White Paper

Visualizations Get Some Snap from R Shiny

Todd Grabowsky
Data Scientist

April 2015

ELDER RESEARCH
DATA SCIENCE · AI · MACHINE LEARNING

Table of Contents

"Numbers have an important story to tell. They rely on you to give them a clear and convincing voice." –Stephen Few

Every data scientist bears the burden of clearly and convincingly communicating results to a client or decision maker. Putting analytic results in front of a client starts a dialogue that not only prepares them to understand and see the results, but may also uncover bugs in your logic, gaps in the data, or even misunderstandings about the problem or business goal. Data visualization is an essential way to explain results in an intuitive way. But what if there was an even better alternative?

The team that developed RStudio, an integrated development environment for the R language, has created an R package called Shiny, which takes data visualization to the next level. What is Shiny? It's a web application framework for R that enables rapid development of interactive web applications. That explanation though, is less helpful than seeing a real Shiny web application in action. Click here to see a live Shiny app in all its shiny glory.

The app displays a simple histogram plot, two checkboxes, and a drop down list that determines the bin size of the histogram. Notice that when you change the number of bins in the drop down list, the histogram plot updates automatically. This demonstrates the interactive and dynamic nature of a Shiny app. Also, notice the code at the bottom of the page. Web developers must juggle HTML, CSS, Javascript, and other programming languages to develop similar web applications, but this app was created with just a few dozen lines of R code!

Shiny seamlessly wraps your R code in the latest and greatest web technologies to create a user-friendly graphical user interface (GUI). Users interact with input widgets such as text boxes, sliders, and date range selectors, which dynamically update outputs such as values, tables, and plots. With Shiny, a client sees an intuitive visualization of your analytic results, and can also interact with the data on the fly.

What's so great about Shiny?

Of the countless use cases for Shiny, I would like to highlight three that show how Shiny can be used as a communication tool to foster meaningful interactions with clients, end users, and subject matter experts.

Communicating Insights/Model Deployment

The ability to dynamically interact with data through a Shiny app draws users in and, if executed correctly, helps accelerate the communication of key insights. If a picture is worth a thousand words, then an interactive Shiny app is worth a million.

The nuanced inner workings of a model can sometimes be opaque to the client or end consumer. This can lead to confusion and mistrust, which makes thoughtful model deployment a critical part of every data science project. This is where Shiny comes in. Shiny allows a data scientist to quickly transform a model from an abstract concept into a concrete application that an end user can see and manipulate. They can ask

questions of the data by simply dragging a slider or clicking a radio button, and then immediately get feedback rendered as a chart or data table. Shiny gives clients a window into your analysis and gives them an opportunity to validate your findings or possibly even uncover new insights.

Rapid Prototyping

Shiny is a great platform for rapid prototyping because app iterations can be developed and tested very quickly. In my experience as a data scientist and as a systems engineer, it is not uncommon to work on projects where the problem you are tasked to solve is vaguely defined. In these circumstances, an agile approach can be helpful. When the requirements for an analytic capability or model deployment platform are unclear, Shiny can be used as a rapid prototyping tool to quickly validate new features or capabilities.

On one of my current projects, Shiny is used in precisely this capacity. We deployed a production software tool to a team of analysts and are using Shiny as a "proving ground" for new analytic capabilities. We pilot new analytic capabilities as Shiny apps for the analysts, and if the capability helps them fulfill their mission, it is then rolled into the production software tool. Using Shiny apps to communicate our analysis helps us to quickly get feedback and identify new requirements. Sometimes the analysts are not even aware of certain requirements until they've seen it packaged into a Shiny app. In this way, building a prototype Shiny app and discussing it with a client can be an effective requirements elicitation technique.

Shiny apps also have myriad applications outside of data science. A friend of mine, an interactive designer at the New York Times, was interested in using Shiny as a platform for rapid prototyping or "sketching" for new visualizations before deploying them in a more polished form. The International Arctic Research Center at the University of Alaska Fairbanks uses Shiny for their production analysis tools which they have posted on their public-facing website.

Data Exploration/Visualization

Shiny allows a data scientist to build a kind of "exploration harness" for automating repetitive data exploration tasks. This is especially useful when exploring data visually. Shiny integrates with many of the most powerful data visualization packages available for R. This makes it easy to develop reusable data exploration apps to accelerate the process of understanding your data. If you uncover an aberrant finding during exploration, the app can then be used as a communication tool with subject matter experts to gain a better understanding of the aberration.

Twitter recently released the R code for their BreakoutDetection package, which detects change points in a time series. The core function within this package is called "breakout" and uses up to 9 input parameters. The impact of these parameters on the function's output (i.e., where the change points occur) is not immediately clear from the documentation, so I decided to create a Shiny app to quickly observe the impact of tweaking each input. I could have manually changed the input parameters in the R console and incrementally plotted the results, but the exploration was a lot faster when I could simply drag a slider or click a toggle and immediately see the effect on where the

change points occurred in time). Furthermore, I was later able to share the app with a colleague, which gave her an easy way to explore the breakout function for herself. You can see the app here and view the code on my github page.

Shiny Tips

I see Shiny as a great tool in a data scientist's toolbox. I'd like to share a few tips that will help accelerate learning of this tool.

Tip #1

You can create your first Shiny app in less than 30 seconds using RStudio. I also use this method when I just need a starting point for a new app. Here are the instructions:

- Open RStudio
- Go to File > New Project…
- Click "New Directory"
- Click "Shiny Web Application"
- Type a Directory Name, select a location for the app and click "Create Project"
- In the upper right corner of the "Source" pane, click "Run App"
- Voila! You just created your first Shiny app

Tip #2

Shiny integrates with many of the most powerful R data visualization packages. My "go-to" visualization packages are ggplot2 and rCharts. ggplot2 has nearly comprehensive plotting options, but only renders static plot images. rCharts is a collection of multiple javascript charting libraries that allow you to create interactive plots with features like mouseover tooltips and responsive legends. The rCharts plots look great and have nice interactive features (see examples here), but can have trouble rendering large amounts of data. If you notice that an rChart is rendering slowly (or not at all), then you may want to consider using ggplot2 instead. Here are links to four additional data visualization packages that integrate with Shiny:

- MetricsGraphics(for scatter plots, line plots, and histograms)
- dygraphs(for time series plots)
- networkD3(for network graphs)
- Leaflet(for geospatial maps)

Tip #3

Two nice UI features offered by Shiny are validation and progress indicators. The validation feature allows you to catch and throw user-friendly errors. Without this feature, any R errors that occur will display just as they would in the R console (not very user-friendly) and in bright red letters. Progress indicators allow you to give the user feedback about how far along a given computation is and how much of the computation remains. Without the progress indicators, a user will not know if the app froze, had an error, or is just churning on a complex computation. Both of these features can be implemented with Shiny using just a few lines of code.

Tip #4

If you need to build a dashboard that contains multiple displays and widgets, then you will want to check out RStudio's shinydashboard package. This package integrates seamlessly with Shiny and includes features for creating beautiful dashboard interfaces. You can read more at the shinydashboard website.

Tip #5

If you'd like to share your Shiny app online, then you can quickly deploy the app via shinyapps.io. You can host up to 5 apps with 25 active hours per month for free, and if you need to host more apps or have more active hours, then you can upgrade your account to one of the three paid monthly subscription services.

The documentation on the RStudio website is also very informative. If you are new to Shiny, then I highly recommend you take the Shiny tutorial to learn the core concepts of developing Shiny apps. For more advanced subjects, I recommend searching through the various Shiny articles to glean more technical insight.

Beware of Bikeshedding

A brief word of caution to future Shiny users. There will be an awful temptation as you develop your apps, especially amongst the more aesthetically-minded, to fall prey to what's known as bikeshedding, or Parkinson's Law of Triviality. This is when one spends much more time than is warranted or deserved on trivial features because they are easy to implement. Shiny gives you almost unlimited flexibility to customize the user interface (UI) for your web application. From custom layouts to color schemes to font choices, the sky is the limit for controlling the aesthetics of an app. This flexibility is useful for crafting a coherent user experience, but can also eat up hours of time that could otherwise be spent on actual data science. Shiny can and should be used to communicate insights about data and deploy models, but if you are a data scientist try to resist the temptation to perfect the "look and feel" of your app. Data scientists are usually hired for their ability to understand data and build models, which are rare and expensive skills; so if you find yourself spending an hour tweaking the style of a button or an afternoon messing with the colors of a plot, then you're probably bikeshedding and need to get back to the analytics! If you're playing with Shiny on your own time, however, then by all means make that UI as slick as possible. In fact, here are a few links to galleries of interesting and beautiful Shiny apps to inspire your efforts:

- http://shiny.rstudio.com/gallery/
- http://www.rstudio.com/products/shiny/shiny-user-showcase/
- http://www.showmeshiny.com/

Hopefully you're chomping at the bit to give Shiny a try. If so, then go ahead and download RStudio, install the Shiny R package (simply type *install.packages("shiny")* in the R console), and start experimenting (see Tip #1 above).

Presenting analytic results to a client in an intuitive way is essential for validating your analysis and ultimately providing value in the form of actionable insights. Static data

visualizations can get you part of the way there, but they lack the dynamic feedback and interactivity afforded by Shiny web apps. Shiny enables a client to manipulate the data and quickly see the outcomes of various “what-if” scenarios. This dynamic feedback can accelerate client understanding and create opportunities for useful dialogue to refine your analysis. Shiny fills the gap between spreadsheets/static visualizations and full-blown software applications as a means of effectively communicating analytic results. So, whether you’re explaining results to a team of fraud investigators or to the readers of your blog, consider using Shiny to give your data a clear and convincing voice.

About the Author

Elder Research Data Scientist Todd Grabowsky, enjoys making complex subject matter accessible and understandable for end users and decision makers. His technical interests include data visualization, anomaly detection, and predictive analytics. Prior to joining Elder Research, Todd worked as a systems engineer for clients in the intelligence community and other federal government agencies. He earned a B.S. in Systems Engineering from the University of Virginia.

www.elderresearch.com

ELDER RESEARCH

DATA SCIENCE · AI · MACHINE LEARNING