An introduction to Shiny

John Henderson

08 February 2014

Intro

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

Basics

- `shiny` works inside of RStudio
- Two files are required to run an application
- `ui.R`: sets page format, user input elements, 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 rapid plot prototyping to examine trends/relationships
- Web-based solution is easily sharable with others

Fiddling with public transportation data

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

Fiddling with public transportation data

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

Demo time!

Example: visualizing insurance costs

- Benefit plan choices are hard; getting harder
- Started making visualizations/walkthroughs in 2011
- Goal: simplify decision making process at 3M

In general, insurance is simple

There are three "phases" of employee out of pocket expenses:

- If the deductible has not been met, employee pays in full
- Once deductible is met, employee pays 10% coinsurance
- When OOP_{max} is reached, the employee pays nothing further

To find OOP_{max}

$$OOP_{max} = Ded + (0.1 \times (Exp_{max} - Ded)) \; ; \; \frac{OOP_{max} - Ded}{0.1} + Ded$$

2011: it was simple back then

- What employees are given... 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

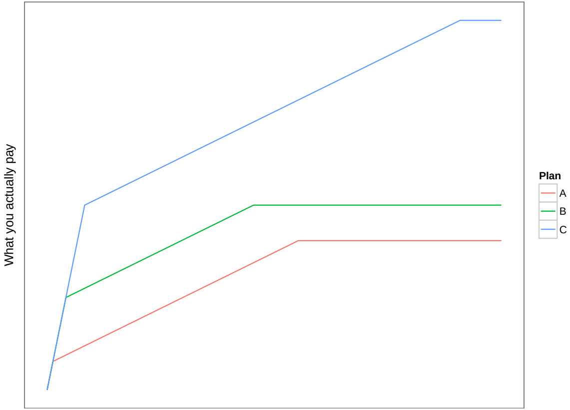
Perhaps math makes it easier?

Let $I(x)$ represent OOP over a range of $Expenses$:

$$I(x) = \begin{cases} Expenses & \text{if} \quad 0 < x < Ded \\ Ded + (0.10 \times (Expenses - Ded)) & \text{if} \quad Ded \leq x < Expenses_{max} \\ OOP_{max} & \text{if} \quad Expenses_{max} \leq x < \infty \end{cases}$$

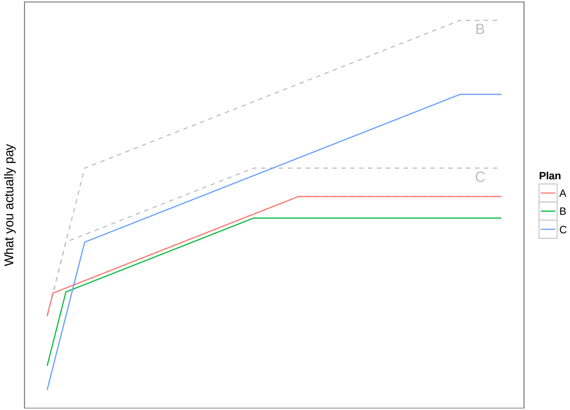
But doesn't visualization take the cake?

- Cost, apparent



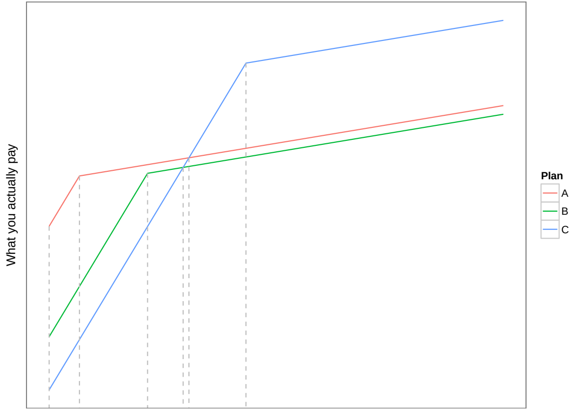
But doesn't visualization take the cake?

- Cost, adjusted for company HSA contribution



But doesn't visualization take the cake?

- Handy dandy intersection points



Let's go all n-dimensional

Split deductible system for Plans B and C

- If a single individual reaches Ded_{ind}, he/she covered at 90%
- Whole family covered when Ded_{fam} is met
- Similarly, OOP_{max} is split ($OOP_{max-ind}$ and $OOP_{max-fam}$)

For example:

- Assume only one family member incurs any medical expenses
- Individual covered at 90% once Ded_{ind} is reached
- Costs capped for individual once the *individual* OOP_{max} is met

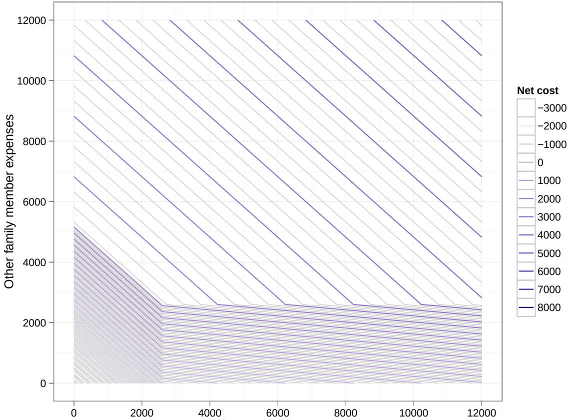
Still so simple?

- Now which plan is best?

Plan	Premium	Ded_{ind}	Ded_{tot}	OOP_{ind}	OOP_{tot}	HSA
A	\$3,120	\$400	\$800	\$2,100	\$4,200	-
B	\$2,088	-	\$2,600	-	\$5,200	\$1,200
C	\$504	\$2,600	\$5,200	\$5,200	\$10,400	\$1,200

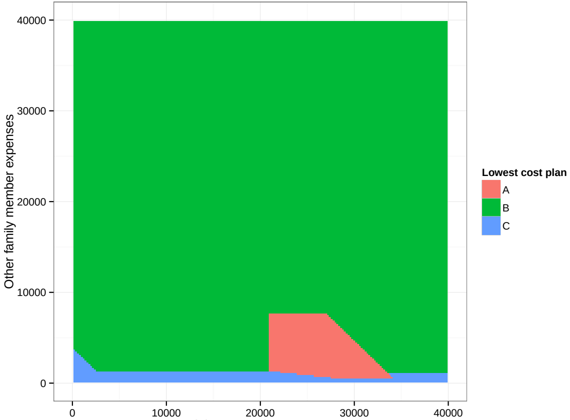
First shot

- Contour of highest spender vs. everyone else



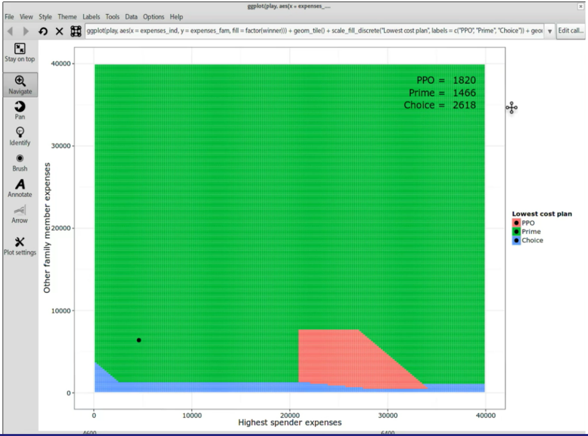
Winning cost map

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



Interaction was already on the horizon

- Demo showing `playwith` + `ggplot2`



So, what about *this* year?

- I used `shiny`, obviously!
- Dynamic UI elements for # of people on plan
- "Interesting" algorithm for dealing with complex criteria
- A bit of `ggplot2` hackery
- Hosted internally at 3M on `shiny` server
- Put an anonymized version on spark.rstudio.com

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}

Break up expenses: highest vs. the rest

```
converter <- function(expenses) {

  exp_ind <- max(expenses)
  exp_rem <- sum(expenses[-which(expenses == exp_ind)[1]])
  list("exp_ind" = exp_ind, "exp_rem" = exp_rem)

}
```

Check against criteria; convert to binary

```
condition <- function(exp_ind, exp_rem, class) {

  compare <- plans[plans$class == class, ]

  test_case <- c(rep(c(exp_ind, exp_rem, exp_ind + exp_rem), each = 2))
  test_case <- rbind(test_case, test_case, test_case)

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

  result <- cbind(compare[, c("ded_ind", "ded_tot", "oop_ind", "oop_tot", "prem", "hsa")],
                  exp_ind, exp_rem, (test_case > limits) %*% (2^(0:5)))
  names(result)[ncol(result)] <- "bin"
  return(result)

}
```

Hacky function lookup

```
map_funcs <- list()
length(map_funcs) <- 17
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 }
)
```

Creating the right data to plot

```
generate_plot_data <- function(binary) {

  plot <- lapply(1:nrow(binary), function(i) {
    temp <- binary[i, ]
    delta <- temp$cost - temp$hsa - hsa_vol
    plot <- data.frame(plan = rep(temp$plan, 4),
                       start = c(min(delta, 0),
                       temp$prem, max(0, delta),
                       c(temp$cost, temp$hsa)[(delta > 0)+1]))
    plot[plot$plan == "PPO", "start"][1] <- 0
    offsets <- c(0, cumsum(plot$start[2:4]))
    plot$end <- offsets - abs(plot$start)
    plot$start <- offsets
    plot$fill <- factor(c("c", "b", "a", "a"))
    plot$alpha <- factor(c(1, 1, 1, 0))
    return(plot)
  } )

}
```

The plot

```
p <- ggplot(plot, aes(x = plan, xend = plan,
                      y = start, yend = end,
                      colour = fill, alpha = alpha))
p <- p + geom_segment(size = 35) + theme_bw()
p <- p + coord_flip() + facet_wrap(~case, ncol = 2)
p <- p + scale_alpha_discrete(range = c(0.35, 1), guide = F)
p <- p + scale_colour_manual("Annual Cost", limits = c("a", "b", "c", "d"),
         labels = c("Expenses", "Premiums", "Carry-over HSA",
                    "Expenses paid \n from HSA/HCRA"),
         values = hcl(c(15, 255, 135, 15), l=65, c=100, alpha = c(1, 1, 1, 0.35)))
p <- p + scale_y_continuous(limits = c(min(c(plot$start, plot$end)),
                                       max(c(plot$start, plot$end))),
         breaks = c(seq(-1000, max(plot$end, plot$start), by = 500)))
p <- p + theme(axis.title = element_blank(), text = element_text(size = 20),
               axis.text.x = element_text(angle = 315, hjust = 0))
p <- p + guides(colour = guide_legend(override.aes = list(size = 7)))
print(p)
```

'Nuff talk, let's take a look!