PROJET 3

## Introduction

In this project, you will have to use an API to get your data from the web.

![](https://d7whxh71cqykp.cloudfront.net/uploads/image/data/2242/04-project-movie.png)

The API used in this project is the ["The Movie DB"](https://www.themoviedb.org).

## Getting access to "The Movie DB"

J’ai créé un compte qui peut être utilisé (si besoin de vérifier la clé )

[Mag.savoye@gmail.com](mailto:Mag.savoye@gmail.com)

Mot de passe Tetra1234!!

Ma clé API : 197aac93c3e105765e64e0d49c5faad2

[My API Settings — The Movie Database (TMDB) (themoviedb.org)](https://www.themoviedb.org/settings/api)

![](https://d7whxh71cqykp.cloudfront.net/uploads/image/data/2243/04-project-movie-setup0.png)

You will probably be asked to verify your account (i.e. click on a link sent to your email and reenter your username and password).

## Exercises

### Part 1

Try to send a `GET` request to [some of the example queries](https://www.themoviedb.org/documentation/api/discover) and inspect the result.

Inspired by these examples, how would you create these new requests:

- What are the highest grossing dramas from 2010?

- Have Will Ferrell and Liam Neeson even been in a movie together?

- Can you find kids movies with Tom Cruise in it?

Include the results in the report.

### Part 2

As you can see in the examples, there are two types of parameters used in the URLs:

- parameters that take an "explicit" value, like `primary\_release\_year=` or `sort\_by=`. When you read their values (e.g. `2014` or `popularity.desc`), you know straight away what is queried.

- parameters that take an "id" value, like `with\_cast=` or `with\_genres=`. When you read their values (e.g. `23659` or `878`), you don't really know what is queried if you don't know which id means what.

This is common as well with APIs.

Parameters that might have complicated/long/confusing spelling (like the title of a movie or the full name of an actor) often use an ID.

What if two movies or two actors have the same name?

That's also a situation where using IDs would help.

The problem is that you need to find these ids before sending the query that you are really interested in.

And to do that you need to prepare another query.

This is when you start reading [the full API documentation](https://developers.themoviedb.org/3/getting-started/introduction).

In this case, you would want to check the [`Search` endpoints](https://developers.themoviedb.org/3/search/search-companies).

The `Search` endpoints let you search by name and find the id for different kinds of resources (companies, people, movie title...).

If you are unsure on how to write these URLs, there is a helpful tab "Try it out" that lets you experiment with the URLs.

From `RStudio`, what query would you make to find the id of the animation movie company "Pixar"?

Show both the query and how you extract the id from the result in your report.

### Part 3

Now that we have the id of Pixar, we should be able to find all the movies that they have worked on.

But you don't know how do a search for movies by companies...

Go read the documentation for the [`/discover/movies` endpoint](https://developers.themoviedb.org/3/discover/movie-discover).

You will see the full list of parameters that you can use for filtering your results.

Some will be familiar since they were used in the examples (e.g. `with\_cast=`, `primary\_release\_year=` or `with\_genre=`).

Other will be new (e.g. `with\_runtime.lte=` that lets you select just the movies that are shorter than a certain time).

Write a query that will give you all the Pixar movies and sort them by descending revenue.

The result will be given to you as a JSON (parsed to a `list` by `{httr}`).

Convert this list to a `tibble` so you have one row per film and one column per interesting piece of information.

Also have a close look at the keys in your `list`.

You will notice that the API sends \*\*"paginated" results\*\* (i.e. look at these `page` and `total\_pages` keys).

It means that you never get more than `x` results at a time (at the time of this writing, this API sends `20` results at a time).

Paginated APIs are \*\*extremely\*\* common as administrators don't want users to send queries that would require a ton of data and block the service for others for a long time.

If you want to get the other pages, you need to play with the `page` parameter in your url.

Further, consider that if you want to repeat the same query over and over for different pages, `purrr::map()` is a useful function.

### Part 4

You may know that Pixar was acquired by Disney in 2006, after they had already been collaborating on films for more than a decade.

For the last part of the report, we are going to look into whether this was a smart strategic decision by Disney, by comparing the popularity of both Disney and Pixar films that came out from 2006 onwards.

- First, acquire the "id" for Disney using the `search` endpoint. Note that if you try to find the company ID for Disney, there will be more than one result (Disney has many subsidiaries with the name "Disney" in it). For this exercise, specifically look for "Walt Disney Pictures" in the USA.

- Second, get the vote averages and vote counts for films from Walt Disney Productions and from Pixar using the `discover/movies` endpoint. Use the API documentation to find out how to get films from 2006 \*onwards\*.

- Now, compare the vote averages using boxplots and a t-test, with the aim of answering the question \*Are the films from Pixar on average more popular than the films from Walt Disney Pictures?\*

A suggestion here would be to filter the data by including only films with a `vote\_count` of at least 50.

Consider that if only a few people voted on the film, the vote average will not be as representative as when lots of people have voted on the film.