

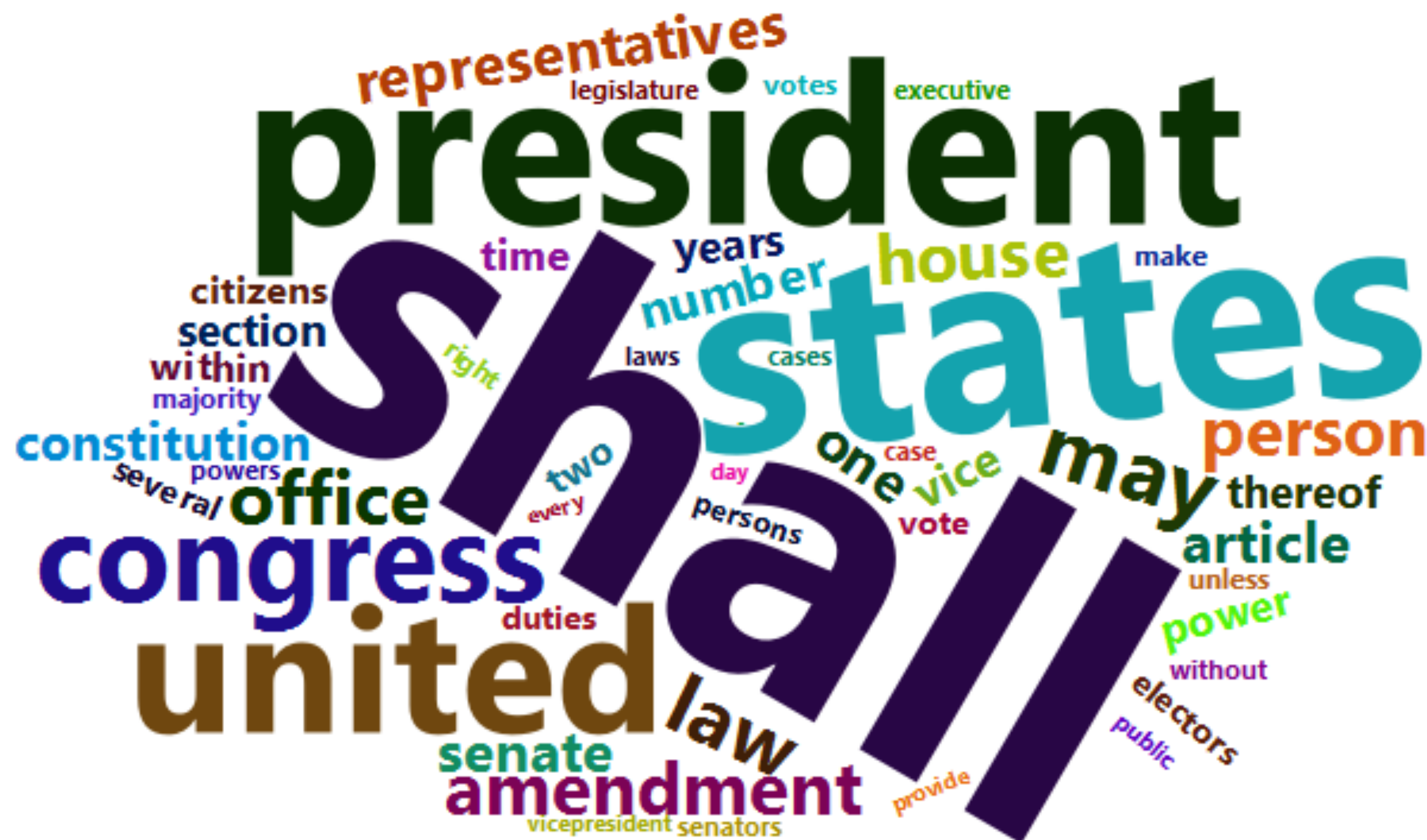


BUILDING WEB APPLICATIONS IN R WITH SHINY: CASE STUDIES

# Word clouds in Shiny

Dean Attali  
Shiny Consultant

- Visual representation of text
- BIG WORDS = COMMON, small words = rare



# Word clouds in R - function

Created by your friend:

```
create_wordcloud(data, num_words = 100, background = "white")
```

- data: text to use in word cloud
  - Single string:  
data = "Some very long story"
  - List of strings:  
data = c("Some very", "long story")
- num\_words: maximum number of words
- background: background colour

# Word clouds in R - usage

```
us_constitution <- "We the People of the United States, ..."  
create_wordcloud(data = us_constitution,  
  num_words = 15,  
  background = "yellow")
```



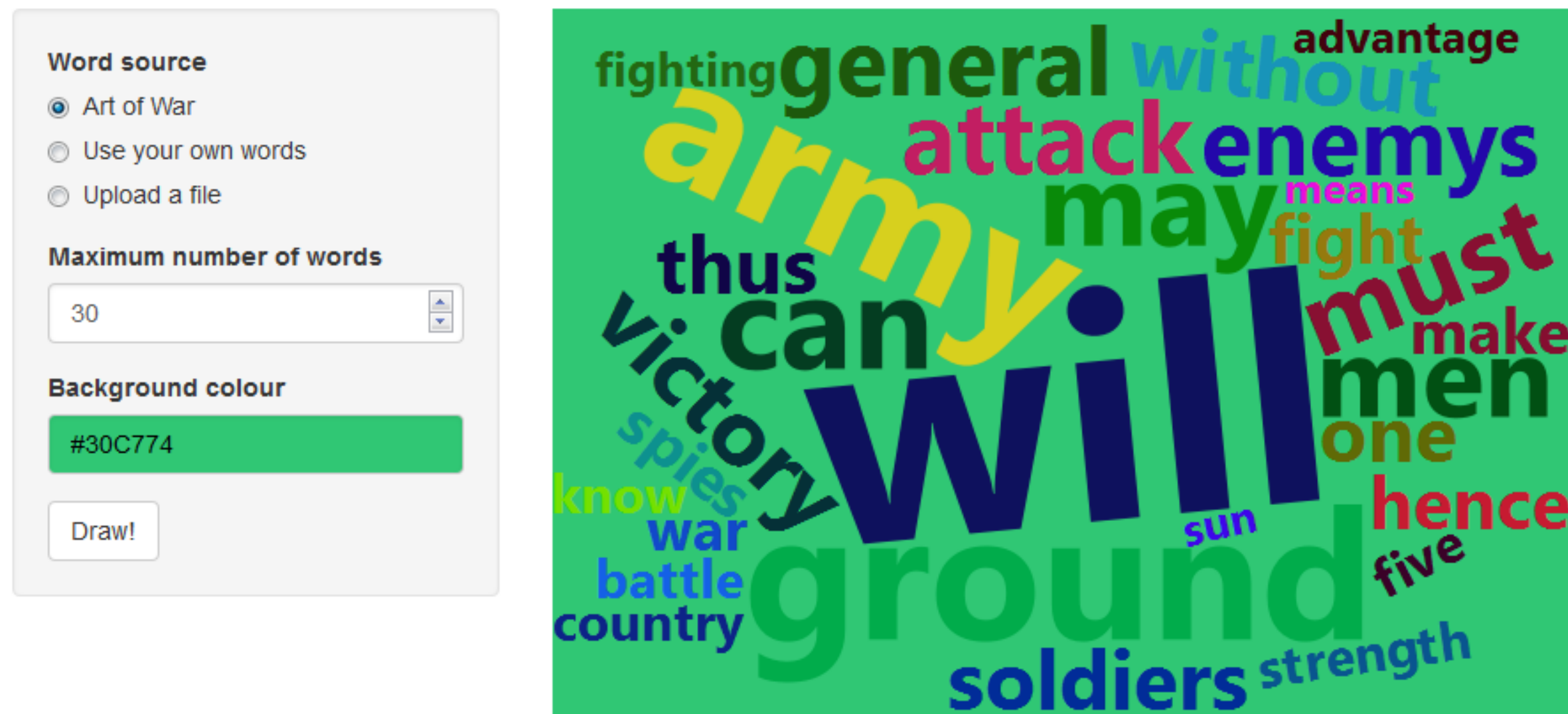
# Word clouds: from R to Shiny

- `create_wordcloud()` requires R knowledge to use
- Create Shiny app  $\Rightarrow$  anyone can create word cloud



# Word clouds in Shiny

## Word Cloud



- Word clouds are new type of output
- `wordcloud2output()` + `renderWordcloud2()`



BUILDING WEB APPLICATIONS IN R WITH SHINY: CASE STUDIES

**Let's practice!**



BUILDING WEB APPLICATIONS IN R WITH SHINY: CASE STUDIES

# Adding word sources

Dean Attali  
Shiny Consultant



# Textarea inputs

- data argument is text, use `textInput()`?

Text for word cloud

We the People of the United States, in Orde

- `textAreaInput()` similar, but provides multiple rows

```
textAreaInput(inputId, label, value, rows, ...)
```

Text for word cloud

We the People of the United States, in  
Order to form a more perfect  
Union, establish Justice, insure domestic  
Tranquility, provide for the common  
defence, promote the general Welfare,

# File inputs - UI

- File inputs for uploading a (text) file to Shiny app

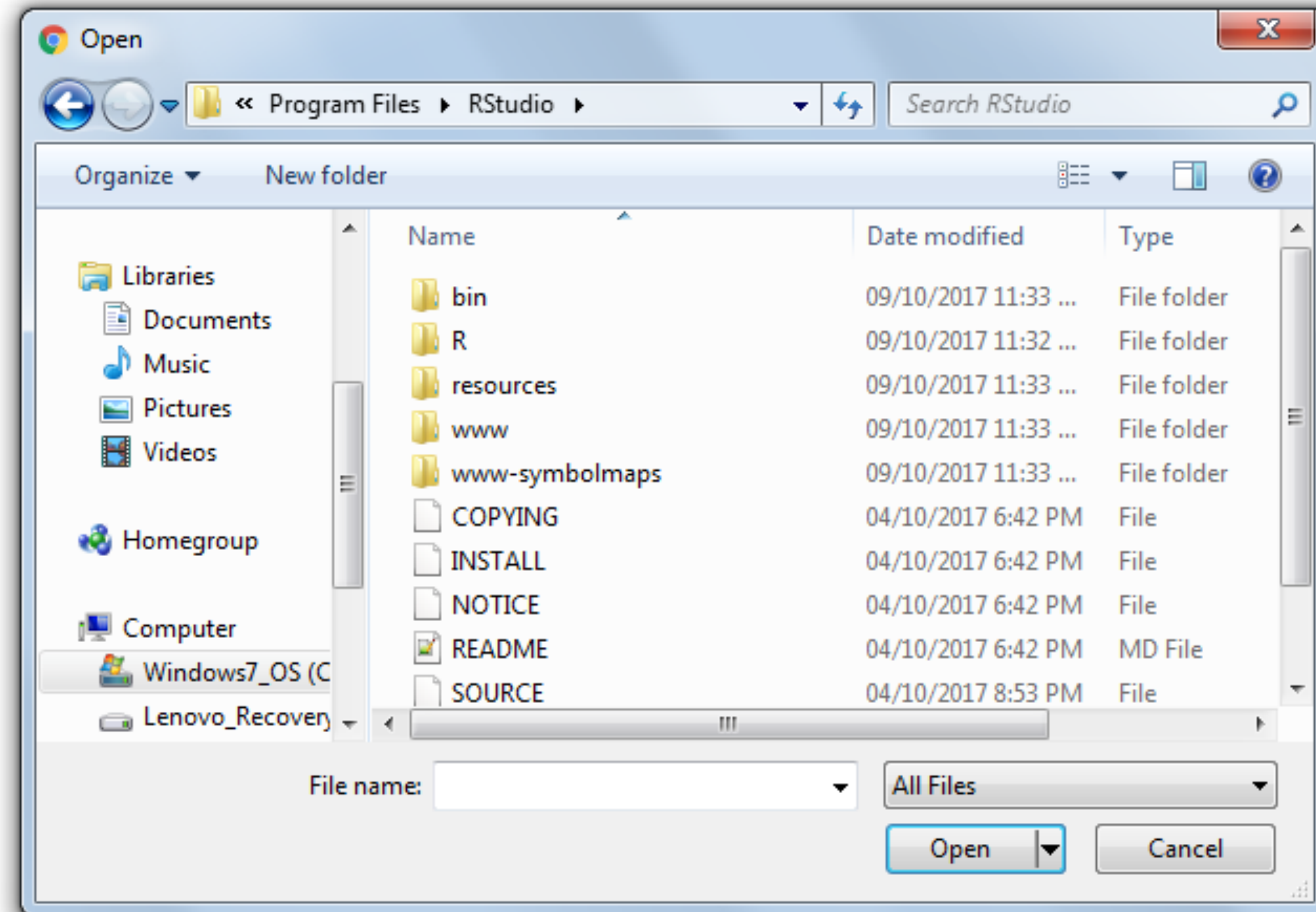
```
fileInput(inputId, label, ...)
```

Select a file

Browse...

No file selected

# File inputs - UI



?fileInput() for more options

# File inputs - server

- After selecting a file, it gets uploaded and available to Shiny
- Text inputs: `input$<inputId>` is text
- Numeric inputs: `input$<inputId>` is number
- File inputs: `input$<inputId>` is NOT a file

# File inputs - server

- File inputs: `input$<inputId>` is dataframe with 1 row per file
- Variables: `name`, `size`, `type`, `datapath`

	name	size	type	datapath
1	myfile.txt	6000	text/plain	C:/path/to/temporary/file/0.txt

- `datapath` is most important: path of file
- Read selected file:

```
input$<inputId>$datapath
```

- Text file:

```
readLines(input$<inputId>$datapath)
```



BUILDING WEB APPLICATIONS IN R WITH SHINY: CASE STUDIES

**Let's practice!**



BUILDING WEB APPLICATIONS IN R WITH SHINY: CASE STUDIES

# Combining all the word sources

Dean Attali  
Shiny Consultant



# Combining all the word sources

- 3 data sources for word cloud
  - Text object: `artofwar`
  - User text: `textAreaInput()`
  - Text file: `fileInput()`
- Next step: allow all together
  - Radio buttons to select word source

## Word source

- ☒ Art of War
- ☐ Use your own words
- ☐ Upload a file

# Radio buttons - review

```
radioButtons(  
  "time_of_day", "Choose your favourite time of day",  
  choices = c("Morning", "Afternoon", "Evening"),  
  selected = "Afternoon"  
)
```

**Choose your favourite time of day**

- ☐ Morning
- ☒ Afternoon
- ☐ Evening

# Radio buttons - advanced

```
radioButtons(  
  "time_of_day", "Choose your favourite time of day",  
  choices = c("I'm a morning person!" = "Morning",  
              "Love my afternoons"    = "Afternoon",  
              "Night owl here!"     = "Evening"),  
  selected = "Afternoon"  
)
```

**Choose your favourite time of day**

- ☐ I'm a morning person!
- ☒ Love my afternoons
- ☐ Night owl here!

```
> str(input$time_of_day)  
chr "Afternoon"
```

# Conditional panels

- Show/hide UI elements based on input value

```
conditionalPanel(condition, ...)
```

- `condition` is similar to R code, but `input$<id>` is replaced by `input.<id>`
- `...` is any UI

# Conditional panels

```
ui <- fluidPage(  
  radioButtons("time_of_day",  
    "Choose your favourite time of day",  
    ...),  
  
  plotOutput("morning_only_plot")  
)
```

```
ui <- fluidPage(  
  radioButtons("time_of_day",  
    "Choose your favourite time of day",  
    ...),  
  
  conditionalPanel(  
    condition = "input.time_of_day == 'Morning'",  
    plotOutput("morning_only_plot")  
  )  
)
```



BUILDING WEB APPLICATIONS IN R WITH SHINY: CASE STUDIES

**Let's practice!**



BUILDING WEB APPLICATIONS IN R WITH SHINY: CASE STUDIES

# **Fine tune the reactivity**

**Dean Attali**  
Shiny Consultant



# Reactivity review

- `reactive()` and `input$` are reactive
- Code depending on reactive variables re-runs when dependencies update
- Accessing reactive value makes it dependency

```
x <- reactive({  
  y() * input$num1 * input$num2  
})
```

# Isolate

- Use `isolate()` to not create reactive dependency
- If reactive value inside `isolate()` is modified, nothing happens

```
x <- reactive({  
  y() * isolate({ input$num1 }) * input$num2  
})
```

```
x <- reactive({  
  y() * isolate({ input$num1 * input$num2 })  
})
```

# Isolate everything

- Sometimes you want to isolate all reactives

```
x <- reactive({  
  isolate({  
    y() * input$num1 * input$num2  
  })  
})
```

- Need a way to trigger x to re-run on demand

# Action buttons

```
actionButton(inputId, label, ...)
```



- Only one simple interaction: click
- Value of button is number of times it was clicked

```
# After clicking on a button twice  
> str(input$button)  
int 2
```

# Action buttons as reactivity triggers

- Accessing button input value in server triggers reactivity
- Add button to UI

```
actionButton(inputId = "calculate_x", label = "Calculate x!")
```

- Access button to make it dependency

```
x <- reactive({  
  input$calculate_x  
  
  isolate({  
    y() * input$num1 * input$num2  
  })  
})
```



BUILDING WEB APPLICATIONS IN R WITH SHINY: CASE STUDIES

**Let's practice!**



BUILDING WEB APPLICATIONS IN R WITH SHINY: CASE STUDIES

**Wrap-up: Go and make  
your own apps!**

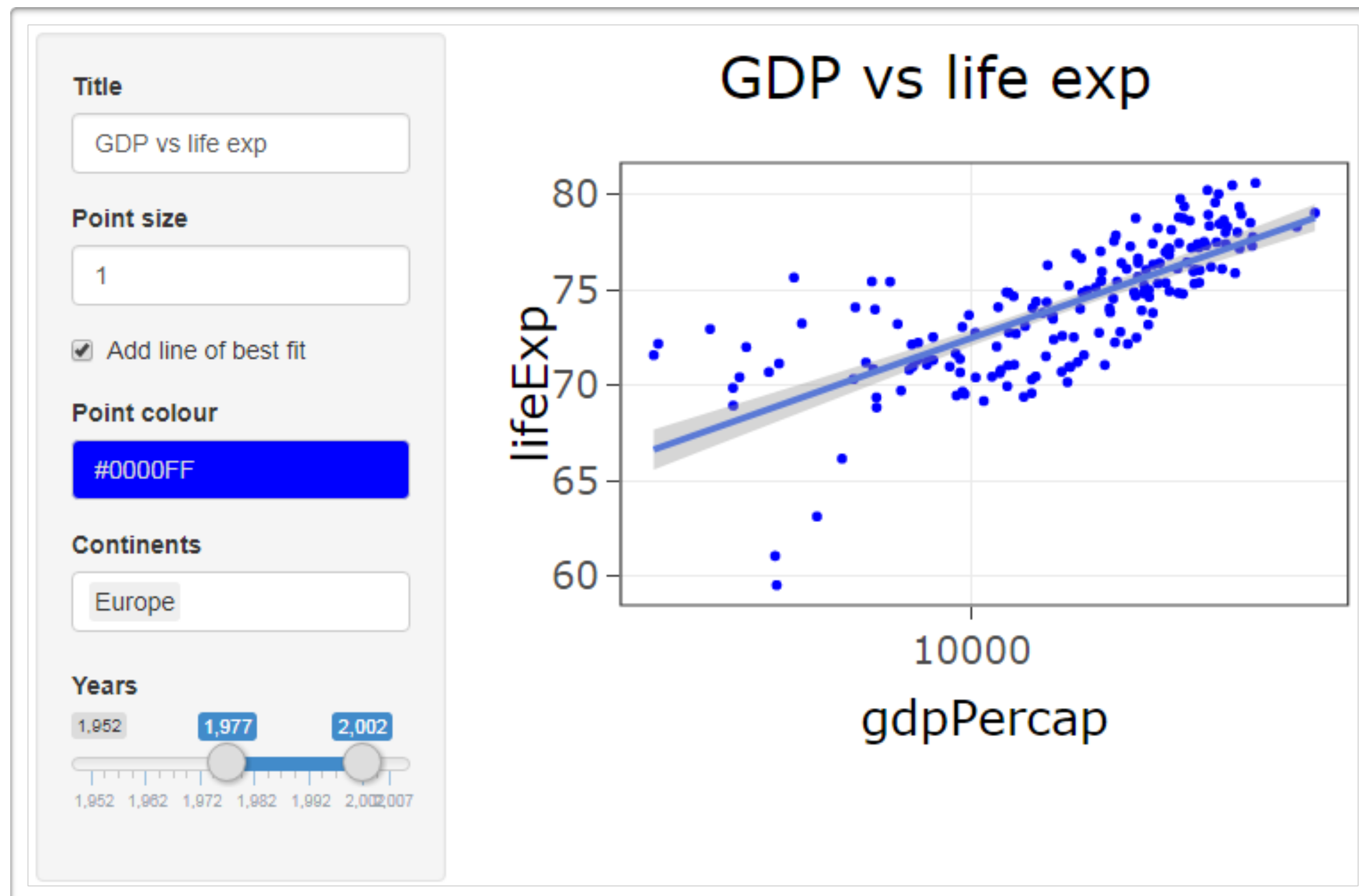
**Dean Attali**  
Shiny Consultant





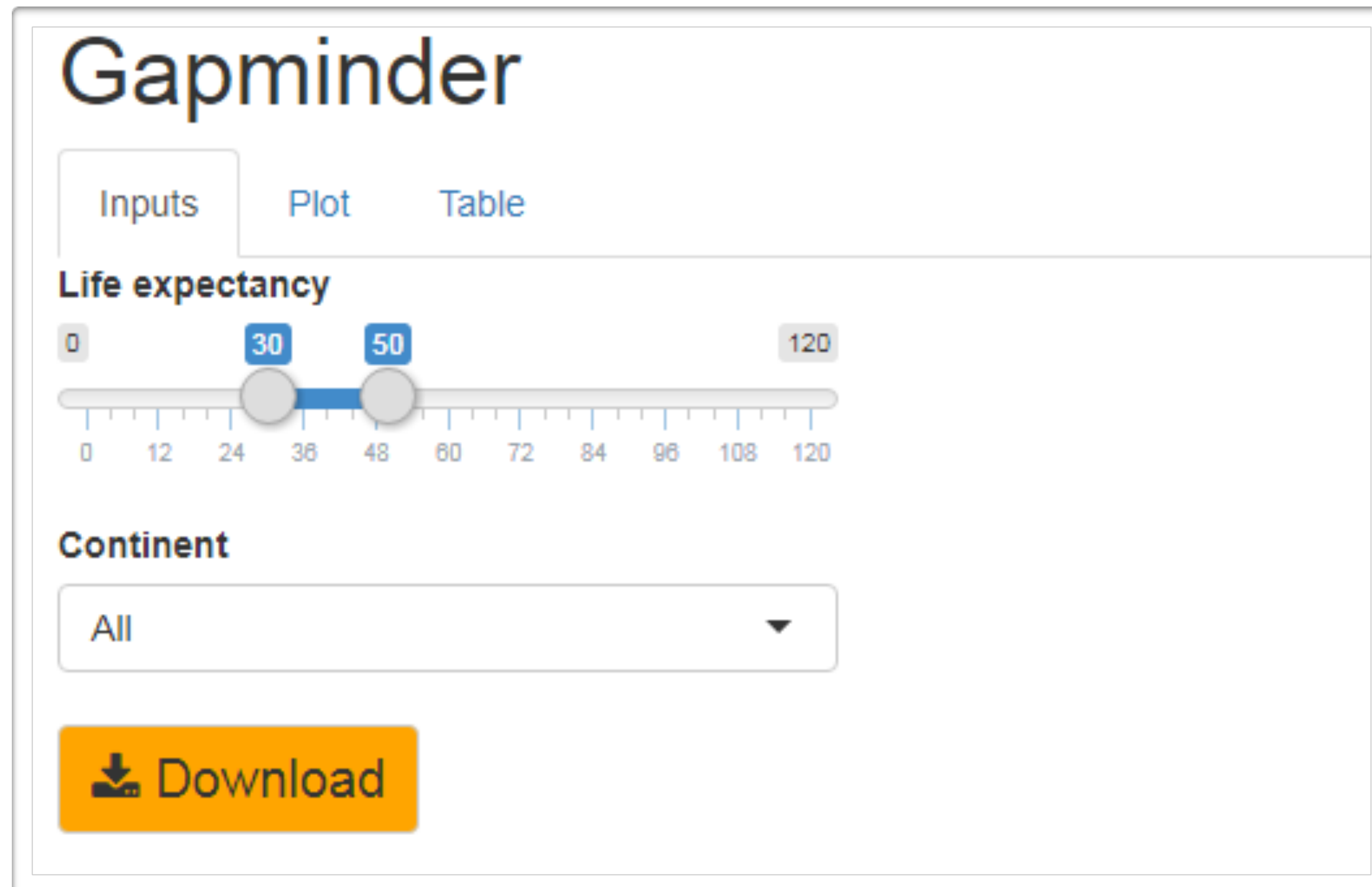
**CONGRATS!**

# Chapter 2: Plotting app



Shiny is great for customizing plots with many parameters

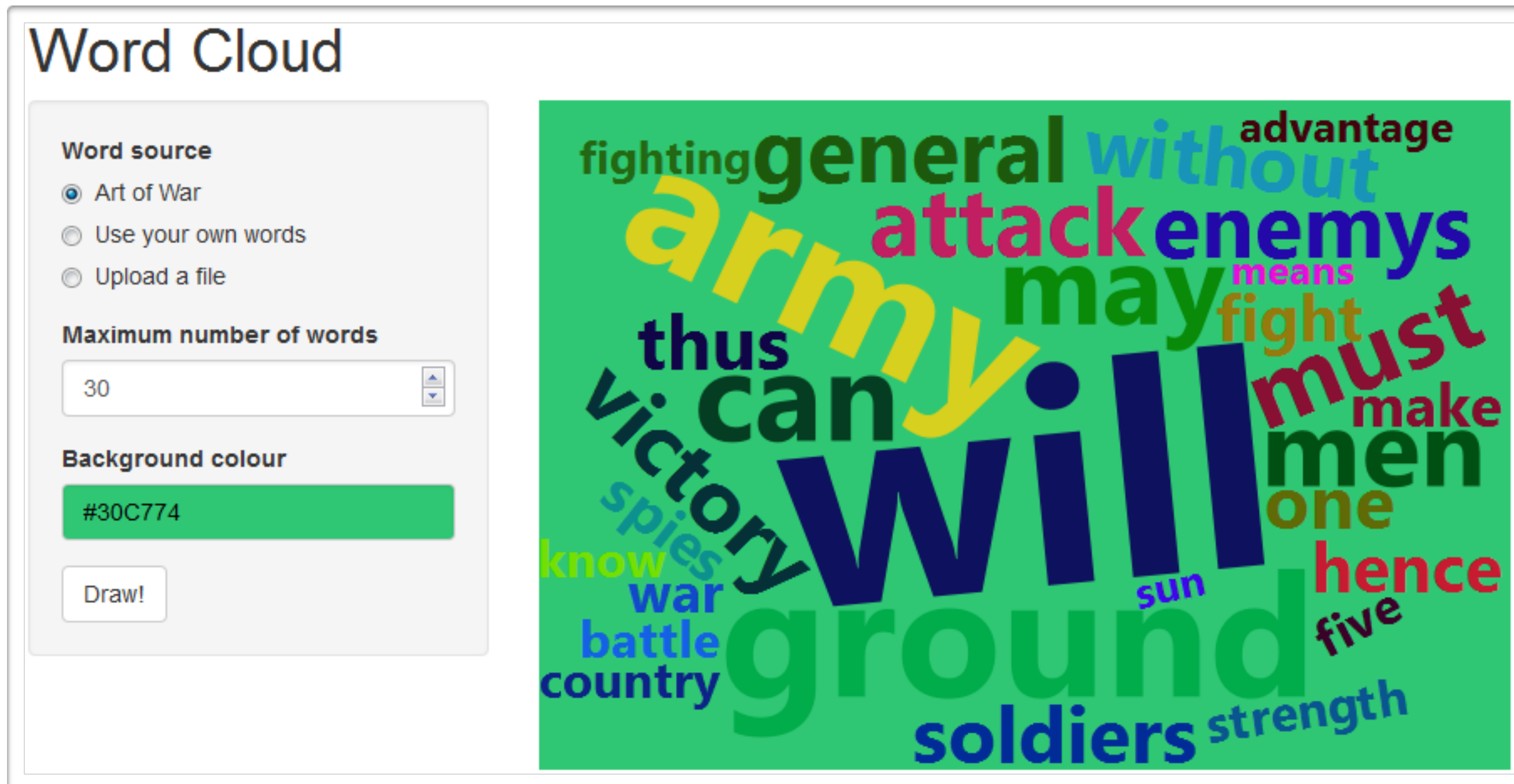
# Chapter 3: Data exploration app



Shiny is great as a data exploration tool.

Think about ease of use and user experience, not only functionality.

# Chapter 4: Cloud word app



Shiny is great for exposing R code as graphical interface, or for sharing your R code with non R users.



BUILDING WEB APPLICATIONS IN R WITH SHINY: CASE STUDIES

**Your turn!**