy-enough to see. First, I'll fiddle my environment

```
op.orig <- options()
options(digits=20)
```

```
a
```

```
[1] 0.57999999999999996003
```

```
b
```

```
[1] 0.08000000000000001665
```

```
a-b
```

Example: Ambiguous Subtraction ...

```
[1] 0.49999999999999994449
```

Then I put the environment back the way it was

```
options(op.orig)
```

- From this, I am humbled

A real life example

This is from a project in Spring, 2017.

```
a <- 100*(23/40)
b <- (100*23)/40
all.equal(a, b)
```

```
[1] TRUE
```

But...

```
round(a)
```

```
[1] 57
```

```
round(b)
```

```
[1] 58
```

Need some hints?

```
print(a, digits = 20)
```

A real life example ...

```
[1] 57.499999999999992895
```

```
print(b, digits = 20)
```

```
[1] 57.5
```

100/40 has an exact representation in floating point numbers in base 10, 2.5

```
print((100/40)*23, digits = 20)
```

```
[1] 57.5
```

It is reasonable to expect 100*23/40 = 2.5 * 23 should be exactly 57.5

23/40 is a non-repeating decimal in base 10

```
print(23/40, digits = 20)
```

```
[1] 0.57499999999999995559
```

Yet another example and the take-away

- Surprise: Even a simple decimal like 0.1 has no exact representation in digital numbers, it must be approximated by a nearby value

```
0.1
```

```
[1] 0.1
```

```
print(0.1, digits = 22)
```

```
[1] 0.1000000000000000055511
```

```
x <- 0.1
x == 0.1
```

```
[1] TRUE
```

```
x == 0.1000000000000000055511
```

```
[1] TRUE
```

Yet another example and the take-away ...

- Computers try to tell us what they think we want.
  - Should "==" succeed in both of those cases? I think NO!
- Major takeaway message: The use of "==" with numeric, non integer variables is very problematic
- The use of inequalities ">=" or "<=" should be cautious, possibly requiring re-design of an algorithm to allow some "numerical wobble"

Outline

Character variables

Why bother with character variables?

- The names of the columns need tidying.
- Character columns have errors
- We need beautiful labels. Convert country names, abbreviate states, etc.

Creating Character Variables: paste and paste0

The most frequently used string functions

- Manufacture a vector of integers, convert them to characters

```
dat$rn <- as.character(1:NROW(dat))
```

- The `paste()` function combines vectors, using an indicated separator

```
dat$wn <- paste(dat$w, dat$rn, sep = "_")
head(dat[ ,c("w", "rn", "wn")])
```

```
  w rn  wn
1 g  1 g_1
2 f  2 f_2
3 i  3 i_3
4 h  4 h_4
5 g  5 g_5
6 i  6 i_6
```

  If you forget to specify sep, then a space will be inserted.

- A convenience function `paste0()`, was created to save us the trouble of typing `sep = ""` when we don't want a separator of any kind.

Creating Character Variables: paste and paste0 ...

```
dat$wn <- paste0(dat$w, dat$rn)
head(dat[ ,c("w", "rn", "wn")])
```

```
  w rn  wn
1 g  1  g1
2 f  2  f2
3 i  3  i3
4 h  4  h4
5 g  5  g5
6 i  6  i6
```

I often need to clean up colnames

- Retrieve column names

```
oldnames <- colnames(dat)
oldnames
```

```
 [1] "x"       "y"       "w"       "z"       "y2"      "x2"      "xexp"
      "xlog"    "xsqrt"   "xlog.2"
[11] "xlog.3" "xcut"    "rn"      "wn"
```

- Sometimes, the needed change is simple, like changing to all CAPITAL letters

```
colnames(dat) <- toupper(oldnames)
head(dat, 2)
```

```
          X  Y W     Z Y2        X2          XEXP     XLOG
1 235.69505 11 g eenie 11 2.3569505 2.296468e+102 5.462539
2  35.26574 14 f eenie NA 0.3526574  2.068785e+15 3.562912
      XSQRT   XLOG.2        XLOG.3    XCUT RN WN
1 15.352363 5.462539 2.296468e+102    Huge  1 g1
2  5.938496 3.562912  2.068785e+15 Minimal  2 f2
```

I often need to clean up colnames ...

- Sometimes they need more information, before combining with other data sets

```
colnames(dat) <- paste0(oldnames, "_", 2017)
head(dat, 2)
```

```
     x_2017 y_2017 w_2017 z_2017 y2_2017    x2_2017
1 235.69505     11      g  eenie      11 2.3569505
2  35.26574     14      f  eenie      NA 0.3526574
      xexp_2017 xlog_2017 xsqrt_2017 xlog.2_2017
1 2.296468e+102  5.462539  15.352363    5.462539
2  2.068785e+15  3.562912   5.938496    3.562912
    xlog.3_2017 xcut_2017 rn_2017 wn_2017
1 2.296468e+102      Huge       1      g1
2  2.068785e+15   Minimal       2      f2
```

- I'd better replace the original names now

```
colnames(dat) <- oldnames
```

Renaming columns can be a tricky business

- In one recent project, all of the column names have an accidentally repeated word, as in {"religion_religion_1", "religion_religion_2", "gender_gender_1", ...}. To fix that, we extracted the column names and applied some fancy "regular expression" code.
- Project had 9 data input files from different prison hospitals. In each one, there were hundreds of columns that represented the SAME information with different variable names. The goal was to reduce the unique names to standard names like `dxcode1, dxcode2, dxcode3`.
- Client had an employee make a "name old" and "name new" roster for each file in an Excel sheet. We import those values, apply *without retyping them* (avoid typos).

Brief list of string functions

1. paste . Mentioned above. Combines vectors, often used for creating row names for data frames.
2. substr(x, start, stop). Chops a character vector x, keeping only the characters from the positions between start and stop

```
x <- c("hello", "jello", "fellow", "mellow")
substr(x, 2, 5)
```

```
[1] "ello" "ello" "ello" "ello"
```

3. strsplit(x, splt) . Creates a new list, in which the elements of x are chopped into pieces separated by the symbol splt .

```
strsplit(x, "ll")
```

Brief list of string functions ...

```
[[1]]
[1] "he" "o"

[[2]]
[1] "je" "o"

[[3]]
[1] "fe" "ow"

[[4]]
[1] "me" "ow"
```

CAUTION: Regular Expressions are discussed in more depth in the 4th day of this workshop. REs are a specialized language that offers powerful filtering tools

4. grep(pattern, x): "GNU regular expression parser", scans for presence of "pattern" in string "x",
   - the argument `fixed = TRUE` turns off regex support, treats the pattern and like ordinary letters.
   - Check for "ee" in dat$z. Note return is index number of matches

Brief list of string functions ...

```
ees <- grep("ee", dat$z, fixed = TRUE)
head(ees)
```

```
[1] 1 2 3 4 5 6
```

- Want a vector of matching values?

```
ees <- grep("ee", dat$z, value = TRUE, fixed = TRUE)
head(ees)
```

```
[1] "eenie" "eenie" "eenie" "eenie" "eenie" "eenie"
```

5. "regex" pattern matching (a staple of formal computer science training)
   - `^` and `$`: 2 key symbols in RE:
     - The "^" symbol stands for "the beginning of the string"
     - The "$" symbol stands for the end of the string
   - Example: which values of dat$wn that begin with g

```
startswd <- grep("^g", dat$wn, value = TRUE)
head(startswd, 10)
```

Brief list of string functions ...

```
[1] "g1"  "g5"  "g10" "g14" "g17" "g19" "g22" "g24" "g29"
    "g31"
```

- The grepl function returns logical TRUE/FALSE instead

```
startswdl <- grepl("^g", dat$wn)
head(startswdl, 10)
```

```
[1]  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
    TRUE
```

6. gsub(x, y, var) : "Substitute all occurrences of x with the characters y in var".
   Allows regex, but fixed = TRUE can disable regex support.

```
dat$wn2 <- gsub("f", "PJ", dat$wn, fixed = TRUE)
head(dat[ , c("wn", "wn2")], 7)
```

Brief list of string functions ...

```
  wn  wn2
1 g1   g1
2 f2  PJ2
3 i3   i3
4 h4   h4
5 g5   g5
6 i6   i6
7 h7   h7
```

- We have a project that imports name data.
  - Character strings for names should not include any characters except letters, numbers, `_` `'` `( )` and `-`. Remove all other characters.

```
y <- gsub("[^a-zA-Z0-9\\\\'_()-]", "", x)
```

  - Regular expression
    - "[]" in regular expression means "any of the following"
    - ^ negation in first position, converts that to "anything but the following"
    - dash between 2 letters or numbers is interpreted as a range
    - dash before] literally means dash
    - four backslashes before '. Will explain Thursday.

Outline

Factor Variables = Categorical Variables

- A factor is a categorical variable, discrete values like {Catholic, Protestant} or {left, right, middle}
- A factor that is subjectively ordered is called an "ordered factor", or simply an "ordered" variable.
- Factors have attributes (the "levels").
- Functions notice that and customize calculations and beautify displays

Factor Variables = Categorical Variables

- Inside R, a factor has an integer index, always beginning with 1
- From the user's point of view, the factor's values usually appear as text strings.

```
table(dat$z)
```

```
eenie  meanie   miney      mo
   12      12      12      12
```

- Inside R, there's a "lookup table", so the value is really stored as 1, 2, 3, 4, but often, when you interact with it, it behaves like a character string variable.

Internal Integer	Level
1	"eenie"
2	"meanie"
3	"miney"
4	"mo"

Creating Factors

- We've already seen that the `cut()` function can create a factor variable
- Functions that create factors
    - `factor()` : Unordered categories
    - `ordered()` : ordinal variables, subjectively ranked levels
- `factor()` has 3 key arguments,
    - a variable: something with values that need to be converted to a factor
    - levels: values to be used, in order we want them to appear
    - labels: character strings. If omitted, R runs `as.character(levels)` to manufacture labels.

Creating Factors ...

factor() example: Convert a character string to a factor

R guesses levels and labels. It guesses levels *in alphabetical order*

- See what you got

```
table(dat$w)
```

```
f  g  h  i  j
8 14  8 13  5
```

- w2 is the default factor

```
dat$w2 <- factor(dat$w)
```

- w2 looks like w, superficially

```
table(dat$w2, dat$w)
```

Creating Factors ...

```
    f  g  h  i  j
f   8  0  0  0  0
g   0 14  0  0  0
h   0  0  8  0  0
i   0  0  0 13  0
j   0  0  0  0  5
```

- but the internal structures are different:

```
str(dat$w)
```

```
chr [1:48] "g" "f" "i" "h" "g" "i" "h" "f" "j" "g" "f" "h" "h"
   "g" "j" "f" "g" "j" "g" "f" "i" ...
```

```
str(dat$w2)
```

```
Factor w/ 5 levels "f","g","h","i",..: 2 1 4 3 2 4 3 1 5 2 ...
```

Internal integers are always 1, 2, 3, . . .

- SPSS and Stata allow any integer values with labels
- R will "throw away" the integers

```
myintegers <- c(1, 3, 5, 7, 9, 9, 7, 5, 3, 1)
myfactor <- factor(myintegers, levels = c(1, 3, 5, 7, 9),
                   labels = c("E", "D", "C", "B", "A"))
```

- Note the numbers for the levels

```
str(myfactor)
```

```
Factor w/ 5 levels "E","D","C","B",..: 1 2 3 4 5 5 4 3 2 1
```

Internal Integer	Level
1	"E"
2	"D"
3	"C"
4	"B"
5	"A"

Internal integers are always 1, 2, 3, ...

- The original numbers cannot be recovered

```
as.integer(myfactor)
```

```
[1] 1 2 3 4 5 5 4 3 2 1
```

Internal integers are always 1, 2, 3, ...

Can Specify particular values

```
myintegers <- c(1, 3, 5, 7, 9, 9, 7, 5, 3, 1)
yourfactor <- factor(myintegers, levels = c(1, 9, 7),
 labels = c("cold", "warm", "hot"))
```

But you still lose the original integer values

```
str(yourfactor)
```

```
Factor w/ 3 levels "cold","warm",..: 1 NA NA 3 2 2 3 NA NA 1
```

Observe:

```
table(as.integer(myfactor), as.integer(yourfactor), exclude = NULL)
```

Internal integers are always 1, 2, 3, ...

```
    1 2 3 <NA>
  1 2 0 0    0
  2 0 0 0    2
  3 0 0 0    2
  4 0 0 2    0
  5 0 2 0    0
```

A Related Problem: Accidental factor-ization of an integer

factor() example: Convert an integer variable to a labeled factor

```
x <- c(1, 2, 1, 2, 2, 7)
xf <- factor(x, levels = c(7, 2, 1), labels = c("seven","two", "one"))
levels(xf)
```

```
[1] "seven" "two"   "one"
```

levels= tells R "in which order should the values of x come in?" I jumbled the order to make a point.

original value	R assigned internal value	label for the level
integer		
7	1	"seven"
2	2	"two"
1	3	"one"

- Suppose we forget the labels argument in the factor function.

A Related Problem: Accidental factor-ization of an integer ...

```
xf2 <- factor(x, levels = c(7, 2, 1))
levels(xf2)
```

```
[1] "7" "2" "1"
```

All appears well, the labels are just the numbers, but with quotation marks.

- Now the part that has caused plenty of confusion:
  Internal numbers are 1, 2, 3, but the named labels are "7", "2", "1".

original value	R assigned internal value	label for the level
7	1	"7"
2	2	"2"
1	3	"1"

- Do you want the face-value labels to turn back into the 7-2-1 scores you started with? (*Not so fast, my friend!*)

```
as.numeric(xf2)
```

A Related Problem: Accidental factor-ization of an integer …

```
[1]  3  2  3  2  2  1
```

- In the help page "?factor", they recommend this method to convert a "levels are numbers" factor back to the numbers:

```
xnew <- as.numeric(levels(xf2))[xf2]
xnew
```

```
[1]  1  2  1  2  2  7
```

```
table(xnew, x)
```

```
    x
xnew  1  2  7
   1  2  0  0
   2  0  3  0
   7  0  0  1
```

- The more obvious method is

A Related Problem: Accidental factor-ization of an integer ...

```
xnew2 <- as.numeric(as.character(xf2))
xnew2
```

```
[1] 1 2 1 2 2 7
```

```
table(xnew2, x)
```

```
      x
xnew2 1 2 7
    1 2 0 0
    2 0 3 0
    7 0 0 1
```

- But it is not recommended in ?factor

Benefits of Using Factors

- Using factors reduces human errors associated with integer scores. "Is '1' a male or female?"
- Procedures notice the levels. Regression in R will notice and create 'dummy variables' ("contrast" variables).

```
m1 <- lm(x ~ z + w2, data = dat)
summary(m1)
```

```
Call:
lm(formula = x ~ z + w2, data = dat)

Residuals:
     Min       1Q   Median       3Q      Max
-132.500  -82.180    9.085   50.083  227.877

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  114.782     42.010   2.732  0.00932 **
zmeanie        8.354     42.119   0.198  0.84378
zminey       -31.003     41.900  -0.740  0.46366
zmo           20.530     46.373   0.443  0.66036
w2g           69.495     45.117   1.540  0.13136
w2h          147.206     52.430   2.808  0.00768 **
```

Benefits of Using Factors ...

```
w2i              8.909      49.897    0.179   0.85919
w2j            109.503      58.473    1.873   0.06843 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 100.8 on 40 degrees of freedom
Multiple R-squared:  0.2233,  Adjusted R-squared:  0.08734
F-statistic: 1.643 on 7 and 40 DF,  p-value: 0.1516
```

- If you give the factor variable to a plotting function, that function should notice it is not a numeric variable and act accordingly.

```
plot(xf, main = "plot noticed its a factor!", xlab = "xf: a
     factor", ylab = "Count")
text(0.2, 2, "R noticed xf \n is a factor. \n So R ran table \n
     and sent output \n to barplot", pos=4)
```

Benefits of Using Factors ...

Examples of Needed Recodes

- Combine erroneously coded labels, reduce labels {Male, Man, Female, Woman} to {M, F} or something like that.
- Take verbose labels and make them shorter. Convert {Strongly Disagree, Disagree, ...} to {SD, D, N, A, SA}
- R's built in tools work well, but have a number of technical details that will frustrate new uses.
- In the rockchalk package, I created a function "combineLevels()" that can work on a problem like that, and it does some error checking to make sure it works correctly.
- In the `plyr` package, there is a very elegant function (`mapvalues()`) that can be used and it makes this kind of chore rather painless.

Working with factors

1. Use levels() to read and set levels.
   - Check existing levels. Use a copy so we don't mangle the original

```
dat$w5 <- dat$w2
levels(dat$w5)
```

```
[1] "f" "g" "h" "i" "j"
```

   - Replace by assigning a new vector of same length.

```
levels(dat$w5) <- c("John", "Paul", "George", "Ringo", "Eric")
table(dat$w5, dat$w2)
```

```
         f  g  h  i  j
 John    8  0  0  0  0
 Paul    0 14  0  0  0
 George  0  0  8  0  0
 Ringo   0  0  0 13  0
 Eric    0  0  0  0  5
```

   - Vital: Must provide names for all levels

Working with factors ...

- Use the factor function and assign levels and labels (here, levels means "current values")

```
dat$w6 <- factor(dat$w2, levels = c("f", "g", "h", "i", "j"),
     labels = c("John", "Paul", "George", "Ringo", "Eric"))
table(dat$w6, dat$w2, exclude = NULL)
```

```
         f  g  h  i  j
 John    8  0  0  0  0
 Paul    0 14  0  0  0
 George  0  0  8  0  0
 Ringo   0  0  0 13  0
 Eric    0  0  0  0  5
```

2. rockchalk::combineLevels has some handy sanity-preserving features :)

```
library(rockchalk)
dat$w5 <- combineLevels(dat$w2, levs = c("f", "j"), newLabel =
     c("fandj"))
```

```
The original levels f g h i j
have been replaced by g h i fandj
```

Working with factors ...

```
table(dat$w5, dat$w2)
```

```
        f  g  h  i  j
g       0 14  0  0  0
h       0  0  8  0  0
i       0  0  0 13  0
fandj   8  0  0  0  5
```

If `dat$w5` is an ordinal variable, `combineLevels` will refuse to "put together" levels that are not adjacent with one another

3. A good general purpose discrete variable recoder is `plyr::mapvalues`.
    1. `mapvalues(x, value_old, value_new)`

       `value_old` and `value_new` must be vectors with the same numbers of elements
    2. works well if x is an integer, character, or factor variable
    3. example

Working with factors ...

```
library(plyr)
dat$w5 <- mapvalues(dat$w2, from = c("f", "g", "h", "i",
     "j"), to = c("John", "Paul", "George", "Ringo", "Erik"))
str(dat$w5)
```

```
Factor w/ 5 levels "John","Paul",..: 2 1 4 3 2 4 3 1 5 2 ...
```

4. To reduce human error (avoid mis-matched elements), I usually create a named vector:

```
newvals <- c("f" = "John", "g" = "Paul", "h" = "George", "i" =
     "Ringo", "j" = "Erik")
```

That puts the paired (old=new) items together, and we can extract the names like so

```
names(newvals)
```

```
[1] "f" "g" "h" "i" "j"
```

or put them to use as:

Working with factors ...

```
dat$w6 <- mapvalues(dat$w2, from = names(newvals), to = newvals)
table(dat$w6, dat$w5)
```

```
        John Paul George Ringo Erik
  John     8    0      0     0    0
  Paul     0   14      0     0    0
  George   0    0      8     0    0
  Ringo    0    0      0    13    0
  Erik     0    0      0     0    5
```

Can use this to reset just a few levels:

```
dat$w8 <- mapvalues(dat$w2, from = c("f", "g"), to =
     c("Roosevelt", "Lincoln"))
table(dat$w8)
```

```
Roosevelt   Lincoln         h         i         j
        8        14         8        13         5
```

To combine some levels and re-label all of them as “fgh”:

Working with factors ...

```
dat$w9 <- mapvalues(dat$w2, from = c("f", "g", "h"),
           to = c("fgh", "fgh","fgh"))
table(dat$w9)
```

```
fgh    i    j
 30   13    5
```

Convert factor values to missings

- Can reset values "i" and "j" as NA either by "indexing"

```
dat$w7 <- dat$w2
dat$w7[dat$w7 %in% c("i","j")] <- NA
table(dat$w7, exclude = NULL)
```

```
   f     g     h     i     j  <NA>
   8    14     8     0     0    18
```

- Or explicitly relabeling the vector

```
dat$w8 <- dat$w2
levels(dat$w8) <- c("f", "g", "h", NA, NA)
table(dat$w8)
```

```
f  g  h
8 14  8
```

```
table(dat$w8, exclude = NULL)
```

```
   f     g     h  <NA>
   8    14     8    18
```

Convert factor values to missings ...

- Tip: When factor values have long, hard-to-type names,
  - copy the levels vector
  - use that with indices, as in

```
(xl <- levels(dat$w2))
```

```
[1] "f" "g" "h" "i" "j"
```

```
xl[4:5]
```

```
[1] "i" "j"
```

```
dat$w10 <- dat$w2
dat$w10[dat$w10 %in% xl[4:5]] <- NA
```

The Unused Levels problem

- Suppose there are possible levels ("a", "b", ..., "g")
- However, for some reason, in a sample, we only collect data on ("a", "b", "c").
- If the factor is created on the range of possible scores, then there will be many "unused levels"

```
x <- c(1, 2, 3, 3, 2, 1)
xf <- factor(x, levels = 1:7, labels = letters[1:7])
str(xf)
```

```
Factor w/ 7 levels "a","b","c","d",..: 1 2 3 3 2 1
```

- The table function displays the empty "unused levels" (by default):

```
table(xf)
```

```
xf
a b c d e f g
2 2 2 0 0 0 0
```

- This creates ugly reports, we might want to get rid of those "unused levels" entirely.

2 workable ways to purge unused levels

1. factor() removes unused levels

```
y <- factor(y)
```

2. R Documentation suggests this is more meaningful (?)

```
y <- y[ , drop = FALSE]
```

Outline

After Recoding, safe a reloadable set

- Suppose there is a data frame named myOldDat

```
wdir <- "workingdata"
# That's our preferred name for created data
saveRDS(myOldDat, file = file.path(wdir, "myWonderful.rds"))
```

- To bring that back into a session

```
wdir <- "workingdata"
awesomeDat <- readRDS(file = file.path(wdir, "myWonderful.rds"))
```

- Allows us to rename the data frame object when it is retrieved
- RDS files are portable. Can email to your friend who has a Mac

Outline

Get my subset from Nat. Election Study 2004

- The data is in the “data” folder, “04245-0001-Data.dta”.
- Otherwise, download: http://pj.freefaculty.org/guides/Rcourse/DataSets/04245-0001-Data.dta.
- Assuming the file “04245-0001-Data.dta” ended up in data, then import.

```
library(foreign)
ddir <- "data"
fp <- file.path(ddir, "04245-0001-Data.dta")
anes1 <- read.dta(fp)
anes1.orig <- anes1
```

- Save copies of the wide and long variable keys, just for comparison

```
library(kutils)
keywide <- kutils::keyTemplate(anes1, file = "anes-widetemp.csv")
```

That creates “keywide” object and immediate file snapshot

- Can create & inspect first, use string functions to recode, then keySave

Get my subset from Nat. Election Study 2004 ...

```
keylong <- kutils::keyTemplate(anes1, long = TRUE)
keySave(keylong, file = "anes-longtemp.csv")
```

- The `keyTemplateStata` (and `keyTamplateSPSS`) functions are recently introduced. These have "value_old" as the integers that were used in original data and "value_new" as the labels. This is very close to a "programmable codebook"
- Interactively, you can run `View(keystata)` :

```
keystata <- kutils::keyTemplateStata(fp, long = TRUE)
```

```
View(keystata)
```

The value labels are verbose, one of the regex chores would be cleaning them up.

- We'd go edit the key files, perhaps rename them, then run `keyImport`
- My edited key is "anes-wide.csv" in the current working directory.

A Codebook Lists the Variables

- Some things we can treat as numeric

```
## V043038      B1a. Feeling Thermometer: GW Bush
## V043039      B1b. Feeling Thermometer: John Kerry
## V043250      Y1x. Summary: Respondent age
```

- Some are clearly categorical

```
## V043210      R1. R position on gay marriage
## V043213      S3. National economy better/worse since GW Bush
    took ofc
## V045145X     H5x. Summary: Pre-Post US flag makes R feel
## V043116      J1x. Summary: R party ID
```

- Some are treated as numeric by some people

```
## V045117      G4a. Liberal/conservative 7-point scale:
    self-placement
## V043116      J1x. Summary: R party ID
```

This is the new way CRMDA has developed

- In the section after this one, we show the line-by-line recode commands needed
- This section uses the Variable Key
- This is a development enterprise in the CRMDA package `kutils`.

Key

- I edited "anes-wide.csv"
- Re-import that key file

```
key <- keyImport("anes-wide.csv")
```

```
keyImport guessed that is a wide format key.
```

- If you are interactive, run View

```
View(key)
```

- Screenshots

<table>
<tr><th></th><th>row.names</th><th>name_old</th><th>name_new</th><th>class_old</th><th>class_new</th><th>value_old</th><th>value_new</th><th>missings</th><th>recodes</th></tr>
<tr><td>1</td><td>V041109A.V041109A</td><td>V041109A</td><td>V041109A</td><td>factor</td><td>factor</td><td>1. Male|2. Female</td><td>M|F</td><td></td><td></td></tr>
<tr><td>2</td><td>V043038.V043038</td><td>V043038</td><td>V043038</td><td>integer</td><td>integer</td><td>.</td><td>.</td><td></td><td></td></tr>
<tr><td>3</td><td>V043039.V043039</td><td>V043039</td><td>V043039</td><td>integer</td><td>integer</td><td>.</td><td>.</td><td></td><td></td></tr>
<tr><td>4</td><td>V043048.V043048</td><td>V043048</td><td>V043048</td><td>integer</td><td>integer</td><td>.</td><td>.</td><td></td><td></td></tr>
<tr><td>5</td><td>V043116.V043116</td><td>V043116</td><td>V043116</td><td>factor</td><td>factor</td><td>0. Strong Democrat (2/1/.)|1. Weak Democrat (2/5-8-9/.)|2. Independent-Democrat (3-4-></td><td>SD|WD|ID|I|IR|WR|SR|.|.|.</td><td></td><td></td></tr>
<tr><td>6</td><td>V043210.V043210</td><td>V043210</td><td>V043210</td><td>factor</td><td>factor</td><td>3. Should not be allowed|5. Should not be allowed to marry but should be allowed|1. S></td><td>No|Some|Allow|.|.|.</td><td></td><td></td></tr>
<tr><td>7</td><td>V043213.V043213</td><td>V043213</td><td>V043213</td><td>factor</td><td>factor</td><td>1. Better|3. Worse|5. The same|8. Don't know|9. Refused</td><td>Better|Worse|Same|.|.</td><td></td><td></td></tr>
<tr><td>8</td><td>V043213.econnew</td><td>V043213</td><td>econnew</td><td>factor</td><td>factor</td><td>3. Worse|5. The same|1. Better|8. Don't know|9. Refused</td><td>Worse|Same|Better|.|.</td><td></td><td></td></tr>
<tr><td>9</td><td>V043250.aged</td><td>V043250</td><td>aged</td><td>integer</td><td>factor</td><td>.</td><td>.</td><td></td><td>cut(x, breaks = c(-1, 57, 200), labels = c("young", "old"))</td></tr>
<tr><td>10</td><td>V043250.V043250</td><td>V043250</td><td>V043250</td><td>integer</td><td>integer</td><td>.</td><td>.</td><td></td><td></td></tr>
<tr><td>11</td><td>V045117.V045117</td><td>V045117</td><td>V045117</td><td>factor</td><td>factor</td><td>01. Extremely liberal|02. Liberal|03. Slightly liberal|04. Moderate;middle of the roa></td><td>EL|L|SL|M|SC|C|EC|.|.|.</td><td></td><td></td></tr>
<tr><td>12</td><td>V045145X.V045145X</td><td>V045145X</td><td>V045145X</td><td>factor</td><td>factor</td><td>1. Extremely good|2. Very good|3. Somewhat good|4. Not very good|7. Don't feel anythi></td><td>EG|VG|SG|NVG|DFA|.|.</td><td></td><td></td></tr>
<tr><td>13</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>14</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

Key ...

- Apply that wide key

```
anes2 <- keyApply(anes1, key, drop = "vars", diagnostic = TRUE)
```

```
        V041109A (old var)
V041109A 1. Male 2. Female
       M      566         0
       F        0       646
[1] "Variable V043038 has 20 unique values. Too large for a table."
[1] "Variable V043039 has 20 unique values. Too large for a table."
[1] "Variable V043048 has 20 unique values. Too large for a table."
       V043116 (old var)
V043116 0. Strong Democrat (2/1/.) 1. Weak Democrat (2/5-8-9/.) 2.
     Independent-Democrat (3-4-5/./5) 3. Independent-Independent 4.
     Independent-Republican (3-4-5/./1) 5. Weak Republican (1/5-8-9/.)
   SD                          203                              0
                                       0                          0
                                         0
                                  0
   WD                            0                            179
                                       0                          0
                                         0
                                  0
```

Key ...

```
ID                                           0                                           0
                                                      210                                         0
                                                            0
                                                  0
I                                            0                                           0
                                                         0                                      118
                                                            0
                                                  0
IR                                           0                                           0
                                                         0                                        0
                                                         138
                                                  0
WR                                           0                                           0
                                                         0                                        0
                                                            0
                                              154
SR                                           0                                           0
                                                         0                                        0
                                                            0
                                                  0
<NA>                                         0                                           0
                                                         0                                        0
                                                            0
                                                  0
      V043116 (old var)
```

Key ...

```
V043116 6. Strong Republican (1/1/.) 7. Other;minor party;refuses to
    say 8. Apolitical (5/./3-8-9 if apolitical) 9. DK (8/./.) <NA>
  SD                                 0
       0                                          0              0    0
  WD                                 0
       0                                          0              0    0
  ID                                 0
       0                                          0              0    0
  I                                  0
       0                                          0              0    0
  IR                                 0
       0                                          0              0    0
  WR                                 0
       0                                          0              0    0
  SR                               193
       0                                          0              0    0
  <NA>                               0
       5                                          0              0   12
      V043210 (old var)
V043210 1. Should be allowed 3. Should not be allowed 5. Should not be
    allowed to marry but should be allowed VOL 8. Don't know 9.
    Refused <NA>
  No                        0                      705
                                                                 0    0
                  0          0    0
```

Key ...

```
  Some                    0                         0
                                                           41    0
                0          0    0
  Allow                 400                         0
                                                            0    0
                0          0    0
  <NA>                    0                         0
                                                            0   30
                0          0   36
        V043213 (old var)
V043213  1. Better 3. Worse 5. The same 8. Don't know 9. Refused <NA>
  Better       190        0           0              0          0    0
  Worse          0      668           0              0          0    0
  Same           0        0         343              0          0    0
  <NA>           0        0           0              0          0   11
        V043213 (old var)
econnew  1. Better 3. Worse 5. The same 8. Don't know 9. Refused <NA>
  Worse          0      668           0              0          0    0
  Same           0        0         343              0          0    0
  Better       190        0           0              0          0    0
  <NA>           0        0           0              0          0   11
       V043250 (old var)
aged     18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38
    39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
    61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81
```

Key ...

```
  young  8 21 17 15 19 25 22 23 18 20 28 23 15 23 14 22 19 23 17 16 23
      25 19 26 19 21 26 25 27 23 24 23 25 25 25 22 19 24 28 26  0  0
       0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  old    0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
       0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 24 14 13
      20 22 27 21 21 16 14 11 10 10  9  9 10 10  8 10 16  8  6  4  5
       V043250 (old var)
aged    82 83 84 85 86 87 88 90
  young  0  0  0  0  0  0  0  0
  old   10  5  4  1  4  1  4  2
[1] "Variable V043250 has 20 unique values. Too large for a table."
       V045117 (old var)
V045117 01. Extremely liberal 02. Liberal 03. Slightly liberal 04.
    Moderate;middle of the road 05. Slightly conservative 06.
    Conservative 07. Extremely conservative
   EL                        20            0                      0
                                      0                              0
                       0                                0
   L                          0           103                     0
                                      0                              0
                       0                                0
   SL                         0            0                    125
                                      0                              0
                       0                                0
```

Key ...

```
   M                             0            0                      0
                                      279                              0
                            0                         0
   SC                            0            0                      0
                                        0                             143
                            0                         0
   C                             0            0                      0
                                        0                              0
                          166                         0
   EC                            0            0                      0
                                        0                              0
                            0                        31
   <NA>                          0            0                      0
                                        0                              0
                            0                         0
       V045117 (old var)
V045117 80. Haven't thought much {DO NOT PROBE} 88. Don't know 89.
    Refused <NA>
   EL                                             0                  0
                       0    0
   L                                              0                  0
                       0    0
   SL                                             0                  0
                       0    0
   M                                              0                  0
                       0    0
```

Key ...

```
   SC                                                  0                0
                  0     0
   C                                                   0                0
                  0     0
   EC                                                  0                0
                  0     0
   <NA>                                                0                0
                  0   345
        V045145X (old var)
V045145X 1. Extremely good 2. Very good 3. Somewhat good 4. Not very
    good 7. Don't feel anything {VOL} 8. Don't know 9. Refused <NA>
    EG                    570            0                 0
         0                              0              0           0    0
    VG                      0          338                 0
         0                              0              0           0    0
    SG                      0            0               175
         0                              0              0           0    0
    NVG                     0            0                 0
         38                              0              0           0    0
    DFA                     0            0                 0
         0                             18              0           0    0
    <NA>                    0            0                 0
         0                              0              0           0   73
```

Key ...

- If interactive, use `kutils::peek` to scan through the variables

```
peek(anes2)
```

- summarize

```
rockchalk::summarize(anes2)
```

```
Numeric variables
          V043038   V043039   V043048   V043250
min            0         0         0       18
med           60        60        60       47
max          100       100       100       90
mean          54.941    53.019    61.111   47.272
sd            33.547    26.361    19.223   17.142
skewness      -0.304    -0.407    -0.347    0.213
kurtosis      -1.172    -0.507     0.958   -0.809
nobs        1207      1191       952     1212
nmissing       5        21       260        0

Nonnumeric variables
              V041109A              V043116
 M: 566                ID         : 210
 F: 646                SD         : 203
```

Key ...

```
                          SR           : 193
                          WD           : 179
                          (All Others): 410
nobs          : 1212.000 nobs          : 1195.000
nmiss         :    0.000 nmiss         :   17.000
entropy       :    0.997 entropy       :    2.781
normedEntropy:     0.997 normedEntropy:     0.991
              V043210                      V043213
No    : 705              Better: 190
Some  : 41               Worse : 668
Allow: 400               Same  : 343


nobs          : 1146.000 nobs          : 1201.000
nmiss         :   66.000 nmiss         :   11.000
entropy       :    1.133 entropy       :    1.408
normedEntropy:     0.715 normedEntropy:     0.888
              econnew                        aged
Worse  : 668             young: 863
Same   : 343             old   : 349
Better: 190


nobs          : 1201.000 nobs          : 1212.000
nmiss         :   11.000 nmiss         :    0.000
entropy       :    1.408 entropy       :    0.866
```

Key ...

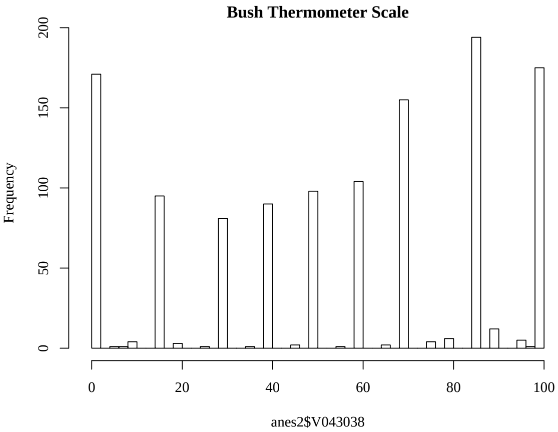
```
normedEntropy:    0.888 normedEntropy:    0.866
              V045117                   V045145X
M            : 279      EG : 570
C            : 166      VG : 338
SC           : 143      SG : 175
SL           : 125      NVG: 38
(All Others): 154       DFA: 18
nobs          : 867.000 nobs          : 1139.000
nmiss         : 345.000 nmiss         :   73.000
entropy       :   2.477 entropy       :    1.693
normedEntropy:    0.882 normedEntropy:     0.729
```

Create a new variable, manually

- The Variable Key can't do everything. It works one-variable-at-a-time (so far).
- Manually calculate the difference in feeling between Bush and Kerry
- Review the thermometer scales

```
hist(anes2$V043038, breaks = 50, xlim = c(-1, 101), main = "Bush
    Thermometer Scale")
```

Create a new variable, manually ...



- Create a new variable for the difference between Bush and Kerry feeling thermometers:

```
anes2$th.bk <- anes2$V043038 - anes2$V043039
```

"numeric" thermometer scores

```
table(anes1$V043038)
```

```
  0    5    7   10   15   20   25   30   35   40   45   49   50   55   60
171    1    1    4   95    3    1   81    1   90    2    1   97    1  104
 65   70   75   80   85   90   95   98  100
  2  155    4    6  194   12    5    1  175
```

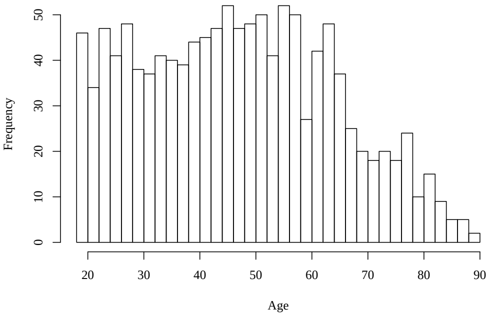
Where do we stand?

- The work's done. We have a recoded data frame "anes2".
- Manager & GRAs have common understanding of variable names and new values
- In the next section, we proceed through the recoding process, variable-by-variable, in the old-fashioned, time-honored tradition

Recode age into a dichotomy

- Inspect the age variable

```
hist(anes1$V043250, breaks = 40, main = "", xlab = "Age")
```



- Use cut to create a “dummy variable” for old people

Recode age into a dichotomy ...

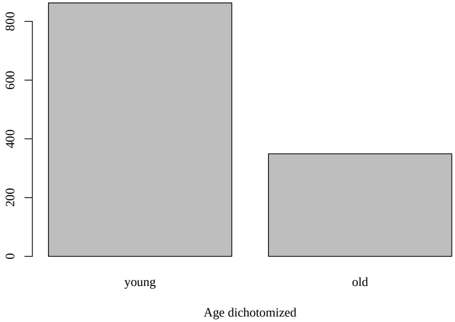
```
anes1$aged <- cut(anes1$V043250, breaks = c(-1, 57, 200), labels
     = c("young", "old"))
table(anes1$aged)
```

```
young   old
  863   349
```

57 is this year's definition of old, in case you wondered.

plot is a generic function, notices aged is not numeric

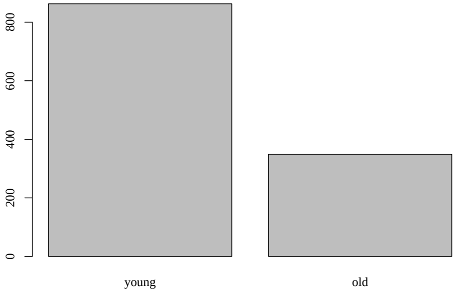
```
plot(anes1$aged, xlab = "Age dichotomized")
```



- Recall that the plot function notices the input type and it tries to make the plot you want. If we were being more systematic, we'd create the frequency table, then plot it.

plot is a generic function, notices aged is not numeric ...

```
t1 <- table(anes1$aged)
barplot(t1, beside = TRUE)
```

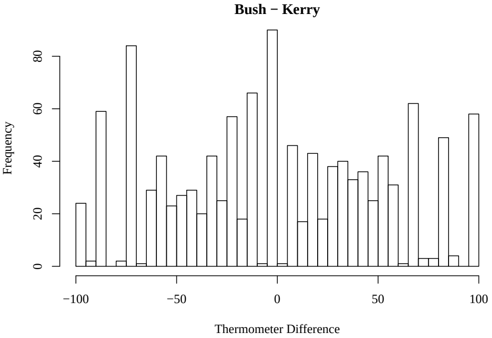


Create a dependent variable: Bush vs Kerry

- Create New Variable: The difference in thermometer between Bush and Kerry

```
anes1$th.bk <- anes1$V043038 - anes1$V043039
```

```
hist(anes1$th.bk, breaks = 40, main = "Bush - Kerry", xlab =
    "Thermometer Difference")
```



Clean up a bunch of variables and value labels

- Party Identification. The impossibly long level names create havoc!

```
##Party
table(anes1$V043116, exclude = NULL)
```

```
            0. Strong Democrat (2/1/.)              1. Weak Democrat
                (2/5-8-9/.)
                                    203

                                        179
    2. Independent-Democrat (3-4-5/./5)                3.
        Independent-Independent
                                    210

                                        118
  4. Independent-Republican (3-4-5/./1)          5. Weak Republican
      (1/5-8-9/.)
                                    138

                                        154
          6. Strong Republican (1/1/.)      7. Other;minor
                party;refuses to say
```

Clean up a bunch of variables and value labels ...

```
                                                 193

                                                       5
8. Apolitical (5/./3-8-9 if apolitical)                                               9. DK
     (8/./.)
10                                                  0

                                                          0
                                                <NA>
                                                  12
```

- This is America. Hardly anybody in the Other party (8). Lets make them MISSING. While we are at it, we will shorten the level names to SD, WD, etc. Here's my strategy to keep the records straight.
  - Get the old levels
  - Revise that as a character variable
  - Use the new labels as names on the old levels, so we can inspect the conversion

Clean up a bunch of variables and value labels ...

```
party_value_old <- levels(anes1$V043116)
party_value_new <- c("SD","WD","ID","I","IR","WR","SR", NA, NA, NA)
names(party_value_old) <- party_value_new
party_value_old
```

```
                                        SD
                                                                  WD
            "0. Strong Democrat (2/1/.)"                  "1. Weak Democrat
                 (2/5-8-9/.)"
                                        ID
                                                                   I
   "2. Independent-Democrat (3-4-5/./5)"                    "3.
       Independent-Independent"
                                        IR
                                                                  WR
 "4. Independent-Republican (3-4-5/./1)"              "5. Weak Republican
     (1/5-8-9/.)"
                                        SR
                                                                <NA>
```

Clean up a bunch of variables and value labels ...

```
            "6. Strong Republican (1/1/.)"     "7. Other;minor
                 party;refuses to say"
                                      <NA>

                                              <NA>
"8. Apolitical (5/./3-8-9 if apolitical)"                        "9.
    DK (8/./.)"
```

```
levels(anes1$V043116) <- names(party_value_old)
## Could instead rely on plyr
## anes1$V043116 <- plyr::mapvalues(anes1$V043116,
##             from = party_value_old,
##             to = names(party_value_old))
table(anes1.orig$V043116, anes1$V043116)
```

Clean up a bunch of variables and value labels ...

```
                                         SD  WD  ID   I  IR  WR  SR
  0. Strong Democrat (2/1/.)            203   0   0   0   0   0   0
  1. Weak Democrat (2/5-8-9/.)            0 179   0   0   0   0   0
  2. Independent-Democrat (3-4-5/./5)     0   0 210   0   0   0   0
  3. Independent-Independent              0   0   0 118   0   0   0
  4. Independent-Republican (3-4-5/./1)   0   0   0   0 138   0   0
  5. Weak Republican (1/5-8-9/.)          0   0   0   0   0 154   0
  6. Strong Republican (1/1/.)            0   0   0   0   0   0 193
  7. Other;minor party;refuses to say     0   0   0   0   0   0   0
  8. Apolitical (5/./3-8-9 if apolitical) 0   0   0   0   0   0   0
  9. DK (8/./.)                           0   0   0   0   0   0   0
```

- Drop unused levels, check final result:

```
anes1$V043116 <- anes1$V043116[, drop = TRUE]
table(anes1$V043116, exclude = NULL)
```

```
 SD   WD   ID    I   IR   WR   SR <NA>
203  179  210  118  138  154  193   17
```

Clean up a bunch of variables and value labels ...

- Ideology: We run into same problem that labels are verbose.
- I'll try a slightly different method here

```
##IDEO
table(anes1$V045117, exclude = NULL)
```

```
                    01. Extremely liberal
                                       20
                             02. Liberal
                                      103
                     03. Slightly liberal
                                      125
          04. Moderate;middle of the road
                                      279
                05. Slightly conservative
                                      143
                         06. Conservative
                                      166
               07. Extremely conservative
                                       31
80. Haven't thought much {DO NOT PROBE}
                                        0
                             88. Don't know
                                        0
```

Clean up a bunch of variables and value labels ...

```
                                        89. Refused
                                                  0
                                               <NA>
                                                345
```

```
levels_old <- levels(anes1$V045117)
levels_old
```

```
 [1] "01. Extremely liberal"
 [2] "02. Liberal"
 [3] "03. Slightly liberal"
 [4] "04. Moderate;middle of the road"
 [5] "05. Slightly conservative"
 [6] "06. Conservative"
 [7] "07. Extremely conservative"
 [8] "80. Haven't thought much {DO NOT PROBE}"
 [9] "88. Don't know"
[10] "89. Refused"
```

```
levels_new <- c("EL", "L", "SL", "M", "SC", "C", "EC", NA, NA, NA)
ideolevels <- data.frame(levels_old = levels_old,
                         levels_new = levels_new,
                         stringsAsFactors = FALSE)
ideolevels
```

Clean up a bunch of variables and value labels ...

```
                              levels_old levels_new
1                  01. Extremely liberal         EL
2                            02. Liberal          L
3                   03. Slightly liberal         SL
4         04. Moderate;middle of the road         M
5             05. Slightly conservative          SC
6                       06. Conservative          C
7            07. Extremely conservative          EC
8  80. Haven't thought much {DO NOT PROBE}     <NA>
9                         88. Don't know       <NA>
10                           89. Refused       <NA>
```

```
anes1$V045117 <- mapvalues(anes1$V045117,
                     from = ideolevels$levels_old,
                     to = ideolevels$levels_new)
anes1$V043116 <- anes1$V045117[, drop = TRUE]
rm(ideolevels) # remove unneeded object
table(anes1.orig$V045117, anes1$V045117) # check
```

Clean up a bunch of variables and value labels ...

```
                                           EL    L   SL    M
  01. Extremely liberal                    20    0    0    0
  02. Liberal                               0  103    0    0
  03. Slightly liberal                      0    0  125    0
  04. Moderate;middle of the road           0    0    0  279
  05. Slightly conservative                 0    0    0    0
  06. Conservative                          0    0    0    0
  07. Extremely conservative                0    0    0    0
  80. Haven't thought much {DO NOT PROBE}   0    0    0    0
  88. Don't know                            0    0    0    0
  89. Refused                               0    0    0    0

                                           SC    C   EC
  01. Extremely liberal                     0    0    0
  02. Liberal                               0    0    0
  03. Slightly liberal                      0    0    0
  04. Moderate;middle of the road           0    0    0
  05. Slightly conservative               143    0    0
  06. Conservative                          0  166    0
  07. Extremely conservative                0    0   31
  80. Haven't thought much {DO NOT PROBE}   0    0    0
  88. Don't know                            0    0    0
  89. Refused                               0    0    0
```

KU

Clean up a bunch of variables and value labels ...

- Gender

```
##Gender
table(anes1$V041109A, exclude = NULL)
```

```
 1. Male 2. Female
     566       646
```

- Re-label the levels

```
levels(anes1$V041109A) <- c("M","F")
```

Clean up a bunch of variables and value labels ...

- Gay Marriage: Note the interesting mismatch between the "value labels" from the original data format and the levels as we see them in R

```
## Gay Marriage
levels(anes1$V043210)
```

```
[1] "1. Should be allowed"
[2] "3. Should not be allowed"
[3] "5. Should not be allowed to marry but should be allowed"
[4] "VOL"
[5] "8. Don't know"
[6] "9. Refused"
```

```
## Shorter names
levels(anes1$V043210) <- c("Allow","No","Some", NA, NA, NA)
anes1$V043210 <- anes1$V043210[ , drop = TRUE]
table(anes1$V043210, exclude = NULL)
```

```
Allow    No  Some  <NA>
  400   705    41    66
```

Clean up a bunch of variables and value labels ...

- Subjectively, those levels seem out of order. Best way to put them right is to run factor

```
anes1$V043210 <- factor(anes1$V043210, levels = c("No", "Some",
     "Allow"))
table(anes1.orig$V043210, anes1$V043210)
```

```
                                                                No Some Allow
  1. Should be allowed                                           0    0   400
  3. Should not be allowed                                     705    0     0
  5. Should not be allowed to marry but should be allowed        0   41     0
  VOL                                                            0    0     0
  8. Don't know                                                  0    0     0
  9. Refused                                                     0    0     0
```

Clean up a bunch of variables and value labels ...

- Expect the economy to get better?

```
## Economy
anes1$V043213 <- anes1$V043213[ , drop = TRUE]
table(anes1$V043213, exclude = NULL)
```

```
 1. Better    3. Worse 5. The same         <NA>
       190         668         343           11
```

Note the levels are subjectively “out of order”. User factor to re-order them

```
lvl <- levels(anes1$V043213)
econnew <- factor(anes1$V043213, levels = lvl[c(2, 3, 1)], labels =
     c("Worse", "Same", "Better"))
table(anes1$V043213, econnew)
```

```
             econnew
              Worse Same Better
 1. Better        0    0    190
 3. Worse       668    0      0
 5. The same      0  343      0
```

```
anes1$V043213 <- econnew
rm(econnew)
```

Clean up a bunch of variables and value labels ...

- How does it make you feel to see the flag?

```
##Flag
(lvl <- levels(anes1$V045145X))
```

```
[1] "1. Extremely good"
[2] "2. Very good"
[3] "3. Somewhat good"
[4] "4. Not very good"
[5] "7. Don't feel anything {VOL}"
[6] "8. Don't know"
[7] "9. Refused"
```

```
anes1$V045145X[anes1$V045145X %in% lvl[6:7]] <- NA
anes1$V045145X <- anes1$V045145X[, drop = TRUE]
table(anes1$V045145X)
```

```
          1. Extremely good                  2. Very good
                        570                           338
           3. Somewhat good              4. Not very good
                        175                            38
7. Don't feel anything {VOL}
                         18
```

Clean up a bunch of variables and value labels ...

```
levels(anes1$V045145X) <- c("EG", "VG", "SG", "NVG", "DFA")
table(anes1$V045145X)
```

```
 EG  VG  SG NVG DFA
570 338 175  38  18
```

What to do about "Don't Feel Anything?"

Should we convert to an ordinal variable?

End result

- Bit by bit, we have brought the data.frame `anes1` into the same coding scheme as `anes2`
- If you want the variable by variable comparison, this is a diagnostic output adapted from some internal functions in kutils.

```
kutils:::keyDiagnostic(anes1, anes2, kutils:::makeKeylist(key))
```

- The manual recodes will show as the "oldvar" columns, while the key displays as the row tables

Lets stash a copy of this working data frame

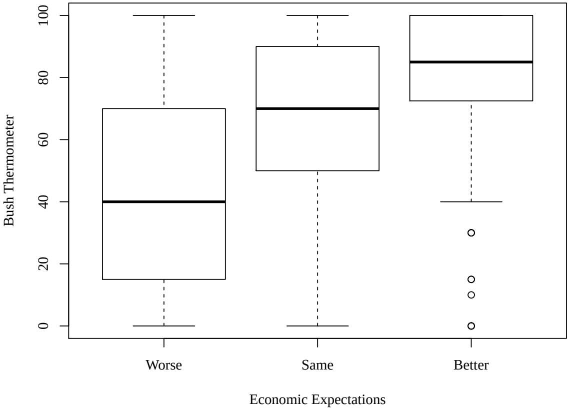
```
today <- format(Sys.time(), "%Y%m%d")
wdir <- "workingdata"
if(!file.exists(wdir)) dir.create(wdir)
saveRDS(anes1, file = file.path(wdir, paste0("nes2004-", today,
    ".rds")))
```

To re-open that, we'd use readRDS().

Outline

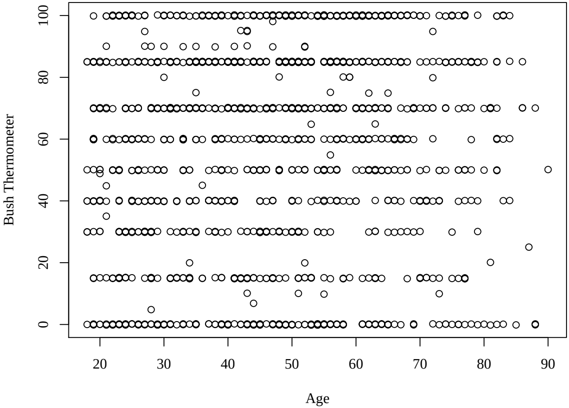
Plot one numeric and one factor variable

```
plot(V043038 ~ V043213, ylab="Bush Thermometer", xlab="Economic
     Expectations", data = anes1)
```



plot sent the work to the boxplot function

How about the Age effect?




```
plot(jitter(V043038) ~ V043250, ylab = "Bush Thermometer", xlab =
    "Age", data = anes1)
```

The jitter() function “scatters” points, avoids pile ups

KU

How about the Age effect? ...

- If you get interested in better plots for large numeric data sets, there are alternatives in addon packages.

Numeric Predictor

- Predict the Bush-Kerry Difference from respondent Age

```
mod1 <- lm(th.bk ~ V043250, data = anes1)
summary(mod1)
```

```
Call:
lm(formula = th.bk ~ V043250, data = anes1)

Residuals:
     Min       1Q   Median       3Q      Max
-108.821  -42.753   -2.003   42.905  103.340

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -6.84133    4.59553  -1.489    0.137
V043250      0.18426    0.09181   2.007    0.045 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 53.89 on 1189 degrees of freedom
  (21 observations deleted due to missingness)
Multiple R-squared:  0.003376,  Adjusted R-squared:  0.002538
F-statistic: 4.028 on 1 and 1189 DF,  p-value: 0.04498
```

Add A Factor as a Predictor

```r
mod2 <- lm(th.bk ~ V043250 + V041109A, data = anes1)
summary(mod2)
```

```
Call:
lm(formula = th.bk ~ V043250 + V041109A, data = anes1)

Residuals:
     Min       1Q   Median       3Q      Max
-113.174  -42.222   -2.782   42.478  107.164

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -3.08501    4.83135  -0.639   0.5232
V043250      0.19128    0.09166   2.087   0.0371 *
V041109AF   -7.71339    3.12318  -2.470   0.0137 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 53.77 on 1188 degrees of freedom
  (21 observations deleted due to missingness)
Multiple R-squared:  0.008467,  Adjusted R-squared:  0.006798
F-statistic: 5.072 on 2 and 1188 DF,  p-value: 0.006404
```

Now Look Back at What R did with the Gender Predictor

- R creates the "design matrix", the purely numerical representation of the variables. Notice it creates the dummy variable for Gender.

```
mod2mm <- model.matrix(mod2)
head(mod2mm)
```

```
  (Intercept) V043250 V041109AF
1           1      50         0
2           1      47         0
3           1      37         1
4           1      71         0
5           1      62         1
6           1      53         0
```

Add Party ID as a Predictor

```
mod3 <- lm(th.bk ~ V043250 + V041109A + V043116, data = anes1)
summary(mod3)
```

```
Call:
lm(formula = th.bk ~ V043250 + V041109A + V043116, data = anes1)

Residuals:
     Min       1Q   Median       3Q      Max
-158.748  -27.161    1.873   27.349  126.605

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -53.30952   10.34966  -5.151 3.22e-07 ***
V043250       0.02894    0.08842   0.327   0.7435
V041109AF    -0.38823    2.94737  -0.132   0.8952
V043116L     -1.53779   10.46541  -0.147   0.8832
V043116SL    25.50127   10.32322   2.470   0.0137 *
V043116M     49.84527    9.92408   5.023 6.21e-07 ***
V043116SC    79.64971   10.23575   7.782 2.08e-14 ***
V043116C    103.70832   10.14592  10.222  < 2e-16 ***
V043116EC   111.47834   12.37917   9.005  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 42.8 on 850 degrees of freedom
  (353 observations deleted due to missingness)
Multiple R-squared:  0.4094,  Adjusted R-squared:  0.4039
F-statistic: 73.67 on 8 and 850 DF,  p-value: < 2.2e-16
```

Check the model matrix now

```
mod3mm <- model.matrix(mod3)
head(mod3mm, 10)
```

```
   (Intercept) V043250 V041109AF V043116L V043116SL V043116M V043116SC V043116C V043116EC
1            1      50         0        0         0        1         0        0         0
2            1      47         0        0         1        0         0        0         0
3            1      37         1        0         0        0         0        1         0
5            1      62         1        0         0        0         0        1         0
6            1      53         0        0         0        1         0        0         0
7            1      49         1        0         0        0         1        0         0
8            1      56         1        0         0        1         0        0         0
10           1      47         1        0         0        0         0        1         0
11           1      30         0        0         0        1         0        0         0
13           1      44         0        0         0        1         0        0         0
```

- In the olden days (or now if you use some software), the user has to create all those "dummy" columns to represent the levels. In R, we avoid it.

Save the regression objects in an RData file

For my lecture about regression tables, I'll need those fitted models, so I might as well save them as well.

```
save(mod1, mod2, mod3, file = file.path(wdir,
    paste0("nes2004-objects-", today,  ".RData")))
```

Outline

What is the focus?

- R uses variable classes which guide plotting and analysis
- The classes we focus on–integer, floating point, character, and factor–are workhorses in statistical analysis
- Re-organizing data requires care, it is easy to get it wrong.

The Variable Key is a new thing from CRMDA

- The Key is our strategy to put research projects on a commonly understood footing, avoid the danger that errors are hidden in details understood only to the research assistants
- Even if you decide you don't want to use it now, please check back on the kutils package from time-to-time because we introduce new features.

References

R Core Team (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Session

```
sessionInfo()
```

```
R version 3.6.0 (2019-04-26)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 19.04

Matrix products: default
BLAS:   /usr/lib/x86_64-linux-gnu/atlas/libblas.so.3.10.3
LAPACK: /usr/lib/x86_64-linux-gnu/atlas/liblapack.so.3.10.3

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
      LC_TIME=en_US.UTF-8
 [4] LC_COLLATE=en_US.UTF-8     LC_MONETARY=en_US.UTF-8
      LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C                  LC_ADDRESS=C
[10] LC_TELEPHONE=C             LC_MEASUREMENT=en_US.UTF-8
     LC_IDENTIFICATION=C

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] kutils_1.69       foreign_0.8-71    plyr_1.8.4
     rockchalk_1.8.144
```

Session ...

```

loaded via a namespace (and not attached):
 [1] Rcpp_1.0.1      lattice_0.20-38 MASS_7.3-51.4   grid_3.6.0
      nlme_3.1-140    xtable_1.8-4
 [7] stats4_3.6.0    zip_2.0.2       carData_3.0-2   minqa_1.2.4
      nloptr_1.2.1    Matrix_1.2-17
[13] pbivnorm_0.6.0  boot_1.3-22     openxlsx_4.1.0  splines_3.6.0
     lme4_1.1-21     tools_3.6.0
[19] compiler_3.6.0  mnormt_1.5-5    lavaan_0.6-3
```