Word clouds in Shiny

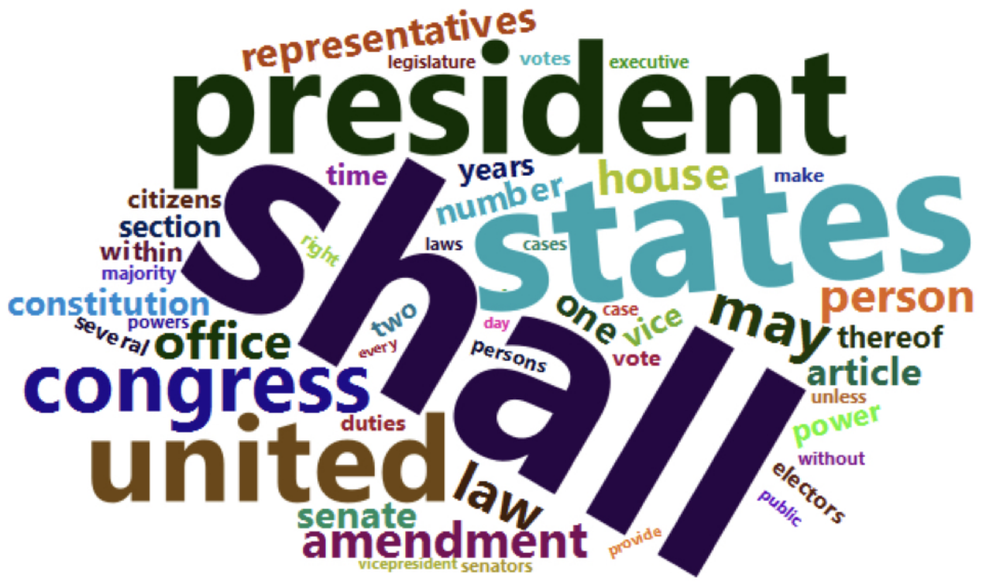
CASE STUDIES: BUILDING WEB APPLICATIONS WITH SHINY IN R

Dean Attali
Shiny Consultant

datacamp

Word clouds

- 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 color

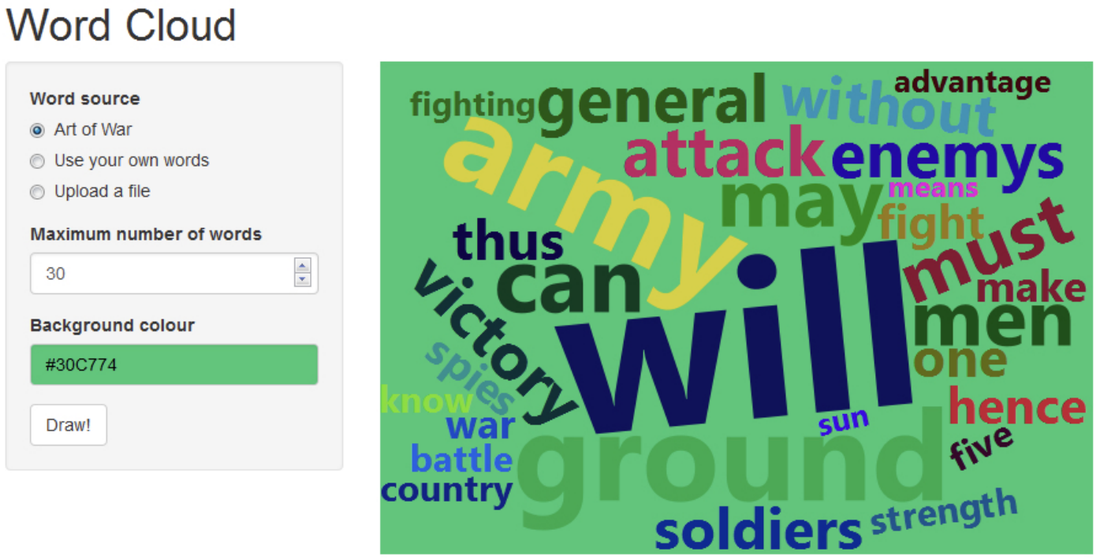
Word clouds in R - usage

```
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 ⇒ anyone can create a word cloud

Word clouds in Shiny



- Word clouds are new type of output
- `wordcloud2Output()` + `renderWordcloud2()`

Let's practice!

CASE STUDIES: BUILDING WEB APPLICATIONS WITH SHINY IN R

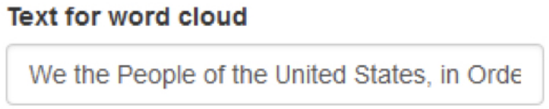
Adding word sources

CASE STUDIES: BUILDING WEB APPLICATIONS WITH SHINY IN R

Dean Attali
Shiny Consultant

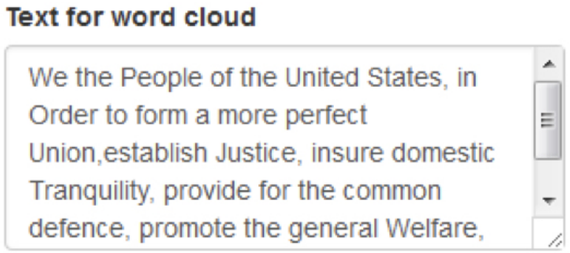
Textarea inputs

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



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

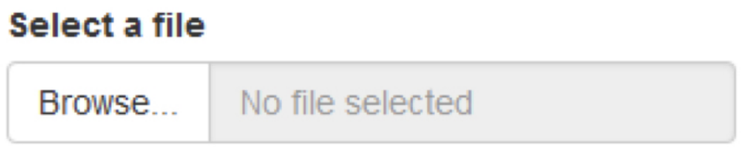
```
textAreaInput(inputId, label, value, rows, ...)
```



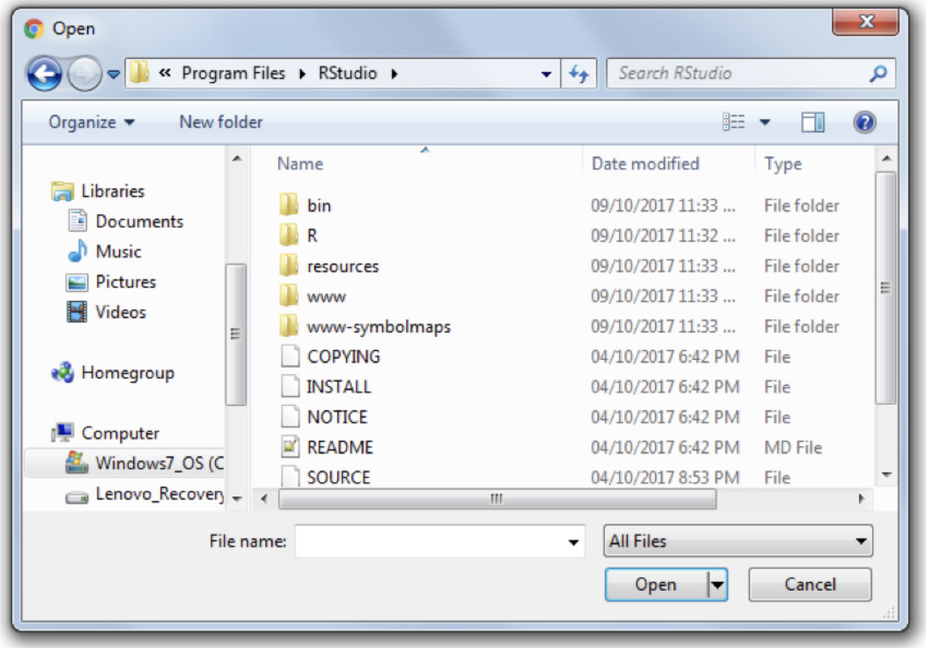
File inputs - UI

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

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



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 a 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)
```

Let's practice!

CASE STUDIES: BUILDING WEB APPLICATIONS WITH SHINY IN R

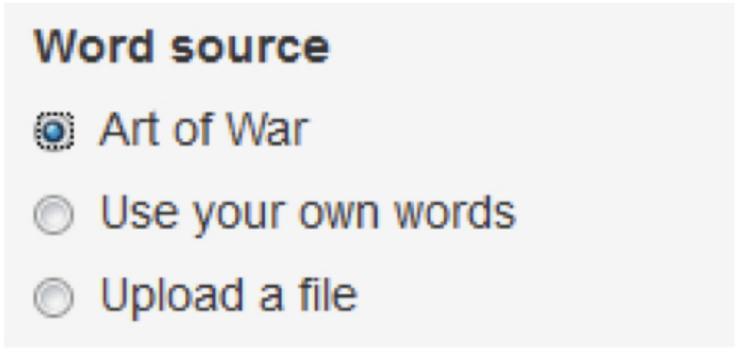
Combining all the word sources

CASE STUDIES: BUILDING WEB APPLICATIONS WITH SHINY IN R

Dean Attali
Shiny Consultant

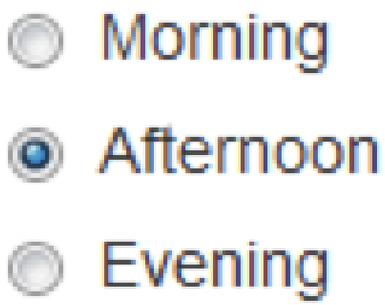
Combining all the word sources

- 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



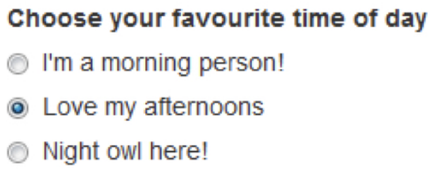
Radio buttons - review

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



Radio buttons - advanced

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




```
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 favorite time of day",
               ...),

  plotOutput("morning_only_plot")
)
```

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

  conditionalPanel(
    condition = "input.time_of_day == 'Morning'",
    plotOutput("morning_only_plot")
  )
)
```

Let's practice!

CASE STUDIES: BUILDING WEB APPLICATIONS WITH SHINY IN R

Fine tune the reactivity

CASE STUDIES: BUILDING WEB APPLICATIONS WITH SHINY IN R

Dean Attali
Shiny Consultant

datacamp

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
    })
})
```

Let's practice!

CASE STUDIES: BUILDING WEB APPLICATIONS WITH SHINY IN R

Wrap-up: Go and make your own apps!

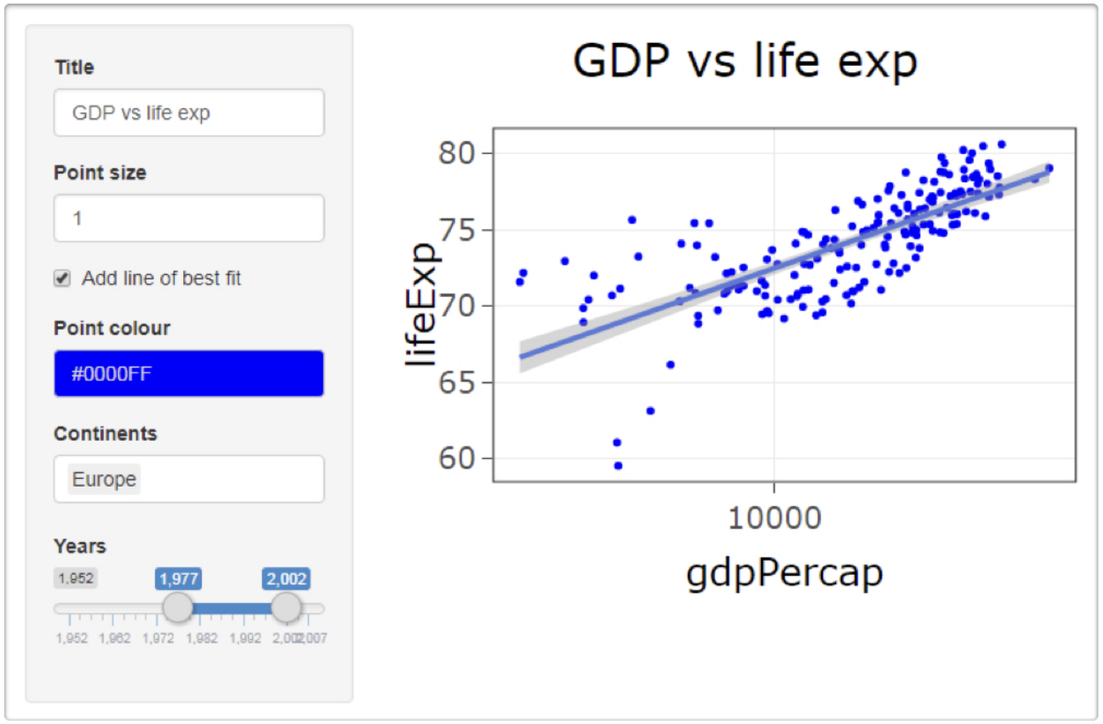
CASE STUDIES: BUILDING WEB APPLICATIONS WITH SHINY IN R

Dean Attali
Shiny Consultant


CONGRATS!

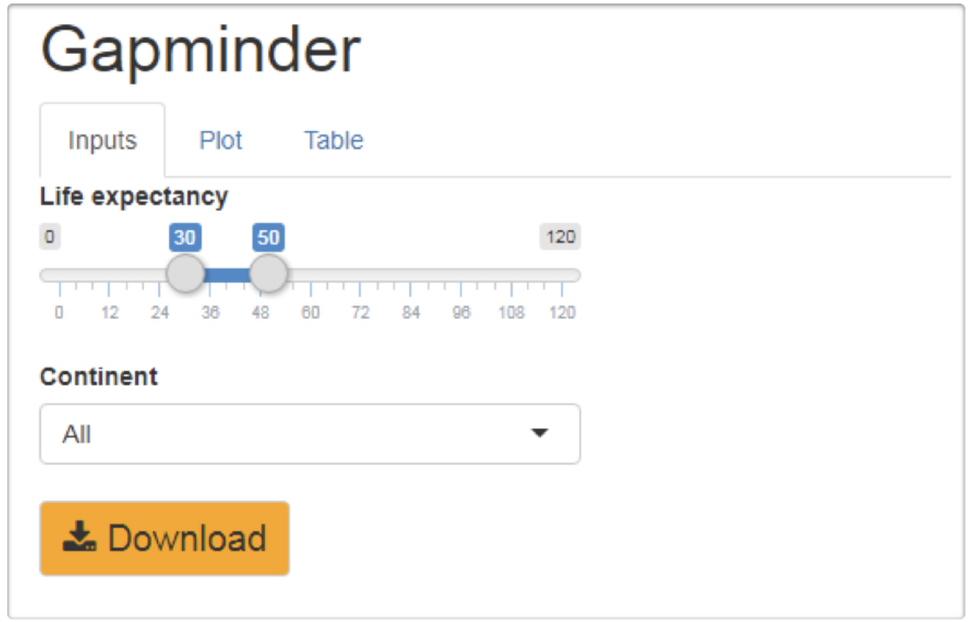
Chapter 2: Plotting app

- 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: Word cloud app

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

Let's practice!

CASE STUDIES: BUILDING WEB APPLICATIONS WITH SHINY IN R