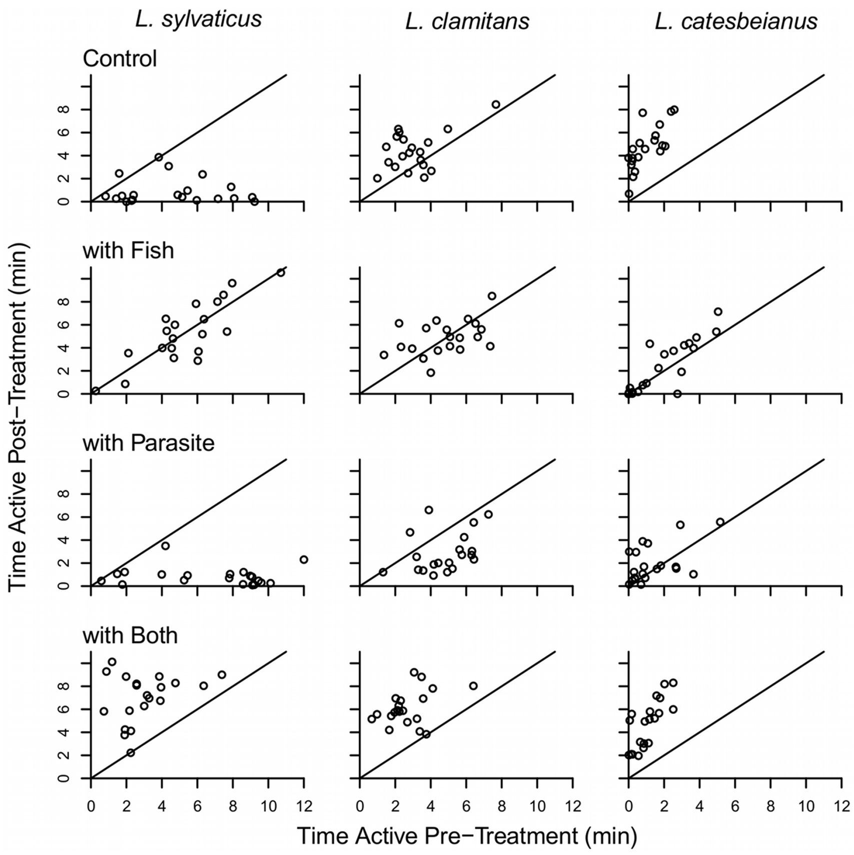
Unit 4: More Advanced Graphics

Multiple Plots on a Page

To place multiple plots on a page, you can use the `par()` option `mfrow` or `mfcol`. Plots returned subsequently will be drawn in the array specified (#row, #col) by row if you use mfrow or by column if you use mfcol. This is a quick and dirty method of getting multiple plots per page – useful for exploratory graphing or checking model assumptions.

If you need a publishable plot that includes multiple panels, you are best to use the function `layout()`. Let's step through a complex example, using a publication graph I put together.



(from: Szuroczki, D & Richardson, JML. 2012. PLOS ONE10.1371/journal.pone.0049592

Now, before I show you the code, be warned – it looks hideous! Rest assured that I also look at example code for graphs and immediately want to throw up my arms and quit. The trick is that while it looks like an overwhelming arduous process, in reality you build it up step by step and it is actually straight-forward and comes together faster than you might expect.

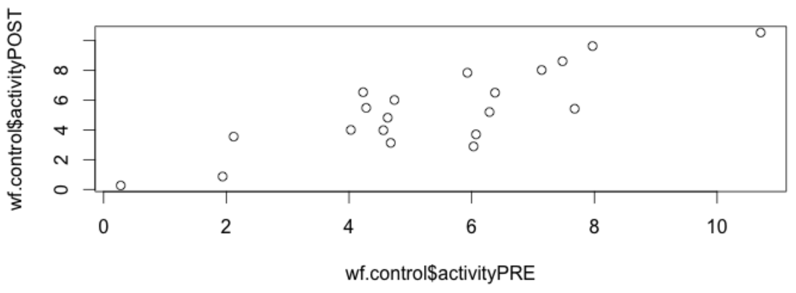
So we will spend this session, building up this plot piece by piece and by the end of it you will understand what each piece of the code below does, and how you could modify any part of the graph. Okay – prepare yourself…

I start by knowing that I want to plot activityPOST versus activityPRE for each species:treatment combination. So after reading in the data and setting the treatment variable to have the levels correctly ordered, I subset the data into each treatment combination.

```
###### IMPORT DATA & MANIPULATE ###########
dat <- read.table("Rdata.csv", sep = ",", header=T)
levels(dat$trtmt) <- c("control", "fish", "parasite", "combo", ordered=TRUE)

# Create separate vectors for each species/treatment combination

bf <- dat[which(dat$Species=="bullfrog"),]
gf <-dat[which(dat$Species=="greenfrog"),]
wf <- dat[which(dat$Species=="woodfrog"),]

wf.control <- subset(wf, trtmt=='control')
wf.fish <- subset(wf, trtmt=='fish')
wf.parasite <- subset(wf, trtmt=='parasite')
wf.both <- subset(wf, trtmt=='combo')

gf.control <- subset(gf, trtmt=='control')
gf.fish <- subset(gf, trtmt=='fish')
gf.parasite <- subset(gf, trtmt=='parasite')
gf.both <- subset(gf, trtmt=='combo')

bf.control <- subset(bf, trtmt=='control')
bf.fish <- subset(bf, trtmt=='fish')
bf.parasite <- subset(bf, trtmt=='parasite')
bf.both <- subset(bf, trtmt=='combo')
```

Next, I try making the basic plot that I want:

```
plot(wf.control$activityPRE, wf.control$activityPOST)
```

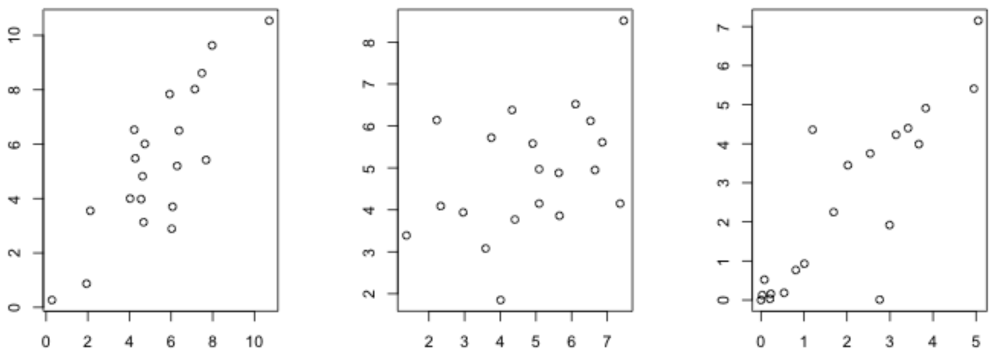


So far, so good. But I don't really want those default axes labels and since I'm not sure what labels I want I'll just leave labels off for now. (Below I highlight the code changes using **bold**.)

```
plot(wf.control$activityPRE, wf.control$activityPOST, xlab="", ylab="")
```

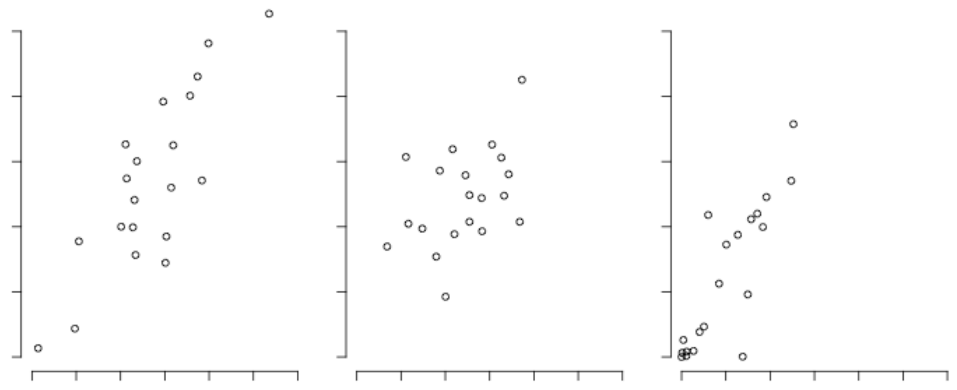
And let's also try putting the 3 plots we want on a line together to see if it will work:

```
par(mfrow=c(1,3))
plot(wf.control$activityPRE, wf.control$activityPOST, xlab="", ylab="")
plot(gf.control$activityPRE, gf.control$activityPOST, xlab="", ylab="")
plot(bf.control$activityPRE, bf.control$activityPOST, xlab="", ylab="")
```



Ok, so now a couple of things become apparent: if we want to compare plots, we need to standardize the y-axes. Doing this also means we can remove the tick labels on plots 2 & 3, which will allow us to put the plots closer together. Finally, as I've already told you guys, I really do not want top and right plot borders – it just adds to the overall too much busyness of the graph. So, let's make those changes.

```
par(mar=c(1,1,2,1))
plot(wf.control$activityPRE, wf.control$activityPOST,  axes=FALSE, xlim=c(0,12.1),
   ylim=c(0,11), xlab="", ylab="")
axis(1, labels=FALSE)
axis(2, labels=FALSE)

plot(gf.control$activityPRE, gf.control$activityPOST, axes=FALSE, xlim=c(0,12.1),
   ylim=c(0,11), xlab="", ylab="")
axis(1, labels=FALSE)
axis(2, labels=FALSE)

plot(bf.control$activityPRE, bf.control$activityPOST, axes=FALSE, xlim=c(0,12.1),
   ylim=c(0,11), xlab="", ylab="")
axis(1, labels=FALSE)
axis(2, labels=FALSE)
```

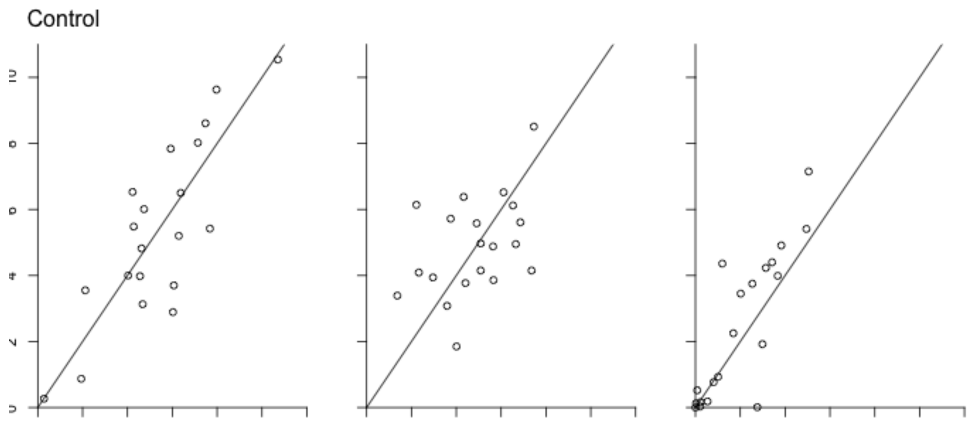


So we are getting there. But I personally like my axes to join, so I'll do that by adding the "pos = " option to axis. Also, because I'm plotting before and after activity for comparison, I'm going to add in the 1:1 line for allowing us to visually detect whether activity increased or decreased.

```
par(mar=c(1,1,2,1))

plot(wf.control$activityPRE, wf.control$activityPOST, type='n', axes=FALSE,
xlim=c(0,12.1), ylim=c(0,11), xlab="", ylab="")

axis(1, pos=0, labels=FALSE)

axis(2, pos=0)  # Another thing I've done here is put the labels back on the
                          far left graph

segments(0,0,11,11)  # Draw 1:1 line

segments(0,10,0,11)     # Extend y-axis line to 11

points(wf.control$activityPRE, wf.control$activityPOST, pch=1, type='p')

mtext("Control", side=3, adj=0)  I've also given this line of plots a label
                    indicating they are the control treatment
```

```
#### plot greenfrogs
plot(gf.control$activityPRE, gf.control$activityPOST, type='n', axes=FALSE, xlim=c(0,12.1),
ylim=c(0,11), xlab="", ylab="")
axis(1,  pos=0, labels=FALSE)
axis(2, pos=0, labels=FALSE, las=1)
segments(0,10,0,11)
segments(0,0,11,11)
points(gf.control$activityPRE, gf.control$activityPOST, pch=1, type='p')

#### plot bullfrogs
plot(bf.control$activityPRE, bf.control$activityPOST, type='n', axes=FALSE, xlim=c(0,12.1),
ylim=c(0,11), xlab="", ylab="")
```

axis(1, pos=0, labels=FALSE)
axis(2, pos=0, labels=FALSE)
segments(0,0,11,11)
segments(0,10,0,11)
points(bf.control$activityPRE, bf.control$activityPOST, pch=1, type='p')

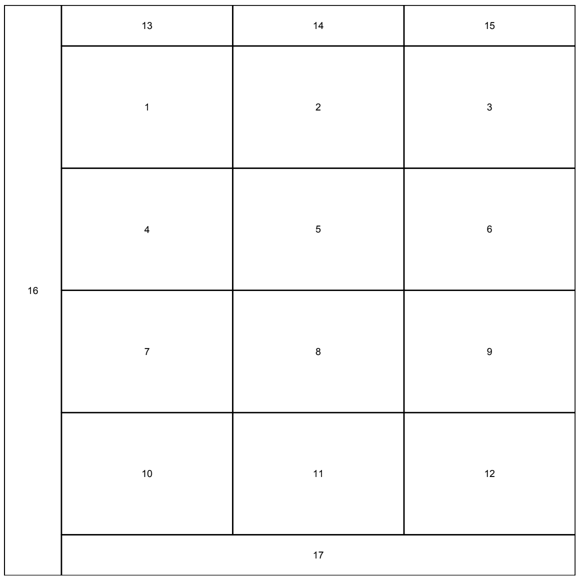


We seem to have the basic structure down now, but we want to have this panel for each of the four treatments, one on top of the other. So, let's tackle the next big step – laying out the plots more precisely in the way that we want them. To do this we use the `layout()` function.

The layout function takes a matrix as an argument that defines each 'plotting' area of the page, numbered according to the order in which you want to areas filled. The trick is that you have to put in as many areas as you need to create your smallest area, in a grid-like fashion and then put areas that you want bigger together again by assigning them the same plot number. As if you are laying on a grid of excel cells and then merge those cells you want merged. It takes some practise. Luckily, there is a `layout.show()` function, so that you can readily see whether the layout you've created is what you really want.

quartz(width=4.8,height=4.8, pointsize=9)

```
nf <- layout(matrix(
          c(16,13,14,15,
           16,1,2,3,
           16, 4,5,6,
           16, 7,8,9,
           16, 10,11,12,
           16,17,17,17), 6,4,
          byrow=TRUE),
          widths=c(1,3,3,3),
          heights=c(1,3,3,3,3,1))

layout.show(nf)
```



So, for here, I started with the 16 plots I want, but then I realize I want a single axis label for the x-axis and the y-axis of all the plots. Also, I want to be able to put a title over each column of the matrix of plots (indicating the species in that column). So I end up inputting a matrix of 24 values, and then indicating that the left-most and bottom-most areas should each be merged.

So, now we have all the pieces that we need. Below is the full code file. I've added large numbers in the left margin to highlight the actual order you might construct the graph in. I've also put the script file on the course website for you to download and play with if you like.

```
----------------------------------------------------------------------
##### Code to generate plot of activity in tadpoles, pre- and post-exposure to parasites.

###### IMPORT DATA & MANIPULATE ###########
dat <- read.table("Rdata.csv", sep = ",", header=T)
levels(dat$trtmt) <- c("control", "fish", "parasite", "combo", ordered=TRUE)

# Create separate vectors for each species/treatment combination

bf <- dat[which(dat$Species=="bullfrog"),]
gf <-dat[which(dat$Species=="greenfrog"),]
wf <- dat[which(dat$Species=="woodfrog"),]

wf.control <- subset(wf, trtmt=='control')
wf.fish <- subset(wf, trtmt=='fish')
wf.parasite <- subset(wf, trtmt=='parasite')
wf.both <- subset(wf, trtmt=='combo')

gf.control <- subset(gf, trtmt=='control')
gf.fish <- subset(gf, trtmt=='fish')
gf.parasite <- subset(gf, trtmt=='parasite')
gf.both <- subset(gf, trtmt=='combo')
```

```
bf.control <- subset(bf, trtmt=='control')
bf.fish <- subset(bf, trtmt=='fish')
bf.parasite <- subset(bf, trtmt=='parasite')
bf.both <- subset(bf, trtmt=='combo')


graphics.off()
#tiff("fig1.tiff", width=12.35, height=12.35,
   units="cm",pointsize=8,compression="lzw",res=300, type=c("quartz"), fonts="Helvetica")
quartz(width=4.8,height=4.8, pointsize=9)
nf <- layout(matrix(
   c(16,13,14,15,
        16,1,2,3,
        16, 4,5,6,
        16, 7,8,9,
        16, 10,11,12,
        16,17,17,17), 6,4, byrow=TRUE), widths=c(1,3,3,3), heights=c(1,3,3,3,3,1))
layout.show(nf)
```

7

```
#######CONTROLS

#### plot woodfrogs
par(mar=c(1,1,2,1))
plot(wf.control$activityPRE, wf.control$activityPOST, xlab="", ylab="")
axis(1,  pos=0, labels=FALSE)
axis(2, pos=0)
segments(0,0,11,11)
segments(0,10,0,11)
points(wf.control$activityPRE, wf.control$activityPOST, pch=1, type='p')
```

1

```
#### plot greenfrogs
plot(gf.control$activityPRE, gf.control$activityPOST, type='n', axes=FALSE, fullaxis(1,  pos=0,
   labels=FALSE)
axis(2, pos=0, labels=FALSE, las=1)
segments(0,10,0,11)
segments(0,0,11,11)
points(gf.control$activityPRE, gf.control$activityPOST, pch=1, type='p')
```

2

```
#### plot bullfrogs
plot(bf.control$activityPRE, bf.control$activityPOST, type='n', axes=FALSE, xlim=c(0,12.1),
   ylim=c(0,11), xlab="", ylab="")
axis(1,  pos=0, labels=FALSE)
axis(2, pos=0, labels=FALSE)
segments(0,0,11,11)
segments(0,10,0,11)
```

3

```
points(bf.control$activityPRE, bf.control$activityPOST, pch=1, type='p')
```

4

```
#################### WITH FISH

plot(wf.fish$activityPRE, wf.fish$activityPOST, type='n', axes=FALSE, xlim=c(0,12.1),
    ylim=c(0,11), xlab="", ylab="")
axis(1,  pos=0, labels=FALSE)
axis(2, pos=0)
segments(0,0,11,11)
segments(0,10,0,11)
points(wf.fish$activityPRE, wf.fish$activityPOST, pch=1, type='p')
mtext("with Fish", side=3, adj=0)

#### plot greenfrogs
plot(gf.fish$activityPRE, gf.fish$activityPOST, type='n', axes=FALSE, xlim=c(0,12.1),
    ylim=c(0,11), xlab="", ylab="")
axis(1,  pos=0, labels=FALSE)
axis(2, pos=0,labels=FALSE)
segments(0,0,11,11)
segments(0,10,0,11)
points(gf.fish$activityPRE, gf.fish$activityPOST, pch=1, type='p')

#### plot bullfrogs
plot(bf.fish$activityPRE, bf.fish$activityPOST, type='n', axes=FALSE, xlim=c(0,12.1),
    ylim=c(0,11), xlab="", ylab="")
axis(1,  pos=0,labels=FALSE)
axis(2, pos=0, labels=FALSE)
segments(0,0,11,11)
segments(0,10,0,11)
points(bf.fish$activityPRE, bf.fish$activityPOST, pch=1, type='p')
#text(1.1,4,"Wood frogs")
```

5

```
#################### WITH PARASITE

plot(wf.parasite$activityPRE, wf.parasite$activityPOST, type='n', axes=FALSE, xlim=c(0,12.1),
    ylim=c(0,11), xlab="", ylab="")
axis(1,  pos=0, labels=FALSE)
segments(0,0,11,11)
segments(0,10,0,11)
points(wf.parasite$activityPRE, wf.parasite$activityPOST, pch=1, type='p')
mtext("with Parasite", side=3, adj=0)

#### plot greenfrogs
```

```
plot(gf.parasite$activityPRE, gf.parasite$activityPOST, type='n', axes=FALSE, xlim=c(0,12.1),
    ylim=c(0,11), xlab="", ylab="")
axis(1,  pos=0,labels=FALSE)
axis(2, pos=0,labels=FALSE)
segments(0,0,11,11)
segments(0,10,0,11)
points(gf.parasite$activityPRE, gf.parasite$activityPOST, pch=1, type='p')

#### plot bullfrogs
plot(bf.parasite$activityPRE, bf.parasite$activityPOST, type='n', axes=FALSE, xlim=c(0,12.1),
    ylim=c(0,11), xlab="", ylab="")
axis(1,  pos=0,labels=FALSE)
axis(2, pos=0,labels=FALSE)
segments(0,0,11,11)
segments(0,10,0,11)
points(bf.parasite$activityPRE, bf.parasite$activityPOST, pch=1, type='p')
```

6

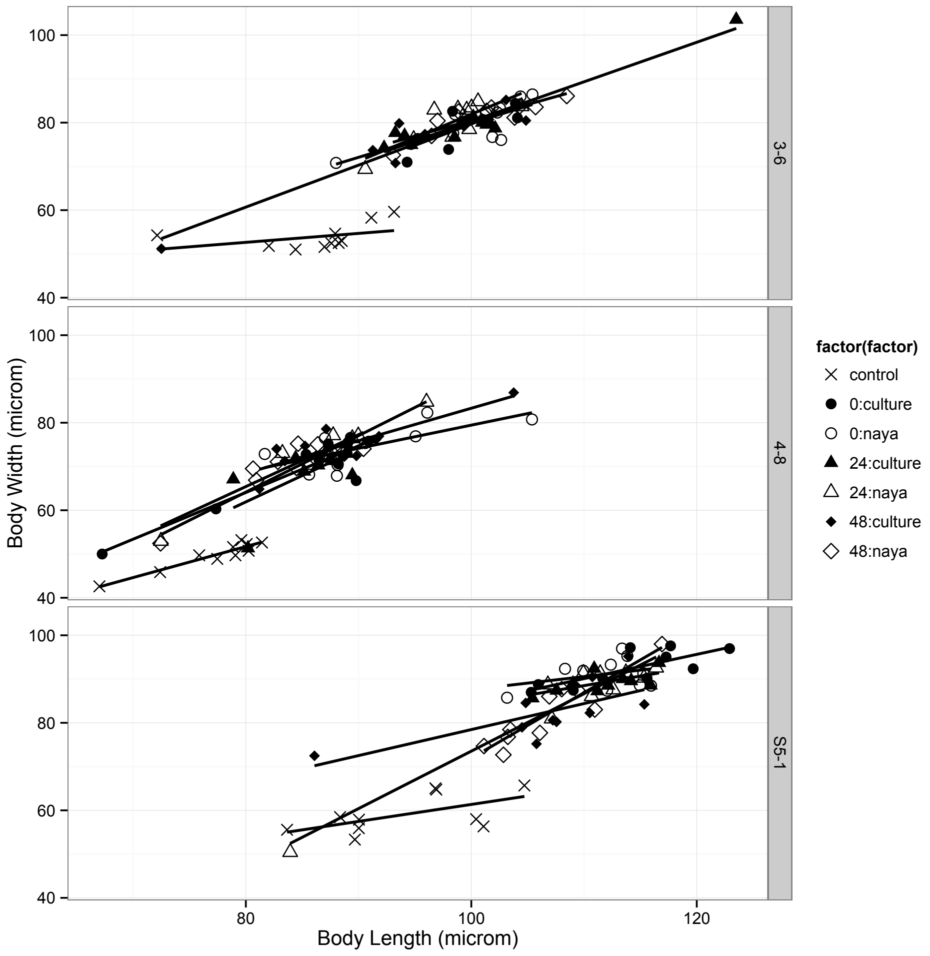
```
################## WITH BOTH

plot(wf.both$activityPRE, wf.both$activityPOST, type='n', axes=FALSE, xlim=c(0,12.1),
    ylim=c(0,11), xlab="", ylab="")
axis(1,  pos=0)
axis(2, pos=0)
segments(0,0,11,11)
segments(0,10,0,11)
points(wf.both$activityPRE, wf.both$activityPOST, pch=1, type='p')
mtext("with Both", side=3, adj=0)

#### plot greenfrogs
plot(gf.both$activityPRE, gf.both$activityPOST, type='n', axes=FALSE, xlim=c(0,12.1),
    ylim=c(0,11), xlab="", ylab="")
axis(1,  pos=0)
axis(2, pos=0, labels=FALSE)
segments(0,0,11,11)
segments(0,10,0,11)
points(gf.both$activityPRE, gf.both$activityPOST, pch=1, type='p')

#### plot bullfrogs
plot(bf.both$activityPRE, bf.both$activityPOST, type='n', axes=FALSE, xlim=c(0,12.1),
    ylim=c(0,11), xlab="", ylab="")
axis(1,  pos=0)
axis(2, pos=0,labels=FALSE)
segments(0,0,11,11)
segments(0,10,0,11)
points(bf.both$activityPRE, bf.both$activityPOST, pch=1, type='p')

## Add text labels
```

8

```
par(mar=c(0,0,0,0))
plot(0:3,0:3, axes=FALSE, ann=FALSE, type='n')
text(1.5,0.5,"L. sylvaticus", cex=1.5, font=3)
plot(0:3,0:3, axes=FALSE, ann=FALSE, type='n')
text(1.5,0.5,"L. clamitans", cex=1.5, font=3)
plot(0:3,0:3, axes=FALSE, ann=FALSE, type='n')
text(1.5,0.5,"L. catesbeianus", cex=1.5, font=3)

plot(0:3,0:3, axes=FALSE, ann=FALSE, type='n')
text(1.5,1.5, "Time Active Post-Treatment (min)", srt=90, cex=1.5)
plot(0:3,0:3, axes=FALSE, ann=FALSE, type='n')
text(1.5,1.3, "Time Active Pre-Treatment (min)", cex=1.5)

dev.off()
```

The only new thing in the code above is the various text labels I've added into the outermost plot areas. Note that for each area I've used a plot command to create a plot area of a known scale, but with no axes, annotations or plotting. Then I use the scale to position my text within the label area.

Here's an example of a more complicated graph done using ggplot2.



For the
purposes of this photocopy I've reproduced the black and white version above, but we'll look at the code for a colour version of the same plot.

```
library(ggplot2)
graph1 <- ggplot(data=dat, aes(x=MEANlength, y=MEANwidth,
group=factor(factor))) + geom_point(aes(colour=factor(factor)),
na.rm=TRUE, size=3) + stat_smooth(method=lm, level=0, na.rm=TRUE,
aes(colour=factor(factor)), size=0.8) +
scale_shape_manual(values=c("0:culture"=16,"24:culture"=17,"48:culture"=18
,"0:naya"=1,"24:naya"=2,"48:naya"=5, "control"=4)) +
scale_linetype_manual(values=c("0:culture"=2,"24:culture"=3,
"48:culture"=4,"0:naya"=2,"24:naya"=3,"48:naya"=4,"control"=1)) +
facet_grid(facets = Strain ~.) + theme_bw() + ylab("Body Width (microm)")
+ xlab("Body Length (microm)") + theme(legend.key=element_blank(),
axis.title.x=element_text(vjust=0))
```

The full code to prepare the data for graphing and the raw data file are both available on the course website