Making Science Open with R

Monica Granados

Introduction

Open science is the principle that science, including the methods, analysis, results and manuscripts, should be accessible to anyone, anywhere in the world. Unfortunately, most of science happens behind closed doors, having only the opportunity to peer at the results when manuscripts are published often behind a paywall - requiring a fee to unlock. However, the open science movement is working to transform the way scientists do science towards making open the default. All over the world scientists are increasingly working in the open - making the entire process from data collection to publication open.

In today's workshop, we will be going over the four general steps (1. data acquisition 2. analysis 3. results 4. publication) in the scientific process and how we can make each element open and reproducible (which future you will thank you for!).

Data acquisition

From the very early stages of the scientific process, there are measures we can take to make our research more accessible. Today we will be using Google Sheet to integrate with R to make our raw data availible to anyone, anywhere in the world (with computer access). Of course if you work with sensitive data (i.e. medical, threatend species) you may not be able to implement this step.

There are many ways of making your data avaiable. One group of "extreme open scientists" publish thier data online as they produce it. Rachel Harding, a post doctoral fellow at the University of Toronto uploads real-time experimental data in its rawest form.

Many scientists that practice in the open use a Github workflow (github.com). However, GitHub has a bit of a learning curve so we will start with something simple and an interface you might already have familiarity with - a Google spreadsheet.

First, you will have to run this code if you don't have the googlesheets and ggplot2 packages installed already:

- install.packages("googlesheets")
- install.packages("plyr")
- install.packages("tidyr")
- install.packages("ggplot2")

```
#Load packages
library(googlesheets)
library(plyr)
library(tidyr)
library(ggplot2)
```

We are going to use a classic iris data set. This famous (Fisher's or Anderson's) iris data set gives the measurements in centimeters of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris. It is available here: https://docs.google.com/spreadsheets/d/1dQtm0aZNroJdiIuOAI4BAxOnyQ7HBfHrUfyFJOLQjic/edit?usp=sharing

This link is open meaning that its now available to anyone who wants to access it, but not edit is as to preserve your data.

First we have to tell Google to let us talk it with R.

```
#authenticate, this should open up a browser window
gs_auth(new_user = TRUE)
```

Then we can see what Google spreadsheets we have access to.

```
#print list of sheets
gs_ls()
```

If you opened the link to the iris data the the sheet should be visible in your list. The next step is to register the sheet.

```
#register sheet to use in R
irisdata<- gs_title("Iris data")
```

Next we want to read in the data like we would with a file on our local computer

```
#read in data
irisdata<- gs_read(irisdata)
```

Great now we have our data in the R environment! Next let's do some analyses!

Analysis

Merging google sheets with R gives the capacity not only adapt to changes in our data but allow others to access and reproduce your analyses. Say for instance we just wanted to publish some very simple analyses with our iris data.

I want to publish a bar graph of mean sepal and petal length of the different iris species. First lets calculate the mean.

```
#calculate sepal and petal length for each species
irisdata.mean<-ddply(irisdata, .(species), .fun= summarise,
                     sepal_length_mean = mean(sepal_length),
                     petal_length_mean = mean(petal_length))

#arange data frame into long format
irisdata.mean.long<-gather(irisdata.mean, length_mean, cm,
                           sepal_length_mean, petal_length_mean)
```

Results

The R code above allows us to make the analyses we performed available to anyone, they only need download the code you used to calculate the mean. You can make that code available on your website or as a supplement in your manuscript. Later we will discuss how you can make all the material available together on Zenodo.org.

Say we wanted to our results to be plotted as a bar graph. The code below integrates our analyses from above and generates a reproducible graph.

```
#generate plot
irisbar.plot<-ggplot(irisdata.mean.long, aes(x=species,y=cm))+
  geom_bar(stat = "identity")+
  facet_wrap("length_mean")
```

Exercise 1

Now let's pretend you went back to the site this year and gathered additional data for a new species. I'll go into the data sheet and add the new data.

Let's reimport the google sheet so we have the new data.

```
#register sheet to use in R
irisdata<- gs_title("Iris data")
#read in data
irisdata<- gs_read(irisdata)
```

For this exercise I want you to use the code to make a new bar plot with the additional data.

Publication

The publication stage affords you perhaps the most options to work open. To make your publication open you can:

- Publish in an open access journal
- Publish in a hybrid journal and pay the APC
- Publish in traditional journal but submit a preprint to a preprint server (i.e. biorxiv.org); you can check journal policies at this repository: www.sherpa.ac.uk/romeo/index.php

Preprints are awesome because not only are they citable you can also solicit feedback and integrate them into your manuscript before you submit them to a traditional publisher. Organizations like PREreview.org will even organize live journal clubs for your preprint.

But how can we make all our data, code and manuscript all reproducible and open? We'll finish up today by talking about two options.

1. Zenodo

Zenodo is a data and code repository where you can upload your data and code to make it available to openly. It will issue the data and/or code a DOI which is then citable! You can have the your R code seperately and then just write a line of code to import the data locally once it is downloaded from Zenodo. The manuscript iteslef would still need to uploaded as a preprint or a an open publication.

But today I want to show you a cool and totally reproducible option

2. R Markdown

What if we could make our ENTIRE manuscript reproducible? By using R markdown we can. In fact this entire document was made in R markdown.

RMarkdown is an extension of the R Studio GUI that allows you to embed code into your documents. Text in RMarkdown works very much like you general word processor, except instead of a toolbar to bold, underline, italicize and headings - you have to use syntax.

Text

- plain text
- two asterisks on each side for **bold**
- one astrisk for *italics*

Code chunks

To embed code chunks you start first have to distinguish it from plain text with the following: “‘{r eval=X, echo=X, r include=X}. In this statement you indicate whether you would like RMarkdown to evaluate and/or print your code.

- run and print the code: r eval=TRUE, echo=TRUE
- run but hide the code: r include=FALSE
- print but not run the code: r eval=FALSE,echo=TRUE

Finally you end your code chunk with: “‘

Manuscript

By combining text and R code chunks you can write your whole manuscript, code, graphs and all in R. If we return to the example of the iris data set, we can make some small changes to our google sheet to make it interact with R Markdown and make a mini manuscript.

First we have to go back to our sheet in the browser and go to File>Publish to the web and click publish. Then we have to write code to have R Markdown automatically download the data.

```
irisdata <- gs_key("1dQtm0aZNroJdiIuOAI4BAxOnyQ7HBfHrUfyFJOLQjic")
irisdata <- gs_read(irisdata)
```

Now we can write our mini manuscript.

Introduction

Irises are cool. And come in all shapes and sizes.

Methods

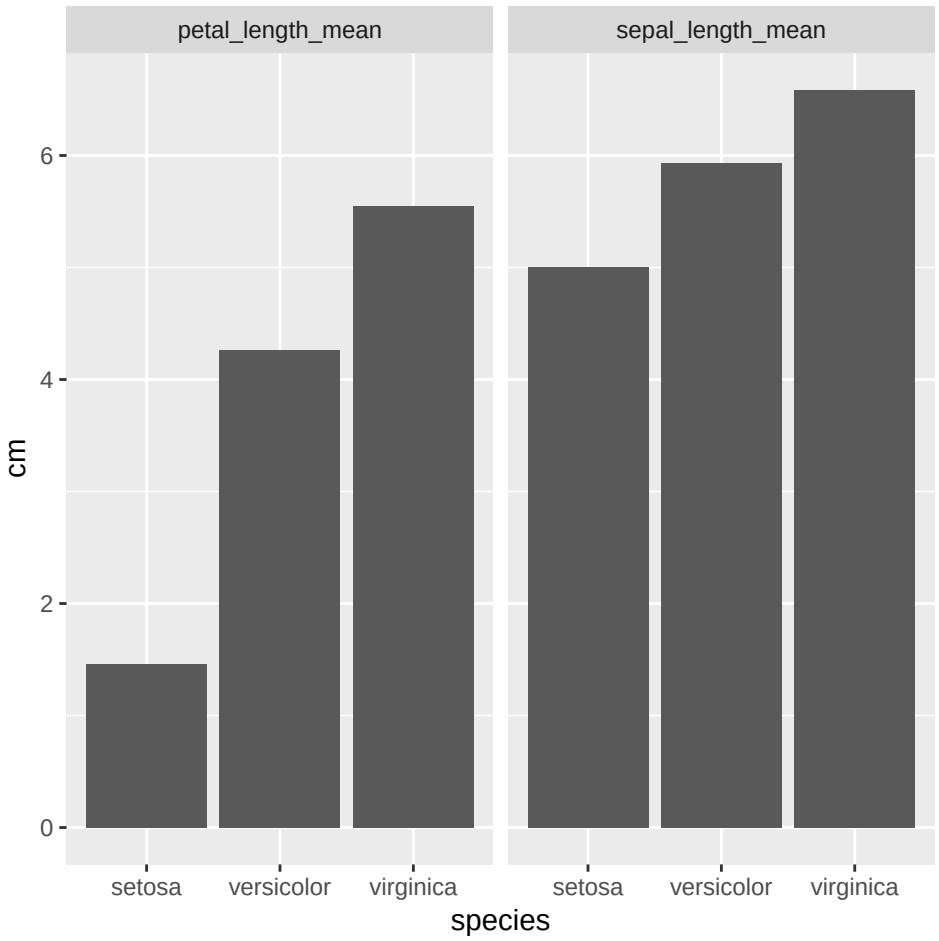
We calculated the mean sepal and petal lengths.

```
#calculate sepal and petal length for each species
irisdata.mean<-ddply(irisdata, .(species), .fun= summarise,
                     sepal_length_mean = mean(sepal_length),
                     petal_length_mean = mean(petal_length))

#arange data frame into long format
irisdata.mean.long<-gather(irisdata.mean, length_mean, cm,
                           sepal_length_mean, petal_length_mean)
```

Results

We found the species setosa was the smallest.

```
#generate plot
irisbar.plot<-ggplot(irisdata.mean.long, aes(x=species,y=cm))+
  geom_bar(stat = "identity")+
  facet_wrap("length_mean")

irisbar.plot
```



Discussion

Irises are cool.

Exercise 2

Make your own mini manuscript but for the sepal and petal width. You'll have to download the R Markdown package: install.packages("markdown")

Questions?

Additional resources

- Google sheets: https://cran.r-project.org/web/packages/googlesheets/vignettes/basic-usage.html
- R Markdown: https://rmarkdown.rstudio.com/