



10. The End

Ken Rice
Ting Ye

University of Washington

Seattle, July 2023

In this session

- Markdown (no slides)
- Shiny
- Other cool interactivity (by example)
- What next?
- Question time (optional)

Please [fill in the post-survey!](#)

Shiny

Markdown helps share finished documents. It's also possible to display data analyses on websites – and have them be interactive. The `shiny` package, by RStudio, builds 'apps' that do this.

The syntax is (roughly) a hybrid of R and HTML, so we give just a short example, showing off the `salary` data again*.

To make an app, in a directory named for your app, you need two files;

- **ui.R** This R script controls the layout and appearance of your app
- **server.R** This script contains the instructions that your computer needs to build your app

Before you start, make sure your R/RStudio is up-to-date, and install the `shiny` package.

*The [online tutorial](#) is excellent

Shiny: ui.R

```
library("shiny") # after installing it
shinyUI(fluidPage(
  # Application title
  titlePanel("Salary boxplots"),

  # Sidebar controlling which variable to plot against salary
  sidebarLayout(
    sidebarPanel(
      selectInput(inputId = "variable", label="Variable:",
                  choices = c("Rank" = "rank", "Year" = "year",
                              "Sex" = "gender", "Field"="field",
                              "Administrator"="admin")
                ),
      checkboxInput(inputId = "horizontal", label="Horizontal?", value=FALSE)
    ),
    # Show the caption and plot - defined in server.R
    mainPanel(
      h3(textOutput("caption")),
      plotOutput("salaryPlot")
    ) # close main Panel
  ) # close sidebarLayout
))
```

Shiny: server.R

```
library("shiny")
# first, a local copy of salary data sits in same directory
salary <- read.table("salaryShinyCopy.txt", header=TRUE)

# make some variable factors - for prettiness
salary$year <- factor(salary$year)
salary$admin <- factor(salary$admin)

# Define server "logic" required to plot salary vs various variables
shinyServer(function(input, output) {

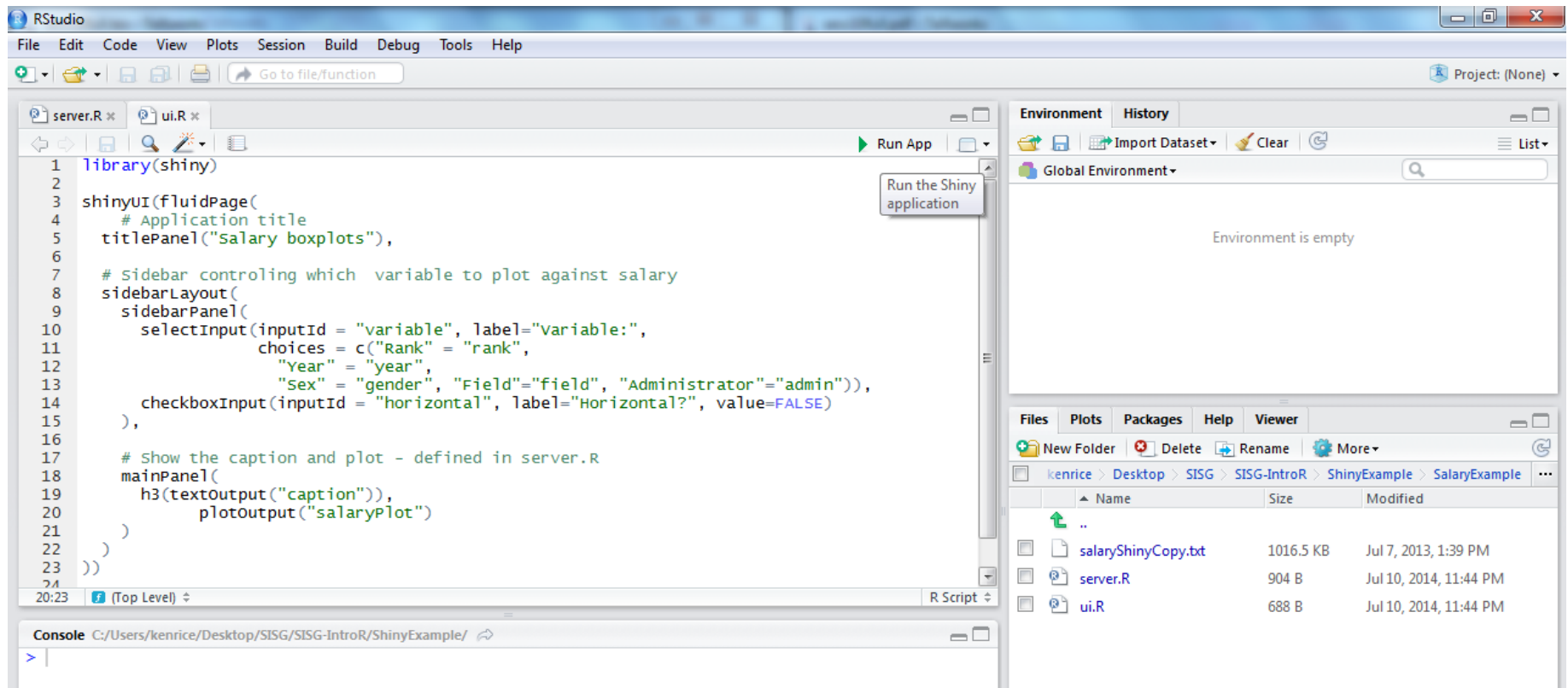
  # Compute the formula text in a "reactive expression"
  # it is shared by output$caption and output$mpgPlot, below
  formulaText <- reactive({ paste("salary ~", input$variable) })

  # Return the formula text for printing as a caption
  output$caption <- renderText({ formulaText() })

  # Do the boxplot, using the formula syntax, and setting horizontal=T/F
  output$salaryPlot <- renderPlot({
    boxplot(as.formula(formulaText()),
            data = salary, horizontal = input$horizontal) })
}) # close function
```

Shiny: making it work in Rstudio

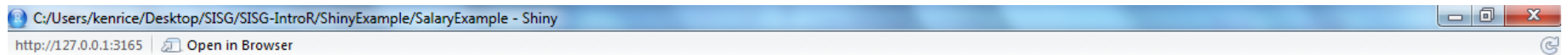
This is remarkably straightforward;



- Hit 'Run App' – and it (should) run
- Note that `ui.R`, `server.R` and the `salaryShinyCopy.txt` data file are *all* in the `SalaryExample` directory

Shiny: making it work in Rstudio

The (interactive) output should look something like this;

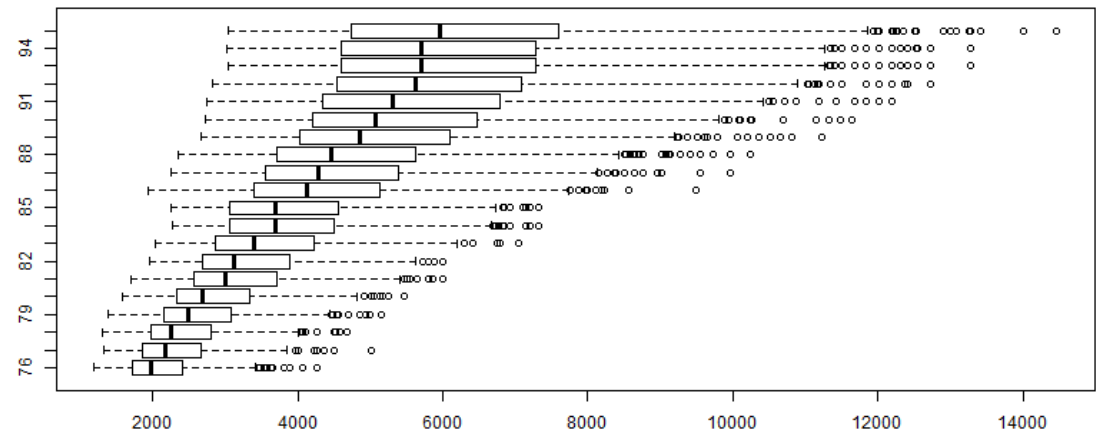


Salary boxplots

Variable:
Year

Horizontal?

salary ~ year



- Expect mild differences, across systems
- To share your app online, go to <https://www.shinyapps.io/> – needs registration, and other packages. [Online example]
- Be careful with personal data!

Interactivity

We've seen R access data files on websites, and download packages. It can do much more:

- Query online data repositories (e.g. [ENCODE explorer](#) – and others on [Bioconductor](#))
- Screen-scraping (e.g. [rvest](#))

But do be aware R isn't a good choice for *every* job;

- Some graphics formats, e.g. [animated GIFs](#), are much better supported by other software
- Calculations may require extremely fast evaluations – can be faster to have R run C code (get started with [inline](#) or [Rcpp](#))
- Calculations requiring different computational architecture, e.g. WinBUGS computes based on a graph system of nodes and edges, not well-suited to R

Here, we can run other software from R (e.g. [R2WinBUGS](#)) and return the output to our R session.

Interactivity

To download and then open files using your machine's default app:

```
library("curl") # install it first, if needed
curl_download("http://faculty.washington.edu/kenrice/rintro/type.gif", "t.gif")
system("open t.gif")
```

Assuming it knows what to do with URLs, on Windows also try

```
shell.exec("http://www.google.com/")
```

And having done that, try this last mammals example;

```
mammals <- read.table("http://faculty.washington.edu/kenrice/rintro/mammals.txt",
                    header=TRUE)
plot(log(brain)~log(body), data=mammals) # usual plot
repeat({
  mychoice <- identify(y=log(mammals$brain), x=log(mammals$body),
                      labels=mammals$species, n=1)
  shell.exec(
    paste("http://images.google.com/images?q=",
          mammals$species[mychoice], sep=""))
})
```

What next?

This concludes our course. To learn more;

- Take another one! Almost all modules use R extensively – practice your skills with applications you care about
- See the recommended books, on the course site
- There are several **R mailing lists**; R-help is the main one. *But* contributors expect you to have read the documentation – all of it! **CrossValidated** is friendlier to beginners
- Emailing package authors may also work
- For questions about *any* software, say;
 - What you did (ideally, with an example)
 - What you expected it to do
 - What it did instead
- Thanks! And please **fill in the post-survey!**