Day 06: APIs

Erin Rossiter

27 February, 2023

Announcements

Announcements

- PS04 graded
- PS05 due tomorrow
  - do not do Question 1 in Rmd
- You all can see each others' code, right?

Announcements

- PS04 graded
- PS05 due tomorrow
  - do not do Question 1 in Rmd
- You all can see each others' code, right?

Announcements

- PS04 graded
- PS05 due tomorrow
  - do not do Question 1 in Rmd
- You all can see each others' code, right?

Announcements

- PS04 graded
- PS05 due tomorrow
  - do not do Question 1 in Rmd
- You all can see each others' code, right?

Today's plan

Today's plan

- Last two sessions
  - basics of websites/html
  - rvest
  - conditions & condition handling
  - extracting data from more complex websites and JSON
  - selenium demo
- Today
  - legality of all this
  - basics of http requests
  - APIs
- Next time
  - Inclass midterm

Today's plan

- Last two sessions
  - basics of websites/html
  - rvest
  - conditions & condition handling
  - extracting data from more complex websites and JSON
  - selenium demo
- Today
  - legality of all this
  - basics of http requests
  - APIs
- Next time
  - Inclass midterm

Today's plan

- Last two sessions
  - basics of websites/html
  - rvest
  - conditions & condition handling
  - extracting data from more complex websites and JSON
  - selenium demo
- Today
  - legality of all this
  - basics of http requests
  - APIs
- Next time
  - Inclass midterm

Today's plan

- Last two sessions
  - basics of websites/html
  - rvest
  - conditions & condition handling
  - extracting data from more complex websites and JSON
  - selenium demo
- Today
  - legality of all this
  - basics of http requests
  - APIs
- Next time
  - Inclass midterm

Today's plan

- Last two sessions
  - basics of websites/html
  - rvest
  - conditions & condition handling
  - extracting data from more complex websites and JSON
  - selenium demo
- Today
  - legality of all this
  - basics of http requests
  - APIs
- Next time
  - Inclass midterm

Today's plan

- Last two sessions
  - basics of websites/html
  - rvest
  - conditions & condition handling
  - extracting data from more complex websites and JSON
  - selenium demo
- Today
  - legality of all this
  - basics of http requests
  - APIs
- Next time
  - Inclass midterm

Today's plan

- Last two sessions
  - basics of websites/html
  - rvest
  - conditions & condition handling
  - extracting data from more complex websites and JSON
  - selenium demo
- Today
  - legality of all this
  - basics of http requests
  - APIs
- Next time
  - Inclass midterm

Today's plan

- Last two sessions
  - basics of websites/html
  - rvest
  - conditions & condition handling
  - extracting data from more complex websites and JSON
  - selenium demo
- Today
  - legality of all this
  - basics of http requests
  - APIs
- Next time
  - Inclass midterm

Today's plan

- Last two sessions
  - basics of websites/html
  - rvest
  - conditions & condition handling
  - extracting data from more complex websites and JSON
  - selenium demo
- Today
  - legality of all this
  - basics of http requests
  - APIs
- Next time
  - Inclass midterm

Today's plan

- Last two sessions
  - basics of websites/html
  - rvest
  - conditions & condition handling
  - extracting data from more complex websites and JSON
  - selenium demo
- Today
  - legality of all this
  - basics of http requests
  - APIs
- Next time
  - Inclass midterm

Today's plan

- Last two sessions
  - basics of websites/html
  - rvest
  - conditions & condition handling
  - extracting data from more complex websites and JSON
  - selenium demo
- Today
  - legality of all this
  - basics of http requests
  - APIs
- Next time
  - Inclass midterm

Today's plan

- Last two sessions
  - basics of websites/html
  - rvest
  - conditions & condition handling
  - extracting data from more complex websites and JSON
  - selenium demo
- Today
  - legality of all this
  - basics of http requests
  - APIs
- Next time
  - Inclass midterm

Midterm

- 45 minutes, then 30 minute break
  - Resume at 4:45 with answers then data cleaning
- **Basics** (first three class sessions)
  - data classes
  - data structures
  - if/else and logicals
  - basic loops
  - basic functions
- I don't imagine you'll need to "study" unless you feel hazy on these things. If so, this is my way to incentivize you to get caught up
  - Review class scripts
  - Review my problem set solutions
- No ChatGPT or messaging each other
- All other resources are fine to use
- Distributed via GitHub
- Submit however you prefer so not stressful
  - Github
  - Email
  - even Slack

Midterm

- 45 minutes, then 30 minute break
  - Resume at 4:45 with answers then data cleaning
- **Basics** (first three class sessions)
  - data classes
  - data structures
  - if/else and logicals
  - basic loops
  - basic functions
- I don't imagine you'll need to "study" unless you feel hazy on these things. If so, this is my way to incentivize you to get caught up
  - Review class scripts
  - Review my problem set solutions
- No ChatGPT or messaging each other
- All other resources are fine to use
- Distributed via GitHub
- Submit however you prefer so not stressful
  - Github
  - Email
  - even Slack

Midterm

- 45 minutes, then 30 minute break
  - Resume at 4:45 with answers then data cleaning
- **Basics** (first three class sessions)
  - data classes
  - data structures
  - if/else and logicals
  - basic loops
  - basic functions
- I don't imagine you'll need to "study" unless you feel hazy on these things. If so, this is my way to incentivize you to get caught up
  - Review class scripts
  - Review my problem set solutions
- No ChatGPT or messaging each other
- All other resources are fine to use
- Distributed via GitHub
- Submit however you prefer so not stressful
  - Github
  - Email
  - even Slack

Midterm

- 45 minutes, then 30 minute break
  - Resume at 4:45 with answers then data cleaning
- **Basics** (first three class sessions)
  - data classes
  - data structures
  - if/else and logicals
  - basic loops
  - basic functions
- I don't imagine you'll need to "study" unless you feel hazy on these things. If so, this is my way to incentivize you to get caught up
  - Review class scripts
  - Review my problem set solutions
- No ChatGPT or messaging each other
- All other resources are fine to use
- Distributed via GitHub
- Submit however you prefer so not stressful
  - Github
  - Email
  - even Slack

Midterm

- 45 minutes, then 30 minute break
  - Resume at 4:45 with answers then data cleaning
- **Basics** (first three class sessions)
  - data classes
  - data structures
  - if/else and logicals
  - basic loops
  - basic functions
- I don't imagine you'll need to "study" unless you feel hazy on these things. If so, this is my way to incentivize you to get caught up
  - Review class scripts
  - Review my problem set solutions
- No ChatGPT or messaging each other
- All other resources are fine to use
- Distributed via GitHub
- Submit however you prefer so not stressful
  - Github
  - Email
  - even Slack

Midterm

- 45 minutes, then 30 minute break
  - Resume at 4:45 with answers then data cleaning
- **Basics** (first three class sessions)
  - data classes
  - data structures
  - if/else and logicals
  - basic loops
  - basic functions
- I don't imagine you'll need to "study" unless you feel hazy on these things. If so, this is my way to incentivize you to get caught up
  - Review class scripts
  - Review my problem set solutions
- No ChatGPT or messaging each other
- All other resources are fine to use
- Distributed via GitHub
- Submit however you prefer so not stressful
  - Github
  - Email
  - even Slack

Midterm

- 45 minutes, then 30 minute break
  - Resume at 4:45 with answers then data cleaning
- **Basics** (first three class sessions)
  - data classes
  - data structures
  - if/else and logicals
  - basic loops
  - basic functions
- I don't imagine you'll need to "study" unless you feel hazy on these things. If so, this is my way to incentivize you to get caught up
  - Review class scripts
  - Review my problem set solutions
- No ChatGPT or messaging each other
- All other resources are fine to use
- Distributed via GitHub
- Submit however you prefer so not stressful
  - Github
  - Email
  - even Slack

Midterm

- 45 minutes, then 30 minute break
  - Resume at 4:45 with answers then data cleaning
- **Basics** (first three class sessions)
  - data classes
  - data structures
  - if/else and logicals
  - basic loops
  - basic functions
- I don't imagine you'll need to "study" unless you feel hazy on these things. If so, this is my way to incentivize you to get caught up
  - Review class scripts
  - Review my problem set solutions
- No ChatGPT or messaging each other
- All other resources are fine to use
- Distributed via GitHub
- Submit however you prefer so not stressful
  - Github
  - Email
  - even Slack

Midterm

- 45 minutes, then 30 minute break
  - Resume at 4:45 with answers then data cleaning
- **Basics** (first three class sessions)
  - data classes
  - data structures
  - if/else and logicals
  - basic loops
  - basic functions
- I don't imagine you'll need to "study" unless you feel hazy on these things. If so, this is my way to incentivize you to get caught up
  - Review class scripts
  - Review my problem set solutions
- No ChatGPT or messaging each other
- All other resources are fine to use
- Distributed via GitHub
- Submit however you prefer so not stressful
  - Github
  - Email
  - even Slack

Midterm

- 45 minutes, then 30 minute break
  - Resume at 4:45 with answers then data cleaning
- **Basics** (first three class sessions)
  - data classes
  - data structures
  - if/else and logicals
  - basic loops
  - basic functions
- I don't imagine you'll need to "study" unless you feel hazy on these things. If so, this is my way to incentivize you to get caught up
  - Review class scripts
  - Review my problem set solutions
- No ChatGPT or messaging each other
- All other resources are fine to use
- Distributed via GitHub
- Submit however you prefer so not stressful
  - Github
  - Email
  - even Slack

Midterm

- 45 minutes, then 30 minute break
  - Resume at 4:45 with answers then data cleaning
- **Basics** (first three class sessions)
  - data classes
  - data structures
  - if/else and logicals
  - basic loops
  - basic functions
- I don't imagine you'll need to "study" unless you feel hazy on these things. If so, this is my way to incentivize you to get caught up
  - Review class scripts
  - Review my problem set solutions
- No ChatGPT or messaging each other
- All other resources are fine to use
- Distributed via GitHub
- Submit however you prefer so not stressful
  - Github
  - Email
  - even Slack

Midterm

- 45 minutes, then 30 minute break
  - Resume at 4:45 with answers then data cleaning
- **Basics** (first three class sessions)
  - data classes
  - data structures
  - if/else and logicals
  - basic loops
  - basic functions
- I don't imagine you'll need to "study" unless you feel hazy on these things. If so, this is my way to incentivize you to get caught up
  - Review class scripts
  - Review my problem set solutions
- No ChatGPT or messaging each other
- All other resources are fine to use
- Distributed via GitHub
- Submit however you prefer so not stressful
  - Github
  - Email
  - even Slack

Midterm

- 45 minutes, then 30 minute break
  - Resume at 4:45 with answers then data cleaning
- **Basics** (first three class sessions)
  - data classes
  - data structures
  - if/else and logicals
  - basic loops
  - basic functions
- I don't imagine you'll need to "study" unless you feel hazy on these things. If so, this is my way to incentivize you to get caught up
  - Review class scripts
  - Review my problem set solutions
- No ChatGPT or messaging each other
- All other resources are fine to use
- Distributed via GitHub
- Submit however you prefer so not stressful
  - Github
  - Email
  - even Slack

Midterm

- 45 minutes, then 30 minute break
  - Resume at 4:45 with answers then data cleaning
- **Basics** (first three class sessions)
  - data classes
  - data structures
  - if/else and logicals
  - basic loops
  - basic functions
- I don't imagine you'll need to "study" unless you feel hazy on these things. If so, this is my way to incentivize you to get caught up
  - Review class scripts
  - Review my problem set solutions
- No ChatGPT or messaging each other
- All other resources are fine to use
- Distributed via GitHub
- Submit however you prefer so not stressful
  - Github
  - Email
  - even Slack

Midterm

- 45 minutes, then 30 minute break
  - Resume at 4:45 with answers then data cleaning
- **Basics** (first three class sessions)
  - data classes
  - data structures
  - if/else and logicals
  - basic loops
  - basic functions
- I don't imagine you'll need to "study" unless you feel hazy on these things. If so, this is my way to incentivize you to get caught up
  - Review class scripts
  - Review my problem set solutions
- No ChatGPT or messaging each other
- All other resources are fine to use
- Distributed via GitHub
- Submit however you prefer so not stressful
  - Github
  - Email
  - even Slack

Midterm

- 45 minutes, then 30 minute break
  - Resume at 4:45 with answers then data cleaning
- **Basics** (first three class sessions)
  - data classes
  - data structures
  - if/else and logicals
  - basic loops
  - basic functions
- I don't imagine you'll need to "study" unless you feel hazy on these things. If so, this is my way to incentivize you to get caught up
  - Review class scripts
  - Review my problem set solutions
- No ChatGPT or messaging each other
- All other resources are fine to use
- Distributed via GitHub
- Submit however you prefer so not stressful
  - Github
  - Email
  - even Slack

Midterm

- 45 minutes, then 30 minute break
  - Resume at 4:45 with answers then data cleaning
- **Basics** (first three class sessions)
  - data classes
  - data structures
  - if/else and logicals
  - basic loops
  - basic functions
- I don't imagine you'll need to "study" unless you feel hazy on these things. If so, this is my way to incentivize you to get caught up
  - Review class scripts
  - Review my problem set solutions
- No ChatGPT or messaging each other
- All other resources are fine to use
- Distributed via GitHub
- Submit however you prefer so not stressful
  - Github
  - Email
  - even Slack

Midterm

- 45 minutes, then 30 minute break
  - Resume at 4:45 with answers then data cleaning
- **Basics** (first three class sessions)
  - data classes
  - data structures
  - if/else and logicals
  - basic loops
  - basic functions
- I don't imagine you'll need to "study" unless you feel hazy on these things. If so, this is my way to incentivize you to get caught up
  - Review class scripts
  - Review my problem set solutions
- No ChatGPT or messaging each other
- All other resources are fine to use
- Distributed via GitHub
- Submit however you prefer so not stressful
  - Github
  - Email
  - even Slack

Webscraping considerations and warnings

Webscraping considerations and warnings

Three main things to consider:

- Terms of service
- Intellectual property rights/copyright material
- Data privacy

Webscraping considerations and warnings

Three main things to consider:

- Terms of service
- Intellectual property rights/copyright material
- Data privacy

Webscraping considerations and warnings

Three main things to consider:

- Terms of service
- Intellectual property rights/copyright material
- Data privacy

Webscraping considerations and warnings

Three main things to consider:

- Terms of service
- Intellectual property rights/copyright material
- Data privacy

Webscraping considerations and warnings

Three main things to consider:

- Terms of service
- Intellectual property rights/copyright material
- Data privacy

In more detail...

- **Don't** exploit vulnerabilities in a website's code to gain access to data you are not legally permitted to have
  - » Ex: Selenium to scrape Facebook
- Always read the terms of service
- If website is okay with you accessing your data, it'll be:
  - » Publicly accessible (i.e., without needing an account)
    - U.S. appeals court said so re: LinkedIn
  - » It has an API you can use
- Note check with IRB, too for human subjects data collection
- Note you often can't share the raw data
  - » Ex: Twitter
- In sum
  - » There are lots of legal regulations you could violate
  - » How litigious are these companies? We don't want to find out.

In more detail...

- **Don't** exploit vulnerabilities in a website's code to gain access to data you are not legally permitted to have
  - » Ex: Selenium to scrape Facebook
- Always read the terms of service
- If website is okay with you accessing your data, it'll be:
  - » Publicly accessible (i.e., without needing an account)
    - U.S. appeals court said so re: LinkedIn
  - » It has an API you can use
- Note check with IRB, too for human subjects data collection
- Note you often can't share the raw data
  - » Ex: Twitter
- In sum
  - » There are lots of legal regulations you could violate
  - » How litigious are these companies? We don't want to find out.

In more detail...

- **Don't** exploit vulnerabilities in a website's code to gain access to data you are not legally permitted to have
  - » Ex: Selenium to scrape Facebook
- Always read the terms of service
- If website is okay with you accessing your data, it'll be:
  - » Publicly accessible (i.e., without needing an account)
    - U.S. appeals court said so re: LinkedIn
  - » It has an API you can use
- Note check with IRB, too for human subjects data collection
- Note you often can't share the raw data
  - » Ex: Twitter
- In sum
  - » There are lots of legal regulations you could violate
  - » How litigious are these companies? We don't want to find out.

In more detail...

- **Don't** exploit vulnerabilities in a website's code to gain access to data you are not legally permitted to have
  - » Ex: Selenium to scrape Facebook
- Always read the terms of service
- If website is okay with you accessing your data, it'll be:
  - » Publicly accessible (i.e., without needing an account)
    - U.S. appeals court said so re: LinkedIn
  - » It has an API you can use
- Note check with IRB, too for human subjects data collection
- Note you often can't share the raw data
  - » Ex: Twitter
- In sum
  - » There are lots of legal regulations you could violate
  - » How litigious are these companies? We don't want to find out.

In more detail...

- **Don't** exploit vulnerabilities in a website's code to gain access to data you are not legally permitted to have
  - » Ex: Selenium to scrape Facebook
- Always read the terms of service
- If website is okay with you accessing your data, it'll be:
  - » Publicly accessible (i.e., without needing an account)
    - U.S. appeals court said so re: LinkedIn
  - » It has an API you can use
- Note check with IRB, too for human subjects data collection
- Note you often can't share the raw data
  - » Ex: Twitter
- In sum
  - » There are lots of legal regulations you could violate
  - » How litigious are these companies? We don't want to find out.

In more detail...

- **Don't** exploit vulnerabilities in a website's code to gain access to data you are not legally permitted to have
  - » Ex: Selenium to scrape Facebook
- Always read the terms of service
- If website is okay with you accessing your data, it'll be:
  - » Publicly accessible (i.e., without needing an account)
    - U.S. appeals court said so re: LinkedIn
  - » It has an API you can use
- Note check with IRB, too for human subjects data collection
- Note you often can't share the raw data
  - » Ex: Twitter
- In sum
  - » There are lots of legal regulations you could violate
  - » How litigious are these companies? We don't want to find out.

In more detail...

- **Don't** exploit vulnerabilities in a website's code to gain access to data you are not legally permitted to have
  - » Ex: Selenium to scrape Facebook
- Always read the terms of service
- If website is okay with you accessing your data, it'll be:
  - » Publicly accessible (i.e., without needing an account)
    - U.S. appeals court said so re: LinkedIn
  - » It has an API you can use
- Note check with IRB, too for human subjects data collection
- Note you often can't share the raw data
  - » Ex: Twitter
- In sum
  - » There are lots of legal regulations you could violate
  - » How litigious are these companies? We don't want to find out.

In more detail...

- **Don't** exploit vulnerabilities in a website's code to gain access to data you are not legally permitted to have
  - » Ex: Selenium to scrape Facebook
- Always read the terms of service
- If website is okay with you accessing your data, it'll be:
  - » Publicly accessible (i.e., without needing an account)
    - U.S. appeals court said so re: LinkedIn
  - » It has an API you can use
- Note check with IRB, too for human subjects data collection
- Note you often can't share the raw data
  - » Ex: Twitter
- In sum
  - » There are lots of legal regulations you could violate
  - » How litigious are these companies? We don't want to find out.

In more detail...

- **Don't** exploit vulnerabilities in a website's code to gain access to data you are not legally permitted to have
  - » Ex: Selenium to scrape Facebook
- Always read the terms of service
- If website is okay with you accessing your data, it'll be:
  - » Publicly accessible (i.e., without needing an account)
    - U.S. appeals court said so re: LinkedIn
  - » It has an API you can use
- Note check with IRB, too for human subjects data collection
- Note you often can't share the raw data
  - » Ex: Twitter
- In sum
  - » There are lots of legal regulations you could violate
  - » How litigious are these companies? We don't want to find out.

In more detail...

- **Don't** exploit vulnerabilities in a website's code to gain access to data you are not legally permitted to have
  - » Ex: Selenium to scrape Facebook
- Always read the terms of service
- If website is okay with you accessing your data, it'll be:
  - » Publicly accessible (i.e., without needing an account)
    - U.S. appeals court said so re: LinkedIn
  - » It has an API you can use
- Note check with IRB, too for human subjects data collection
- Note you often can't share the raw data
  - » Ex: Twitter
- In sum
  - » There are lots of legal regulations you could violate
  - » How litigious are these companies? We don't want to find out.

In more detail...

- **Don't** exploit vulnerabilities in a website's code to gain access to data you are not legally permitted to have
  - » Ex: Selenium to scrape Facebook
- Always read the terms of service
- If website is okay with you accessing your data, it'll be:
  - » Publicly accessible (i.e., without needing an account)
    - U.S. appeals court said so re: LinkedIn
  - » It has an API you can use
- Note check with IRB, too for human subjects data collection
- Note you often can't share the raw data
  - » Ex: Twitter
- In sum
  - » There are lots of legal regulations you could violate
  - » How litigious are these companies? We don't want to find out.

In more detail...

- **Don't** exploit vulnerabilities in a website's code to gain access to data you are not legally permitted to have
  - Ex: Selenium to scrape Facebook
- Always read the terms of service
- If website is okay with you accessing your data, it'll be:
  - Publicly accessible (i.e., without needing an account)
    - U.S. appeals court said so re: LinkedIn
  - It has an API you can use
- Note check with IRB, too for human subjects data collection
- Note you often can't share the raw data
  - Ex: Twitter
- In sum
  - There are lots of legal regulations you could violate
  - How litigious are these companies? We don't want to find out.

In more detail...

- **Don't** exploit vulnerabilities in a website's code to gain access to data you are not legally permitted to have
  - » Ex: Selenium to scrape Facebook
- Always read the terms of service
- If website is okay with you accessing your data, it'll be:
  - » Publicly accessible (i.e., without needing an account)
    - U.S. appeals court said so re: LinkedIn
  - » It has an API you can use
- Note check with IRB, too for human subjects data collection
- Note you often can't share the raw data
  - » Ex: Twitter
- In sum
  - » There are lots of legal regulations you could violate
  - » How litigious are these companies? We don't want to find out.

In more detail...

- **Don't** exploit vulnerabilities in a website's code to gain access to data you are not legally permitted to have
  - » Ex: Selenium to scrape Facebook
- Always read the terms of service
- If website is okay with you accessing your data, it'll be:
  - » Publicly accessible (i.e., without needing an account)
    - U.S. appeals court said so re: LinkedIn
  - » It has an API you can use
- Note check with IRB, too for human subjects data collection
- Note you often can't share the raw data
  - » Ex: Twitter
- In sum
  - » There are lots of legal regulations you could violate
  - » How litigious are these companies? We don't want to find out.

API intuition

What does it stand for?

- Application
  - » They have an application, we have an “application,” we’re trying to communicate with each other
- Programming
- Interface
  - » how they’re having you interact
  - » API documentation will say how your structure your **requests** for data and what to expect for a **response**, or how the data will be returned

What does it stand for?

- Application
  - They have an application, we have an "application," we're trying to communicate with each other
- Programming
- Interface
  - how they're having you interact
  - API documentation will say how your structure your **requests** for data and what to expect for a **response**, or how the data will be returned

What does it stand for?

- Application
  - » They have an application, we have an "application," we're trying to communicate with each other
- Programming
- Interface
  - » how they're having you interact
  - » API documentation will say how your structure your **requests** for data and what to expect for a **response**, or how the data will be returned

What does it stand for?

- Application
  - They have an application, we have an "application," we're trying to communicate with each other
- Programming
- Interface
  - how they're having you interact
  - API documentation will say how your structure your **requests** for data and what to expect for a **response**, or how the data will be returned

What does it stand for?

- Application
  - They have an application, we have an "application," we're trying to communicate with each other
- Programming
- Interface
  - how they're having you interact
  - API documentation will say how your structure your **requests** for data and what to expect for a **response**, or how the data will be returned

What does it stand for?

- Application
  - » They have an application, we have an "application," we're trying to communicate with each other
- Programming
- Interface
  - » how they're having you interact
  - » API documentation will say how your structure your **requests** for data and what to expect for a **response**, or how the data will be returned

Twitter example

Twitter (an app) is happy to share data with other apps, but through a very structured process

- This U.S. trending topics app
- Lots of apps to help with marketing
- . . . and of course academic data collection

Twitter example

Twitter (an app) is happy to share data with other apps, but through a very structured process

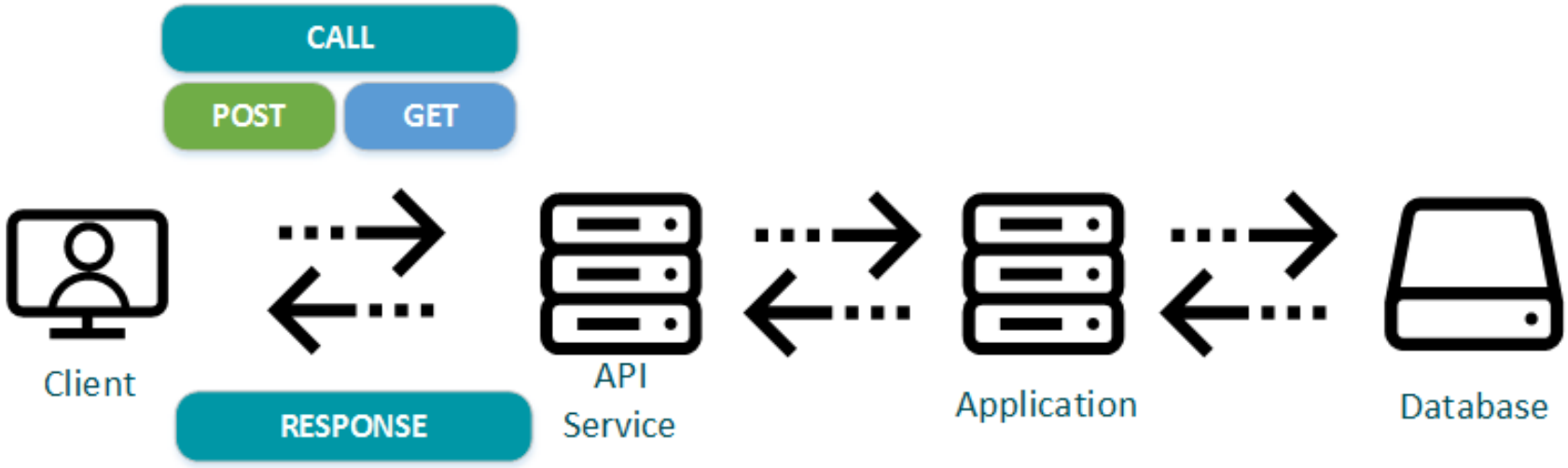
- This U.S. trending topics app
- Lots of apps to help with marketing
- . . . and of course academic data collection

Twitter example

Twitter (an app) is happy to share data with other apps, but through a very structured process

- This U.S. trending topics app
- Lots of apps to help with marketing
- . . . and of course academic data collection

How does it work? Very big picutre



- We are the client wanting responses from the server
- Image from here

How does it work? http as the very basic ingredients

- HyperText Transfer Protocol is the basic way that data is passed and accessed across the web
- GET, POST, PUT, DELETE and HEAD are the main commands
  - Mostly you will use GET and POST
    - GET = get data :)
  - We will use the httr package to do this in R

How does it work? http as the very basic ingredients

- HyperText Transfer Protocol is the basic way that data is passed and accessed across the web
- GET, POST, PUT, DELETE and HEAD are the main commands
  - Mostly you will use GET and POST
    - GET = get data :)
  - We will use the httr package to do this in R

How does it work? http as the very basic ingredients

- HyperText Transfer Protocol is the basic way that data is passed and accessed across the web
- GET, POST, PUT, DELETE and HEAD are the main commands
  - Mostly you will use GET and POST
    - GET = get data :)
  - We will use the httr package to do this in R

How does it work? http as the very basic ingredients

- HyperText Transfer Protocol is the basic way that data is passed and accessed across the web
- GET, POST, PUT, DELETE and HEAD are the main commands
  - Mostly you will use GET and POST
    - GET = get data :)
  - We will use the httr package to do this in R

How does it work? http as the very basic ingredients

- HyperText Transfer Protocol is the basic way that data is passed and accessed across the web
- GET, POST, PUT, DELETE and HEAD are the main commands
  - Mostly you will use GET and POST
    - GET = get data :)
  - We will use the httr package to do this in R

How does it work? http as the very basic ingredients

- HyperText Transfer Protocol is the basic way that data is passed and accessed across the web
- GET, POST, PUT, DELETE and HEAD are the main commands
  - Mostly you will use GET and POST
    - GET = get data :)
  - We will use the httr package to do this in R

Example

How would we use this API in R to find out how many people are in space right now?

- Open Notify API for some of NASA's data
  - No R package to help us!
  - We need to know the basics
- APIs have *endpoints* – think of these as function that are available.
  - In the documentation it will explain what arguments these endpoints accept.
  - These are (sometimes) passed in via the `http` request.
  - They will then return to you a message (did it work) and an object

Example from here

Example

How would we use this API in R to find out how many people are in space right now?

- Open Notify API for some of NASA's data
  - » No R package to help us!
  - » We need to know the basics
- APIs have *endpoints* – think of these as function that are available.
  - » In the documentation it will explain what arguments these endpoints accept.
  - » These are (sometimes) passed in via the `http` request.
  - » They will then return to you a message (did it work) and an object

Example from here

Example

How would we use this API in R to find out how many people are in space right now?

- Open Notify API for some of NASA's data
  - » No R package to help us!
  - » We need to know the basics
- APIs have *endpoints* – think of these as function that are available.
  - » In the documentation it will explain what arguments these endpoints accept.
  - » These are (sometimes) passed in via the `http` request.
  - » They will then return to you a message (did it work) and an object

Example from here

Example

How would we use this API in R to find out how many people are in space right now?

- Open Notify API for some of NASA's data
  - » No R package to help us!
  - » We need to know the basics
- APIs have *endpoints* – think of these as function that are available.
  - » In the documentation it will explain what arguments these endpoints accept.
  - » These are (sometimes) passed in via the `http` request.
  - » They will then return to you a message (did it work) and an object

Example from here

Example

How would we use this API in R to find out how many people are in space right now?

- Open Notify API for some of NASA's data
  - No R package to help us!
  - We need to know the basics
- APIs have *endpoints* – think of these as function that are available.
  - In the documentation it will explain what arguments these endpoints accept.
  - These are (sometimes) passed in via the `http` request.
  - They will then return to you a message (did it work) and an object

Example from here

Example

How would we use this API in R to find out how many people are in space right now?

- Open Notify API for some of NASA's data
  - No R package to help us!
  - We need to know the basics
- APIs have *endpoints* – think of these as function that are available.
  - In the documentation it will explain what arguments these endpoints accept.
  - These are (sometimes) passed in via the `http` request.
  - They will then return to you a message (did it work) and an object

Example from here

Example

How would we use this API in R to find out how many people are in space right now?

- Open Notify API for some of NASA's data
  - No R package to help us!
  - We need to know the basics
- APIs have *endpoints* – think of these as function that are available.
  - In the documentation it will explain what arguments these endpoints accept.
  - These are (sometimes) passed in via the `http` request.
  - They will then return to you a message (did it work) and an object

Example from here

Example

How would we use this API in R to find out how many people are in space right now?

- Open Notify API for some of NASA's data
  - No R package to help us!
  - We need to know the basics
- APIs have *endpoints* – think of these as function that are available.
  - In the documentation it will explain what arguments these endpoints accept.
  - These are (sometimes) passed in via the `http` request.
  - They will then return to you a message (did it work) and an object

Example from here

day06-httr.R

Summary

We used the `httr` and `jsonlite` packages to:

1. Use `http` requests to pass the right arguments to the right API endpoint.
2. Get a JSON object back.
3. Optionally use that result to structure a query to another endpoint.
4. Repeat (1-3) until you have the data you want.

Summary

We used the `httr` and `jsonlite` packages to:

1. Use `http` requests to pass the right arguments to the right API endpoint.
2. Get a JSON object back.
3. Optionally use that result to structure a query to another endpoint.
4. Repeat (1-3) until you have the data you want.

Summary

We used the `httr` and `jsonlite` packages to:

1. Use `http` requests to pass the right arguments to the right API endpoint.
2. Get a JSON object back.
3. Optionally use that result to structure a query to another endpoint.
4. Repeat (1-3) until you have the data you want.

Summary

We used the `httr` and `jsonlite` packages to:

1. Use `http` requests to pass the right arguments to the right API endpoint.
2. Get a JSON object back.
3. Optionally use that result to structure a query to another endpoint.
4. Repeat (1-3) until you have the data you want.

Summary

We used the `httr` and `jsonlite` packages to:

1. Use `http` requests to pass the right arguments to the right API endpoint.
2. Get a JSON object back.
3. Optionally use that result to structure a query to another endpoint.
4. Repeat (1-3) until you have the data you want.

Other notes on APIs

- All APIs have different interfaces so there's a learning curve for each
  - But, don't forget the logic discussed so far is the same
- You also might run into...
  - *rate limits* controlling how much data they will return to you, usually over a set period of time
    - Ex: 1000 queries a day
    - Ex: 20 queries an hour
  - needing *authentication* often using a *key*
    - Note that some (e.g., Open Secrets and Twitter) require you to register or even apply

Other notes on APIs

- All APIs have different interfaces so there's a learning curve for each
  - But, don't forget the logic discussed so far is the same
- You also might run into...
  - *rate limits* controlling how much data they will return to you, usually over a set period of time
    - Ex: 1000 queries a day
    - Ex: 20 queries an hour
  - needing *authentication* often using a *key*
    - Note that some (e.g., Open Secrets and Twitter) require you to register or even apply

Other notes on APIs

- All APIs have different interfaces so there's a learning curve for each
  - But, don't forget the logic discussed so far is the same
- You also might run into...
  - *rate limits* controlling how much data they will return to you, usually over a set period of time
    - Ex: 1000 queries a day
    - Ex: 20 queries an hour
  - needing *authentication* often using a *key*
    - Note that some (e.g., Open Secrets and Twitter) require you to register or even apply

Other notes on APIs

- All APIs have different interfaces so there's a learning curve for each
  - But, don't forget the logic discussed so far is the same
- You also might run into...
  - *rate limits* controlling how much data they will return to you, usually over a set period of time
    - Ex: 1000 queries a day
    - Ex: 20 queries an hour
  - needing *authentication* often using a *key*
    - Note that some (e.g., Open Secrets and Twitter) require you to register or even apply

Other notes on APIs

- All APIs have different interfaces so there's a learning curve for each
  - But, don't forget the logic discussed so far is the same
- You also might run into...
  - *rate limits* controlling how much data they will return to you, usually over a set period of time
    - Ex: 1000 queries a day
    - Ex: 20 queries an hour
  - needing *authentication* often using a *key*
    - Note that some (e.g., Open Secrets and Twitter) require you to register or even apply

Other notes on APIs

- All APIs have different interfaces so there's a learning curve for each
  - But, don't forget the logic discussed so far is the same
- You also might run into...
  - *rate limits* controlling how much data they will return to you, usually over a set period of time
    - Ex: 1000 queries a day
    - Ex: 20 queries an hour
  - needing *authentication* often using a *key*
    - Note that some (e.g., Open Secrets and Twitter) require you to register or even apply

Other notes on APIs

- All APIs have different interfaces so there's a learning curve for each
  - But, don't forget the logic discussed so far is the same
- You also might run into...
  - *rate limits* controlling how much data they will return to you, usually over a set period of time
    - Ex: 1000 queries a day
    - Ex: 20 queries an hour
  - needing *authentication* often using a *key*
    - Note that some (e.g., Open Secrets and Twitter) require you to register or even apply

Other notes on APIs

- All APIs have different interfaces so there's a learning curve for each
  - But, don't forget the logic discussed so far is the same
- You also might run into...
  - *rate limits* controlling how much data they will return to you, usually over a set period of time
    - Ex: 1000 queries a day
    - Ex: 20 queries an hour
  - needing *authentication* often using a *key*
    - Note that some (e.g., Open Secrets and Twitter) require you to register or even apply

API wrappers

API wrappers

- To make life easier, people write API "**wrappers**"
  - Allow you to use R functions instead of clunky GET and POST
  - returns response nicely ready to use as R objects
  - . . . but in the background httr requests is all R package is doing
  - don't rebuild the wheel!

API wrappers

- To make life easier, people write API "**wrappers**"
  - Allow you to use R functions instead of clunky GET and POST
  - returns response nicely ready to use as R objects
  - . . . but in the background httr requests is all R package is doing
  - don't rebuild the wheel!

API wrappers

- To make life easier, people write API "**wrappers**"
  - Allow you to use R functions instead of clunky GET and POST
  - returns response nicely ready to use as R objects
  - . . . but in the background httr requests is all R package is doing
  - don't rebuild the wheel!

API wrappers

- To make life easier, people write API "**wrappers**"
  - Allow you to use R functions instead of clunky GET and POST
  - returns response nicely ready to use as R objects
  - . . . but in the background httr requests is all R package is doing
  - don't rebuild the wheel!

API wrappers

- To make life easier, people write API "**wrappers**"
  - Allow you to use R functions instead of clunky GET and POST
  - returns response nicely ready to use as R objects
  - . . . but in the background httr requests is all R package is doing
  - don't rebuild the wheel!

API wrappers

- To make life easier, people write API "**wrappers**"
  - Allow you to use R functions instead of clunky GET and POST
  - returns response nicely ready to use as R objects
  - . . . but in the background httr requests is all R package is doing
  - don't rebuild the wheel!

Example 1

- MTurk user interface online (point and click)
- MTurk API documentation
- `pyMTurkR` R package

Example 1

- MTurk user interface online (point and click)
- MTurk API documentation
- `pyMTurkR` R package

Example 1

- MTurk user interface online (point and click)
- MTurk API documentation
- `pyMTurkR` R package

Example 1

- MTurk user interface online (point and click)
- MTurk API documentation
- `pyMTurkR` R package

Other examples

- Wikipedia
  - general API documentation
  - R wrapper
- Twitter
  - general API documentation
  - R wrapper
- Reddit
- OpenSecrets
- World Bank
- . . . always look for the API and R wrapper

Other examples

- Wikipedia
  - general API documentation
  - R wrapper
- Twitter
  - general API documentation
  - R wrapper
- Reddit
- OpenSecrets
- World Bank
- . . . always look for the API and R wrapper

Other examples

- Wikipedia
  - » general API documentation
  - » R wrapper
- Twitter
  - » general API documentation
  - » R wrapper
- Reddit
- OpenSecrets
- World Bank
- . . . always look for the API and R wrapper

Our next example together

- Google Civic R Wrapper
- API Documentation
  - Requires a little more set up!
- Follow instructions here **until you have your key ready to copy/paste** somewhere
- Then we'll work on the next script together

Our next example together

- Google Civic R Wrapper
- API Documentation
  - Requires a little more set up!
- Follow instructions here **until you have your key ready to copy/paste** somewhere
- Then we'll work on the next script together

Our next example together

- Google Civic R Wrapper
- API Documentation
  - Requires a little more set up!
- Follow instructions here **until you have your key ready to copy/paste** somewhere
- Then we'll work on the next script together

Our next example together

- Google Civic R Wrapper
- API Documentation
  - Requires a little more set up!
- Follow instructions here **until you have your key ready to copy/paste** somewhere
- Then we'll work on the next script together

day06-wrappers.R