useR! 2020 Tutorials – Afternoon Session

How green was my valley - Spatial analytics with PostgreSQL, PostGIS, R, and PL/R

by J. Conway

This tutorial combines the use of PostgreSQL with PostGIS and R spatial packages. It covers data ingestion including cleaning and transformation, and basic analysis, of vector and raster based geospatial data from open sources.

Learning outcomes: At the end of the tutorial, the participants will:

- learn the advantages and disadvantages of using PostgreSQL, PostGIS, R, and PL/R to process geospatial data;
- perform a basic installation and setup of the technology stack, including PostgreSQL, PostGIS, R, and PL/R;
- gain experience on the data cleaning, subsetting, and ingestion process for that data;
- perform several simply types of visualization and analysis on the ingested data.

Requirements: The audience should be comfortable with R, SQL, and spatial data at an intermediate level. The attendees need only a modern OS/web browser. The hands-on exercises will be done via Katacoda.

Creating beautiful data visualization in R: a `ggplot2` crash course

by S. Tyner

Learning `ggplot2` brings joy and "aha moments" to new R users, keeping them more engaged and eager to grow their R skills. Newer R users will be and feel more empowered with data visualization skills. In addition to experiencing joy in creating beautiful graphics, advanced R users will learn to take advantage of `ggplot2`'s elegant defaults, saving time on manual plotting tasks like drawing legends. Thus, time and energy can be spent on advanced analyses, not fights with plotting commands.

Learning outcomes: Upon completion of this tutorial, the participants will be able to:

- identify the appropriate plot types and corresponding `ggplot2` *geom*s to consider when visualizing their data;
- implement the `ggplot2` grammar of graphics by using ggplot() and building up plots with the + operator;
- iterate through multiple visualizations of their data by changing the aesthetic mappings, geometries, and other graph properties;
- incorporate custom elements (colors, fonts, etc.) into their visualizations by adjusting `ggplot2` theme elements;
- investigate the world of `ggplot2` independently to expand upon the skills learned in the course.

Requirements: Users of any subject matter background who are interested in data visualization with R are welcome. New R users (less than 1 year of continual experience) will get the most out of this course. Some basic knowledge,such as understanding of different R data types & structures (character, data frame, etc.) and authoring simple functions & loops, is required. The course, however, is not just for beginners. More advanced R users who have little to no experience with `ggplot2` and other packages in the tidyverse will learn many new tools for data visualization in R. Participants should have a laptop with access to the internet. Having RStudio on the laptop is recommended, but if this is not possible the student may use the RStudio Cloud workspace. Students not using the RStudio Cloud should have the tidyverse suite of packages and

the `plotly` package installed. Or, for the minimalist student, `ggplot2`, `dplyr`, `tidyr`, and `plotly` will be sufficient for the bulk of the material. Following skills are expected:

- basic understanding of R data types and structures;
- ability to write simple functions and loops;
- willingness to learn new ways of doing things in R.

Building interactive web applications with Dash for R

by R. Kyle

The tutorial will appeal to those interested in building web applications, regardless of their skill level in R programming. Examples presented will not be overly technical, but will sample from common use cases for data analysis and visualization in R: report generation, model validation and assessment, interactive maps, financial analysis, as well as applications suited to bioinformatics and pharmaceutical research.

Learning outcomes: At the end of the tutorial, attendees can expect to develop competency in the following areas:

- a basic understanding the general architecture of Dash: the React-based component ecosystem, the stateless nature of functional callbacks, and why this helps Dash apps scale well, as well as the nature of the framework;
- proficiency in developing Dash applications and a functional awareness of components for the most commonly used interface elements (forms, buttons, sliders, gauges, etc.).

Requirements: The tutorial is intended for new users of Dash for R, who have at least a passing familiarity with functional programming in R. Prospective attendees need not be savvy web developers, as the framework's underlying goal is to make web technologies accessible to non-developers. Any users who are comfortable with R and also interested in building interactive web applications, regardless of whether they know JavaScript, should find the material meaningful, engaging, and relevant to their work.

To take full advantage of the tutorial, attendees will need to provide their own computer, upon which R 3.2.0 or greater is installed, along with Dash for R and the core suite of component library packages (Dash Table, Dash HTML Components, and Dash Core Components). The R packages used by the framework are not computationally intensive, but web development is generally more pleasant with at least 8 GB of RAM and a relatively modern computer.

Easy Larger-than-RAM data manipulation with `disk.frame`

by ZJ Dai

Every R user would have run into the "cannot allocate vector of size xxxB." error at some point. For most applications, R needs to load the data in its entirety into RAM. However, RAM is a precious resource and often do run out. `disk.frame` solves this issue by providing a framework to manipulate data on disk and minimize RAM usage. By using `disk.frame`, the user can perform analysis on much larger data than is normally possible with R. A modern laptop with `disk.frame` can comfortably handle 100GB's of data. In this tutorial, the user will learn how to use `disk.frame` effectively to manipulate datasets up to 100GBs in size on a single computer.

Learning outcomes: After the tutorial, the user should be able to:

- learn what the pros and cons of common "big data" technologies like DBMS, Spark, and `disk.frame` are;
- understand how `disk.frame` can help with manipulating large amounts of data;
- understand when to use `disk.frame` and when not to;
- confidently use `disk.frame` for any data manipulation task and not worry about running out of RAM;
- load large datasets into `disk.frame` format;
- manipulate large `disk.frame` files;

- summarize large `disk.frame` files;
- performing sorting and merging and other `disk.frame` related activities.

Requirements: The tutorial aims to introduce `disk.frame` to users with the needs to manipulate large amounts of data. The audience will benefit the most from this tutorial if they are already familiar with popular R data frameworks such as `dplyr` and `data.table` because `disk.frame` uses `dplyr` and `data.table` primitives like `select`, and `dt[i, j, by]`. Some users rely on DBMS (e.g. PostgresSQL), Spark, or SAS to manage their large dataset. They will find tremendous benefit in switching to `disk.frame`, which will allow them to keep their workflow in R for as long as possible.

The requirement for `disk.frame` is R 3.4 and the ability to install packages via `install.packages()`. A laptop is advised but not necessary.If attendees wish to follow along, I recommend that they should have RStudio or Jupyter notebook set up. However, that is not necessary as I will running through code that they can obtain from Github.

R Markdown recipes

by Y. Xie

This tutorial is based on my latest book, "R Markdown Cookbook", which is freely available at https://bookdown.org/yihui/rmarkdown-cookbook/. One highlight of this book is that it is written in a manner to answer the most frequently asked questions on Stack Overflow, so the content is highly practical and hopefully also very useful. The recipes in this tutorial should improve your efficiency of using R Markdown.

To check if you'd find this tutorial useful to you, you may watch my talk, "15 Tips on Making Better Use of R Markdown", (slides: https://bit.ly/dahshu-down). This tutorial at useR! will use a similar format.

Learning outcomes: Markdown is a fairly simple language, but it has a lot potential when combined with the power of Pandoc, R, and `knitr`. Basically we want attendees to be able to create sophisticated applications based on this simple language. The goals of this tutorial include:

- learn the basic idea of literate programming and apply it to data analysis using dynamic documents;
- learn how to use and customize existing output formats in `rmarkdown`;
- learn how to generate and format various document elements (figures, tables, and custom blocks, etc.);
- learn how to author a long-form document such as a book using `bookdown`;
- learn to build applications including journal papers, websites, slides, posters, resume, and so on;
- understand how HTML widgets and Shiny apps work in R Markdown;
- learn how to use other languages (besides R) in R Markdown.

Requirements: The attendees are only required to know the basics of R. However, to benefit as much as possible from the tutorial, the attendees should have some knowledge about HTML/CSS, JavaScript, LaTeX, and Word/PowerPoint. Before you come to the tutorial, please make sure you have the latest version of R installed. I will use the latest version of the RStudio IDE for demos, but you can use your own favorite editor. I recommend that you install these R packages in advance:

```
install.packages(c('rmarkdown', 'bookdown', 'blogdown', 'xaringan', 'pagedown', 'rticles',
'tufte', 'shiny', 'htmlwidgets'))
```

You can find instructions to install LaTeX at https://bookdown.org/yihui/rmarkdown/installation.html.

Getting the most out of Git

by C. Gillespie and R. Davies

Do you wonder what a pull request is? Do you browse GitHub and see lots of repositories with fancy looking badges, and think, I would like some of them? And what is a git-hook and why should you care? In this tutorial, we'll look at how we can leverage external tools, such as travis, covr, and Docker hub. We'll show you how to use these tools to make your code more robust and reproducible.

This tutorial will cover a variety of techniques that will increase the productivity of anyone using Git. The key idea is that a single commit can automatically launch a variety of other services. This can lead to massive savings in time, as well are reduce bugs.

Learning outcomes: An overview of the topics covered are:

- continuous integration with `travis` and `appveyor`;
- automatically creating Docker images via DockerHub;
- generating static HTML documentation with `pkgdown` & `bookdown`;
- organisations vs users;
- using client and server-side git-hooks;
- code coverage with codecov.io;
- Github vs Gitlab.

Requirements: The intended audience are intermediate R users. They will have experience in writing functions and basic package development. Participants should also be familiar with simple git commands, such as pull, push and commit. Minimal computing configuration:

- RStudio;
- A few well-known R packages, e.g. `pkgdown`, `knitr`;
- A GitHub username and password.

We'll also run RStudio in the Cloud in case software hasn't been installed.

End-to-end machine learning with `Metaflow`: Going from prototype to production with Netflix's open source project for reproducible data science

by S. Goyal and B. Galvin

`Metaflow` is a human-friendly R package that helps scientists and engineers build and manage real-life data science projects. `Metaflow` was originally developed at Netflix to boost the productivity of data scientists who work on a wide variety of projects from classical statistics to state-of-the-art deep learning. You can use `Metaflow` with any of your favorite machine learning or data science libraries. In this tutorial, you will learn how by using `Metaflow`, you can write your models and business logic as idiomatic R code and not worry about any infrastructural concerns.

Models are only a small part of an end-to-end data science project. Production-grade projects rely on a thick stack of infrastructure. At a minimum, projects need data and a way to perform computation on it. Data is accessed from a data warehouse, which can be a folder of files or a multi-petabyte data lake. The modeling code that crunches the data is executed in compute environments which can range from a laptop to large-scale container management system.

How do you architect the code to be executed? How do you version the code, input data, and models produced? After the model has been deployed to production, how do you monitor its performance? How do you deploy new versions of your code to run in parallel with the previous version? The software industry has spent over a decade perfecting DevOps best practices for normal software. We are just getting started with data science. `Metaflow` allows you to write your models and business logic as idiomatic R code with not much new to learn.

Learning outcomes: In this tutorial, participants will be able to write, iterate upon and deploy their own production-ready models using packages they all know and love, such as `caret` and `mlr`.

Requirements: Participants are expected to have some experience building machine learning models or performing data engineering with R. Technical concepts will be explained in layman's terms, but experience with using the command line and/or Linux will be helpful. No knowledge of the cloud required. We require our users to bring an internet-connected laptop (UNIX or Linux) with rstudio installed (or terminal + text editor). We would be providing sandbox accounts on AWS to all the attendees (metaflow.org/sandbox) to run their workloads on the cloud for the duration of the tutorial.

Package development

by J. Hester and H. Wickham

Packages are the fundamental units of reproducible R code. They include reusable R functions, the documentation that describes how to use them, and sample data. One of the strength of the R community is the large and varied packages available. Being able to create your own packages allows your work to be used by many more people than it could otherwise and can provide a major benefit to the community as a whole as well as your own personal collaborations.

The key to well-documented, well-tested and easily-distributed R code is the package. This half day class will teach you how to make package development as easy as possible with devtools and usethis. You'll also learn `roxygen2` to document your code so others (including future you!) can understand what's going on, and testthat to avoid future breakages.

Learning outcomes: This class will be a good fit for you if you've developed a number of R scripts, and now you want to learn:

- a more efficient workflow, iterating between writing code and checking that it works;
- how to document your code so others (including future you!) can understand what's going on;
- automated testing principles, so that if you accidentally break something in your code you find out right away;
- how to package and distribute your code to others, whether it's inside your research group, your company, or to the whole world.

Requirements: Participants should bring a laptop setup to build R packages. Detailed instructions are available here.

What they forgot to teach you about teaching R

by M. Cetinkaya-Rundel

What makes this tutorial appealing to audiences from a broad range of backgrounds is useRs from a broad range of backgrounds are teaching R. The topics covered, and the tooling introduced are applicable in a broad range of courses, from a semester-long introductory statistics course that introduces R as tool for simple data analysis to a two-day advanced Shiny course that focuses on programming with R. The goal of the course is to introduce educators to pedagogical considerations for effective, welcoming, and inclusive teaching of R and the tooling that reduces the prep and pain associated with implementing these ideas.

Learning outcomes: By the end of this workshop, participants will:

- master using RStudio Cloud for teaching;
- effectively use GitHub as their "learning management system" for assignment submission, providing feedback, collaboration, automated feedback, and peer review;
- live code with pedagogical best practices in mind;
- learn about accessibility considerations in learning materials built with the *down packages;
- build interactive tutorials and leverage data collected from students via these tutorials to help student learning.

Requirements: This workshop is aimed at participants who teach R at any capacity (from semester-long academic courses to short workshops), in any medium (in person or online), at any stage of their teaching journey (just starting off or redesigning a well-established course/curriculum).

Each participant is expected to have a WiFi enabled laptop. If the participants would like to use their local setup for R, it is recommended that they have latest R and RStudio installed as well as git. Alternatively, an RStudio Cloud instance will be provided for the exercises, however we anticipate that educators might prefer to use their local setup. Detailed instructions for setup and packages will be provided prior to the workshop.