18: Data(base) Management

Environmental Data Analytics | Kateri Salk

Spring 2019

LESSON OBJECTIVES

1. Discuss data challenges in the environmental field
2. Evaluate how data management fits into the pipeline of data analysis
3. Acquire data through database searches, R packages, and webpage scraping

CONTEMPORARY DATA CHALLENGES

Environmental fields are experiencing massive changes related to data. Many new challenges exist today, including:

- Big data: volume, variety, frequency
- Open data
- Long-term records
- Interconnected networks
- Verifying accuracy and integrity

DATA MANAGEMENT

Adapted from the DataOne data management primer and the University of Alabama Library guide on data management

1. Choose and assemble data management toolbox
2. Create (and revisit) your data management plan

- Volume and type of data
- File and folder structures/formats
- Roles and responsibilities of personnel
- Version control
- Access
- Preservation

3. Collect data
4. Quality assurance/quality control (QA/QC)
5. Describe and document data
6. Store and preserve data in a repository

ACCESSING DATA

Database searches

Various disciplines

re3data

DataOne

Google Dataset Search

EDI Data Portal

NEON

LTER

Water

CUAHSI HydroClient

CUAHSI HydroShare

Spatial Data

ArcGIS

Search one or more of these databases to find a dataset that interests you. What did you find?

> ANSWER:

R Packages

- NHANES: National Health and Nutrition Examination Survey
- TidyCensus: U.S. Census data
- FedData: Geospatial data from federal sources
- dataRetrieval: USGS and EPA water quality, streamflow, and metadata
- LAGOSNE: Multiscaled geospatial and temporal data for U.S. lakes

Data scraping

Sometimes, there may be data that we can access online but are not available in downloadable formats. Data scraping is a technique that allows us to convert unstructured data (e.g., those presented in a webpage) into a structured format (e.g., a csv file).

Methods for scraping data (from Analytics Vidhya data scraping guide):

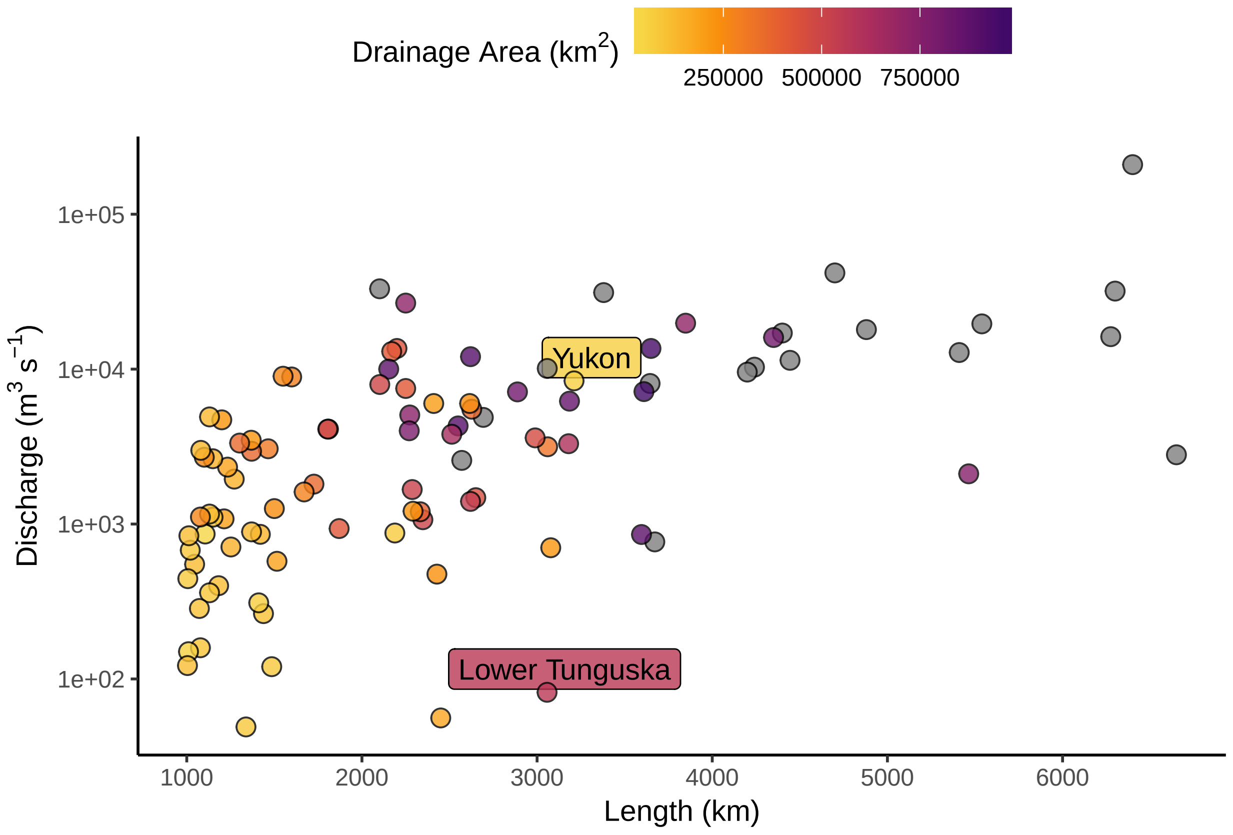
- Manual copy-paste
- API (retrieve data from standard code; data must be in prescribed format)
- DOM parsing

We will be scraping today using DOM parsing. First, we need to install a tool on our web browser to be able to call the web text we need. The tool is called a Selector Gadget, which for Chrome can be found here.

Now that our selector gadget is in operation, we can start the data scraping process with the `rvest` package.

```r
library(tidyverse)
#install.packages("rvest")
library(rvest)

# Specify website to be scraped
url <- "https://en.wikipedia.org/wiki/List_of_rivers_by_length"

# Reading the HTML code from the website
webpage <- read_html(url)

# Grab specific components of the website
Name = webpage %>% html_nodes("td:nth-child(2)") %>% html_text()
Length.km = webpage %>% html_nodes("td:nth-child(3)") %>% html_text()
```

```r
DrainageArea.km2 = webpage %>% html_nodes("td:nth-child(5)") %>% html_text()
Discharge.m3s = webpage %>% html_nodes("td:nth-child(6)") %>% html_text()
Outflow = webpage %>% html_nodes("td:nth-child(7)") %>% html_text()

# Coerce into a data frame, ensure consistent lengths
riverdata <- data_frame(Name = Name[7:189],
                        Length.km = Length.km[2:184],
                        DrainageArea.km2 = DrainageArea.km2[2:184],
                        Discharge.m3s = Discharge.m3s[2:184],
                        Outflow = Outflow)

# Remove unnecessary text from within cells
riverdata$Name <- str_replace(riverdata$Name, "\\[.*\\]", "")
riverdata$Name <- str_replace(riverdata$Name, "\n", "")


riverdata$Length.km <- str_replace(riverdata$Length.km, "\\*", "")
riverdata$Length.km <- str_replace(riverdata$Length.km, "\\(.*\\)", "")
riverdata$Length.km <- str_replace(riverdata$Length.km, "\\[.*\\]", "")

riverdata$DrainageArea.km2 <- str_replace(riverdata$DrainageArea.km2, " \\(.*\\)", "")
riverdata$DrainageArea.km2 <- str_replace(riverdata$DrainageArea.km2, "\\(.*\\)", "")
riverdata$DrainageArea.km2 <- str_replace(riverdata$DrainageArea.km2, " \\[.*\\]", "")

riverdata$Discharge.m3s <- str_replace(riverdata$Discharge.m3s, " \\(.*\\)", "")
riverdata$Discharge.m3s <- str_replace(riverdata$Discharge.m3s, "\\[.*\\]", "")

# Turn numeric values from character to numeric
riverdata$Length.km <- as.numeric( sub(",", "", riverdata$Length.km))
riverdata$DrainageArea.km2 <- as.numeric( sub(",", "", riverdata$DrainageArea.km2))
riverdata$Discharge.m3s <- as.numeric( sub(",", "", riverdata$Discharge.m3s))

# Plot the relationships
ggplot(riverdata, aes(x = Length.km, y = Discharge.m3s, fill = DrainageArea.km2)) +
  geom_label(data = subset(riverdata, Name == "Yukon" | Name == "Lower Tunguska"),
             aes(label = Name), alpha = 0.8, nudge_x = 100, nudge_y = 0.15) +
  geom_point(shape = 21, size = 3, alpha = 0.8) +
  scale_fill_viridis_c(option = "inferno", begin = 0.2, end = 0.9, direction = -1) +
  theme_classic() +
  scale_y_log10() +
  labs(x = "Length (km)", y = expression("Discharge (m"^"3"*" s"^"-1"*")"),
       fill = expression("Drainage Area (km"^"2"*")")) +
  theme(legend.position = "top", legend.key.width = unit(1, "cm"))
```



Drainage Area (km^2)
250000
500000
750000
Discharge (m^3 s^{-1})
1e+05
1e+04
1e+03
1e+02
Yukon
Lower Tunguska
1000
2000
3000
4000
5000
6000
Length (km)