plyr

BISC 888–1 Simon Fraser University
2013-10-09

Sean Anderson
sean@seananderson.ca

why?

1. it's everywhere
2. less code, simple syntax
3. it runs faster

look familiar?

```
> d
   year  count
1 2000     16
2 2000      4
3 2000     12
4 2001     15
5 2001      7
6 2001     12
7 2002     20
...
```

```
> d
  year  count
1 2000     16
2 2000      4
3 2000     12
4 2001     15
5 2001      7
6 2001     12
7 2002     20
...
```

```
  year     mean
1 2000 10.66667
2 2001 11.33333
3 2002 13.66667
```

Warning

2 (intentionally)

scary slides ahead!

```r
d.split <- split(d, d$year)
results <- vector("list", length =
  length(d.split))

for(i in 1:length(d.split)) {
  temp <- d.split[[i]]
  temp.mean <- mean(temp$count)
  results[[i]] <- data.frame(
    year = unique(temp$year),
    mean = temp.mean)
  }

do.call("rbind", results)
```

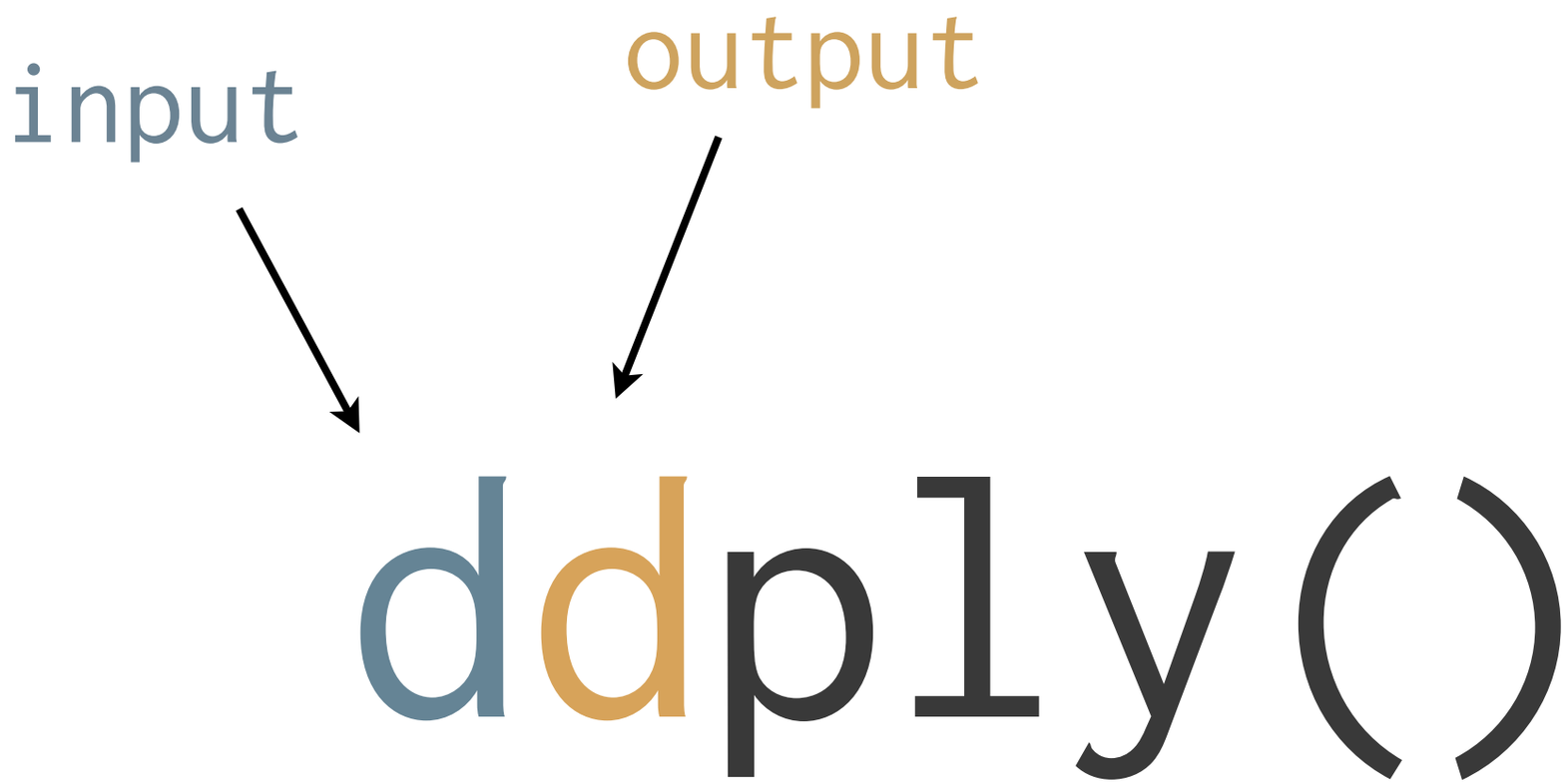
inspired by Hadley Wickham:
http://had.co.nz/plyr/

```
apply(array, 1 or 2, func)
sapply(vector, func)
lapply(list, func)
tapply(vector, index, func)
aggregate(object, by, func)
...
```

enter plyr

Hadley

```
ddply(d, "year", summarize,
      mean = mean(count))
```


input
output
ddply()

d - data frame

l - list

a - array

_ - discard

```
ddply(data, "split", function)
```

```
ddply(d, "year", summarise,
      mean.count = mean(count))
```

```
  year     mean
1 2000 10.66667
2 2001 11.33333
3 2002 13.66667
```

```
ddply(d, "year", transform,
  total.count = sum(count))
```

	year	count	total
1	2000	16	32
2	2000	4	32
3	2000	12	32
4	2001	15	34
5	2001	7	34
6	2001	12	34
7	2002	20	41
8	2002	15	41
9	2002	6	41

```
library(doParallel)
registerDoParallel(cores = 2)
ddply(..., .parallel = TRUE))
```

remember

1. it's everywhere
2. less code, simple syntax
3. it runs faster (sometimes)