plyr

Wrap up

Hadley Wickham

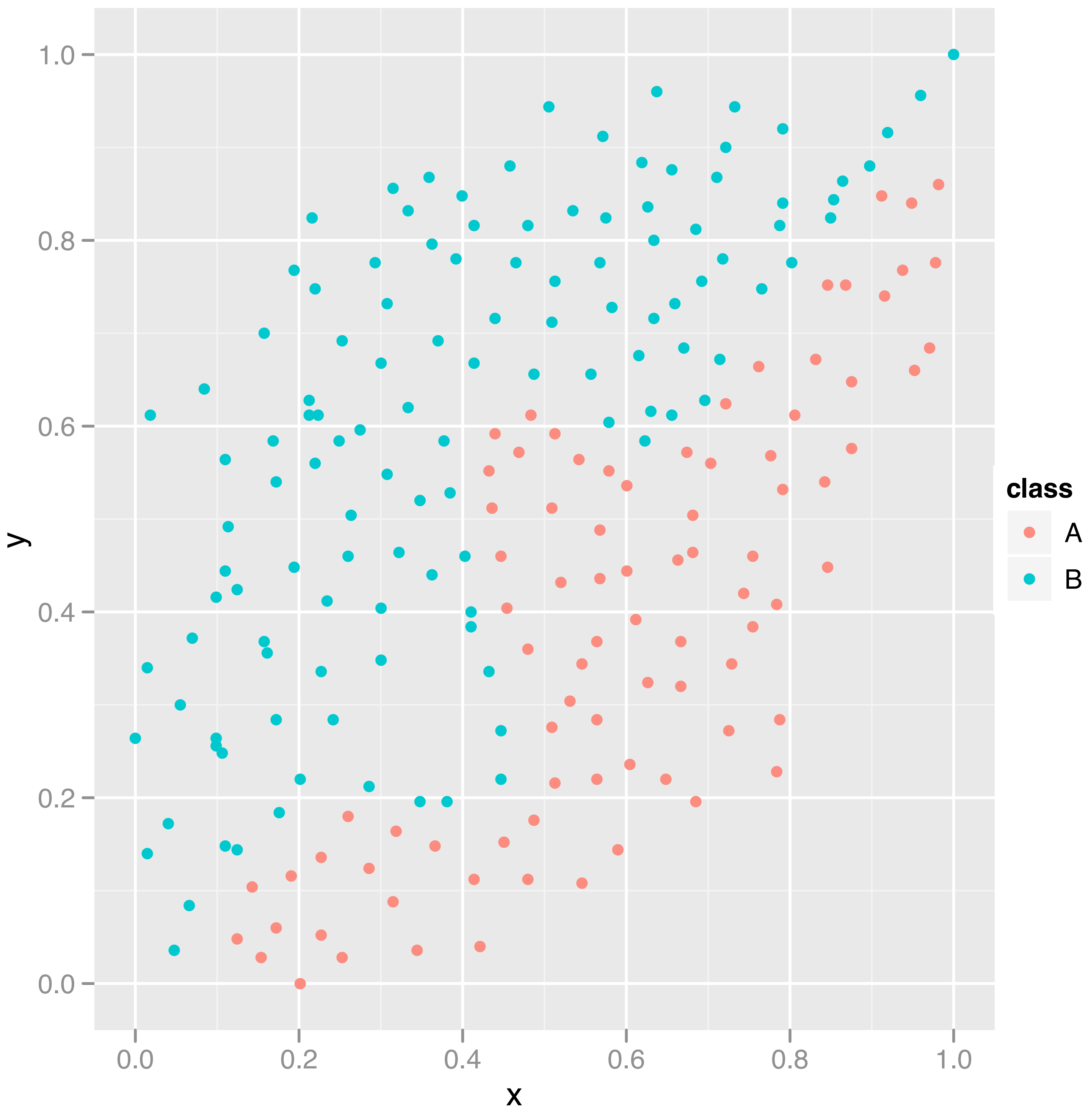
1. Fitting multiple models to the same data
2. Reporting progress & dealing with errors
3. Overall structure & correspondence to base R functions
4. Plans
5. Feedback

Multiple models

May need to fit multiple models to the same data, with varying parameters or many random starts.

Two plyr functions make this easy: `rlply` & `mlply`

Example: fitting a neural network


1.0
0.8
0.6
0.4
0.2
0.0
y
0.0
0.2
0.4
0.6
0.8
1.0
x
class
A
B

```r
library(nnet)
library(ggplot2)

w <- read.csv("wiggly.csv")
qplot(x, y, data = w, colour = class)

accuracy <- function(mod, true) {
  pred <- factor(predict(mod, type = "class"),
    levels = levels(true))
  tb <- table(pred, true)
  sum(diag(tb)) / sum(tb)
}

nnet(class ~ x + y, data = w, size = 3)
```

rlply

A little different to the other plyr functions: first argument is number of times to run, second argument is an **expression** (not a function).

Automatically adds run number (.n) to labels.

```
models <- rlply(50, nnet(class ~ x + y, data = w,
size = 3, trace = FALSE))

accdf <- ldply(models, "accuracy", true = w$class)
accdf
qplot(accuracy, data = accdf, binwith = 0.02)
```

mlply

What if we want to systematically vary the input parameters?

`mlply` allows us to vary all of the arguments to the applicator function, not just the first argument

Input is a data frame of parameter values

```
wiggly_nnet <- function(...) {
  nnet(class ~ x + y, data = w, trace = FALSE, ...)
}
rlply(5, wiggly_nnet(size = 3))

# Unfortunately need 2+ parameters because of bug
opts <- data.frame(size = 1:10, maxiter = 50)
opts

models <- mlply(opts, wiggly_nnet)
ldply(models, "accuracy", true = w$class)

# expand.grid() useful if you want to explore
# all combinations
```

Progress & errors

Progress bars

Things seem to take much less time when you are regularly updated on their progress.

Plyr provides convenient method for doing this: all arguments accept `.progress = "text"` argument

Error handling

Helper function: `failwith`

Takes a function as an input and returns a function as output, but instead of throwing an error it will return the value you specify.

```
failwith(NULL, lm)(cty ~ displ, data = mpg)

failwith(NULL, lm)(cty ~ displ, data = NULL)
```

Overall structure

	array	data frame	list	nothing
array	aaply	adply	alply	a_ply
data frame	daply	**ddply**	**dlply**	d_ply
list	laply	**ldply**	llply	l_ply
n replicates	raply	rdply	**rlply**	r_ply
function arguments	maply	mdply	**mlply**	m_ply

No output

Useful for functions called purely for their side effects: write.table, save, graphics.

If `.print = TRUE` will print each result (particularly useful lattice and ggplot2 graphics)

Your turn

With your partner, using your collective R knowledge, come up with all of the functions in base R (or contributed packages) that do the same thing.

	array	data frame	list	nothing
array	apply	adply	alply	a_ply
data frame	daply	*aggregate*	by	d_ply
list	sapply	ldply	lapply	l_ply
n replicates	replicate	rdply	replicate	r_ply
function arguments	mapply	mdply	mapply	m_ply

Plans

Deal better with large and larger data: trivial parallelisation & on-disk data (sql etc)

Stay tuned for details.

http://hadley.wufoo.com/forms/course-evaluation/