

APIs & Plumber

Dr Olly Butters

Public Health, Policy and Systems

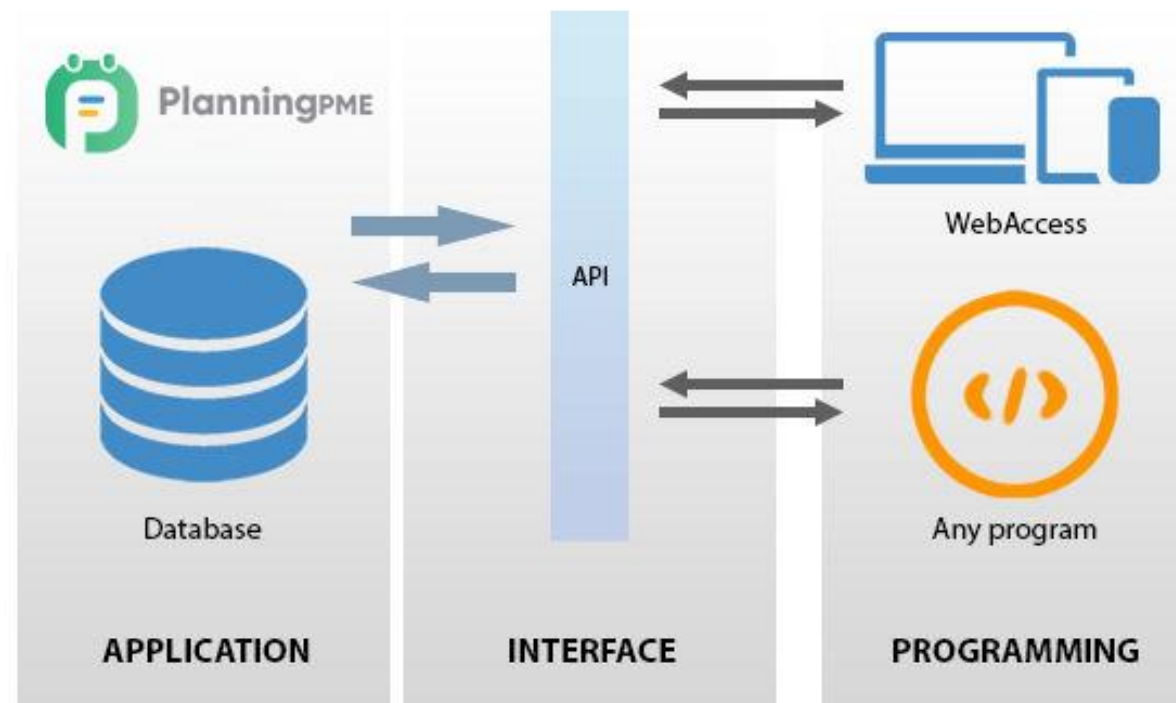
olly.butters@liverpool.ac.uk

Session overview

- What is an API, where are they used? (~25 mins)
- Exercise - Two R examples of using existing external APIs. (~15 mins)
- Introduction to plumber. (~15 mins)
- Exercise - Write some plumber code. (~30 mins)
- Final comments. (~5 mins)

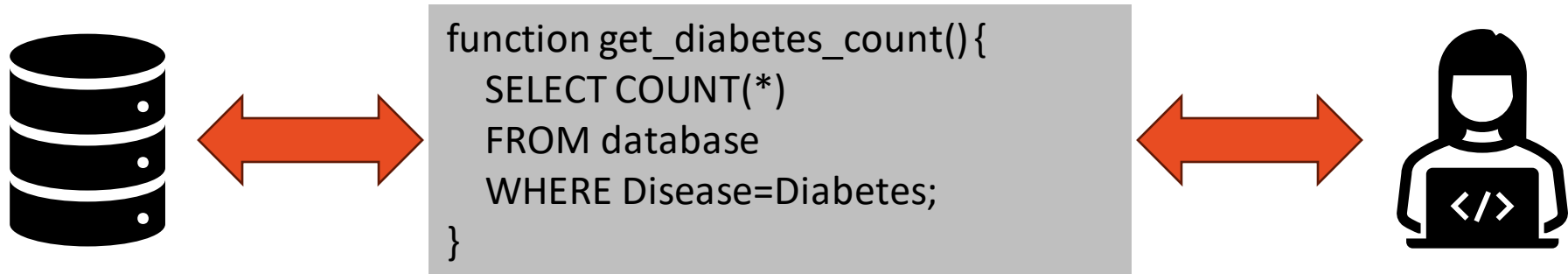
What is an API?

- Application Programming Interface.



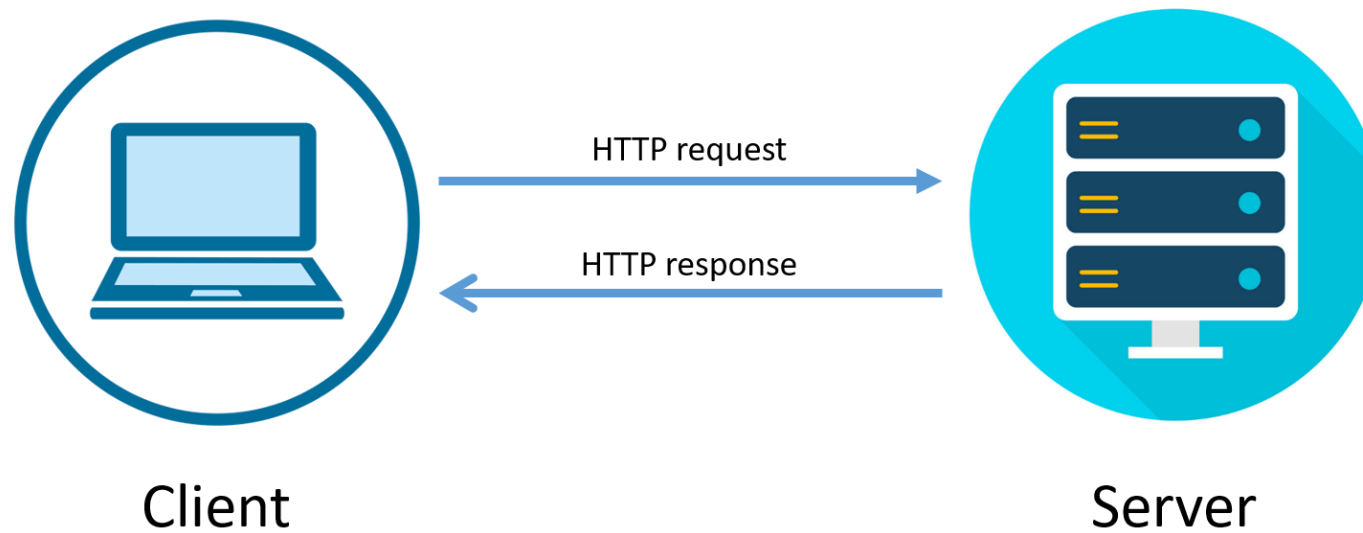
<https://www.planningpme.com/planningpme-api.htm>

Steps towards an API



Name	ID	Disease
John	2345	Diabetes
Jane	7853	Asthma
Judy	1337	Diabetes
Joe	8867	COPD

API messaging

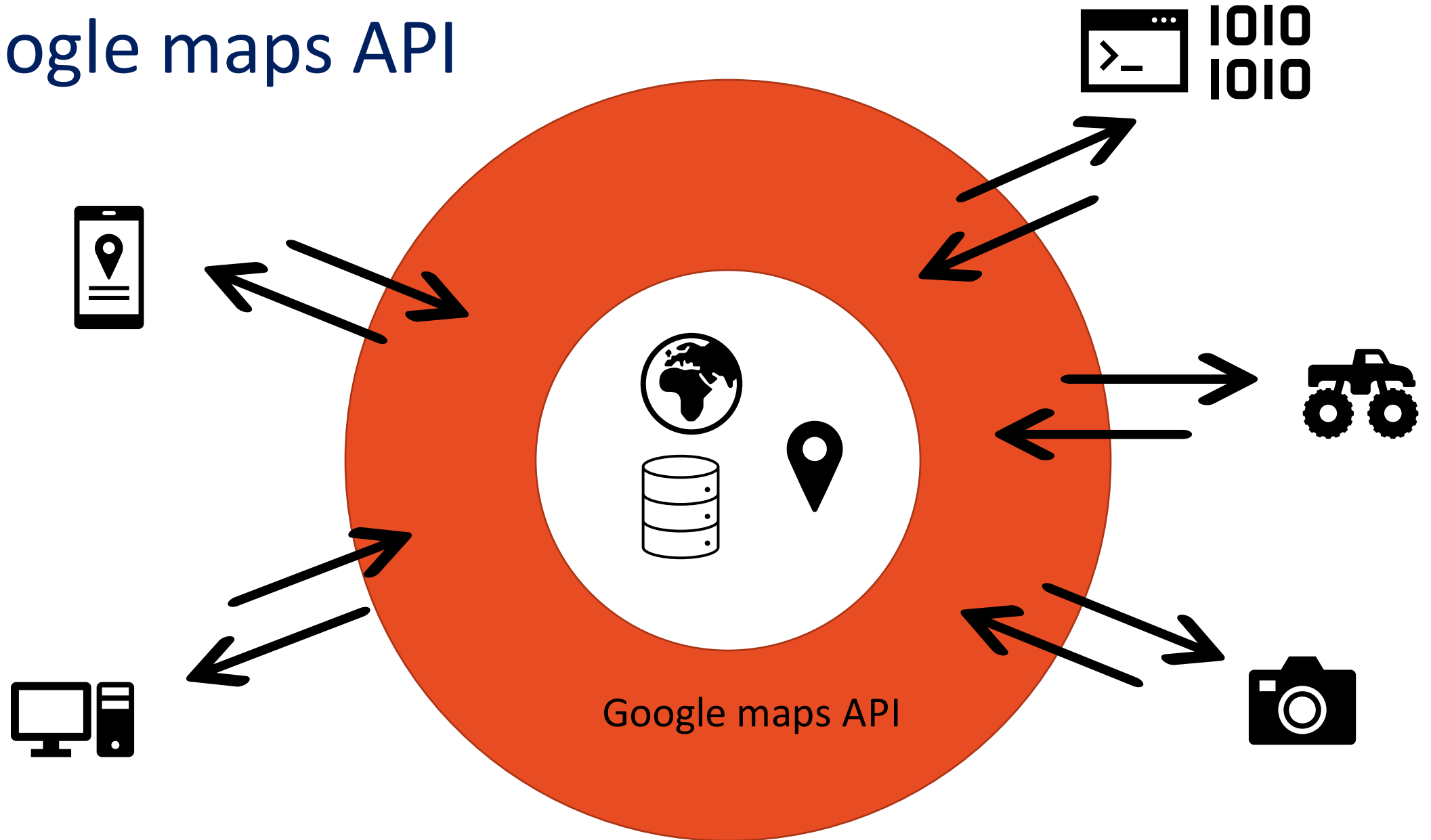


<https://bytesofgigabytes.com/networking/how-http-request-and-response-works/>

Why bother?

- Easier than connecting to underlying applications.
- Can make subset of data/application available.
- Common language.
- Static interface.

Google maps API



Example APIs

Get data

- Google maps
- Wikidata
- Fitbit
- British library
- Data.parliament.uk

Add data

- Twitter
- Facebook
- Instagram

Book appointments

- GP systems
- Restaurants

Manage services

- Amazon Web Services
- Microsoft Azure

Real world interaction

- Google Nest
- Dishwasher
- Burglar alarm

Health data APIs

- Bioportal -> Look up ontologies etc.
- Gov health stats
- NHS Digital -> Loads of APIs to find/get/add data
- Air quality.
- [UK government API catalogue](#)
- [Urban Observatory](#)
- <https://data.police.uk/docs/>
- Care Quality Commission -> locations of care homes

Anatomy of a query URL (the request)

http://www.domain.com:1234/path/to/resource?a=b&x=y

The diagram illustrates the components of the URL `http://www.domain.com:1234/path/to/resource?a=b&x=y`. Red horizontal bars are placed under each component, with red lines connecting them to labels below or above the URL:

- protocol**: Points to `http`
- host**: Points to `www.domain.com`
- port**: Points to `1234`
- resource path**: Points to `/path/to/resource`
- query**: Points to `?a=b&x=y`

Mongolairun, CC BY-SA 4.0 <<https://creativecommons.org/licenses/by-sa/4.0>>, via Wikimedia Commons

Anatomy of a query URL (the request)

port query

http://www.domain.com:1234/path/to/resource?a=b&x=y

protocol host resource path

Pavement Liver building Flat number 3 Kitchen/cupboard/get_mug colour=red

Road Liver building Garage number 2 Park/car

Mongolarius, CC BY-SA 4.0 <<https://creativecommons.org/licenses/by-sa/4.0>>, via Wikimedia Commons

Response

- Typically get a header and content in the response
- Response codes in header
 - 200 – OK
 - 404 – Not found
 - 500 – Internal server error
- Content is usually JSON or XML

Google maps elevation API call example

Request: <https://maps.googleapis.com/maps/api/elevation/json?locations=39.7391536%2C-104.9847034>

Response:

```
{
  "results":
  [
    {
      "elevation": 1608.637939453125,
      "location": { "lat": 39.7391536, "lng": -104.9847034 },
      "resolution": 4.771975994110107,
    },
  ],
  "status": "OK",
}
```

- Exercises 1: How many people are on the International Space Station right now?

1. <https://github.com/OllyButters/HDS-plumber/archive/refs/heads/main.zip>
2. Open README.md
3. Open exercise_1_api_who_is_on_the_ISS_now.R
4. Run the R file a line at a time (Ctrl-Enter) and read the comments as you do.

Exercise 2: Write your own R script to find out what the English Covid-19 rate is and plot it

1. Start a new R file called `exercise_2_api_covid_rate.R`
2. Copy the relevant parts from the first exercise
3. The query URL (the request) is:
[https://api.coronavirus.data.gov.uk/v1/data?filters=areaType=nation;areaName=england&structure={"date":"date","newCases":"newCasesByPublishDate"}](https://api.coronavirus.data.gov.uk/v1/data?filters=areaType=nation;areaName=england&structure={)
4. Can copy URL from the README.md file.
5. Get the data from the API and plot it.
6. If you get really stuck you can look at `exercise_2_api_covid_answer.R`

Exercise 1 & 2 summary

- Used two APIs to get data from remote services.
- Now we are going to build our own APIs and connect to them in a similar way.

Swagger

- Web tool to help explore and use compliant APIs
- <https://api.openaq.org/>

OpenAQ 2.0.0 OAS3
/openapi.json
API for OpenAQ LCS

default

GET	/check-email	Check Email	▼
GET	/verify/{verification_code}	Verify	▼
GET	/register	Get Register	▼
POST	/register	Post Register	▼

v3

GET	/v3/countries/{countries_id}	Get a country by ID	▼
GET	/v3/countries/{countries_id}/locations	Get locations within a country	▼
GET	/v3/countries	Get countries	▼

v1

GET	/v1/countries	Get countries	▼
GET	/v1/latest	Get latest measurements	▼
GET	/v1/latest/{location_id}	Get latest measurements by location ID	▼
GET	/v1/locations	Get locations	▼
GET	/v1/locations/{location_id}	Get location by ID	▼
GET	/v1/measurements	Get a list of measurements	▼

Swagger

GET

/v1/locations

Get locations

▼

GET

/v1/locations/{location_id}

Get location by ID

▼

GET

/v1/measurements

Get a list of measurements

^

Parameters

Try it out

Name	Description
format string (query)	<input type="text" value="format"/>
date_from (query)	<div>Default value : 2000-01-01T00:00:00+00:00</div> <input type="text" value="2000-01-01T00:00:00+00:00"/>
date_to (query)	<div>Default value : 2023-06-08T09:22:00+00:00</div> <input type="text" value="2023-06-08T09:22:00+00:00"/>
limit integer (query) maximum: 100000	<div>Change the number of results returned. e.g. limit=1000 will return up to 1000 results</div> <div>Default value : 100</div> <input type="text" value="100"/>
page integer (query) maximum: 6000	<div>Paginate through results. e.g. page=1 will return first page of results</div> <div>Default value : 1</div> <input type="text" value="1"/>
offset integer (query) maximum: 10000 minimum: 0	<div>Default value : 0</div> <input type="text" value="0"/>
sort string (query)	<div>Available values : asc, desc</div> <div>Default value : desc</div> <div><input type="text" value="desc"/> ▼</div>
has_geo	<input type="text" value=""/>

Decorators

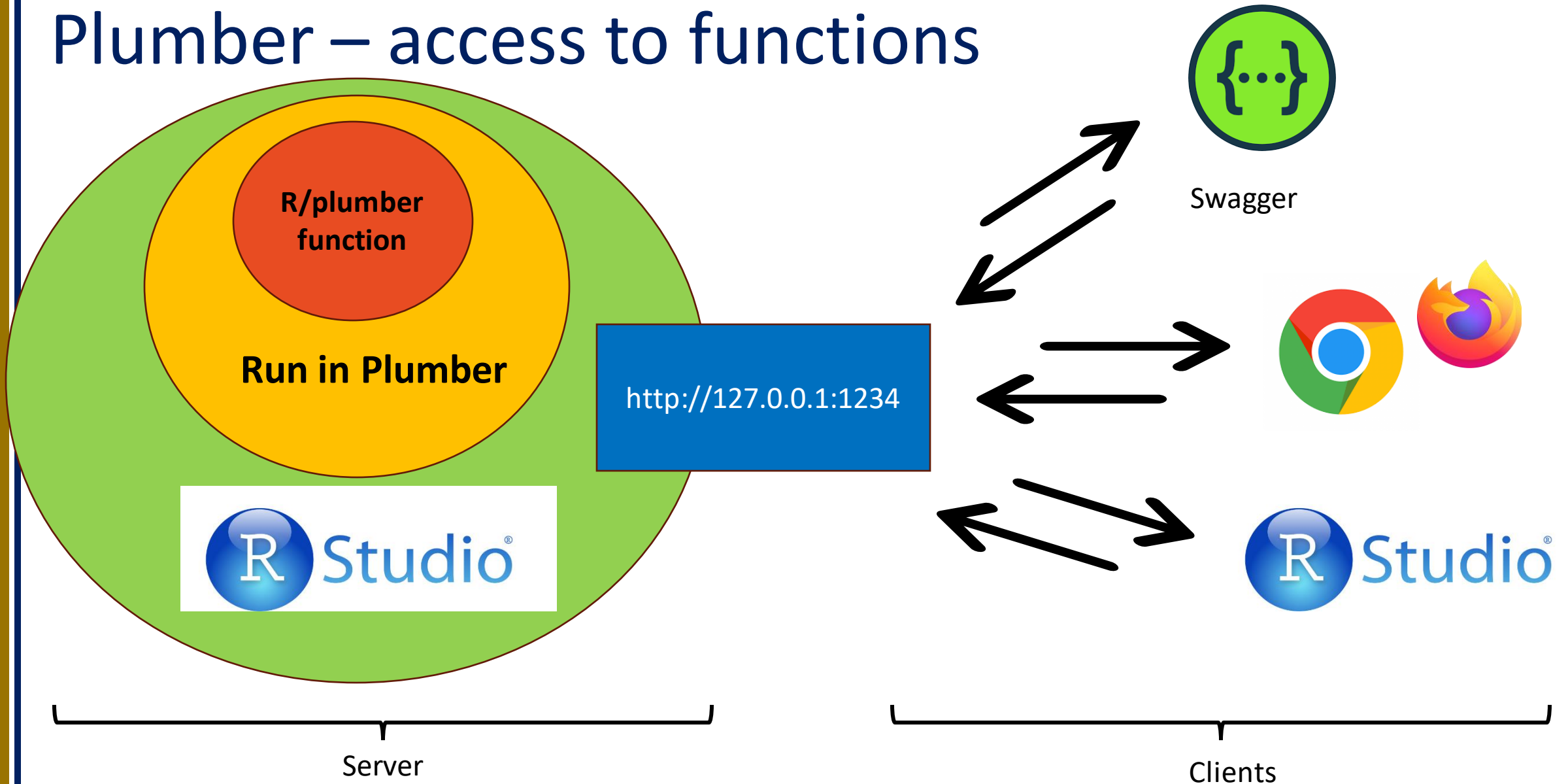
- Decorators let you modify function behaviour without modifying the function code!
- Start with a # so ignored most of the time.
- Common in other languages.

```
* I am a decorator  
my_function <- function()  
{  
  #do awesome stuff  
}
```

Plumber function

```
#* Return the square of a number
#* @param a The number to square
#* @get /square
function(a) {
  as.numeric(a) * as.numeric(a)
}
```

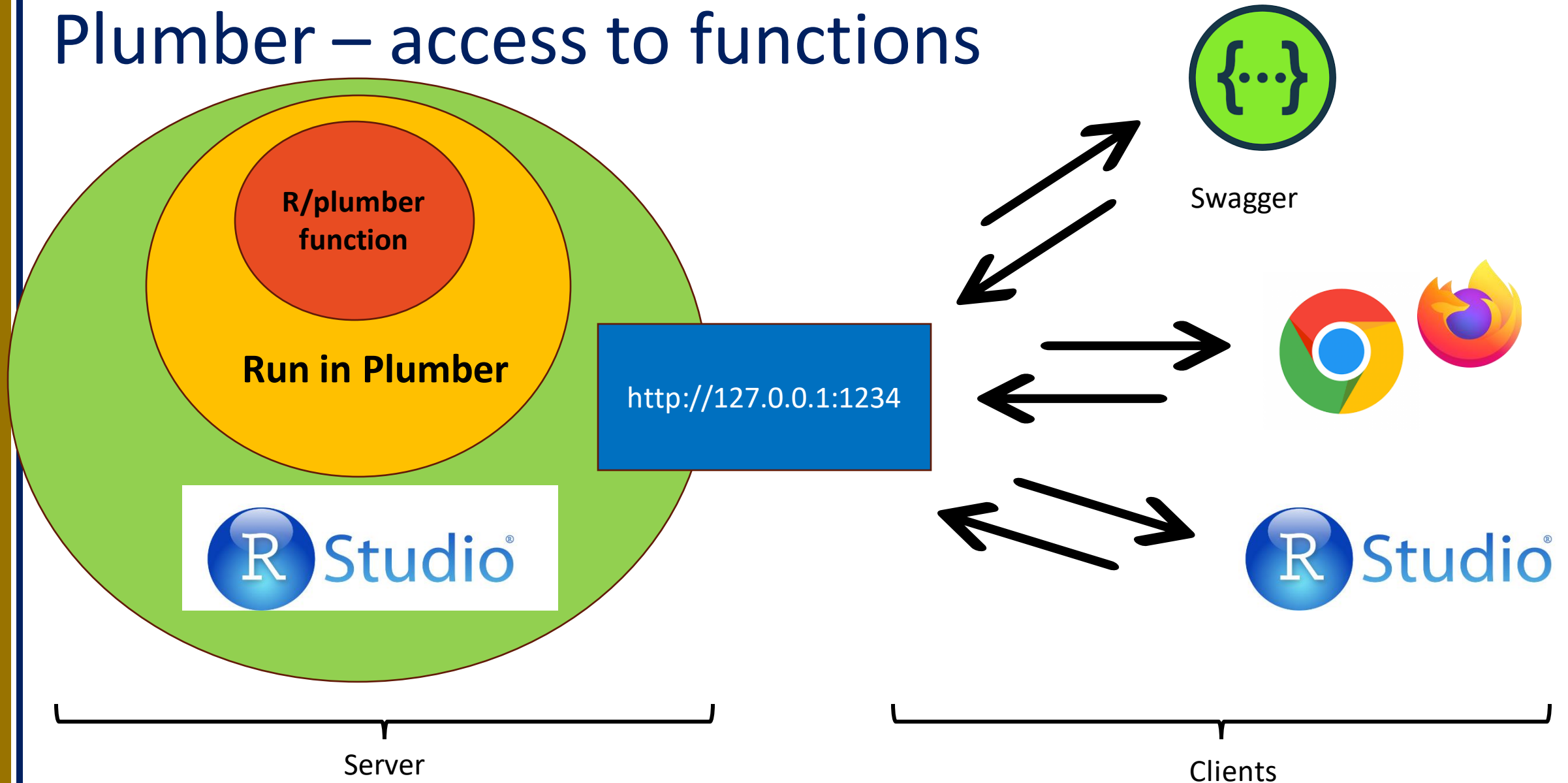
Plumber – access to functions



Exercise 3: Run some plumber code

- Open `exercise_3_plumber_example_server.R`
- Click on the "Run API" button on the top right of the code.
- This will open a web browser with swagger running in it
- Run example plumber functions (`/hello`, `/square`, `/plot`)
- Open request URLs directly in a web browser
- Open a second RStudio instance (Session > New Session), open `exercise_3_plumber_example_client.R`, update the `port_number` variable, run examples.
- More info in the README.md document

Plumber – access to functions



Exercises: write some plumber code

- Exercise 4: Write a plumber function to use gapminder data to show population of the UK in 1982. (Gapminder is a dataset of populations of various countries from 1952 - 2007).
- Exercise 5: Write a plumber function to allow a user to find out the population of any country during any year in gapminder.
- Exercise 6: Write a plumber function to plot the population change of a user defined country.

Additional points

- GET/POST
- Bounds checks
- Security is vital on public APIs
- Good list of public APIs - <https://github.com/public-apis/public-apis>