How to use LexMapr

For people who have
little to no experience with command line

and

For macOS and Linux users only

Outline

Open up your terminal

macOs:

 cmd + *space* to open spotlight search

 type terminal

 hit *return*

Linux:

 ctrl + *alt* + *t*

Terminal tutorial

The terminal opens in your Home (also known as ~) directory

Type the following command into your terminal:

```
ls
```

This will print out all the sub-directories and files in your home directory

Terminal tutorial

You can change terminal directories with the `cd` command:

    `cd Desktop`   Navigate to your Desktop subdirectory

    `ls`   View the contents of your Desktop

    `cd ..`   Navigate to the parent directory

    `ls`   View the contents of your Home again

If you ever get lost, you can return Home with `cd ~`

Do you have Conda?

In your terminal, type:

```
conda -V
```

If you have Conda, the terminal will tell you your Conda version number

If you do not have Conda, the terminal will tell you the `conda` command was not found

Install Conda

Go to https://docs.conda.io/en/latest/miniconda.html

Install the appropriate **64-bit bash installer** for **Python 3.7**

In your terminal, navigate to the folder containing the installer

Assuming this was the Downloads folder:

```
cd ~

cd Downloads
```

Install Conda

In this folder, we run the installer through a terminal command

macOS:

```
bash Miniconda3-latest-MacOSX-x86_64.sh
```

Linux:

```
bash Miniconda3-latest-Linux-x86_64.sh
```

Follow the prompts to completion

Bioconda

Bioconda is a channel that allows users to easily install various bioinformatic packages, including LexMapr

Run the following terminal commands to set up Bioconda:

```
conda config --add channels default

conda config --add channels bioconda

conda config --add channels conda-forge
```

RUN THESE COMMANDS IN ORDER

Install LexMapr into a Conda environment

Enter the following command from any directory:

```
conda create -n LexMapr python=3.6 lexmapr
```

Follow the prompts to completion

This creates a Conda environment called "LexMapr", that contains a Bioconda installation of LexMapr

Running LexMapr

Whenever you want to run LexMapr, all you have to do is:

Start the terminal

Activate your LexMapr Conda environment:

```
conda activate LexMapr
```

Run LexMapr commands

```
lexmapr --help
```

LexMapr input files

LexMapr takes input csv files with the following format:

SampleId,Sample

1,Chicken Breast

2,Baked Potato

3,Canned Corn

4,Frozen Yogurt

5,Apple Pie

LexMapr input files

Open up Microsoft Excel, Apple Numbers or LibreOffice Calc

Create a spreadsheet like so:

	A	B
1	SampleId	Sample
2	1	Chicken Breast
3	2	Baked Potato
4	3	Canned Corn
5	4	Frozen Yogurt
6	5	Apple Pie

Save as "`small_simple.csv`"

Running LexMapr

In the terminal, `cd` to the folder containing `small_simple.csv`

With the LexMapr Conda environment activated, try:

```
lexmapr small_simple.csv
```

Taking a while?

The first run will be slow

LexMapr caches a lookup table on the first run

LexMapr uses this table to make subsequent runs much faster

Running LexMapr

Output:

```
Sample_Id        Sample_Desc     Cleaned_Sample   Matched_Components
small_simple1    Chicken Breast  chicken breast   ['breast:uberon_0000310']
small_simple2    Baked Potato    baked potato     []
small_simple3    Canned Corn     canned corn      []
small_simple4    Frozen Yogurt   frozen yogurt    ['frozen:pato_0001985']
small_simple5    Apple Pie       apple pie        []
```

Running LexMapr

You can specify an "output" tsv file for your results to be written to

Saves your results ✅

tsv files can be opened in Excel, Numbers or Calc for easier viewing ✅

Try:

```
lexmapr small_simple.csv -o small_simple_output.tsv
```

Running LexMapr

`small_simple_output.tsv` was created in the directory

Open it up in Excel, Numbers or Calc for easy viewing of your output, separated into columns

A problem

Output:

```
Sample_Id      Sample_Desc     Cleaned_Sample  Matched_Components
small_simple1  Chicken Breast  chicken breast  ['breast:uberon_0000310']
small_simple2  Baked Potato    baked potato    []  ⟵
small_simple3  Canned Corn     canned corn     []  ⟵
small_simple4  Frozen Yogurt   frozen yogurt   ['frozen:pato_0001985']
small_simple5  Apple Pie       apple pie       []  ⟵
```

Missing matches

Mapping samples against online ontologies

Unless told otherwise, LexMapr maps samples against a highly limited collection of pre-defined resources → missed matches

You should instead map samples against terms of your own choosing

You can do this by supplying LexMapr with a **config file**

Config files

When running LexMapr, use a config file that specifies the online ontology terms you want to map your samples against

e.g., we could create a config file that tells LexMapr to map against terms belonging to the Food ontology "Foodon", and use it whenever we run LexMapr on a collection of food samples

Config files can specify terms from:

Entire ontologies

Subsets of ontologies

Multiple ontologies

You re-use config files for multiple runs, or create new config files when you want to switch things up, but you can only provide one config file each time you run LexMapr

21

Config file format

```
[

    {"ONTOLOGY_IRI_1":"ROOT_IRI_1"},

    {"ONTOLOGY_IRI_2":"ROOT_IRI_2"},

    ...,

]
```

Config file format

```
[

    {"ONTOLOGY_IRI_1":"ROOT_IRI_1"},

    {"ONTOLOGY_IRI_2":"ROOT_IRI_2"},

    ...,

]
```

IRI of ontology you want to map your samples against

e.g., "http://purl.obolibrary.org/obo/foodon.owl"

Ontology refresher

```
                    ┌──────────┐
                    │ Livestock │
                    └──────────┘
            ┌──────────┼──────────┐
            ↓          ↓          ↓
      ┌─────────┐  ┌───────┐  ┌────────┐
      │ Poultry │  │ Swine │  │ Cattle │
      └─────────┘  └───────┘  └────────┘
        ┌────┴────┐
        ↓         ↓
   ┌────────┐ ┌─────────┐
   │ Turkey │ │ Chicken │
   └────────┘ └─────────┘
```

Ontology refresher

Livestock

Poultry

Swine

Cattle

Turkey

Chicken
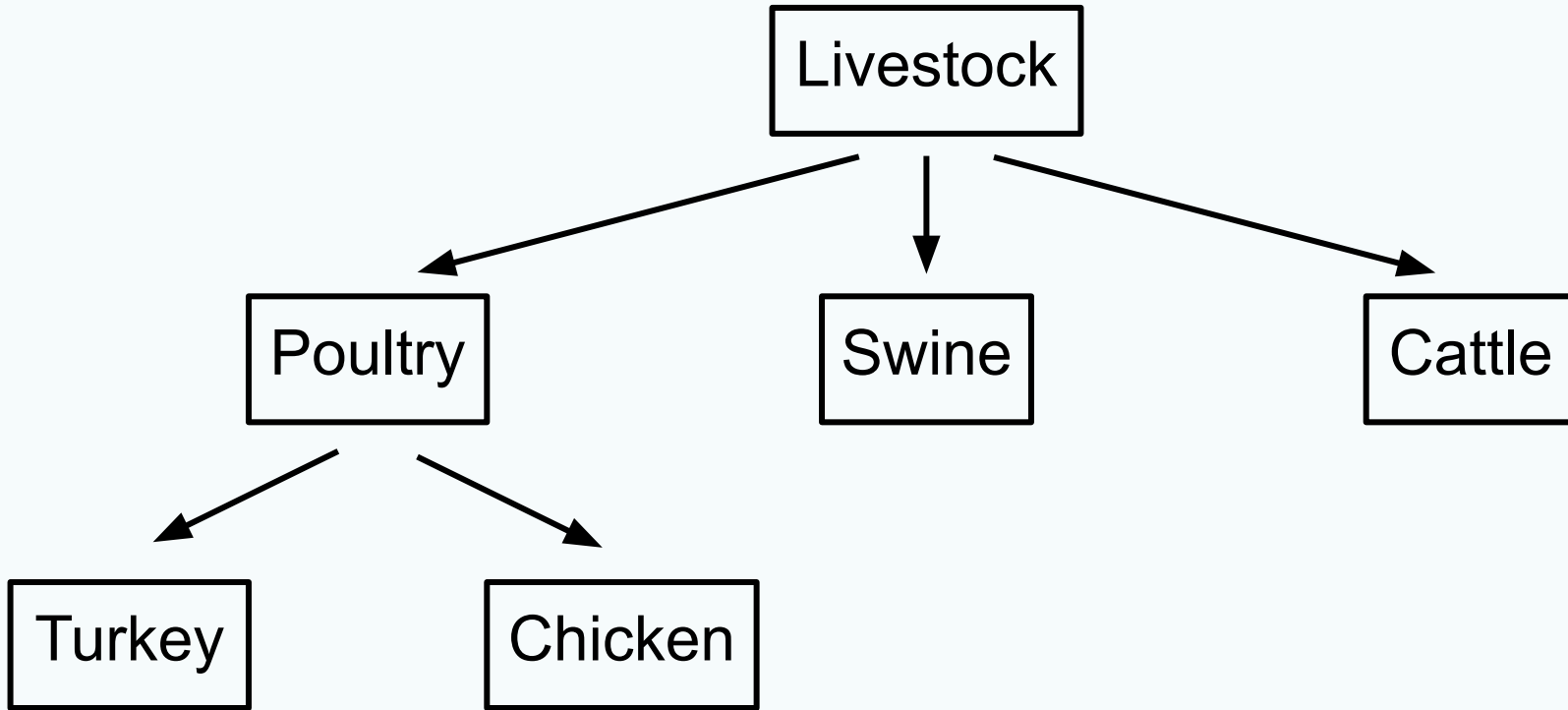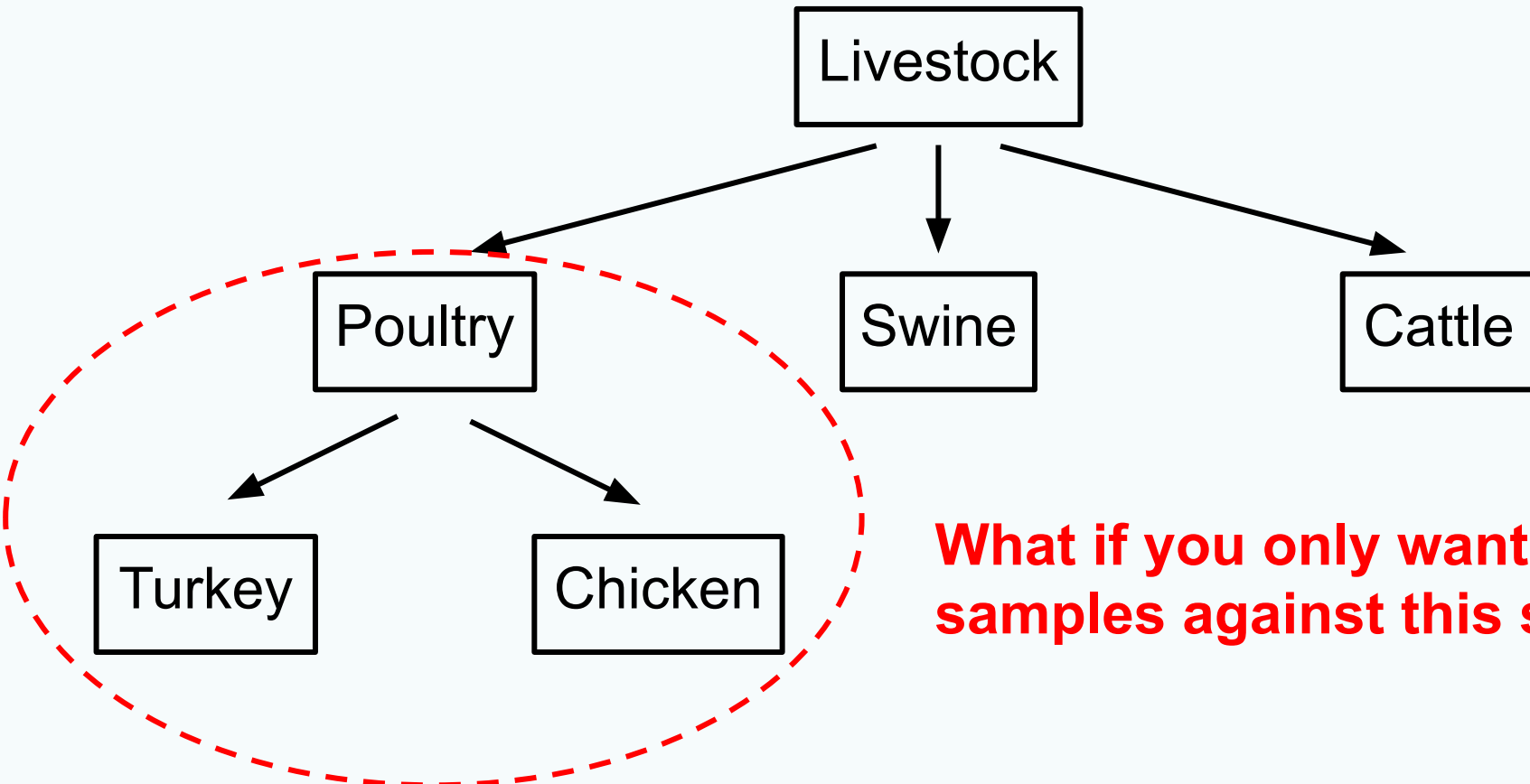
What if you only want to match samples against this sub-tree?

Config file format

```
[

    {"ONTOLOGY_IRI_1":"ROOT_IRI_1"},

    {"ONTOLOGY_IRI_2":"ROOT_IRI_2"},

    ...,

]
```

IRI of the "top" of the sub-tree, within the specified ontology

e.g., the IRI for "poultry or game bird" within FOODON:
"http://purl.obolibrary.org/obo/FOODON_03411563"

Create a config file

We're going to map our `small_simple.csv` file against everything under "plant food product" in FOODON

Create a config file

We need IRI values

Go to https://www.ebi.ac.uk/ols/ontologies

Search up "foodon"

Jump to the ontology page when located

Create a config file

Under the box titled "Ontology info" on the Foodon ontology page, note the Ontology IRI:

Create a config file

Under the box titled "Ontology info" on the Foodon ontology page, note the Ontology IRI:

http://purl.obolibrary.org/obo/foodon.owl

This is your **ONTOLOGY_IRI**

Create a config file

You can then use the "browse terms" tab, or the search bar, to locate the OLS page for "plant food product"

Create a config file

The IRI for "plant food product" is under the title of its OLS page:

plant food product

http://purl.obolibrary.org/obo/FOODON_00001015

This food product type includes food products which are derived from or produced by a plant.

Tree view **Term history**

- entity
  - continuant
    - independent continuant
      - material entity
        - fiat object part
          - environmental material
            - food material
              - foodon product type
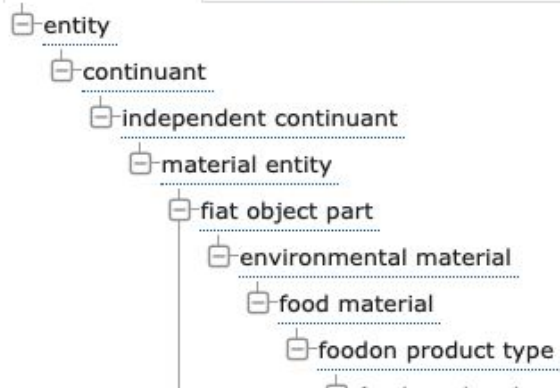
Graph view

Reset tree

Show all siblings

Term info

has curation status
  http://purl.obolibrary.org/obo/IAO_0000428

Term relations

Equivalent to:
- food product by organism and *derives from* some plant

Search FOODON

Create a config file

The IRI for "plant food product" is under the title of its OLS page:

http://purl.obolibrary.org/obo/FOODON_00001015

This is your **ROOT_IRI**

Create a config file

Open a text editor, and write the following:

```
[
{"http://purl.obolibrary.org/obo/foodon.owl": "http://purl.obolibrary.org/obo/FOODON_00001015"}
]
```

Save the file as `small_simple_config.json`, in the same folder as `small_simple.csv`

Notice that we're using the IRI values we just searched for

So you now have a config file that tells LexMapr to map samples against everything under "plant food product" in Foodon

Create a config file

Note for macOs users:

TextEdit may not allow you to save JSON files

You can use another text editor, or google for an online JSON editor

Mapping samples against online ontologies

In terminal:

Navigate to the folder with `small_simple.csv` and `small_simple_config.json`

With your LexMapr conda environment activated, try:

```
lexmapr small_simple.csv -c small_simple_config.json -o small_simple_output.tsv
```

This may take a while, as lexmapr creates a lookup table for each config file (to speed up subsequent runs)

Mapping samples against online ontologies

Output:

```
Sample_Id        Sample_Desc      Cleaned_Sample   Matched_Components
[small_simple1   Chicken Breast   chicken breast   ['breast:uberon_0000310']
[small_simple2   Baked Potato     baked potato     ['potato (whole, baked):foodon_03302196']
small_simple3    Canned Corn      canned corn      ['corn (canned):foodon_03302665']
small_simple4    Frozen Yogurt    frozen yogurt    ['frozen:pato_0001985']
small_simple5    Apple Pie        apple pie        ['apple pie:foodon_00002475']
```

That's better!

But `small_simple1` and `small_simple4` could be better.

Create a config file

Fetch the IRI values for "meat food product" and "dairy food product" under Foodon the same way, and extend your config file:

```
[
{"http://purl.obolibrary.org/obo/foodon.owl": "http://purl.obolibrary.org/obo/FOODON_00001015"},
{"http://purl.obolibrary.org/obo/foodon.owl": "http://purl.obolibrary.org/obo/FOODON_00001006"},
{"http://purl.obolibrary.org/obo/foodon.owl": "http://purl.obolibrary.org/obo/FOODON_00001256"}
]
```

Note the commas

Save the file

Your config file now tells LexMapr to map samples against everything under "plant food product", "meat food product" and "dairy food product" in Foodon

Mapping samples against online ontologies

Run:

```
lexmapr small_simple.csv -c small_simple_config.json -o small_simple_output.tsv --no-cache
```

The `--no-cache` flag is needed whenever you update a config file, so LexMapr can update the corresponding lookup table

Output:

```
Sample_Id      Sample_Desc     Cleaned_Sample   Matched_Components
small_simple1  Chicken Breast  chicken breast   ['chicken breast:foodon_00002703']
small_simple2  Baked Potato    baked potato     ['potato (whole, baked):foodon_03302196']
small_simple3  Canned Corn     canned corn      ['corn (canned):foodon_03302665']
small_simple4  Frozen Yogurt   frozen yogurt    ['frozen yogurt:foodon_03307445']
small_simple5  Apple Pie       apple pie        ['apple pie:foodon_00002475']
```

Perfect

Third-party classification schemes

Try:

```
lexmapr small_simple.csv -c small_simple_config.json -o small_simple_output.tsv -b
```

The -b flag classifies all samples into broad categories:
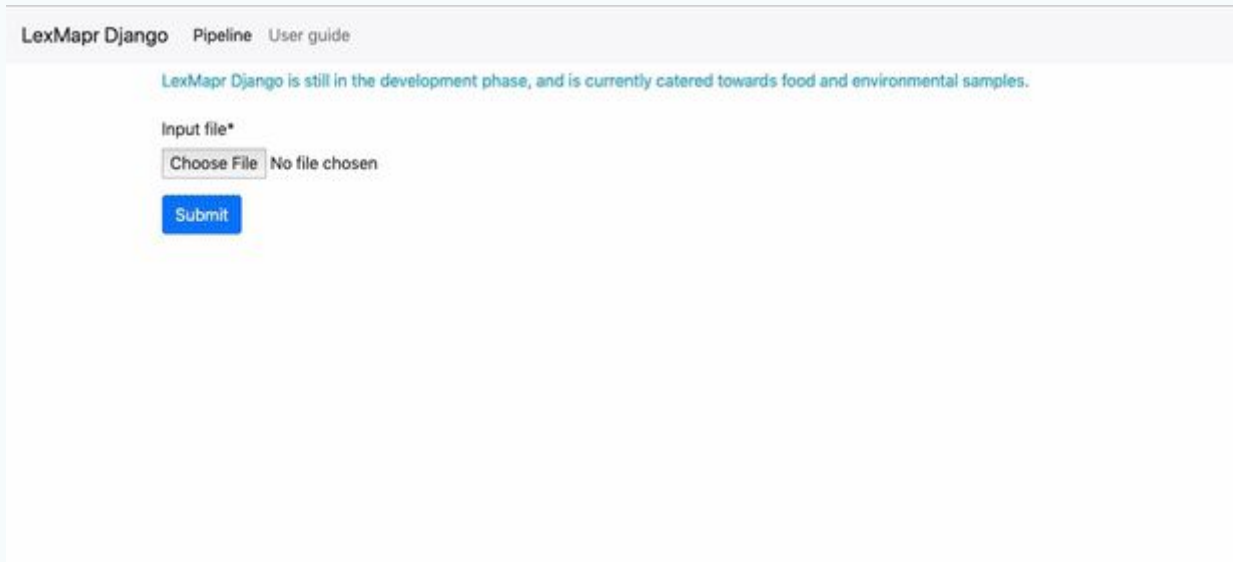
```
Sample_Id       Sample_Desc     Cleaned_Sample  Matched_Components        Third Party Classification
small_simple1   Chicken Breast  chicken breast  ['chicken breast:foodon_00002703']       ['chicken']
small_simple2   Baked Potato    baked potato    ['potato (whole, baked):foodon_03302196']       ['root/underground (tubers)']
small_simple3   Canned Corn     canned corn     ['corn (canned):foodon_03302665']       ['seeded vegetables (other)']
small_simple4   Frozen Yogurt   frozen yogurt   ['frozen yogurt:foodon_03307445']       ['dairy']
small_simple5   Apple Pie       apple pie       ['apple pie:foodon_00002475']   ['pome fruit']
```

The categories were selected by a third-party, and follow their highly specific classification scheme, which is geared towards food

In the future, LexMapr will allow you to introduce classification schemes of your own

40

LexMapr Django

https://watson.bccdc.med.ubc.ca/lexmapr/

LexMapr Django   Pipeline   User guide

LexMapr Django is still in the development phase, and is currently catered towards food and environmental samples.

Input file*

Choose File   No file chosen

Submit

Django-powered web interface, that allows you to run LexMapr from your browser

LexMapr Django

Not as powerful as the command-line tool yet

    Built-in config file specialized for food and environment samples

        Cannot specify your own config file

    Highly specific third-party classification scheme always used

Will become more powerful in the future