Handbook for the dm R Package

Robert Schnitman

April 04, 2020 (Last updated: July 02, 2020)

Contents

Introduction

The `dm` package is inspired by the data management functions in other statistical packages such as Stata and SPSS. The purpose of this package is to simplify data management/auditing.

Installation

Currently, this package is only available on Github, so please use `devtools` to install this package.

```
if (!require(devtools)) {

  install.packages('devtools')
  library(devtools)

}

install_github('robertschnitman/dm')
```

```
library(dm)
```

Chapter 1

between()

The `between()` function acts similar to SQL's `BETWEEN` clause[1], conditioning values between lower and upper limits.

The function has four parameters:

1. `x`, a vector;
2. `low`, a user-specified lower limit;
3. `high`, a user-specified upper limit; and
4. `inclusive = TRUE`, an input that determines whether the `between()` function will condition inclusive (if `TRUE`) of the specified limits or not (if `FALSE`).

A vector is returned by the function.

1.1 Vector Case

```
k <- 1:100
between(k, 5, 24)
```

```
##   [1] FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [13]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [37] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [49] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [61] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [73] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [85] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [97] FALSE FALSE FALSE FALSE
```

[1] https://libguides.library.kent.edu/SPSS/RecodeVariables

1.2 Data Frame Case

```
subset(mtcars, between(mpg, 18, 21, inclusive = FALSE))
```

```
##                    mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## Merc 280          19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
## Pontiac Firebird  19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
## Ferrari Dino      19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
```

Chapter 2

dedup()

The `dedup()` function deduplicates data, functioning the same as `unique()`.

```
dedup(iris$Species)
```

```
## [1] setosa     versicolor virginica
## Levels: setosa versicolor virginica
```

2.1 dup_rows()

The `dup_rows()` functions returns the duplicated rows in a 2-dimensional dataset. It is functionally the same as `x[duplicated(x), ]`, where `x` is a matrix or data frame.

```
dup_rows(iris)
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
## 143          5.8         2.7          5.1         1.9 virginica
```

2.2 dup_nrow()

The `dup_nrow()` function counts the number of duplicates in a dataset.

```
dup_nrow(iris)
```

```
## [1] 1
```

2.3 dup_mark()

The `dup_mark()` function codes duplicate rows with a 1 and non-duplicates with a 0.

```
dup_mark(iris)
```

```
##   [1] 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
##  [38] 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
##  [75] 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
## [112] 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 1 0 0 0 0 0
## [149] 0 0
```

Chapter 3

differ2()

The `differ2()` function codes differences between two vectors as a 1; otherwise, 0. This function is based on Stata's `egen differ()` command. [1]

```
x <- 1:5
y <- c(1, 6, 3, 8, 5)

z <- differ2(x, y)

cbind(x, y, z)
```

```
##      x y z
## [1,] 1 1 0
## [2,] 2 6 1
## [3,] 3 3 0
## [4,] 4 8 1
## [5,] 5 5 0
```

[1] https://libguides.library.kent.edu/SPSS/RecodeVariables

Chapter 4

expected()

The `expected()` function stops an execution if a specific class is not expected.

```
expected(2, 'character')
```

```
## Error in expected(2, "character"): Class "character" expected. Received class numeric.
```

Chapter 5

fill()

When reporting the gender distribution of a school to U.S. state education agencies, for example, you may have missing student data on gender but are still required to produce the full count of students. The `fill()` function can be useful in this regard, as it will fill in missing values while maintaining the existing proportions of the unique values (excluding NA values).

```
# Always set the seed before using randomizing functions.
set.seed(1)

# Our vector to fill in.
k <- c(0, 1, NA, 1, 0, NA, NA, NA)

# Original proportions
prop_original <- prop.table(table(k))

prop_original
```

```
## k
##   0   1
## 0.5 0.5
```

```
# Apply fill() function
fill_k <- fill(k) # A warning may occur

# Compare original vector to the new vector.
cbind(k, fill_k)
```

```
##       k fill_k
## [1,]  0      0
## [2,]  1      1
## [3,] NA      1
```

```
## [4,]  1      1
## [5,]  0      0
## [6,] NA      1
## [7,] NA      0
## [8,] NA      0
```

```r
# Check if proportions were maintained
prop_new <- prop.table(table(fill_k))

prop_new
```

```
## fill_k
##   0   1
## 0.5 0.5
```

```r
all(prop_original == prop_new)
```

```
## [1] TRUE
```

Chapter 6

keep()

The `keep()` function behaves the same as `subset()`. The naming scheme is inspired by Stata's `keep` command.[1]

```
keep(mtcars, between(mpg, 18, 21))
```

```
##                    mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## Merc 280          19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
## Pontiac Firebird  19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
## Ferrari Dino      19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
```

[1] https://libguides.library.kent.edu/SPSS/RecodeVariables

Chapter 7

`load_libraries()`

The `load_libraries()` function checks whether a set of libraries have been installed: if the libraries exist, the function will load them; otherwise, the function will install and then load them.

```r
load_libraries('tidyverse', 'abind', 'ggformula')
```

Chapter 8

numNA()

Similar to Stata,[1] numNA counts the number of missing values: if m is set to 1, then it counts the number of missing values per row; if set to 2, then it counts the number of missing values per column; otherwise, it counts the total number of missing values.

```
numNA(airquality)
```

```
## [1] 44
```

8.1 rowNA()/colNA()

The function `rowNA()` is the equivalent of `numNA(x, 1)`, where x is a data frame. It counts the number of missing values per row. The function `colNA()` does the same but per column.

```
rowNA(airquality)
```

```
##   [1] 0 0 0 0 2 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 2 0 0 0 0 1 1 1 1 1 1 1
##  [38] 0 1 0 0 1 1 0 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 1 0 0 0 0 0 0 1 0 0
##  [75] 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 1 1 0 0 0 1 0 0 0 0
## [112] 0 0 0 1 0 0 0 1 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
## [149] 0 1 0 0 0
```

```
colNA(airquality)
```

```
##   Ozone Solar.R    Wind    Temp   Month     Day 
##      37       7       0       0       0       0
```

[1] https://libguides.library.kent.edu/SPSS/RecodeVariables

Chapter 9

parity()

The function `parity()` determines whether a number is even or odd. This function and its related functions `is.even()` and `is.odd()` are inspired by Julia's `iseven()` and `isodd()` functions.[1] The `get_even()` and `get_odd()` functions subset a vector for its even and odd numbers respectively.

```
parity(mtcars$carb)
```

```
##  [1] "even" "even" "odd"  "odd"  "even" "odd"  "even" "even" "even" "even"
## [11] "even" "odd"  "odd"  "odd"  "even" "even" "even" "odd"  "even" "odd"
## [21] "odd"  "even" "even" "even" "even" "odd"  "even" "even" "even" "even"
## [31] "even" "even"
```

```
is.even(1)
```

```
## [1] FALSE
```

```
is.even(mtcars$carb)
```

```
##  [1]  TRUE  TRUE FALSE FALSE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
## [13] FALSE FALSE  TRUE  TRUE  TRUE FALSE  TRUE FALSE FALSE  TRUE  TRUE  TRUE
## [25]  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
```

```
is.odd(2)
```

```
## [1] FALSE
```

```
is.odd(mtcars$carb)
```

```
##  [1] FALSE FALSE  TRUE  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE
## [13]  TRUE  TRUE FALSE FALSE FALSE  TRUE FALSE  TRUE  TRUE FALSE FALSE FALSE
## [25] FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
```

[1] https://libguides.library.kent.edu/SPSS/RecodeVariables

```
get_odd(mtcars$carb)
```

```
##  [1] 1 1 1 3 3 3 1 1 1 1
```

```
get_even(mtcars$carb)
```

```
##  [1] 4 4 2 4 2 2 4 4 4 4 4 2 2 2 4 2 2 2 4 6 8 2
```

Chapter 10

recode()

The `recode()` function replaces an initial set of values with a new set of values for a vector. The function `recode_all()` applies `recode()` to all columns in a dataset.

The inputs for `recode()` are the following:

1. a vector;
2. a set of values to replace ("initial values"); and
3. a set of values values that will replace the initial values.

```
mtcars$am
```

```
##  [1] 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 1 1 1 1
```

```
recode(mtcars$am, 0:1, 2:3)
```

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

10.1 NAvl()

The `NAvl()` function replaces a vector's missing values with a specified value set. The function `NAvl0` replaces a vector's missing values with 0.

```
with(airquality, NAvl(Ozone, mean(Ozone, na.rm = TRUE)))
```

```
##   [1]  41.00000  36.00000  12.00000  18.00000  42.12931  28.00000  23.00000
##   [8]  19.00000   8.00000  42.12931   7.00000  16.00000  11.00000  14.00000
##  [15]  18.00000  14.00000  34.00000   6.00000  30.00000  11.00000   1.00000
##  [22]  11.00000   4.00000  32.00000  42.12931  42.12931  42.12931  23.00000
##  [29]  45.00000 115.00000  37.00000  42.12931  42.12931  42.12931  42.12931
##  [36]  42.12931  42.12931  29.00000  42.12931  71.00000  39.00000  42.12931
##  [43]  42.12931  23.00000  42.12931  42.12931  21.00000  37.00000  20.00000
```

```
## [50]  12.00000  13.00000  42.12931  42.12931  42.12931  42.12931  42.12931
## [57]  42.12931  42.12931  42.12931  42.12931  42.12931 135.00000  49.00000
## [64]  32.00000  42.12931  64.00000  40.00000  77.00000  97.00000  97.00000
## [71]  85.00000  42.12931  10.00000  27.00000  42.12931   7.00000  48.00000
## [78]  35.00000  61.00000  79.00000  63.00000  16.00000  42.12931  42.12931
## [85]  80.00000 108.00000  20.00000  52.00000  82.00000  50.00000  64.00000
## [92]  59.00000  39.00000   9.00000  16.00000  78.00000  35.00000  66.00000
## [99] 122.00000  89.00000 110.00000  42.12931  42.12931  44.00000  28.00000
## [106]  65.00000  42.12931  22.00000  59.00000  23.00000  31.00000  44.00000
## [113]  21.00000   9.00000  42.12931  45.00000 168.00000  73.00000  42.12931
## [120]  76.00000 118.00000  84.00000  85.00000  96.00000  78.00000  73.00000
## [127]  91.00000  47.00000  32.00000  20.00000  23.00000  21.00000  24.00000
## [134]  44.00000  21.00000  28.00000   9.00000  13.00000  46.00000  18.00000
## [141]  13.00000  24.00000  16.00000  13.00000  23.00000  36.00000   7.00000
## [148]  14.00000  30.00000  42.12931  14.00000  18.00000  20.00000
```

```r
NAv10(airquality$Ozone)
```

```
##   [1]  41  36  12  18   0  28  23  19   8   0   7  16  11  14  18  14  34   6
##  [19]  30  11   1  11   4  32   0   0   0  23  45 115  37   0   0   0   0   0
##  [37]   0  29   0  71  39   0   0  23   0   0  21  37  20  12  13   0   0   0
##  [55]   0   0   0   0   0   0   0 135  49  32   0  64  40  77  97  97  85   0
##  [73]  10  27   0   7  48  35  61  79  63  16   0   0  80 108  20  52  82  50
##  [91]  64  59  39   9  16  78  35  66 122  89 110   0   0  44  28  65   0  22
## [109]  59  23  31  44  21   9   0  45 168  73   0  76 118  84  85  96  78  73
## [127]  91  47  32  20  23  21  24  44  21  28   9  13  46  18  13  24  16  13
## [145]  23  36   7  14  30   0  14  18  20
```

10.2 switchv()

The `switchv()` function behaves the same as `switch()` except that it applies over a vector (i.e., it is a vectorized version of the latter). The function `swap` is a shorthand synonym for `switchv()`. The function `switchv_all()`/`swap_all()` applies `switchv()`/`swap()` to all columns in a data frame.

These functions are inspired by SPSS's `RECODE` command.[1]

```r
switchv(iris$Species, setosa = 4, versicolor = 5, virginica = 6)
```

```
##   [1] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
##  [38] 4 4 4 4 4 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
##  [75] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6 6 6 6 6
## [112] 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
## [149] 6 6
```

[1] https://libguides.library.kent.edu/SPSS/RecodeVariables

Chapter 11

rename_file()

The function `rename_file()` renames files in a directory based on a given pattern. It has three inputs:

1. `filepath`, which specifies the desired directory of where the files to be renamed are;
2. `pattern_now`, which is a regular expression pattern of the file names as they currently are; and
3. `pattern_new`, which is the desired string that replaces the file names detected in `pattern_now`.

```
# From a folder, replace file names having "Fall 2018" with "Spring 2019."
rename_file('C:/my/folder/', 'Fall 2018', 'Spring 2019')
```

Chapter 12

`stopif()`

The `stopif()` function halts an execution if a condition is met.

```
stopif(2 < 3)
```

```
## Error in stopif(2 < 3): Stop condition met.
```

```
stopif(2 < 3, 'This is a custom message! STOP!')
```

```
## Error in stopif(2 < 3, "This is a custom message! STOP!"): This is a custom message! STOP!
```

Chapter 13

type_class()

The function `type_class()` provides the type and class of an object–`tc()` is a synonym. The `type()` function provides the object's type.

```
type_class(mtcars)
```

```
##         type        class
##       "list" "data.frame"
```

```
tc(mtcars)
```

```
##         type        class
##       "list" "data.frame"
```

```
type(mtcars)
```

```
## [1] "list"
```

Chapter 14

zero_flag()

The `zero_flag()` function flags a zero at the start of a value. This function is useful for formatting numbers with a specified number of digits such as Social Security Numbers.

```
zero_flag(700, 4, format = 'd') # == '0700'
```

```
## [1] "0700"
```

References

Julia documentation. *numbers*. https://docs.julialang.org/en/release-0.4/stdlib/numbers/

Kent State University. *SPSS Recode*. https://libguides.library.kent.edu/SPSS/RecodeVariables

Stata. *egen*. https://www.stata.com/manuals13/degen.pdf

w3schools.com. *SQL BETWEEN OPERATOR*. https://www.w3schools.com/sql/sql_between.asp

See also

`dm` GitHub Page. https://github.com/robertschnitman/dm

IBM's SPSS. https://www.ibm.com/analytics/spss-statistics-software

Robert Schnitman's Profile. https://robertschnitman.netlify.app/

Stata's website. https://www.stata.com/