Quickstart

andy henrie

9/3/2021

Table of Contents

Overview

In order to use the NDCN ‘omics’ browser we need to set a few things up. This should be four simple steps:

• Setup the environment: The underlying tools/packages from R and python

• Create the browser itself: which is an R package available on github

• Prepare the ‘omics’ data for browsing, and finally

• Configure the browser for data ingest

• Run the browser!

The following materials should provide a simple set of steps to accomplish this and start browsing ‘omics’!

Note: although the NDCN browser is not yet officially named, there are references to the browser/package as omicser or omxr.

Environment setup

We will need an R environment, which could be as simple as the base RStudio R, and a python environment which we will specify with conda. (In the future we can encapsulate everything in a docker container (including RSTudio?) to make it even more straightforward.)

Within RStudio / R

install.packages("devtools")  
install.packages("BiocManager")  
BiocManager::install("SingleCellExperiment")  
  
install.packages("reticulate")  
  
require(reticulate)  
reticulate::install\_miniconda()  
reticulate::conda\_create("omxr",python\_version = 3.9)  
reticulate::conda\_install(envname = "omxr", packages = "scanpy")  
reticulate::conda\_install(envname="omxr",  
 channel = "conda-forge",  
 packages = c("leidenalg") )  
  
# all other dependencies will be loaded by "omicser"

WIth Command Line

Make sure you ahve anaconda or mini-conda installed. e.g.

wget https://repo.anaconda.com/miniconda/Miniconda3-latest-MacOSX-x86\_64.sh -O ~/miniconda.sh  
bash ~/miniconda.sh -b -p $HOME/miniconda

Then make a conda environment to use with reticulate

conda create --name omxr python=3.9 scanpy   
conda install -c conda-forge leidenalg   
conda activate omxr

This is how you tell reticulate to use this conda environment:

reticulate::use\_condaenv("omxr")

NDCN Browser spin-up

We will need an R environment, which could be as simple as the base RStudio R, and a python environment which we will specify with conda. (In the future we can encapsulate everything in a docker container (including RSTudio?) to make it even more straightforward.) (https://github.com/ergonyc/omicser)

There are two ways to set up the omicser:  
1. Passive: Simply install the browser and curate the data. 2. Development mode: Clone the repository and customize for your needs.

Option 1: passive mode

devtools::install\_github("ergonyc/omicser")

Option 2: development mode

Here you want to clone the repo and develop a local version. e.g.

rstuido clone

or with the github cli:

gh repo clone ergonyc/omicser

Data Prep

For this stage you are the ‘curator’ of the data. The data needs to be formatted for the browser and placed somewhere the browser can access it.

The curate\_domenico\_stem\_cell.R script in the examples subdirectory illustrates the steps needed to curate the data for reading into the browser.

There are several steps, but the first step will be to make the .yml file that will let the browser know what/where the data will be. Your options are to edit the “omxr\_options.yml” directly or make a new one. e.g. for the stem cell proteomics example:

dataset\_names <- list(  
 "Domenico DIA" = "domenico\_stem\_cell"  
)  
# where do our data files live. # WARNING do not use the ~ alias for Home  
# MUST BE FULL or RELATIVE PATH... ~ will cause loading to  
ds\_root\_path = 'omicsdata'  
  
# python environment  
conda\_environment = 'omxr'  
  
omicser\_options <- list(dataset\_names=dataset\_names,  
 ds\_root\_path=ds\_root\_path,  
 conda\_environment=conda\_environment)  
  
require(configr)  
configr::write.config(config.dat = omicser\_options, file.path = "omxr\_options.yml",  
 write.type = "yaml", indent = 4)

The anndata scheme requires us to define three main tables: 1. the omic data matrix - e.g. transcriptomics - count matrix of cells X genes. 2. omic annotation data - e.g. gene families, “highly variable genes”, “is marker gene” 3. sample meta data - e.g. sex, experiemntal condition, etc.

We also need to define differential omic-expression tables.

Overall the basic task is to take the data being curated and place it into the NDCN browsers schema. I have separated this into a sequence straightforward steps:

• preamble / setup - make sure all the required tools/reposiries are laoded

• provenance & meta data setup - define the meta-data and context for the dataset

• helper function definition - any helpers you need

• load the raw data - load the raw data from whatever format they live in

• pack into the browser data format - pack into the anndata structure (scanpy/python)

• post-processing - compute relavent marginal quantities, define additional annotation and grouping variables, etc. 5a. dimension reduction - compute and cluster if needed

• differential expression tables - compute and/or formate existing tables

• write database - write the files to the database location

Here are some of the key helper functions and he section they fall into.

Here is an example of loading three data files and then pakaging them with a “helper function” (defined in section #2 of the example curation script) into the data\_list which will be used by the next stage.

#==== 3. load data -========================================================================================  
matrix\_data\_file <- "20210524\_093609\_170805\_aging\_against\_SC\_merged\_all\_lib\_2\_Report.xls"  
# candidate table without filter  
annot\_de\_file <- "170805\_aging\_against\_SC\_merged\_all\_lib\_2\_candidates.xls"  
# condition setup  
conditions\_table\_file <- "170805\_aging\_against\_SC\_merged\_all\_lib\_2\_ConditionSetup.xls"  
data\_list <- prep\_DIA\_files(matrix\_data\_file,annot\_de\_file,conditions\_table\_file,RAW\_DIR)  
# save diff expression data for later...  
diff\_exp <- data\_list$de  
# saveRDS(diff\_exp, file = file.path(DB\_DIR, "diff\_expr\_table.rds"))  
saveRDS(diff\_exp, file.path(DS\_ROOT\_PATH,DB\_NAME, "diff\_expr\_table.rds"))

The omicser::setup\_database() function packages the separate tables – data matrix, omic annotations, sample meta – into the anndata object. This function can also take the name of a seurat object file, or some other transcritpomic data objects, but most generically the data\_list is data from .csv or other simple data tables.

#==== 4. pack into anndata =========================================================================  
ad <- omicser::setup\_database(database\_name=DB\_NAME,  
 db\_path=DS\_ROOT\_PATH,  
 data\_in=data\_list,  
 db\_meta=NULL ,  
 re\_pack=TRUE)

Although this stage is not nescessary, its included as an example of how the python backend can be leveraged to do dimension reduction and clustering.

#==== 5-a. dimension reduction - PCA / umap ========================================================  
  
sc$pp$pca(ad)  
sc$pp$neighbors(ad)  
sc$tl$leiden(ad)  
sc$tl$umap(ad)

Although most proteomic, metabelomic and lipidomic data has differnetial calculations at the output of the instrumentation (which leverages know statastical assumptions of the quantifications) we can also use scanpys tools to compute differentalial expression. The diff\_exp tables will be needed for volcano plots either way.

#==== 6. differential expression =====================================================================  
sc <- import("scanpy")  
test\_types <- c('wilcoxon','t-test\_overestim\_var')  
comp\_types <- c("grpVrest")  
obs\_names <- c('disease','cell\_type')  
diff\_exp <- omicser::compute\_de\_table(ad,comp\_types, test\_types, obs\_names)

Finally we write this the anndata file to our database location specified in the omxr\_options.yml.

#==== 8. write data file to load =========================================================================  
ad$write\_h5ad(filename=file.path(DS\_ROOT\_PATH,DB\_NAME,"omxr\_data.h5ad"))

Please refer to the full curation script for more context.

TODO: more information on the diff\_exp data schema expected by the browser.

Configuration

This involves executing a few “ingestor” helper functions, and a few choices by the ‘curator’ to tell the browser where to find the data.

Edit the .yml or better yet include it in the curation script. e.g. part 7

Finally we need to define the configuration. Most of these fields *could* be inferred from the anndata file, but this is where curation is important. Lets choose the most reasonable quantities *only*.

#==== 7. create configs =========================================================================  
# differentials #if we care we need to explicitly state. defaults will be the order...  
conf\_list <- list(  
 x\_obs = c("Is.Reference","Condition","Replicate", "Label"),  
 y\_obs = c("expr\_var", "expr\_mean", "expr\_frac", "sample\_ID", "leiden"), #MEASURES  
 obs\_groupby = c("Is.Reference","Condition","Replicate", "Label"),  
 obs\_subset = c("Is.Reference","Condition","Replicate", "Label"),  
  
 x\_var = character(0),  
 y\_var = c("expr\_geomean", "expr\_mean", "expr\_var", "expr\_frac" ),  
  
 var\_groupby = character(0),  
 var\_subset = character(0),  
  
 diffs = list(diff\_exp\_comps = levels(factor(diff\_exp$versus)),  
 diff\_exp\_comp\_type = levels(factor(diff\_exp$comp\_type)), #i don't think we need this  
 diff\_exp\_obs\_name = levels(factor(diff\_exp$obs\_name)),  
 diff\_exp\_tests = levels(factor(diff\_exp$test\_type))  
 ),  
  
 layers = c("X","raw","X\_is\_scaled\_na\_to\_0","scaled","zro\_na"),  
  
 # Dimred  
 dimreds = list(obsm = ad$obsm\_keys(),  
 varm = ad$varm\_keys()),  
  
 # what ad$obs do we want to make default values for...  
 # # should just pack according to UI?  
 default\_factors = c("Condition","Color","Replicate")  
  
)  
configr::write.config(config.dat = conf\_list, file.path = file.path(DS\_ROOT\_PATH,DB\_NAME,"config.yml" ),  
 write.type = "yaml", indent = 4)

Browse!!

Assuming you have already loaded the omicser package, once the .yml files have been generated and teh data placed in the right directories you are good to browse!

run\_app(options = list(launch.browser = TRUE))

Note, that if you are using a development setup, you might want to run the run\_dev script which will handle unloading / re-loading for you.

Background (etc.)

The NDCN Browser (Omicser) was developing using {golem}: “an opinionated framework for building production-grade shiny applications.” - https://github.com/ThinkR-open/golem

Caveats

Several “big” updates are planned.

• Consistent roxygen2 headers. Currently importing functions (e.g. require(“package”)) or using the package::function syntax is inconsistent.

• Migrate all dplyr:: tidyverse table manipulation to data.frame syntax and/or dtdplyr.  
(test speed improvements on mac, with caveat of apple disabled multithreading)

• Clean up directory structures Remove /data and move more things to the /inst subdirectory

• migration to the NDCN githyb. Currently things are in my personal github (“ergonyc”)