I'm a bit of an r nerd. Ok, that is a lie, I am a lot of an R nerd. But for good reason, R is incredibly useful for helping make the science process more streamlined and reproducible and with less human error.

The biggest reason I'm often talking to my fellow graduate students about learning R is because of the near constant need to redo things as a grad student, or even within science as a whole. If you spend an entire week using a GUI to run all your models, and then you present them to your advisor, and they realize that you need to change 'just one thing' it will take you another week.

Often, when you have scripted that process, it is often much faster to go back and make small changes without having to redo large parts of your work by hand, or remember every step of the process. This is true whether you are summarizing your data, running models or making graphs. The learning curve is steep, but especially if you plan to continue doing research after your degree, it is totally worth it. It is also helpful when you are working on multiple projects where you might not touch the data for three months and then need to jump right back in. If you have well commented scripts (a file with the code and lots of detail about what is going on) you can more easily jump back into the project, remember where you left out, why you dropped all the data from site 23, and what kind of model you ran. I'm juggling five dissertation chapters and three side projects right now. My to-do list and R scripts are the only thing keeping me above water.

In addition it is helpful when you need to add more data to your analysis, either because more has been collected or because you did it on a subset of the data before. [Noam Ross](http://kbroman.org/Tools4RR/assets/lectures/01\_intro\_withnotes.pdf) has some nice slides on why reproducible research is a valuable goal. These slides discuss some other tools, but the concepts apply very well to R. R can help you make your analysis as transparent as possible and take some of the headache out of the constant rerunning of analysis and regenerating of graphs with 'just one more fix'.

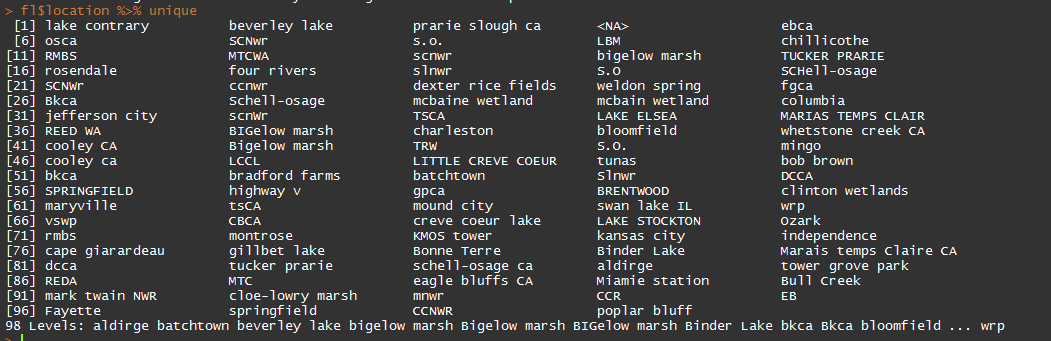
**Data Cleaning**

I don't enter my data in R. Some do, but I find it tedious and also have my technician enter about half of my data it is easier to use a spreadsheet. Once it is entered and proofed though I export it to csv and script everything else in R. I have an awful memory. I will never remember what I did with my data, so I script it so I can look back and figure everything out and also redo it if/WHEN I need to. The raw data file (the original one that was entered from my field data sheets) NEVER gets changed, any altering of that data is done within R and then written to a different csv.

The first part of this is data cleaning. As much as I talk about #otherpeoplesdata on twitter, I often spend a great deal of time cleaning my own data. This is checking for numeric values which are not possible, and often for misspellings. Since I collect my data each fall over a three month period it also lets me explore my data along the way and see what is going on.

Cleaning data may sound tedious, and it can be, but it is very important, otherwise when you try to run analysis or summarize your data and you get summaries of water depth for "missouri", "Missouri", and "MiSsouri" which is less then helpful.

Here is an example of all the unique values of location for a different data set I've been working with. So many inconsistencies. Bigelow marsh and BIGelow marsh for instance.



In addition if you have a data set that will be growing over time, you can use this method to bring together data files from all those different periods in a script so that you don't have to paste things, copy things, worry about destroying things. This is also very helpful if you are receiving data from different people, so you can make sure its all in the same format.

This method also retains the original files as they are incase you need to go back later and figure out where strange values came from. During my field work each morning I export the survey data from our GPS management software and clean them/check them/stitch them together using the 'combining\_night\_files.R'

[you can find all the code and example data files [here](https://github.com/aurielfournier/R\_in\_ecology)]

file\_names <- list.files(path="./R\_in\_ecology/night\_files/",pattern=".csv")

# creates a list of the files so that we can read them all in, and not have to copy and paste all the file names, or worry about updating that list by hand as these files grow in number

# these data are highly subsetted files from my dissertation, I realize that latitude typically has longitude with it, but, this is all I'm using for this example.

library(tidyr)

library(dplyr) # gives us the gift of pipes "%>%"

library(auriel) # my personal R package, used here for the ordinal\_date\_con() function

nights <- list() # creates the list into which all of our files are going to be stored

for(i in 1:length(file\_names)){

# first I have to pull the data I need out of the file name, since I store a bunch of information there

dat <- as.data.frame(file\_names[i])

colnames(dat) <- "name"

names <- dat %>% separate(name, into=c("year","month","day","obs","round","region","area","impound","treat","night"),sep="\_")

names <- names %>% separate(night, into=c("night","file"), sep=-5)

# now we read in the csv and stitch together all of the data from above with this data frame, along with splitting things apart into multiple columns

int <- read.csv(paste0("./R\_in\_ecology/night\_files/",file\_names[i]))

lesscol <- int[,c("lat","name")]

lesscol$name <- as.character(lesscol$name)

lesscol$name <- ifelse(nchar(lesscol$name)==7,paste0(lesscol$name,"N"),lesscol$name)

lesscol <- lesscol %>% separate(name, into=c("name","distance"),sep=5) %>% separate(distance, into=c("species","distance"), sep=1) %>% separate(distance, into=c

("distance","flush\_walk"), sep=-2)

lesscol$distance <- as.numeric(lesscol$distance)

lesscol$species <- tolower(lesscol$species)

lesscol$year <- as.numeric(names$year)

lesscol$month <- as.numeric(names$month)

lesscol$day <- as.numeric(names$day)

lesscol$obs <- names$obs

lesscol$round <- names$round

lesscol$region <- names$region

lesscol$area <- names$area

lesscol$impound <- names$impound

lesscol$treat <- names$treat

lesscol$night <- names$night

lesscol$odat <- ordinal\_date\_con(lesscol[,c("month","day","year")])

nights[[i]] <- lesscol # throws it back into the list in the same place it came from, some will hate this

}

masterdat <- do.call(rbind, nights) # this binds together all of the items in the nights[[]] list. This script is great because it can grow with each file.

# now we can write this out, or do whatever needs doing to it

I also have a script for our survey data, and our vegetation data. Same process, loop through and check each file individually, check it over, and then stitch together. The example data files on github have one file for the same of simplicity.

I use amazing things like match (%in%) to make sure my plants are spelled correctly and my sites don't have typos so that later when I go to analyze my data I don't have to spend a lot of time messing with it.

int <- read.csv("./R\_in\_ecology/example\_survey\_data.csv")

regions <- c("nw","nc","ne","se")

fs <- c("fed","stat")

print(paste0(int[(int$region %in% regions==FALSE),]$region," "," region"))

print(paste0(int[(int$fs %in% fs==FALSE),]$area," "," area"))

is.numeric(dat$sora)

is.numeric(dat$jdate)

When this file just returns a bunch of “ region” and “ area” instead of “SDF region” or whatever else incorrect value is there we are good to go. When doing this over the 100+ files I will collect throughout the season I will run it with a for loop to check each one individually (like I did in combining\_night\_files.R).

The dplyr and tidyr packages are essential to making all of this happen.

**Reports**

One of the big ways that I use R almost every day is through RMarkdown files, which let me take some of the pain-in-the-butt-ness out of report writing and regenerating a series of graphs. I can write in R and knit it all together and come out with a nice looking and clear html or pdf file.

This is SUPER GREAT for creating regular reports for collaborators, or for generating a bunch of graphs every day without having to do a bunch of work (work smart, not hard!)

It takes all of the copy paste out of things, and can make BEAUTIFUL REPORTS with ease ([example here](https://github.com/aurielfournier/R\_in\_ecology/blob/master/example\_report.pdf)). I love RMarkdown, it is the best. Now if only I could get all my collaborators to write in it as well. Instead I send them pdfs, and they give me comments back, which works. This is a [good place to start](http://rmarkdown.rstudio.com/) with RMarkdown.

What is great is you can write things like this

# Fun with Graphs

You can create graphs in RMarkdown and print them with or without the code being shown. the echo, message and warning arguments are sent to false in the

first graph to hide the code and any other outputs.

```{r, echo=F, message=F, warning=F}

dat <- read.csv('./example\_survey\_data.csv')

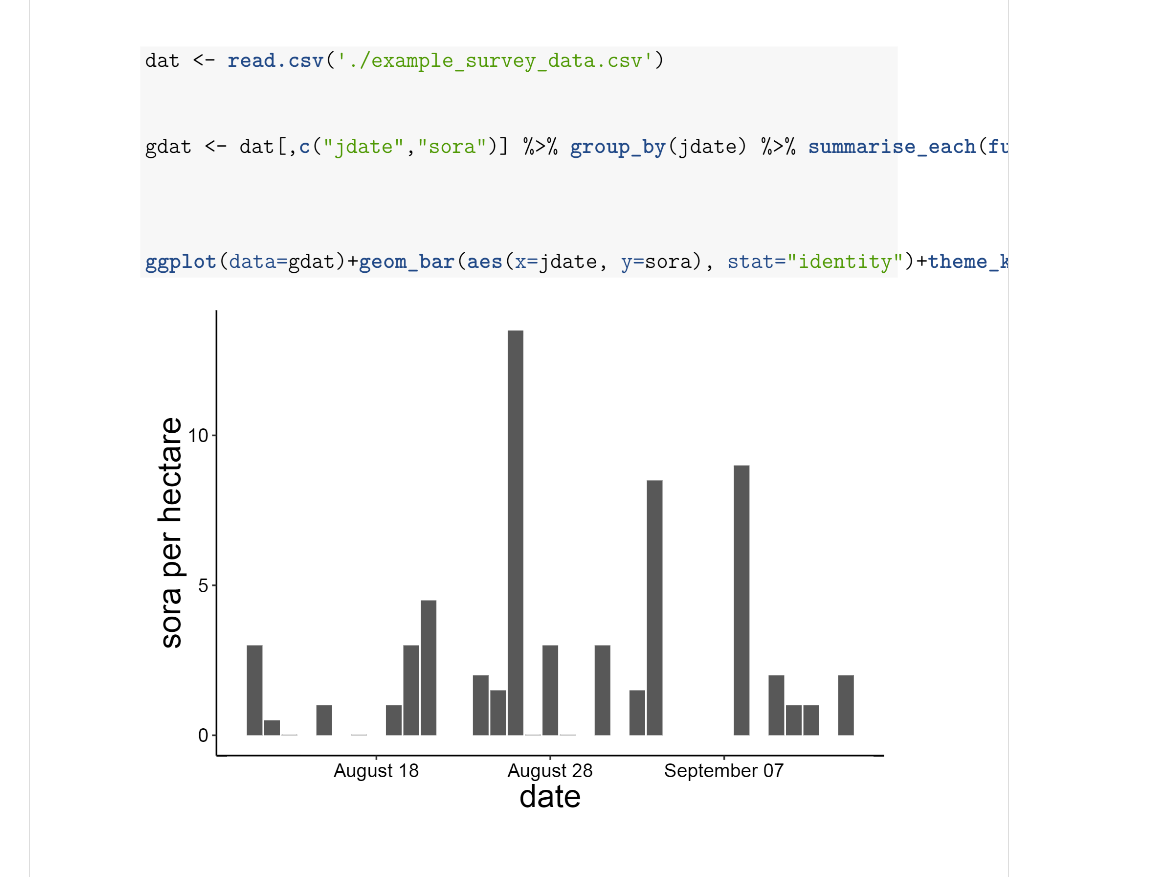
gdat <- dat[,c("jdate","sora")] %>%

group\_by(jdate) %>% summarise\_each(funs(median))

ggplot(data=gdat)+geom\_bar(aes(x=jdate, y=sora), stat="identity")+theme\_krementz()+scale\_x\_ordinaldate(year=2015)+xlab("date")+ylab("sora per hectare")

```

and knit it and get this [see the entire document here]( <https://github.com/aurielfournier/R_in_ecology/blob/master/example_report.pdf>) and the file which generated it [here](https://github.com/aurielfournier/R\_in\_ecology/blob/master/example\_report.Rmd)



Pretty cool, eh? Not exporting graphs and having to copy and paste them into word and deal with formatting and such is a huge lifesaver and can make it much easier to create regularly reports with the same graphs/tables or other data analysis. I use this for my regular field work reports, for my annual report which contains over 50 maps and 25 figures, and for many other smaller products that I use daily/weekly/monthly.   
  
You can also include latex code in the markdown document to help customize it more. I do this for my [CV](https://github.com/aurielfournier/CV)

**Graphing**

I adore ggplot2. I get far too much joy out of making lots of beautiful graphs. Tweaking graphs is what I do when I’m frustrated with the rest of my dissertation.

ggplot is a great way to make publication level graphs that can be used for presentations, to make fun graphics and generated through those RMarkdown reports I mentioned earlier to keep an eye on my data and create updates for collaborators.

One of the things I love about ggplot is that I have built a custom theme named for my adviser that makes graphs that he likes, which saves me A TON of time, which is a valuable thing, and helps keep my adviser happy, which is a valuable thing. So instead of having to write out/copy and paste this every time I want to make a graph that looks the way I want

ggplot(data=masterdat)+

geom\_histogram(aes(x=odat))+

scale\_x\_ordinaldate(2015)+

theme(axis.text.x = element\_text(size=12,color="black"),

axis.text.y = element\_text(size=12,color="black"),

axis.title.y=element\_text(size=20),

plot.background = element\_blank(),

panel.border=element\_blank(),

panel.grid.major= element\_line(colour=NA),

panel.grid.minor=element\_line(colour=NA),

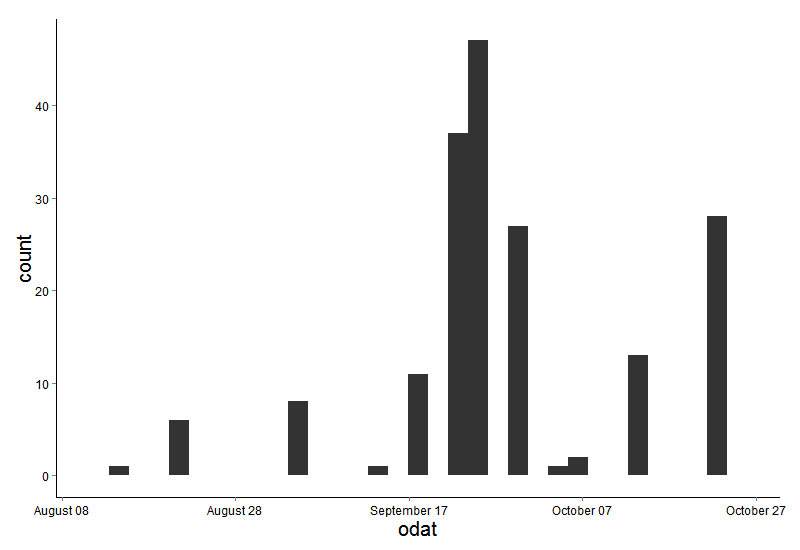
title=element\_text(size=20),

panel.background = element\_rect(fill = "white"),

axis.line=element\_line(colour="black"),

strip.background=element\_rect(fill="white", color="black"),

strip.text=element\_text(size=15))



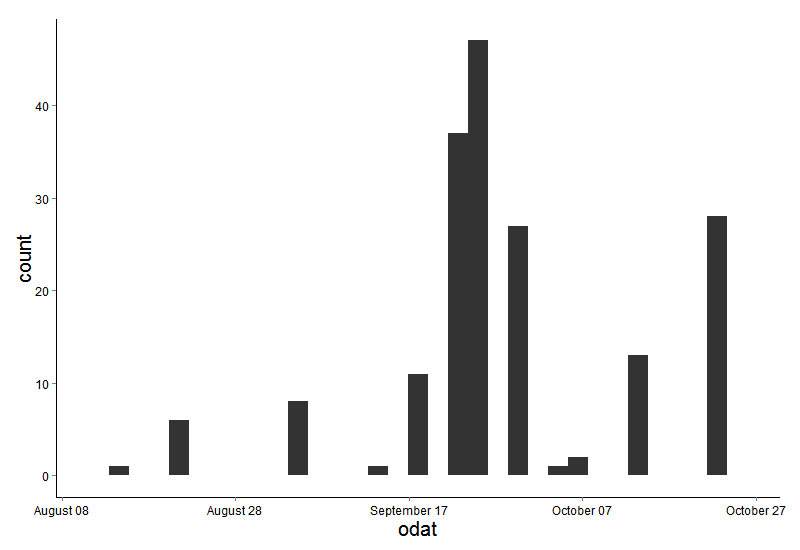
I can write this

ggplot(data=masterdat)+

geom\_histogram(aes(x=odat))+

scale\_x\_ordinaldate(2015)+

theme\_krementz()



Which is shorter, cleaner and doesn't require me to remember lots of things, which is honestly my goal in every R script, to save my poor memory.

**Custom R Packages**

Having your own R package is really useful, its a great place to save things like custom themes for ggplot, and also useful mods for other parts of ggplot, like this one.

I keep all of my dates in ordinal date (day of the year) and so I built this custom scaler for ggplot that makes creating graphs that have real looking dates (August 1), instead of ordinal dates a snap.

scale\_x\_ordinaldate <- function(year){scale\_x\_continuous(label=function(x) strftime(chron(x, origin=c(month=1, day=1,year=year)), "%B %d"))

}

So I can just add scale\_x\_ordinaldate(2015) onto my ggplot graph and it will change my ordinal dates (day of the year) onto my graph as more readable dates (August 1).

Hilary Parker has a great intro written up on creating your own R package (http://hilaryparker.com/2014/04/29/writing-an-r-package-from-scratch/). This is great for saving and using functions you have written, and for saving data sets that you need to see frequently. I use the same datasets in many, many projects, and if I just worked on one computer this wouldn't be a big deal, I can use file paths to keep everything organized and saved in one folder somewhere. But instead in an average week I work on two, if not three different computers, so these file paths just cause frustration, if not waste a ton of my time.

There are a few ways you could get around this, but one of them that I had suggested to be by Oliver Keyes and Jeff Hollister was to save those data files into an R package. Then you just load the package on your computer and bam, data files. These data files can be easily updated, and through the use of github you can move the packages between computers simply. You don't have to worry about any issues of data privacy since you can have private packages, though on github if you don't have the education discount you will have to pay for them (bitbucket is one way of getting around this).

Saving data in an R package also helps encourage you (though does not force you) to make documentation for your data, which is often one of the big issues when you had your data over to someone else, lack of documentation (what do all those column headers mean anyway?).

This is also great practice for writing functions, trouble shooting functions, and maintaining a package so someday you can work on packages that will be shared with others. Maybe you'll never do that, but knowing how packages work, and how to trouble shoot them can save you a ton of headaches in the future.

You can check out some functions I often use at my personal R package [here](https://github.com/aurielfournier/rel). I've got a private data package as well, that I sadly cannot share publicly.

**Modeling**

Most of the modeling I do, and most of the headaches in my day are all because of package 'unmarked' which allows me to use distance sampling data to examine questions about the density of birds with habitat and study their migration.

There are so many ways to model in R, and so many different kinds. I'm most familiar with the hierarchical models supported by unmarked. These models allow for

The estimation of abundance of animals, occupancy (or availability) and detection probability at the same time which is very powerful and allows me to use a variety of covariates to explain each level of the model.

**How to get started with R**

One reason I really like using R is how flexible it can be in making the re-arrangement of data straight forward so that when my collaborators come back with yet another new way of thinking about the data I can quickly rearrange things and move forward with running the models. It helps remove a lot of the memory from doing science, which is a good thing and allows you to go back and figure out where things went wrong when they do (because they will). R does have a learning curve, especially if you've never done programming before, but it can be very fun and rewarding and once you are over the learning curve its super powerful. I'm an instructor for [Software Carpentry](http://software-carpentry.org/) and [Data Carpentry](http://www.datacarpentry.org/) and their lessons, which are available on their websites, are fantastic. I often direct people to start with Software Carpentry to get a handle on the language of R before diving into Data Carpentry but it really depends where you and what your goals are. Packages like [tidyr](http://blog.rstudio.org/2014/07/22/introducing-tidyr/) and [dplyr](https://cran.rstudio.com/web/packages/dplyr/vignettes/introduction.html) are great for cleaning, organizing and manipulating your data within a set of consistent functions both were written by [Hadley Wickham](http://had.co.nz/) who is an active member of the R community, which I recommend people get involved with via Twitter. The #rstats hashtag on twitter is a great place to go and ask questions, I often have multiple helpful responses within 20 minutes. And it is a good place to escape from the snark that can plague other sources of R help.