Demonstrations

Sasha D. Hafner

13 March, 2022

Load functions

```
ff <- list.files(pattern = '\\.R$')
for(i in ff) source(i)
```

aggregate2

A wrapper for `aggregate` that accepts multiple functions and simpler arguments. Does not accept formula notation.

Example from `aggregate` help file:

```
aggregate(breaks ~ wool + tension, data = warpbreaks, mean)
```

```
##   wool tension   breaks
## 1    A       L 44.55556
## 2    B       L 28.22222
## 3    A       M 24.00000
## 4    B       M 28.77778
## 5    A       H 24.55556
## 6    B       H 18.77778
```

To include sd and n, use `aggregate2`:

```
aggregate2(warpbreaks, x = 'breaks', by = c('wool', 'tension'),
           FUN = list(mean = mean, sd = sd, n = length))
```

```
##   wool tension breaks.mean breaks.sd breaks.n
## 1    A       L    44.55556 18.097729        9
## 2    B       L    28.22222  9.858724        9
## 3    A       M    24.00000  8.660254        9
## 4    B       M    28.77778  9.431036        9
## 5    A       H    24.55556 10.272671        9
## 6    B       H    18.77778  4.893306        9
```

Accepts multiple variables (as in `aggregate`).

dfcombos

Something like `expand.grid` for data frames.

dfsumm

Generate a data frame summary more detailed and compact than `summary` output.

```r
dfsumm(attenu)
```

```
##
##  182 rows and 5 columns
##  182 unique rows
##                        event     mag station    dist   accel
## Class                numeric numeric  factor numeric numeric
## Minimum                    1       5    1008     0.5   0.003
## Maximum                   23     7.7    c266     370    0.81
## Mean                    14.7    6.08     262    45.6   0.154
## Unique (excld. NA)        23      17     117     153     120
## Missing values             0       0      16       0       0
## Sorted                  TRUE   FALSE   FALSE   FALSE   FALSE
##
```

Compare to `summary`.

```r
summary(attenu)
```

```
##      event            mag            station          dist       
##  Min.   : 1.00   Min.   :5.000   117    :  5   Min.   :  0.50  
##  1st Qu.: 9.00   1st Qu.:5.300   1028   :  4   1st Qu.: 11.32  
##  Median :18.00   Median :6.100   113    :  4   Median : 23.40  
##  Mean   :14.74   Mean   :6.084   112    :  3   Mean   : 45.60  
##  3rd Qu.:20.00   3rd Qu.:6.600   135    :  3   3rd Qu.: 47.55  
##  Max.   :23.00   Max.   :7.700   (Other):147   Max.   :370.00  
##                                  NA's   : 16                   
##      accel        
##  Min.   :0.00300  
##  1st Qu.:0.04425  
##  Median :0.11300  
##  Mean   :0.15422  
##  3rd Qu.:0.21925  
##  Max.   :0.81000  
##
```

interpm

Fill in missing observations for multiple columns via interpolation. `interpm` calls `approx`.

```r
args(interpm)
```

```
## function (dat, x, ys, ...) 
## NULL
```

```r
dat <- data.frame(time = 1:30, a = rnorm(30), b = rnorm(30), c = rnorm(30))
dat[5:10, -1] <- NA
dat[20:22, 'a'] <- NA

dat
```

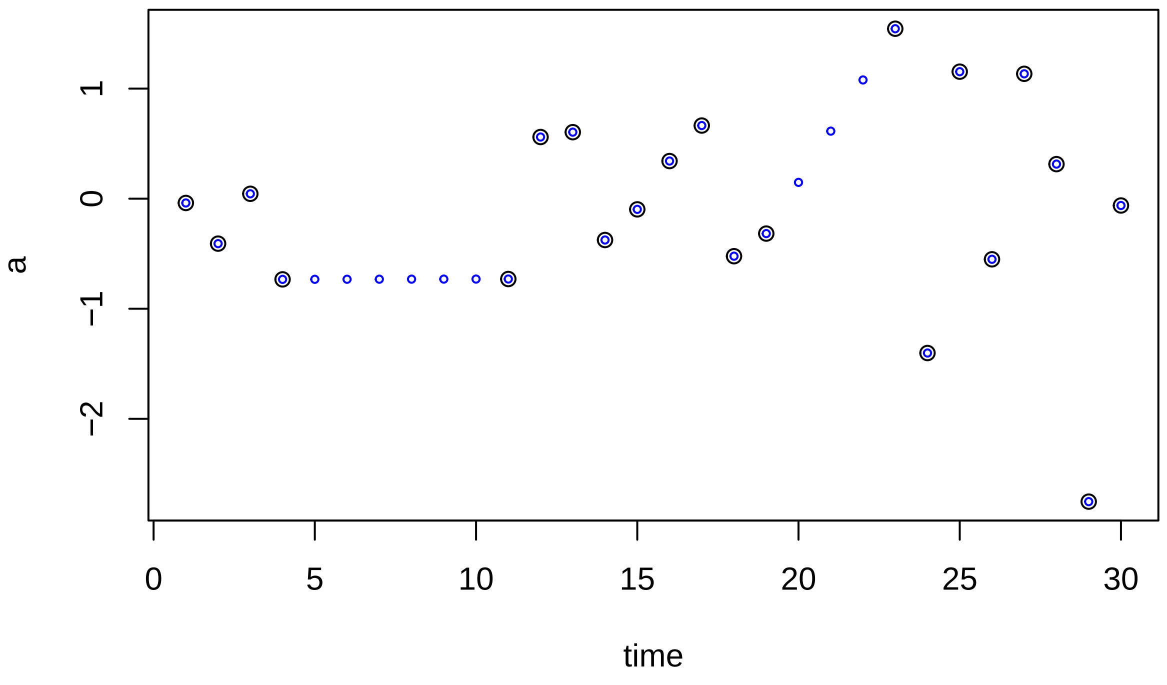
```
##    time           a           b          c
## 1     1 -0.03888817 -1.79609791 -0.9602124
## 2      2 -0.40818793  1.53860069 -0.9564085
## 3      3  0.04456668  0.61719530  0.2146302
## 4      4 -0.73286907 -0.10239525  0.5617518
## 5      5          NA          NA         NA
## 6      6          NA          NA         NA
## 7      7          NA          NA         NA
## 8      8          NA          NA         NA
## 9      9          NA          NA         NA
## 10    10          NA          NA         NA
## 11    11 -0.72969274 -1.18508484 -1.0277732
## 12    12  0.56030426  0.43406589 -0.2376317
## 13    13  0.60447539 -2.19346365  1.5675643
## 14    14 -0.37524293 -1.12635012  1.1589916
## 15    15 -0.09690025  0.06199881 -0.5146457
## 16    16  0.34195729  1.34036462 -0.6743375
## 17    17  0.66440598  1.14009113  1.3159003
## 18    18 -0.52239752 -0.93336913  0.3507864
## 19    19 -0.31703752 -0.45748016  1.3880237
## 20    20          NA -1.29839740 -0.1653403
## 21    21          NA  0.08024910 -1.2119020
## 22    22          NA -1.20218739 -2.4181308
## 23    23  1.54411969 -2.03472777  0.2837416
## 24    24 -1.40202912  2.08610654 -0.6737569
## 25    25  1.15391820  0.41642578 -0.5024951
## 26    26 -0.55058327 -0.83719097  0.1497341
## 27    27  1.13456418 -0.11426171  0.2227607
## 28    28  0.31416379  0.09100579  0.3502516
## 29    29 -2.75216321  0.31733766 -0.8209082
## 30    30 -0.06183819  0.71872316 -0.1603179
```

```r
dat2 <- interpm(dat, 'time', c('a', 'b', 'c'))

dat2
```

```
##    time           a           b          c
## 1     1 -0.03888817 -1.79609791 -0.9602124
## 2     2 -0.40818793  1.53860069 -0.9564085
## 3     3  0.04456668  0.61719530  0.2146302
## 4     4 -0.73286907 -0.10239525  0.5617518
## 5     5 -0.73241531 -0.25706520  0.3346768
## 6     6 -0.73196155 -0.41173514  0.1076018
## 7     7 -0.73150778 -0.56640508 -0.1194732
## 8     8 -0.73105402 -0.72107502 -0.3465482
## 9     9 -0.73060026 -0.87574496 -0.5736232
## 10   10 -0.73014650 -1.03041490 -0.8006982
## 11   11 -0.72969274 -1.18508484 -1.0277732
## 12   12  0.56030426  0.43406589 -0.2376317
## 13   13  0.60447539 -2.19346365  1.5675643
## 14   14 -0.37524293 -1.12635012  1.1589916
## 15   15 -0.09690025  0.06199881 -0.5146457
## 16   16  0.34195729  1.34036462 -0.6743375
## 17   17  0.66440598  1.14009113  1.3159003
## 18   18 -0.52239752 -0.93336913  0.3507864
## 19   19 -0.31703752 -0.45748016  1.3880237
## 20   20  0.14825178 -1.29839740 -0.1653403
## 21   21  0.61354108  0.08024910 -1.2119020
## 22   22  1.07883038 -1.20218739 -2.4181308
## 23   23  1.54411969 -2.03472777  0.2837416
## 24   24 -1.40202912  2.08610654 -0.6737569
## 25   25  1.15391820  0.41642578 -0.5024951
## 26   26 -0.55058327 -0.83719097  0.1497341
## 27   27  1.13456418 -0.11426171  0.2227607
## 28   28  0.31416379  0.09100579  0.3502516
## 29   29 -2.75216321  0.31733766 -0.8209082
## 30   30 -0.06183819  0.71872316 -0.1603179
```

```r
plot(a ~ time, data = dat)
points(a ~ time, data = dat2, cex = 0.5, col = 'blue')
```



logaxis

Add log axis to base R plots.

logistic

The logistic function for transformations.

rbindf

Like `rbind` but data frame columns do not need to match. From monitoR package.

rounddf

Round complete data frames.

```r
dat <- data.frame(a = 1:10, b = rnorm(10), c = letters[1:10])
dat
```

```
##     a          b c
## 1   1 -0.3896677 a
## 2   2 -0.2842547 b
## 3   3  0.3740670 c
## 4   4  0.4697172 d
## 5   5  1.9896795 e
## 6   6  1.7054422 f
## 7   7  0.3225014 g
## 8   8  0.5365866 h
## 9   9  0.5526101 i
## 10 10 -0.3931458 j
```

```r
rounddf(dat)
```

```
##     a     b c
## 1   1 -0.39 a
## 2   2 -0.28 b
## 3   3  0.37 c
## 4   4  0.47 d
## 5   5  1.99 e
## 6   6  1.71 f
## 7   7  0.32 g
## 8   8  0.54 h
## 9   9  0.55 i
## 10 10 -0.39 j
```

```r
rounddf(dat, digits = c(0, 4))
```

```
## Warning in rounddf(dat, digits = c(0, 4)): First value in digits repeated to
## match length.
```

```
##     a       b c
## 1   1 -0.3897 a
## 2   2 -0.2843 b
## 3   3  0.3741 c
## 4   4  0.4697 d
## 5   5  1.9897 e
## 6   6  1.7054 f
## 7   7  0.3225 g
## 8   8  0.5366 h
## 9   9  0.5526 i
## 10 10 -0.3931 j
```

```r
rounddf(dat, digits = c(0, 4), func = signif)
```

```
## Warning in rounddf(dat, digits = c(0, 4), func = signif): First value in digits
## repeated to match length.
```

```
##     a       b c
## 1   1 -0.3897 a
## 2   2 -0.2843 b
## 3   3  0.3741 c
## 4   4  0.4697 d
## 5   5  1.9900 e
## 6   6  1.7050 f
## 7   7  0.3225 g
## 8   8  0.5366 h
## 9   9  0.5526 i
## 10 10 -0.3931 j
```

```
rounddf(dat, digits = c(2, 2), func = signif)
```

```
## Warning in rounddf(dat, digits = c(2, 2), func = signif): First value in digits
## repeated to match length.
```

```
##     a     b c
## 1   1 -0.39 a
## 2   2 -0.28 b
## 3   3  0.37 c
## 4   4  0.47 d
## 5   5  2.00 e
## 6   6  1.70 f
## 7   7  0.32 g
## 8   8  0.54 h
## 9   9  0.55 i
## 10 10 -0.39 j
```