

Data aggregation

Dr. Jennifer (Jenny) Bryan

Department of Statistics and Michael Smith Laboratories

University of British Columbia

How to do <sthg> for various 'chunks' of your dataset

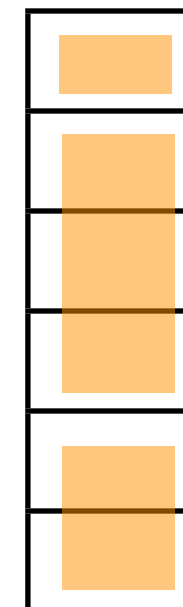
| chunks are ... | my recommendation |
|------------------------------------------------------------------|----------------------------|
| rows, columns, etc. of matrices / arrays | <code>apply()</code> |
| groups induced by levels of ≥ 1 factor(s) -- vector | <code>aggregate()</code> |
| groups induced by levels of ≥ 1 factor(s) -- data.frame | <code>plyr::ddply()</code> |
| components of a list (remember data.frames are lists!) | <code>plyr::l*ply()</code> |

How to do <sthg> for various 'chunks' of your dataset

... using only built-in functions, i.e. no `plyr`

| chunks are ... | relevant functions |
|------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------|
| rows, columns, etc. of matrices / arrays | <code>apply()</code> |
| components of a list (remember data.frames are lists!) | <code>sapply()</code> , <code>lapply()</code> |
| groups induced by levels of ≥ 1 factor(s) | <code>aggregate()</code> <code>tapply()</code> <code>by()</code> <code>split() + [sl]apply()</code> |

| | |
|------------------------------------------------------------------|-----------------------------|
| chunks are ... | my recommendation |
| rows, columns, etc. of matrices / arrays | <code>apply()</code> |
| groups induced by levels of ≥ 1 factor(s) -- vector | <code>aggregate()</code> |
| groups induced by levels of ≥ 1 factor(s) -- data.frame | <code>plyr::ddply()</code> |
| components of a list (remember data.frames are lists!) | <code>plyr::l*ply()</code> |



| | |
|-----------------------------------------------------------------|-----------------------------|
| chunks are ... | my recommendation |
| rows, columns, etc. of matrices / arrays | <code>apply()</code> |
| groups induced by levels of ≥ 1 factor(s) -- vector | <code>aggregate()</code> |
| groups induced by levels of ≥ 1 factor(s) -- data.frame | <code>plyr::ddply()</code> |
| components of a list (remember data.frames are lists!) | <code>plyr::l*ply()</code> |

| | | |
|--|--|--|
| | | |
| | | |
| | | |
| | | |
| | | |

The plyr package is what I advise long-term for data aggregation.



The screenshot shows a web browser window with several tabs. The active tab is titled 'plyr' and displays the website for the 'plyr' package. The website has a light gray background with a subtle texture. At the top, the word 'plyr' is written in a large, bold, serif font, followed by the tagline 'The split-apply-combine strategy for R'. To the right of this text is a large image of a pair of yellow-handled pliers. Below the main heading, there is a paragraph explaining the package's purpose: 'plyr is a set of tools for a common set of problems: you need to **split** up a big data structure into homogeneous pieces, **apply** a function to each piece and then **combine** all the results back together. For example, you might want to:'. This is followed by a bulleted list of three use cases: 'fit the same model to subsets of a data frame', 'quickly calculate summary statistics for each group', and 'perform group-wise transformations like scaling or standardising'. Below the list, a paragraph states: 'It's already possible to do this with base R functions (like split and the apply family of functions), but plyr makes it all a bit easier with:'. This is followed by another bulleted list of six features: 'totally consistent names, arguments and outputs', 'convenient parallelisation through the foreach package', 'input from and output to data.frames, matrices and lists', 'progress bars to keep track of long running operations', 'built-in error recovery, and informative error messages', and 'labels that are maintained across all transformations'. At the bottom of the page, a line of text reads: 'Considerable effort has been put into making plyr fast and memory efficient, and in many...'. On the right side of the page, there are two sections: 'News' and 'Learning more'. The 'News' section has a list of three links: 'Plyr 1.7', 'Plyr 1.6', and 'Plyr 1.5'. The 'Learning more' section has a paragraph stating: 'The best place to start is the article published in JSS: [The Split-Apply-Combine Strategy for Data Analysis](#).' followed by another paragraph: 'You might also find [the notes](#) from a tutorial I offered at User! 2009 useful.' and a final paragraph: 'You are welcome to ask plyr questions on R-help, but if you'd like to participate in a more focussed mailing list, please sign up for the manipulatr mailing list:'.

plyr

<http://plyr.had.co.nz>

plyr

The split-apply-combine strategy for R

plyr is a set of tools for a common set of problems: you need to **split** up a big data structure into homogeneous pieces, **apply** a function to each piece and then **combine** all the results back together. For example, you might want to:

- fit the same model to subsets of a data frame
- quickly calculate summary statistics for each group
- perform group-wise transformations like scaling or standardising

It's already possible to do this with base R functions (like split and the apply family of functions), but plyr makes it all a bit easier with:

- totally consistent names, arguments and outputs
- convenient parallelisation through the foreach package
- input from and output to data.frames, matrices and lists
- progress bars to keep track of long running operations
- built-in error recovery, and informative error messages
- labels that are maintained across all transformations

Considerable effort has been put into making plyr fast and memory efficient, and in many

News

- [Plyr 1.7](#)
- [Plyr 1.6](#)
- [Plyr 1.5](#)

Learning more

The best place to start is the article published in JSS: [The Split-Apply-Combine Strategy for Data Analysis](#).

You might also find [the notes](#) from a tutorial I offered at User! 2009 useful.

You are welcome to ask plyr questions on R-help, but if you'd like to participate in a more focussed mailing list, please sign up for the manipulatr mailing list:

Hadley Wickham.

The split-apply-combine strategy for data analysis.

Journal of Statistical Software, vol. 40, no. 1, pp. 1–29, 2011.

<http://www.jstatsoft.org/v40/i01/paper>



Journal of Statistical Software

April 2011, Volume 40, Issue 1.

<http://www.jstatsoft.org/>

The Split-Apply-Combine Strategy for Data Analysis

Hadley Wickham
Rice University

Abstract

Many data analysis problems involve the application of a split-apply-combine strategy, where you break up a big problem into manageable pieces, operate on each piece independently and then put all the pieces back together. This insight gives rise to a new R package that allows you to smoothly apply this strategy, without having to worry about the type of structure in which your data is stored.

The paper includes two case studies showing how these insights make it easier to work with batting records for veteran baseball players and a large 3d array of spatio-temporal ozone measurements.

Keywords: R, apply, split, data analysis.

split apply combine

| <i>Input</i> | <i>Output</i> | | | |
|--------------|---------------|------------|-------|-----------|
| | Array | Data frame | List | Discarded |
| Array | aaply | adply | alply | a_ply |
| Data frame | daply | ddply | dlply | d_ply |
| List | laply | ldply | llply | l_ply |

- `a*ply(.data, .margins, .fun, ..., .progress = "none")`
- `d*ply(.data, .variables, .fun, ..., .progress = "none")`
- `l*ply(.data, .fun, ..., .progress = "none")`

Take this
data.frame ...

divide it into baby
data.frames, based
on this factor
and ...

apply this
function to
each chunk ...



`ddply(.data, .variables, .fun = NULL)`

... glue the results back together and
return as a data.frame