HOW TO BUILD A MID-SIZED ANALYTICAL APPLICATION WITH R + SHINY

30h March 2015

Delivering Transformation. Together.

sopra steria

AGENDA

- About me
- The context / Why R + shiny
- Editors & IDEs
- Client vs. Server-side coding & data-driven UIs
- Reactivity
- Advanced Visualisation & Htmlwidgets
- Debugging
- The developing environment
- The Development team
- Summary

ABOUT ME

- Principal Consultant at SopraSteria since 2010
- Focusing on: NoSQL Architectures - Data Science & Data Visualization
  - +18 years experience in business analytics / data warehousing
  - +3 years experience in R & data science
  - +1 year experience in shiny
- Github page: smartinsightsfromdata

- Web:
  - www.smartinsightsfromdata.com (under construction!)
- Contact email :
  - enzo.martoglio@soprasteria.com work
  - Enzo@smartinsightsfromdata.com personal

THE CONTEXT / WHY R + SHINY

- Requirements
  - Data entry
  - Quick prototyping
  - 9 business areas (more in the future)
  - Extensible without coding
  - >20 users
  - Need for:
    - advanced visualisation
    - Montecarlo simulation / forecasting (future phases)

EDITORS & IDEs

- RStudio
  - Data analysis
  - Project management (project folder)
  - Debugging
- Not enough!
- SublimeText
  - Coding page-wide / side-by-side
  - compatible with Rstudio project folder
  - Syntax checking
- SublimeText Extensions
  - Bracket highlighter
  - R-box
  - SublimeREPL

CLIENT VS. SERVER-SIDE CODING & DATA-DRIVEN UIS

- Html / JS code on the client
- Data sent to the client
- Example (ui.R)

```
sliderInput(inputId, label, min, max,
value, step = NULL, round = FALSE,
format = NULL, locale = NULL, ticks =
TRUE, animate = FALSE, width =
NULL, sep = ",", pre = NULL, post =
NULL)
```

- Html / JS code sent to the client
- Data sent to the client
- Example (server.R)

```
 lapply(1:iter, function(i) {
sliderInput(inputId =
paste0(cidRes$LPA[i], "_v1"), label =
paste0(cidRes$LPA[i],"headcount","")
, min = round(cidRes$head[i]*0.2),
max = round(cidRes$head[i]*1.8),
value = cidRes$head[i], ticks = FALSE,
animate = FALSE) })
```

- Using renderUI

REACTIVITY

Wikipedia

- In computing, reactive programming is a programming paradigm oriented around data flows and the propagation of change.
- For example, in an imperative programming setting, a := b + c would mean that a is being assigned the result of b + c in the instant the expression is evaluated. Later, the values of b and c can be changed with no effect on the value of a.
- In reactive programming, the value of a would be automatically updated based on the new values.

Features

- Fat client / single page
- a publish–subscribe pattern to automatically propagate data changes to clients in real-time without requiring the developer to write any synchronization code

REACTIVITY EXAMPLE

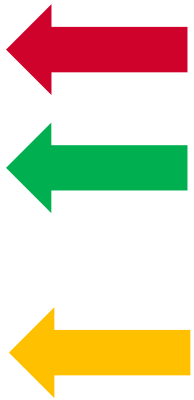
```
observe({
if(input$saveButtn == 0) return()

isolate({

if(is.null(rmod()  ) ) return()
if(is.null(input$inSubDep)) return()

[…]

status<- fSave(…)
}) # end of isolate
})
```



ADVANCED VISUALISATION & HTMLWIDGETS

- Today the most advanced visualisations are coded in Javascript in libraries like:
  - D3
  - C3
  - NVD3
  - And many many others
- Htmlwidgets is a wrapper of Javascript libraries
  - Use JavaScript visualization libraries at the R console, just like plots
  - Embed widgets in R Markdown documents and Shiny web applications
  - Develop new widgets using a framework that seamlessly bridges R and JavaScript
- My htmlwidgets (available on my github pages, on Cran in the future)
  - rpivotTable
  - Rdatamaps
  - Rd3pie
  - Everybody is welcome to download them and use them!!

DEBUGGING

- Print
- Browser() (wrapper of debug())
- options(shiny.trace = T)
- options(shiny.reactlog=TRUE)
- Learn to use the javascript console!

THE DEVELOPING ENVIRONMENT

- Need to bridge different environments
- Client (browser choice: IE)
- Shiny Pro Server(Linux)
- Development in Windows and OSX

- (Why we didn’t use dockers?!?s)

CONCLUSIONS

- R + shiny are fit for complex analytical applications
- You *mainly* need R + shiny skills (but html, css, JS, D3, chrome dev. Etc. are useful!)
- Today interactivity and advanced visualization are a must for collaborative data analysis
- Plan your widgets carefully
- Work at the bleeding edge, but use your wisdom…

Delivering Transformation. Together.