ggplot for disease ecologists

Jennie Lavine (jslavine@umich.edu) and Ben Bolker (bolker@mcmaster.ca)

May 20, 2012

1 Introduction

ggplot2 is an R package by Hadley Wickham (Wickham, 2009), based on *The Grammar of Graphics* by Leland Wilkinson (Wilkinson, 1999), that attempts to systematize the construction of exploratory and statistical graphics by defining a system of mappings between variables in the data and "aesthetics" — characteristics of the plot such as x and y coordinates, colour, point size, line width, etc..

The basic structure of a `ggplot` command is that you define the overall plot by specifying a data set, always in the form of a data frame, a set of default mappings, and then *add* so-called "geoms" to add objects of layers to the plot.

Each layer contains:

1. a description of the 'geometric object' (`geom`) you want to produce, such as points, bars, lines, etc.
2. a statistical transformation (`stat`), which could be the binned data to make a histogram, a summary statistic such as the mean of groups with error bars, a smoothed curve from e.g. a linear model summarizing a relationship between two variables, etc.
3. a scale (`scale`) to map quantities in the data to visual aspects of the plot, such as a color scale, different shaped points for different groups, size of points determined by some characteristic in the data, etc.

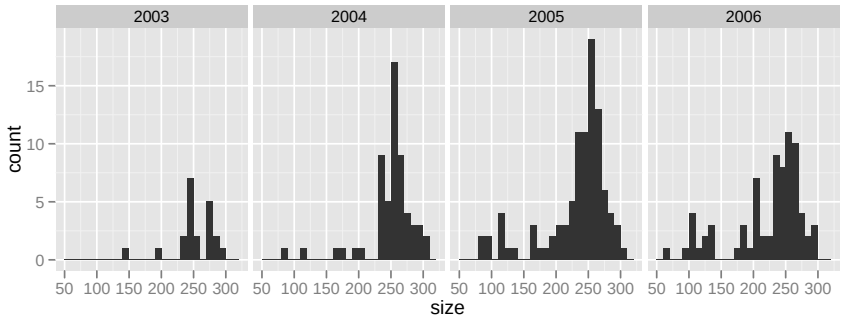
Before you begin, make sure you have the `ggplot`, `plyr`, and `reshape2` packages installed, and download the `gophertortoise.txt` and `malaria.csv` data files from `http://www.math.mcmaster.ca/bolker/eeid/private` and put them in your working directory (the username and password will be available in class).

2 `ggplot` objects

Now we look at a dataset on seroprevalence by size in tortoises. We read in data, take a quick look at the distribution of sizes, and familiarize ourselves with the concept of layers in `ggplot`. Create a `ggplot` object called `hist.size`. This object does not have any layers in it, so there is nothing to plot. You can take a look at the object by typing `names(hist.size)`. Then make a histogram by adding a `geom` to it.

```
require(ggplot2)  ## require() and library() are similar
require(plyr)
dat <- read.table("gophertortoise.txt",header=TRUE)
head(dat, 3)
##   TortID    Date YEAR size Sex    ELISA SITE status age
## 1  CF022  8/6/06 2006  269   M POSITIVE   CF      1 269
## 2  CF023  5/6/03 2003  270   M POSITIVE   CF      1 270
## 3  CF023 5/24/05 2005  268   M POSITIVE   CF      1 268
```

```
hist.size <- ggplot(dat, aes(x=size))
hist.size+geom_histogram(binwidth=10)+facet_grid(.~YEAR)
```

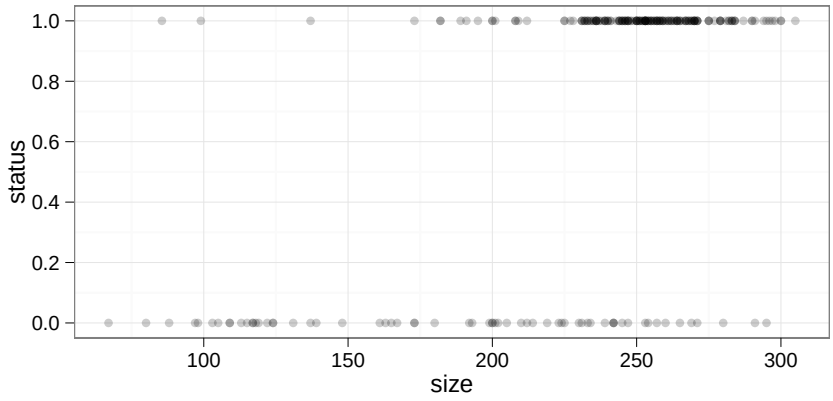


To make a histogram of densities instead of counts, add the argument `aes(y=..density..)` to `geom_histogram`.

3 Looking at binary data

Now we want to see how serological status is related to size. We 1. create a `ggplot` object called `plot1` with size mapped to the x axis and serological status to the y axis, 2. change the default theme to black and white. 3. plot the binary data as points using transparency (`alpha=0.2`) with a black and white background (because the transparency (`alpha`) is specified as a fixed value rather than depending on a variable in our data set, we put it on its own, not inside an `aes` statement).

```
plot1 <- ggplot(dat, aes(size, status))
theme_update(theme_bw())
##using transparency to show overlapping points
(plot1a <- plot1 + geom_point(alpha=0.2))
```



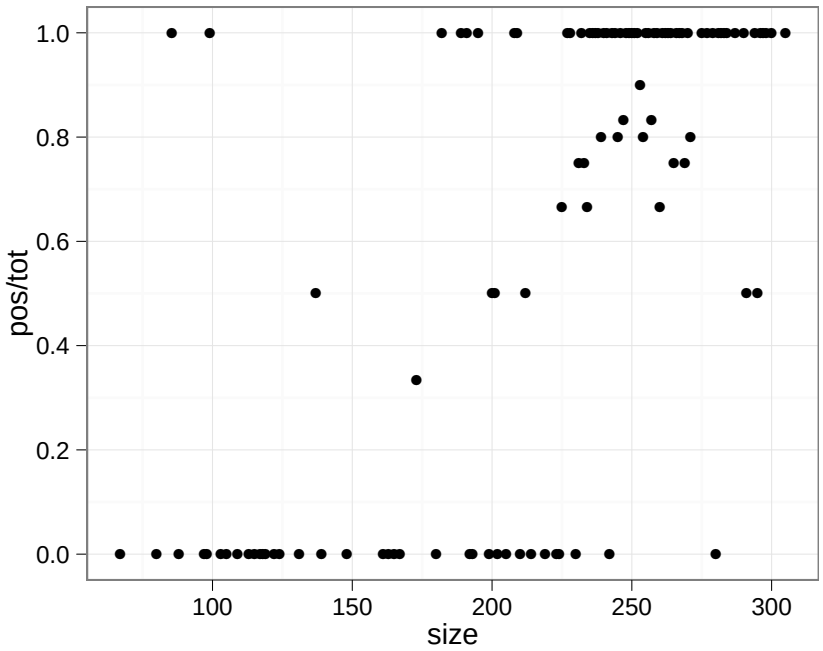
Perhaps it would be easier to look at the data as proportions rather than counts. When you download `ggplot`, you will also download two useful packages for manipulating data objects: `plyr` and `reshape2`. Here we use the function `ddply` to take the dataframe `dat`, split it by `size`, apply a function to each subset, then reshape the results into a new dataframe, which we store in the object `sizetab`. If you are familiar with `tapply`, `sapply`, etc., the input will feel familiar to you. (If not, don't worry about it too much.)

```
sizetab <- ddply(dat,"size",
                 function(x) c(tot=nrow(x),pos=sum(x$status)))
head(sizetab,3)
##   size tot pos
```

```
## 1 67.0   1    0
## 2 80.0   1    0
## 3 85.5   1    1
```

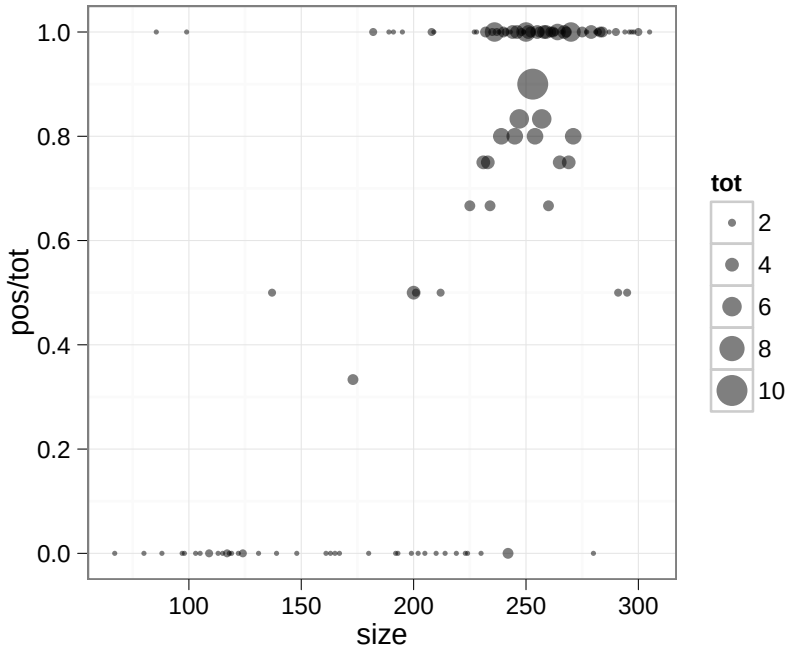
Now we render a basic plot, with just the barebones: data, an aesthetic saying what values get mapped to the x and y axes, and the type of geometric object

```
ggplot(data=sizetab,aes(x=size,y=pos/tot))+geom_point()
```



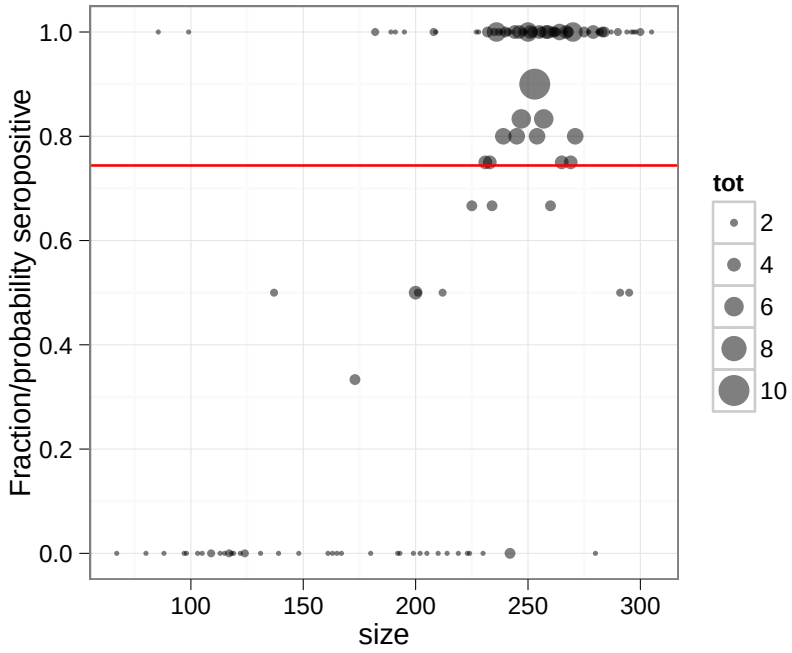
Now (1) make the size of the points scale with the number of individuals represented by each data point (`tot`) and (2) make the points partway transparent.

```
(g1 <- ggplot(sizetab,aes(x=size,y=pos/tot,size=tot))+geom_point(alpha=0.5))
```



Now we will add two more layers, one to add a horizontal line specifying the overall probability and the other to change the y labels (we use `with` as a quick way to avoid using to many `$` to refer to variables inside the `sizetab` data frame). Once again, since `yintercept` is an absolute value and not a mapping between a variable and an aesthetic, we don't put it inside `aes()`.

```
prob0 <- with(sizetab,sum(pos)/sum(tot))
g1+geom_hline(yintercept=prob0,width=2,colour=2)+
    ylab("Fraction/probability seropositive")
```

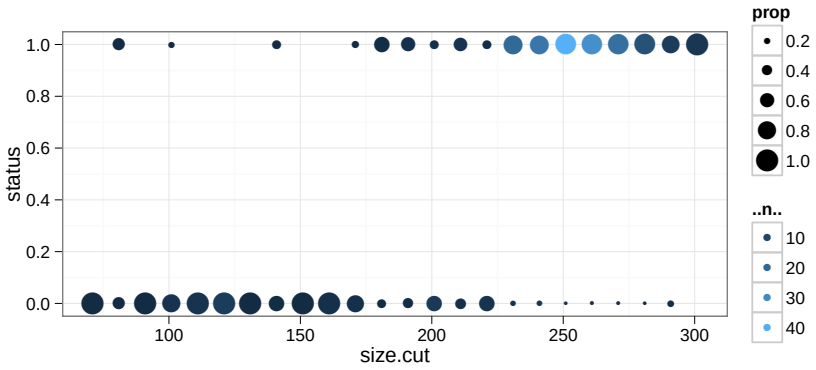


We can visualize the proportions a different way using `stat_sum`. To do this, first we bin the data into size groups

```
cut.ind <- seq(from=min(dat$size)-1, to=max(dat$size)+1, length=25)
dat$size.cut <- cut(dat$size, cut.ind,
labels=head(cut.ind,-1)+diff(cut.ind)/2)
dat$size.cut <- as.numeric(as.character(dat$size.cut))
```

Next look at the proportion of positive and negative serostatuses in each one. The size of the dot represents the proportion of tortoises in size class X with serostatus Y.

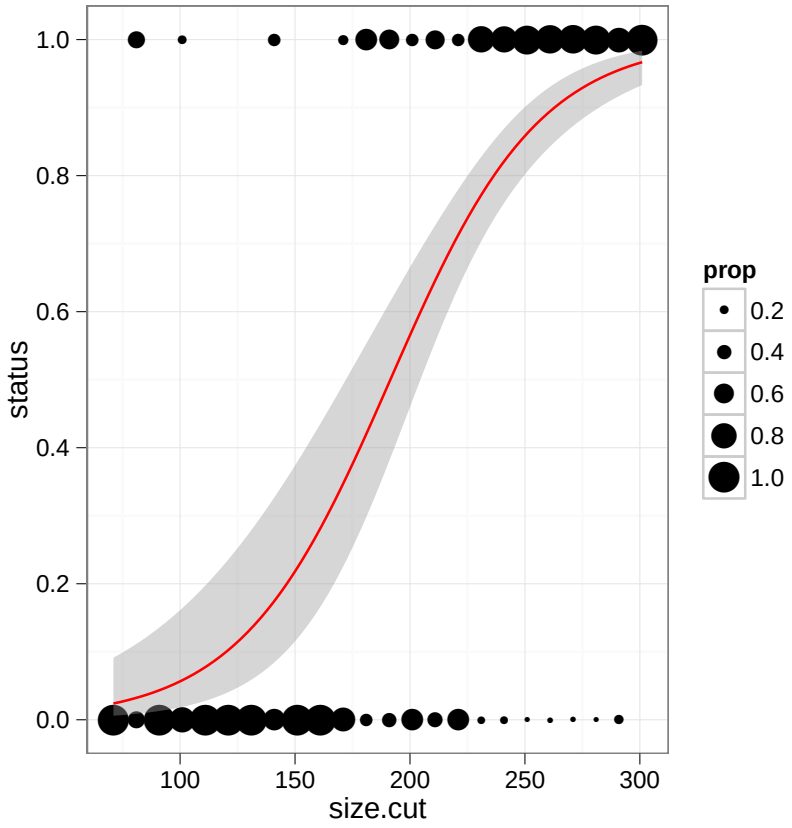
```
plot2 <- ggplot(dat, aes(size.cut, status))
plot2a <- plot2+stat_sum(aes(group=size.cut))
(plot2b <- plot2a+aes(colour=..n..))
```



Similarly to our density-based histogram above, we use the weird-looking variable `..n..` to refer to a variable that is calculated internally by `ggplot` — in this case we want to plot the number of co-occurring data values at each point, rather than the (default) proportion of values.

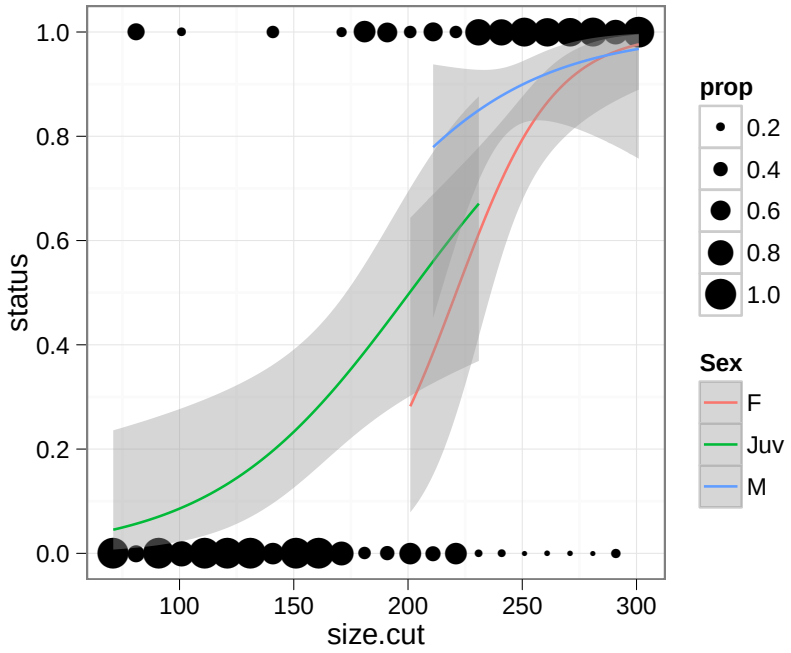
Add a layer to the plot showing the result of a logistic regression.

```
(plot2c <- plot2a +
 geom_smooth(method="glm", family="binomial",colour="red"))
```



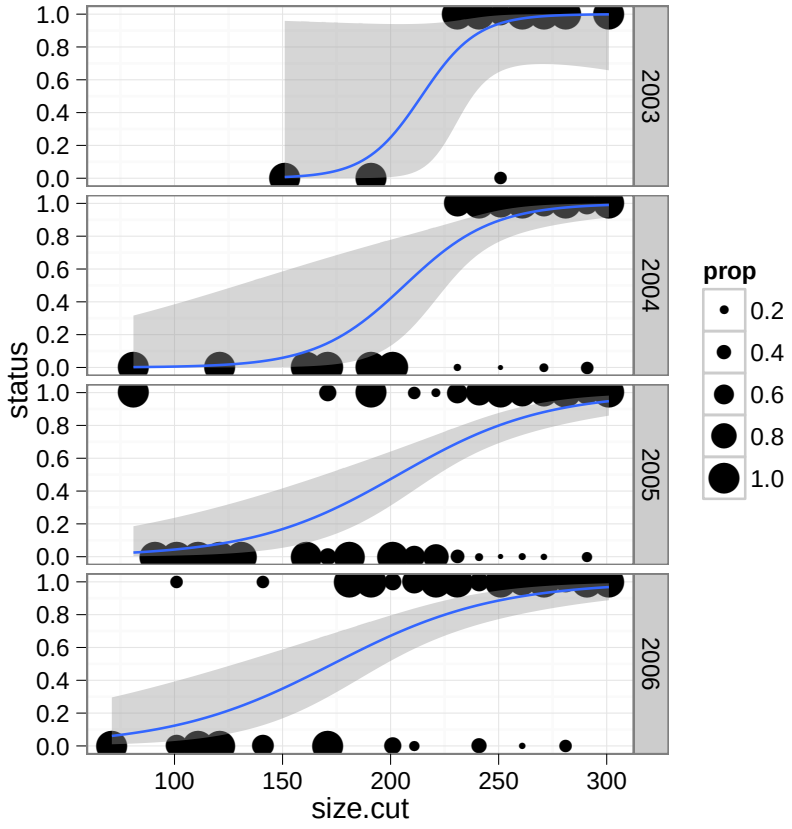
Now fit a separate curve to the data from each sex category (male, female, juvenile).

```
(plot2d <- plot2a + geom_smooth(aes(group=Sex, colour=Sex),
                                method='glm', family='binomial'))
```



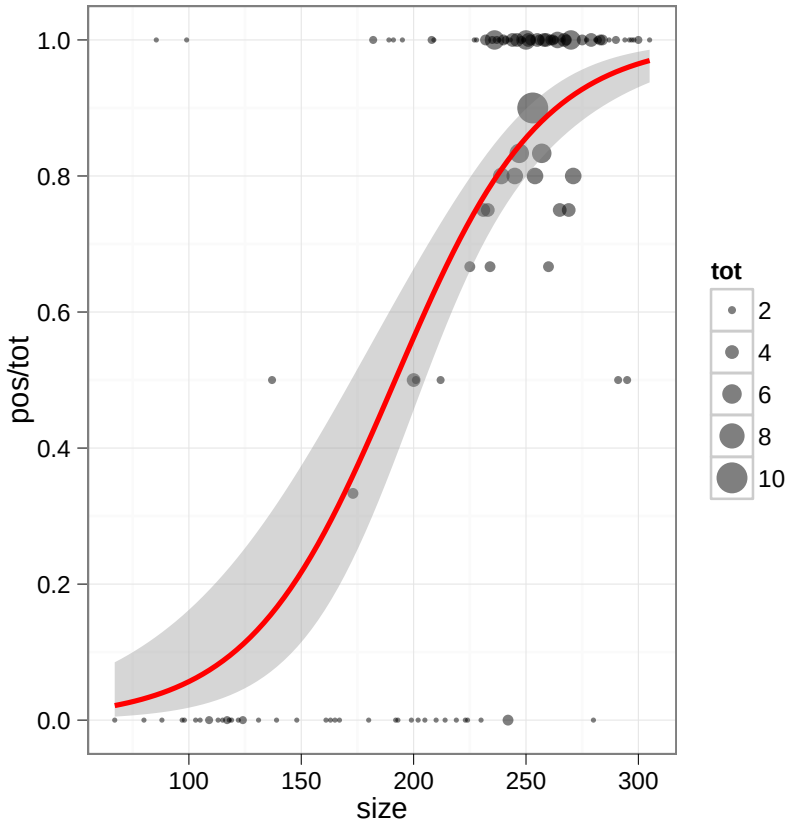
We can do the same thing by year instead of sex, but it gets very confusing — try it for yourself. A better choice may be to plot the data from and curve from each year in a separate panel, using `facet_grid`.

```
(plot2e <- plot2a +
 geom_smooth(aes(group=YEAR),method='glm', family='binomial')+
 facet_grid(YEAR~.))
```



We can add the same sort of curve to the proportion data we looked at before with the following:

```
g1+geom_smooth(method="glm",family=binomial,
               aes(weight=tot),
               size=1,
               colour="red")
```

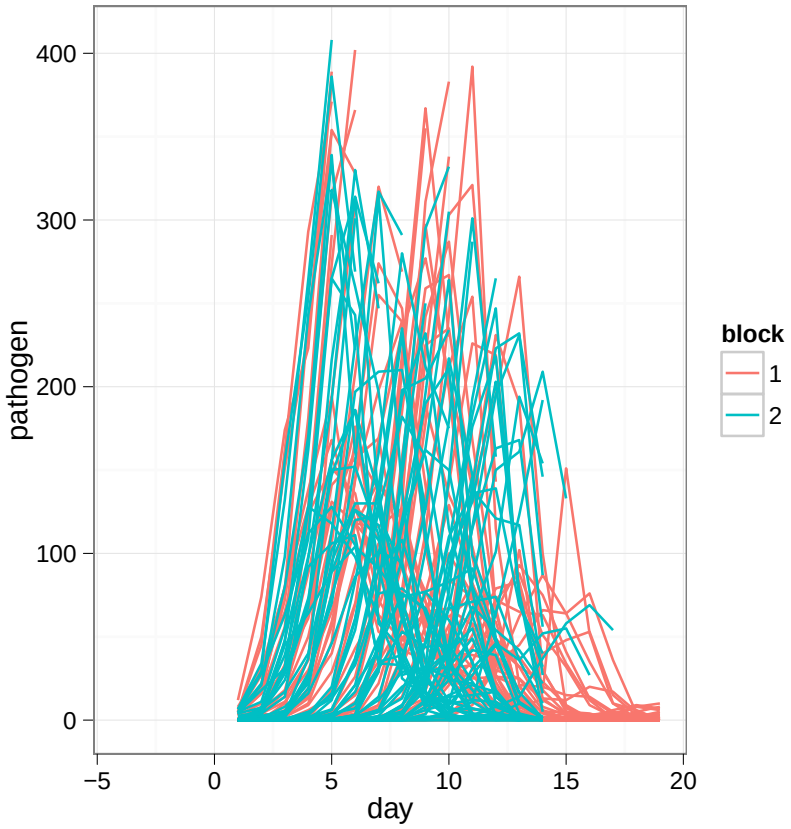


(We use `size=1` here because we previously mapped `size` to the `tot` variable, and we want to override/ignore it for the `geom_smooth` addition.)

Now we take a quick look at a very different sort of dataset. As has been mentioned, these data are not to be freely used after the workshop, but it is a very cool within-host dataset from Andrew Read's lab on malaria parasite and red blood cell densities. Again, we read in data, modify it a little and then make a first fairly simple plot. We look at the trajectory through time for each mouse, (`group=mouse`), and we differentiate two blocks of experiments by color.

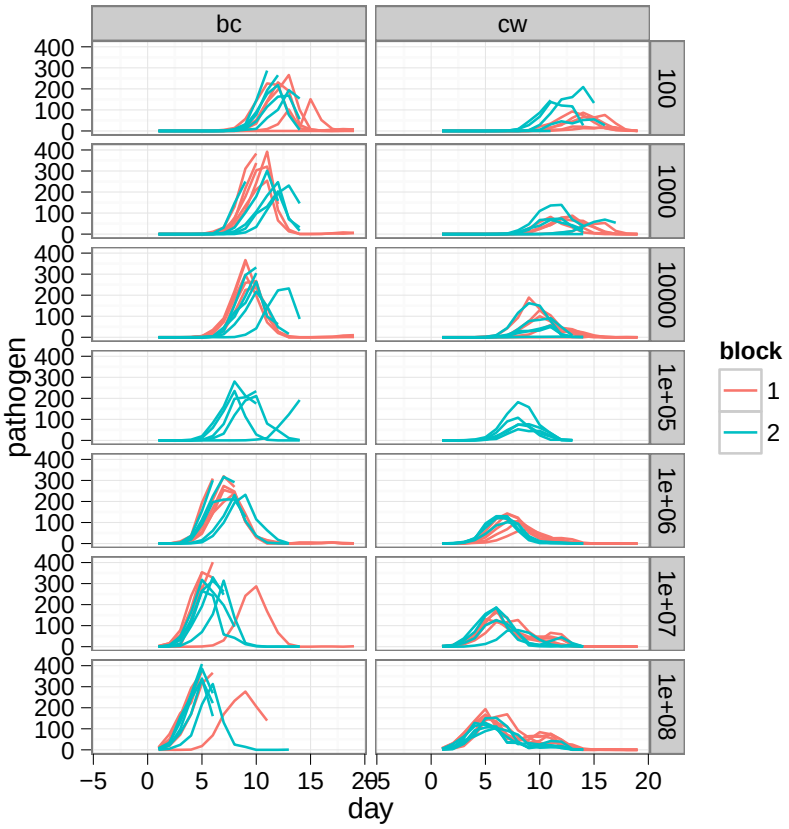
```
alldat <- read.csv('malaria.csv',
colClasses=c(rep('factor',6),rep('numeric',3)))
infec.dat <- alldat[alldat$strain!='con'&alldat$day<20,]

f=ggplot(infec.dat, aes(day, pathogen))
(f1=f+geom_line(aes(group=mouse, color=block)))
```



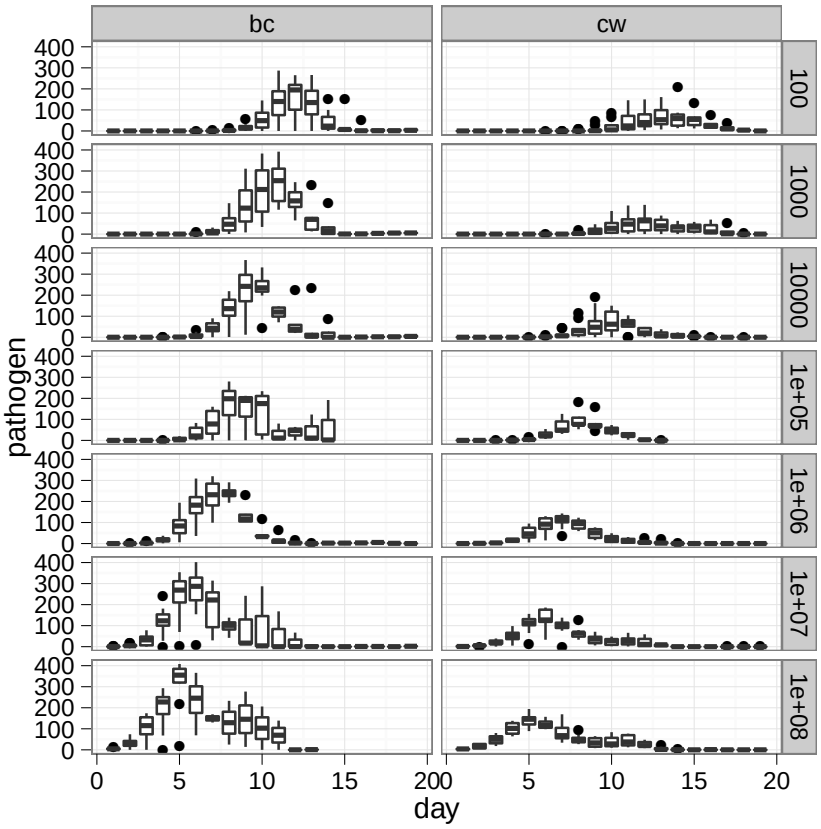
That plot is not so useful because there is so much data! We can use `facet_grid` to split up the data by dose and strain:

```
f1+facet_grid(dose~strain)
```



Finally, perhaps we are not interested in individual trajectories, but instead how the distributions change over the course of the experiment. Let's use boxplots to look at that.

```
f+ facet_grid(dose~strain)+geom_boxplot(aes(group=day))
```



(this graph may be a little slow).
Other resources:

- The ggplot2 web page has a list of the available components of `ggplot` graphs, and there is a wiki
- There is an active ggplot Google group; a lot of answered questions on StackOverflow; and a big variety of blog posts on the R bloggers site
- If you use `lattice` graphics, this link and this one are both useful
- If you get really into `ggplot` you can buy the author's book; it's a little expensive for a paperback, but hey, the software was free!

References

Wickham, H. (2009, August). *ggplot2: Elegant Graphics for Data Analysis*. Springer.
Wilkinson, L. (1999). *The grammar of graphics*. New York: Springer.