ggplot2: visualizing our data and IO

Peter D Smits

Committee on Evolutionary Biology
University of Chicago

January 24, 2013

Introduction

Last time, we covered the basics of ggplot2 and how we modify plots.

Today's talk will be in two parts.

1. introduction to maps and ggplot
2. Sample workflow from IO through munging to visualizing.

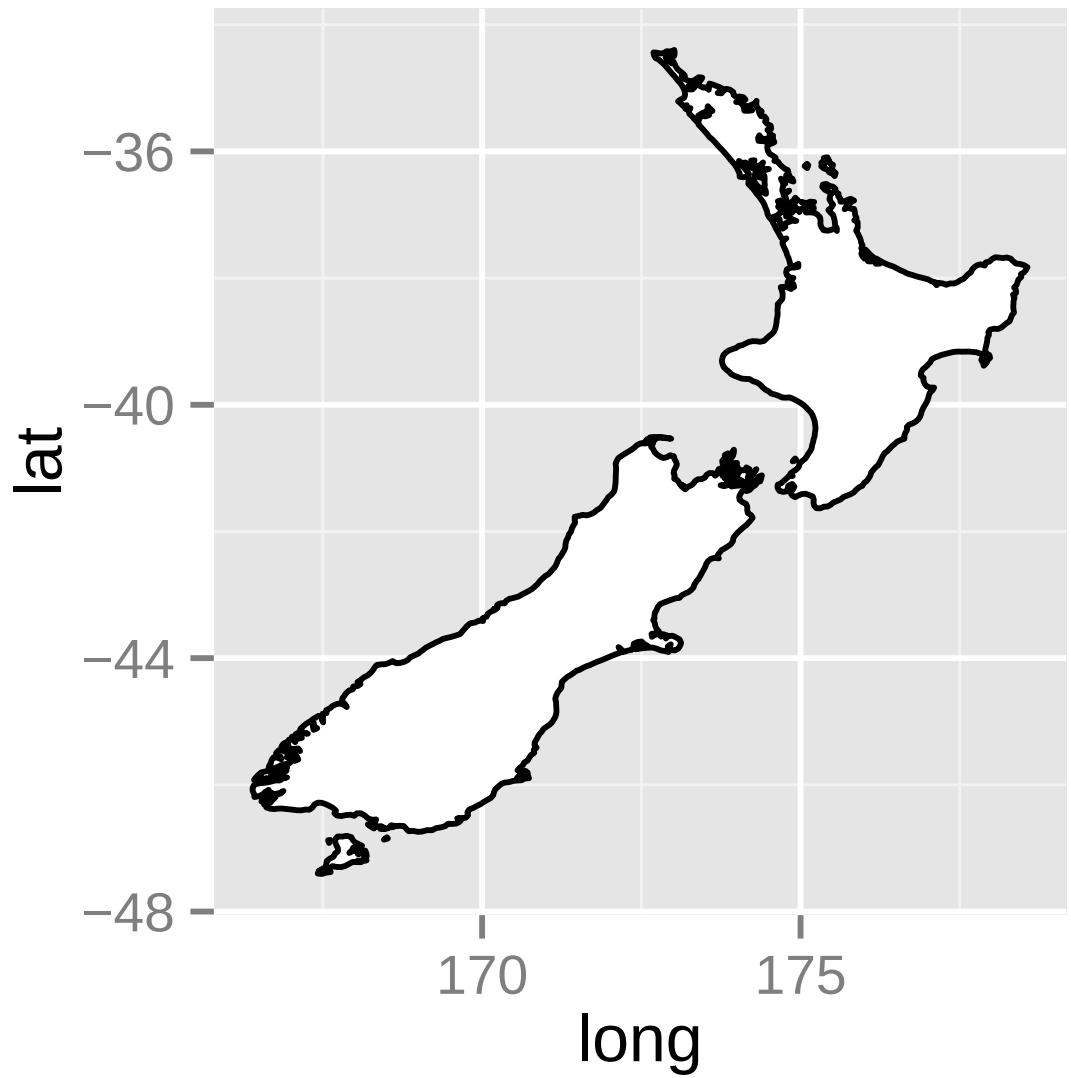
Maps

```
require(ggplot2)
library(maps)
nz <- map_data('nz')
```

	long	lat	group	order	region	subregion
1	172.74	-34.44	1.00	1	North.Island	
2	172.80	-34.46	1.00	2	North.Island	
3	172.85	-34.45	1.00	3	North.Island	
4	172.90	-34.42	1.00	4	North.Island	
5	172.96	-34.43	1.00	5	North.Island	
6	173.02	-34.40	1.00	6	North.Island	

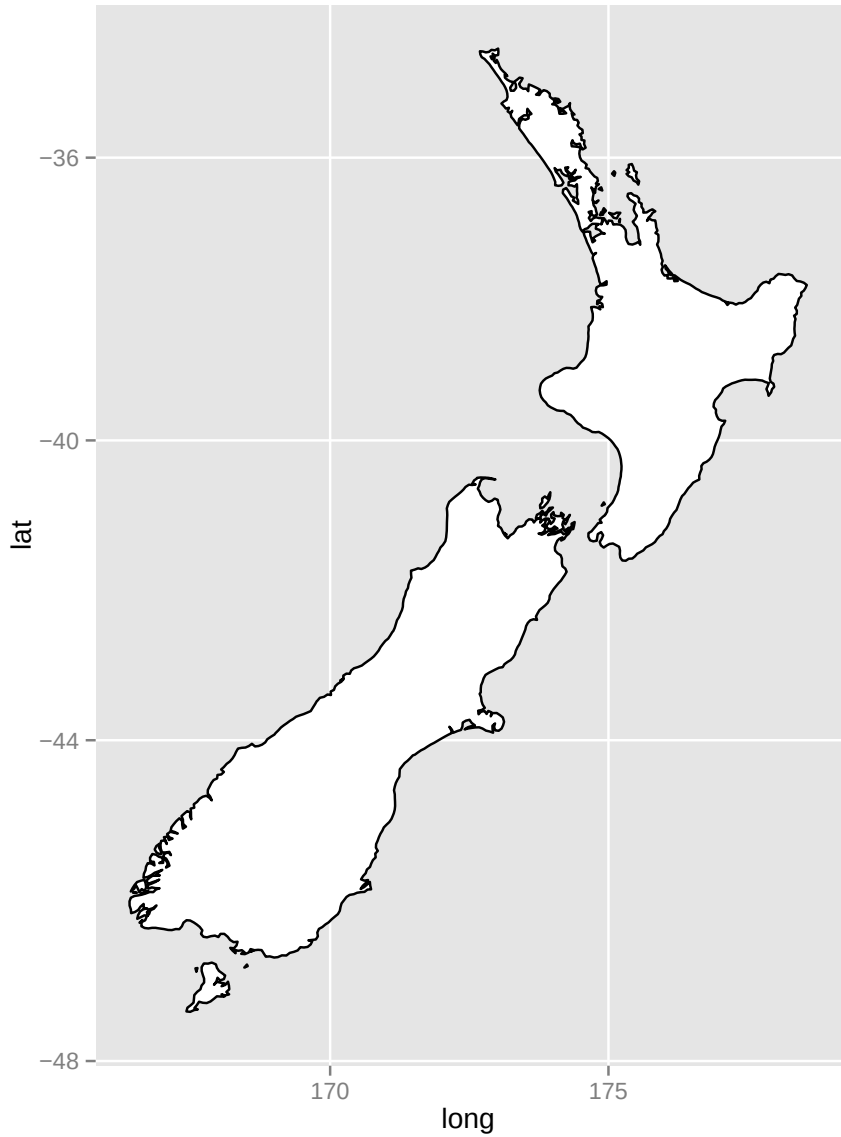
Looking at NZ: Cartesian coordinates

```
nzmap <- ggplot(nz, aes(x = long, y = lat, group = group))
nzmap <- nzmap + geom_polygon(fill = 'white', colour = 'black')
nzmap
```


−36
−40
−44
−48
lat
170
175
long

Looking at NZ: Mercator projection

```
library(mapproj)
nzmap + coord_map()
```



Looking NZ: Other projections

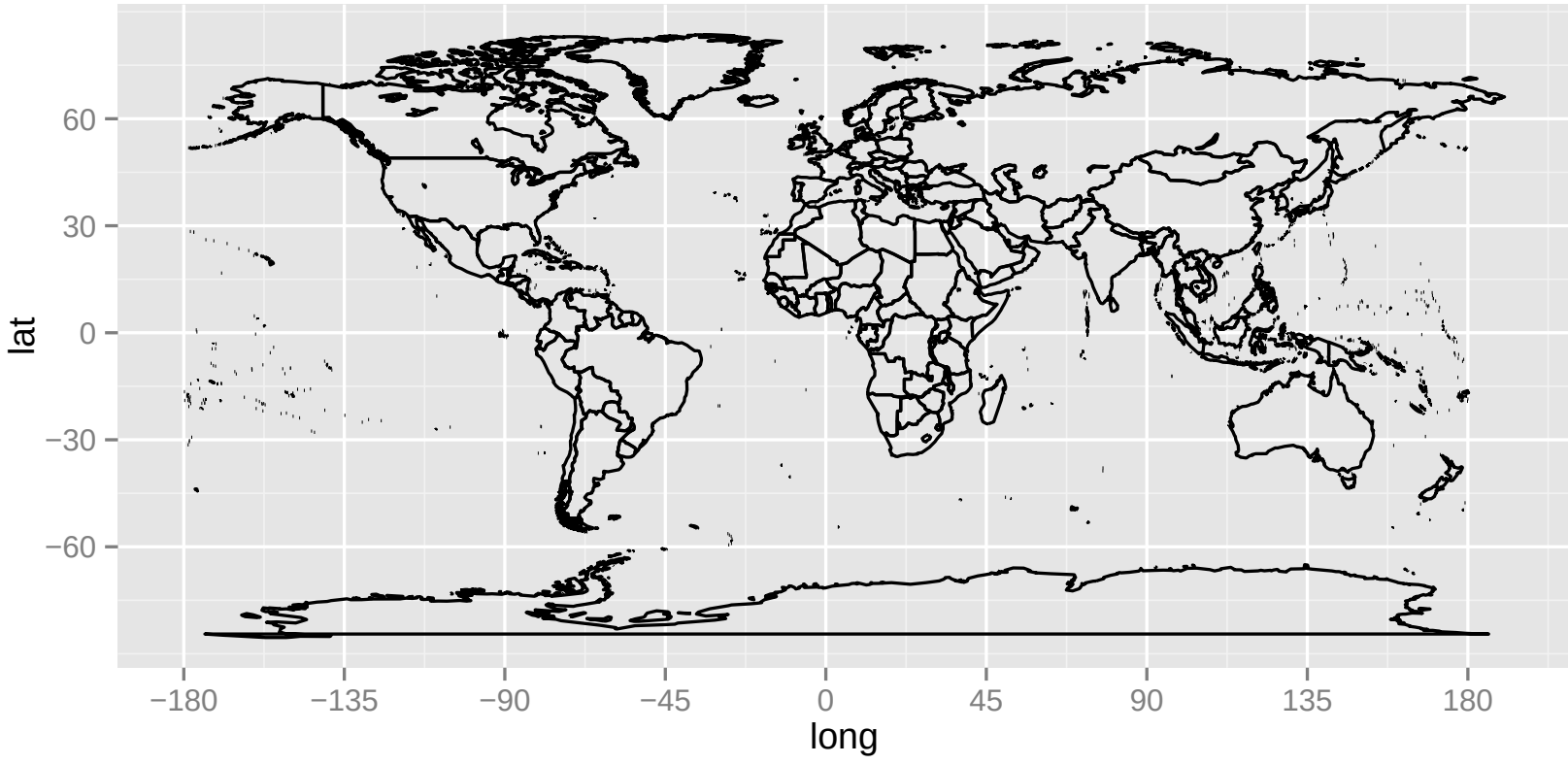
```
nzmap + coord_map('cylindrical')

nzmap + coord_map('azequalarea',
                  orientation = c(-36.92, 174.6, 0))

## ?mapproject for coordinate systems and their parameters
## there are a lot of them
```

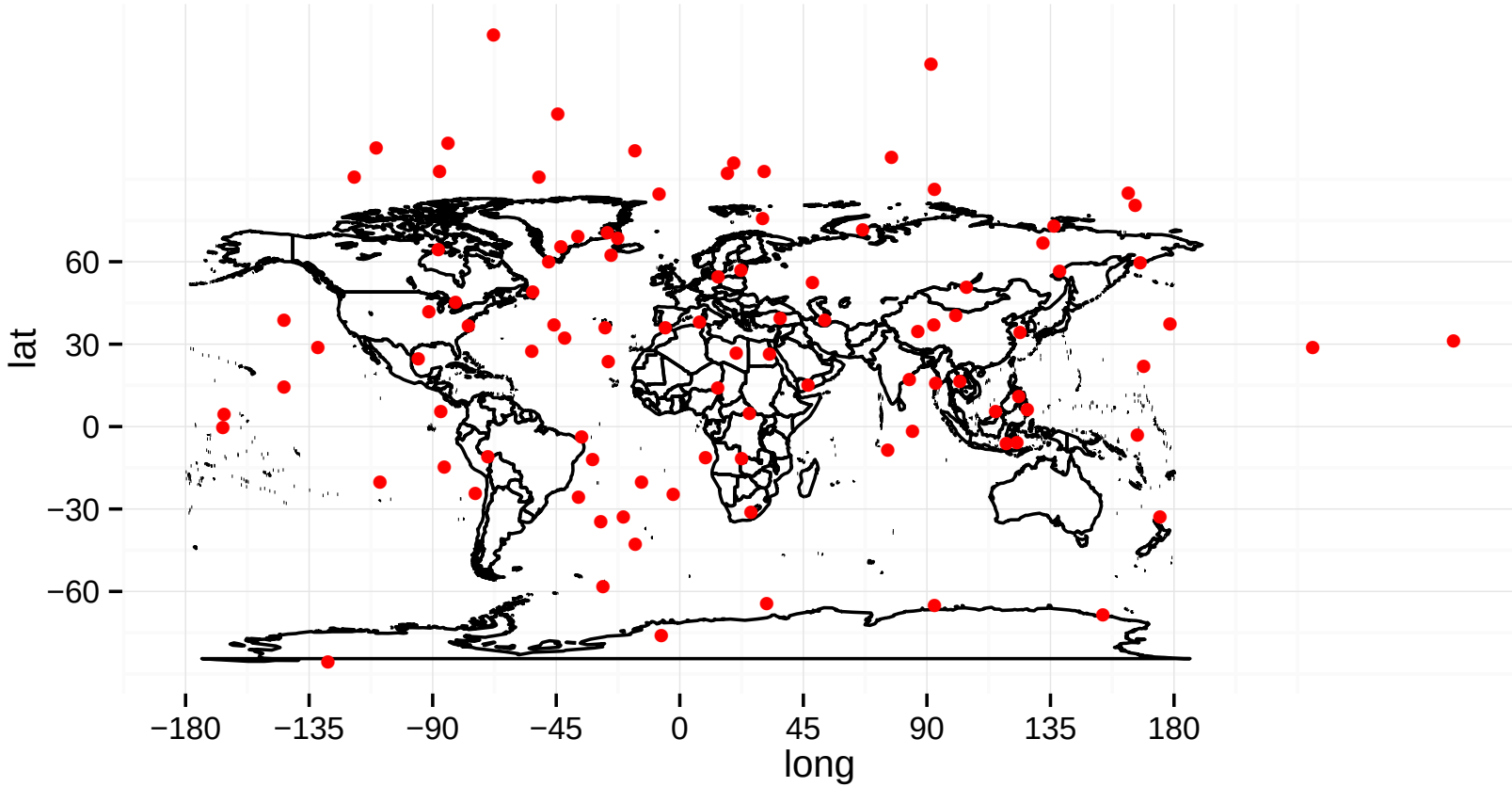
World map!

```
world <- map_data('world')
worldmap <- ggplot(world, aes(x = long, y = lat, group = group))
worldmap <- worldmap + geom_path()
worldmap <- worldmap + scale_y_continuous(breaks = (-2:2) * 30) +
            scale_x_continuous(breaks = (-4:4) * 45)
worldmap <- worldmap + coord_equal()
```


60
30
0
−30
−60
lat
−180
−135
−90
−45
0
45
90
135
180
long

Lets add random dots to it!

```
x.long <- rnorm(100, mean = mean(world$long), sd = sd(world$long))
y.lat <- rnorm(100, mean = mean(world$lat), sd = sd(world$lat))
rpoints <- data.frame(x.long, y.lat)
## there are prettier ways to do this, but they are more complicated
worldmap <- worldmap + geom_point(data = rpoints,
                                  mapping = aes(x = x.long,
                                                y = y.lat,
                                                group = NULL),
                                  colour = 'red')
worldmap <- worldmap + theme_minimal()
```


lat
60
30
0
−30
−60
−180
−135
−90
−45
0
45
90
135
180
long

More on maps

Try a lot of projections to get the one that distorts your reality the least.

You can cut down the amount of the map displayed. You can look that up on your own.

Shape files can be read into R and used as maps. (maptools)

There are packages to use google maps. (RGoogleMaps, ggmap)

IO

Input/Output: the act of getting things in and out or your program/script/what have you.

```
read.table()
read.csv()
read.tree()
read.nexus()
dget()
## etc.

write.table()
write.csv()
write.tree()
write.nexus()
dput()
save()
save.image()
ggsave() # save ggplot object
## etc.
```

Anatomy of read.table()

```
args(read.table)

## function (file, header = FALSE, sep = "", quote = "\"'", dec = ".",
##     row.names, col.names, as.is = !stringsAsFactors, na.strings = "NA",
##     colClasses = NA, nrows = -1, skip = 0, check.names = TRUE,
##     fill = !blank.lines.skip, strip.white = FALSE, blank.lines.skip = TRUE,
##     comment.char = "#", allowEscapes = FALSE, flush = FALSE,
##     stringsAsFactors = default.stringsAsFactors(), fileEncoding = "",
##     encoding = "unknown", text)
## NULL
```

returns an object of class "data.frame"

(Review of) reading in a file

```
pantheria <- read.table(file = 'pantheria_mung.csv',
                        sep = ',', header = T, row.names = 1)
```

Very rough summary

```
summary(pantheria)
```

```
##           order                     family              genus
##  Rodentia    :2277   Muridae          : 730   Crocidura   : 172
##  Chiroptera  :1116   Cricetidae       : 681   Myotis      : 103
##  Soricomorpha: 428   Vespertilionidae: 407   Rhinolophus :  77
##  Primates    : 376   Soricidae        : 376   Sorex       :  77
##  Carnivora   : 286   Sciuridae        : 278   Hipposideros:  67
##  Artiodactyla: 240   Pteropodidae     : 186   Rattus      :  66
##  (Other)     : 693   (Other)          :2758   (Other)     :4854
##       species     adultbodymass  adultforearmlen adultheadbodylen
##  australis:  15   Min.    : 0.7   Min.   :3       Min.   : 3
##  thomasi  :  15   1st Qu.: 3.2   1st Qu.:4       1st Qu.: 5
##  macrotis :  14   Median : 4.6   Median :4       Median : 5
##  major    :  14   Mean   : 5.5   Mean   :4       Mean   : 6
##  grandis  :  11   3rd Qu.: 7.2   3rd Qu.:4       3rd Qu.: 6
##  minor    :  11   Max.   :18.9   Max.   :6       Max.   :10
##  (Other)  :5336   NA's   :1874   NA's   :4513    NA's   :3475
##   basalmetrate  basalmetratemass  dietbreadth     homerange
##  Min.   : 2     Min.   : 1       Min.   :1      Min.   :    0
##  1st Qu.: 4     1st Qu.: 3       1st Qu.:1      1st Qu.:    0
##  Median : 5     Median : 5       Median :2      Median :    0
##  Mean   : 5     Mean   : 5       Mean   :3      Mean   :  234
##  3rd Qu.: 6     3rd Qu.: 7       3rd Qu.:4      3rd Qu.:    1
##  Max.   :12     Max.   :13       Max.   :8      Max.   :79245
##  NA's   :4843   NA's   :4843     NA's   :3255   NA's   :4709
##  terrestriality   weaningage
```

Summarizing

Hadley coined term "split-apply-combine" in his J. Stat. Soft. paper on "plyr".

This is essentially using higher-order functions to ease aspects of data munging.

EXTREME mungning will not be covered here (plug for other course) because it requires a lot of knowledge of the R language as an actual language. Here, we do a quick usage of the function `ddply()`

Summarize by order

```
library(plyr)
order.summary <- ddply(pantheria, .(order), summarise,
                       mean.mass = mean(adultbodymass, na.rm = TRUE),
                       mass.count = length(na.omit(adultbodymass)),
                       mean.body = mean(adultheadbodylen, na.rm = TRUE),
                       body.count = length(na.omit(adultheadbodylen)),
                       mean.metrate = mean(basalmetrate, na.rm = TRUE),
                       metrate.count = length(na.omit(basalmetrate)))
```

Summarize by order

	order	mean.mass	mass.count	mean.body	body.count	mean.metrate	metrate.count
1	Afrosoricida	3.58	39	4.76	32	4.04	13
2	Artiodactyla	10.87	210	7.16	146	9.25	12
3	Carnivora	8.66	250	6.49	241	7.99	62
4	Cetacea	13.44	76	8.48	45		0
5	Chiroptera	2.95	695	4.58	158	3.48	80
6	Cingulata	7.65	20	5.88	10	6.52	9
7	Dasyuromorphia	4.00	61	5.10	29	4.16	20
8	Dermoptera	7.07	2	5.94	2		0
9	Didelphimorphia	4.31	64	4.89	66	5.25	11
10	Diprotodontia	7.58	121	5.96	55	6.21	25
11	Erinaceomorpha	5.69	12	5.27	20	5.39	5
12	Hyracoidea	7.96	4	6.17	3	6.60	3
13	Lagomorpha	7.03	60	5.80	69	6.72	6
14	Macroscelidea	4.46	14	5.01	9	4.16	7
15	Microbiotheria	3.22	1	4.66	1		0

Reshaping

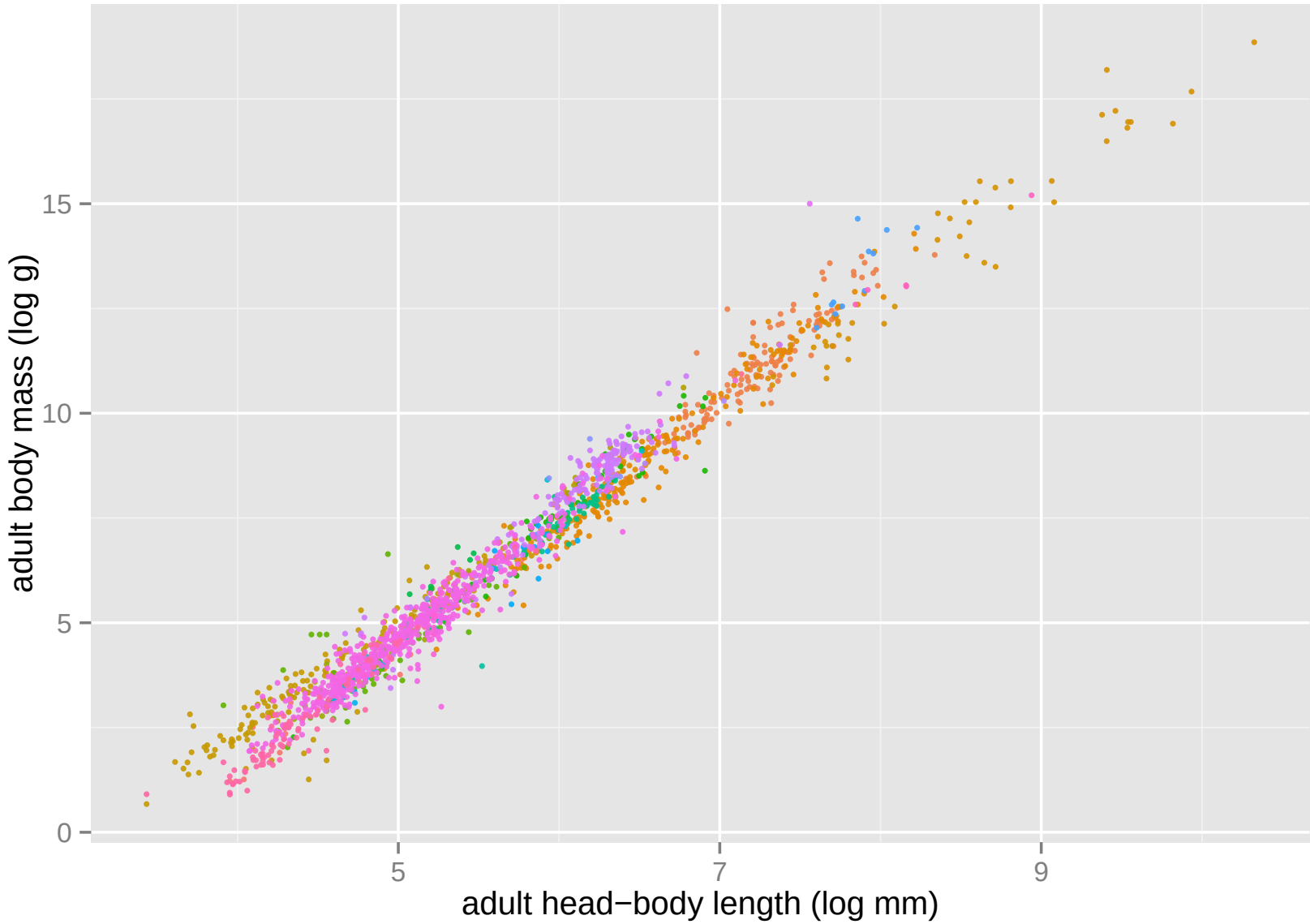
Wide versus long format. ggplot likes long, we think in wide.

```
library(reshape2)
```

Visualizing

```
require(scales)
gpan <- ggplot(pantheria, aes(x = adultheadbodylen,
                              y = adultbodymass,
                              colour = order))
gpan <- gpan + geom_point(alpha = 0.9,  size = 1)
gpan <- gpan + theme(legend.position = 'none')
gpan <- gpan + labs(x = 'adult head-body length (log mm)',
                    y = 'adult body mass (log g)')
```

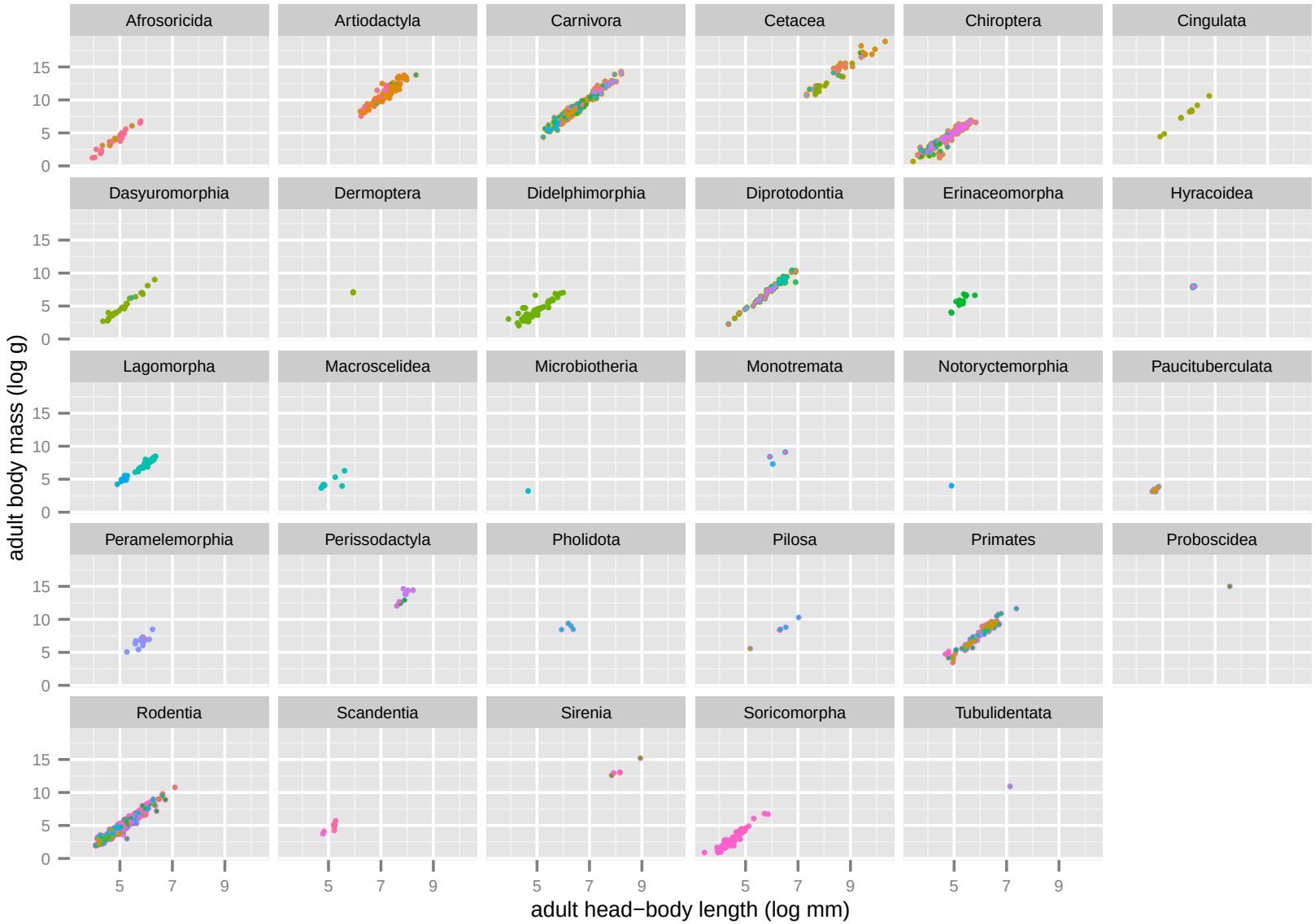
Visualizing



Visualizing

```
gpan <- gpan + geom_point(alpha = 0.9, size = 0.7,
                          mapping = aes(colour = family))
gpan <- gpan + theme(axis.text = element_text(size = 6),
                     axis.title = element_text(size = 8),
                     strip.text = element_text(size = 6))
gpan <- gpan + facet_wrap(~ order)
```

Visulizing


Afrosoricida
Artiodactyla
Carnivora
Cetacea
Chiroptera
Cingulata
Dasyuromorphia
Dermoptera
Didelphimorphia
Diprotodontia
Erinaceomorphia
Hyracoidea
Lagomorpha
Macroscelidea
Microbiotheria
Monotremata
Notoryctemorphia
Paucituberculata
Peramelemorphia
Perissodactyla
Pholidota
Pilosa
Primates
Proboscidea
Rodentia
Scandentia
Sirenia
Soricomorpha
Tubulidentata
adult body mass (log g)
adult head−body length (log mm)
0
5
10
15
5
7
9

Save our plot

If you aren't using knitr/Sweave (you should!), you can save your ggplot for use later.

```
ggsave(filename = 'pantheria_facet.png',
       plot = gpan)
```