

Split-Apply-Combine with Dynamic Grouping

Edwin de Jonge[1], Mark van der Loo[1,2]

[1] Statistics Netherlands, [2] Leiden University

2024-06-18

Contents



- ▶ Hierarchy and aggregation
- ▶ Dynamic grouping
- ▶ The accumulate R package

Split-Apply-Combine

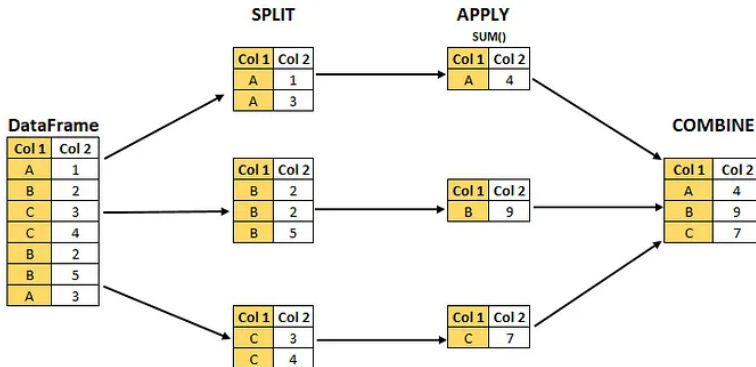
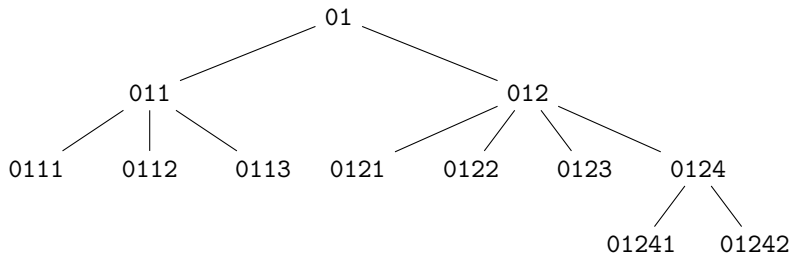


Image credit: Anurag Pandey

Dynamic grouping by NACE code



- Report the mean by the 4-digit code. If there are less than 5 records, compute the mean by using all records one level higher. Recurse until 5 records or stop and return NA.

Dynamic Grouping with Multiple variables



- ▶ Report the mean by size class $\times$ 4-digit NACE
- ▶ If there are less than 5 records (collapsing scheme)
 1. Drop size class
 2. Collapse to 3-digit NACE
 3. Collapse to 2-digit NACE
 4. Otherwise return NA.

The Accumulate package



```
library("accumulate")
```

Example data

	sbi	size	industrial	trade	other	other_income	total
1	3410	8	151722	2135	0	-1775	152082
2	2840	7	50816	NA	158	949	59876
3	2752	5	4336	NA	0	36	4959
4	3120	6	18508	NA	0	80	20682
5	2524	7	21071	0	0	442	21513
6	3410	6	24220	1069	0	239	25528

Interfaces

To specify:

what	How
Collapsing scheme	<code>formula</code> or <code>data.frame</code>
Collapse criterion	<code>function</code>
Data to aggregate	<code>base-</code> or <code>dplyr</code> -like



Interfaces



To specify:

what	How
Collapsing scheme	<code>formula</code> or <code>data.frame</code>
Collapse criterion	<code>function</code>
Data to aggregate	<code>base-</code> or <code>dplyr-like</code>

`accumulate()`

Aggregate over all non-grouping variables. Just like `base::aggregate()`.

`cumulate()`

Specify output columns. Just like `dplyr::summarize()`

accumulate()



```
a <- accumulate(producers
                 , collapse = sbi*size ~ sbi
                 , test      = function(d) nrow(d) >= 5
                 , fun       = mean, na.rm=TRUE)
head(round(a), 3)
```

	sbi	size	level	industrial	trade	other	other_income	total
1	3410	8	1	364397	2859	33	353	546117
2	2840	7	0	23160	823	49	329	25812
3	2752	5	NA	NA	NA	NA	NA	NA

accumulate()



```
producers$sbi3 <- substr(producers$sbi,1,3)
a <- accumulate(producers
                 , collapse = sbi*size ~ sbi + sbi3
                 , test      = min_records(5)
                 , fun       = mean, na.rm=TRUE)
head(round(a), 3)
```

	sbi	size	level	industrial	trade	other	other_income	total
1	3410	8	1	364397	2859	33	353	546117
2	2840	7	0	23160	823	49	329	25812
3	2752	5	2	19526	39	52	151	20603

cumulate()

```
a <- cumulate(producers
               , collapse      = sbi*size ~ sbi + sbi3
               , test          = min_records(5)
               , industrial     = median(industrial, na.rm=TRUE)
               , other_income   = mean(other_income, na.rm=TRUE))
head(round(a), 3)
```

	sbi	size	level	industrial	other_income
1	3410	8	1	97395	353
2	2840	7	0	17729	329
3	2752	5	2	9540	151



Facilities

Derive collapsing scheme from hierarchical classification

```
csd_from_digits(c("123","124"),levels=1)
```

A0 A1

1 123 12

2 124 12



Facilities

Derive collapsing scheme from hierarchical classification

```
csd_from_digits(c("123", "124"), levels=1)
```

A0 A1

1 123 12

2 124 12

Test functions

`min_records()`, `frac_complete()`, `min_complete()`

```
# use validation rules for testing
```

```
library(validate)
```

```
rules <- validator(var(x, na.rm=TRUE)<2, nrow(.) >= 5)
```

```
test <- from_validator(rules)
```



Facilities

Derive collapsing scheme from hierarchical classification

```
csh_from_digits(c("123", "124"), levels=1)
```

```
A0 A1  
1 123 12  
2 124 12
```

Test functions

```
min_records(), frac_complete(), min_complete()
```

```
# use validation rules for testing  
library(validate)  
rules <- validator(var(x, na.rm=TRUE)<2, nrow(.) >= 5)  
test <- from_validator(rules)
```

Test your test() function

```
smoke_test(data, test)
```

Computational complexity

Operation	Complexity
SAC	$\mathcal{O}(m)$
SACCG	$\mathcal{O}(nm)$

Where m is the number of groups,
 n the number of collapse levels.

Observation

Split-Apply-Combine is a formal special case of Split-Apply-Combine with Collapsing Groups. Setting the test function to TRUE reduces SACCG to SAC.

Algorithm 2: Split-Apply-Combine with Collapsing Groups: SACCG(U, ϕ, β, C)

Input : A finite set U , an aggregator $\phi : 2^U \rightarrow X$, a test function $\beta : 2^U \rightarrow \mathbb{B}$, and a collapsing sequence $C \equiv U \xrightarrow{f_1} A \xrightarrow{f_2} A_1 \xrightarrow{f_3} \dots \xrightarrow{f_n} A_n$.

Output: R : the value of ϕ for every part of U , for which a suitable collapsing group can be found, as a set of triples $(a, k, x) \in A \times \underline{n} \times X$ where $\underline{n} = \{0, 1, \dots, n\}$.

```

1  $R = \{\}$ ;
2 for  $a \in A$  do
3    $i = 0$ ;                                     // Initiate collapse level
4    $d = f^*(\{a\})$ ;                             // Get subset of  $U$ 
5   while  $i < n \wedge \neg \beta(d)$  do
6      $i = i + 1$ ;                                 // Increase collapse level
7      $d = (f^* \circ F_i^* \circ F_i)(a)$ ;             // Collapse and get subset
8   end
9   if  $i < n \vee \beta(d)$  then
10     $R = R \cup \{(a, i, \phi(d))\}$ ;
11  end
12 end

```

Summary



Results

- ▶ Formal algorithm for split-apply-combine with collapsing groups
- ▶ Implemented in `accumulate` R package (on CRAN)
- ▶ Collapsing scheme specifyable via
 - ▶ formula (based on variables in data)
 - ▶ data frame (user-defined, external collapsing scheme)
- ▶ Flexible testing of subgroups
 - ▶ Built-in functions
 - ▶ Custom function
 - ▶ Derive from `validate` rules

Thank you



markvanderloo.eu/publications.html



Journal of Statistical Software

MMMMM YYYY, Volume YY, Issue II, doi:10.18637/jss.v000.i00

Split-Apply-Combine with Dynamic Grouping

Mark P.J. van der Loo

Statistic Netherlands and University of Leiden

Abstract

Partitioning a data set by one or more of its attributes and computing an aggregate for each part is one of the most common operations in data analysis. There are two cases where the partitioning is determined dynamically by collapsing smaller subsets into larger ones, to ensure sufficient support for the computed aggregate. These two cases are not supported by software implementing split-apply-combine types of operations. This paper presents the R package *accumbate* that offers convenient interfaces for defining grouped aggregation where the grouping itself is dynamically determined, based on user-defined conditions on subsets, and a user-defined subset collapsing scheme. The formal underlying algorithm is described and analysed as well.

Keywords: data analysis, R.

1. Introduction

The operation of splitting a data set into non-overlapping groups, computing an aggregate for each group, and combining the results into a new dataset is one of the most common operations in data analysis. Indeed, any software for data analysis includes some functionality for this. For example, the combination of *split*/*apply*/*map* in R as well as *aggregate* have been a part of the S (Becker, Chambers, and Wilks 1988) and R (R Core Team 2022) languages for a long time. For R there are several packages that implement functionality for this, including *plyr* (Wickham 2011), its successor *dplyr* (Wickham, François, Henry, and Miller 2022), its drop-in replacement *poorman* (Eastwood 2022), and performance-focused R packages *collapse* (Kunst 2022) and *data.table* (Dowle and Srinivasan 2022). In Python the *pandas* package implements several methods for grouping records and aggregating over one or more columns in data frame objects. The more recent *Polars* (2023) library for Python and Rust also implements such features. Similarly, the *DataFrames.jl* package for Julia implements split-apply-combine functionality (Kornblau, White, Boucard-Vialat, paverdistribition, Gachon, Quinn, Kornblau, cjeval, Stoklos, Bates, Short, Dufala, Harris, Squires, Amkan, pde-

MPJ van der Loo (2024)
The Journal of Statistical Software
(Accepted for publication)