Advanced and Fast Data Transformation with collapse : : **CHEAT SHEET**

Basics

collapse is a powerful (C/C++ based) package supporting advanced (grouped, weighted, time series, panel data and recursive) operations in R.

It also offers a fast, class-agnostic approach to data manipulation - handling matrix and data frame based objects in a uniform, non-destructive way.

It is well integrated with *dplyr* ((grouped) tibbles), *data.table, sf* and *plm* classes for panel data, and can be programmed using pipes (`%>%`, `|>`), standard or non-standard evaluation.

Fast Statistical Functions

Fast functions to perform column–wise grouped and weighted computations on matrix-like objects:

```
fmean, fmedian, fmode, fsum, fprod, fsd,
fvar, fmin, fmax, fnth, ffirst, flast,
fnobs, fndistinct
```

Syntax:

```
FUN(x, g = NULL, [w = NULL], TRA = NULL,
    [na.rm = TRUE], use.g.names = TRUE,
    [drop = TRUE])
```

`x` – vector, matrix, or (grouped) data frame
`g` – [optional]: (list of) vectors / factors or `GRP()` object
`w` – [optional]: vector of weights
`TRA` – [optional]: operation to transform data with computed statistics (can also be done in post, see section below)

Examples:

```
fmean(data[3:5], data$grp1, data$weights)
data %>% fgroup_by(grp1) %>% fmean(weights)
```

Using dplyr grouped tibble & centering on the median:

```
data %>% dplyr::group_by(grp1) %>%
  fmedian(weights, TRA = "-")
```

Transform by (Grouped) Replacing or Sweeping out Statistics

```
TRA(x, STATS, FUN = '-', g = NULL)
```

STATS – statistics matching columns of x (e.g. aggregated matrix or data frame)
`FUN` – string indicating transformation to perform:
`'replace_fill'` – overwrite values with statistic
`'replace'` – same but keep missing values in data,
`'-'` – center, `'-+'` – center on overall average statistic,
`'/'` – scale / divide , `'%'` – percentages, `'+'` – add,
`'*'` – multiply, `'%%'` – modulus, `'-%%'` – flatten

Examples:

```
TRA(mat, fmedian(mat, g), "-", g)
fmedian(mat, g, TRA = "-")  – same thing
```

Advanced Transformations

Fast functions to perform common and specialized data transformations (for panel data econometrics)

Scaling, (Quasi-)Centering and Averaging

```
fscale(x, g = NULL, w = NULL, na.rm = TRUE,
       mean = 0, sd = 1, ...)
fwithin(x, g = NULL, w = NULL, na.rm = TRUE,
        mean = 0, theta = 1, ...)
fbetween(x, g = NULL, w = NULL, na.rm = TRUE,
         fill = FALSE, ...)
```

High-Dimensional Centering and Averaging

```
fhdwithin(x, fl, w = NULL, na.rm = TRUE,
          variable.wise = FALSE, ...)
fhdbetween(x, fl, w = NULL, na.rm = TRUE,
       fill = FALSE, variable.wise = FALSE, )
```

Operators (function shortcuts with extra features):

```
STD(), W(), B(), HDW(), HDB()
```

Linear Models

```
flm(y, X, w = NULL, add.icpt = FALSE, method =
c('lm','solve','qr','arma','chol','eigen'), )
```
– fast (barebones) linear model fitting with 6 different solvers

```
fFtest(y, exc, X = NULL, w = NULL, ...)
```
– fast F-test of exclusion restrictions for lm's (with HD FE)

Time Series and Panel Series

Fast functions to perform time-based computations on (unordered) time series and (unbalanced) panels

Lags / Leads, Differences and Growth Rates

```
flag(x, n = 1, g = NULL, t = NULL, fill = NA, )
fdiff(x, n = 1, diff = 1, g = NULL, t = NULL,
      fill = NA, log = FALSE, rho = 1, ...)
fgrowth(x, n = 1, diff = 1, g = NULL, t = NULL,
        fill = NA, logdiff = FALSE,
        scale = 100, power = 1, ...)
```

Operators: `L()`, `F()`, `D()`, `Dlog()`, `G()`
Cumulative Sums: `fcumsum(x, g, o, na.rm, fill, )`

Panel-ACF/PACF/CCF | Panel-Data → Array

```
psacf(), pspacf(), psccf()  |  psmat()
```

Other Computations

Apply functions to rows or columns (by groups)

```
dapply(x, FUN, ..., MARGIN = 2)
```
– column/row apply
`BY(x, g, FUN, ...)` – split-apply-combine computing

Advanced Data Aggregation

Fast multi-data-type, multi-function, weighted, parallelized and fully customized data aggregation

```
collap(data, by, FUN = fmean, catFUN = fmode,
       cols = NULL, w = NULL, wFUN = fsum,
       custom = NULL, ...)
```

Where:

`by` – one- or two-sided formula ([vars] ~ groups) or data (like `g`)
`FUN` – (list of) functions applied to numeric columns in data
`catFUN` – (list of) functions applied to categorical columns
`cols` – [optional]: columns to aggregate (if `by` is one-sided)
`w` – [optional]: one-sided formula or vector giving weights
`wFUN` – (list of) functions to aggregate weights passed to `w`
`custom` – [alternatively]: list mapping functions to columns e.g. `list(fmean = 1:3, fsum = 4:5, ...)`

Examples:

```
collap(data, var1 + var2 ~ grp1 + grp2)
collap(data, ~ grp1, fmedian, w = ~ weights)
```

collapg supports grouped data frames and NS eval:

```
data %>% gby(grp1) %>% collapg(w = weights)
```

Grouping and Ordering

Optimized functions for grouping, ordering, unique values, and for creating and interacting factors

`GRP(data, ~ grp1 + grp2)` – create a grouping object (class 'GRP') from grp1 and grp2 – can be passed to `g` argument – useful for programming and C/C++ development

`fgroup_by(data, grp1, grp2)` – attach 'GRP' object to `data` – a flexible grouped data frame that preserves the attributes of `data` and supports fast computations

`fgroup_vars(), fungroup()` – get group vars & ungroup

`qF(), qG()` – quick conversion to factor and vector grouping object (a factor-light class 'qG')

`groupid()` – fast run-length-type group id (class 'qG')

`seqid()` – group-id from integer-sequences (class 'qG')

`radixorder[v]()` – fast Radix-based ordering

`finteraction()` – fast factor interactions

`fdroplevels()` – fast removal of unused factor levels

`funique()` – fast unique values / rows (by `cols`)

Quick Conversions

`qDF(), qDT(), qTBL(),` – convert vectors, arrays, data.frames or lists to data.frame, data.table or tibble

`qM()` – to matrix, `m[r/c]tl()` – matrix rows/cols to list

`as_numeric_factor(), as_character_factor()` – convert factors or all factors in a list / data.frame

Fast Data Manipulation

`fselect[<-]()` – select/replace cols
`fsubset()` – subset data (rows and cols)
`colorder[v]()` – reorder cols ('v FUN's aid programming)
`roworder[v]()` – sort (reorder) rows
`[f/set]transform[v][<-]()` – transform cols (by reference)
`fcompute[v]()` – compute new cols dropping existing ones
`[f/set]rename()` – rename (any object with 'names' attr.)
`get_vars[<-]()` – select/replace cols (standard evaluation)
`num_vars[<-](), cat_vars[<-](), char_vars[<-](), fact_vars[<-](), logi_vars[<-](),`
`date_vars[<-]()` – select/replace cols by data type
`add_vars[<-]()` – add (column - bind) cols

List-Processing

Functions to process (nested) lists (of data objects)

`ldepth()` – level of nesting of list
`is_unlistable()` – is list composed of atomic objects
`has_elem()` – search if list contains certain elements
`get_elem()` – pull out elements from list / subset list
`atomic_elem[<-](), list_elem[<-]()` – get list with atomic / sub-list elements, examining only first level of list
`reg_elem(), irreg_elem()` – get full list tree leading to atomic ('regular') or non-atomic ('irregular') elements
`rsplit()` – efficient (recursive) splitting
`rapply2d()` – recursive apply to lists of data objects
`unlist2d()` – recursive row-binding to data.frame

Summary Statistics

`qsu()` – fast (grouped, weighted, panel-decomposed) summary statistics for cross-sectional and panel data
`descr()` – detailed statistical description of data.frame
`varying()` – check variation within groups (panel-id's)
`pwcor(), pwcov(), pwnobs()` – pairwise correlations, covariance and obs. (with P-value and pretty printing)

Recode and Replace Values

`recode_num(), recode_char()` – recode numeric / character values (+ regex recoding) in matrix-like objects
`replace_NA(), replace_Inf(), replace_outliers()` – replace special values `pad()` – add observations / rows.

Utility Functions

```
.c, vlabels[<-], namlab, na_rm, na_omit,
allNA, missing_cases, ckmatch, add_stub,
rm_stub, fnrow, seq_row, %!in%, unattrib etc...
```