**galah talk**

Biodiversity data is growing rapidly in large biodiversity databases! The Global Biodiversity Information Facility (or GBIF) currently holds over **2.1 billion records** of species observations globally. As you might imagine, this amount of data can be very useful for understanding ecosystems, and to monitor terrestrial and marine life. These data can also enable researchers to test big-data hypotheses about species distributions and interactions.

A lot of this data growth has happened over the last 10 or 15 years. In GBIF’s case, record numbers have grown by over 1,600% since 2007 when they held 125 million records. To put that in context, if a human grew in height by that amount, they would be taller than 2 of the Hollywood signs stacked on top of each other!

GBIF aggregates data globally from a series of partner node organisations, usually for individual countries, who are responsible for aggregating data from providers in their nation or region. For example, there are biodiversity databases for Tanzania, Spain, Sweden and Australia, just to name a few.

And as of a few months ago, there’s even a new website where you can see all the living atlases!

I work at the Atlas of Living Australia, or the ALA, which, like other databases, consolidates data from museums and collections, government organisations, research groups, monitoring programs, and citizen science apps like iNaturalist and eBird. Some of you might even use some of these apps yourself!

Nodes like the ALA are important for maintaining data sharing arrangements among all of these groups. They can also provide more specific data, or data services - for example, providing specific state or regional threatened species lists. And so, it can be valuable to search for biodiversity through specific nodes. Not only on GBIF.

R is a popular interface for ecologists & biologists who want to statistically analyse species observations data. R is **particularly** useful because it is possible to make the entire process of analysis – from first downloading the data to the final results – transparent and reproducible. This is an essential part of good science practices.

A few packages have made huge strides to make this possible. But with the growing popularity of the tidyverse among R users, especially new R users, these packages – useful as they are – may not the easiest to first jump into using.

Two popular packages for working with and downloading biodiversity data are the taxize package (for searching taxonomic information), and the rgbif package (for downloading data from GBIF). Both of these packages are awesome, but not super easy for a beginner user to pick up.

For example, the **output** of taxize provides a **list** of attributes, which probably looks quite intimidating for a beginner R user who’s used to tidyverse syntax.

You can use taxize with rgbif as one way to filter a data download query, which is composed of these `pred`, or “predicate,” functions that specify one filter each. Although this works quite well, it can get…

**…unwieldly** pretty quickly. Again, a beginner user might find this intimidating or unintuitive.

These packages aren’t bad – we stand on the shoulders of their innovation – but they highlight how working with APIs, or programming interfaces that return results from a query, can be a **nightmare**, and translating them into R syntax isn’t easy.

And so, we built **the galah package** to query data from the ALA, named for that beautiful Australian bird in our hex.

galah improves on previous R packages in two ways:

1. galah queries a national biodiversity database – at the time it was one of the first to do so – and it is flexible enough to work with other national databases (which I’ll come back to at the end)
2. Somewhat accidentally, we realised that filtering a query works a lot like filtering a dataset. The galah package builds queries in a tidy, pipeable way

To start, let’s build a query with the `galah` package to see how it works

Let’s download the number of *Eolophus roseicapilla* records – the scientific name for galahs.

This basic query demonstrates the major components of how the galah package works.

You start a query with galah\_call(), which effectively works like a simple **.data** argument in dplyr.

Functionally, it accepts subsequent `galah\_` functions as filters – here we identify our taxonomic group to be galahs –

and our query ends when an `atlas\_` function is called, which specifies the type of data we want returned - in this case record counts.

When run, a tibble is returned with the total number of records, 991,859

Here are other galah functions to help narrow query results. tidyverse users will notice they have the same names as `filter`, `group\_by` and `select` in dplyr, and that’s because they essentially do the same thing to your query that they would to your dataset.

galah\_filter filters rows.

For our query, this means we can filter to only return the number of galah records since 2010 recorded through iNaturalist. What would normally be columns to filter your data in dplyr are “fields” that you can filter searches by on the website (and they’re effectively the same as columns anyway). The galah package has a few helper functions to search for these fields.

We can group our counts by using galah\_group\_by.

Here we group by year, and the output looks like something we might get by running `group\_by` and then `count` functions with dplyr

And we can group by multiple fields, in this case by year and by the data source

We can also get a list of species. In this case I want to find all the bandicoot and bilby species found in 2021 in New South Wales. If you don’t know what a bandicoot is, there’s a picture for you. They are an adorable mostly nocturnal marsupial, and they are **great**

By calling atlas\_species at the end of the query, we return a list of the 3 species found

Now we might want to download species occurrence records to analyse the data ourselves.

To do this, we have to first supply a registered email with the ALA, which is a requirement of our database, even on the website.

Then we call atlas\_occurrences at the end of our query, and we get a tibble of all 2,156 records, with their location, scientific name, date observed, and the specific record ID on the ALA.

Sometimes you might not want all that information, and getting all of the default columns **can** increase the size of a download.

galah\_select selects columns, and here I’ve selected only the scientific name and the event date to be returned

When combined with other great packages like the ozmaps package and the sf package for plotting spatial features, and ggplot2 for dataviz, going from downloading data to making something nice like a map of observations can be **quick** and **transparent**. For example, if I wanted to map the species occurrences of bandicoots and bilbies,

I can download my data with a few filters

Fix up the map projection to what the ALA uses

And plot the points with the supplied Lat-Lon coordinates

Thanks to ggplot being awesome, this looks pretty nice already!

Something that we realised somewhat by accident is that the APIs for many Living Atlases, including GBIF, are very similar. Information is stored in similar places, uses similar fields and is returned in similar formats. Some were even built using the ALA’s API structure.

What this means is that internally, we just need to point galah to use the url of the API on the Living Atlas we want, and it can begin using that API to return data. The APIs based on the ALA’s API can already do some basic queries without changing anything in galah.

For example, if you set the atlas argument in galah\_config to Spain, you can return the total counts for the Spanish Atlas.

You can do the same thing with the Austrian Atlas, and a few others. Other atlases are **a little** different because they use GBIF architecture, and we are currently experimenting with how much work it would take to make galah use their APIs correctly.

But there is a lot of potential here, and could be very exciting! We have already been contacted by a few atlases about the potential to support or make an R package for their Atlas!

Although making data easier to access in R is a good first step, biodiversity data is only truly useful when it is applied well and transparently.

To help do this,

we made the ALA Labs website. This website was designed as a resource for people who are interested in more technical code-based ways to make data visualisations and run statistical analyses in R (at the moment)

The website’s main focus is to share “how-to” posts about how to use galah to make or do cool things. I encourage you to check it out 😊

We made galah to make downloading biodiversity data from the ALA easier. And what I’ve shown isn’t everything it can do either.

Inspired by the tidyverse:

* galah makes **downloading** data like **wrangling** data with dplyr, which helps people who have used the tidyverse jump in quickly
* The architecture of galah is flexible for APIs of other atlases, and we are currently trying to see how it can help with other biodiversity databases too

With these innovations, and with the help of ALA Labs, we hope that the galah package will make it easier for people to access and analyse biodiversity data reproducibly!

Thanks for listening!

* We tried to make an R package ALA4R origin story
* We deliberately set up other nodes to use our API architecture
* Mention galah pkgdown website before ALA Labs