UPTAKE

Integrating Elasticsearch into Analytics Workflows

REV2
May 24, 2019

STEPHANIE KIRMER

@data_stephanie
github.com/skirmer/elastic_analytics

AGENDA

Introducing Elasticsearch

Libraries for R and Python

Querying and Filtering

Summarizing

Further Reading

Introducing Elasticsearch

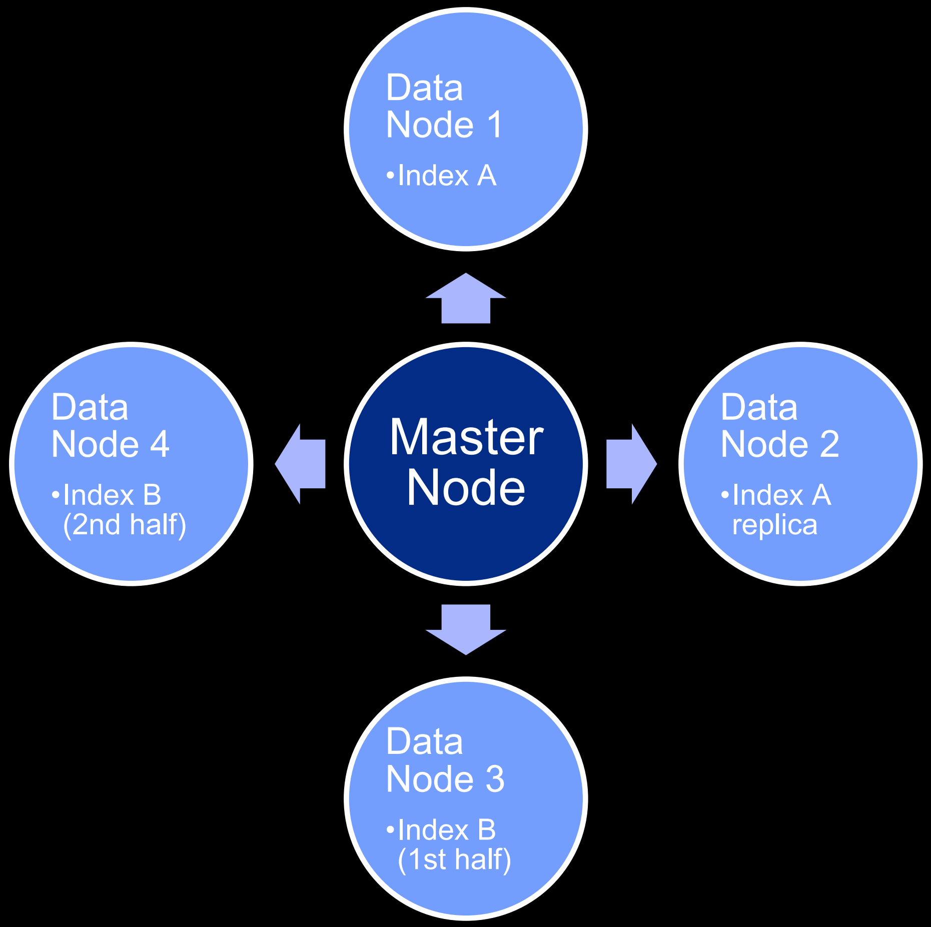
Overview

- Part of a family of data storage options called **NoSQL**
  - **NOT the same as tabular or SQL style data storage**
- Optimized for fast and **powerful searching**
- **Scales to “big data”**– but usable for small projects
- **Open source tool**

*Sometimes abbreviated “ES” – don’t get confused by this!

Visualizing Elasticsearch Storage

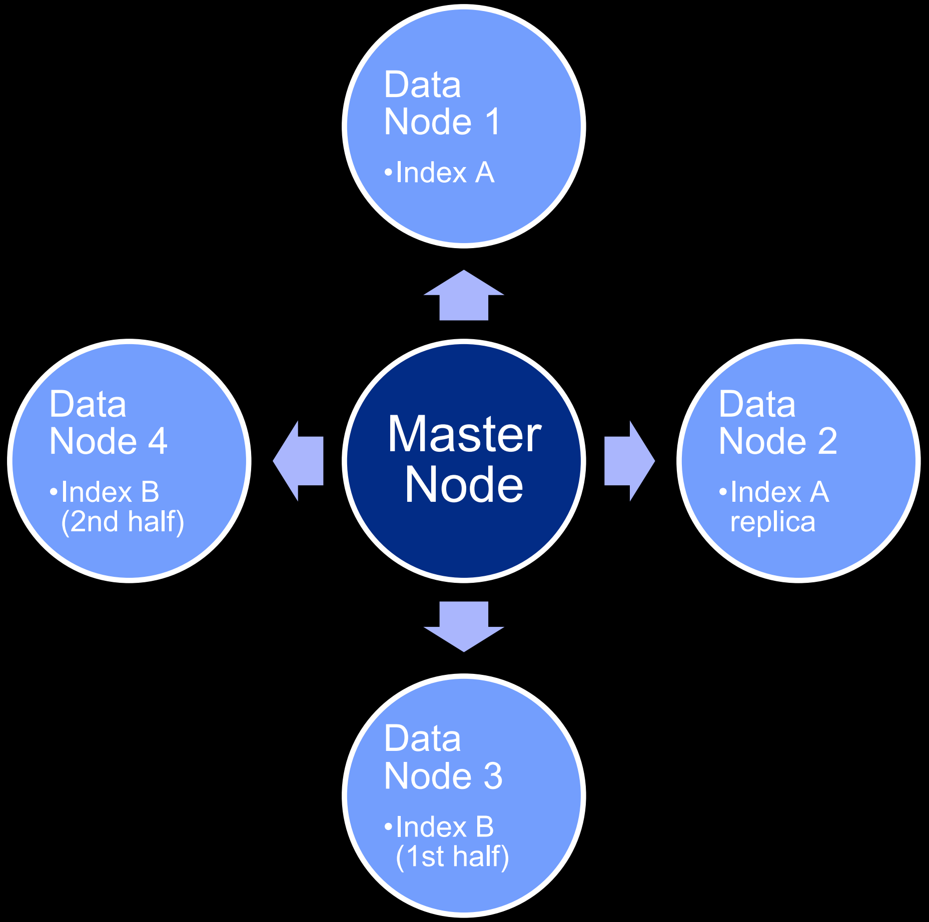
- **Cluster** = group of nodes
- **Master Node** = central brain
  - Home of search capabilities
- **Data Node** = where data is stored
  - Where master node looks for documents



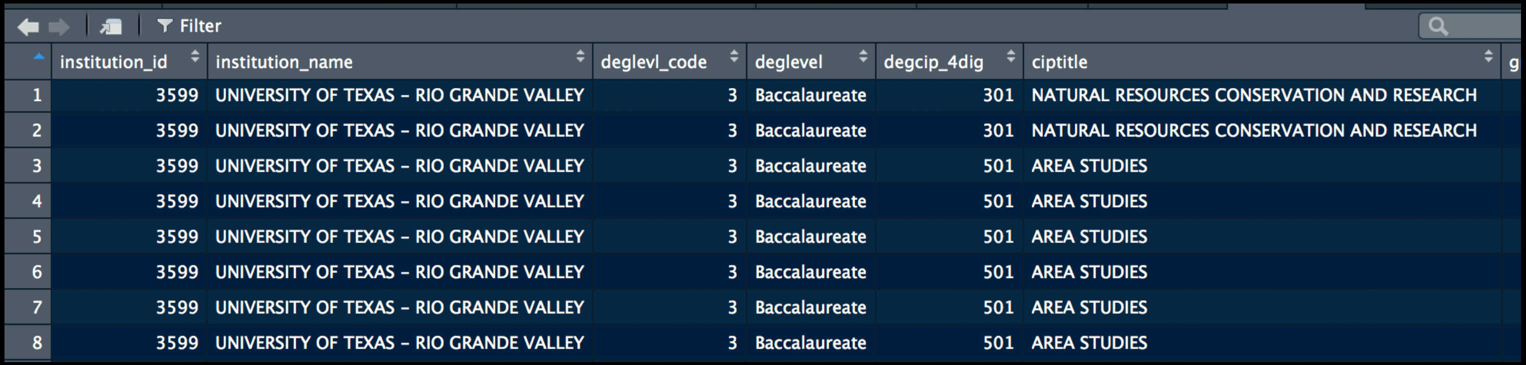
Visualizing Elasticsearch Storage

Data Architecture

- Data is divided into **indices**
- **Indices :**
  - Are user-defined groupings of data with some commonality
  - can live on one node, or
  - can be “sharded” and broken across nodes
  - can be duplicated on different nodes



Tabular Data vs Document-Based Data



	institution_id	institution_name	deglevl_code	deglevel	degcip_4dig	ciptitle	g
1	3599	UNIVERSITY OF TEXAS – RIO GRANDE VALLEY	3	Baccalaureate	301	NATURAL RESOURCES CONSERVATION AND RESEARCH	
2	3599	UNIVERSITY OF TEXAS – RIO GRANDE VALLEY	3	Baccalaureate	301	NATURAL RESOURCES CONSERVATION AND RESEARCH	
3	3599	UNIVERSITY OF TEXAS – RIO GRANDE VALLEY	3	Baccalaureate	501	AREA STUDIES	
4	3599	UNIVERSITY OF TEXAS – RIO GRANDE VALLEY	3	Baccalaureate	501	AREA STUDIES	
5	3599	UNIVERSITY OF TEXAS – RIO GRANDE VALLEY	3	Baccalaureate	501	AREA STUDIES	
6	3599	UNIVERSITY OF TEXAS – RIO GRANDE VALLEY	3	Baccalaureate	501	AREA STUDIES	
7	3599	UNIVERSITY OF TEXAS – RIO GRANDE VALLEY	3	Baccalaureate	501	AREA STUDIES	
8	3599	UNIVERSITY OF TEXAS – RIO GRANDE VALLEY	3	Baccalaureate	501	AREA STUDIES	

```
{"_index":"utexas",
"_type":"data",
"_id":"AWbU6WJiWX1fgzrfh4p1",
"_score":1.0,
"_source":
        {"institution_id":3599,
        "institution_name":"UNIVERSITY OF TEXAS - RIO GRANDE VALLEY",
        "deglevl_code":3,
        "deglevel":"Baccalaureate",
        "degcip_4dig":901,
        "ciptitle":"COMMUNICATION AND MEDIA STUDIES",
        "grad_cohort":2007,
        "grad_cohort_label":"2007-2009",
        "year_postgrad":1,
        "p25_earnings":26518.57,
        "p50_earnings":42166.31,
        "p75_earnings":50439,
        "system":"utsys",
        "cellcount":70}}
```

NoSQL is a different paradigm for thinking about data.

Why Use Elasticsearch?

Safe

- Copying your data easily and conveniently (via replicas) = if a node fails, your data is safe

Fast

- ES can search in parallel on multiple nodes and replicas, and find your data faster

Scalable

- Once you establish your ES database, you can add nodes and allow your database to grow

Open Source

- Free to use at small scale, substantial documentation, community support

Follow Along!

When you see this arrow, you can try it out yourself!

System Requirements:

- Docker installed and running
- Cloned: www.github.com/skirmer/elastic_analytics

Setup Steps (see the README to copy/paste)

- Get into the top level of the cloned repo
- At Terminal:

```
1. ./supporting_materials/setup_texas.sh 5.5
2. curl -X POST 'http://localhost:9200/utexas/_bulk' -H 'Content-Type:
   application/json' --data-binary @supporting_materials/ut_data.json
```

Start up R/Rstudio or Python as you prefer, and run further commands from there.

Libraries for R and Python

Choosing the right tool for your needs

Library Characteristics

Library	Returns	Query Language	Supports Authentication	R	Python
uptasticsearch	Tabular	Required	✖	●	●
elastic	JSON	Supported, not required	●	●	✖
elasticsearch-py	JSON	Supported, not required	●	✖	●

KEY CONSIDERATIONS

Secure Authentication

Do you need to securely log in?

Output Format

Do you mind handling JSON output?

Query Construction

Is writing query language a barrier?

R Options

Uptasticsearch (R)

```
test_up <- uptasticsearch::es_search(
   es_host = "http://localhost:9200"
   , es_index = "utexas"
   , query_body = query_string
   , max_hits = 10
   , size = 10)
```

```
query_string <- '{"query": {"match_all":{}}}'
```

Elastic (R)

```
elastic::connect(es_host =
"http://localhost:9200")

test_e <- elastic::Search(index =
"utexas”
    , body = query_string
    , size = 10
    , raw = TRUE)

test_e2 <-
jsonlite::fromJSON(test_e)$hits$hits
```

Non-Query Language Option:

```
test_e <- elastic::Search(
   index = "utexas"
    , q = "grad_cohort:*"
    , size = 10
    , raw = TRUE)
```

Python Options

Uptasticsearch (Py)

```
import json
import uptasticsearch

es_search(
   es_host="http://localhost:9200",
   query_body=query_string,
   max_hits = 10,
   es_index="utexas”
)
```

```
query_text = {"query": {"match_all": {}}}
query_string = '{"query": {"match_all": {}}}'
```

Elasticsearch-py (Py)

```
from elasticsearch import Elasticsearch

es = Elasticsearch(['http://localhost:9200'])

res = es.search(
   index="utexas",
   body= query_text
)

res['hits']['hits']
```

Non-Query Language Option (Elasticsearch_dsl):

```
from elasticsearch_dsl import Search

res2 = Search(using = es).query("match",
_index = 'utexas').execute()

res2.to_dict()['hits']['hits']
```

Querying and Filtering

Get what you need out of your database

Query Language Crash Course

Elastic Query DSL (domain specific language): a JSON-style syntax built to interact with ES databases.

Why use query language?

Consistency across interfaces and media

Precision and power in search, filtering, and aggregating – ES was built to work with this.

Downsides?

It’s sometimes hard to work with – idiosyncratic rules of syntax.

Identifying Available Fields

R:

```
uptasticsearch::get_fields(es_host = "http://localhost:9200",
es_indices = "utexas")
```

At Command Line:

```
curl http://localhost:9200/utexas/_mapping > fields.json
```

Constructing a Basic Query

Return all records :

```
{
“query”: { "match_all": { } }
}
```

```
query_text = {"query": {"match_all": {}}}
query_string = '{"query": {"match_all": {}}}'
```

Remember this from earlier!
All the queries we look at can be passed to R or Python this way.

Constructing a Basic Query

Return all records :

```
{
“query”: { "match_all": { } }
}
```

Constructing a Basic Query

Match one field :

```
{
“query”: { "match": { "ciptitle.raw": "AREA STUDIES” } }
}
```

Constructing a Basic Query

Match one field AND Greater Than one field :

```
{
“query”:
   { "bool" : {
      "must" : { "match": { "ciptitle.raw": "AREA STUDIES" } }
      , "must" : { "range" : { "cellcount" : { "gte" : 0 } } }
   } }
}
```

Constructing a Basic Query

Match two fields AND Greater Than one field :

```
{
“query”:
   { "bool" : {
      "must" : [ { "match": { "ciptitle.raw": "AREA STUDIES" } }
               , { "match": { "institution_id": "3599" } }
               , { "range" : { "cellcount" : { "gte" : 0 } } } ]
   } }
}
```

Some Other Querying Options

match_phrase

Match a set of words all together.

exists

Supply a field, returns documents that have at least one non-null value in the original field.

wildcard

Pass a string with a wildcard anywhere – but be careful, it can be a slow search!

filter

Just like “must” except without scoring – we’ll talk about this in a moment.

must_not

Instead of “must” – use to omit records with a word or phrase.

This is just a small sample- ES query language offers many very powerful search options!

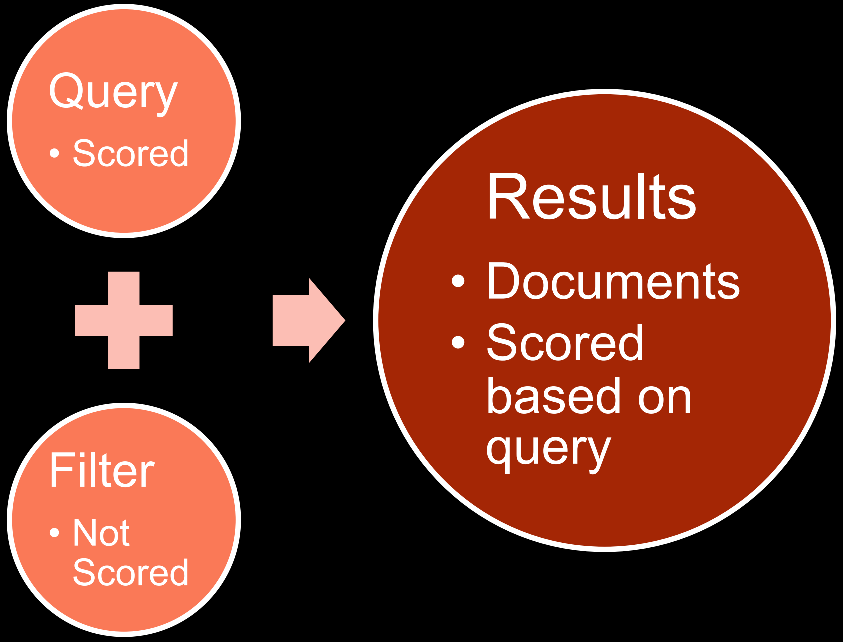
Query vs Filter

Scoring Results

ES queries can provide a **numeric score** indicating how well the document meets the criteria given.

When you use **“query”** at the beginning of the query, you get a score returned alongside your results.

When you use **“filter”,** Elasticsearch does not score the results on the given criteria.



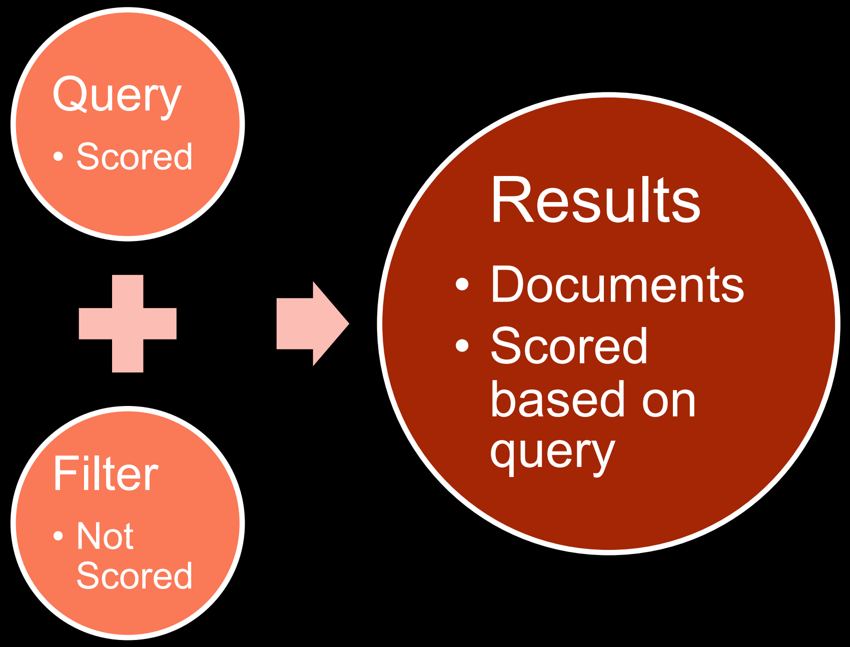
Query vs Filter

Example

Do we want scores returned for this search? Yes.

```
{ "query":
   { "bool": {
      "must": [
         { "match": { "ciptitle.raw": "AREA STUDIES” } }
         , { "match": { ″deglevel": ″Baccalaureate" } }
      ] ,
      "filter": [
         { ″match": {"institution_id": "3599" } }
         , { "range": { "cellcount" : { "gte" : 0 } } }
      ]
   } }
}
```

Do we want the scores to include these criteria? NO.



Query
- Scored

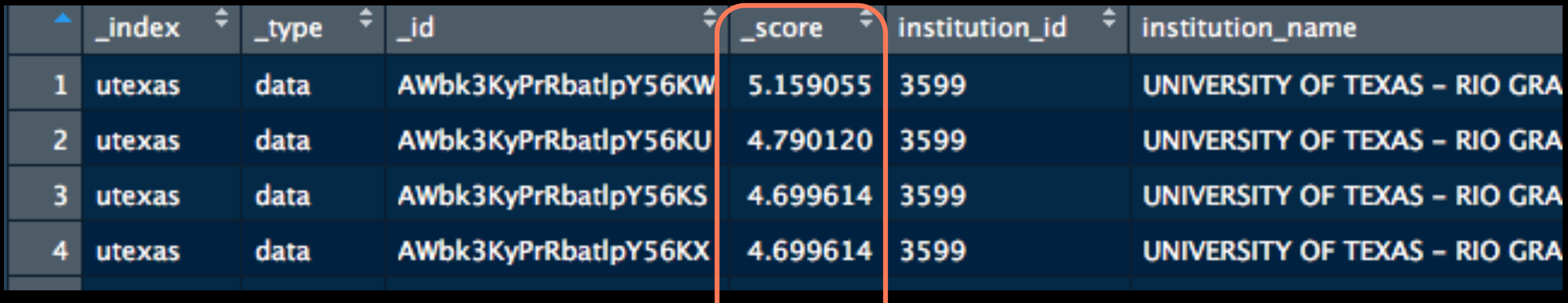
Filter
- Not Scored

Results
- Documents
- Scored based on query

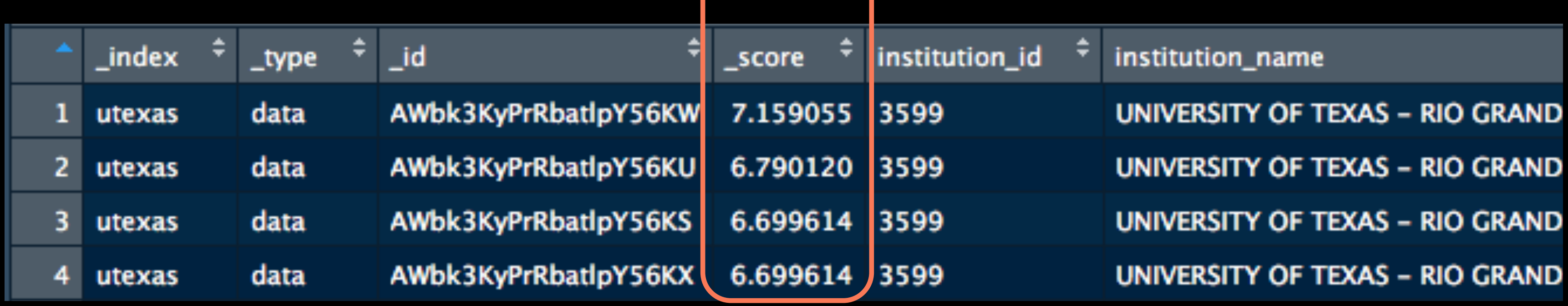
Query vs Filter

Example

Same query, first with one criterion scored (2 in filter) and second with all 3 criteria scored.



	_index	_type	_id	_score	institution_id	institution_name
1	utexas	data	AWbk3KyPrRbatlpY56KW	5.159055	3599	UNIVERSITY OF TEXAS – RIO GRA
2	utexas	data	AWbk3KyPrRbatlpY56KU	4.790120	3599	UNIVERSITY OF TEXAS – RIO GRA
3	utexas	data	AWbk3KyPrRbatlpY56KS	4.699614	3599	UNIVERSITY OF TEXAS – RIO GRA
4	utexas	data	AWbk3KyPrRbatlpY56KX	4.699614	3599	UNIVERSITY OF TEXAS – RIO GRA



	_index	_type	_id	_score	institution_id	institution_name
1	utexas	data	AWbk3KyPrRbatlpY56KW	7.159055	3599	UNIVERSITY OF TEXAS – RIO GRAND
2	utexas	data	AWbk3KyPrRbatlpY56KU	6.790120	3599	UNIVERSITY OF TEXAS – RIO GRAND
3	utexas	data	AWbk3KyPrRbatlpY56KS	6.699614	3599	UNIVERSITY OF TEXAS – RIO GRAND
4	utexas	data	AWbk3KyPrRbatlpY56KX	6.699614	3599	UNIVERSITY OF TEXAS – RIO GRAND

Summarizing

Get fancier with your searching!

Summarizing in Query

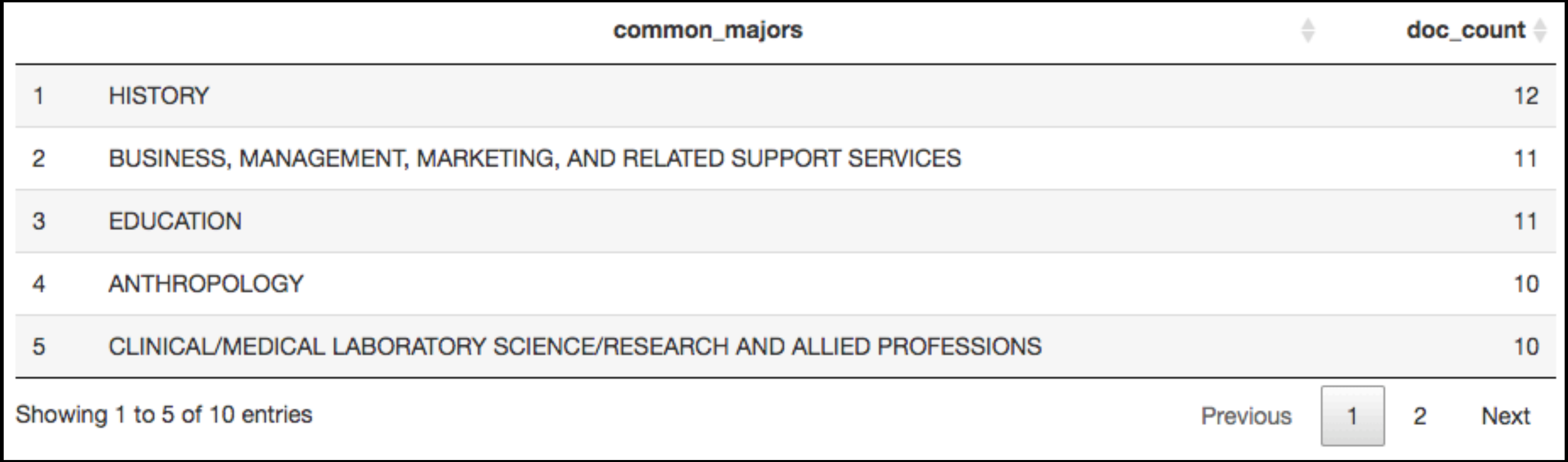
Match one field, Summarize one field:

```
{
"query": {"match": {"institution_id": "3599" }} ,
   "aggs" : {
      "common_majors" : {
         "terms" : {
            "field" : "ciptitle.raw"
         }
      }
   }
}
```

Create a new field called common_majors, which sums up the unique values in ciptitle.raw for this specific search.

Summarizing in Query

Produces:



	common_majors	doc_count
1	HISTORY	12
2	BUSINESS, MANAGEMENT, MARKETING, AND RELATED SUPPORT SERVICES	11
3	EDUCATION	11
4	ANTHROPOLOGY	10
5	CLINICAL/MEDICAL LABORATORY SCIENCE/RESEARCH AND ALLIED PROFESSIONS	10

Showing 1 to 5 of 10 entries

If you’re following along, clean up

- At Terminal: `./supporting_materials/cleanup_local.sh`

This shuts down the docker container, destroying our demo database – but you can create it again just by going back to the beginning.

Further Reading

Explore More about Elasticsearch!

github.com/skirmer/elastic_analytics
www.stephaniekirmer.com
@data_stephanie

ES Query Language

- http://elasticsearch-cheatsheet.jolicode.com/
- https://elasticsearch-dsl.readthedocs.io/en/latest/search_dsl.html
- https://www.elastic.co/guide/en/elasticsearch/reference/current/_introducing_the_query_language.html
- https://www.elastic.co/guide/en/elasticsearch/reference/6.4/query-dsl-bool-query.html
- https://www.elastic.co/guide/en/elasticsearch/reference/6.4/query-filter-context.html

Library Docs

- https://elasticsearch-py.readthedocs.io/en/master/index.html
- https://github.com/ropensci/elastic
- https://github.com/UptakeOpenSource/uptasticsearch – Make contributions, the packages are always improving!

Data Credit:

The data being used in this tutorial is from data.world, and comes out of the hard work done by Annie Millerbernd of the San Antonio Express-News. You can learn more about it and see the original dataset here: https://data.world/amillerbernd/ut-system-post-grad-earnings