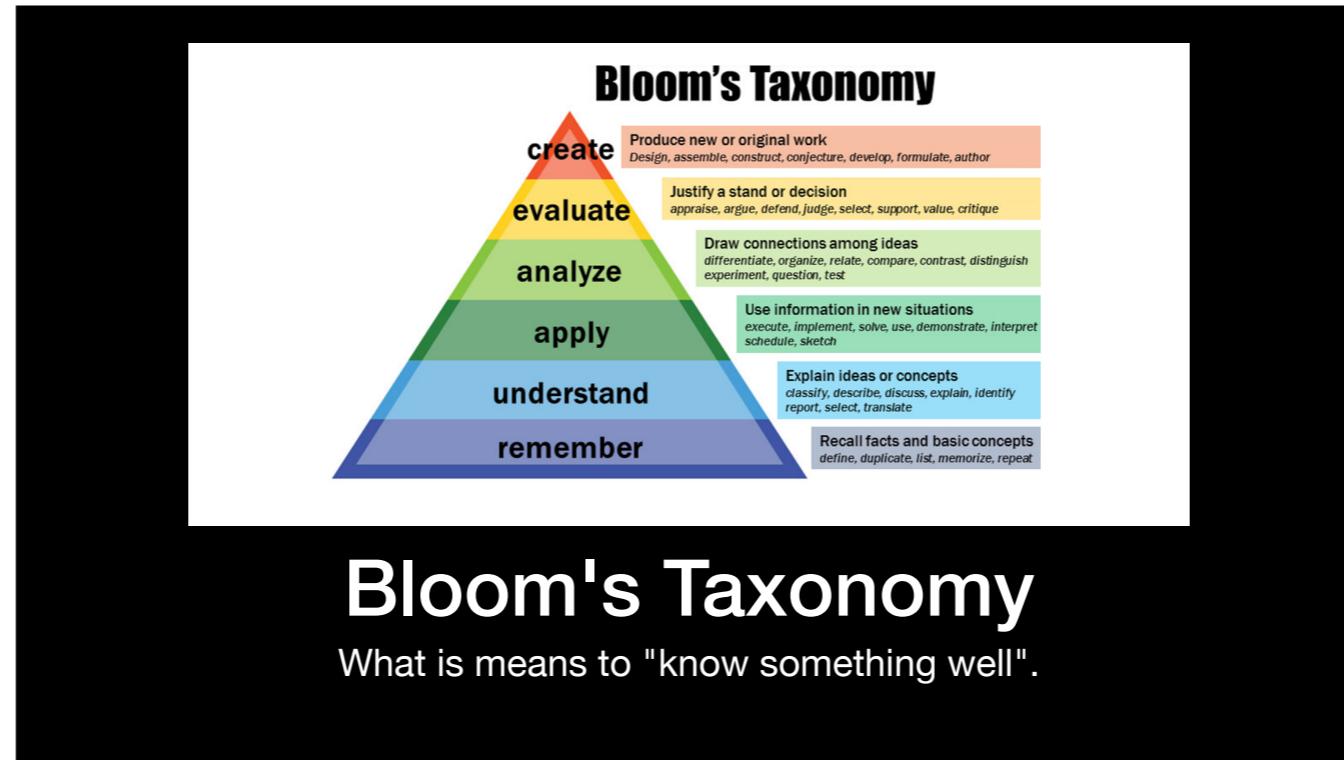


What's next?

Ari Lamstein

We've now finished the workshop, and I'm sure many of you are wondering "What's next?"



This is "Bloom's Taxonomy". It was created by Benjamin Bloom, who was an educational psychologist. It seeks to answer "What does it mean to know a field", and it can be used in R as well. Applying this to our workshop:

The reason I gave you handouts, and asked you to write a summary of what you learned along the way, is to help you remember what you've learned.

The reason why I lectured was to help you understand the code.

And the reason why I gave you exercises was to help you apply what you learned. This is the basic hierarchy of learning.

If you want to continue improving at R, you should work with your supervisor to get more chances to apply what you learned in the workshop. In R, the "apply" phase lasts a very long time, because data is always different, there are a ton of cryptic error messages, and so on.

3 Levels of R Programming

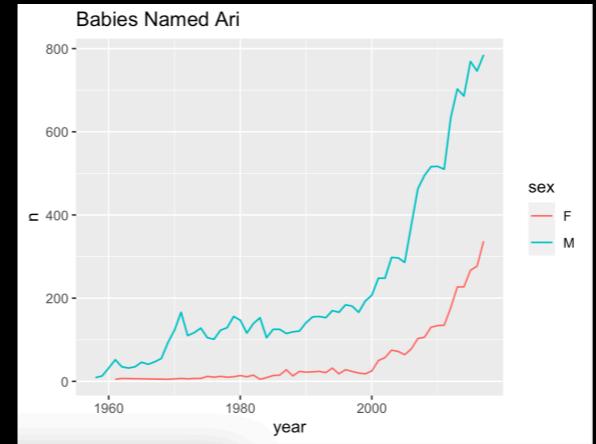
1. A single analysis (scripts)
2. Programs to do multiple analysis (functions)
3. Sharing programs (packages, shiny)

I'd now like to explain why you might want to continue to learn R. What you learned in this workshop was how to do a single analysis in R. This is a great place to start. But the truth is that you can do that with lots of other tools. What makes R special, I think, is that it's a full-fledged programming language. And I'd like to talk for a minute about what that means.

A Single Analysis (Scripts)

```
library(ggplot2)
library(babynames)

babynames %>%
  filter(name == "Ari") %>%
  ggplot(aes(year, n, color=sex)) +
  geom_line() +
  ggtitle("Babies Named Ari")
```



This is an example of a single analysis. You're given a dataset of baby name popularity, and you create a graph that shows the popularity of a single name over time.

3 Levels of R Programming

1. A single analysis (scripts)
2. Programs to do multiple analysis (functions)
3. Sharing programs (packages, shiny)

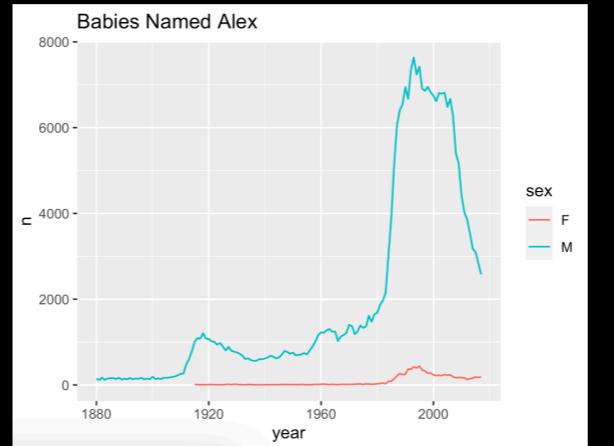
I think that R gets really interesting when you start writing functions in it. Let me explain.

Programs to do multiple analyses

```
library(ggplot2)
library(babynames)
library(glue)

baby_name_graph = function(baby_name) {
  babynames %>%
    filter(name == baby_name) %>%
    ggplot(aes(year, n, color=sex)) +
    geom_line() +
    ggtitle(glue("Babies Named {baby_name}"))
}

baby_name_graph("Alex")
```



Here I encapsulated all the code to create a graph of name popularity in a single function called "baby_name_graph". It takes a parameter called baby_name, and then makes a graph of that name's popularity. As a proof of concept, I am calling it with the parameter "Alex".

What does this do?

```
# Don't try this at home
all_names = unique(babynames$name)
for (one_name in all_names) {
  baby_name_graph(one_name)
}
```

Once you encapsulate an analysis in a function, an entire new world of options opens up. Can you guess what this program does?

Once you become fluent in writing a single analysis, the next step is start packing up useful components of those analyses into functions, and then reusing them in new ways. The analyses that you spend so much time perfecting become something you can reuse instantaneously in another context. This is why people love R.

The screenshot shows the homepage of the "R for Data Science" website at r4ds.had.co.nz. The page has a dark background. On the left, there is a sidebar with a list of chapters. The chapters under the "Program" section (17 to 21) are highlighted with a yellow background. The main content area features a large heading "Welcome" and a paragraph describing the book's purpose. To the right of the text is an image of the book cover for "R for Data Science" by Hadley Wickham & Garrett Grolemund.

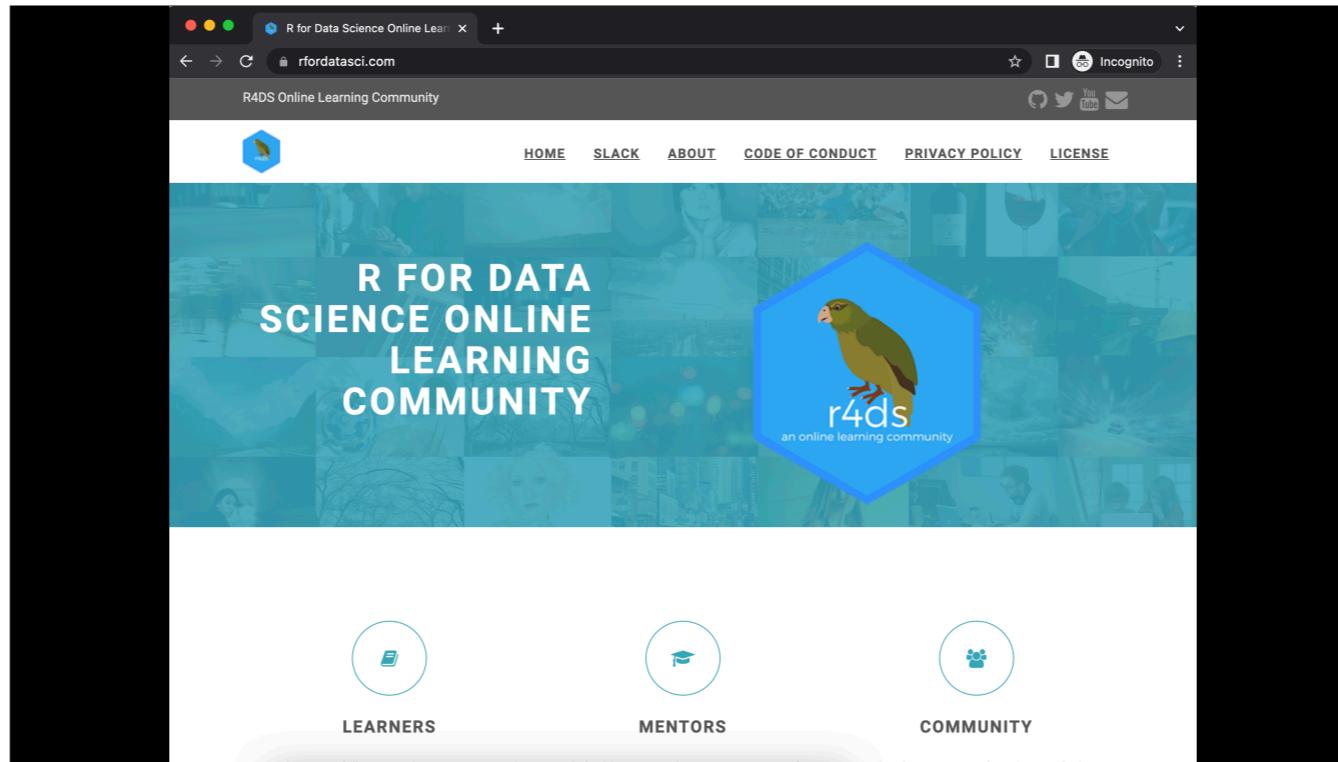
12 Tidy data
13 Relational data
14 Strings
15 Factors
16 Dates and times
Program
17 Introduction
18 Pipes
19 Functions
20 Vectors
21 Iteration
Model
22 Introduction
23 Model basics
24 Model building
25 Many models
Communicate
26 Introduction

Welcome

This is the website for "**R for Data Science**". This book will teach you how to do data science with R: You'll learn how to get your data into R, get it into the most useful structure, transform it, visualise it and model it. In this book, you will find a practicum of skills for data science. Just as a chemist learns how to clean test tubes and stock a lab, you'll learn how to clean data and draw plots—and many other things besides. These are the skills that allow data science to happen, and here you will find the best practices for doing each of these things with R. You'll learn how to use the grammar of graphics, literate programming, and reproducible research to save time. You'll also learn how to manage cognitive resources to facilitate discoveries when wrangling, visualising, and exploring data.

The book cover for "R for Data Science" is displayed. It features a green parrot-like bird on a red background. The title "R for Data Science" is prominently displayed in white. Below the title, it says "VISUALIZE, MODEL, TRANSFORM, TIDY, AND IMPORT DATA". At the bottom, the authors' names are listed: "Hadley Wickham & Garrett Grolemund".

The best introduction to programming in R is probably the "Program" section of "R for Data Science". Specifically the sections on functions and iteration.



I also want to mention the R for Data Science Learning Community. While I haven't joined it myself, I have heard good things about it. If you wind up joining, I'd love to know about your experience.

3 Levels of R Programming

1. A single analysis (scripts)
2. Programs to do multiple analysis (functions)
3. Sharing programs (packages, shiny)

Finally, R lets you share your programs with other people in two really novel ways: packages and Shiny. This is another reason, I think, why people love using R: when you finish doing a single analysis, you can do so much more with it.

Packages

```
library(myPackage)  
baby_name_graph("Andy")
```

An R package lets you encapsulate all the code and dependencies of your analysis.

Here we packaged up our babynname graphing function in the package "myPackage". The end user can use that function without knowing about the babynames package, the tidyverse, and so on.

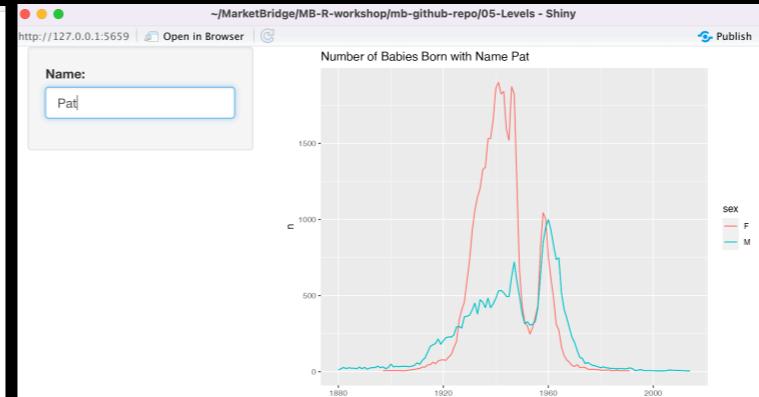
Shiny - Interactive Web Application

```
library(shiny)
library(babynames)
library(tidyverse)
library(scales)

ui <- fluidPage(
  sidebarLayout(
    sidebarPanel(
      textInput("name", "Name:", value="Ari")
    ),
    mainPanel(
      plotOutput("namePlot")
    )
  )
)

server <- function(input, output) {
  output$namePlot <- renderPlot({
    title = paste0("Number of Babies Born with Name ", input$name)
    babynames %>%
      filter(name == input$name) %>%
      ggplot() +
      geom_line(aes(year,n, color=sex)) +
      ggtitle(title)
  })
}

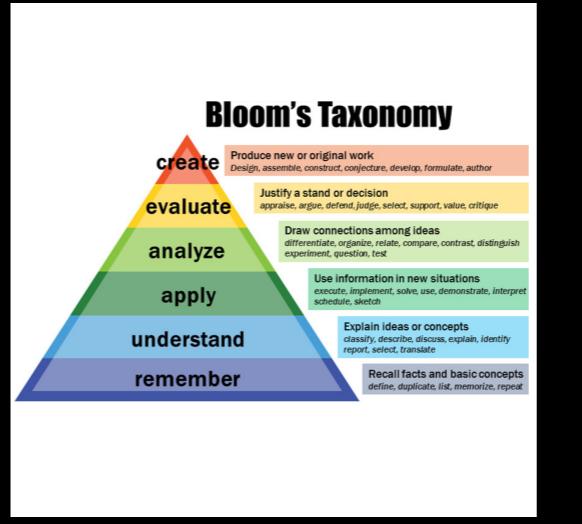
# Run the application
shinyApp(ui = ui, server = server)
```



Packages have been around for a long time. Shiny is much newer. Here's an app that lets you enter any name into a web application and get a graph of that name's popularity over time. Here's the R code to create it.

Closing Thought

1. A single analysis (scripts)
2. Programs to do multiple analysis (functions)
3. Sharing programs (packages, shiny)



If you'd like to continue to improve at R, please just seek out opportunities to apply what you've learned in this workshop to your projects.