Lecture 9, MTH 400, Fall 2024

Data Formats, Acquiring Data, APIs

R provides several methods and packages to import various file types into your R environment. Here are some commonly used methods and packages:

1. readr: The readr package, part of the tidyverse, offers fast and efficient functions for reading structured text files. It supports file formats like CSV, TSV, and fixed-width files. The package includes functions such as read_csv(), read_tsv(), and read_delim().
2. readxl: The readxl package allows you to import data from Excel files (.xls and .xlsx) into R. It provides functions like read_excel() to read specific sheets or ranges within Excel files. It can handle large Excel files and preserve formatting, formulas, and other Excel features.
3. foreign: The foreign package provides functions to read data files used by other statistical software packages. It supports file formats like SPSS, SAS, and Stata. Functions such as read.spss(), read.sas7bdat(), and read.dta() can be used to import data from these formats.
4. haven: The haven package is designed for reading and writing data in SAS, SPSS, and Stata file formats. It provides functions like read_sas(), read_spss(), and read_dta() to import data from these formats into R.
5. RODBC: The RODBC package allows you to connect to relational databases using the ODBC (Open Database Connectivity) interface. It enables you to import data from databases such as MySQL, PostgreSQL, Oracle, and Microsoft SQL Server. Functions like odbcConnect() and sqlFetch() can be used to establish a connection and retrieve data.
6. jsonlite: The jsonlite package provides functions to work with JSON (JavaScript Object Notation) data. It allows you to import JSON data into R using functions like fromJSON() or stream_in(). It can handle both simple and complex JSON structures.
7. feather: The feather package enables efficient reading and writing of data frames between R and Python using the feather file format. It offers fast data import/export capabilities and is particularly useful when working with large datasets.

These are just a few examples of methods and packages available in R for importing different file types. The choice of package or method depends on the specific file format you are working with, and it's always recommended to refer to the documentation and examples provided by each package for detailed usage instructions.

In R, you can interact with APIs (Application Programming Interfaces) to retrieve data, send requests, and receive responses from web services. Here's a general process for using APIs in R:

1. Install and load necessary packages: Depending on the API and the type of requests you need to make, you may need to install and load specific packages. Some commonly used packages for working with APIs in R include httr, jsonlite, httr, and curl.
2. Obtain API key or authentication credentials: Many APIs require authentication in the form of an API key or access token. You'll typically need to register for an account with the service providing the API and obtain the necessary credentials.
3. Make API requests: Use the appropriate functions from the package you're using to make API requests. The httr package is commonly used for this purpose. Functions like GET(), POST(), PUT(), and DELETE() allow you to send HTTP requests to the API endpoints.
4. Handle the response: Once you've made the API request, you'll receive a response from the server. The response is usually in JSON format, which you can parse using the jsonlite package. The response may contain the data you requested or other relevant information, such as status codes or error messages.

5. Process and analyze the data: After receiving the API response, you can process and analyze the data using R's data manipulation and analysis capabilities. You may need to transform the data into a suitable format for your analysis, such as a data frame or other data structure.

Here's a simplified example that demonstrates how to make a GET request to an API using the httr package and process the response:

```
library(httr)
library(jsonlite)

# Set the API endpoint and parameters
url <- "https://api.example.com/data"
params <- list(api_key = "your_api_key")

# Make the API request
response <- GET(url, query = params)

# Check the status code
if (status_code(response) == 200) {
  # Parse the JSON response
  json <- content(response, "text")  # or "parsed" to parse as R objects

  # Process and analyze the data
  # ...
} else {
  # Handle error response
  # ...
}
```

Note that the specific details of working with an API will depend on the service you're accessing and its documentation. You'll need to refer to the API documentation to understand the required parameters, request formats, authentication methods, and available endpoints.

Additionally, some APIs may provide R packages specifically designed for their service, which can simplify the process of interacting with the API. It's worth checking if there are any dedicated packages available for the API you are working with.

Accessing Census data through an API in R involves using specific packages designed to interact with the U.S. Census Bureau's APIs. One of the most commonly used packages for this purpose is tidycensus. Below are the steps to get started with accessing and analyzing Census data through the API in R:

Step-by-Step Guide to Accessing Census Data Using tidycensus

1. Install and Load Required Packages

First, you need to install and load the tidycensus and tidyverse packages:

```
install.packages("tidycensus")
install.packages("tidyverse")  # For data manipulation and visualization
library(tidycensus)
library(tidyverse)
```

Obtain a Census API Key

To access the Census data, you need an API key from the U.S. Census Bureau:

- Go to the U.S. Census Bureau's Request for API Key page. https://api.census.gov/data/key_signup.html
- Fill out the form to receive your API key via email.

Once you have the key, you can set it up in your R session:

```
census_api_key("YOUR_CENSUS_API_KEY", install = TRUE)
```

This will store the API key in your R environment for future use. If you prefer to set it up for each session, you can omit the install = TRUE part.

3. Load Census Variables
To know which variables you can query, you might want to list available variables for a specific dataset and year. For example, to list variables from the 2019 American Community Survey (ACS):

```
v19 <- load_variables(2019, "acs5", cache = TRUE)
view(v19)
```

Retrieve Census Data
You can now use the get_acs function to retrieve data. Here's an example of how to get population data by state:

```
acs_data <- get_acs(geography = "state",
            variables = "B01003_001",
            year = 2019,
            survey = "acs5")

# View the data
head(acs_data)
```

In this example:

- geography = "state" specifies the level of geography (e.g., state, county, tract).
- variables = "B01003_001" specifies the variable (total population in this case).
- year = 2019 specifies the year of the data.
- survey = "acs5" specifies the survey type (5-year ACS data).

Analyzing and Visualizing Census Data
You can use tidyverse functions to manipulate and visualize the data. For example, creating a bar plot of population by state:

```
acs_data %>%
  mutate(NAME = str_replace(NAME, " \\(State\\)", "")) %>%
  ggplot(aes(x = reorder(NAME, estimate), y = estimate)) +
  geom_col() +
  coord_flip() +
  labs(title = "Population by State",
       x = "State",
     y = "Population") +
  theme_minimal()
```

Example: Detailed Workflow
Here is a more detailed example, including data retrieval, manipulation, and visualization:

```
# Load libraries
library(tidycensus)
library(tidyverse)

# Set Census API key
census_api_key("YOUR_CENSUS_API_KEY", install = TRUE)

# Retrieve median household income data by county
income_data <- get_acs(geography = "county",
                       variables = "B19013_001",
                       year = 2019,
                       survey = "acs5",
                       state = "TX")  # Example: Texas

# View the data
head(income_data)

# Clean and visualize the data
income_data %>%
  mutate(NAME = str_replace(NAME, " County, Texas", "")) %>%
  ggplot(aes(x = reorder(NAME, estimate), y = estimate)) +
  geom_col() +
  coord_flip() +
  labs(title = "Median Household Income by County in Texas",
     x = "County",
     y = "Median Household Income (USD)") +
  theme_minimal()
```

Using the tidycensus package in R provides a powerful and flexible way to access and analyze Census data via the U.S. Census Bureau's API. By following these steps, you can efficiently retrieve, manipulate, and visualize Census data to gain valuable insights for your analyses.

There are numerous online resources where you can acquire data for analysis in R. These datasets span a wide range of domains, including government, finance, health, social sciences, and more. Here are some notable sources:

Government and Public Sector Data

- U.S. Census Bureau: Offers extensive demographic, economic, and geographic data.
  data.census.gov
- American Community Survey (ACS)
  Data.gov: A comprehensive portal for U.S. government data across various sectors.
  data.gov

- European Union Open Data Portal: Access data published by EU institutions and bodies.
  data.europa.eu
- World Bank Open Data: Provides free and open access to global development data.
  data.worldbank.org
- United Nations Data: Offers a wide range of economic, social, financial, and development data.
  data.un.org

Health Data
- Centers for Disease Control and Prevention (CDC): Offers data on various health topics.
  data.cdc.gov
- World Health Organization (WHO): Provides global health statistics and information.
  who.int/data
- National Institutes of Health (NIH): Offers various datasets related to biomedical research.
  nlm.nih.gov/databases

Financial and Economic Data
- Yahoo Finance: Provides stock prices, financial news, and historical data.
  finance.yahoo.com
- Quandl: A wide range of financial, economic, and alternative data.
  quandl.com
- Federal Reserve Economic Data (FRED): Economic data provided by the Federal Reserve Bank of St. Louis.
  fred.stlouisfed.org

Social Science Data
- Pew Research Center: Provides data on social issues, public opinion, and demographic trends.
  pewresearch.org/data
- ICPSR (Inter-university Consortium for Political and Social Research): Provides access to an extensive archive of social science data.
  icpsr.umich.edu
- General Social Survey (GSS): Contains data on demographic characteristics and attitudes of the U.S. population.
  gss.norc.org

Environmental and Geographic Data
- U.S. Geological Survey (USGS): Offers data on natural resources, natural hazards, and the environment.
  usgs.gov/data
- NASA Earth Data: Provides satellite imagery and data related to Earth's climate, environment, and resources.
  earthdata.nasa.gov
- Global Biodiversity Information Facility (GBIF): Provides access to data about all types of life on Earth.
  gbif.org

Miscellaneous Data Repositories
- Kaggle: A platform for data science competitions that offers a vast repository of datasets.

  kaggle.com/datasets
- Google Dataset Search: A tool to find datasets stored across the web.
  datasetsearch.research.google.com
- FiveThirtyEight: Offers datasets used in the analyses featured on the FiveThirtyEight website.
  data.fivethirtyeight.com
- Open Data on AWS: Amazon Web Services hosts a variety of open datasets on their platform.
  registry.opendata.aws

Academic and Research Data
- KDNuggets Datasets: A curated list of datasets for data science and machine learning.
  kdnuggets.com/datasets/index.html
- UCI Machine Learning Repository: A collection of databases, domain theories, and datasets for machine learning research.
  archive.ics.uci.edu/ml/index.php

Using APIs to Access Data
Many of these sources provide APIs for programmatic access. Here are a few examples of R packages to interact with these APIs:
- tidycensus: For accessing U.S. Census data.

```
install.packages("tidycensus")
library(tidycensus)
```

- quantmod: For financial data from sources like Yahoo Finance.

```
install.packages("quantmod")
library(quantmod)
```

- rworldmap: For mapping and analyzing global data.

```
install.packages("rworldmap")
library(rworldmap)
```

- httr and jsonlite: General-purpose packages for working with web APIs.

```
install.packages("httr")
install.packages("jsonlite")
library(httr)
library(jsonlite)
```

By leveraging these resources and tools, you can find and analyze a wide variety of datasets to suit your research and analytical needs.

Resources:
1. https://www.computerworld.com/article/2497164/business-intelligence-beginner-s-guide-to-r-get-your-data-into-r.html

2. https://www.infoworld.com/article/3434627/get-api-data-with-r.html
3. https://www.datafiles.samhsa.gov/get-help/format-specific-issues/how-do-i-read-data-r
4. https://www.datacamp.com/tutorial/r-data-import-tutorial
5. https://www.geeksforgeeks.org/reading-files-in-r-programming/
6. https://www.datacamp.com/tutorial/importing-data-r-part-two
7. https://www.geeksforgeeks.org/how-to-import-data-from-a-file-in-r-programming/
8. https://www.dataquest.io/blog/r-api-tutorial/
9. https://info201.github.io/apis.html
10. https://www.geeksforgeeks.org/access-collect-data-with-apis-in-r/
11. http://lab.rady.ucsd.edu/sawtooth/business_analytics_in_r/DataApi.html