dm

Henrik Renlund

March 4, 2018

Contents

1 About **1**

2 `dm` and related functions **2**

2.1 The problem that is trying to be solved 2

2.2 The elevator pitch description 2

2.3 The stuck-in-an-elevator description 2

2.4 Toy Example 4

1 About

`dm` is a package for functions relating to data management.

2 `dm` and related functions

2.1 The problem that is trying to be solved

A statistical report sometimes build an *analytical data base* (ADB) from multiple sources, variables that might need to be renamed and (if categorical) recoded (and possibly transformed), the documentation of which is *significantly boring*.

The `dm` functions is an interactive-ish way of creating an ADB which both inspects the chosen variables and "documents" the process.

2.2 The elevator pitch description

1. point to variables (from possibly different sources), one at the time, with `dm` (along with possible renaming, recoding and transformation). This gives a summary of the variable pointed to [1], and the information is stored in a list somewhere.

2. create the ADB by `dm_create`.

3. get easy-to-print documentation of where variables came from (`dm_doc2latex`) and what recodings have been done (`dm_recode2latex`).

So, the point really is to get (3) "for free" in a way that is connected to the creation of the ADB.

2.3 The stuck-in-an-elevator description

If most variables are picked form the same source, this can be set in options.

```
opts_dm$set('default_db' = 'MyDataBase')
```

If that is done, `dm` only needs a `var` argument, the name of the var you want to add. But you can use

- `var`, name of variable in source

- `name`, optional, if you want a new name for the variable (else it is set to `var`)

- `db`, name of data frame (or similar) where `var` exists (else will look at the default location, if set)

- `recode`, a list that specifies the recoding. This is the `L` argument for the `recode` function that this package provides (see the help for that functions)

- `transf` a function for transforming (this might be something like a character-to-date function like `ymd` from the lubridate package)

[1]Typically one wants to to this procedure anyway to sanity check all variables that are to be included.

- `comment` if you want to keep some comment about the variable
- `label` if you want to give the variable a ”label” (i.e. the value of the label attribute)
- `keep.label` if `var` already has a label in `db`, should this be kept? (only if no `label` is provided)

Then as `dm` is evaluated, information about the variable is printed (to see range, levels and such).

```
dm(var = 'gEndEr', name = 'gender', label = "Perceived Gender")
    ## is followed by information being printed
```

The information about the options is stored in a list (by default ”dm_doc” in an environment ”dm_envir”).[2] The key is the 'name' element, so as long as that is not changed, you can rerun the function with new arguments if something went wrong

```
dm('gEndEr', 'gender', label = "Biological Gender") ## overwrites
    ## the 'gender' entry
```

Else, kill all documentation and start again

```
dm_doc(kill = TRUE, prompt = FALSE) ## or possibly kill only this
  ## entry dm:::dm_doc_set('gender', NULL)
```

The documentation can be accessed

```
myDoc <- dm_doc()
print(myDoc)  ## N.B not all information is printed
```

Once all variables are created you can either store the ”documentation” (and point to it later) or go on to create the ADB with `dm_create`. Specify a set if individuals (vector of id's) and, if necessary a vector of how individuals are indentified in different data frames. If the `doc` argument is not provided it will just look in `dm_doc()`.

```
id_key = c('MyDataBase' = 'id', 'Other1' = 'ID', 'Other2' = 'idno')
ADB <- dm_create(set = MyDataBase$id, id.name = id_key)
```

Now you have an ADB and you can print `dm_doc()` to show where all variables come from. You can get all recodings from

[2]This is due to it begin poor for functions practice to write to objects in the global environment.

```r
lapply(dm_doc(), FUN = function(x) x$recode_table)
```

There is also convience functions `dm_recode2latex` and `dm_doc2latex` which will print all tables and documentation, respectively, in LaTeX format.

2.4 Toy Example

We create some toy data

```r
n <- 200
BL <- data.frame(
    id = structure(sprintf("id%d", 1:n),
                   label = "identification"),
    aalder = structure(round(rnorm(n, 50, 10)),
                       label = "Age at some time",
                       foo = 'whatever'),
    vikt = rpois(n, 50),
    gr = sample(c('A1', 'A2', 'B1', 'c', 'unknown'), n, TRUE),
    koon = structure(sample(c('M', 'K'), n, T),
                     label = "The Sex"),
    nar = as.Date("2001-01-01") + runif(n, 0, 3650),
    stringsAsFactors = FALSE
)
BL$vikt[sample(1:n, 15)] <- NA
BL$gr[sample(1:n, 10)] <- NA
m <- .9*n
COMP <- data.frame(
    ID = structure(sample(BL$id, m),
                   label = "identification"),
    foo = rbinom(m, 1, .2),
    bar = structure(rexp(m, 1/150),
                    label = "Time passed")
)
```

There are some functions to help look for relevant variables.

```r
db_info(BL) ## prints names and 'label' attributes

##   source variable            label     class
## 1     BL       id   identification character
## 2     BL   aalder Age at some time   numeric
## 3     BL     vikt                    integer
## 4     BL       gr                  character
## 5     BL     koon          The Sex character
## 6     BL      nar                       Date
```

```
dm_find(pattern = 'time') ## looks in names and labels

## dm_find found:

##   source variable             label   class
## 2     BL   aalder Age at some time numeric
## 3   COMP      bar      Time passed numeric
```

Most variables of interest are in `BL` so set this as deault.

```
opts_dm$set('default_db' = 'BL')
```

Next, we add the first variable (and view the output)

```
dm('koon', 'Gender',
   recode = list('Male' = 'M', 'Female' = 'K'))

## ---------------------------------------------------------------
## Adding data base 'BL' entry 'koon' as variable 'Gender'
## A variable of class: character
##     with attributes: label
##           and label: 'The Sex'
## There are 0 (0 percent) missing
##       and 2 (1 percent) unique values
## Since there are less than 20 unique vales we tabulate them:
##
##
##    K    M <NA>
##   83  117    0
##
## Cross-tabulating the recoding:
##
##     Gender
## koon Male Female
##    K    0     83
##    M  117      0
```

Next, we add some more variables (but hide the output)

```
dm('aalder', 'Age')
dm('nar', 'When', comment = "wtf?")
dm('foo', 'event', db = 'COMP',
   recode = list('No' = '0', 'Yes' = 1),
   label = "An event at some time")
dm('bar', 'time', db = 'COMP', transf = log)
## 'gr' will be recoded in a more complex way
```

```
L <- list('A' = c('A1', 'A2'),
          'BC' = c('B1', 'c'),
          'Unknown' = c('unknown', NA))
dm('gr', recode = L, label = 'Group')
```

When we are done, we create the ADB with

```
ADB <- dm_create(set = BL$id,
                 id.name = c('BL' = 'id', 'COMP' = 'ID'))

## Fixing variable no.1: Gender
## Fixing variable no.2: Age
## Fixing variable no.3: When
## Fixing variable no.4: event
## Fixing variable no.5: time
## Fixing variable no.6: gr

db_info(ADB)

##   source variable                  label   class
## 1    ADB       id                         factor
## 2    ADB   Gender                The Sex  factor
## 3    ADB      Age      Age at some time numeric
## 4    ADB     When                           Date
## 5    ADB    event An event at some time  factor
## 6    ADB     time           Time passed numeric
## 7    ADB       gr                  Group  factor
```

We can view, or get the information

```
## myDoc <- dm_doc()
dm_doc() ## only prints partial information in the doc

##     name    var   db transf                label comment
## 1 Gender   koon   BL                      The Sex
## 2    Age aalder   BL             Age at some time
## 3   When    nar   BL                                wtf?
## 4  event    foo COMP         An event at some time
## 5   time    bar COMP    log           Time passed
## 6     gr     gr   BL                        Group
```

If we are using LaTeX, we can get the code for this with

```
dm_doc2latex(caption = "Variables and their origin.")
```

and all recode-information with

Table 1: Variables and their origin.

name	var	db	label	comment
`Gender`	`koon`	**BL**	The Sex	
`Age`	`aalder`	**BL**	Age at some time	
`When`	`nar`	**BL**		*wtf?*
`event`	`foo`	**COMP**	An event at some time	
`time`	`bar`	**COMP**	Time passed	
`gr`	`gr`	**BL**	Group	

```
dm_recode2latex()
```

Table 2: Recoding of data base entry `koon` into `Gender`.

old ↓ new →	Male	Female
K	0	83
M	117	0

Table 3: Recoding of data base entry `foo` into `event`.

old ↓ new →	No	Yes
0	148	0
1	0	32

Else, you can get the information from the 'print' of the `dm_doc()` and the recodings with, respectively,

```
d <- print(dm_doc(), print = FALSE)
lapply(dm_doc(), FUN = function(x) x$recode_table)
```

Table 4: Recoding of data base entry `gr` into `gr`.

old $\downarrow$ new $\rightarrow$	A	BC	Unknown
A1	40	0	0
A2	35	0	0
B1	0	37	0
c	0	37	0
unknown	0	0	41
NA	0	0	10