BIO8068 Data visualisation in ecology

Further use of ggplot and model interpretation

## 1. Introduction

Often you will receive complex data-sets, but initial analyses can be confusing, and it may require careful interpretation of model outputs to understand what is the problem. This practical will use a real ecological dataset to illustrate some of the problems. You can learn about fine-tuning and polishing ggplot2 graphics in numerous books and websites, and I particularly recommend The R Cookbook <http://www.cookbook-r.com/Graphs/> which contains the same text as the associated book. The book R for Data Science by Hadley Wickham (author of ggplot2 etc.) and its website <https://r4ds.had.co.nz/index.html> are also excellent sources. The aims of this practical are to:

* show you how to explore ecological data, with common mistakes
* use diagnostic plots to gain better insights

## 2. The data

The data are from a 3-year study into American oystercatchers, *Haematopus palliatus*, inhabitating coastal areas near Buenos Aires, Argentina. Oystercatchers establish nesting territories along the shoreline, of about 50 to 500 metres in size, and when chicks are being reared these are defended by adults, with the parents chasing away other oystercatchers. We will look at a sub-set of the data, for two months, December and January, when the birds are breeding (southern hemispher summer).

Oystercatchers use two techniques to break open clam shells, either a hammering technique or stabbing method. One question in the study was whether the shells eaten by hammerers are larger than those eaten by stabbers. Time of year, and location may also affect what is happening, so we could be looking at a complex 3-way interaction between feeding type (stabber/hammerer), feeding plot and month.

## 3. Import the data and initial inspection

Download the file “OystercatcherData.txt” from Blackboard, create a new project for your oystercatcher data, and within the project create a subfolder called “data” in which to store the downloaded data file. Create an R script, and import the oystercatcher data into a tibble OC. As this is tab-separated text format (readable in Excel on Windows), we’ll use read\_tsv rather than read\_csv:

library(readr)  
OC <- read\_tsv("data/OystercatcherData.txt")

## Parsed with column specification:  
## cols(  
## ShellLength = col\_double(),  
## Month = col\_character(),  
## FeedingType = col\_character(),  
## FeedingPlot = col\_character()  
## )

summary(OC)

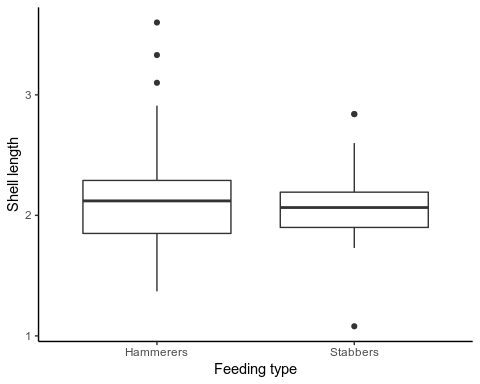
## ShellLength Month FeedingType FeedingPlot   
## Min. :1.080 Length:197 Length:197 Length:197   
## 1st Qu.:1.850 Class :character Class :character Class :character   
## Median :2.110 Mode :character Mode :character Mode :character   
## Mean :2.093   
## 3rd Qu.:2.290   
## Max. :3.600

# Set the Month, FeedingType and FeedingPlot as factors  
OC$Month <- as.factor(OC$Month)  
OC$FeedingType <- as.factor(OC$FeedingType)  
OC$FeedingPlot <- as.factor(OC$FeedingPlot)  
summary(OC)

## ShellLength Month FeedingType FeedingPlot  
## Min. :1.080 Dec: 79 Hammerers:165 A:66   
## 1st Qu.:1.850 Jan:118 Stabbers : 32 B:53   
## Median :2.110 C:78   
## Mean :2.093   
## 3rd Qu.:2.290   
## Max. :3.600

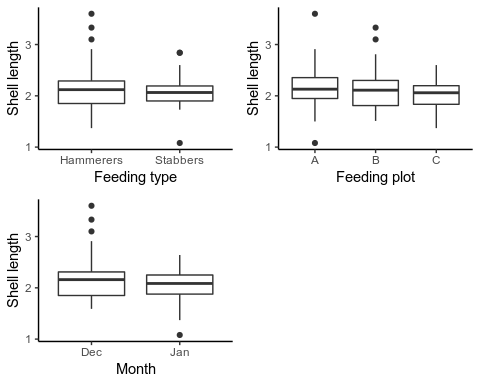
### Initial boxplots

A good initial starting point is to produce some boxplots to explore the data. Using ggplot2, see if you can produce some boxplots similar to this, for each of the three factors. Store the plots in three ggplot objects p1, p2 and p3.



It would be useful to display all three ggplot graphs in one plot, rather than three separate ones. Go to the R Graphics Cookbook website <http://www.cookbook-r.com/Graphs/Multiple_graphs_on_one_page_(ggplot2)/> or search on Google for “R Graphics Cookbook multiplot”. You will see the page has a simple(?!) R function called “multiplot”, so copy this into your R script and run it to make it available. Note that multiplot uses the “grid” package so check that it is installed. Then all you need is:

multiplot(p1, p2, p3, cols=2)



Finally, using the table function provides an easy way of checking the number of oberservations per month, per feeding plot, and per feeding type. This is useful to check that there seem to be a reasonable number of observations to proceed with the analysis.

table(OC$Month)

##   
## Dec Jan   
## 79 118

table(OC$FeedingPlot)

##   
## A B C   
## 66 53 78

table(OC$FeedingType)

##   
## Hammerers Stabbers   
## 165 32

*Note*: as we will see later, we have actually made a major error here. There is in reality a problem with the data, and so really we should not have stopped exploring it now before going ahead with our analyses. We will come back to this issue…

## 4. Applying a linear regression model

We’re going to do a simple linear model with the lm command, looking at all interactions. With so many different predictors in the model it is sometimes useful to use the drop1 command to check that the higher-level 3-way interaction terms are significant:

M1 <- lm(ShellLength ~ FeedingType \* FeedingPlot \* Month,  
 data = OC)  
print(summary(M1), digits = 2)

##   
## Call:  
## lm(formula = ShellLength ~ FeedingType \* FeedingPlot \* Month,   
## data = OC)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.698 -0.192 0.009 0.178 1.392   
##   
## Coefficients:  
## Estimate Std. Error t value  
## (Intercept) 2.208 0.076 29.0  
## FeedingTypeStabbers 0.632 0.234 2.7  
## FeedingPlotB 0.124 0.113 1.1  
## FeedingPlotC -0.197 0.098 -2.0  
## MonthJan -0.076 0.090 -0.8  
## FeedingTypeStabbers:FeedingPlotB -0.936 0.286 -3.3  
## FeedingTypeStabbers:FeedingPlotC -0.573 0.255 -2.2  
## FeedingTypeStabbers:MonthJan -0.987 0.286 -3.5  
## FeedingPlotB:MonthJan -0.234 0.135 -1.7  
## FeedingPlotC:MonthJan 0.105 0.121 0.9  
## FeedingTypeStabbers:FeedingPlotB:MonthJan 1.308 0.380 3.4  
## FeedingTypeStabbers:FeedingPlotC:MonthJan 0.901 0.357 2.5  
## Pr(>|t|)   
## (Intercept) <2e-16 \*\*\*  
## FeedingTypeStabbers 0.008 \*\*   
## FeedingPlotB 0.275   
## FeedingPlotC 0.045 \*   
## MonthJan 0.401   
## FeedingTypeStabbers:FeedingPlotB 0.001 \*\*   
## FeedingTypeStabbers:FeedingPlotC 0.026 \*   
## FeedingTypeStabbers:MonthJan 7e-04 \*\*\*  
## FeedingPlotB:MonthJan 0.085 .   
## FeedingPlotC:MonthJan 0.389   
## FeedingTypeStabbers:FeedingPlotB:MonthJan 7e-04 \*\*\*  
## FeedingTypeStabbers:FeedingPlotC:MonthJan 0.013 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.31 on 185 degrees of freedom  
## Multiple R-squared: 0.15, Adjusted R-squared: 0.094   
## F-statistic: 2.9 on 11 and 185 DF, p-value: 0.0018

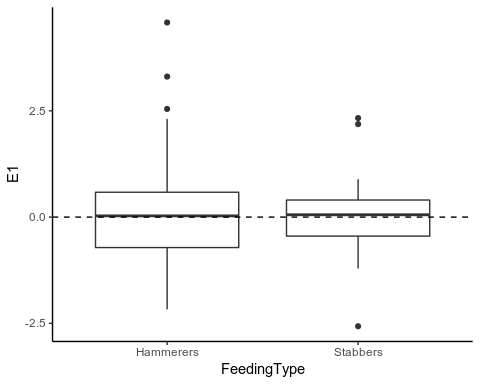
drop1(M1, test = "F")

## Single term deletions  
##   
## Model:  
## ShellLength ~ FeedingType \* FeedingPlot \* Month  
## Df Sum of Sq RSS AIC F value Pr(>F)  
## <none> 18.170 -445.54   
## FeedingType:FeedingPlot:Month 2 1.1962 19.366 -436.98 6.0896 0.002746  
##   
## <none>   
## FeedingType:FeedingPlot:Month \*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

So, we can conclude that the 3-way term of Feeding Type x Feeding Plot x Month is significant. Of course, being good ecological data scientists you know by now that it is not sufficient just to look at the tabular output of a linear model, but also want the graphical output. Issue the command plot(M1) to check the overall model diagnostics; what is your interpretation of the four standard plots, especially the first (Residuals vs Fitted) and second (Normal Q-Q plot)??

Here you have three categorical predictors, so sometimes it is useful to look at the residuals for each predictor separately. Here is the code for the Feeding Type:

E1 <- rstandard(M1) # Extract standardised residuals  
p1r <- ggplot(data=OC, aes(x=FeedingType, y=E1)) +  
 geom\_boxplot() +  
 geom\_hline(aes(yintercept=0), linetype="dashed") +  
 theme\_classic()  
p1r



Modify the code for Feeding plot and month - do the residuals seem roughly similar for each level of each factor, and centred around zero? Use multiplot to but all three plots side-by-side.

## 5. Model interpretation

Given that there seem to be no major problems with our linear model, we can now go ahead and interpret what is going on in more detail. Remember that the value labelled (Intercept) is the overall mean value for the “base” level of a factor, which by default is alphabetical. So here with three factors, the base level is for “hammerers”, sample plot “A”, in month “December”

To calculate hammerers in Plot B in December:

and for stabbers in Plot B in January:

*Question* : How many combinations are there in your data given the numbers of levels in each of your factors? Obviously, having to write out all those equations manually will be slow and tedious. Fortunately, we can use the predict function, in combination with the expand.grid function. The latter creates every combination of all the factor levels:

MyData <- expand.grid(  
 FeedingType = levels(OC$FeedingType),  
 FeedingPlot = levels(OC$FeedingPlot),  
 Month = levels(OC$Month))  
MyData

## FeedingType FeedingPlot Month  
## 1 Hammerers A Dec  
## 2 Stabbers A Dec  
## 3 Hammerers B Dec  
## 4 Stabbers B Dec  
## 5 Hammerers C Dec  
## 6 Stabbers C Dec  
## 7 Hammerers A Jan  
## 8 Stabbers A Jan  
## 9 Hammerers B Jan  
## 10 Stabbers B Jan  
## 11 Hammerers C Jan  
## 12 Stabbers C Jan

You can see that MyData simply contains all the level combinations of every factor. Now push this through the predict function; for completeness we will also calculate the standard errors and upper and lower 95% confidence intervals, and add them to MyData:

#Do the actual prediction  
P1 <- predict(M1, newdata = MyData, se = TRUE)   
  
#Add the predicted values  
MyData$Fit <- P1$fit  
MyData$SE <- P1$se.fit  
MyData$se.low <- P1$fit - 1.96 \* P1$se.fit  
MyData$se.up <- P1$fit + 1.96 \* P1$se.fit  
print(MyData, digits = 3)

## FeedingType FeedingPlot Month Fit SE se.low se.up  
## 1 Hammerers A Dec 2.21 0.0760 2.06 2.36  
## 2 Stabbers A Dec 2.84 0.2216 2.41 3.27  
## 3 Hammerers B Dec 2.33 0.0838 2.17 2.50  
## 4 Stabbers B Dec 2.03 0.1402 1.75 2.30  
## 5 Hammerers C Dec 2.01 0.0615 1.89 2.13  
## 6 Stabbers C Dec 2.07 0.0809 1.91 2.23  
## 7 Hammerers A Jan 2.13 0.0478 2.04 2.23  
## 8 Stabbers A Jan 1.78 0.1567 1.47 2.08  
## 9 Hammerers B Jan 2.02 0.0563 1.91 2.13  
## 10 Stabbers B Jan 2.04 0.1809 1.69 2.39  
## 11 Hammerers C Jan 2.04 0.0537 1.93 2.15  
## 12 Stabbers C Jan 2.01 0.1809 1.66 2.37

It would be useful to be able to visualise these data in ggplot to compare them more readily. We’ll create a new variable combining FeedingType, FeedingPlot and Month into a single text object, then pass the result into ggplot :

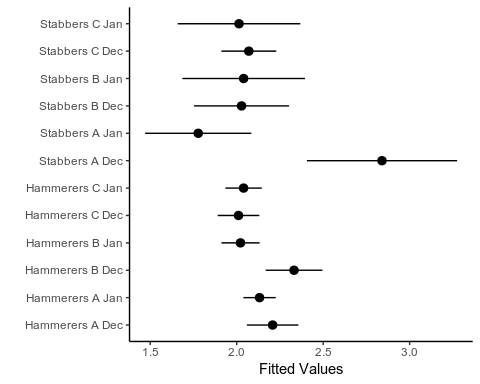
library(magrittr) # pipe operator %>%  
library(dplyr) # mutate function

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

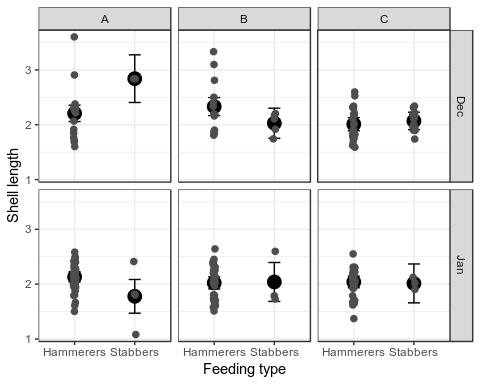
MyData %>%  
 mutate(treatment = paste(FeedingType, FeedingPlot, Month)) %>%   
 ggplot(aes(x = treatment, y=Fit)) +  
 geom\_pointrange(aes(ymin=se.low, ymax=se.up)) +  
 labs(x="", y="Fitted Values") +  
 coord\_flip() +  
 theme\_classic()

 Note the use of coord\_flip to swap the axes for nicer presentation, the setting of the x-axis label (now the y-axis after flipping) to be blank. *Question* : Try reproducing the plot, but with the treatment levels on the y-axis ranked according to the size of the fitted values.

## 5. Trouble ahead

So you have now completed your analysis, know which are the important variables, checked that the model assumptions are correct standard residual plots, and so go ahead and write up an exciting scientific paper. *Not so fast!!* Does anything worry you about the last plot that you have created, and if so what? Something is a little suspect, but we need to investigate futher to find out exactly what is wrong. Let’s replot our fitted values with 95% CI, but in a slightly different format, with observed values superimposed. As there are quite a lot of observations we will “jitter” them to make them easier to see. The next section of code is quite complicated, and therefore check each intermediate ggplot graph as you go along (simply enter p in the Console window). Please ask if you are unsure what any of the options are doing.

# Set out basic structure of plot  
p <- ggplot()  
p <- p + xlab("Feeding type") + ylab("Shell length")  
p <- p + theme(text = element\_text(size=15)) + theme\_bw()  
  
# Add points for the fitted values  
p <- p + geom\_point(data = MyData,   
 aes(x = FeedingType,   
 y = MyData$Fit,   
 size = 6),   
 col = ("black"))  
  
# Add error bars  
p <- p + geom\_errorbar(data = MyData,  
 aes(x = FeedingType,   
 ymax = se.up,   
 ymin = se.low),   
 width=0.2)  
  
# Add observations, with random jitter. See what happens when you alter jitter width  
p <- p + geom\_point(data = OC,   
 aes(x = FeedingType, y = ShellLength),  
 position = position\_jitter(width = .02), #  
 color = grey(0.3),  
 size = 2)  
  
# Change to a grid layout for each combination of factor levels  
p <- p + facet\_grid(Month ~ FeedingPlot, # defines vertical ~ horizontal  
 scales = "fixed") # same scaling in every plot  
  
# Get rid of the legend which isn't doing anything  
p <- p + theme(legend.position="none")   
  
p

 Now, studying the fitted values and observations, as well as what you saw in the previous horizontal plot of fitted values and 95% CI, it is obvious that there is something very odd about there stabbers in December at Site A: where are the observations??! In fact there are observations, but they are so close in value to the fitted value, that you can’t actually see them!

So what on earth is going on? Recall our initial data exploration did not identify any problems, but I said at the time that we should have explored the data further. We produced some tables, but not the following one (try to adapt your original R code to produce this table):

## , , = Hammerers  
##   
##   
## A B C  
## Dec 17 14 26  
## Jan 43 31 34  
##   
## , , = Stabbers  
##   
##   
## A B C  
## Dec 2 5 15  
## Jan 4 3 3

The numbers of stabbers is very low at several site-month combinations, with only 2 in December at Site A. Ironically, when you looked at the plots of overall model performance the “Cook’s distance” plot looked fine, because it could not detect any excess leverage on those observations because there were only a couple in the first place.

# 6. Conclusions: Re-run a better model

Hopefully this exercise has demonstrated how easy it is to be misled by model output, and how good data exploration and visualisation are essential. What should be done in this case? I’m always suspcious of 3-way interactions, as they are usually a nightmare to interpret biologically even when the data are solid. Try re-running your linear model, but omit the 3-way interaction, and see if your results stand up to scrutiny better. You can create similar plots of the fitted values to check that everything is working better without the 3-way interactions.