MP Oyster Depuration Project

Garth C

January 9, 2018

Load the libraries we’ll need. You’ll have to install these if you don’t already have them.

library(plyr)  
library(reshape2)  
library(ggplot2)  
library(glmmADMB)

## Loading required package: MASS

##   
## Attaching package: 'glmmADMB'

## The following object is masked from 'package:MASS':  
##   
## stepAIC

## The following object is masked from 'package:stats':  
##   
## step

library(MuMIn)

## Load, organize, and plot data

Import the oyster depuration data and view it.

oysterdep <-   
 read.csv("~/GRAD school/ACRDP/Depuration/Data/Analysis/R code/oysterdep.csv")  
  
head(oysterdep)

## id sampleday size.cat Table Tank species DOFSample length  
## 1 MIS1 0 twentyto50 n/a n/a C. gigas 11-03-2016 93  
## 2 MIS1 0 fiftyto100 n/a n/a C. gigas 11-03-2016 93  
## 3 M1S1 0 onehundredto500 n/a n/a C. gigas 11-03-2016 93  
## 4 M1S1 0 fivehundredto1000 n/a n/a C. gigas 11-03-2016 93  
## 5 M1S1 0 onethousandto5000 n/a n/a C. gigas 11-03-2016 93  
## 6 MIS2 0 twentyto50 n/a n/a C. gigas 11-03-2016 66  
## width depth cont.weight cont.dryweight dry.weight shell.dry.weight CI  
## 1 43 21 259.28 260.96 1.68 17.65 0.095  
## 2 43 21 259.28 260.96 1.68 17.65 0.095  
## 3 43 21 259.28 260.96 1.68 17.65 0.095  
## 4 43 21 259.28 260.96 1.68 17.65 0.095  
## 5 43 21 259.28 260.96 1.68 17.65 0.095  
## 6 40 21 262.04 263.63 1.59 12.62 0.126  
## time.in.oven time.out.oven dat.filter dat.count observer  
## 1 2016-11-05/ 17:03 2016-11-10/18:00 11-17-2016 11-29-2016 MD  
## 2 2016-11-05/ 17:03 2016-11-10/18:00 11-17-2016 11-29-2016 MD  
## 3 2016-11-05/ 17:03 2016-11-10/18:00 11-17-2016 11-29-2016 MD  
## 4 2016-11-05/ 17:03 2016-11-10/18:00 11-17-2016 11-29-2016 MD  
## 5 2016-11-05/ 17:03 2016-11-10/18:00 11-17-2016 11-29-2016 MD  
## 6 2016-11-05/ 17:03 2016-11-10/18:00 11-17-2016 11-28-2016 GC  
## red.fib yell.fib green.fib turq.fib blu.fib purp.fib gray.fib clear.fib  
## 1 0 0 0 0 0 0 0 0  
## 2 0 0 0 0 0 0 0 0  
## 3 0 0 0 0 1 0 0 0  
## 4 0 0 0 0 1 0 0 0  
## 5 0 0 0 0 0 0 0 0  
## 6 0 0 0 0 0 0 0 0  
## pink.fib brown.fib orang.fib black.fib red.frag yell.frag green.frag  
## 1 0 0 0 0 0 0 0  
## 2 0 0 0 0 0 0 0  
## 3 0 0 0 1 0 0 0  
## 4 0 0 0 0 0 0 0  
## 5 0 0 0 0 0 0 0  
## 6 0 0 0 0 0 0 0  
## turq.frag blu.frag purp.frag gray.frag clear.frag pink.frag brown.frag  
## 1 0 0 0 0 0 0 0  
## 2 0 0 0 0 0 0 0  
## 3 0 0 0 0 0 0 0  
## 4 0 0 0 0 0 0 0  
## 5 0 0 0 0 0 0 0  
## 6 0 0 0 0 0 0 0  
## orang.frag black.frag whit.spher clear.spher  
## 1 0 0 0 0  
## 2 0 0 0 0  
## 3 0 0 0 0  
## 4 0 0 0 0  
## 5 0 0 0 0  
## 6 0 0 0 0

Get all of the variables into the right format.

oysterdep$id <- as.factor(oysterdep$id)  
oysterdep$sampleday <- as.factor(oysterdep$sampleday)  
oysterdep$size.cat <- as.factor(oysterdep$size.cat)  
oysterdep$Table <- as.factor(oysterdep$Table)  
oysterdep$Tank <- as.factor(oysterdep$Tank)  
oysterdep$observer <- as.factor(oysterdep$observer)

Make a new dataframe to work with and rename size categories.

oysterdep2 <- oysterdep  
  
oysterdep2$size.cat <- factor(oysterdep2$size.cat,   
 levels=c('twentyto50', 'fiftyto100',  
 'onehundredto500', 'fivehundredto1000',   
 'onethousandto5000')) # reorder  
  
  
oysterdep2$size.cat <- mapvalues(oysterdep2$size.cat,   
 from = c('twentyto50', 'fiftyto100',   
 'onehundredto500',   
 'fivehundredto1000',   
 'onethousandto5000'),   
 to = c('20-50', '50-100', '100-500',  
 '500-1000', '1000-5000')) # rename  
summary(oysterdep2$size.cat) # view

## 20-50 50-100 100-500 500-1000 1000-5000   
## 49 50 50 50 50

We need to melt the dataframe so that the particle types are organized by row instead of by column.

oysterdep3 <- melt(oysterdep2,  
 id.vars = c('id', 'sampleday', 'size.cat', 'Table', 'Tank',  
 'species', 'DOFSample', 'length', 'width',   
 'depth', 'cont.weight', 'cont.dryweight',   
 'dry.weight', 'shell.dry.weight', 'CI',  
 'time.in.oven', 'time.out.oven',  
 'dat.filter', 'dat.count', 'observer'))  
  
## Take out unnecessary columns  
  
oysterdep3 <- oysterdep3[c('id', 'sampleday', 'size.cat', 'Table', 'Tank',   
 'DOFSample', 'length', 'width', 'depth', 'dry.weight',  
 'shell.dry.weight', 'CI', 'dat.filter', 'dat.count',  
 'observer', 'variable', 'value')]  
  
head(oysterdep3) # now 'variable' is particle category and 'value is the count'

## id sampleday size.cat Table Tank DOFSample length width depth  
## 1 MIS1 0 20-50 n/a n/a 11-03-2016 93 43 21  
## 2 MIS1 0 50-100 n/a n/a 11-03-2016 93 43 21  
## 3 M1S1 0 100-500 n/a n/a 11-03-2016 93 43 21  
## 4 M1S1 0 500-1000 n/a n/a 11-03-2016 93 43 21  
## 5 M1S1 0 1000-5000 n/a n/a 11-03-2016 93 43 21  
## 6 MIS2 0 20-50 n/a n/a 11-03-2016 66 40 21  
## dry.weight shell.dry.weight CI dat.filter dat.count observer  
## 1 1.68 17.65 0.095 11-17-2016 11-29-2016 MD  
## 2 1.68 17.65 0.095 11-17-2016 11-29-2016 MD  
## 3 1.68 17.65 0.095 11-17-2016 11-29-2016 MD  
## 4 1.68 17.65 0.095 11-17-2016 11-29-2016 MD  
## 5 1.68 17.65 0.095 11-17-2016 11-29-2016 MD  
## 6 1.59 12.62 0.126 11-17-2016 11-28-2016 GC  
## variable value  
## 1 red.fib 0  
## 2 red.fib 0  
## 3 red.fib 0  
## 4 red.fib 0  
## 5 red.fib 0  
## 6 red.fib 0

Now we want to see what kind of particles we found and remove any categories that we didn’t end up using.

cat.sums <- aggregate(oysterdep3$value ~   
 oysterdep3$variable, FUN = sum) # sums everything  
print(cat.sums)

## oysterdep3$variable oysterdep3$value  
## 1 red.fib 1  
## 2 yell.fib 3  
## 3 green.fib 4  
## 4 turq.fib 1  
## 5 blu.fib 52  
## 6 purp.fib 0  
## 7 gray.fib 0  
## 8 clear.fib 302  
## 9 pink.fib 2  
## 10 brown.fib 9  
## 11 orang.fib 3  
## 12 black.fib 158  
## 13 red.frag 1  
## 14 yell.frag 0  
## 15 green.frag 1  
## 16 turq.frag 0  
## 17 blu.frag 2  
## 18 purp.frag 0  
## 19 gray.frag 0  
## 20 clear.frag 2  
## 21 pink.frag 2  
## 22 brown.frag 0  
## 23 orang.frag 0  
## 24 black.frag 0  
## 25 whit.spher 6  
## 26 clear.spher 1

So we can remove the purple fibre, gray fibre, yellow fragment, purple fragment, gray fragment, brown fragment, orange fragment, and black fragment categories because we didn’t count any of these.

oysterdep4 <- subset(oysterdep3, variable != 'purp.fib' &   
 variable != 'gray.fib' & variable != 'yell.frag' &  
 variable != 'purp.frag' & variable != 'gray.frag' &   
 variable != 'brown.frag' & variable != 'orang.frab' &  
 variable != 'black.frag') # removes all these rows  
  
oysterdep4$variable <- as.character(oysterdep4$variable)  
oysterdep4$variable <- as.factor(oysterdep4$variable) # gets rid of those names  
  
summary(oysterdep4$variable) # view

## black.fib blu.fib blu.frag brown.fib clear.fib clear.frag   
## 249 249 249 249 249 249   
## clear.spher green.fib green.frag orang.fib orang.frag pink.fib   
## 249 249 249 249 249 249   
## pink.frag red.fib red.frag turq.fib turq.frag whit.spher   
## 249 249 249 249 249 249   
## yell.fib   
## 249

Let’s make the category names a bit more useful for plotting

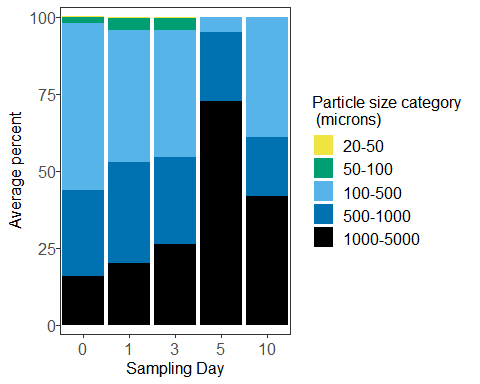
oysterdep4$variable <- mapvalues(oysterdep4$variable,  
 from = c(levels(oysterdep4$variable)),   
 to = c('Black fibres', 'Blue fibres',   
 'Blue fragments', 'Brown fibres',   
 'Clear fibres', 'Clear fragments',   
 'Clear spherules', 'Green fibres',  
 'Green fragments', 'Orange fibres',  
 'Orange fragments', 'Pink fibres',   
 'Pink fragments', 'Red fibres',  
 'Red fragments', 'Turquoise fibres',   
 'Turquoise fragments', 'White spherules',   
 'Yellow fibres'))  
  
## And reorder them  
oysterdep4$variable <- factor(oysterdep4$variable,   
 levels = c('Black fibres', 'Blue fibres',  
 'Brown fibres', 'Clear fibres',  
 'Green fibres', 'Orange fibres',  
 'Pink fibres', 'Red fibres',  
 'Turquoise fibres',  
 'Yellow fibres', 'Blue fragments',  
 'Clear fragments', 'Green fragments',  
 'Orange fragments', 'Pink fragments',  
 'Red fragments', 'Turquoise fragments',  
 'Clear spherules', 'White spherules'))  
  
summary(oysterdep4$variable)

## Black fibres Blue fibres Brown fibres   
## 249 249 249   
## Clear fibres Green fibres Orange fibres   
## 249 249 249   
## Pink fibres Red fibres Turquoise fibres   
## 249 249 249   
## Yellow fibres Blue fragments Clear fragments   
## 249 249 249   
## Green fragments Orange fragments Pink fragments   
## 249 249 249   
## Red fragments Turquoise fragments Clear spherules   
## 249 249 249   
## White spherules   
## 249

Next let’s make plots of the distribution of sizes and particle types and colours.

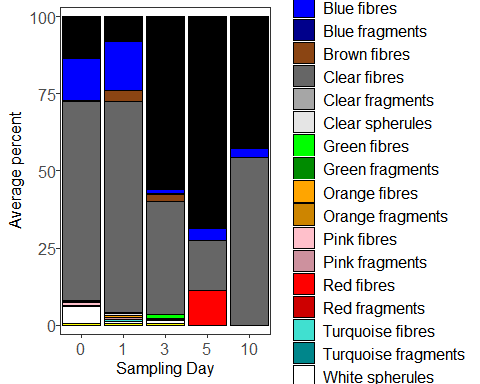
Starting with size

## First we need to make a new data frame of sums according to each size category  
  
oystersizesums <- ddply(oysterdep4, c('id', 'sampleday', 'size.cat', 'Table',   
 'Tank', 'DOFSample', 'length', 'width',  
 'depth', 'dry.weight', 'shell.dry.weight',  
 'CI', 'dat.filter', 'dat.count',   
 'observer'), summarize, sum = sum(value))  
  
  
  
## Now calculate proportions of each particle size category present in each   
## sample and take the means for each sampling day  
  
oysterprop <- ddply(oystersizesums, 'id', transform, prop = sum/sum(sum))  
  
oysterprop2 <- subset(oysterprop, !(is.na(oysterprop["prop"]))) # remove any NAs  
  
oysterprop2$size.cat <- as.character(oysterprop2$size.cat)  
oysterprop2$size.cat <- as.factor(oysterprop2$size.cat)  
  
oysterprop3 <- ddply(oysterprop2, c("sampleday", "size.cat"), summarize,  
 mean = mean(prop))  
  
## Put the size categories in order  
  
oysterprop3$size.cat <- factor(oysterprop3$size.cat,   
 levels = c("10-20", "20-50", "50-100",   
 "100-500", "500-1000",   
 "1000-5000"))  
  
## Now plot  
  
ggplot(oysterprop3, aes(x=sampleday, y=100\*mean, fill = size.cat)) +   
 geom\_col(size = 1) +   
 xlab('Sampling Day') +   
 ylab('Average percent') +  
 guides(fill=guide\_legend(title="Particle size category \n (microns)")) +  
 theme\_bw() +  
 scale\_fill\_manual(values = c("#F0E442", "#009E73",   
 "#56B4E9", "#0072B2", "black")) +  
 theme(legend.text = element\_text(size=12), text = element\_text(size=12),   
 panel.spacing = unit(1, "lines"),   
 axis.text.x = element\_text(size = 12),  
 axis.text.y = element\_text(size = 12,   
 margin = margin(t=0, r=0, b=0, l=2)),  
 strip.background = element\_blank(),   
 panel.grid.major = element\_blank(),   
 panel.grid.minor = element\_blank(),  
 strip.placement = "outside") +  
 scale\_x\_discrete(expand = expand\_scale(0,0.5)) +   
 scale\_y\_continuous(expand = expand\_scale(0.03,0))



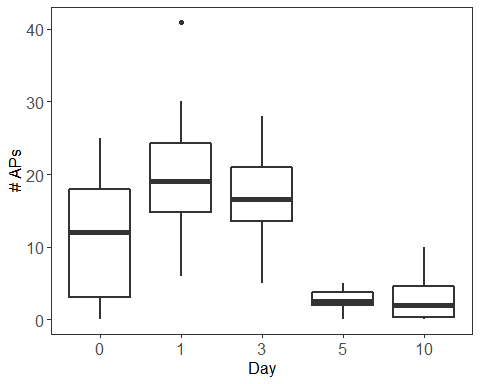
Now with particle type/colour

## First we need to make a new data frame of sums according to each particle  
## type/colour  
  
oystertypesums <- ddply(oysterdep4, c('id', 'sampleday', 'Table',   
 'Tank', 'DOFSample', 'length', 'width',  
 'depth', 'dry.weight', 'shell.dry.weight',  
 'CI', 'dat.filter', 'dat.count',   
 'observer', 'variable'), summarize,   
 sum = sum(value))  
  
  
  
## Now calculate proportions of each particle type/colour present in each   
## sample and take the means for each sampling day  
  
oystertypeprop <- ddply(oystertypesums, 'id', transform, prop = sum/sum(sum))  
  
oystertypeprop2 <- subset(oystertypeprop, !(is.na(oystertypeprop["prop"]))) # remove any NAs  
  
oystertypeprop2$variable <- as.character(oystertypeprop2$variable)  
oystertypeprop2$variable <- as.factor(oystertypeprop2$variable)  
  
oystertypeprop3 <- ddply(oystertypeprop2, c("sampleday", "variable"), summarize,  
 mean = mean(prop))  
  
## Now plot  
  
ggplot(oystertypeprop3, aes(x=sampleday, y=100\*mean, fill = variable)) +   
 geom\_col(size = 0.5, colour = 'black') +   
 xlab('Sampling Day') +   
 ylab('Average percent') +  
 guides(fill=guide\_legend(title="Particle size category \n (microns)")) +  
 theme\_bw() +  
 scale\_fill\_manual(values = c('black', 'blue', 'blue4', 'saddlebrown', 'grey40',   
 'grey65', 'grey90', 'green', 'green4',   
 'orange', 'orange3', 'pink', 'pink3',   
 'red', 'red3', 'turquoise', 'turquoise4',   
 'white', 'yellow')) +  
 theme(legend.text = element\_text(size=12), text = element\_text(size=12),   
 panel.spacing = unit(1, "lines"),   
 axis.text.x = element\_text(size = 12),  
 axis.text.y = element\_text(size = 12,   
 margin = margin(t=0, r=0, b=0, l=2)),  
 strip.background = element\_blank(),   
 panel.grid.major = element\_blank(),   
 panel.grid.minor = element\_blank(),  
 strip.placement = "outside") +  
 scale\_x\_discrete(expand = expand\_scale(0,0.5)) +   
 scale\_y\_continuous(expand = expand\_scale(0.03,0))

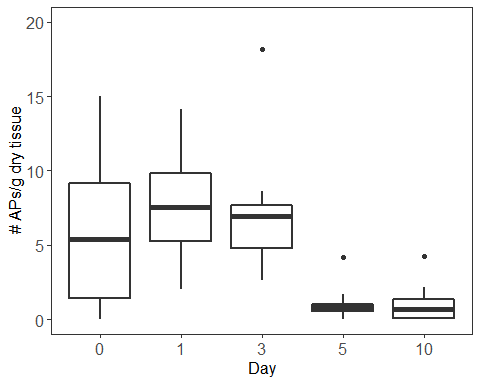


Take total sums overall and plot the results.

totalsums <- ddply(oysterdep4, c('id', 'sampleday', 'Table', 'Tank',   
 'length', 'width', 'depth', 'dry.weight',  
 'shell.dry.weight', 'CI', 'observer'),  
 summarise, sum = sum(value)) # adds all the counts together  
  
## Plot according to count per individual  
  
ggplot(totalsums, aes(x=sampleday , y=sum)) +   
 geom\_boxplot(position=position\_dodge(), size=1) +   
 xlab('Day') +   
 coord\_cartesian(ylim = c(0, 41)) +  
 ylab('# APs') +  
 guides(fill=FALSE) +  
 theme\_bw() +  
 theme(text = element\_text(size=12),   
 axis.text.x = element\_text(size = 12),  
 axis.text.y = element\_text(size = 12),  
 strip.background = element\_blank(),  
 panel.grid.major = element\_blank(),   
 panel.grid.minor = element\_blank())



## Plot according to concentration (particles/dry tissue weight)  
  
ggplot(totalsums, aes(x=sampleday , y=sum/dry.weight)) +   
 geom\_boxplot(position=position\_dodge(), size=1) +   
 xlab('Day') +   
 coord\_cartesian(ylim = c(0, 20)) +  
 ylab('# APs/g dry tissue') +  
 guides(fill=FALSE) +  
 theme\_bw() +  
 theme(text = element\_text(size=12),   
 axis.text.x = element\_text(size = 12),  
 axis.text.y = element\_text(size = 12),  
 strip.background = element\_blank(),  
 panel.grid.major = element\_blank(),   
 panel.grid.minor = element\_blank())



Let’s print out some summary stats.

## Summary stats according to count per individual  
  
with(totalsums, tapply(sum, sampleday, mean)) # takes mean for each day

## 0 1 3 5 10   
## 11.36364 20.40000 16.60000 2.60000 2.90000

with(totalsums, tapply(sum, sampleday, min)) # takes minimum for each day

## 0 1 3 5 10   
## 0 6 5 0 0

with(totalsums, tapply(sum, sampleday, max)) # takes maximum for each day

## 0 1 3 5 10   
## 25 41 28 5 10

with(totalsums, tapply(sum,   
 sampleday, sd)) # takes standard deviation for each day

## 0 1 3 5 10   
## 9.531765 9.968840 7.260242 1.505545 3.281260

## Summary stats according to concentration  
  
with(totalsums, tapply(sum/dry.weight,   
 sampleday, mean)) # takes mean for each day

## 0 1 3 5 10   
## 5.698379 7.612293 7.261758 1.106082 1.043232

with(totalsums, tapply(sum/dry.weight,   
 sampleday, min)) # takes minimum for each day

## 0 1 3 5 10   
## 0.000000 2.027027 2.641509 0.000000 0.000000

with(totalsums, tapply(sum/dry.weight,   
 sampleday, max)) # takes maximum for each day

## 0 1 3 5 10   
## 14.970060 14.150943 18.181818 4.201681 4.237288

with(totalsums, tapply(sum,   
 sampleday, sd)) # takes standard deviation for each day

## 0 1 3 5 10   
## 9.531765 9.968840 7.260242 1.505545 3.281260

## Proportions of particles sizes  
  
sum(oysterdep4$value[oysterdep4$size.cat == '10-20'])/  
 sum(oysterdep4$value)\*100 # 0%

## [1] 0

sum(oysterdep4$value[oysterdep4$size.cat == '20-50'])/  
 sum(oysterdep4$value)\*100 # 0.55 %

## [1] 0.5454545

sum(oysterdep4$value[oysterdep4$size.cat == '50-100'])/  
 sum(oysterdep4$value)\*100 # 3.09 %

## [1] 3.090909

sum(oysterdep4$value[oysterdep4$size.cat == '100-500'])/  
 sum(oysterdep4$value)\*100 # 41.82 %

## [1] 41.81818

sum(oysterdep4$value[oysterdep4$size.cat == '500-1000'])/  
 sum(oysterdep4$value)\*100 # 28.00 %

## [1] 28

sum(oysterdep4$value[oysterdep4$size.cat == '1000-5000'])/  
 sum(oysterdep4$value)\*100 # 26.55 %

## [1] 26.54545

## Sums by particle type  
  
summary(oysterdep4$variable)

## Black fibres Blue fibres Brown fibres   
## 249 249 249   
## Clear fibres Green fibres Orange fibres   
## 249 249 249   
## Pink fibres Red fibres Turquoise fibres   
## 249 249 249   
## Yellow fibres Blue fragments Clear fragments   
## 249 249 249   
## Green fragments Orange fragments Pink fragments   
## 249 249 249   
## Red fragments Turquoise fragments Clear spherules   
## 249 249 249   
## White spherules   
## 249

sum(oysterdep4$value[oysterdep4$variable == 'Blue fragments' |   
 oysterdep4$variable == 'Clear fragments' |  
 oysterdep4$variable == 'Green fragments' |   
 oysterdep4$variable == 'Orange fragments' |  
 oysterdep4$variable == 'Pink fragments' |  
 oysterdep4$variable == 'Red fragments' |  
 oysterdep4$variable == 'Turquoise fragments'])/  
 sum(oysterdep4$value) \* 100

## [1] 1.454545

sum(oysterdep4$value[oysterdep4$variable == 'Clear spherules' |   
 oysterdep4$variable == 'White spherules'])/  
 sum(oysterdep4$value) \* 100

## [1] 1.272727

## 1.45% fragments  
## 1.27% spherules  
## 97.28% fibres  
  
## This provides the percentage of total particles found accounted for by each   
## colour/type of particles  
oystersummarytable <- with(oysterdep4, tapply(value, variable, sum))  
oystersummarytable <- oystersummarytable/sum(oysterdep4$value)\*100  
print(oystersummarytable)

## Black fibres Blue fibres Brown fibres   
## 28.7272727 9.4545455 1.6363636   
## Clear fibres Green fibres Orange fibres   
## 54.9090909 0.7272727 0.5454545   
## Pink fibres Red fibres Turquoise fibres   
## 0.3636364 0.1818182 0.1818182   
## Yellow fibres Blue fragments Clear fragments   
## 0.5454545 0.3636364 0.3636364   
## Green fragments Orange fragments Pink fragments   
## 0.1818182 0.0000000 0.3636364   
## Red fragments Turquoise fragments Clear spherules   
## 0.1818182 0.0000000 0.1818182   
## White spherules   
## 1.0909091

### Blanks

Now load in the blanks dataset and set it up for plotting, same as with the oyster data set. First we load in the data.

depblanks <-   
 read.csv("~/GRAD school/ACRDP/Depuration/Data/Analysis/depblanks.csv")  
  
head(depblanks)

## id size.cat DOFSample time.in.oven  
## 1 MControl1 (0,1) twentyto50 11-17-2016 2016-11-05/ 17:03  
## 2 MControl1 (0,1) fiftyto100 11-17-2016 2016-11-05/ 17:03  
## 3 MControl1 (0,1) onehundredto500 11-17-2016 2016-11-05/ 17:03  
## 4 MControl1 (0,1) fivehundredto1000 11-17-2016 2016-11-05/ 17:03  
## 5 MControl1 (0,1) onethousandto5000 11-17-2016 2016-11-05/ 17:03  
## 6 MControl2 (0,1) twentyto50 11-17-2016 2016-11-05/ 17:03  
## time.out.oven dat.filter dat.count observer red.fib yell.fib  
## 1 2016-11-10/18:00 11-17-2016 11-24-2016 GC 0 0  
## 2 2016-11-10/18:00 11-17-2016 11-24-2016 GC 0 0  
## 3 2016-11-10/18:00 11-17-2016 11-24-2016 GC 0 0  
## 4 2016-11-10/18:00 11-17-2016 11-24-2016 GC 0 0  
## 5 2016-11-10/18:00 11-17-2016 11-24-2016 GC 0 0  
## 6 2016-11-10/18:00 11-17-2016 11-24-2016 GC 0 0  
## green.fib turq.fib blu.fib purp.fib gray.fib clear.fib pink.fib  
## 1 0 0 0 0 0 0 0  
## 2 0 0 0 0 0 0 0  
## 3 0 0 0 0 0 1 0  
## 4 0 0 0 0 0 1 0  
## 5 0 0 0 0 0 3 0  
## 6 0 0 0 0 0 0 0  
## brown.fib orang.fib black.fib tot.fib red.frag yell.frag green.frag  
## 1 0 0 0 0 0 0 0  
## 2 0 0 0 0 0 0 0  
## 3 0 0 0 1 0 0 0  
## 4 0 0 0 1 0 0 0  
## 5 0 0 0 3 0 0 0  
## 6 0 0 0 0 0 0 0  
## turq.frag blu.frag purp.frag gray.frag clear.frag pink.frag brown.frag  
## 1 0 0 0 0 0 0 0  
## 2 0 0 0 0 0 0 0  
## 3 0 0 0 0 0 0 0  
## 4 0 0 0 0 0 0 0  
## 5 0 0 0 0 0 0 0  
## 6 0 0 0 0 0 0 0  
## orang.frag black.frag whit.spher clear.spher tot.frag.spher tot.part  
## 1 0 0 0 0 0 0  
## 2 0 0 0 0 0 0  
## 3 0 0 0 0 0 0  
## 4 0 0 0 0 0 0  
## 5 0 0 0 0 0 0  
## 6 0 0 0 0 0 0

depblanks$id <- as.factor(depblanks$id)  
depblanks$size.cat <- as.factor(depblanks$size.cat)  
depblanks$observer <- as.factor(depblanks$observer)

Now we make a new dataframe to work with and rename size categories.

depblanks2 <- depblanks  
  
depblanks2$size.cat <- factor(depblanks2$size.cat,   
 levels=c('twentyto50', 'fiftyto100',  
 'onehundredto500', 'fivehundredto1000',   
 'onethousandto5000')) # reorder  
  
  
depblanks2$size.cat <- mapvalues(depblanks2$size.cat,   
 from = c('twentyto50', 'fiftyto100',   
 'onehundredto500',   
 'fivehundredto1000',   
 'onethousandto5000'),   
 to = c('20-50', '50-100', '100-500',  
 '500-1000', '1000-5000')) # rename  
summary(depblanks2$size.cat) # view

## 20-50 50-100 100-500 500-1000 1000-5000   
## 9 9 9 9 9

Again, we melt the dataframe so that the particle types are organized by row instead of by column.

depblanks3 <- melt(depblanks2,  
 id.vars = c('id', 'size.cat', 'DOFSample', 'time.in.oven',   
 'time.out.oven', 'dat.filter', 'dat.count',   
 'observer'))  
  
## Take out unnecessary columns  
  
depblanks3 <- depblanks3[c('id', 'size.cat', 'DOFSample', 'dat.filter', 'dat.count',  
 'observer', 'variable', 'value')]  
  
head(depblanks3) # now 'variable' is particle category and 'value is the count'

## id size.cat DOFSample dat.filter dat.count observer  
## 1 MControl1 (0,1) 20-50 11-17-2016 11-17-2016 11-24-2016 GC  
## 2 MControl1 (0,1) 50-100 11-17-2016 11-17-2016 11-24-2016 GC  
## 3 MControl1 (0,1) 100-500 11-17-2016 11-17-2016 11-24-2016 GC  
## 4 MControl1 (0,1) 500-1000 11-17-2016 11-17-2016 11-24-2016 GC  
## 5 MControl1 (0,1) 1000-5000 11-17-2016 11-17-2016 11-24-2016 GC  
## 6 MControl2 (0,1) 20-50 11-17-2016 11-17-2016 11-24-2016 GC  
## variable value  
## 1 red.fib 0  
## 2 red.fib 0  
## 3 red.fib 0  
## 4 red.fib 0  
## 5 red.fib 0  
## 6 red.fib 0

Now sum by totals across all samples and particles sizes to get an idea of what we’re working with.

cat.sums.blanks <- aggregate(depblanks3$value ~   
 depblanks3$variable, FUN = sum) # sums everything  
print(cat.sums.blanks)

## depblanks3$variable depblanks3$value  
## 1 red.fib 0  
## 2 yell.fib 0  
## 3 green.fib 0  
## 4 turq.fib 0  
## 5 blu.fib 3  
## 6 purp.fib 0  
## 7 gray.fib 0  
## 8 clear.fib 33  
## 9 pink.fib 1  
## 10 brown.fib 0  
## 11 orang.fib 0  
## 12 black.fib 0  
## 13 tot.fib 37  
## 14 red.frag 0  
## 15 yell.frag 0  
## 16 green.frag 0  
## 17 turq.frag 0  
## 18 blu.frag 1  
## 19 purp.frag 0  
## 20 gray.frag 0  
## 21 clear.frag 0  
## 22 pink.frag 0  
## 23 brown.frag 0  
## 24 orang.frag 0  
## 25 black.frag 0  
## 26 whit.spher 0  
## 27 clear.spher 0  
## 28 tot.frag.spher 1  
## 29 tot.part 1

So we can remove the red, yellow, green turquoise, purple, gray, brown, orange, and black fibre categories, and all other categories other than blue fragements.

depblanks4 <- subset(depblanks3, c(variable == 'blu.fib' |   
 variable == 'clear.fib' | variable == 'pink.fib' |  
 variable == 'blu.frag')) # keep only these rows  
  
depblanks4$variable <- as.character(depblanks4$variable)  
depblanks4$variable <- as.factor(depblanks4$variable) # gets rid of those names  
  
summary(depblanks4$variable) # view

## blu.fib blu.frag clear.fib pink.fib   
## 45 45 45 45

Let’s make the category names a bit more useful for plotting

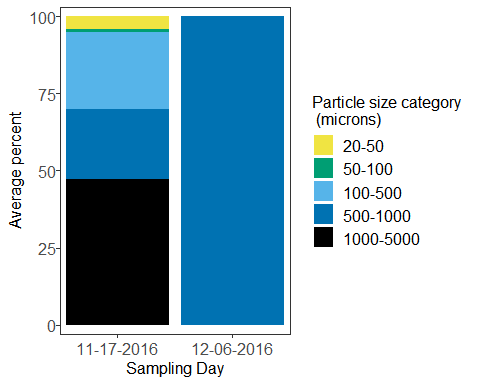
depblanks4$variable <- mapvalues(depblanks4$variable,  
 from = c(levels(depblanks4$variable)),   
 to = c('Blue fibres', 'Blue fragments',   
 'Clear fibres', 'Pink fibres'))  
  
## And reorder them  
depblanks4$variable <- factor(depblanks4$variable,   
 levels = c('Blue fibres', 'Clear fibres',  
 'Pink fibres', 'Blue fragments'))  
  
summary(depblanks4$variable)

## Blue fibres Clear fibres Pink fibres Blue fragments   
## 45 45 45 45

Next let’s make plots of the distribution of sizes and particle types and colours.

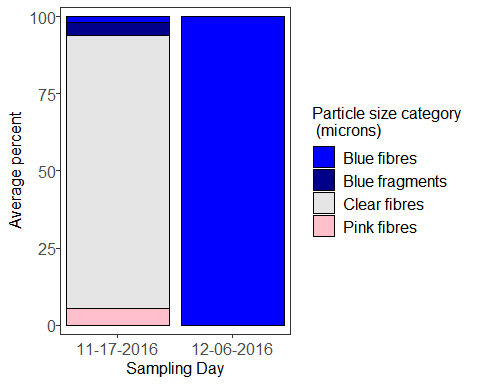
Starting with size

## First we need to make a new data frame of sums according to each size category  
  
blankssizesums <- ddply(depblanks4, c('id', 'size.cat', 'DOFSample',   
 'dat.filter', 'dat.count',   
 'observer'), summarize, sum = sum(value))  
  
  
  
## Now calculate proportions of each particle size category present in each   
## sample and take the means for each sampling day  
  
blanksprop <- ddply(blankssizesums, 'id', transform, prop = sum/sum(sum))  
  
blanksprop2 <- subset(blanksprop, !(is.na(blanksprop["prop"]))) # remove any NAs  
  
blanksprop2$size.cat <- as.character(blanksprop2$size.cat)  
blanksprop2$size.cat <- as.factor(blanksprop2$size.cat)  
  
blanksprop3 <- ddply(blanksprop2, c("DOFSample", "size.cat"), summarize,  
 mean = mean(prop))  
  
## Put the size categories in order  
  
blanksprop3$size.cat <- factor(blanksprop3$size.cat,   
 levels = c("10-20", "20-50", "50-100",   
 "100-500", "500-1000",   
 "1000-5000"))  
  
## Now plot  
  
ggplot(blanksprop3, aes(x=DOFSample, y=100\*mean, fill = size.cat)) +   
 geom\_col(size = 1) +   
 xlab('Sampling Day') +   
 ylab('Average percent') +  
 guides(fill=guide\_legend(title="Particle size category \n (microns)")) +  
 theme\_bw() +  
 scale\_fill\_manual(values = c("#F0E442", "#009E73",   
 "#56B4E9", "#0072B2", "black")) +  
 theme(legend.text = element\_text(size=12), text = element\_text(size=12),   
 panel.spacing = unit(1, "lines"),   
 axis.text.x = element\_text(size = 12),  
 axis.text.y = element\_text(size = 12,   
 margin = margin(t=0, r=0, b=0, l=2)),  
 strip.background = element\_blank(),   
 panel.grid.major = element\_blank(),   
 panel.grid.minor = element\_blank(),  
 strip.placement = "outside") +  
 scale\_x\_discrete(expand = expand\_scale(0,0.5)) +   
 scale\_y\_continuous(expand = expand\_scale(0.03,0))



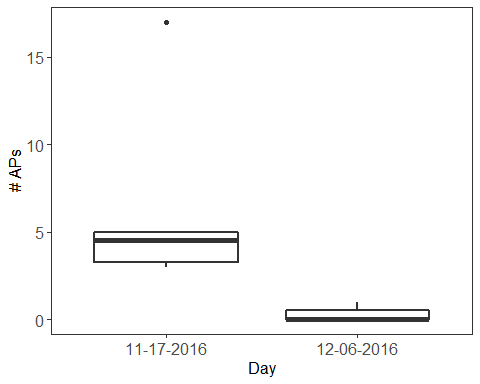
Now with particle type/colour

## First we need to make a new data frame of sums according to each particle  
## type/colour  
  
blankstypesums <- ddply(depblanks4, c('id', 'DOFSample', 'dat.filter',   
 'dat.count', 'observer', 'variable'),   
 summarize, sum = sum(value))  
  
  
  
## Now calculate proportions of each particle type/colour present in each   
## sample and take the means for each sampling day  
  
blankstypeprop <- ddply(blankstypesums, 'id', transform, prop = sum/sum(sum))  
  
blankstypeprop2 <- subset(blankstypeprop, !(is.na(blankstypeprop["prop"]))) # remove any NAs  
  
blankstypeprop2$variable <- as.character(blankstypeprop2$variable)  
blankstypeprop2$variable <- as.factor(blankstypeprop2$variable)  
  
blankstypeprop3 <- ddply(blankstypeprop2, c("DOFSample", "variable"), summarize,  
 mean = mean(prop))  
  
## Now plot  
  
ggplot(blankstypeprop3, aes(x=DOFSample, y=100\*mean, fill = variable)) +   
 geom\_col(size = 0.5, colour = 'black') +   
 xlab('Sampling Day') +   
 ylab('Average percent') +  
 guides(fill=guide\_legend(title="Particle size category \n (microns)")) +  
 theme\_bw() +  
 scale\_fill\_manual(values = c('blue', 'blue4', 'grey90', 'pink')) +  
 theme(legend.text = element\_text(size=12), text = element\_text(size=12),   
 panel.spacing = unit(1, "lines"),   
 axis.text.x = element\_text(size = 12),  
 axis.text.y = element\_text(size = 12,   
 margin = margin(t=0, r=0, b=0, l=2)),  
 strip.background = element\_blank(),   
 panel.grid.major = element\_blank(),   
 panel.grid.minor = element\_blank(),  
 strip.placement = "outside") +  
 scale\_x\_discrete(expand = expand\_scale(0,0.5)) +   
 scale\_y\_continuous(expand = expand\_scale(0.03,0))



Take total sums and plot the results.

totalblanksums <- ddply(depblanks4, c('id', 'observer', 'dat.filter', 'DOFSample'),  
 summarise, sum = sum(value)) # adds all the counts together  
  
## Plot according to count per filtration day  
  
ggplot(totalblanksums, aes(x=DOFSample, y=sum)) +   
 geom\_boxplot(position=position\_dodge(), size=1) +   
 xlab('Day') +   
 ylab('# APs') +  
 guides(fill=FALSE) +  
 theme\_bw() +  
 theme(text = element\_text(size=12),   
 axis.text.x = element\_text(size = 12),  
 axis.text.y = element\_text(size = 12),  
 strip.background = element\_blank(),  
 panel.grid.major = element\_blank(),   
 panel.grid.minor = element\_blank())



Now do the same summary stats for the blanks

## Means, range, and SD  
  
with(totalblanksums, tapply(sum, DOFSample, mean)) # takes mean for each day

## 11-17-2016 12-06-2016   
## 6.1666667 0.3333333

with(totalblanksums, tapply(sum, DOFSample, min)) # takes minimum for each day

## 11-17-2016 12-06-2016   
## 3 0

with(totalblanksums, tapply(sum, DOFSample, max)) # takes maximum for each day

## 11-17-2016 12-06-2016   
## 17 1

with(totalblanksums, tapply(sum,   
 DOFSample, sd)) # takes standard deviation for each day

## 11-17-2016 12-06-2016   
## 5.3820690 0.5773503

## Proportions of particles sizes  
  
sum(depblanks4$value[depblanks4$size.cat == '10-20'])/  
 sum(depblanks4$value)\*100 # 0%

## [1] 0

sum(depblanks4$value[depblanks4$size.cat == '20-50'])/  
 sum(depblanks4$value)\*100 # 2.63 %

## [1] 2.631579

sum(depblanks4$value[depblanks4$size.cat == '50-100'])/  
 sum(depblanks4$value)\*100 # 2.63 %

## [1] 2.631579

sum(depblanks4$value[depblanks4$size.cat == '100-500'])/  
 sum(depblanks4$value)\*100 # 42.11 %

## [1] 42.10526

sum(depblanks4$value[depblanks4$size.cat == '500-1000'])/  
 sum(depblanks4$value)\*100 # 23.68 %

## [1] 23.68421

sum(depblanks4$value[depblanks4$size.cat == '1000-5000'])/  
 sum(depblanks4$value)\*100 # 28.95 %

## [1] 28.94737

## This provides the percentage of total particles found accounted for by each   
## colour/type of particles  
blanksummarytable <- with(depblanks4, tapply(value, variable, sum))  
blanksummarytable <- blanksummarytable/sum(depblanks4$value)\*100  
print(blanksummarytable)

## Blue fibres Clear fibres Pink fibres Blue fragments   
## 7.894737 86.842105 2.631579 2.631579

Now load in the water sample data.

depwater <-   
 read.csv("~/GRAD school/ACRDP/Depuration/Data/Analysis/depwater.csv")  
  
head(depwater)

## id day size.cat Table Tank  
## 1 Water Control Control 1 Control twentyto50 Control n/a  
## 2 Water Control Control 1 Control fiftyto100 Control n/a  
## 3 Water Control Control 1 Control onehundredto500 Control n/a  
## 4 Water Control Control 1 Control fivehundredto1000 Control n/a  
## 5 Water Control Control 1 Control onethousandto5000 Control n/a  
## 6 Water Control Control 2 Control twentyto50 Control n/a  
## DOFSample time.in.oven time.out.oven dat.filter dat.count  
## 1 12-13-2017 2017-01-09/17:00 2017-01-17/09:30 12-13-2017 12-20-2017  
## 2 12-13-2017 2017-01-09/17:00 2017-01-17/09:30 12-13-2017 12-20-2017  
## 3 12-13-2017 2017-01-09/17:00 2017-01-17/09:30 12-13-2017 12-20-2017  
## 4 12-13-2017 2017-01-09/17:00 2017-01-17/09:30 12-13-2017 12-20-2017  
## 5 12-13-2017 2017-01-09/17:00 2017-01-17/09:30 12-13-2017 12-20-2017  
## 6 12-13-2017 2017-01-09/17:00 2017-01-17/09:30 12-13-2017 12-20-2017  
## observer red.fib yell.fib green.fib turq.fib blu.fib purp.fib gray.fib  
## 1 MD 0 0 0 0 0 0 0  
## 2 MD 0 0 0 0 0 0 0  
## 3 MD 0 0 0 0 0 0 0  
## 4 MD 0 0 0 0 0 0 0  
## 5 MD 0 0 0 0 0 0 0  
## 6 MD 0 0 0 0 0 0 0  
## clear.fib pink.fib brown.fib orang.fib black.fib tot.fib red.frag  
## 1 0 0 0 0 0 0 0  
## 2 0 0 0 0 0 0 0  
## 3 0 0 0 0 0 0 0  
## 4 0 0 0 0 0 0 0  
## 5 0 0 0 0 0 0 0  
## 6 0 0 0 0 0 0 0  
## yell.frag green.frag turq.frag blu.frag purp.frag gray.frag clear.frag  
## 1 0 0 0 0 0 0 0  
## 2 0 0 0 0 0 0 0  
## 3 0 0 0 0 0 0 0  
## 4 0 0 0 0 0 0 0  
## 5 0 0 0 0 0 0 0  
## 6 0 0 0 0 0 0 0  
## pink.frag brown.frag orang.frag black.frag whit.spher clear.spher  
## 1 0 0 0 0 0 0  
## 2 0 0 0 0 0 0  
## 3 0 0 0 0 0 0  
## 4 0 0 0 0 0 0  
## 5 0 0 0 0 0 0  
## 6 0 0 0 0 0 0  
## tot.frag.spher tot.part  
## 1 0 0  
## 2 0 0  
## 3 0 0  
## 4 0 0  
## 5 0 0  
## 6 0 0

depwater$id <- as.factor(depwater$id)  
depwater$day <- as.factor(depwater$day)  
depwater$size.cat <- as.factor(depwater$size.cat)  
depwater$Table <- as.factor(depwater$Table)  
depwater$Tank <- as.factor(depwater$Tank)  
depwater$observer <- as.factor(depwater$observer)

Now we make a new dataframe to work with and rename size categories.

depwater2 <- depwater  
  
depwater2$size.cat <- factor(depwater2$size.cat,   
 levels=c('twentyto50', 'fiftyto100',  
 'onehundredto500', 'fivehundredto1000',   
 'onethousandto5000')) # reorder  
  
  
depwater2$size.cat <- mapvalues(depwater2$size.cat,   
 from = c('twentyto50', 'fiftyto100',   
 'onehundredto500',   
 'fivehundredto1000',   
 'onethousandto5000'),   
 to = c('20-50', '50-100', '100-500',  
 '500-1000', '1000-5000')) # rename  
summary(depwater2$size.cat) # view

## 20-50 50-100 100-500 500-1000 1000-5000   
## 23 23 23 23 23

Again, we melt the dataframe so that the particle types are organized by row instead of by column.

depwater3 <- melt(depwater2,  
 id.vars = c('id', 'day', 'size.cat', 'Table', 'Tank',   
 'DOFSample', 'time.in.oven', 'time.out.oven',   
 'dat.filter', 'dat.count', 'observer'))  
  
## Take out unnecessary columns  
  
depwater3 <- depwater3[c('id', 'day', 'size.cat', 'Table', 'Tank', 'DOFSample',   
 'dat.filter', 'dat.count', 'observer', 'variable',   
 'value')]  
  
head(depwater3) # now 'variable' is particle category and 'value is the count'

## id day size.cat Table Tank DOFSample  
## 1 Water Control Control 1 Control 20-50 Control n/a 12-13-2017  
## 2 Water Control Control 1 Control 50-100 Control n/a 12-13-2017  
## 3 Water Control Control 1 Control 100-500 Control n/a 12-13-2017  
## 4 Water Control Control 1 Control 500-1000 Control n/a 12-13-2017  
## 5 Water Control Control 1 Control 1000-5000 Control n/a 12-13-2017  
## 6 Water Control Control 2 Control 20-50 Control n/a 12-13-2017  
## dat.filter dat.count observer variable value  
## 1 12-13-2017 12-20-2017 MD red.fib 0  
## 2 12-13-2017 12-20-2017 MD red.fib 0  
## 3 12-13-2017 12-20-2017 MD red.fib 0  
## 4 12-13-2017 12-20-2017 MD red.fib 0  
## 5 12-13-2017 12-20-2017 MD red.fib 0  
## 6 12-13-2017 12-20-2017 MD red.fib 0

Now sum by totals across all samples and particles sizes to get an idea of what we’re working with.

cat.sums.blanks <- aggregate(depwater3$value ~   
 depwater3$variable, FUN = sum) # sums everything  
print(cat.sums.blanks)

## depwater3$variable depwater3$value  
## 1 red.fib 3  
## 2 yell.fib 0  
## 3 green.fib 0  
## 4 turq.fib 1  
## 5 blu.fib 7  
## 6 purp.fib 0  
## 7 gray.fib 0  
## 8 clear.fib 37  
## 9 pink.fib 0  
## 10 brown.fib 0  
## 11 orang.fib 0  
## 12 black.fib 0  
## 13 tot.fib 48  
## 14 red.frag 0  
## 15 yell.frag 0  
## 16 green.frag 0  
## 17 turq.frag 0  
## 18 blu.frag 0  
## 19 purp.frag 0  
## 20 gray.frag 0  
## 21 clear.frag 0  
## 22 pink.frag 0  
## 23 brown.frag 0  
## 24 orang.frag 0  
## 25 black.frag 0  
## 26 whit.spher 0  
## 27 clear.spher 0  
## 28 tot.frag.spher 0  
## 29 tot.part 0

So we can remove the yellow, green, purple, gray, pink, brown, orange, and black fibre categories, and all other categories as well.

depwater4 <- subset(depwater3, c(variable == 'red.fib' |   
 variable == 'turq.fib' | variable == 'blu.fib' |  
 variable == 'clear.fib')) # keep only these rows  
  
depwater4$variable <- as.character(depwater4$variable)  
depwater4$variable <- as.factor(depwater4$variable) # gets rid of those names  
  
summary(depwater4$variable) # view

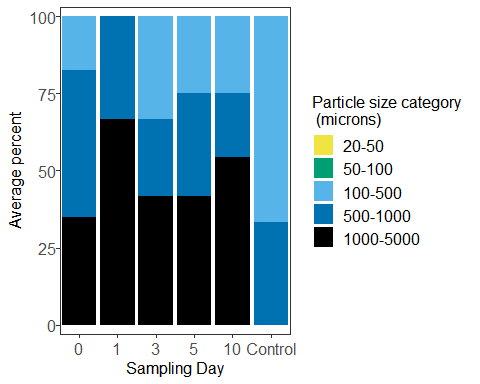
## blu.fib clear.fib red.fib turq.fib   
## 115 115 115 115

Let’s make the category names a bit more useful for plotting

depwater4$variable <- mapvalues(depwater4$variable,  
 from = c(levels(depwater4$variable)),   
 to = c('Blue fibres', 'Clear fibres',   
 'Red fibres', 'Turquoise fibres'))  
  
## And reorder them  
depwater4$variable <- factor(depwater4$variable,   
 levels = c('Blue fibres', 'Clear fibres',  
 'Red fibres', 'Turquoise fibres'))  
  
summary(depwater4$variable)

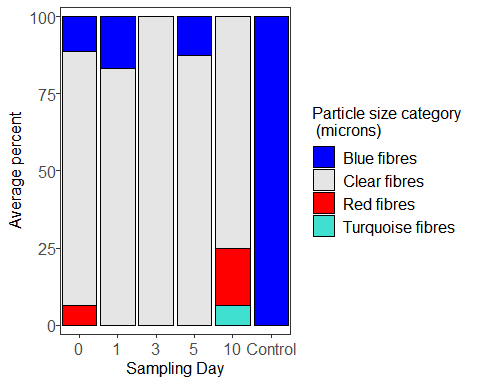
## Blue fibres Clear fibres Red fibres Turquoise fibres   
## 115 115 115 115

## First we need to make a new data frame of sums according to each size category  
  
depwater4$day <- factor(depwater4$day,   
 levels =   
 c('0', '1', '3', '5', '10', 'Control')) # put in order  
  
watersizesums <- ddply(depwater4, c('id', 'day', 'size.cat', 'Table', 'Tank',   
 'DOFSample', 'dat.filter', 'dat.count',   
 'observer'), summarize, sum = sum(value))  
  
  
  
## Now calculate proportions of each particle size category present in each   
## sample and take the means for each sampling day  
  
waterprop <- ddply(watersizesums, 'id', transform, prop = sum/sum(sum))  
  
waterprop2 <- subset(waterprop, !(is.na(waterprop["prop"]))) # remove any NAs  
  
waterprop2$size.cat <- as.character(waterprop2$size.cat)  
waterprop2$size.cat <- as.factor(waterprop2$size.cat)  
  
waterprop3 <- ddply(waterprop2, c("day", "size.cat"), summarize,  
 mean = mean(prop))  
  
## Put the size categories in order  
  
waterprop3$size.cat <- factor(waterprop3$size.cat,   
 levels = c("10-20", "20-50", "50-100",   
 "100-500", "500-1000",   
 "1000-5000"))  
  
## Now plot  
  
ggplot(waterprop3, aes(x=day, y=100\*mean, fill = size.cat)) +   
 geom\_col(size = 1) +   
 xlab('Sampling Day') +   
 ylab('Average percent') +  
 guides(fill=guide\_legend(title="Particle size category \n (microns)")) +  
 theme\_bw() +  
 scale\_fill\_manual(values = c("#F0E442", "#009E73",   
 "#56B4E9", "#0072B2", "black")) +  
 theme(legend.text = element\_text(size=12), text = element\_text(size=12),   
 panel.spacing = unit(1, "lines"),   
 axis.text.x = element\_text(size = 12),  
 axis.text.y = element\_text(size = 12,   
 margin = margin(t=0, r=0, b=0, l=2)),  
 strip.background = element\_blank(),   
 panel.grid.major = element\_blank(),   
 panel.grid.minor = element\_blank(),  
 strip.placement = "outside") +  
 scale\_x\_discrete(expand = expand\_scale(0,0.5)) +   
 scale\_y\_continuous(expand = expand\_scale(0.03,0))



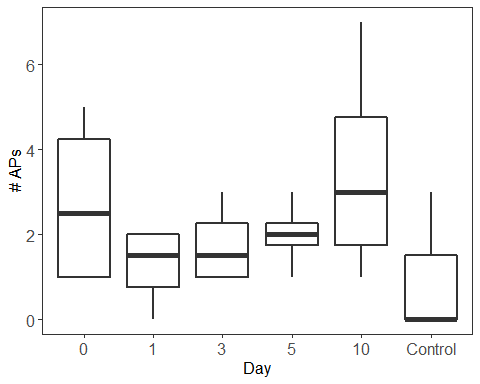
Now with particle type/colour

## First we need to make a new data frame of sums according to each particle  
## type/colour  
  
watertypesums <- ddply(depwater4, c('id', 'day', 'Table', 'Tank',   
 'DOFSample', 'dat.filter', 'dat.count',   
 'observer', 'variable'),   
 summarize, sum = sum(value))  
  
  
  
## Now calculate proportions of each particle type/colour present in each   
## sample and take the means for each sampling day  
  
watertypeprop <- ddply(watertypesums, 'id', transform, prop = sum/sum(sum))  
  
watertypeprop2 <- subset(watertypeprop, !(is.na(watertypeprop["prop"]))) # remove any NAs  
  
watertypeprop2$variable <- as.character(watertypeprop2$variable)  
watertypeprop2$variable <- as.factor(watertypeprop2$variable)  
  
watertypeprop3 <- ddply(watertypeprop2, c("day", "variable"), summarize,  
 mean = mean(prop))  
  
## Now plot  
  
ggplot(watertypeprop3, aes(x=day, y=100\*mean, fill = variable)) +   
 geom\_col(size = 0.5, colour = 'black') +   
 xlab('Sampling Day') +   
 ylab('Average percent') +  
 guides(fill=guide\_legend(title="Particle size category \n (microns)")) +  
 theme\_bw() +  
 scale\_fill\_manual(values = c('blue', 'grey90', 'red', 'turquoise')) +  
 theme(legend.text = element\_text(size=12), text = element\_text(size=12),   
 panel.spacing = unit(1, "lines"),   
 axis.text.x = element\_text(size = 12),  
 axis.text.y = element\_text(size = 12,   
 margin = margin(t=0, r=0, b=0, l=2)),  
 strip.background = element\_blank(),   
 panel.grid.major = element\_blank(),   
 panel.grid.minor = element\_blank(),  
 strip.placement = "outside") +  
 scale\_x\_discrete(expand = expand\_scale(0,0.5)) +   
 scale\_y\_continuous(expand = expand\_scale(0.03,0))



Take total sums and plot the results.

totalwatersums <- ddply(depwater4, c('id', 'day', 'Table', 'Tank', 'observer',   
 'dat.filter'),  
 summarise, sum = sum(value)) # adds all the counts together  
  
## Plot according to count per filtration day  
  
ggplot(totalwatersums, aes(x=day, y=sum)) +   
 geom\_boxplot(position=position\_dodge(), size=1) +   
 xlab('Day') +  
 ylab('# APs') +  
 guides(fill=FALSE) +  
 theme\_bw() +  
 theme(text = element\_text(size=12),   
 axis.text.x = element\_text(size = 12),  
 axis.text.y = element\_text(size = 12),  
 strip.background = element\_blank(),  
 panel.grid.major = element\_blank(),   
 panel.grid.minor = element\_blank())



Now run the same summary stats for the water samples.

## Means, range, and SD  
  
with(totalwatersums, tapply(sum, day, mean)) # takes mean for each day

## 0 1 3 5 10 Control   
## 2.75 1.25 1.75 2.00 3.50 1.00

with(totalwatersums, tapply(sum, day, min)) # takes minimum for each day

## 0 1 3 5 10 Control   
## 1 0 1 1 1 0

with(totalwatersums, tapply(sum, day, max)) # takes maximum for each day

## 0 1 3 5 10 Control   
## 5 2 3 3 7 3

with(totalwatersums, tapply(sum,   
 day, sd)) # takes standard deviation for each day

## 0 1 3 5 10 Control   
## 2.0615528 0.9574271 0.9574271 0.8164966 2.6457513 1.7320508

## Proportions of particles sizes  
  
sum(depwater4$value[depwater4$size.cat == '10-20'])/  
 sum(depwater4$value)\*100 # 0%

## [1] 0

sum(depwater4$value[depwater4$size.cat == '20-50'])/  
 sum(depwater4$value)\*100 # 0 %

## [1] 0

sum(depwater4$value[depwater4$size.cat == '50-100'])/  
 sum(depwater4$value)\*100 # 0 %

## [1] 0

sum(depwater4$value[depwater4$size.cat == '100-500'])/  
 sum(depwater4$value)\*100 # 20.83 %

## [1] 20.83333

sum(depwater4$value[depwater4$size.cat == '500-1000'])/  
 sum(depwater4$value)\*100 # 31.25 %

## [1] 31.25

sum(depwater4$value[depwater4$size.cat == '1000-5000'])/  
 sum(depwater4$value)\*100 # 47.92 %

## [1] 47.91667

## This provides the percentage of total particles found accounted for by each   
## colour/type of particles  
watersummarytable <- with(depwater4, tapply(value, variable, sum))  
watersummarytable <- watersummarytable/sum(depwater4$value)\*100  
print(watersummarytable)

## Blue fibres Clear fibres Red fibres Turquoise fibres   
## 14.583333 77.083333 6.250000 2.083333

## MODELLING

Now fit a generalized linear mixed model (GLMM) to the oyster data using the package glmmADMB to determine differences over time. We’ll specify AP count as the response variable, sampling day as the predictor, and tank nested within table as the random, or mixed, effects. You’ll notice that sample day is specified twice. The second time is to set it up as the intercept for the random effect, which will allow the linear relationship between AP sum and sampling day to vary in both slope and intercept according separately within each tank, within each table. This accounts for any the variation that may have been caused by any variables that we did not account for and may have varied somehow in different tanks and or/tables.

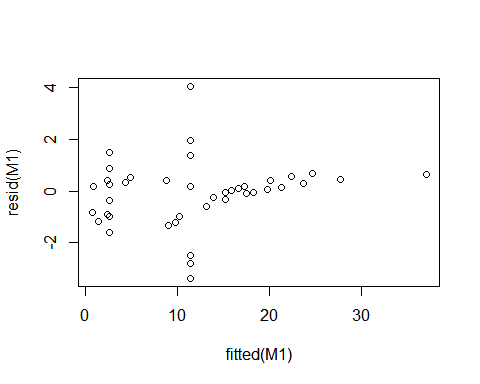
M1 <- glmmadmb(sum ~ sampleday + (sampleday | Table/Tank), family = 'poisson',   
 data = totalsums) ## Fits a GLMM  
  
summary(M1) # model output

##   
## Call:  
## glmmadmb(formula = sum ~ sampleday + (sampleday | Table/Tank),   
## data = totalsums, family = "poisson")  
##   
## AIC: 384.2   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 2.4304 0.0895 27.16 < 2e-16 \*\*\*  
## sampleday1 0.5013 0.1764 2.84 0.0045 \*\*   
## sampleday3 0.3146 0.1696 1.86 0.0636 .   
## sampleday5 -1.4749 0.2156 -6.84 7.8e-12 \*\*\*  
## sampleday10 -1.6250 0.5986 -2.71 0.0066 \*\*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Number of observations: total=51, Table=5, Table:Tank=11   
## Random effect variance(s):

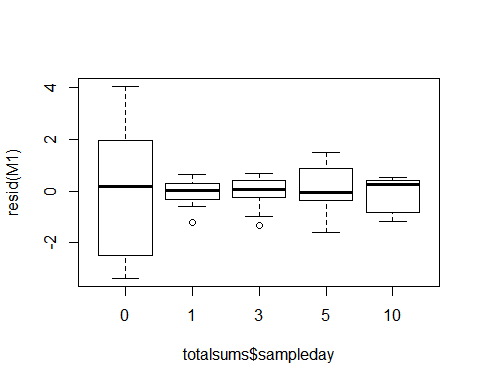
## Warning in .local(x, sigma, ...): 'sigma' and 'rdig' arguments are present  
## for compatibility only: ignored

## Group=Table  
## Variance StdDev  
## (Intercept) 1.127e-07 0.0003356  
## sampleday1 3.806e-06 0.0019509  
## sampleday3 2.116e-05 0.0045997  
## sampleday5 1.355e-06 0.0011641  
## sampleday10 8.624e-01 0.9286334  
## Group=Table:Tank  
## Variance StdDev  
## (Intercept) 7.395e-06 0.002719  
## sampleday1 1.717e-01 0.414355  
## sampleday3 1.356e-01 0.368171  
## sampleday5 2.560e-05 0.005060  
## sampleday10 2.814e-01 0.530471  
##   
##   
## Log-likelihood: -177.092

plot(resid(M1) ~ fitted(M1)) # plot residuals vs. fitted



plot(resid(M1) ~ totalsums$sampleday) # plot residuals vs. predictor



Looking at the plot of residuals vs. fitted and residuals vs. sampe day, we can see that the variance of the residuals varies according to the AP count and the sampling day. This violates the assumption of homogeneity of variance.

Let’s try fitting the same model structure, but assuming that the response variable (AP count) varies at each value of x (sampling day) according to a negative binomial distribution.

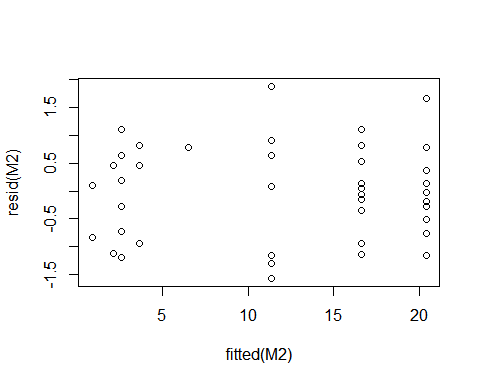
M2 <- glmmadmb(sum ~ sampleday + (sampleday | Table/Tank), family = 'nbinom',   
 data = totalsums) ## Fits a NB GLMM  
  
summary(M2) # model output

##   
## Call:  
## glmmadmb(formula = sum ~ sampleday + (sampleday | Table/Tank),   
## data = totalsums, family = "nbinom")  
##   
## AIC: 338.4   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 2.430 0.192 12.65 < 2e-16 \*\*\*  
## sampleday1 0.585 0.271 2.16 0.031 \*   
## sampleday3 0.379 0.274 1.39 0.166   
## sampleday5 -1.475 0.327 -4.51 6.6e-06 \*\*\*  
## sampleday10 -1.485 0.576 -2.58 0.010 \*\*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Number of observations: total=51, Table=5, Table:Tank=11   
## Random effect variance(s):

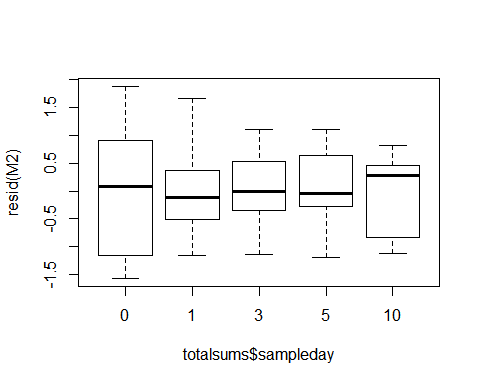
## Warning in .local(x, sigma, ...): 'sigma' and 'rdig' arguments are present  
## for compatibility only: ignored

## Group=Table  
## Variance StdDev  
## (Intercept) 1.137e-07 0.0003372  
## sampleday1 1.399e-06 0.0011827  
## sampleday3 2.064e-04 0.0143656  
## sampleday5 5.603e-06 0.0023670  
## sampleday10 8.160e-01 0.9033438  
## Group=Table:Tank  
## Variance StdDev  
## (Intercept) 2.438e-05 0.0049374  
## sampleday1 3.457e-07 0.0005879  
## sampleday3 5.310e-07 0.0007287  
## sampleday5 1.819e-05 0.0042649  
## sampleday10 3.480e-04 0.0186561  
##   
## Negative binomial dispersion parameter: 3.1455 (std. err.: 0.9495)  
##   
## Log-likelihood: -153.196

plot(resid(M2) ~ fitted(M2)) # plot residuals vs. fitted



plot(resid(M2) ~ totalsums$sampleday) # plot residuals vs. predictor



This model is a much better fit, and the variance is much more evenly distributed across the model predictions and across the predictor variable (sampling day).

Now let’s try pulling out the predictor (sampling day) and using the anova() function to compare the first model with the null models to determine whether sampling day significantly predicts AP concentration.

M3 <- glmmadmb(sum ~ (sampleday | Table/Tank), family = 'nbinom',   
 data = totalsums) ## Same structure except no predictor  
  
anova(M2, M3) # compare the null model with the previously fitted model

## Warning in anova.glmmadmb(M2, M3): rearranging models in order of  
## increasing complexity

## Analysis of Deviance Table  
##   
## Model 1: sum ~ 1  
## Model 2: sum ~ sampleday  
## NoPar LogLik Df Deviance Pr(>Chi)   
## 1 12 -164.23   
## 2 16 -153.20 4 22.076 0.0001936 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Sampling day is significant with a p-value of <0.001. We can now go into the model to determine which days are significantly different from each other. Looking back at the summar of M2 we can see that day 3 is the only time where the AP concentrations are not significantly different from day 0 (the reference level).

Now we reorder the levels in sampleday to vary the reference level and figure out which days are significantly different from which other days.

totalsums$sampleday <- relevel(totalsums$sampleday, '1')  
M2 <- glmmadmb(sum ~ sampleday + (sampleday | Table/Tank), family = 'nbinom',   
 data = totalsums) ## Fits a NB GLMM  
  
summary(M2) # model output

##   
## Call:  
## glmmadmb(formula = sum ~ sampleday + (sampleday | Table/Tank),   
## data = totalsums, family = "nbinom")  
##   
## AIC: 338.4   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 3.016 0.192 15.74 < 2e-16 \*\*\*  
## sampleday0 -0.585 0.271 -2.16 0.03103 \*   
## sampleday3 -0.206 0.273 -0.76 0.45018   
## sampleday5 -2.060 0.327 -6.30 3e-10 \*\*\*  
## sampleday10 -2.070 0.575 -3.60 0.00032 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Number of observations: total=51, Table=5, Table:Tank=11   
## Random effect variance(s):

## Warning in .local(x, sigma, ...): 'sigma' and 'rdig' arguments are present  
## for compatibility only: ignored

## Group=Table  
## Variance StdDev  
## (Intercept) 2.392e-07 0.0004891  
## sampleday0 3.162e-06 0.0017783  
## sampleday3 2.309e-06 0.0015195  
## sampleday5 3.129e-06 0.0017688  
## sampleday10 8.131e-01 0.9017261  
## Group=Table:Tank  
## Variance StdDev  
## (Intercept) 1.126e-07 0.0003355  
## sampleday0 3.162e-06 0.0017783  
## sampleday3 7.570e-06 0.0027514  
## sampleday5 2.792e-05 0.0052836  
## sampleday10 8.908e-06 0.0029847  
##   
## Negative binomial dispersion parameter: 3.1452 (std. err.: 0.94931)  
##   
## Log-likelihood: -153.194

Similarly, day 3 is the only day not significantly different from day 1.

totalsums$sampleday <- relevel(totalsums$sampleday, '3')  
M2 <- glmmadmb(sum ~ sampleday + (sampleday | Table/Tank), family = 'nbinom',   
 data = totalsums) ## Fits a NB GLMM  
  
summary(M2) # model output

##   
## Call:  
## glmmadmb(formula = sum ~ sampleday + (sampleday | Table/Tank),   
## data = totalsums, family = "nbinom")  
##   
## AIC: 338.4   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 2.809 0.194 14.45 < 2e-16 \*\*\*  
## sampleday1 0.206 0.273 0.76 0.4501   
## sampleday0 -0.379 0.273 -1.39 0.1657   
## sampleday5 -1.854 0.329 -5.64 1.7e-08 \*\*\*  
## sampleday10 -1.864 0.577 -3.23 0.0012 \*\*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Number of observations: total=51, Table=5, Table:Tank=11   
## Random effect variance(s):

## Warning in .local(x, sigma, ...): 'sigma' and 'rdig' arguments are present  
## for compatibility only: ignored

## Group=Table  
## Variance StdDev  
## (Intercept) 5.424e-07 0.0007365  
## sampleday1 2.776e-07 0.0005269  
## sampleday0 2.427e-05 0.0049269  
## sampleday5 1.252e-07 0.0003539  
## sampleday10 8.156e-01 0.9031002  
## Group=Table:Tank  
## Variance StdDev  
## (Intercept) 1.219e-07 0.0003492  
## sampleday1 2.306e-06 0.0015187  
## sampleday0 2.427e-05 0.0049269  
## sampleday5 1.469e-06 0.0012121  
## sampleday10 5.067e-05 0.0071184  
##   
## Negative binomial dispersion parameter: 3.1458 (std. err.: 0.94947)  
##   
## Log-likelihood: -153.194

Only days 5 and 10 are significantly different from day 3.

totalsums$sampleday <- relevel(totalsums$sampleday, '5')  
M2 <- glmmadmb(sum ~ sampleday + (sampleday | Table/Tank), family = 'nbinom',   
 data = totalsums) ## Fits a NB GLMM  
  
summary(M2) # model output

##   
## Call:  
## glmmadmb(formula = sum ~ sampleday + (sampleday | Table/Tank),   
## data = totalsums, family = "nbinom")  
##   
## AIC: 338.4   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 0.9555 0.2650 3.61 0.00031 \*\*\*  
## sampleday3 1.8539 0.3287 5.64 1.7e-08 \*\*\*  
## sampleday1 2.0600 0.3270 6.30 3.0e-10 \*\*\*  
## sampleday0 1.4749 0.3273 4.51 6.6e-06 \*\*\*  
## sampleday10 -0.0103 0.6045 -0.02 0.98635   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Number of observations: total=51, Table=5, Table:Tank=11   
## Random effect variance(s):

## Warning in .local(x, sigma, ...): 'sigma' and 'rdig' arguments are present  
## for compatibility only: ignored

## Group=Table  
## Variance StdDev  
## (Intercept) 7.576e-07 0.0008704  
## sampleday3 1.068e-05 0.0032686  
## sampleday1 1.130e-07 0.0003361  
## sampleday0 2.210e-07 0.0004701  
## sampleday10 8.163e-01 0.9035154  
## Group=Table:Tank  
## Variance StdDev  
## (Intercept) 1.962e-06 0.0014006  
## sampleday3 4.937e-07 0.0007026  
## sampleday1 3.239e-07 0.0005691  
## sampleday0 2.210e-07 0.0004701  
## sampleday10 1.636e-04 0.0127910  
##   
## Negative binomial dispersion parameter: 3.1458 (std. err.: 0.94948)  
##   
## Log-likelihood: -153.194

Day 10 is the only day not significantly different from day 5.

So in summary:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Day | 0 | 1 | 3 | 5 | 10 |
| 0 | x | sig | - | sig | sig |
| —- | — | — | — | — | —– |
| 1 | x | x | - | sig | sig |
| —- | — | — | — | — | —– |
| 3 | x | x | x | sig | sig |
| —- | — | — | — | — | —– |
| 5 | x | x | x | x | - |
| —- | — | — | — | — | —– |
| 10 | x | x | x | x | x |

Now let’s replot the data with letters used to indicate which days are significantly different from one another. Those with different letters are statistically different.

## First we have to get the days back in the right order  
  
totalsums$sampleday <- factor(totalsums$sampleday,   
 levels = c("0","1","3","5", "10"))  
  
## Now plot  
  
ggplot(totalsums, aes(x=sampleday , y=sum)) +   
 geom\_boxplot(position=position\_dodge(), size=1) +   
 xlab('Day') +   
 coord\_cartesian(ylim = c(0, 41)) +  
 ylab('# APs') +  
 guides(fill=FALSE) +  
 theme\_bw() +  
 theme(text = element\_text(size=12),   
 axis.text.x = element\_text(size = 12),  
 axis.text.y = element\_text(size = 12),  
 strip.background = element\_blank(),  
 panel.grid.major = element\_blank(),   
 panel.grid.minor = element\_blank()) +  
 annotate("text", x=1, y=27, label = "A") +   
 annotate("text", x=2, y=32, label = "B") +  
 annotate("text", x=3, y=30, label = "A,B") +  
 annotate("text", x=4, y=7, label = "C") +  
 annotate("text", x=5, y=12, label = "C")

