Advanced data visualization with R Workshop Day 2

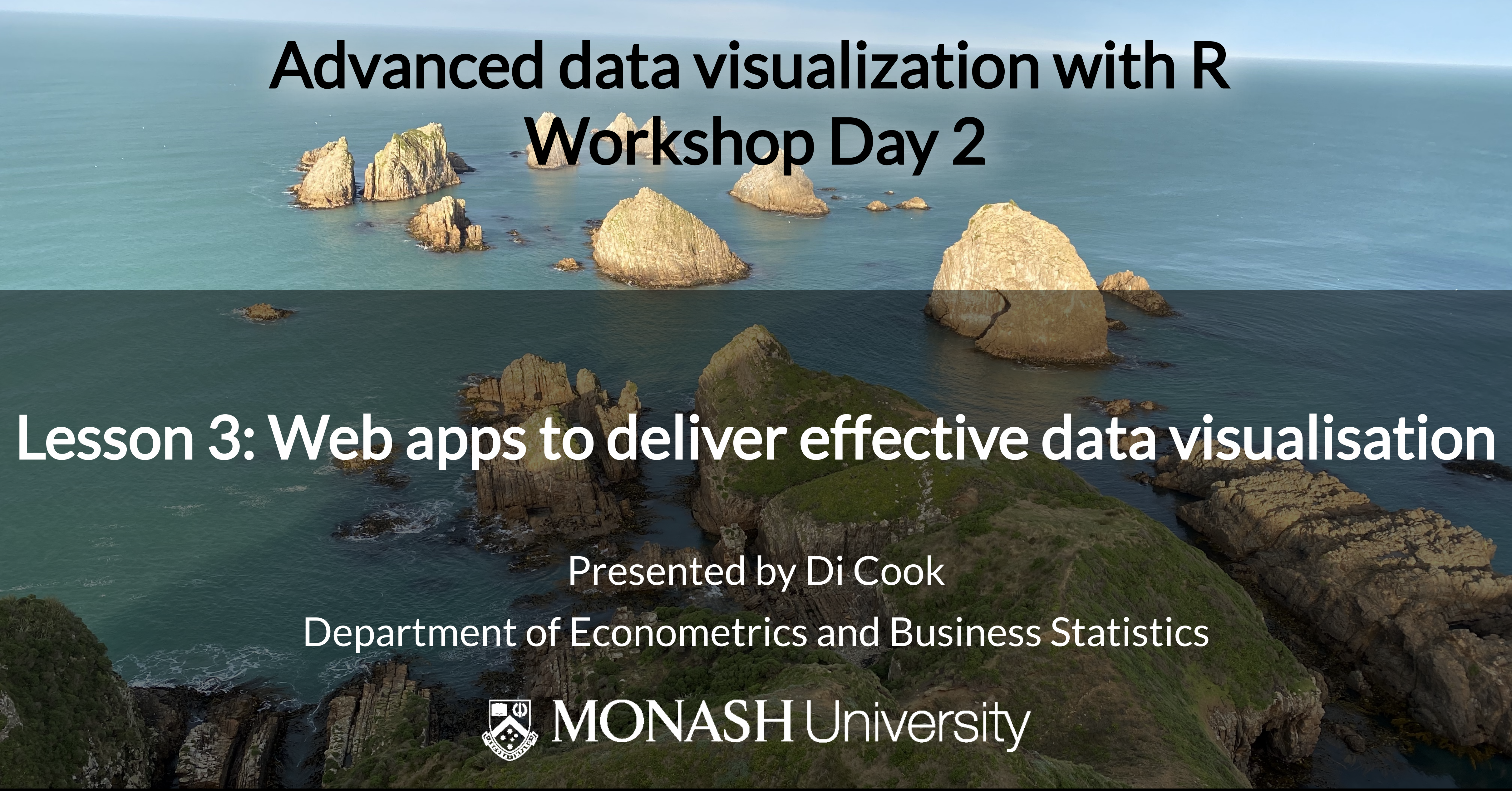
Lesson 3: Web apps to deliver effective data visualisation

Presented by Di Cook

Department of Econometrics and Business Statistics

MONASH University

8-9th Dec 2021 @ Statistical Society of Australia | Zoom



Outline

Mapping out an app

Choices in packages for apps

Building a shiny app

demo data for apps

learningtower: PISA Scores

```r
load(here::here("data/student.rda"))
student %>% glimpse()

## Rows: 2,929,621
## Columns: 22
## $ year        <fct> 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000…
## $ country     <fct> ALB, ALB, ALB, ALB, ALB, ALB, ALB, ALB, ALB, ALB, ALB, ALB…
## $ school_id   <fct> 1001, 1001, 1001, 1001, 1001, 1001, 1001, 1001, 1001, 1001…
## $ student_id  <fct> 1, 3, 6, 8, 11, 12, 17, 20, 21, 22, 24, 27, 28, 29, 30, 32…
## $ mother_educ <fct> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA…
## $ father_educ <fct> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA…
## $ gender      <fct> female, female, male, female, female, female, male, female…
## $ computer    <fct> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA…
## $ internet    <fct> no, no, no, no, no, no, yes, no, no, no, no, no, no, no, n…
## $ math        <dbl> 324.35, NA, NA, 235.79, NA, 290.74, NA, 266.31, NA, NA, 29…
## $ read        <dbl> 397.87, 368.41, 294.17, 241.49, 287.16, 307.84, 181.73, 33…
## $ science     <dbl> 345.66, 385.83, 327.94, 341.09, 307.15, 277.04, 279.19, 46…
## $ stu_wgt     <dbl> 2.1600, 2.1600, 2.1600, 2.1600, 2.1600, 2.1600, 2.1600, 2.…
```

Mapping out an app

Your goal is to make the analysis easy for someone without coding skills.

Decide on what key insights might be made, and structure the app around these, eg

- How do scores vary between countries?
- Is there a difference between genders?
- Are there trends in scores over time?

Keep in mind that an app needs to be responsive. Users need the change to happen very quickly.

Web apps with R

There are many options for making web apps using R, including:

- shiny
- learnr
- flexdashboard
- shinydashboard

shiny is...

- an R package that makes it easy to build interactive web apps straight from R.
- You can also extend your Shiny apps with CSS themes, htmlwidgets, and JavaScript actions.
- It can be used to build dashboards.

learnr is...

- an R package that makes it easy to turn any R Markdown document into an interactive tutorial.
- Tutorials consist of content along with interactive components for checking and reinforcing understanding.
- Tutorials can include any or all of the following:
  - Narrative, figures, illustrations, and equations.
  - Code exercises (R code chunks that users can edit and execute directly).
  - Quiz questions.
  - Videos (supported services include YouTube and Vimeo).
  - Interactive Shiny components.

flexdashboard is...

- Rmarkdown based, allows multi-page tabbed layouts
- Static dashboards, but can use `shiny` with `runtime: shiny` in the YAML.
- Easy to customise, like writing a regular `html_document`
- Easy to deploy

shinydashboard is...

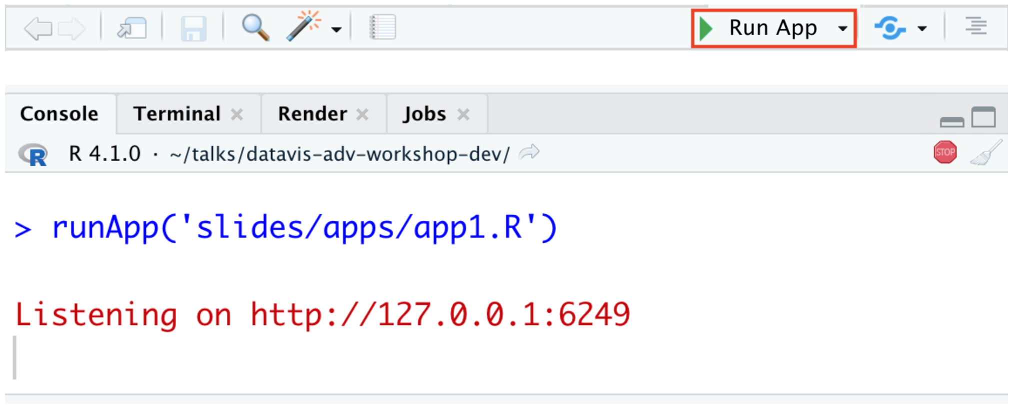
- `shiny` based but introduces dashboard visual motifs
- Interactive or static dashboards but requires you to know how to set up a `shiny` app
- Harder to customise but could be used to build fully fledged web apps

Start with shiny

Single file: app.r

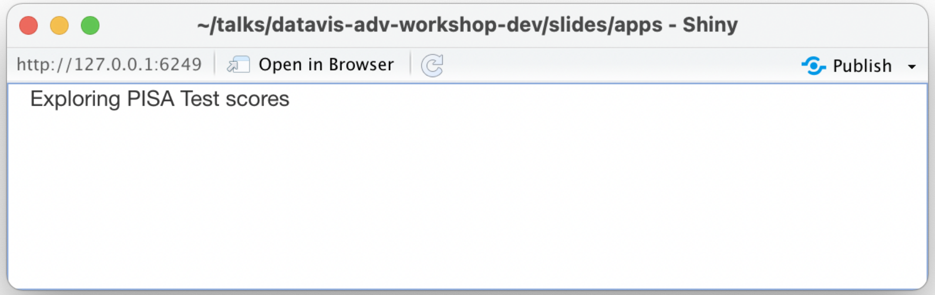
Create a new directory, and a R script file called `app1.r`, with these contents:

```r
library(shiny)
ui <- fluidPage(
  "Exploring PISA Test scores"
)
server <- function(input,
                   output,
                   session) {
}
shinyApp(ui, server)
```



```
> runApp('slides/apps/app1.R')

Listening on http://127.0.0.1:6249
```



Exploring PISA Test scores

This app simply writes "Exploring PISA Test scores" into a new window.

Alternatively, using the RStudio window, you can create a new app from template.

Or, you can start from an app that mostly has the features you want for your own app.

YOUR TURN

In your R Studio window, open `app1.r` file, and click `Run App`.

App structure

There are two main parts of a shiny app:

- What we see and interact with:
    - user interface (ui): layout with user input and (plot) output
- What is going on underneath:
    - the server: glue between user input and output

Shiny inputs

Shiny has many different input options, see the widget gallery:

- `actionButton()` - creates a clickable button
- `selectInput()` create a select list
- `checkboxInput()` and `checkboxGroupInput()`
- `dateInput()` - calendar to select a date
- `dateRangeInput()` - select a range of dates
- `fileInput()` - upload a file
- `numericInput()` - input a numeric value
- `radioButtons()` - select one or more items
- `sliderInput()` - slide along a range of values
- `textInput()` - input a string

Shiny outputs

These are what is drawn, written or shown in your app:

- `renderDataTable()` - outputs an interactive, sortable data table
- `htmlOutput()` - output html elements
- `plotlyOutput()` - output with plotly elements
- `renderPlot()` - output an R plot
- `renderPlotly()` - output plotly interactive plot
- `renderPrint()` - output text from print() in R
- `renderTable()` - output an HTML table
- `renderText()` - output text from R
- `renderUI()` - output a custom part of the user interface
- `renderImage()` - print an image to the page

PISA scores app

Our app will have these elements:

- Purpose: Compare temporal trend in scores across countries
- UI:
  - Menu to choose subject: math, read, science
  - Text entry/menu to select country to highlight
- Output (server): time plots of scores for each country

Pre-process data prior to making app, to have smaller, focused data to help with responsiveness.

```
data(countrycode)
student_app <- student %>%
  group_by(year, country) %>%
  summarise(math = weighted.mean(math,
              w=stu_wgt, na.rm=TRUE),
            read = weighted.mean(read,
              w=stu_wgt, na.rm=TRUE),
            science = weighted.mean(sci
              w=stu_wgt, na.rm=TRUE),
            .groups = "drop") %>%
  left_join(countrycode) %>%
  select(year, country_name, math, scie
  rename(country = country_name) %>%
  mutate(year = as.numeric(as.character
save(student_app, file=here::here("data
```

🔧 YOUR TURN

1. In your RStudio Window open and run the `pisa_app`
2. Change the highlight colour and re-run
3. ADVANCED: Change the output to be an interactive plotly

Trouble-shooting

Tracebacks printed into the Console are your friend. These will pinpoint the location where the code is failing.

```
Error in *: non-numeric argu
  169: g [app.R#4]
  168: f [app.R#3]
  167: renderPlot [app.R#13]
  165: func
  125: drawPlot
  111: <reactive:plotObj>
   95: drawReactive
   82: renderFunc
   81: output$plot
    1: runApp
```

Alternatively, you can add `browser()` to any part of the code. This stops at that point and allows you to step through line by line, and check what values are being created.

```
if (input$value == "a") {
  browser()
}
# Or maybe
if (my_reactive() < 0) {
  browser()
}
```

🔧 YOUR TURN

1. We are going to trouble shoot an error. Change the `selectInput` on line 13 to `menuInput`.
2. Run the app, and watch the Console window. You should see an error, pointing to line 13.

Customising with themes and css

A range of pre-made themes is available, and can be viewed here. THese can be applied to your app in the `ui` setup:

```r
ui <- fluidPage(
  theme =
    bslib::bs_theme(bootswatch = "sandstone"),
  titlePanel("Exploring PISA Test scores"),
```

Full control over the design can be done using `css` and a good place to start is Nick Strayer's RStudio post.

Deploying your app

- Sign up for an account on https://www.shinyapps.io/
- Authenticate your account
- You may need to do some setup in your session, e.g. install the library `rsconnect`

🔧 YOUR TURN

Have a go at deploying the `pisa_app`

Embedding apps in websites, slides

A shiny app can be embedded into a presentation slide or report using:

```
knitr::include_app(
  "https://ebsmonash.shinyapps.io/VICfire/",
  height = "550px")
```

A standalone interactive graphic saved to a file can be embedded into a presenation or report using:

```
<iframe src="images/abs_ply.html" width="100%" height="500"
</iframe>
```

Note: The supporting `lib` directory needs to be in the same location as the main file.

Learning more

- https://rmarkdown.rstudio.com/flexdashboard/
- https://rstudio.github.io/shinydashboard/get_started.html
- https://mastering-shiny.org/index.html

Session Information

```r
devtools::session_info()
```

```
## ─ Session info ───────────────────────────────────────────────────────────────
##  setting  value
##  version  R version 4.1.0 (2021-05-18)
##  os       macOS Big Sur 10.16
##  system   x86_64, darwin17.0
##  ui       X11
##  language (EN)
##  collate  en_AU.UTF-8
##  ctype    en_AU.UTF-8
##  tz       Australia/Melbourne
##  date     2021-12-08
##
## ─ Packages ───────────────────────────────────────────────────────────────────
##  package       * version    date       lib source
##  anicon          0.1.0      2021-07-14 [1] Github (emitanaka/anicon@0b756df)
##  assertthat      0.2.1      2019-03-21 [1] CRAN (R 4.1.0)
##  backports       1.2.1      2020-12-09 [1] CRAN (R 4.1.0)
##  broom           0.7.9      2021-07-27 [1] CRAN (R 4.1.0)
##  bslib           0.3.1      2021-10-06 [1] CRAN (R 4.1.0)
##  cachem          1.0.6      2021-08-19 [1] CRAN (R 4.1.0)
```