An introduction to R markdown

Chris B. Wall – chris.wall@hawaii.edu

9/18/2018

This is an R Markdown!

R markdown is great, and when I say great, **I mean REALLY great.**

When exporting your markdown you can... *(1) embed figures*
(2) view tables
(3) export models
(4) archive your data pipeline

You can write code and execute functions in a similar way to normal R script files. However, there are a few distinct differences between R scripts and R markdown scripts. Some of these differences relate to how code is executed. Other differences can be seen in the added benefit of integrating scripts with outputs and other embedded properties.

The benefit of using markdown with colleagues is they can actually see the script, the figures, the path of model selection, even the assumptions of ANOVA (QQ plots, etc) that normally are hidden behind the curtain. In short, **R markdown helps make your science repeatable, sharable, and data analysis coherent!**

Here are some common commands and examples of what Markdown can do for you. I reccommend visiting the `R Bookdown for a complete and exhaustive guide` https://bookdown.org/yihui/rmarkdown/ or the R markdown cheat sheet

Markdown basics

Common Commands

Notice the script at the very top of your markdown (this is not standard, but is customized)

---

output:
html_document:
code_folding: hide
toc: yes
toc_depth: 4
toc_float: yes ***

Let's go through this line by line...

- `html_document`: an option you originally set when you open the markdown. This is how your file will be exported.
- `code_folding: hide` will allow you to *show* or *hide* all code (option at top of file)
- `toc`: for table of contents on the left
- `toc_depth`: how long you will have table of contents before folding over
- `toc_float`: lets your contents move as you advance in your document.

Play around with these options, and see how they affect your code

The setup chunk

- `r setup`: This is your set up code and is a useful way to get around the fact that knitr will look for your .Rmd file and all files in the this directory. By setting the root.dir you can force knitr to look for files in the directory you specify and the folders within.

In the setup chunk (the first code chunk) you can set 'global' options, such as message/warning exclusion, or hiding results or outputs.
`knitr::opts_chunk$set(warning=FALSE, message=FALSE)`: This command here is requiring the package `knitr` in the code chunk ('set' for set up) and is saying to "make all warning and message FALSE" i.e., hide them from output.

- `include=FALSE`: this command makes your code absent from final html. It is executed but the code is hidden.
- `eval=FALSE`: this command allows you to show the code in your script within R studio, but not evaluate (or run) the code. It is effectively silent in your analysis and output html.
- `echo=FALSE`: this will show the output but not the code chunk... notice the difference.
- `results='hide'`: this will hide all results in a code chunk, or any returned results from your commands.

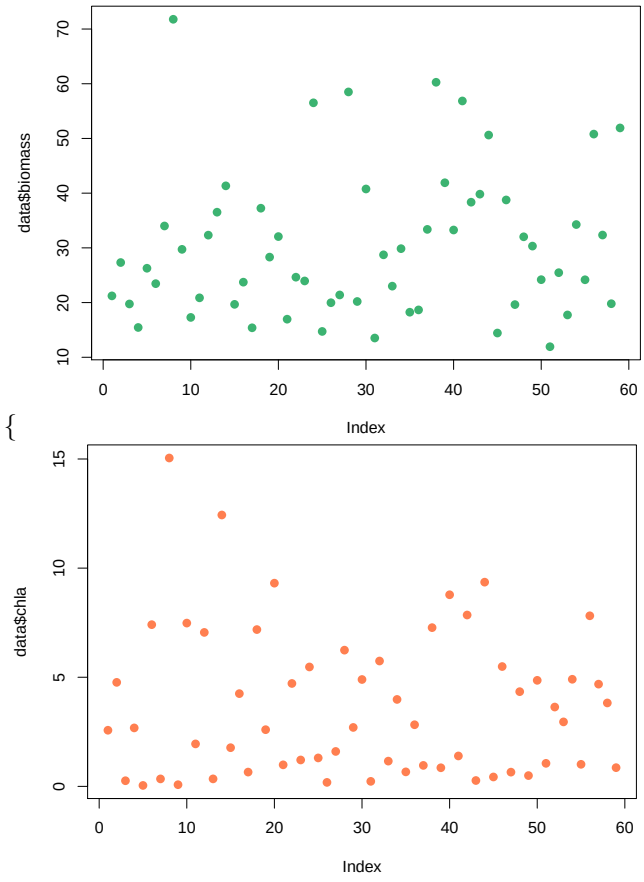
Figure formatting

- `fig.show=FALSE`: this will hide the figures you generate. This may be useful for some of those assumptions of ANOVA– you want to see it, but do you want all of them in your final archived datafile? Maybe not.
- `fig.width=5, fig.height=3`: The (graphical device) size of R plots in inches. This records your output figure as a graphical device using *knitr* and will write this to files. There are other ways to edit figure size from this intial dimension (*see below*). You can also specify the two (width and height using) `fig.dim`, as in the previous examle `fig.dim = c(5, 3)`
- `out.width` and `out.height`: These set the output of a figure in output document. This is best to scale the image realtive to document dimensions, such as `out.width= "50%"` would be 50% of page width.
- `fig.align`: gives you figure alignment as either: right, center, left.
- `dev`: graphical devices, typically `png` (html) or `pdf` for LaTex.
- `fig.cap`: caption your figure
- `fig.show`: options here control your figures. Setting to `'asis'` will show plots in the location they were generated (similar to R terminal–this is the default). If you set to `'hide'` then the figures are generated but not shown in your output file. Another option below is my favorite.
- `fig.show='hold'`: this is a really cool option. This option holds you plots and doesn't display until the end of the code chunk. You can use it to place multiple figures side-by-side as long as the `out.width` is called. For example, two plots side-by-side at `out.widh="50%"`

Now let's use some of these options to see how this code could be executed.

```
data<-read.csv("data/coral_data.csv")
par(mar = c(4, 4, 0.2, 0.1))
plot(data$biomass, pch=16, cex=1.2, col="mediumseagreen")
plot(data$chla, pch=16, cex=1.2, col="coral")
```

\begin{figure}



{

}

\caption{*Figure 1. Example of side-by-side plots at 50% with a 'fig hold' option*} \end{figure}

But something to consider... R scripts in code chunks operate a bit differently than classic R scripts. If you want to run your plot commands line-by-line you may need to include calls, such as `with()` or `par(new=T)` to specify that you do NOT want a new plot device to be made, but instead to keep working on the device you have opened.

```
# use results="hide" here to suppress messages associate with "dev.off" and dev.print to save figure

MC.df<-data[(data$Species=="MC"),] ## Montipora capitata alone
PC.df<-data[(data$Species=="PC"),] ## Porites compress alone

par(mar = c(4, 4, 3, 0.1))

plot(density(MC.df$biomass), lwd=2, col="darksalmon", xlim=c(0,100), main="Biomass in two coral species
lines(density(PC.df$biomass), lwd=2, col="darkslategray3", yaxt="n", ylab="", xaxt="n", xlab="")
legend("topright", legend=c(expression(italic("Montipora capitata")), expression(italic("Porites compres
```

```
dev.print(pdf, "figures/biomass in two species.pdf", encod="MacRoman", height=4, width=6) # export in d
dev.off()
```

Text formatting

It is useful to understand how to modify text and format headings. You can do this in a variety of ways.

- `line break`: this is executed by two spaces at the end of previous line, and a return

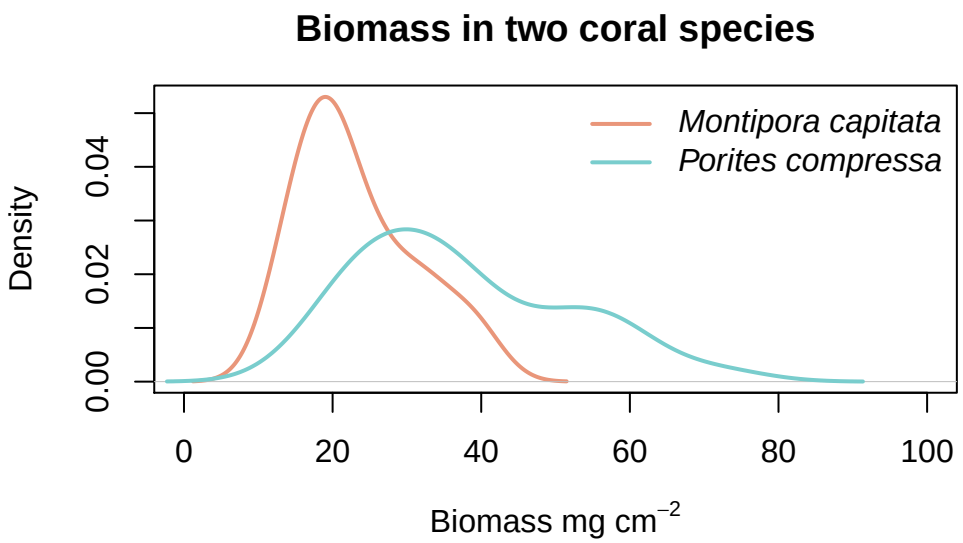


Figure 1: **Figure 2**. Tissue biomass in two coral species.

Figure 2: **Figure 3**. Pa'āni is *beautiful* (and **sassy**). Beware!

- `headings`: use `#` for headings, `#.... ####` and so on. # is largest and #### smallest heading

  `Italics` comes from two calls *italics* or *italics*; `bold` is done the same way **bold** and **bold**
- `Superscript` superscript^2 or `strikethroughs` ~~strikethrough~~ can also be useful text modifiers.
- `add links` to urls like this R studio link
- `endash` : –
- `emdash`: —
- `ellipsis`: ...
- `inline equation`: $A = \pi * r^2$ or $y = a + b * x$
- (a line can inserted with `***`)

  ***

  And don't forget about adding in figures to your markdown from files/images in your directory...

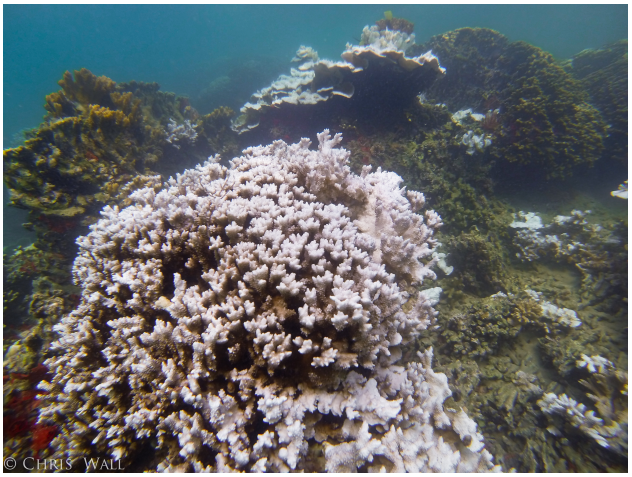


Figure 3: **Figure 4.** This is a figure of a bleached *Montipora capitata* coral during the 2014 bleaching event in Kāne'ohe Bay, O'ahu, Hawai'i. (PC: CB Wall)

Note in the code above I used the html codes for centering the image and the caption. You must leave a line between the first html call `<center>`, the caption and image code, and the end html code line of `</center>`.

Code chunks

```
## love your data and it will love you back
getwd()
```

```
## [1] "/Users/chriswall/Desktop/zenodo/Intro to Markdown"
```

Code chunks are where your code is executed. If you do not set the working directory in *setup* each code chunk will revert to its original directory, or where your .Rmd file lives. Below: This is a code chunk, and this is how you enter your data into R markdown. Note the `code chunks` always start with a ```` ```{r... ```` and ends with a `...}`

The code chunk below is an example of attaching the data (.csv), familiar to you R-heads. Since the data file is in the directory specified by `knitr::opts_knit$set(root.dir =...` above, you can reference the .csv easily.

```
################################################
# import data, observe structure
################################################

# data file is in the folder 'data', within main working directory
data<-read.csv("data/coral_data.csv")
names(data)
```

```
##  [1] "Time.point" "Period"     "Site"       "Species"    "Status"
##  [6] "Sample.ID"  "Pair"       "Depth.ft"   "biomass"    "chla"
```

Including figures

You can include figures from script output as results, or figures from files in your directory. First, let's see how you can add an image from a file to your markdown.

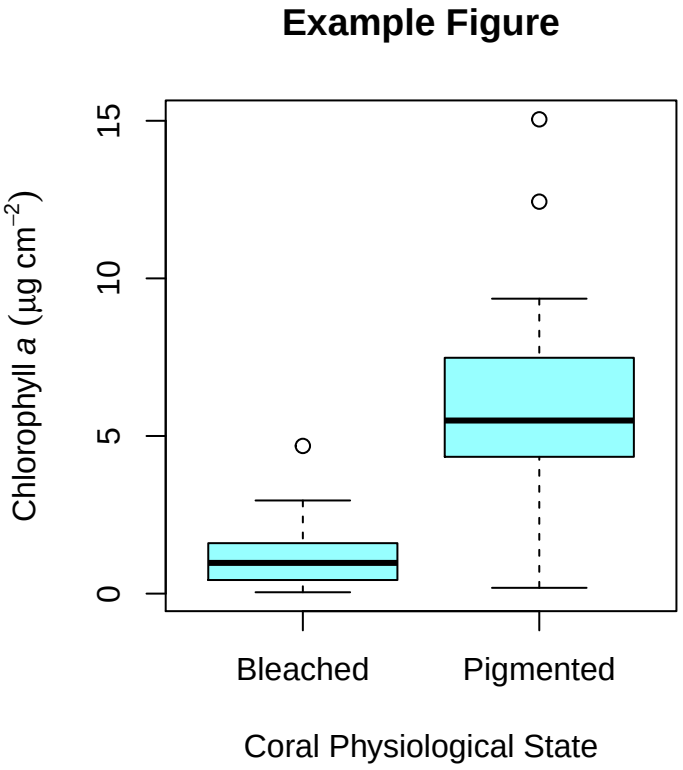


Figure 4: **Figure 5**. Chlorophyll a density. Caption set in code chunk setting

In this chunk we will make a figure of chlorophyll *a* concentrations cm^{-2}. Some of the syntax here should be familiar.

```
# message = FALSE to hide any messages with exporting the figure with dev.off(), or anything else that

par(mfrow=c(1,1), pty="sq")
Status.label<-c("Bleached", "Pigmented") # for x labels

plot(chla~Status, data=data, xaxt="n", col="darkslategray1",
     ylab=expression(paste("Chlorophyll", ~italic("a"), ~(mu*g~cm^-2), sep="")),
     xlab="Coral Physiological State",
     main="Example Figure")
axis(1, at=1:2, labels=Status.label) # this plots new labels on the x axis (axis = 1)
```

```
##### save the figure and export to directory? ####
# in this case, there is a file in my directory names 'figures' where I want plots to go

dev.copy(pdf, "figures/Chlorophyll.pdf", encod="MacRoman", height=5, width=5)
# height and width set here for the output figure
dev.off() # close the object
```

If you change export to the same location as your working directory and not a subfile "chlorophyll.pdf"‘ will do the job.

Examples

- see a code chunk in my markdown file, but it isn't executed if `eval=FALSE` is set

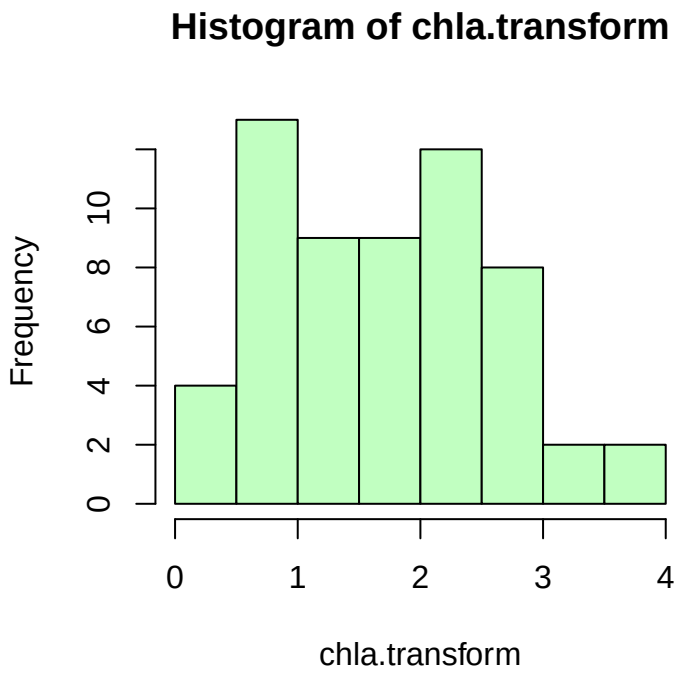


Figure 5: **Figure 6.** Boxplot of square-root transformed chlorophyll *a* data

```
mean(data$chla)
```

- execute the code and hide results `results='hide'`

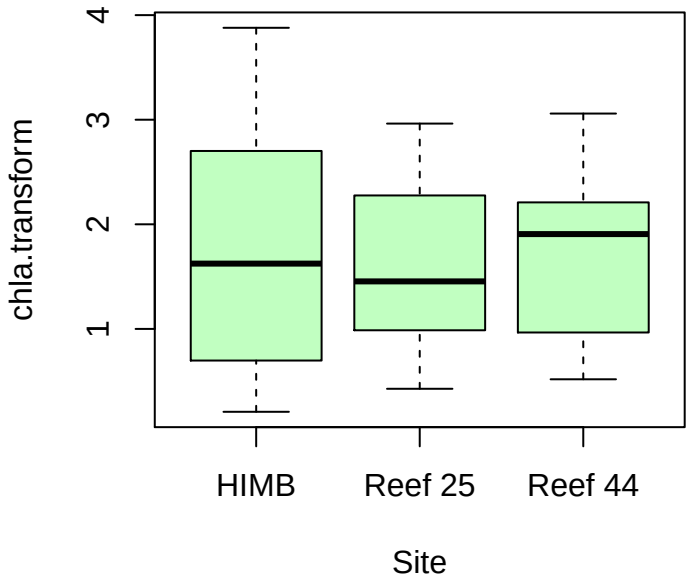
```
chla.transform<-sqrt(data$chla)
mean(chla.transform)
```

- execute and show code and SEE the figure I generate, but hide results (returned data) `results='hide', fig.height=4, fig.width=4, fig.align='center'`

```
mean(chla.transform)

# make figure 4 x 4 and center align
hist(chla.transform, data=data, col="darkseagreen1")
```

. . . or just print the figure with `echo=FALSE` sans the code



. . . or show NOTHING with `results='hide', fig.show=FALSE`

```
plot(chla.transform~Site, data=data, col="darkseagreen1")
```

Make a table of summary data

plot raw data 'as is'

```
knitr::kable(data[c(1:5), c(1:10)])
```

Time.point	Period	Site	Species	Status	Sample.ID	Pair	Depth.ft	biomass	chla
2014 Oct	Bleaching	HIMB	MC	B	3	2	0.6666667	21.22296	2.5716810
2014 Oct	Bleaching	HIMB	MC	NB	4	2	0.6666667	27.31830	4.7656113
2014 Oct	Bleaching	HIMB	MC	B	5	3	1.0000000	19.74599	0.2585994
2014 Oct	Bleaching	HIMB	MC	NB	6	3	1.0000000	15.44902	2.6788595
2014 Oct	Bleaching	HIMB	PC	B	9	5	1.6666667	26.27286	0.0430638

Or make summary data and include this table in your markdown

```
chl.mean<-aggregate(chla~Time.point + Site + Species + Status, data, mean)
biomass.mean<-aggregate(biomass~Time.point + Site + Species + Status, data, mean)
data.table<-cbind(chl.mean[, c(1:5)], biomass.mean[,5])
colnames(data.table)<-c("Time Point", "Site", "Species", "Status", "mean chla", "mean AFDW")
knitr::kable(data.table)
```

Time Point	Site	Species	Status	mean chla	mean AFDW
2014 Oct	HIMB	MC	B	1.4407045	19.38053
2014 Oct	Reef 25	MC	B	1.1771529	16.72889
2014 Oct	Reef 44	MC	B	1.9240275	21.35410
2014 Oct	HIMB	PC	B	0.6808416	30.96592
2014 Oct	Reef 25	PC	B	1.1561949	28.72490
2014 Oct	Reef 44	PC	B	0.8367104	38.47826
2014 Oct	HIMB	MC	NB	5.1866494	27.21420
2014 Oct	Reef 25	MC	NB	3.6726268	26.55296

Time Point	Site	Species	Status	mean chla	mean AFDW
2014 Oct	Reef 44	MC	NB	4.5428889	28.49259
2014 Oct	HIMB	PC	NB	10.3363654	37.18556
2014 Oct	Reef 25	PC	NB	6.3499443	47.68068
2014 Oct	Reef 44	PC	NB	7.3410721	42.94596

Running models

markdown makes running models easy. You can leave all candidate models in the script or you can only report final models. In either case, it is an easy way to keep your data from analysis easy to understand and to QA/QC before publication

chlorophyll a model Load packages and look at structure

```
library("lme4")
library("lmerTest")
library('effects')

str(data)
data$Sample.ID<-as.factor(data$Sample.ID)
```

Run a linear mixed effect model, see the anova output, random effects, and plot effects

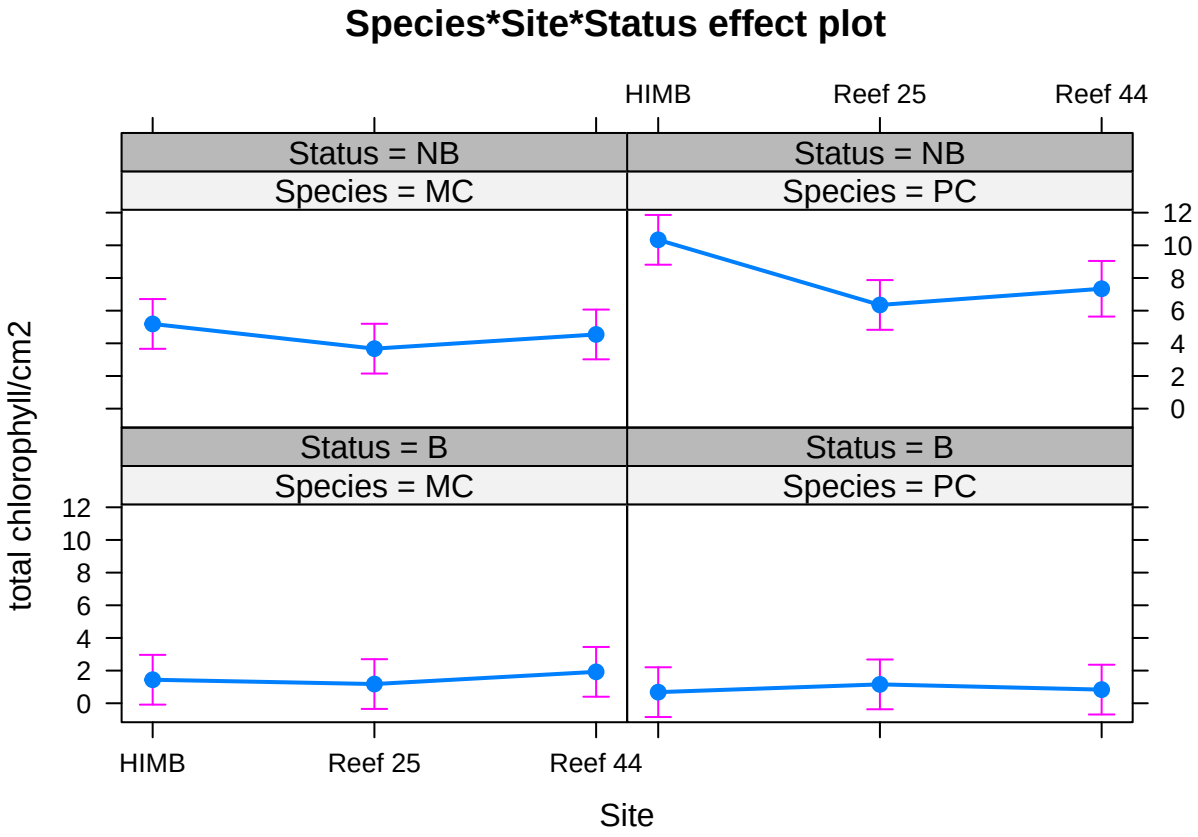
```
mod<-lmer(chla~Species*Site*Status+(1|Pair), data=data)
anova(mod, type=2)
```

```
## Analysis of Variance Table of type II  with  Satterthwaite
## approximation for degrees of freedom
##                      Sum Sq Mean Sq NumDF DenDF F.value    Pr(>F)
## Species               30.28   30.28     1    47  10.566  0.002133 **
## Site                  17.52    8.76     2    47   3.056  0.056521 .
## Status               369.23  369.23     1    47 128.814 4.663e-15 ***
## Species:Site           4.46    2.23     2    47   0.778  0.465048
## Species:Status        63.93   63.93     1    47  22.304 2.134e-05 ***
## Site:Status           21.98   10.99     2    47   3.833  0.028692 *
## Species:Site:Status    6.59    3.29     2    47   1.149  0.325777
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
rand(mod)
```

```
## Analysis of Random effects Table:
##        Chi.sq Chi.DF p.value
## Pair 2.84e-14      1       1
```

```
plot(allEffects(mod), ylab="total chlorophyll/cm2")
```


Species*Site*Status effect plot
HIMB
Reef 25
Reef 44
Status = NB
Species = MC
Status = NB
Species = PC
Status = B
Species = MC
Status = B
Species = PC
total chlorophyll/cm2
0
2
4
6
8
10
12
HIMB
Reef 25
Reef 44
Site