An introduction to Shiny

John Henderson

08 February 2014

Intro

- `shiny` is an `R` package that enables web based applications
- Overview of `shiny` basics
- Three examples
- The code/data necessary to reproduce anything in this talk is on github

Basics

- `shiny` works inside of RStudio
- Two files are required to run an application
- `ui.R`: page format, user inputs, and outputs you're going to create
- `server.R`: contains the R code which will generate your dynamic output

Don't forget to run `install.packages("shiny")`!

Minimal `ui.R`

```
library(shiny)
# page format
shinyUI(pageWithSidebar(
  # title
  headerPanel("Hello Shiny!"),

  sidebarPanel(
    # user inputs go here
  ),

  mainPanel(
    plotOutput("plot") # what you're going to output, e.g. a plot
  )
))
```

Minimal server.R

```
library(shiny)

shinyServer(function(input, output) {

  # general R code here: load libraries, set variables/functions/etc.

  # output$name has to match ui.R's plotOutput("name")
  output$plot <- renderPlot({

    # code to make a plot goes here

  })
})
```

It works!

- After defining the above files...

```
# run from within R studio
library(shiny)
setwd("/path/to/ui-and-server.R")
runApp()
```

Example: data analysis/exploration

- Enable rapid and dynamic switching of plot variables
- Allows for "plot prototyping" to examine trends/relationships
- Web-based solution is easily sharable with others

Fiddling with public transportation data

- Grabbed data on public transportation centers around US (more here)
- Some are quite efficient, some are horrible
- Can `shiny` help find some interesting tidbits?

Fiddling with public transportation data

- Grabbed data on public transportation centers around US (more here)
- Some are quite efficient, some are horrible
- Can `shiny` help find some interesting tidbits?

Demo time!

Example: interactive contour plots

- Applied machine learning in R on product test data
- Contour plots can be nice for visualizing effect of inputs vs. outputs
- How to share the results with co-workers who don't use R?

Example: interactive contour plots

- Applied machine learning in R on product test data
- Contour plots can be nice for visualizing effect of inputs vs. outputs
- How to share the results with co-workers who don't use R?

Demo time!

Example: visualizing insurance costs

- Benefit plan choices are tough!
- Started making visualizations/walkthroughs at 3M in 2011
- Goal: simplify decision process through visualization

The main issue

- HR typically sends you a table like this on glossy paper; which plan is best?

	Plan A	Plan B
Premium	\$150/mo	\$250/mo
3M Contribution	\$1,000	\$0
Deductible	\$2,500	\$750
OOP_{max}	\$5,000	\$4,000

The main issue

- These employees are not smiling because they understood the table

Image credit: http://jtsfs.com/employee-benefits-2/group-health-insurance/

In 2011, it was so simple!

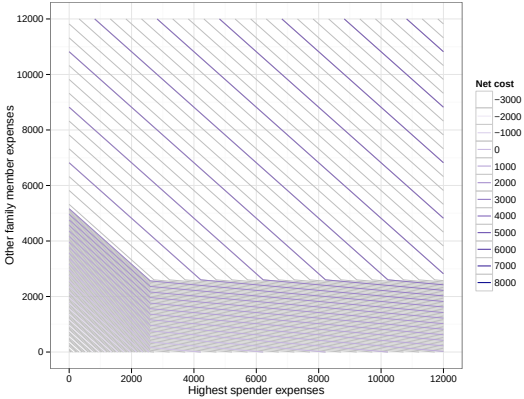
Fast-foward to 2013

- 3M introduces split deductibles on two plans
- Now which plan is best?

Plan	Premium	Ded_{ind}	Ded_{tot}	OOP_{ind}	OOP_{tot}	*HSA*
A	\$3,500	\$500	\$1,000	\$2,000	\$4,000	-
B	\$2,200	-	\$2,750	-	\$5,500	\$1,250
C	\$600	\$2,750	\$5,500	\$5,500	\$11,000	\$1,250

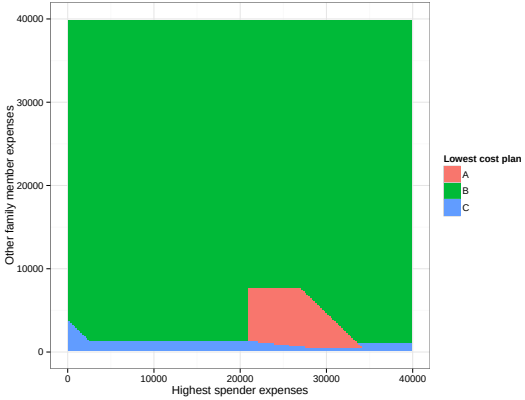
First shot

- Now we need axes for max spender vs. everyone else... contour plot!



Winning cost map

- "Stack" the contours, figure out which one is lowest



So, what about *this* year?

- I used `shiny`, obviously!
- Dynamic UI elements for # of people on plan
- "Interesting" algorithm for dealing with complex criteria
- Hosted internally at 3M with `shiny-server`
- Put an anonymized version on RStudio server

Table of possible outcomes

ded_{ind}	oop_{ind}	ded_{rem}	oop_{rem}	ded_{tot}	oop_{tot}	bin	formula
0	0	0	0	0	0	0	$exp_{ind} + exp_{rem}$
1	0	0	0	0	0	1	$ded_{ind} + 0.1\ (exp_{ind}$ - $ded_{ind}) + exp_{rem}$
0	0	1	0	0	0	4	$exp_{ind} + exp_{rem}$
1	0	0	0	1	0	17	$ded_{ind} + 0.1\ (exp_{ind}$ - $ded_{ind}) + exp_{rem}$
1	1	0	0	1	0	19	$oop_{ind} + exp_{rem}$
0	0	1	0	1	0	20	$ded_{tot} + 0.1\ (exp_{ind} + exp_{rem}$ - $ded_{tot})$
1	0	1	0	1	0	21	$ded_{tot} + 0.1\ (exp_{ind} + exp_{rem}$ - $ded_{tot})$
1	1	1	0	1	0	23	$oop_{ind} + ded_{ind} + 0.1\ (exp_{rem}$ - $ded_{ind})$
1	0	1	1	1	0	29	$ded_{tot} + 0.1\ (exp_{ind} + exp_{rem}$ - $ded_{tot})$
1	1	0	0	1	1	51	$oop_{ind} + exp_{rem}$
1	1	1	0	1	1	55	$oop_{ind} + ded_{ind} + 0.1\ (exp_{rem}$ - $ded_{ind})$
1	0	1	1	1	1	61	oop_{tot}
1	1	1	1	1	1	63	oop_{tot}

Check against criteria; convert to binary

```
test_case <- c(rep(c(exp_ind, exp_rem, exp_ind + exp_rem),   # vector of predicted costs
                   each = 2))                                # for max vs. others

test_case <- rbind(test_case, test_case, test_case)          # three sets for three plans

limits <- cbind(compare$ded_ind, compare$exp_max_ind,        # criteria values
                compare$ded_ind, compare$exp_max_ind,
                compare$ded_tot, compare$exp_max_tot)

result <- cbind(compare[, c("ded_ind", "ded_tot", "oop_ind", # store cutoffs in result
                            "oop_tot", "prem", "hsa")],
                exp_ind, exp_rem,
                (test_case > limits) %*% (2^(0:5)))          # convert T/F to binary
```

Hacky function lookup

```
map_funcs <- list(
  "0" = function(binary) { binary$exp_ind + binary$exp_rem },
  "1" = function(binary) { binary$ded_ind + (0.1* (binary$exp_ind - binary$ded_ind)) + binary$exp_rem },
  "4" = function(binary) { binary$exp_ind + binary$exp_rem },
  "16" = function(binary) { binary$ded_tot + (0.1 * (binary$exp_ind + binary$exp_rem - binary$ded_tot)) },
  "17" = function(binary) { binary$ded_ind + (0.1* (binary$exp_ind - binary$ded_ind)) + binary$exp_rem },
  "19" = function(binary) { binary$oop_ind + binary$exp_rem },
  "20" = function(binary) { binary$ded_tot + (0.1 * (binary$exp_ind + binary$exp_rem - binary$ded_tot)) },
  "21" = function(binary) { binary$ded_tot + (0.1 * (binary$exp_ind + binary$exp_rem - binary$ded_tot)) },
  "23" = function(binary) { binary$oop_ind + binary$ded_ind + (0.1 * (binary$exp_rem - binary$ded_ind)) },
  "28" = function(binary) { binary$ded_tot + (0.1 * (binary$exp_ind + binary$exp_rem - binary$ded_tot)) },
  "29" = function(binary) { binary$ded_tot + (0.1 * (binary$exp_ind + binary$exp_rem - binary$ded_tot)) },
  "48" = function(binary) { binary$oop_tot },
  "51" = function(binary) { binary$oop_ind + binary$exp_rem },
  "55" = function(binary) { binary$oop_ind + binary$ded_ind + (0.1 * (binary$exp_rem - binary$ded_ind)) },
  "60" = function(binary) { binary$oop_tot },
  "61" = function(binary) { binary$oop_tot },
  "63" = function(binary) { binary$oop_tot }
)
```

'Nuff talk, let's take a look!

Sharing `shiny` apps

- Method 1: tar/zip all files, send, have user run locally
- Method 2: install shiny-server on local machine
- Method 3: request account for RStudio server account (still available?)
  - Create/upload files; `http://spark.rstudio.com/uname/appName`
- Method 4: request account on *new* RStudio server here
  - Create apps locally, then follow shinyapps instructions
  - When satisfied, just run `deployApp()`!
  - Visit app at `http://uname.shinyapps.io/appName/`

References

- Getting started with `shiny`
- `shiny` mailing list
- RStudio server application
- SO question on creating dymanic input elements
- SO question on global variables (not intuitive!)
- SO question on sizing plots in `shiny`
- SO question that solved my contour plot issue; repaid with `shiny` example

Apps in this presentation

- Transpo exploration: spark.rstudio or shinyapps.io
- Interactive contour
- Benefit analysis
- Everything's also on github!