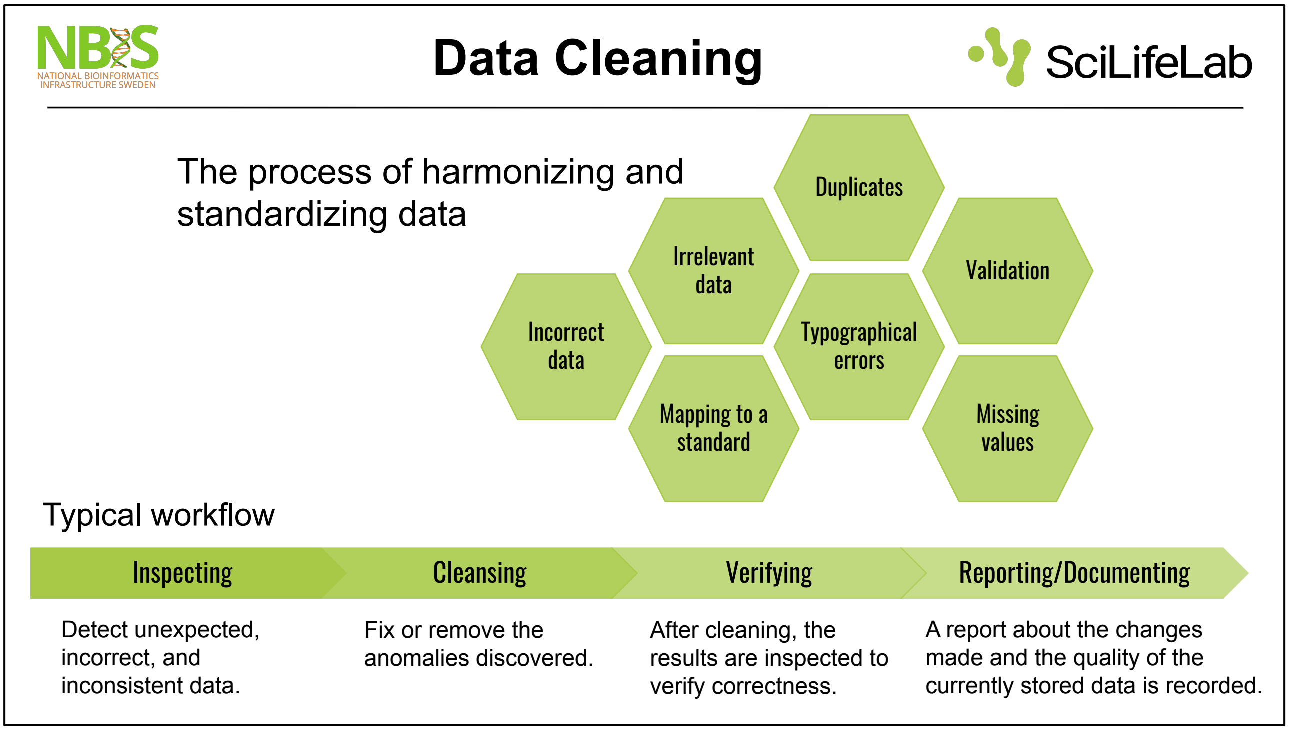
NATIONAL BIOINFORMATICS INFRASTRUCTURE SWEDEN

Cleaning Data with OpenRefine

Introduction to Data Management Practices course

NBIS DM Team
data@nbis.se

https://nbisweden.github.io/module-openrefine-dm-practices/index.html



Data Cleaning

The process of harmonizing and standardizing data

Duplicates

Irrelevant data

Validation

Incorrect data

Typographical errors

Mapping to a standard

Missing values

Typical workflow

Inspecting	Cleansing	Verifying	Reporting/Documenting
Detect unexpected, incorrect, and inconsistent data.	Fix or remove the anomalies discovered.	After cleaning, the results are inspected to verify correctness.	A report about the changes made and the quality of the currently stored data is recorded.

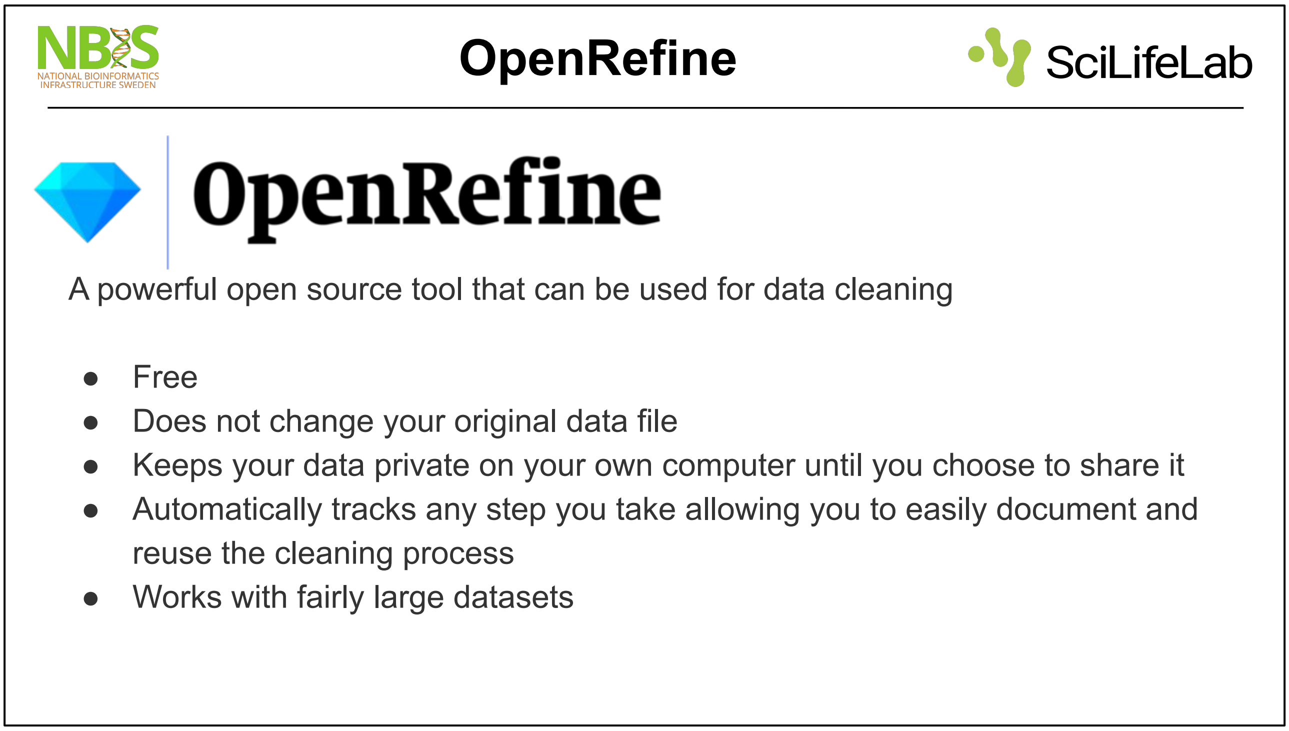
Data can often be very messy, in previous lessons we have seen examples of this. Different people or projects can use different terms for the same concept for example. As part of the pre-processing of data before you start your analysis or before you share it with others you will most likely need to do some data cleaning.
Data cleaning is the process of harmonizing and standardizing data.

What does data cleaning entail? For example:

- Removing irrelevant data
- Identify and handle incorrect data
- Correcting Typos
- Removing/consolidating duplicates
- Mapping values against a standard
- Handle missing data and missing values
- Validation of data

Data cleaning is often an iterative process. A common workflow for data cleaning could be:
Inspection: Detect unexpected, incorrect, and inconsistent data.
Cleaning: Fix or remove the anomalies discovered.
Verifying: After cleaning, the results are inspected to verify correctness.
Reporting/Documentation: A report about the changes made and the quality of the currently stored data is recorded.

We have already seen some examples of data cleaning in Excel but today we will introduce a powerful tool for data cleaning called OpenRefine.



OpenRefine

OpenRefine

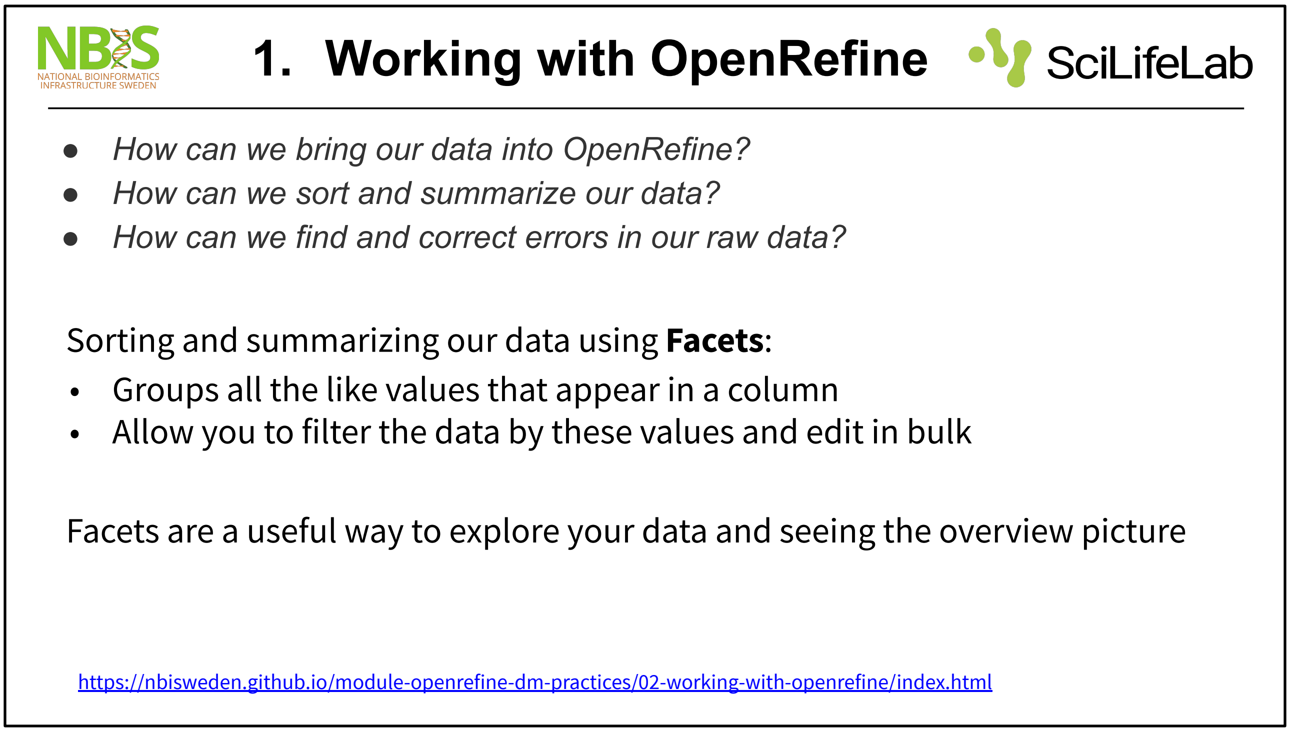
A powerful open source tool that can be used for data cleaning

- Free
- Does not change your original data file
- Keeps your data private on your own computer until you choose to share it
- Automatically tracks any step you take allowing you to easily document and reuse the cleaning process
- Works with fairly large datasets

What is so great with OpenRefine?

- Free and Open source (source on GitHub), with a large growing community, from novice to expert, ready to help.
- Does not change your original data file
- OpenRefine always keeps your data private on your own computer until you choose to share it. It works by running a small server on your computer and using your web browser to interact with it, but your private data never leaves your computer unless you want it to.
- Automatically tracks any step you take allowing you to document and reuse the cleaning process
- Works with large-ish datasets (100,000 rows). Can adjust memory allocation to accommodate larger datasets.

Most of this lesson, you will be working in OpenRefine alongside me or independently but I will sometimes come back to the slides to introduce concepts.



1. Working with OpenRefine

- *How can we bring our data into OpenRefine?*
- *How can we sort and summarize our data?*
- *How can we find and correct errors in our raw data?*

Sorting and summarizing our data using **Facets**:
- Groups all the like values that appear in a column
- Allow you to filter the data by these values and edit in bulk

Facets are a useful way to explore your data and seeing the overview picture

https://nbisweden.github.io/module-openrefine-dm-practices/02-working-with-openrefine/index.html

First episode:
- How can we bring our data into OpenRefine?
- How can we sort and summarize our data?
- How can we find and correct errors in our raw data?

The first step in the data cleaning process is to inspect or look at the data to get an overview. We are going to create a new project in OpenRefine, load our data file and then start off by using Facets to explore our data.

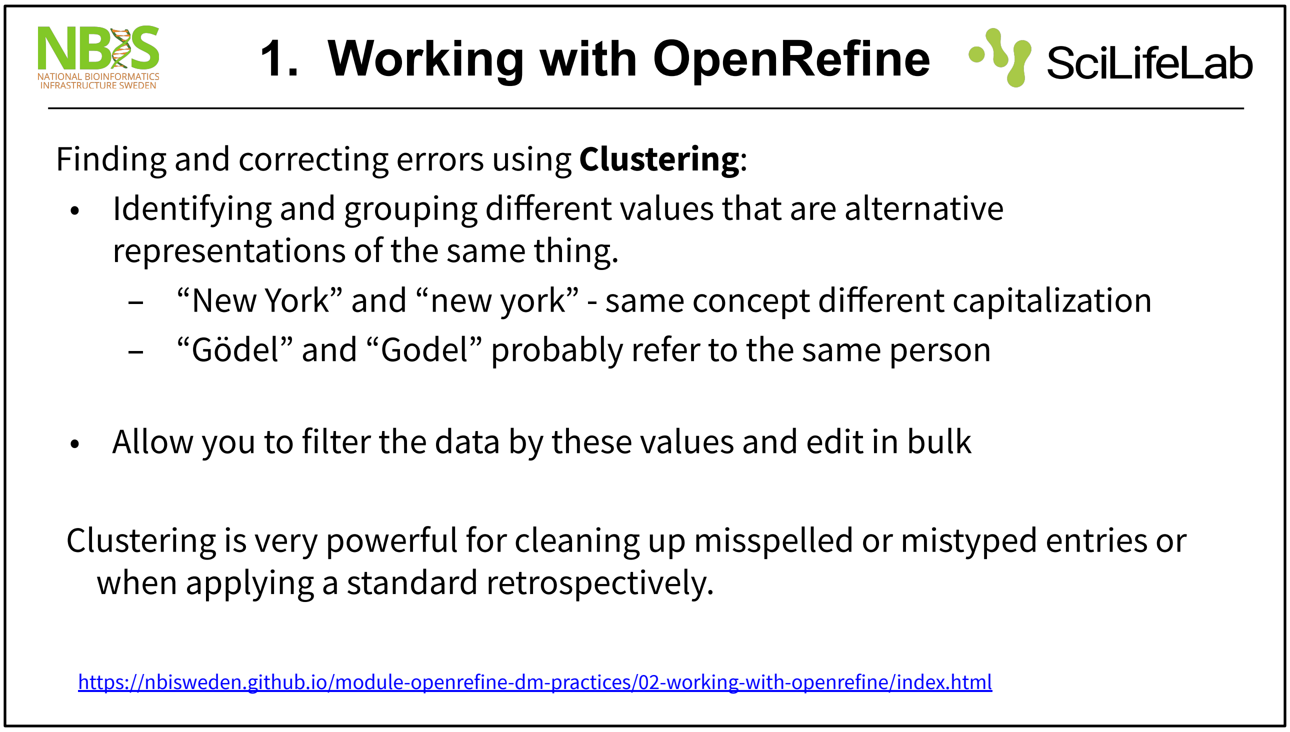
Facets
A Face' groups all the like values that appear in a column, and then allow you to filter the data by these values and edit values across many records at the same time. It is one of the most useful features of OpenRefine and can help both to get an overview of the data in a project as well as helping you bring more consistency to the data. OpenRefine supports faceted browsing as a mechanism for

- seeing a big picture of your data, and
- filtering down to just the subset of rows that you want to change in bulk.

Interactive steps

1. Open OpenRefine, check that everyone is up and running
2. Demo: Creating a new project, click and unclick header row for example. Check that everyone has loaded the data
3. Give a quick tour of the drop down menu and showing more mores of the data etc

1. Demo Facets, After step 4, allow some time for discussion
2. Exercise 1.1 on Facets
3. Mention other types of facets and that more info can be found in the teaching material
4. Demo exercise 1.2 (remember to remove timeline facet)



1. Working with OpenRefine

Finding and correcting errors using **Clustering**:

- Identifying and grouping different values that are alternative representations of the same thing.
  - "New York" and "new york" - same concept different capitalization
  - "Gödel" and "Godel" probably refer to the same person

- Allow you to filter the data by these values and edit in bulk

Clustering is very powerful for cleaning up misspelled or mistyped entries or when applying a standard retrospectively.

https://nbisweden.github.io/module-openrefine-dm-practices/02-working-with-openrefine/index.html

First episode:

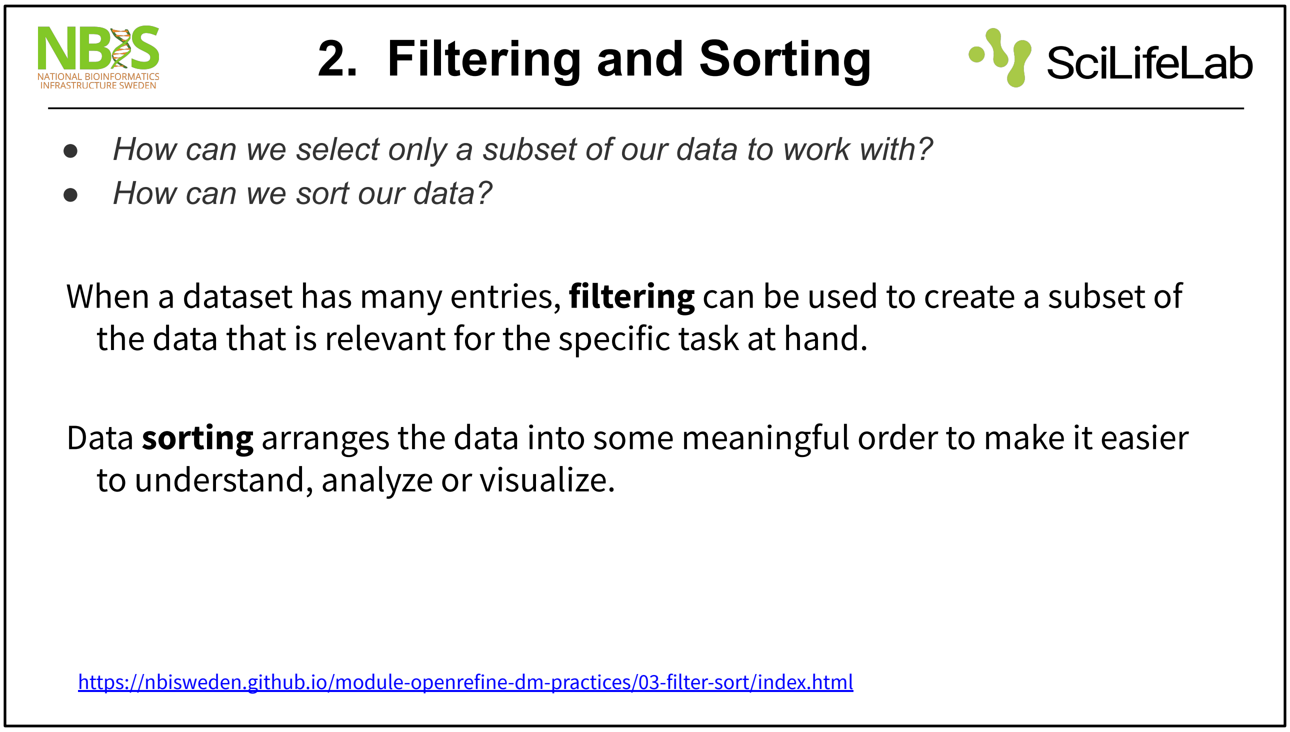
- How can we bring our data into OpenRefine?
- How can we sort and summarize our data?
- How can we find and correct errors in our raw data?

Now we are going to look at different ways of finding and correcting errors in our raw data. We will start by clustering.

Clustering
In OpenRefine, clustering means "finding groups of different values that might be alternative representations of the same thing". For example, the two strings `New York` and `new york` are very likely to refer to the same concept and just have capitalization differences. Likewise, `Gödel` and `Godel` probably refer to the same person. Clustering is a very powerful tool for cleaning datasets which contain misspelled or mistyped entries. OpenRefine has several clustering algorithms built in. Experiment with them, and learn more about these algorithms and how they work.

Interactive steps
1. Make sure everyone undid the transform to date step
2. Demo Clustering step 1-5 Ask what standard for sex they decided on during the metadata lesson. Correct answer: male, female unknown
3. Ask the learners to try different settings for a few minutes. Did anyone find any

1. more clustering suggestions?
2. Do step 7 together and verify that everyone has 3 clusters.
3. Mention that the technical details of how the different clustering algorithms work can be found in the training material.
4. Demo Split
5. Students do the Exercise 1.3
6. Demo Whitespaces



2. Filtering and Sorting

- *How can we select only a subset of our data to work with?*
- *How can we sort our data?*

When a dataset has many entries, **filtering** can be used to create a subset of the data that is relevant for the specific task at hand.

Data **sorting** arranges the data into some meaningful order to make it easier to understand, analyze or visualize.

https://nbisweden.github.io/module-openrefine-dm-practices/03-filter-sort/index.html

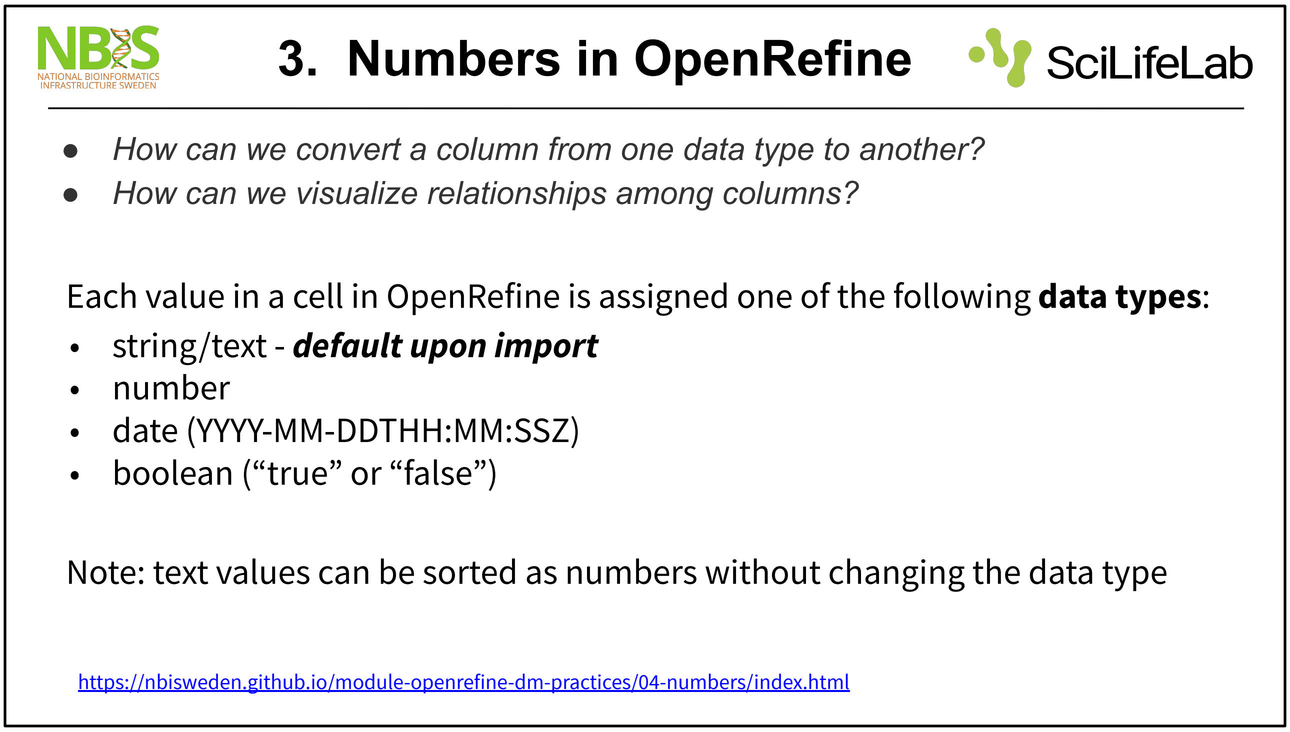
Second episode:

- *How can we select only a subset of our data to work with?*
- *How can we sort our data?*

Two other useful functions to facilitate exploring and working with your data is to filter and sort it.

Interactive steps

1. Demo Filtering
2. Exercise 2.1
3. Exercise 2.2
4. Demo Exercise 2.3
5. Exercise 2.4
6. Demo sorting by multiple columns
7. Exercise 2.5 and optional 2.6



3. Numbers in OpenRefine

- *How can we convert a column from one data type to another?*
- *How can we visualize relationships among columns?*

Each value in a cell in OpenRefine is assigned one of the following **data types**:

- string/text - ***default upon import***
- number
- date (YYYY-MM-DDTHH:MM:SSZ)
- boolean ("true" or "false")

Note: text values can be sorted as numbers without changing the data type

https://nbisweden.github.io/module-openrefine-dm-practices/04-numbers/index.html

Third episode:

- *How can we convert a column from one data type to another?*
- *How can we visualize relationships among columns?*

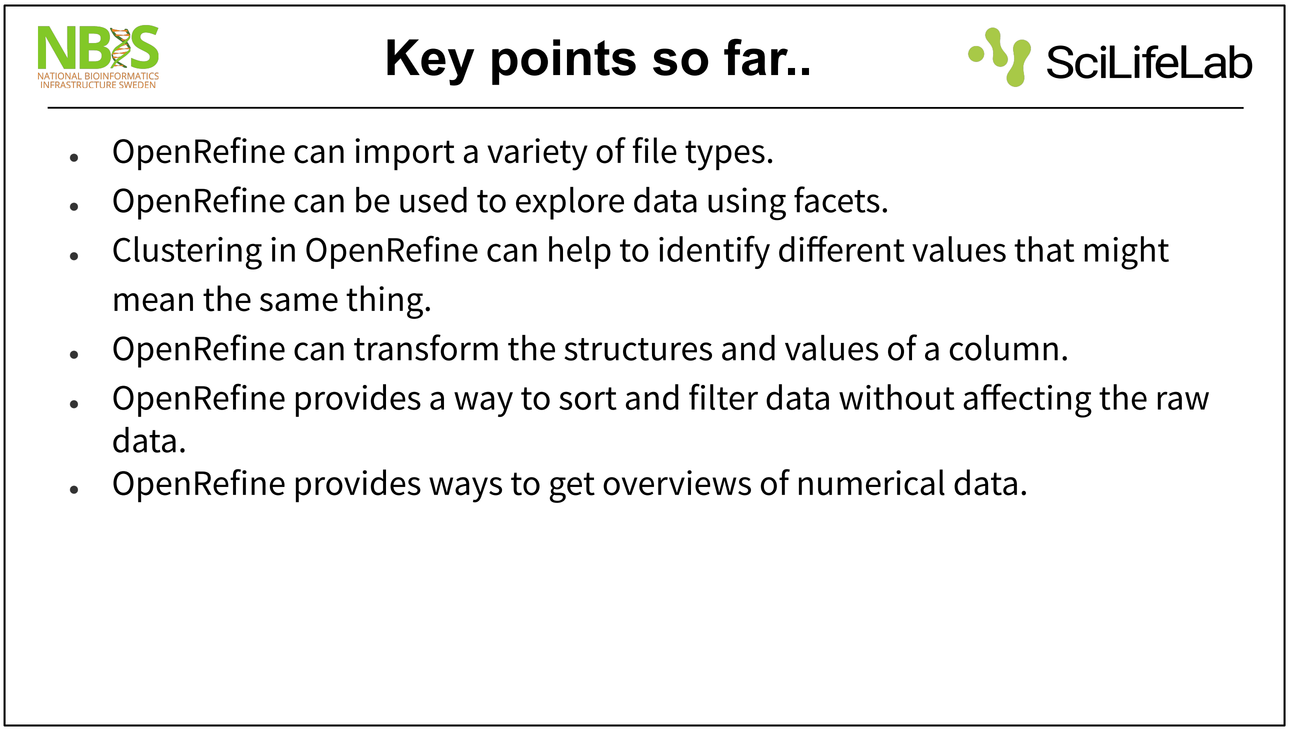
Supported data types:

- string (one or more text characters)
- number (one or more characters of numbers only)
- date (ISO-8601-compliant extended format with time in UTC: YYYY-MM-DDTHH:MM:SSZ)
- boolean (values of "true" or "false") A 'Boolean' is a binary value that can either be 'true' or 'false'. Boolean values can be used directly in OpenRefine cell

When a table is imported into OpenRefine, all columns are treated as having text values. We saw earlier how we can sort column values as numbers, but this does not change the cells in a column from text to numbers. Rather, this interprets the values as numbers for the purposes of sorting but keeps the underlying data type as is. We can, however, transform columns to other data types (e.g. number or date) using the `Edit cells` > `Common transforms` feature. Here we will experiment changing columns to numbers and see what additional capabilities that grants us.

Interactive steps:

1. Demo Numbers and exercise 3.1 - briefly discuss
2. Exercise 3.2
3. Revisit Undo/Redo by demoing the points after exercise 3.2



Key points so far..

- OpenRefine can import a variety of file types.
- OpenRefine can be used to explore data using facets.
- Clustering in OpenRefine can help to identify different values that might mean the same thing.
- OpenRefine can transform the structures and values of a column.
- OpenRefine provides a way to sort and filter data without affecting the raw data.
- OpenRefine provides ways to get overviews of numerical data.

We have already seen that OpenRefine has many functionalities that are very useful for cleaning up messy data. This is just a taste of what you can use OpenRefine for, there are many advanced and customizable functions that can be used and learned at a different time for those interested.
However, one key aspect of OpenRefine that we are going to have a look at next is that every change you make to your data is recorded automatically. This makes OpenRefine a great tool for aspects of Reusability and reproducibility

Reusability and Reproducibility

- OpenRefine tracks and documents all the modifications done to the data
- OpenRefine allows you to export the documentation in order to apply the same modifications to another dataset with the same structure

Why is this important?

- It makes your own work more efficient

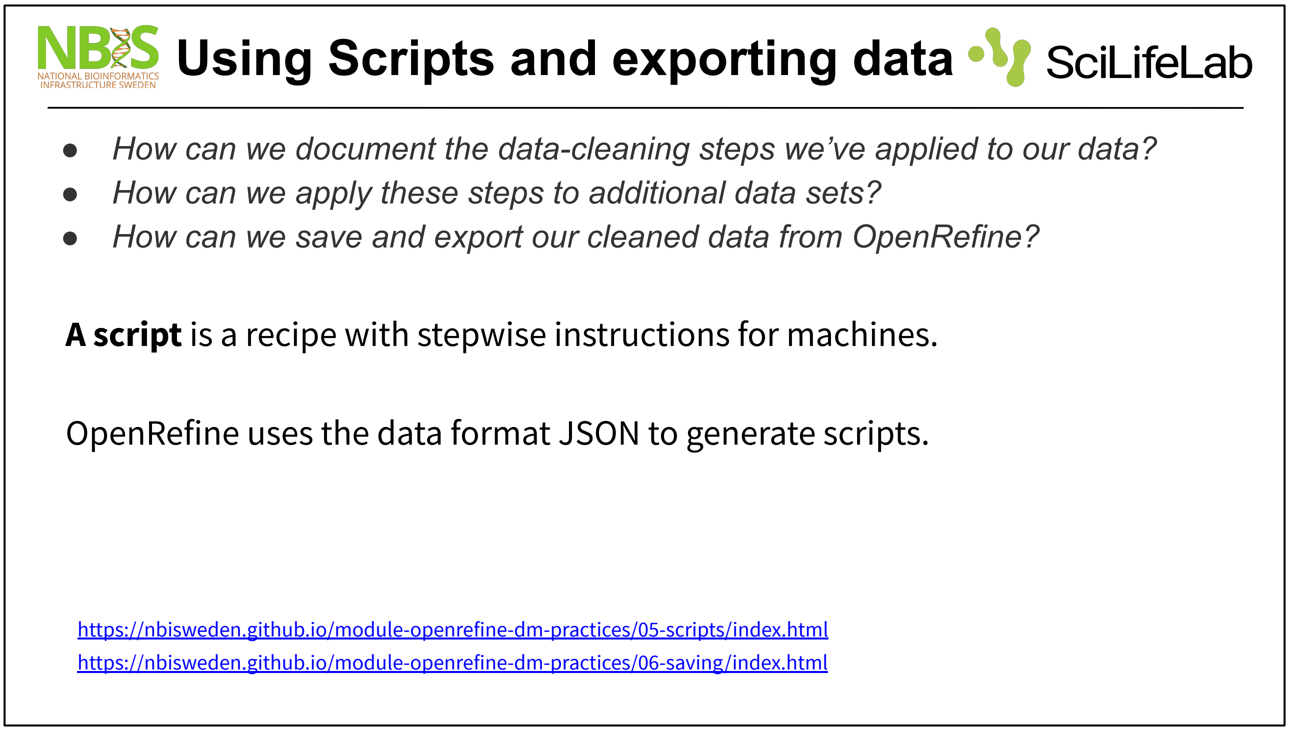
- It provides documentation for yourself and others to understand how the data has been modified

- It provides everything necessary to reproduce your cleaned data

We saw in the undo/redo tab that OpenRefine tracks and documents all the modifications done to the data.
Instead of keeping track of these steps yourself in a handwritten log as was mentioned in the data organisation lesson, OpenRefine does this for you. In addition, it allows you to export those steps in order to apply the same steps to another dataset with the same structure - Saving you time the next time around.

Benefits:

- It makes your own work more efficient
- Provides documentation for yourself and others to understand what has been done
- Provides everything necessary to reproducing your cleaned data.



NBIS NATIONAL BIOINFORMATICS INFRASTRUCTURE SWEDEN

Using Scripts and exporting data

SciLifeLab

- *How can we document the data-cleaning steps we've applied to our data?*
- *How can we apply these steps to additional data sets?*
- *How can we save and export our cleaned data from OpenRefine?*

A script is a recipe with stepwise instructions for machines.

OpenRefine uses the data format JSON to generate scripts.

https://nbisweden.github.io/module-openrefine-dm-practices/05-scripts/index.html
https://nbisweden.github.io/module-openrefine-dm-practices/06-saving/index.html

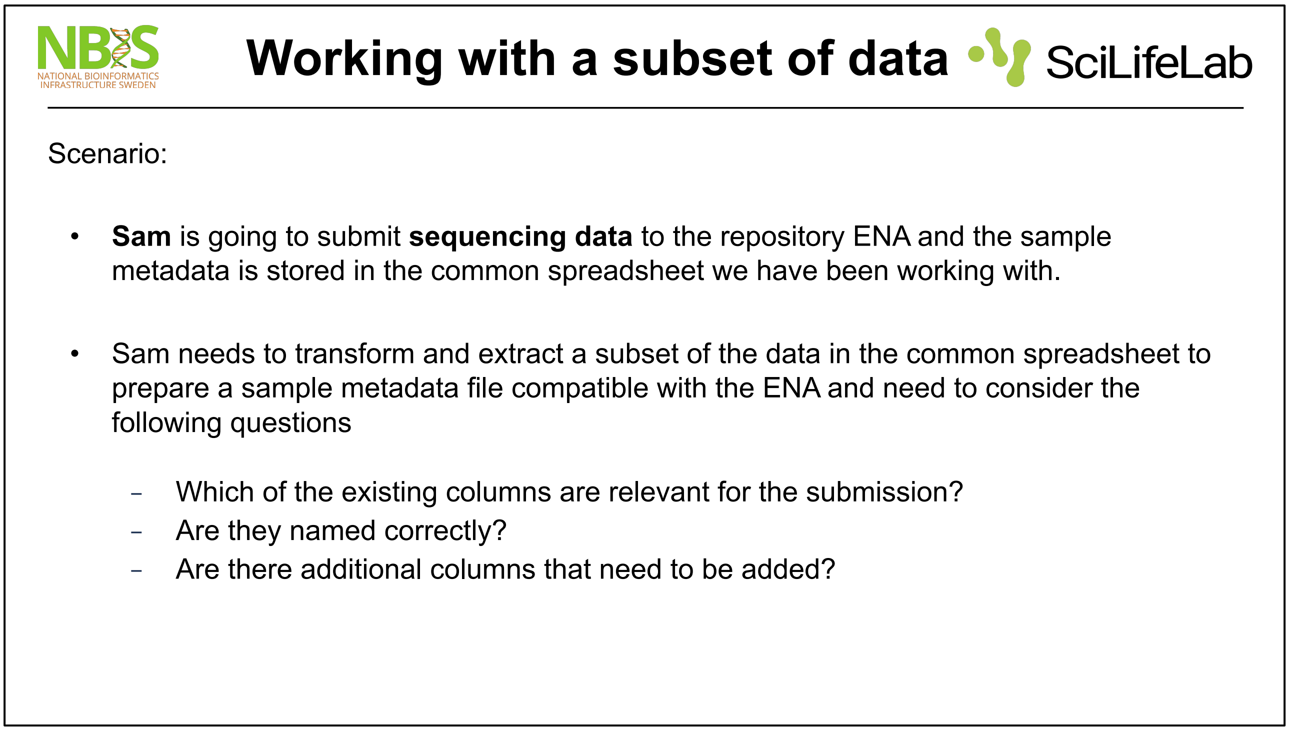
Fourth and fifth episode:

- *How can we document the data-cleaning steps we've applied to our data?*
- *How can we apply these steps to additional data sets?*
- *How can we save and export our cleaned data from OpenRefine?*

A script can be described as a recipe with stepwise instructions for machines. Scripts can have be generated and stored in many different formats or languages. OpenRefin uses a the format JSON {Jason} which stands for JavaScript Object Notation

Interactive steps:

1. Demo copying Script to txt.file.
! Have a look at the JSON and point out
    a) how to identify one element
    b) how to identify the different clustering edits. In the list there are several called 'Mass edit cells in column sex' and you need to look for the values changed ie. M to male.
2. Demo importing Script to use against new dataset from episode 5
3. Demo saving and exporting from episode 6



Working with a subset of data

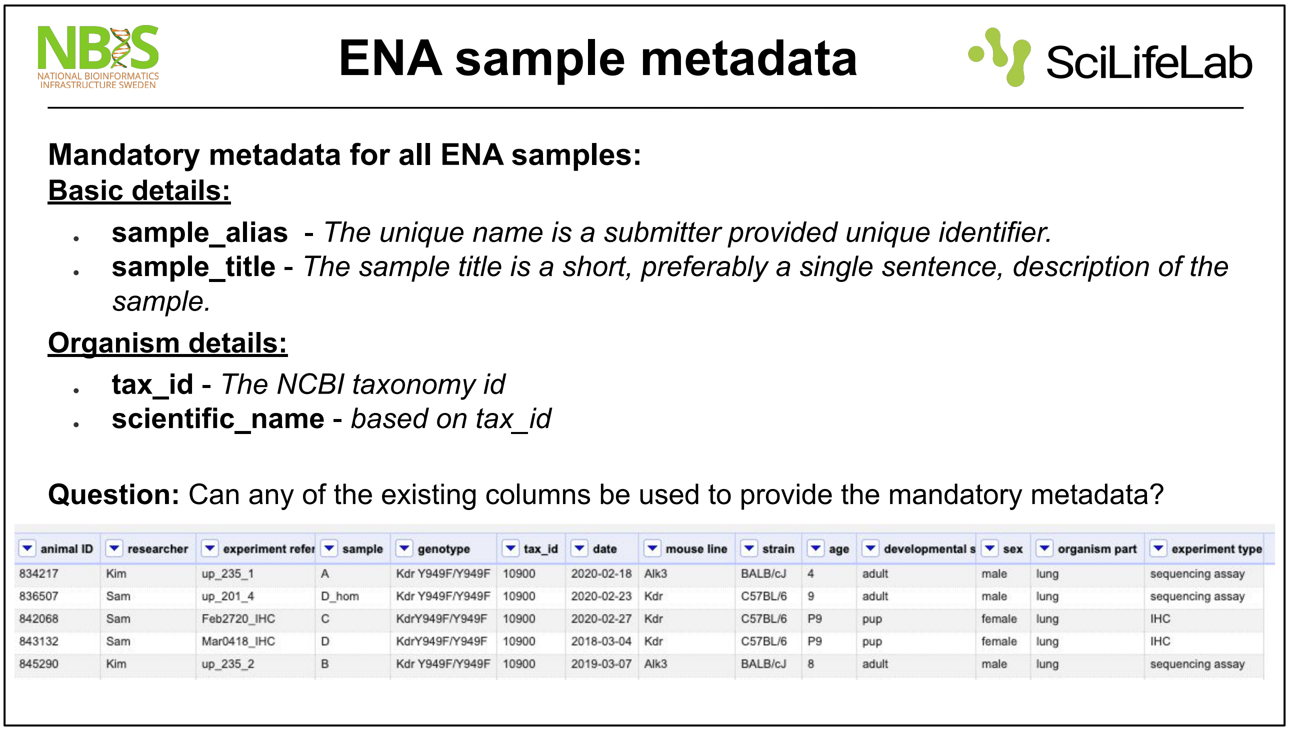
Scenario:

- **Sam** is going to submit **sequencing data** to the repository ENA and the sample metadata is stored in the common spreadsheet we have been working with.
- Sam needs to transform and extract a subset of the data in the common spreadsheet to prepare a sample metadata file compatible with the ENA and need to consider the following questions
  - Which of the existing columns are relevant for the submission?
  - Are they named correctly?
  - Are there additional columns that need to be added?

Sometimes you would like to export a file that only contains a subset of the data in your project that conforms to a specific standard

In the next lesson of the we will act as the researcher **Sam**
who wants to submit data to the repository ENA and need to prepare a sample metadata file that can be uploaded to register samples.

Looking at the table in OpenRefine Sam need to consider the following questions:
- **Which of the existing columns are relevant for the submission?**
- **Are they named correctly?**
- **Are there additional columns that need to be added?**



ENA sample metadata

Mandatory metadata for all ENA samples:

Basic details:

- **sample_alias** - *The unique name is a submitter provided unique identifier.*
- **sample_title** - *The sample title is a short, preferably a single sentence, description of the sample.*

Organism details:

- **tax_id** - *The NCBI taxonomy id*
- **scientific_name** - *based on tax_id*

Question: Can any of the existing columns be used to provide the mandatory metadata?

animal ID	researcher	experiment refer	sample	genotype	tax_id	date	mouse line	strain	age	developmental s	sex	organism part	experiment type
834217	Kim	up_235_1	A	Kdr Y949F/Y949F	10900	2020-02-18	Alk3	BALB/cJ	4	adult	male	lung	sequencing assay
836507	Sam	up_201_4	D_hom	Kdr Y949F/Y949F	10900	2020-02-23	Kdr	C57BL/6	9	adult	male	lung	sequencing assay
842068	Sam	Feb2720_IHC	C	KdrY949F/Y949F	10900	2020-02-27	Kdr	C57BL/6	P9	pup	female	lung	IHC
843132	Sam	Mar0418_IHC	D	KdrY949F/Y949F	10900	2018-03-04	Kdr	C57BL/6	P9	pup	female	lung	IHC
845290	Kim	up_235_2	B	Kdr Y949F/Y949F	10900	2019-03-07	Alk3	BALB/cJ	8	adult	male	lung	sequencing assay

You already did some of this work when creating the data dictionary in the metadata lesson where you identified ENA variables based on the default checklist. We will come back to these in a while but first we will look at the mandatory metadata for all samples submitted to ENA regardless of the checklist chosen.
The mandatory sample metadata can be divided in basic details and organism details.

- **sample_alias** - *The unique name is a submitter provided unique identifier.*
- **sample_title** - *The sample title is a short, preferably a single sentence, description of the sample.*
- **tax_id** - *The NCBI taxonomy id*
- **scientific_name** - *based on tax_id*

Discuss if any of the existing columns can be used to provide the mandatory metadata

Solution:

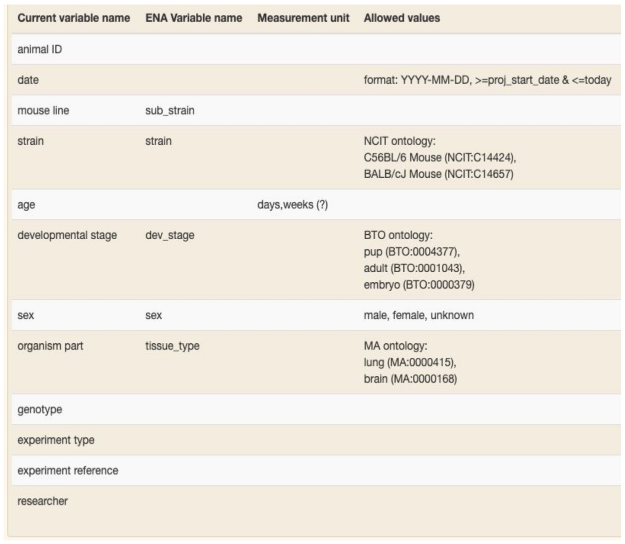
Sample_alias use experiment reference

sample_title could be created by joining sample and genotype columns

tax_id already exists

scientific_name need to create a new column with the value "Mus musculus"

ENA sample metadata



Current variable name	ENA Variable name	Measurement unit	Allowed values
animal ID			
date			format: YYYY-MM-DD, >=proj_start_date & <=today
mouse line	sub_strain		
strain	strain		NCIT ontology: C56BL/6 Mouse (NCIT:C14424), BALB/cJ Mouse (NCIT:C14657)
age		days,weeks (?)	
developmental stage	dev_stage		BTO ontology: pup (BTO:0004377), adult (BTO:0001043), embryo (BTO:0000379)
sex	sex		male, female, unknown
organism part	tissue_type		MA ontology: lung (MA:0000415), brain (MA:0000168)
genotype			
experiment type			
experiment reference			
researcher			

Checklist-derived metadata:

- strain
- sub_strain
- dev_stage
- sex
- tissue_type

To specify the ontology terms we will add **custom fields:**

- strain_ID
- dev_stage_ID
- tissue_type_ID

Let’s have a look at the data dictionary from the metadata lesson yesterday. Based on the ENA default checklist you identified 5 variables to use and identified ontology terms to be used for three of them.

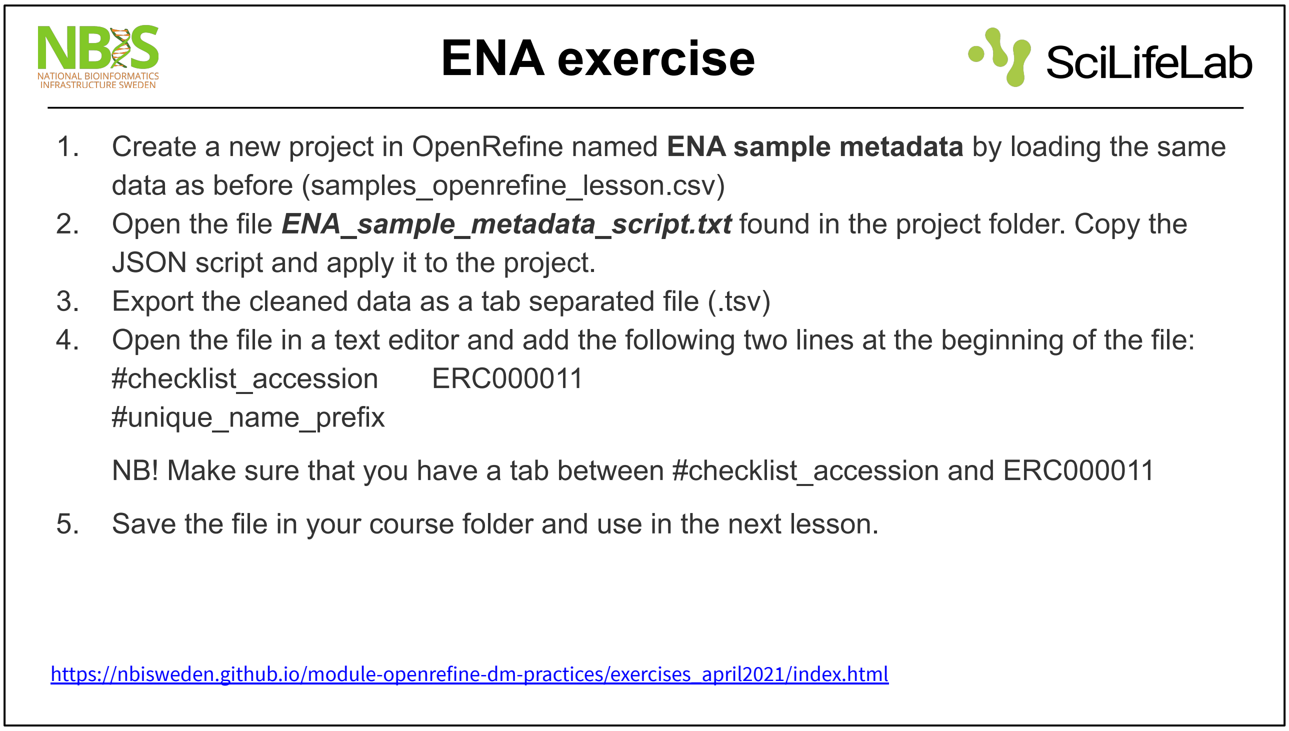
The ENA variable names are

a. Strain
b. sub_strain
c. dev_stage
d. sex
e. Tissue_type

We need to rename the 3 columns mouse line, developmental stage and organism part to conform to the ENA standard.

In order to add the ontology-IDs you documented in the data dictionary we will use the option to add custom fields to ENA sample metadata . We need to create 3 new columns based on the ENA variables that you identified ontology-IDs for and add “_ID” to the end

In sake of time, we will apply a script to the original sample file that removes, renames and add columns to create the file we need. All the steps are explained in detail in the episode 7 of the course material .



ENA exercise

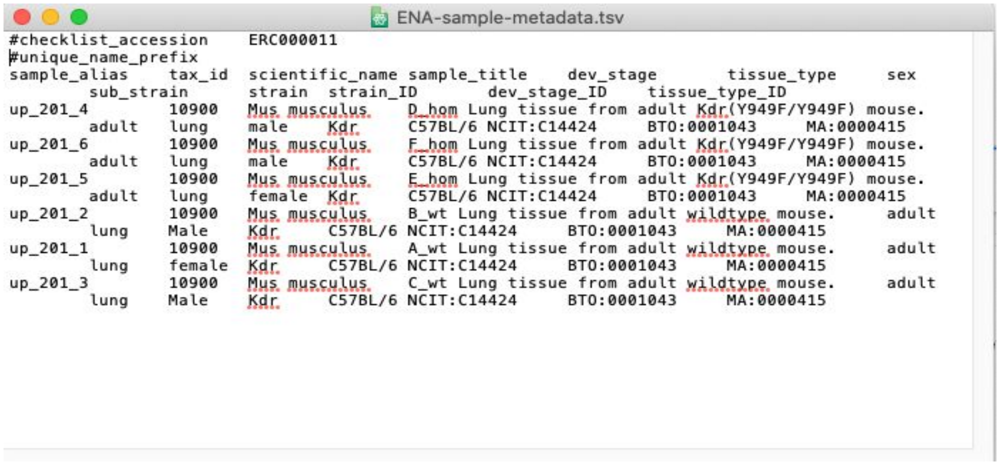
1. Create a new project in OpenRefine named **ENA sample metadata** by loading the same data as before (samples_openrefine_lesson.csv)
2. Open the file ***ENA_sample_metadata_script.txt*** found in the project folder. Copy the JSON script and apply it to the project.
3. Export the cleaned data as a tab separated file (.tsv)
4. Open the file in a text editor and add the following two lines at the beginning of the file:
   #checklist_accession	ERC000011
   #unique_name_prefix

   NB! Make sure that you have a tab between #checklist_accession and ERC000011
5. Save the file in your course folder and use in the next lesson.

https://nbisweden.github.io/module-openrefine-dm-practices/exercises_april2021/index.html

Depending on time.
Little time - Just show the resulting file
Some more time - demo the exercise
Enough time -
1) Let them do the exercise themselves, Instructions and solution are available in the extra>exercises_april2021
2) Have a look at the resulting file together.

Resulting .tsv file


ENA-sample-metadata.tsv

```
#checklist_accession    ERC000011
#unique_name_prefix
sample_alias    tax_id  scientific_name sample_title    dev_stage       tissue_type     sex
        sub_strain      strain  strain_ID       dev_stage_ID    tissue_type_ID
up_201_4        10900   Mus musculus    D_hom Lung tissue from adult Kdr(Y949F/Y949F) mouse.
        adult   lung    male    Kdr     C57BL/6 NCIT:C14424     BTO:0001043     MA:0000415
up_201_6        10900   Mus musculus    F_hom Lung tissue from adult Kdr(Y949F/Y949F) mouse.
        adult   lung    male    Kdr     C57BL/6 NCIT:C14424     BTO:0001043     MA:0000415
up_201_5        10900   Mus musculus    E_hom Lung tissue from adult Kdr(Y949F/Y949F) mouse.
        adult   lung    female  Kdr     C57BL/6 NCIT:C14424     BTO:0001043     MA:0000415
up_201_2        10900   Mus musculus    B_wt Lung tissue from adult wildtype mouse.     adult
        lung    Male    Kdr     C57BL/6 NCIT:C14424     BTO:0001043     MA:0000415
up_201_1        10900   Mus musculus    A_wt Lung tissue from adult wildtype mouse.     adult
        lung    female  Kdr     C57BL/6 NCIT:C14424     BTO:0001043     MA:0000415
up_201_3        10900   Mus musculus    C_wt Lung tissue from adult wildtype mouse.     adult
        lung    Male    Kdr     C57BL/6 NCIT:C14424     BTO:0001043     MA:0000415
```

Other resources

- *What other resources are available for working with OpenRefine?*

OpenRefine has its own web site with documentation and a book:

- OpenRefine web site
- OpenRefine Documentation for Users
- Using OpenRefine book by Ruben Verborgh, Max De Wilde and Aniket Sawant
- OpenRefine history from Wikipedia

You have now gotten an introduction to OpenRefine and seen some of the things you can use OpenRefine for but there are many advanced and customizable functions that can be used and learned at a different time for those interested.

If you go to the training materials you will find some more exercises and a section about resources to learn more.