

plyr vs. base:

about 1/10th of the ways to skin this particular cat

Matt Parker

2012-06-19

Code:

<https://github.com/mmparker/plyrvsbase>

Special thanks to DRUG's sponsors!



NEPTUNE AND COMPANY, INC.

Improving the quality of environmental decision making

<http://www.neptuneandco.com/index.php/services/statistical-consulting>

REVOLUTION
ANALYTICS

<http://www.revolutionanalytics.com/>

split-apply-combine

- Groupwise operations are extremely common:
 - Summaries of groups
 - Models built on subsets
 - Working with repeated measures

split-apply-combine in base R

- Loops are okay, y'all!
- *apply family is more canonical
 - lapply, sapply, tapply, mapply, rapply...
- Functions built on *apply:
 - by(), aggregate(), etc.

plyr

A package written by Hadley Wickham to standardize SAC operation syntax

A plyr function:

```
ddply(.data = x,  
      .var = "group",  
      .fun = function(x) ...)
```

plyr anatomy

```
ddply(.data = x,  
      .var = "group",  
      .fun = function(x) ...)
```

plyr anatomy

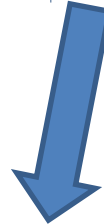
The type of object going in (here, a data.frame)



```
ddply(.data = x,  
      .var = "group",  
      .fun = function(x) ...)
```

plyr anatomy

The type of object going in (here, a data.frame)



ddp_ly(.data = x,

.var = “group”,

.fun = function(x) ...)

plyr anatomy

The type of object you hope to get out (another data.frame)



```
ddply(.data = x,  
      .var = "group",  
      .fun = function(x) ...)
```

plyr inputs and outputs

- d: a data.frame
- l: a list
- a: an array
- _: The Abyss

plyr anatomy

ddply(.data = x,

Variable(s)
for
grouping



.var = “group”,

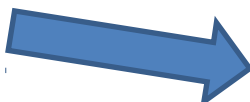
.fun = function(x),

...)


plyr anatomy

```
ddply(.data = x,  
      .var = "group",  
      .fun = function(x),  
      ...)
```

Function to
apply to
each chunk



Additional
arguments to
that function



Common plyr functions

- `ddply`: split a `data.frame`, return a `data.frame`
 - Common for summarizing across subgroups
- `ldply`: iterate over a list and return a `df`
 - Useful for converting list results into `data.frames`
- `dlply`: split a `data.frame` and return a list
 - Useful for functions that return objects that aren't easily coerced into `data.frames`
- `d_ply`: split a `data.frame`, return nothing
 - ... so you'd better have side effects

Code time:

<https://github.com/mmparker/plyrvsbase>

More to Consider

- `r*ply`: repeats a function
 - analogous to `replicate()`; useful for simulation
- `m*ply`: takes in a matrix or data.frame of arguments and iterates over each
 - Like base's `mapply()`
- Whole slew of useful plyr functions:
 - `count`, `arrange`, `join`, `vagggregate`, `rename`, `mutate`

More to Consider

- `data.table`
 - A package/data structure that speeds up many table operations – aggregating, joins, etc.
- Parallel performance
 - split-apply-combine ops are naturally parallel; both base and plyr functions can hook into a variety of parallelization packages (snow, Rmpi, foreach, segue, RHIPE)
- External code calls
 - C, C++, Fortran...
 - Potential for major efficiencies, but outside my experience