IMPC

International Mouse Phenotyping Consortium

@impc

An Introduction to Querying Solr and Use of the IMPC Solr APIs

Marina Kan
Data Engineer at EMBL-EBI
marinak@ebi.ac.uk

20 May 2024

mousephenotype.org

Before We Start...

- We will have both theoretical and practical sections.
- Practical part includes 7 exercises divided into 4 blocks.
- We will use Jupyter notebook to do exercises.
- We created an individual workspace for everyone. To access it:
  - Open the link: https://www.ebi.ac.uk/mi/impc/jupyter/
  - Use your email to log in.
  - Password was sent to you via email.
- If you have any problems, let us know in the chat.

What is Apache Solr?

- Solr is an open-source **search platform**.
  Some of the key features include:

  - Full-text search, that allow to perform complex queries.
  - Faceted search: users can filter search results based on different criteria or attributes.

- We use Solr to access IMPC data for several reasons:

  - It quickly finds what we're looking for in large amounts of data.
  - It provides precise search results, even in complex datasets.

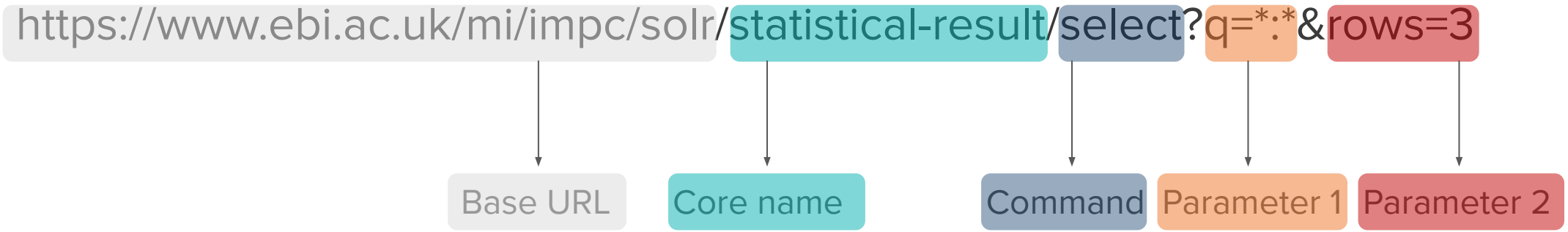
Solr Cores

- Different Solr cores provide access to the IMPC data.
- Solr core is a specific collection of data. Each core has its own data structure and set of fields. Available Solr cores:

- We are going to work with `genotype-phenotype` and `phenodigm` cores during exercises.

- IMPC Solr cores documentation: https://www.ebi.ac.uk/mi/impc/solrdoc/

How to Make Simplest Request in Browser?



Meaning: select all fields and first three docs from the statistical-result core.

- There are different Solr commands, but select is the only available command for the IMPC data.

How to Make Simplest Request in Browser?

https://www.ebi.ac.uk/mi/impc/solr/statistical-result/select?q=*:*&rows=3


```
[
  "responseHeader":{
    "status":0,
    "QTime":0,
    "params":{
      "q":"*:*",
      "rows":"3"}},
  "response":{"numFound":4841785,"start":0,"docs":[
      {
        "doc_id":"e6a2aa38e246824a75ed33ba5c783315",
        "allele_accession_id":"MGI:5707439",
        "allele_name":"targeted mutation 1.1, Velocigene",
        "allele_symbol":"Gprc6a<tm1.1(KOMP)Vlcg>",
        "classification_tag":"Not significant [level = 1e-04, pvalue = 1] ",
        "colony_id":"ET10010",
        "data_type":"categorical",
        "effect_size":6.30914826498463E-4,
        "female_control_count":799,
        "female_ko_effect_p_value":1.0,
        "female_ko_parameter_estimate":0.0,
        "female_mutant_count":4,
        "male_control_count":786,
        "male_ko_effect_p_value":1.0,
        "male_ko_parameter_estimate":0.0,
        "male_mutant_count":2,
        "marker_accession_id":"MGI:2429498",
        "marker_symbol":"Gprc6a",
        "metadata":["Experimenter ID = 43|Location of test = Open bench|Number
          "Experimenter ID = 43|Location of test = Open bench|Number of animals
          "Experimenter ID = 43|Location of test = Open bench|Number of animals
        "metadata_group":"a654d8c674f11e31045504723d61bff8",
        "p_value":1.0,
        "parameter_name":"Forepaw - size",
        "parameter_stable_id":"IMPC_CSD_040_002",
        "phenotype_sex":["male",
          "female"],
        "phenotyping_center":"UC Davis",
        "pipeline_name":"UCD Pipeline",
        "pipeline_stable_id":"UCD_001",
        "procedure_name":"Combined SHIRPA and Dysmorphology",
        "procedure_stable_id":["IMPC_CSD_003"],
        "procedure_stable_key":[801],
        "statistical_method":"Fisher Exact Test framework",
```


- Let's paste the request above to the browser…
- The output is JSON structured like this:
  - `responseHeader` = metadata:
    - Request status
    - Query type
    - Parameters used
  - `response` = data:
    - Total number of documents matched
    - Start of output relative to all documents
    - List of documents contents

How to Make Simplest Request in Browser?

https://www.ebi.ac.uk/mi/impc/solr/statistical-result/select?q=*:*&rows=3


```
{
  "responseHeader":{
    "status":0,
    "QTime":0,
    "params":{
      "q":"*:*",
      "rows":"3"}},
  "response":{"numFound":4841785,"start":0,"docs":[
      {
        "doc_id":"e6a2aa38e246824a75ed33ba5c783315",
        "allele_accession_id":"MGI:5707439",
        "allele_name":"targeted mutation 1.1, Velocigene",
        "allele_symbol":"Gprc6a<tm1.1(KOMP)Vlcg>",
        "classification_tag":"Not significant [level = 1e-04, pva
        "colony_id":"ET10010",
        "data_type":"categorical",
        "effect_size":6.30914826498463E-4,
        "female_control_count":799,
        "female_ko_effect_p_value":1.0,
        "female_ko_parameter_estimate":0.0,
        "female_mutant_count":4,
        "male_control_count":786,
        "male_ko_effect_p_value":1.0,
        "male_ko_parameter_estimate":0.0,
        "male_mutant_count":2,
        "marker_accession_id":"MGI:2429498",
        "marker_symbol":"Gprc6a",
        "metadata":["Experimenter ID = 43|Location of test = Open
          "Experimenter ID = 43|Location of test = Open bench|Num
          "Experimenter ID = 43|Location of test = Open bench|Num
        "metadata_group":"a654d8c674f11e31045504723d61bff8",
        "p_value":1.0,
        "parameter_name":"Forepaw - size",
        "parameter_stable_id":"IMPC_CSD_040_002",
        "phenotype_sex":["male",
          "female"],
```


	allele_accession_id	allele_name	allele_symbol	colony_id	data_type
0	MGI:5609345	targeted mutation 1b, Helmholtz Zentrum Muench...	Lypla1<tm1b(EUCOMM)Hmgu>	LYPAB	categorical
1	MGI:6399913	endonuclease-mediated mutation 1, The Centre f...	Osbpl8<em1(IMPC)Tcp>	TCPR1519_AEKV	adult-gross-path
2	MGI:5637089	targeted mutation 1b, Wellcome Trust Sanger In...	Raph1<tm1b(EUCOMM)Wtsi>	MUFZ	unidimensional

Python Helper Function for Exercises

- In Jupyter notebooks which we will use for exercises, there is a helper Python function called **solr_request.**
- It takes two parameters: **core** name and dictionary of **params.**
- It will:
  - Output the generated URL
  - Report total number of results
  - Display results in an easy to read format. Note that large tables are scrollable horizontally.

```
num_found, df = solr_request(
    core='statistical-result',
    params={
        'q': '*:*', # Request all records.
        'rows': 3   # Request the first three rows
    }
)
```

```
Your request:
https://www.ebi.ac.uk/mi/impc/solr/statistical-result/select?q=%2A%3A%2A&rows=3

Number of found documents: 4070676
```

	allele_accession_id	allele_name	allele_symbol	colony_id	data_type
0	MGI:5609345	targeted mutation 1b, Helmholtz Zentrum Muench...	Lypla1<tm1b(EUCOMM)Hmgu>	LYPAB	categorical
1	MGI:6399913	endonuclease-mediated mutation 1, The Centre f...	Osbpl8<em1(IMPC)Tcp>	TCPR1519_AEKV	adult-gross-path
2	MGI:5637089	targeted mutation 1b, Wellcome Trust Sanger In...	Raph1<tm1b(EUCOMM)Wtsi>	MUFZ	unidimensional

Requesting Specific Fields

- Some Solr cores have many dozens of fields.
- To save time and bandwidth, it makes sense to only request the fields you need.
- The `fl` (field list) parameter controls which fields are returned.
- Warning: if you misspell field name, no error will be generated and this field will be silently omitted from the final result.

```
num_found, df = solr_request(
    core='statistical-result',
    params={
        'q': '*:*', # Request all records.
        'fl': 'marker_symbol,top_level_mp_term_name,effect_size,p_value',
        'rows': 3   # Request the first three rows
    }
)
```

```
Your request:
https://www.ebi.ac.uk/mi/impc/solr/statistical-result/select?q=%2A%3A%2A&fl=marker_symbol%2Ctop_level_mp_term_name%2Ceffect_size%2Cp_value&rows=3

Number of found documents: 4070676
```

	marker_symbol	top_level_mp_term_name	effect_size	p_value
0	Lypla1	[skeleton phenotype]	NaN	NaN
1	Osbpl8	NaN	0.000000	1.000000
2	Raph1	[hearing/vestibular/ear phenotype]	0.154034	0.066499

IMPC

International Mouse Phenotyping Consortium

@impc

Exercises Block A

Querying Specific Fields

- Data can be filtered by specifying *field:value* in the **q** (query) parameter.

- For example, to get the gene with the symbol *Gprc6a*, using the *marker_symbol* field, we can add `marker_symbol:Gprc6a`.

```
num_found, df = solr_request(
    core='statistical-result',
    params={
        'q': 'marker_symbol:Gprc6a', # Request Gprc6a records.
        'fl': 'marker_symbol,top_level_mp_term_name,effect_size,p_value',
        'rows': 3   # Request the first three rows
    }
)
```

```
Your request:
https://www.ebi.ac.uk/mi/impc/solr/statistical-result/select?q=marker_symbol%3AGprc6a&fl=marker_symbol%2Ctop_level_mp_term_name%2Ceffect_size%2Cp_value&rows=3

Number of found documents: 702
```

	marker_symbol	top_level_mp_term_name
0	Gprc6a	[behavior/neurological phenotype]
1	Gprc6a	NaN
2	Gprc6a	[homeostasis/metabolism phenotype]

Range Search

- An asterisk * may be used for either or both endpoints to specify an open-ended range query.

field:[* TO 100]	finds all field values less than or equal to 100.
field:[100 TO *]	finds all field values greater than or equal to 100.
field:[* TO *]	finds any document with a value between the effective values of -Infinity and +Infinity for that field type.

```
num_found, df = solr_request(
    core='statistical-result',
    params={
        'q': 'p_value:[0 TO 1e-4]', # Request p-value from 0 to 1e-4.
        'fl': 'marker_symbol,top_level_mp_term_name,effect_size,p_value',
        'rows': 3   # Request the first three rows
    }
)
```

```
Your request:
https://www.ebi.ac.uk/mi/impc/solr/statistical-result/select?q=p_value%3A%5B0
+TO+1e-4%5D&fl=marker_symbol%2Ctop_level_mp_term_name%2Ceffect_size%2Cp_value
&rows=3

Number of found documents: 45387
```

	effect_size	marker_symbol	p_value	top_level_mp_term_name
0	2.715391	Tbx22	7.228019e-08	[behavior/neurological phenotype]
1	-0.973679	Avpr1a	2.229254e-06	[behavior/neurological phenotype]
2	1.962857	Trarg1	1.889197e-09	[behavior/neurological phenotype]

MPI2

Boolean Operators: AND/OR

- Search parameters can be combined using logical operators.
  - To match both conditions, you specify `filter1 AND filter2`
  - To match any one of the conditions, use specify `filter1 OR filter2`
- For example, to find documents with marker symbol Gprc6a and p-value less than 1e-4, you can use: `marker_symbol:Gprc6a AND p_value:[0 TO 1e-4]`

```
num_found, df = solr_request(
    core='statistical-result',
    params={
        'q': 'marker_symbol:Gprc6a AND p_value:[0 TO 1e-4]',
        'fl': 'marker_symbol,top_level_mp_term_name,effect_size,p_value',
        'rows': 3   # Request the first three rows
    }
)
```

```
Your request:
https://www.ebi.ac.uk/mi/impc/solr/statistical-result/select?q=marker_symbol%3AGprc6a+AND+p_value%3A%5B0+TO+1e-4%5D&fl=marker_symbol%2Ctop_level_mp_term_name%2Ceffect_size%2Cp_value&rows=3

Number of found documents: 3
```

	effect_size	marker_symbol	p_value	top_level_mp_term_name
0	1.000000	Gprc6a	0.000000	[cardiovascular system phenotype, growth/size/...
1	2.307854	Gprc6a	0.000024	[immune system phenotype, endocrine/exocrine g...
2	1.253108	Gprc6a	0.000088	[hematopoietic system phenotype]

IMPC

International Mouse Phenotyping Consortium

@impc

Exercises Block B

How to Exclude Data

- Logical filters can be inverted by using the NOT operator, for example `NOT marker_symbol:Gprc6a`
- Tip: when you are combining multiple filters, it's best to use parentheses to make sure the operators are apply as you intend to.
- For example:
  `(field1:valueA OR field1:valueB) AND (NOT field2:value)`

```
num_found, df = solr_request(
    core='statistical-result',
    params={
        'q': 'NOT marker_symbol:Gprc6a', # Request Gprc6a records.
        'fl': 'marker_symbol,top_level_mp_term_name,effect_size,p_value',
        'rows': 3   # Request the first three rows
    }
)
```

Your request:
https://www.ebi.ac.uk/mi/impc/solr/statistical-result/select?q=NOT+marker_symbol%3AGprc6a&fl=marker_symbol%2Ctop_level_mp_term_name%2Ceffect_size%2Cp_value&rows=3

Number of found documents: 4069974

	marker_symbol	top_level_mp_term_name	effect_size	p_value
0	Lypla1	[skeleton phenotype]	NaN	NaN
1	Osbpl8	NaN	0.000000	1.000000
2	Raph1	[hearing/vestibular/ear phenotype]	0.154034	0.066499

How to Deal with Null Values

- For different fields it is possible encounter **null** values
  - When working in Python (NumPy/Pandas), they are represented as **NaN** in tables.
- A null value means that there is no data in this field for a given document, or that for this document this field is not defined / not used.
- You can filter out null values by applying this range filter that we have seen before: `field:[* TO *]`

```
num_found, df = solr_request(
    core='statistical-result',
    params={
        'q': 'p_value:[* TO *]', # Request p-value from 0 to 1e-4.
        'fl': 'marker_symbol,top_level_mp_term_name,effect_size,p_value',
        'rows': 3   # Request the first three rows
    }
)
```

```
Your request:
https://www.ebi.ac.uk/mi/impc/solr/statistical-result/select?q=p_value%3A%5B%2A+TO+%2A%5D&fl=marker_symbol%2Ctop_level_mp_term_name%2Ceffect_size%2Cp_value&rows=3

Number of found documents: 2068440
```

	effect_size	marker_symbol	p_value	top_level_mp_term_name
0	0.000000	Osbpl8	1.000000	NaN
1	0.154034	Raph1	0.066499	[hearing/vestibular/ear phenotype]
2	0.000929	Ccdc50	1.000000	[behavior/neurological phenotype]

Query Responsibly!

- Do not request all the data, if you don't need everything.
- Optimize query performance by requesting only the necessary data to minimize data transfer and processing overhead:
  - Use filters.
  - Combine different filters together.
  - Include only relevant fields.
- During the query construction, first request small data (3-5 documents) to make sure it works correctly, and only then request all data that you need.
- Use pagination to avoid overwhelming the system with large data requests.

Pagination and Downloading

- If you want to request all the data from one core, do NOT execute solr_request function. It will return just a set number of documents (by default: 10), instead of returning everything.

- Use `batch_request` function to download the data. It retrieves results in several chunks.

- `batch_size` parameter is a size of chunk, that will be used for getting the data.

```
solr_request(
    core="statistical-result",
    params={
        "q": "*:*",
    }
)
```

```
batch_request(
    core="statistical-result",
    params={
        "q": "*:*",
    },
    batch_size=1000
)
```

How to Download the Data?

- To download the data, execute the `batch_request` function. And then choose format of your preferences.

```
batch_request(
    core="statistical-result",
    params={
        "q": "*:*",
    },
    batch_size=1000
)
```

- You can download data in different formats: JSON, CSV.
- To save dataframe as JSON use this:

```
df.to_json("example_df.json", orient="records")
```

- We can save as CSV, but note that fine structure such as lists and nested data will be lost. To save dataframe:

```
df.to_csv("example_df.csv", index=False)
```

IMPC

International Mouse Phenotyping Consortium

@impc

Exercises Block C

Faceting

- In Solr, faceting is a feature that allows to categorize search results into different groups based on specified criteria.
- To make a faceting query, execute `facet_request` function to estimate counts types of the categories.
- Required parameters:
  - 'rows': '0'
  - 'facet': 'on'
  - 'facet.field':'field_name'
  - `facet.limit` — specifies the maximum number of facets for a field that should be returned.
  - `facet.mincount` — specifies the minimum counts required for a facet field to be included in the response.

```
num_found, df = facet_request(
    core='statistical-result',
    params={
        'q': '*:*',
        'rows': 0,
        'facet': 'on',
        'facet.field': 'zygosity',
        'facet.limit': 15,
        'facet.mincount': 1
    }
)
```

```
Your request:
https://www.ebi.ac.uk/mi/impc/solr/statistical-result/select?q=*:*&rows=0&facet=on&facet.field=zygosity&facet.limit=15&facet.mincount=1

Number of found documents: 4070676
```

	zygosity	count_per_category
0	homozygote	2520286
1	heterozygote	1435388
2	hemizygote	101488
3	wildtype	13514

MPI2

IMPC

International Mouse Phenotyping Consortium

@impc

Exercises Block D

Conclusion

- Solr is an API for accessing **IMPC data.**
  - It consists of multiple **cores**, which contain different data types.
  - You can request specific fields and filter results according to a combination of conditions.
- You can make requests directly in browser when exploring,
  but **wrapper functions** make the querying process easier – **use** them!
- Query **responsibly**: only request what you need and use pagination.