Demonstrations

Sasha D. Hafner

03 October, 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.

```
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`.

```
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`.

```
args(interpm)
```

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

```
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 -0.658322571 -0.24070080  0.82282002
## 2     2  0.985199923 -2.21241540  0.39210169
## 3     3  0.442369634  0.65035686 -1.10430110
## 4     4  0.583489374 -2.61997746 -1.36512002
## 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 -0.348785408  0.84713689 -0.11743105
## 12   12 -0.536996083 -0.88099528  0.26263513
## 13   13 -0.007889072  0.59540020 -1.41427977
## 14   14  1.286765802  0.38394401 -0.95988356
## 15   15  0.158508059  0.09446546  0.10420799
## 16   16  0.272114543  0.04283546 -0.91642135
## 17   17 -0.380748167  0.68755413 -0.46927474
## 18   18  1.091602060 -1.71390421  0.52764192
## 19   19  2.619139535  2.02139824  1.01120909
## 20   20           NA  1.33025670  1.12288656
## 21   21           NA -1.24454810 -1.23241721
## 22   22           NA -2.27575187  1.98796992
## 23   23 -0.101694013  0.09513275 -1.38358983
## 24   24 -0.095173335 -0.76104368 -1.37941009
## 25   25  1.442112514  0.23558592 -0.74829721
## 26   26 -1.753320237  0.38345219  0.70747081
## 27   27 -0.661725658  1.52225410  0.82989784
## 28   28 -0.221754589 -0.89760176 -0.03546709
## 29   29  0.654125455  1.27112934  1.19604898
## 30   30 -0.349367852 -0.13851745  1.42337658
```

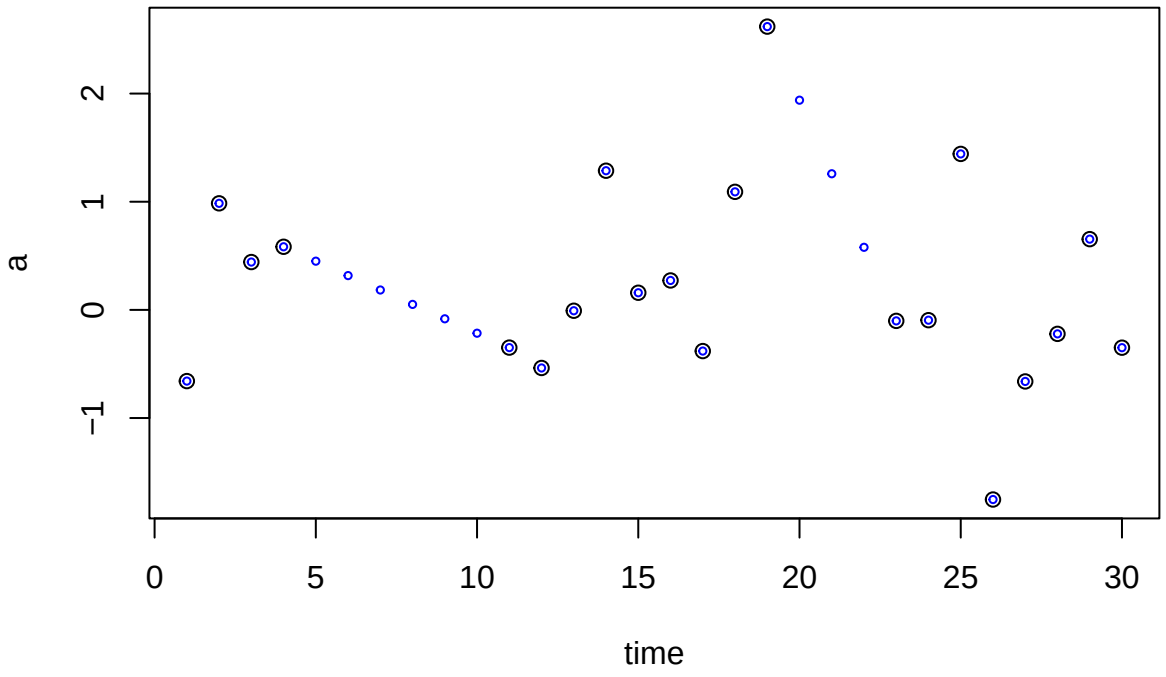
```
dat2 <- interpm(dat, 'time', c('a', 'b', 'c'))

dat2
```

```
##    time           a           b           c
## 1     1 -0.658322571 -0.24070080  0.82282002
## 2     2  0.985199923 -2.21241540  0.39210169
## 3     3  0.442369634  0.65035686 -1.10430110
## 4     4  0.583489374 -2.61997746 -1.36512002
## 5     5  0.450307262 -2.12467541 -1.18687874
## 6     6  0.317125151 -1.62937336 -1.00863746
## 7     7  0.183943039 -1.13407131 -0.83039617
## 8     8  0.050760927 -0.63876926 -0.65215489
## 9     9 -0.082421184 -0.14346721 -0.47391361
## 10   10 -0.215603296  0.35183484 -0.29567233
## 11   11 -0.348785408  0.84713689 -0.11743105
## 12   12 -0.536996083 -0.88099528  0.26263513
## 13   13 -0.007889072  0.59540020 -1.41427977
## 14   14  1.286765802  0.38394401 -0.95988356
## 15   15  0.158508059  0.09446546  0.10420799
## 16   16  0.272114543  0.04283546 -0.91642135
## 17   17 -0.380748167  0.68755413 -0.46927474
## 18   18  1.091602060 -1.71390421  0.52764192
## 19   19  2.619139535  2.02139824  1.01120909
```

```
## 20   20  1.938931148  1.33025670  1.12288656
## 21   21  1.258722761 -1.24454810 -1.23241721
## 22   22  0.578514374 -2.27575187  1.98796992
## 23   23 -0.101694013  0.09513275 -1.38358983
## 24   24 -0.095173335 -0.76104368 -1.37941009
## 25   25  1.442112514  0.23558592 -0.74829721
## 26   26 -1.753320237  0.38345219  0.70747081
## 27   27 -0.661725658  1.52225410  0.82989784
## 28   28 -0.221754589 -0.89760176 -0.03546709
## 29   29  0.654125455  1.27112934  1.19604898
## 30   30 -0.349367852 -0.13851745  1.42337658
```

```
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.

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

```
##     a           b c
## 1   1 -0.77188694 a
## 2   2  1.86589618 b
## 3   3 -0.50631729 c
## 4   4 -0.45417784 d
## 5   5  0.23283768 e
## 6   6 -0.05495233 f
## 7   7  1.25037829 g
## 8   8 -0.24655178 h
## 9   9  0.76134226 i
## 10 10 -0.03272690 j
```

```
rounddf(dat)
```

```
##     a     b c
## 1   1 -0.77 a
## 2   2  1.87 b
## 3   3 -0.51 c
## 4   4 -0.45 d
## 5   5  0.23 e
## 6   6 -0.05 f
## 7   7  1.25 g
## 8   8 -0.25 h
## 9   9  0.76 i
## 10 10 -0.03 j
```

```
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 -0.7719 a
## 2   2  1.8659 b
## 3   3 -0.5063 c
## 4   4 -0.4542 d
## 5   5  0.2328 e
## 6   6 -0.0550 f
## 7   7  1.2504 g
## 8   8 -0.2466 h
## 9   9  0.7613 i
## 10 10 -0.0327 j
```

```
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 -0.77190 a
## 2   2  1.86600 b
```

```
## 3   3 -0.50630 c
## 4   4 -0.45420 d
## 5   5  0.23280 e
## 6   6 -0.05495 f
## 7   7  1.25000 g
## 8   8 -0.24660 h
## 9   9  0.76130 i
## 10 10 -0.03273 j
```

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

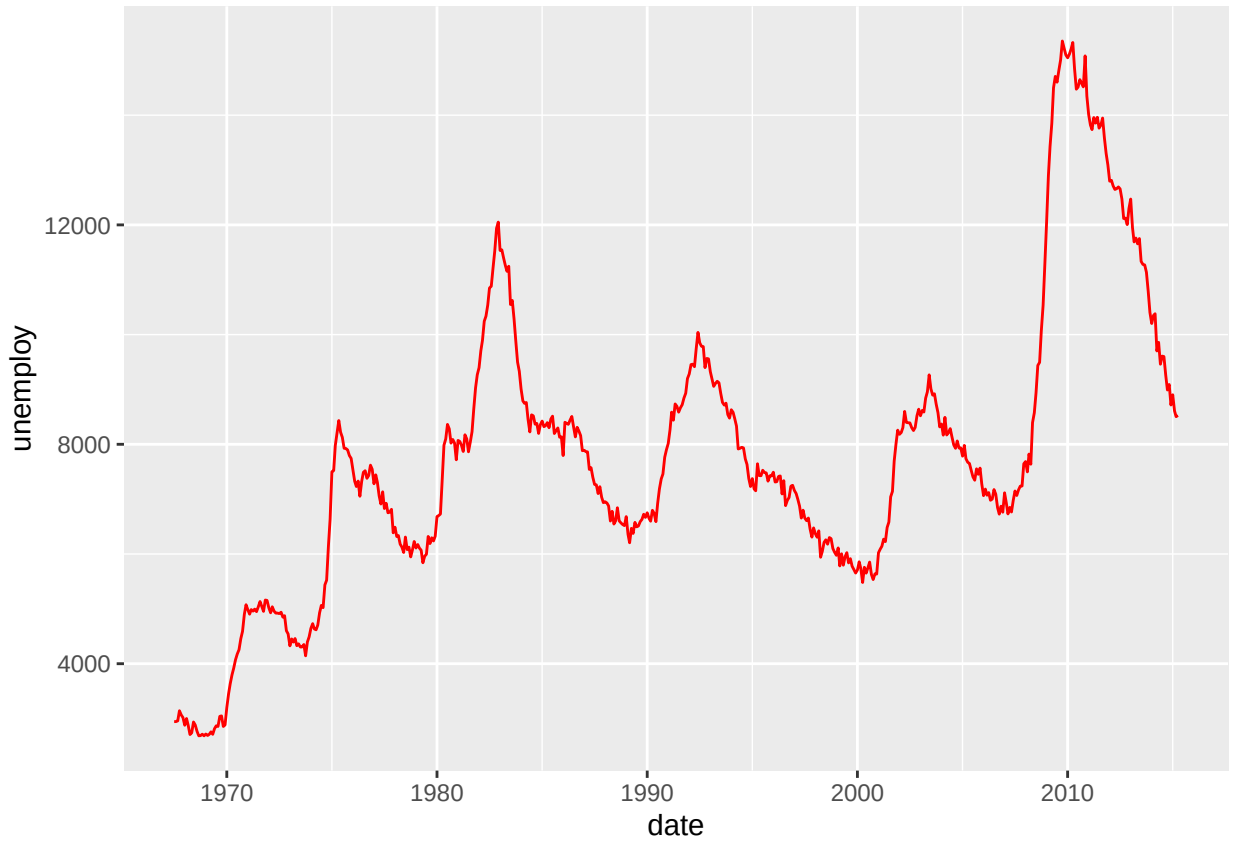
```
## Warning in rounddf(dat, digits = c(2, 2), func = signif): First value in digits
## repeated to match length.
```

```
##     a      b c
## 1   1 -0.770 a
## 2   2  1.900 b
## 3   3 -0.510 c
## 4   4 -0.450 d
## 5   5  0.230 e
## 6   6 -0.055 f
## 7   7  1.300 g
## 8   8 -0.250 h
## 9   9  0.760 i
## 10 10 -0.033 j
```

ggsave2x

Save a ggplot2 figure in one than one format.

```
ggplot(economics, aes(date, unemploy)) +
  geom_line(colour = "red")
```




```
ggsave2x('economics', width = 5, height = 5)
```

Saves png and pdf by default, add more with `type` argument. Use `...` optional arguments for more flexibility.