CellGenIT Docs

Release 0.0.1

Mar 02, 2022

Contents

1 The group **3**
- 1.1 Our story . . . 4
- 1.2 Our remit . . . 4
- 1.3 Our vision . . . 5

2 Jupyter Hub **7**
- 2.1 How to get access . . . 7
- 2.2 Flavours . . . 7
- 2.3 Notebooks . . . 8

3 Data sharing **9**
- 3.1 Globus . . . 9
- 3.2 cram files . . . 9

4 Data submission **11**
- 4.1 ArrayExpress . . . 11
- 4.2 EGA/ENA . . . 11

5 Nextflow pipelines **13**

6 ML Warehouse **15**
- 6.1 Show me information for all the samples in a study . . . 15
- 6.2 Show me information for all the samples with a cost code . . . 16
- 6.3 Show me run ID, lane number and tag index for a sample . . . 16

7 Single Cell Visualizations **19**
- 7.1 cellxgene . . . 19
- 7.2 AnnData . . . 19
- 7.3 Seurat -> h5ad . . . 21
- 7.4 SingleCellExperiment -> h5ad . . . 22
- 7.5 Loom -> h5ad . . . 22
- 7.6 Examples . . . 23

8 Contact us **25**
- 8.1 Submit a ticket . . . 25
- 8.2 Coffee standups . . . 25
- 8.3 Chat . . . 25

CHAPTER 1

The group

1.1 Our story

1.1.1 Cellular Genetics programme (2015)

The Cellular Genetics programme at the Sanger Institute was created in 2015 by merging with the Computational Genomics programme. In the first two years the programme didn't have any core facility groups. In 2017 it was decided to create several core facility groups including ours to help the faculty groups do their research.

1.1.2 Early days

In the early days of the programme (the first 2 years) all of the users analysed their data by themselves on the LSF batch compute farm available at the Sanger. This included both computational and non-computational users. Some people had never coded before. This introduced a lot of heterogeneity in the programme data.

1.1.3 Cloud (2017)

In 2017 it was decided by Wellcome Trust to adapt to the modern cloud technologies and to move their compute to the cloud. The Sanger Institute decided to start with having a private OpenStack cloud on premises and gradually move analysis from the LSF farm to the cloud. It created additional challenges for the end users and therefore it was decided to create the Cellular Genetics core facility informatics group which would standardize the analysis and help with transitioning to the cloud.

1.1.4 Cellular Genetics Informatics group (2018)

Our group was created in the beginning of 2018 and Vladimir Kiselev started as the head of the group. Stijn van Dongen joined the group in May 2018 and Anton Khodak joined the group in June 2018.

1.2 Our remit

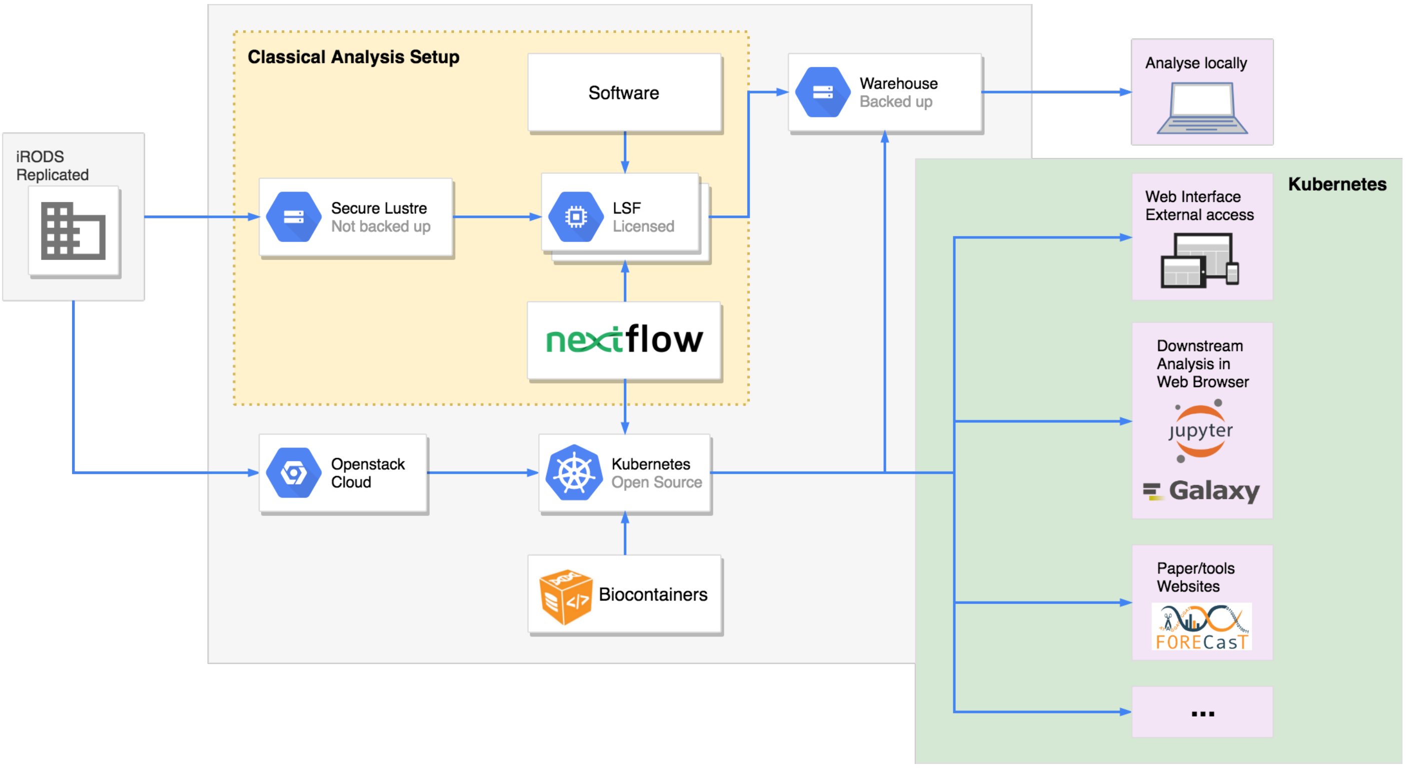
1.2.1 Group

Our team provides efficient access to cutting-edge analysis methods for Cellular Genetics programme at the Sanger Institute, which leads a number of world-class research initiatives including the Human Induced Pluripotent Stem Cell Initiative (HIPSCI) and Human Cell Atlas. Our team's focus is the development and operation of workflows, tools and infrastructure for data analysis that support the programme's research goals.

1.2.2 Compute framework

To run our pipelines we use both the LSF compute cluster and OpenStack private cloud solutions. We extensively use Kubernetes to orchestrate pipelines on the cloud. To run the pipelines we use both Nextflow and CWL.

Our GitHub organisation



Classical Analysis Setup
Software
Warehouse
Backed up
Analyse locally
iRODS
Replicated
Secure Lustre
Not backed up
LSF
Licensed
Kubernetes
Web Interface
External access
nextflow
Downstream
Analysis in
Web Browser
jupyter
Galaxy
Openstack
Cloud
Kubernetes
Open Source
Biocontainers
Paper/tools
Websites
FORECasT
...

1.3 Our vision

When working on our projects and pipelines we follow several important principles.

1.3.1 No duplication of effort

If something has been or can be done by others we will not develop our own tool for this.

1.3.2 Agile

In our work we try use elements of the Agile methodology. We use Jira as our main project management software.

1.3.3 User-oriented

We value our users not only as our customers (we use Jira Service Desk for all of the users' requests) but also as the most important source of ideas and feedback. Anyone can either email us on vk6@sanger.ac.uk or talk to us in our Mattermost group (Sanger users can login using their Sanger credentials).

1.3.4 Feedback

We also have monthly meetings with faculty group's representatives and weekly coffee standups where anyone can come, comment on our work, feedback or ask us any question.

CHAPTER 2

Jupyter Hub

We support a Jupyter Hub server running on Sanger Cloud. Jupyter allows you to run your analysis in multiple environments (`R`, `python`, `Julia`, etc.) and also to create and share notebooks containing your analysis, code, equations and visualizations. We think this is an ideal environment for any kind of downstream analysis. For more details please refer to Jupyter Hub documentation.

2.1 How to get access

Our Jupyter Hub service is available in a web browser on any computer anywhere in the world. You will need to provide us with your GitHub ID to be able to login. Once we notify you that your account is created you can login using your GitHub credentials.

Before logging in to JupyterHub please read our instructions.

2.2 Flavours

There are three flavours of Jupyter Hub that we provide. Please choose the one required for your needs and access ti via links below.

Note: Please only use **Large** and **Extra Large** flavours when this much resources is really required because this might impact the availability of large instances to other users.

2.2.1 Default

https://jupyter.cellgeni.sanger.ac.uk

RAM: 33 Gb

vCPUs: 2-4
Disk: 30 Gb

2.2.2 Large

https://jupyter-large.cellgeni.sanger.ac.uk

RAM: 100 Gb
vCPUs: 4-16
Disk: 100 Gb

2.2.3 Extra Large

https://jupyter-xl.cellgeni.sanger.ac.uk

RAM: 150 Gb
vCPUs: 4-16
Disk: 150 Gb

2.3 Notebooks

We provide some notebook templates with the pre-installed software. These are located in the `notebooks` folder. Corresponding example data is located in the `data` folder. Before running your analysis, please make a copy of a notebook template, save it to your home folder and work with the copy. At the moment we have the following notebook templates:

1. scanpy notebook for analysis of 10X data
2. Seurat notebook for analysis of 10X data
3. Batch correction notebook 1
4. Batch correction notebook 2

2.3.1 How to get help

For any Jupyter Hub related questions please use our MatterMost channel. There are lots of users there who can quickly answer your questions.

CHAPTER 3

Data sharing

3.1 Globus

We use Globus network to share the data with external collaborators. It allows us to share data from e.g. a specific folder on the Sanger LFS cluster directly with the external world.

The sharing process consists of the following steps:

1. We share the data with the user's personal/work email address
2. The user creates/logs in their Globus account using the sharing email
3. The user needs to create a personal Globus endpoint either on their Linux laptop / compute cluster or on their Mac laptop or on their Windows laptop.
4. The user activates their personal Globus endpoint by starting globus from the command line if on a cluster/Linux, or by starting the globus application if on Mac/Windows.
5. Once the users personal endpoint is setup they can transfer the data by simply logging in to their Globus account using the sharing email address and drag and dropping the data.

For more information please visit the Globus official documentation.

Note: If the user would like to check MD5 hash, the MD5 sum files will be located in the same sharing folder with the data files.

3.2 cram files

Sanger default file format for storing NGS data is `CRAM` and this is what we provide to the user when share data with them. Typically `CRAM` achieves 40-50% space saving over the alternative `BAM` format and much more than that over the compressed `fastq` files. For more information please visit this page.

Once the user obtained the data from Globus, the data can be converted from `CRAM` to `fastq` format using the following steps:

- Install `samtools` with version **>=1.8** (in this case `samtools` should automatically download the right genome reference if your local installation does not have it)
- Run the following commands (set NCPU to a number of CPUs, if you are on a multi-cpu machine). This will create paired fastq files `samplename_1.fastq.gz` and `samplename_2.fastq.gz`:

```
samtools collate -O -u -@ NCPU samplename.cram tmppfx | \
    samtools fastq -N -F 0x900 -@ NCPU -1 samplename_1.fastq.gz -2 samplename_2.fastq.
↪gz -
```

If this does not work, you could try running these first:

```
unset REF_PATH
unset REF_CACHE
```

CHAPTER 4

Data submission

4.1 ArrayExpress

To submit to ArrayExpress please follow their instructions. You will first need to create an account at Annotare and upload the experimental metadata corresponding to your data. After this is done you will be provided with an ftp folder name. Please forward this name to us with the corresponding list of Sanger sample IDs. We will then upload your data to ArrayExpress ftp folder based on the list of samples you provided to us.

4.2 EGA/ENA

To submit your data to a public repository, e.g. EGA or ENA, please contact datahose@sanger.ac.uk.

CHAPTER 5

Nextflow pipelines

We offer a range of sequencing processing pipelines. We develop them using Nextflow workflow language.

For the end-users we offer Nextflow pipelines available to run on both Sanger *LSF* farm or *OpenStack* flexible compute environment. The pipelines can be run either by us or by the end-users themselves.

Here is the list of pipelines we develop:

1. RNA-seq (both bulk and Smartseq2)
2. ATAC-seq
3. TraCeR/BraCer
4. Droplet based protocols (inDrop, Seq-Well, Drop-seq)
5. 10X cellranger

Note: If you would like to run `cellranger` by *10X* please put in a ticket for the NPG group by emailing to new-seq-pipe@sanger.ac.uk and specify which version of the cellranger and the genome you would like to use.

CHAPTER 6

ML Warehouse

Note: To see the content below you need to be on the Sanger network (either cable or WTSI wifi). Also, we recommend to use either Chrome or Safari browser.

6.1 Show me information for all the samples in a study

MySQL query:

```sql
SELECT
    study.id_study_lims,
    study.study_title,
    study.faculty_sponsor,
    study.ethically_approved,
    study.accession_number,
    sample.name,
    sample.supplier_name,
    sample.donor_id,
    sample.organism,
    sample.reference_genome,
    flowcell.cost_code,
    product_metrics.id_run,
    flowcell.position,
    flowcell.tag_index,
    flowcell.tag_sequence,
    flowcell.tag2_sequence

FROM mlwarehouse.iseq_flowcell as flowcell

JOIN (mlwarehouse.iseq_product_metrics as product_metrics,
    mlwarehouse.sample,
    mlwarehouse.study)
```

(continues on next page)

(continued from previous page)

```
ON (product_metrics.id_iseq_flowcell_tmp = flowcell.id_iseq_flowcell_tmp
    AND sample.id_sample_tmp = flowcell.id_sample_tmp
    AND flowcell.id_study_tmp = study.id_study_tmp)

WHERE study.id_study_lims = 4775
```

6.2 Show me information for all the samples with a cost code

MySQL query:

```
SELECT
    study.id_study_lims,
    study.study_title,
    study.faculty_sponsor,
    study.ethically_approved,
    study.accession_number,
    sample.name,
    sample.supplier_name,
    sample.donor_id,
    sample.organism,
    sample.reference_genome,
    flowcell.cost_code,
    product_metrics.id_run,
    flowcell.position,
    flowcell.tag_index,
    flowcell.tag_sequence,
    flowcell.tag2_sequence

FROM mlwarehouse.iseq_flowcell as flowcell

JOIN (mlwarehouse.iseq_product_metrics as product_metrics,
    mlwarehouse.sample,
    mlwarehouse.study)
ON (product_metrics.id_iseq_flowcell_tmp = flowcell.id_iseq_flowcell_tmp
    AND sample.id_sample_tmp = flowcell.id_sample_tmp
    AND flowcell.id_study_tmp = study.id_study_tmp)

WHERE flowcell.cost_code = 'S2435'
```

6.3 Show me run ID, lane number and tag index for a sample

MySQL query:

```
SELECT DISTINCT
    sample.name,
    study.id_study_lims,
    flowcell.pipeline_id_lims,
    product_metrics.id_run,
    flowcell.position,
    flowcell.tag_index,
```

(continues on next page)

(continued from previous page)

```
    flowcell.tag_sequence,
    flowcell.tag2_sequence,
    run_status_dict.description,
    run_status.date

FROM mlwarehouse.sample

JOIN (mlwarehouse.iseq_flowcell as flowcell,
    mlwarehouse.iseq_run_status as run_status,
    mlwarehouse.iseq_product_metrics as product_metrics,
    iseq_run_status_dict as run_status_dict,
    mlwarehouse.study as study)

ON (flowcell.id_sample_tmp = sample.id_sample_tmp
    AND product_metrics.id_iseq_flowcell_tmp = flowcell.id_iseq_flowcell_tmp
    AND run_status.id_run = product_metrics.id_run
    AND run_status.id_run_status_dict = run_status_dict.id_run_status_dict
    AND flowcell.id_study_tmp = study.id_study_tmp)

WHERE sample.name = 'QC1Hip-11155' AND run_status.iscurrent = 1;
```

CHAPTER 7

Single Cell Visualizations

Authors: Batuhan Cakir and Vladimir Kiselev

If you would like to make your single-cell RNA-seq data publicly available on a website, for example as a supplement for a publication, we can help you with that!

7.1 cellxgene

We use cellxgene to visualize single cell RNA-seq data. **cellxgene** is an interactive data explorer which is very scalable and flexible.

To be able for us to create a **cellxgene** website for your data we need to have your data in the h5ad (AnnData) format.

7.2 AnnData

AnnData format usually contains the following slots:

- **X** contains the expression matrix.
- **obsm** contains the embeddings data.
- **obs** contains the cell metadata.
- **var** contains the gene metadata.

When you work with **cellxgene** you only need to modify two of the slots above: **obsm** and **obs**.

7.2.1 X slot

cellxgene works faster when the expression matrix is stored in `CSC` (compressed sparse column) format instead of `CSR` (compressed sparse row) format or dense Numpy array (which sometimes can create a smaller `h5ad` file depending on the sparsity of your data).

To convert your expression matrix into the `CSC` format please use:

```
adata.X = scipy.sparse.csc_matrix(adata.X)
```

To convert your expression matrix into the Numpy array please use:

```
adata.X = scipy.sparse.csc_matrix.toarray(adata.X)
```

7.2.2 obsm slot

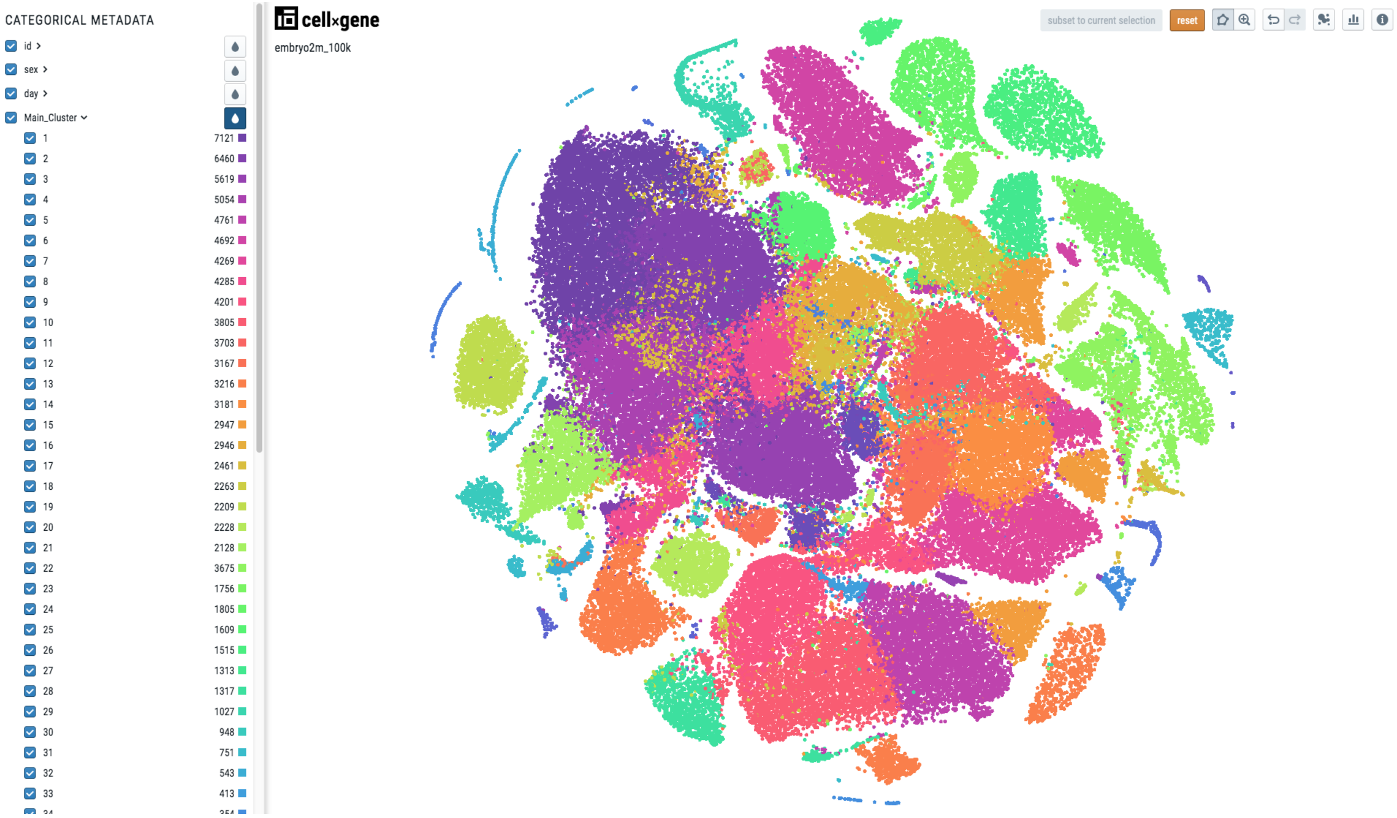
To visualize your cells in 2D **cellxgene** uses **obsm** slot. If there are multiple embeddings stored in this slot they will all be available on the web interface.

Note: **cellxgene** requires that all of the embeddings' names are prefixed with `X_`. For example, `X_umap`, `X_pca` or `X_some_embedding`.

7.2.3 obs slot

To highlight and colour your cells **cellxgene** uses **obs** slot. The colouring will depend on the type of you cell metadata contained in the **obs** slot.

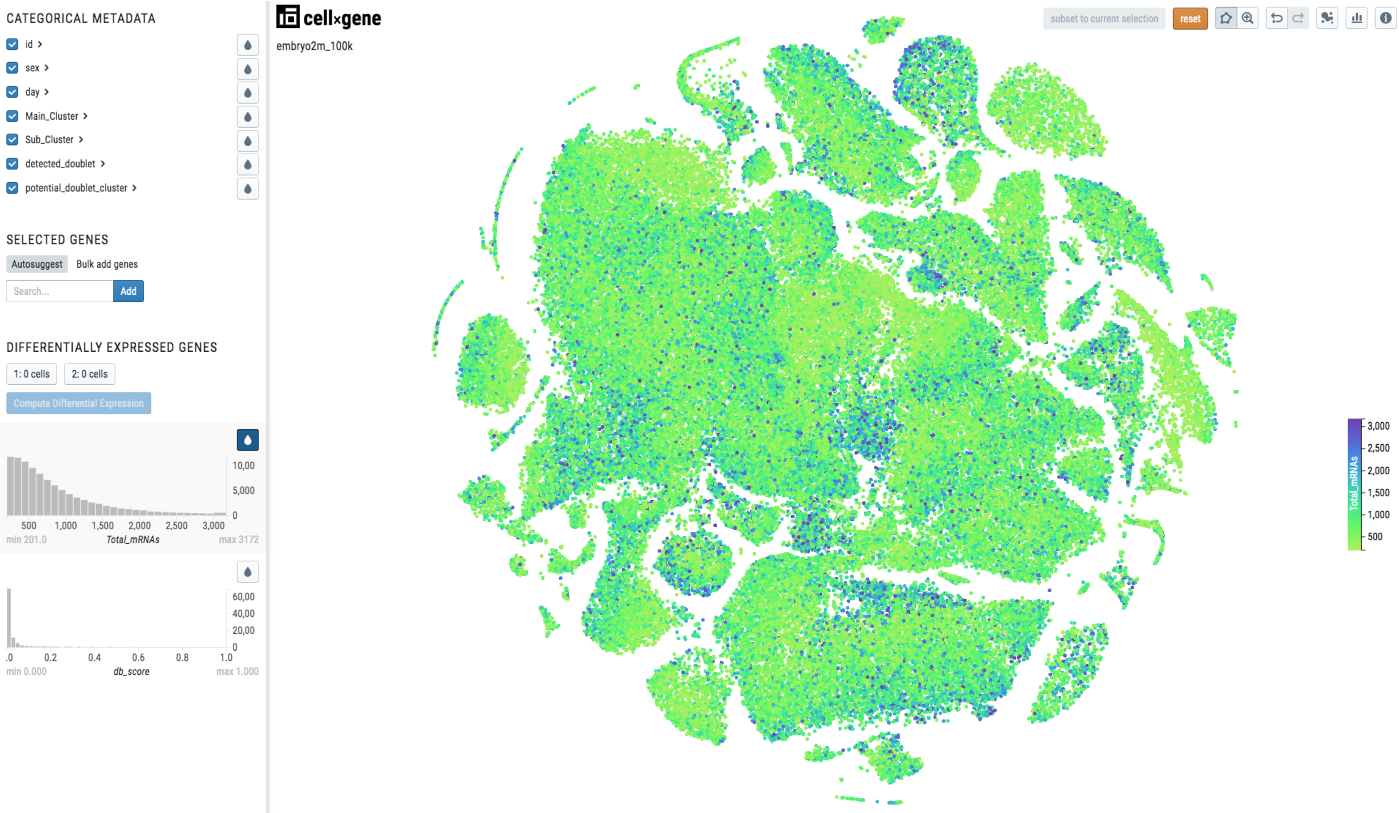
When the metadata is *categorical*, i.e. there is one colour per category, the visualization will look like this:


CATEGORICAL METADATA
☑ id >
☑ sex >
☑ day >
☑ Main_Cluster ⌄
☑ 1 7121
☑ 2 6460
☑ 3 5619
☑ 4 5054
☑ 5 4761
☑ 6 4692
☑ 7 4269
☑ 8 4285
☑ 9 4201
☑ 10 3805
☑ 11 3703
☑ 12 3167
☑ 13 3216
☑ 14 3181
☑ 15 2947
☑ 16 2946
☑ 17 2461
☑ 18 2263
☑ 19 2209
☑ 20 2228
☑ 21 2128
☑ 22 3675
☑ 23 1756
☑ 24 1805
☑ 25 1609
☑ 26 1515
☑ 27 1313
☑ 28 1317
☑ 29 1027
☑ 30 948
☑ 31 751
☑ 32 543
☑ 33 413
cellxgene
embryo2m_100k
subset to current selection
reset

To make your cell metadata categorical please use the following code:

```
import pandas as pd
adata.obs['metadata_name'] = pd.Categorical(adata.obs['metadata_name'])
```

When the metadata is *continuous*, the visualization will look like this:


CATEGORICAL METADATA
☑ id ›
☑ sex ›
☑ day ›
☑ Main_Cluster ›
☑ Sub_Cluster ›
☑ detected_doublet ›
☑ potential_doublet_cluster ›
SELECTED GENES
Autosuggest
Bulk add genes
Search...
Add
DIFFERENTIALLY EXPRESSED GENES
1: 0 cells
2: 0 cells
Compute Differential Expression
10,00
5,000
0
500
1,000
1,500
2,000
2,500
3,000
min 201.0
Total_mRNAs
max 3172
60,00
40,00
20,00
0
.0
0.2
0.4
0.6
0.8
1.0
min 0.000
db_score
max 1.000
cell×gene
embryo2m_100k
subset to current selection
reset
Total_mRNAs
3,000
2,500
2,000
1,500
1,000
500

Note: Note there is a continuous scale on the right side of the plot.

To make your cell metadata continuous please use the following code:

```
import numpy as np
adata.obs['metadata_name'] = np.float32(adata.obs['metadata_name'])
```

Warning: Before converting your data to h5ad format please make sure **anndata** library is installed on your system. Use the either of the following commands to install it: `pip install anndata` or `conda install anndata -c bioconda`.

7.3 Seurat -> h5ad

To convert a **Seurat** object `seurat_object` to `h5ad` format, you can use the **reticulate** package:

```
library(reticulate)
anndata <- import("anndata", convert = FALSE)
adata <- anndata$AnnData(
```

(continues on next page)

(continued from previous page)

```
    X = t(GetAssayData(object = seurat_object)),
    obs = data.frame(seurat_object@meta.data),
    obsm  = list(
        "X_emb1" = Embeddings(seurat_object[["emb1"]]),
        "X_emb2" = Embeddings(seurat_object[["emb2"]])
    )
)
anndata$AnnData$write(adata, 'filename.h5ad')
```

7.4 SingleCellExperiment -> h5ad

To convert a **SingleCellExperiment** object `sce_object` to `h5ad` format, you can use the **reticulate** package:

```
library(reticulate)
anndata <- import("anndata", convert = FALSE)
adata <- anndata$AnnData(
    X = t(counts(sce_object)),
    obs = data.frame(colData(sce_object)),
    obsm  = list(
        "X_emb1" = as.matrix(reducedDim(sce_object, "emb1")),
        "X_emb2" = as.matrix(reducedDim(sce_object, "emb2"))
    )
)
anndata$AnnData$write(adata, 'filename.h5ad')
```

7.5 Loom -> h5ad

To convert a **loom** file to `h5ad` format, you can use the following code (here we use an example dataset from Linnarson Lab which can be downloaded using this link):

```
import loompy
import scanpy as sc
import pandas
import numpy
import scipy

adata = sc.read_loom('L6_Immune_cells.loom')

# Move embeddings info to the right place and right format
x = pandas.Series.to_numpy(adata.obs['_X'])
y = pandas.Series.to_numpy(adata.obs['_Y'])
xy = numpy.stack((x,y)).transpose().reshape(-1,2)
adata.obsm['X_test'] = xy

# Only include necessary metadata:
adata.obs['Clusters'] = pandas.Categorical(adata.obs['Clusters'])
adata.obs = adata.obs[{'Clusters','Age','Sex'}]

# Change the matrix format
adata.X = scipy.sparse.csc_matrix(adata.X)

```

(continues on next page)

(continued from previous page)

```
# Make variable and observation names unique
adata.var_names_make_unique()
adata.obs_names_make_unique()

# Write h5ad file
adata.write('filename.h5ad')
```

7.6 Examples

We have already created a couple of websites for some our programme members. You can have a look at them at the following links:

https://www.kidneycellatlas.org

https://hemocytes.cellgeni.sanger.ac.uk

CHAPTER 8

Contact us

8.1 Submit a ticket

We operate using JIRA ticketing system. To be able to put in tickets you need to contact us by either emailing to vk6@sanger.ac.uk or using our Mattermost channel (please use your Sanger credentials to login).

Once registered you can put in a ticket using either our web portal.

8.2 Coffee standups

We organize coffee standups every Wednesday at 2pm in the Morgan Building Atrium. Please come along, meet our group, have a cup of coffee and ask any question you may have.

8.3 Chat

For small technical questions and informal interactions we use Mattermost. You can loging there with your Sanger credentials.

Note: The creation of this travel guide was inspired by Patrick Kua