Lab presentation
-
SingleCellViz, a R Shiny App

Valentine Gilbart

17/05/2024

Current use of scRNA-seq datasets at the lab

More than 30 scRNA-seq samples generated from the lab

Are they used up to their potential?

- cleaned object with validated annotation
- basic exploration without coding
- easily accessible to all

→ Create lab's scRNA-seq library accessible online

Online single cell visualizer/explorer

Single Cell Expression Atlas

https://www.ebi.ac.uk/gxa/sc/home

All data are re-analysed using their methods
→ won't be the exact same results as the published study

Single Cell PORTAL

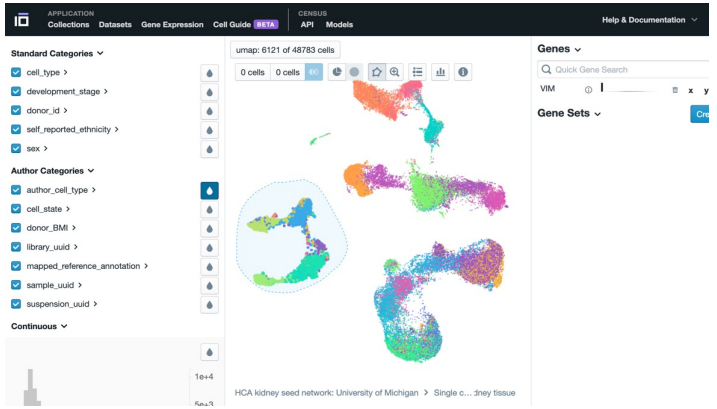
Reducing barriers and accelerating single-cell research

https://singlecell.broadinstitute.org/single_cell

Account (google) dependent to download and upload data

CZ CELL×GENE DISCOVER

https://cellxgene.cziscience.com/



Discuss with them whether or not they accept the data

Explorer with super nice features, but:

- we don't have the full hand on the site
- you are not guided through the results of the study
- we wanted to highlight the lab's studies

Aim of the app

Create a web application to quickly and dynamically:

- **Visualize**
- **Explore**
- **Highlight**
- **Download**

the results of the lab's scRNA-seq datasets

→ Maximize the use of already available dataset
→ Easily accessible for all
→ Visibility/showcase of the lab's studies

NOT MEANT TO:

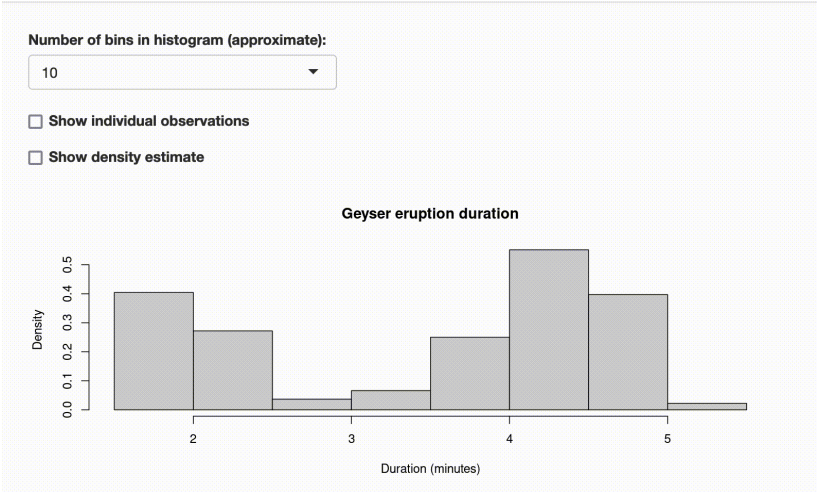
- Perform new analysis

What is Rshiny?

R package providing framework for building **interactive web applications**

The app can be:
- run on **local** device
- deployed to the **cloud** (Shinyapps.io)
- deployed to a **server** (Shiny Server)

Example of web app



It is usually separated into **two** files:
- **ui**.R: user interface, i.e. the **look** of the app
- **server**.R: server logic, i.e. the **behaviour** of the app

They are dependent of one another, as the **input of the user** (defined in the ui) must have **consequences on the output** (handled in the server)

Example of ui.R file

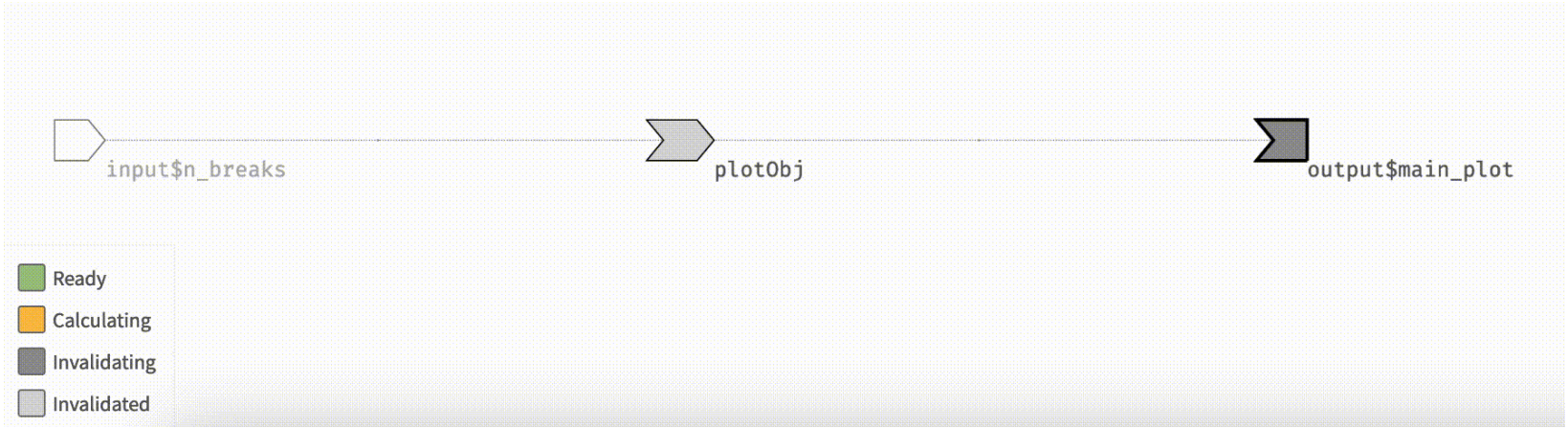
```
bootstrapPage(
  selectInput(inputId = "n_breaks",
      label = "Number of bins in histogram
(approximate):",
      choices = c(10, 20, 35, 50),
      selected = 20),
  …
  plotOutput(outputId = "main_plot", height
= "300px"),
)
```

Example of server.R file

```
function(input, output) {
  output$main_plot <- renderPlot({
    hist(faithful$eruptions,
      probability = TRUE,
      breaks = as.numeric(input$n_breaks),
      xlab = "Duration (minutes)",
      main = "Geyser eruption duration")
…
  })
}
```

How does Shiny work?

Example of reactivity log when to output a plot with a change of input



Example of ui.R file

```
bootstrapPage(
  selectInput(inputId = "n_breaks",
      label = "Number of bins in histogram
(approximate):",
      choices = c(10, 20, 35, 50),
      selected = 20),
  …
  plotOutput(outputId = "main_plot", height
= "300px"),
)
```

1) **Initialization** of the page: output$main_plot needs to be calculated
2) Everything is "**ready**"
3) Change in **input$n_breaks**: every object that depends on it gets invalidated
4) **Recalculation** of output$main_plot
5) Everything is "**ready**" again

→ This is what happens to every **reactive element** of the app

Example of server.R file

```
function(input, output) {
  output$main_plot <- renderPlot({
    hist(faithful$eruptions,
      probability = TRUE,
      breaks = as.numeric(input$n_breaks),
      xlab = "Duration (minutes)",
      main = "Geyser eruption duration")
…
  })
}
```

Structure of the App

Name	Last commit message	Last commit date
R6.R	Documentation: Remove TODOs	last month
_disable_autoload.R	Implement: first draft	last month
app_config.R	Initial commit	3 months ago
app_server.R	Implement: Download button	last month
app_ui.R	Implement: Download button	last month
fct_dataset.R	Implement: Handle database inside singlecell...	last month
fct_plots.R	Fix: remove VlnPlot jitter	last month
fct_tables.R	Implement: plot and table aesthetics	last month
mod_common.R	Implement: plot and table aesthetics	last month
mod_dataset.R	Implement: first draft	last month
mod_differential.R	Documentation: Remove TODOs	last month
mod_download.R	Documentation: Remove TODOs	last month
mod_dropdownmenu.R	Implement: first draft	last month
mod_explore.R	Documentation: Remove TODOs	last month
mod_homepage.R	Implement: plot and table aesthetics	last month
mod_information.R	Implement: first draft	last month
mod_markers.R	Documentation: Remove TODOs	last month
run_app.R	Bug fix: cache for server	last month

Important R files of the app

Server and ui files

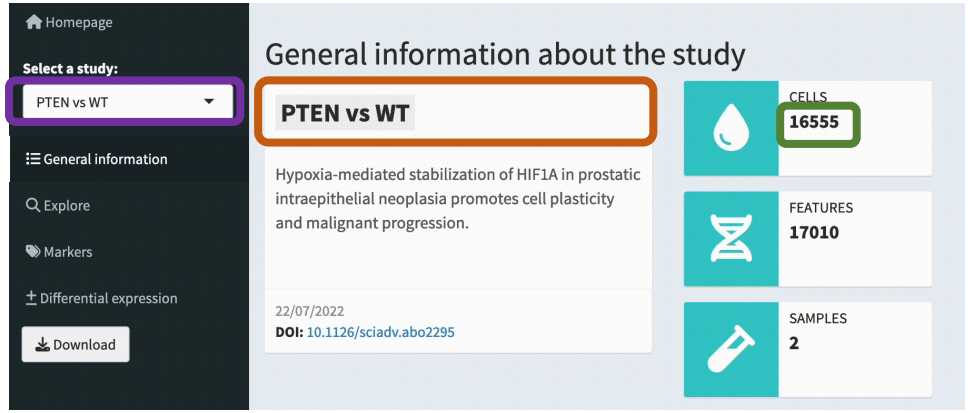
Common functions to handle dataset/plots/tables

Shiny modules (~independent parts of the app, e.g. each tab has its module)

Modules are called in the app_server.R and app_ui.R

General information tab

Screen recording of the app



The study is saved in a reactive object called **r$selected_study**, when it changes, it recalculates elements of the page (e.g. text in this example)

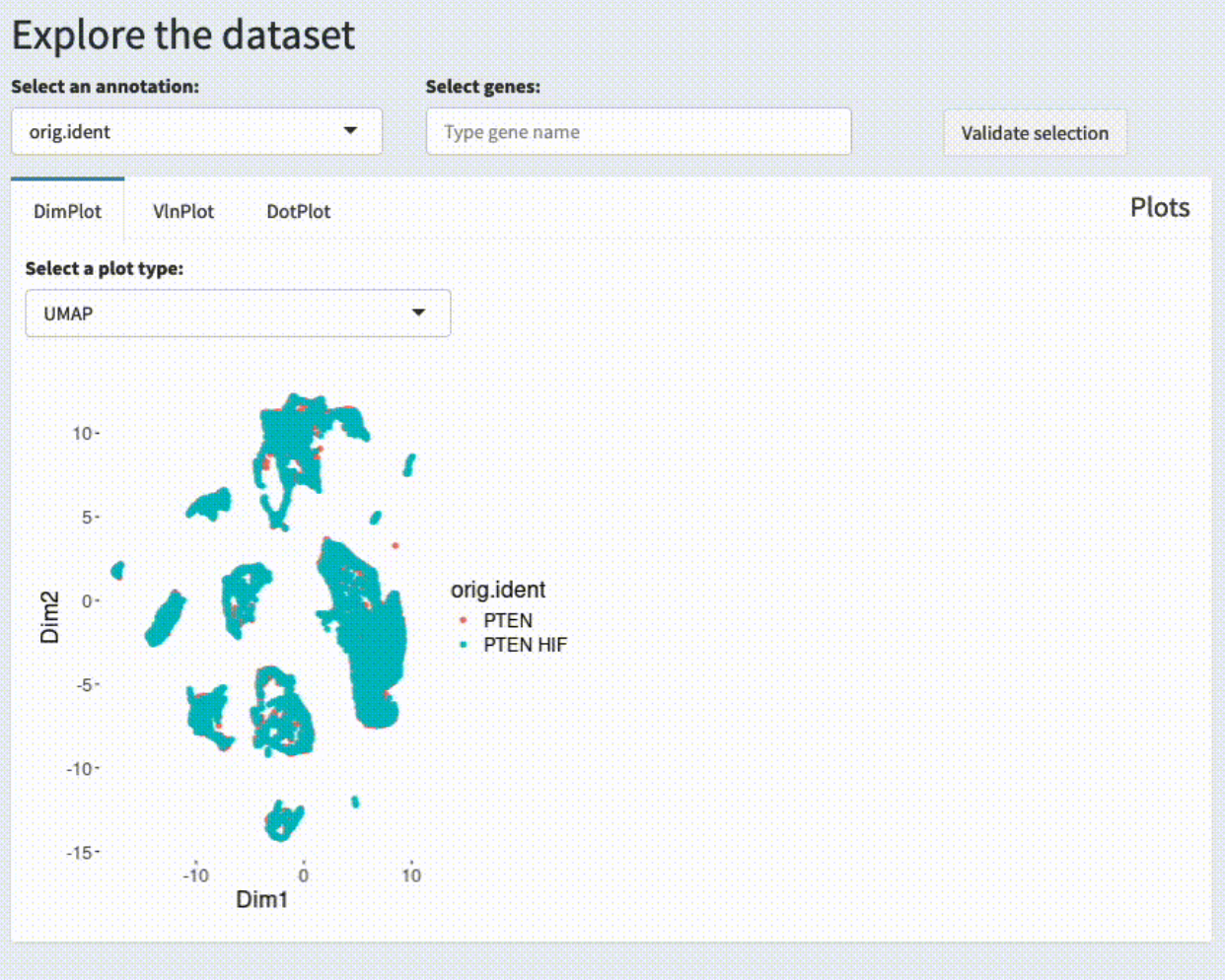
Example of ui module

```
mod_information_ui <- function(id) {
  ns <- NS(id)
  fluidRow(
    box(
      title = textOutput(ns("study_title")) %>%
tagAppendAttributes(class = 'study-title')
…
    ),
    div(
      infoBoxOutput(ns("ncells")),
…
    )
  )
}
```

Example of server module

```
mod_information_server <- function(id, COMMON_DATA, r) {
  moduleServer(id, function(input, output, session) {
    output$study_title <- renderText({
      studies[r$selected_study,, drop = F]$title
    })
…
    output$ncells <- renderInfoBox({
      ncells <- studies[r$selected_study,, drop = F]$ncells
      infoBox(
        "Cells",
        ncells,
        icon = icon("droplet"),
        color = "teal"
      )
    })
  })
}
```

Explore tab - DimPlot



Screen recording of the app

Possible:

- plot types
- annotation
- genes

are dependant on the selected study.

Explore tab - VlnPlot

Screen recording of the app

Possible:

- plot types
- annotation
- genes

are dependant on the selected study.

Explore tab - DotPlot

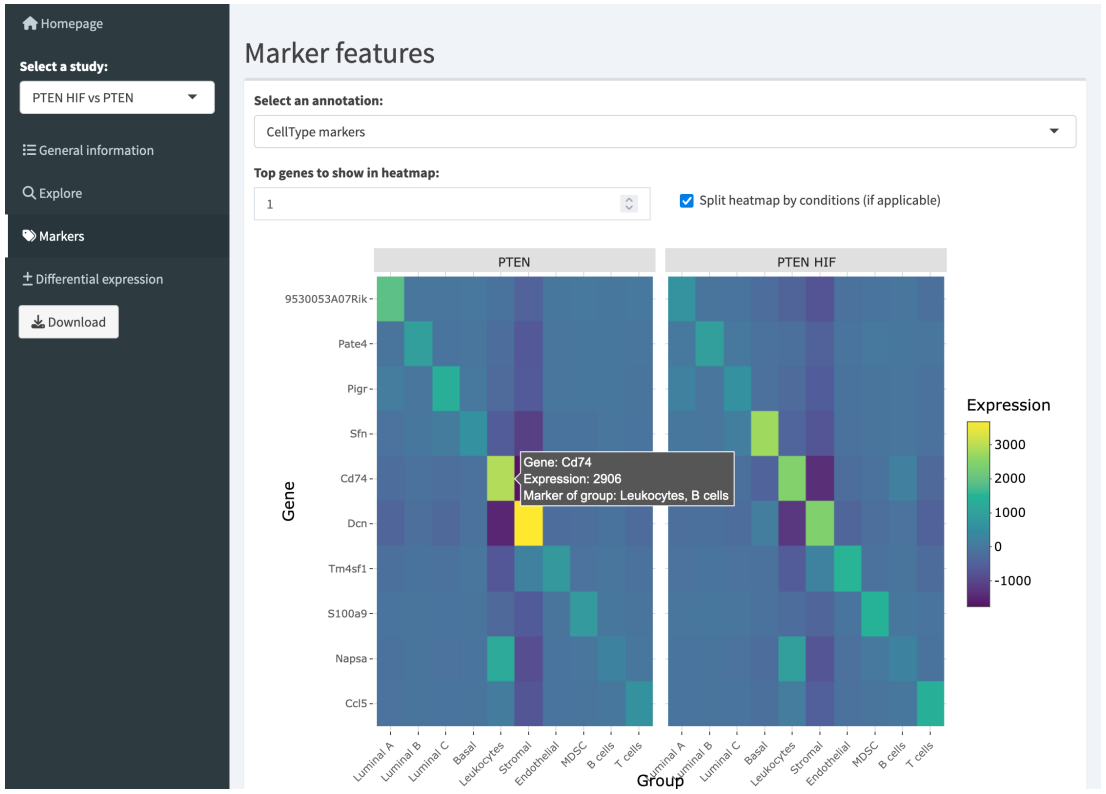
Screen recording of the app

Possible:

- plot types
- annotation
- genes

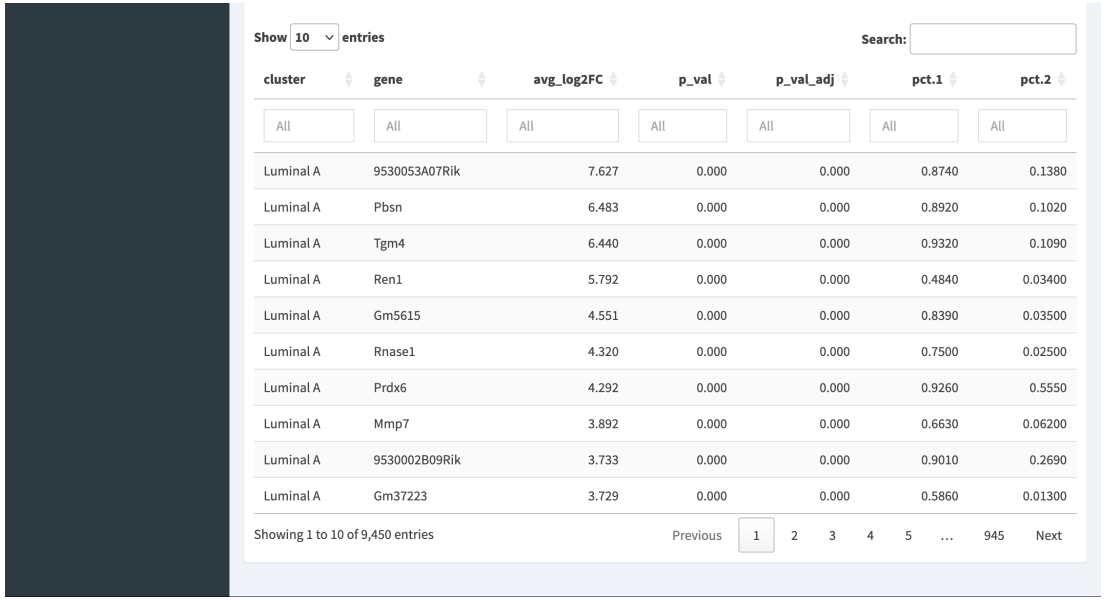
are dependant on the selected study.

Markers tab



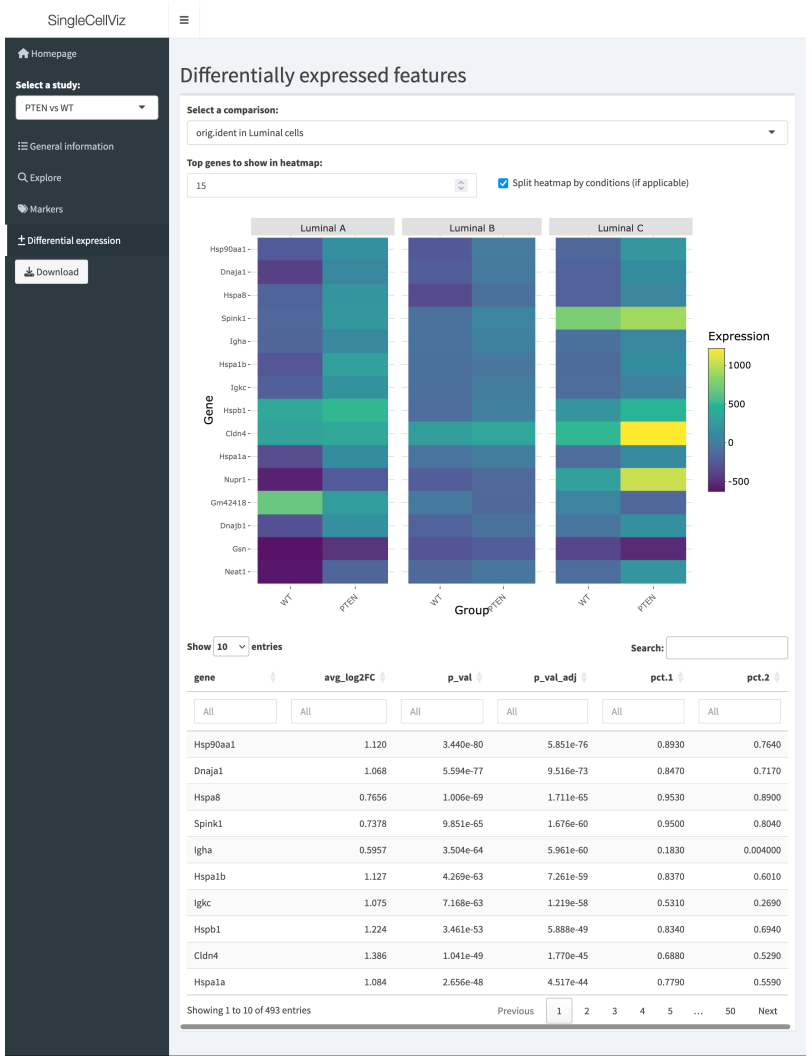
Screenshot of the app

- Table result = Seurat's **FindAllMarkers**() table
- Precalculated and saved in a database (more details later)
- Filter based on columns in the table → heatmap updates



Screenshot of the app

Differential expression tab



Screenshot of the app

Similar to previous tab:

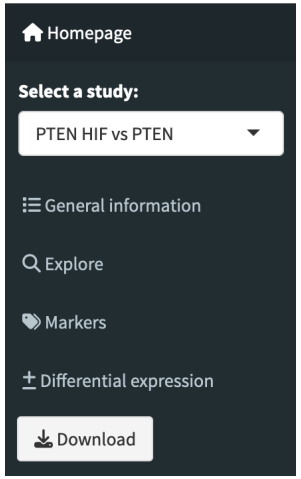
- Table result = Seurat's **FindMarkers**() table
- Precalculated and saved in a database (more details later)
- Filter based on columns in the table → heatmap updates

Download a dataset

An R object containing:

- Seurat object
- Markers table and aggregated expression
- Differential expression table and aggregated expression

can be downloaded for further exploration in R.



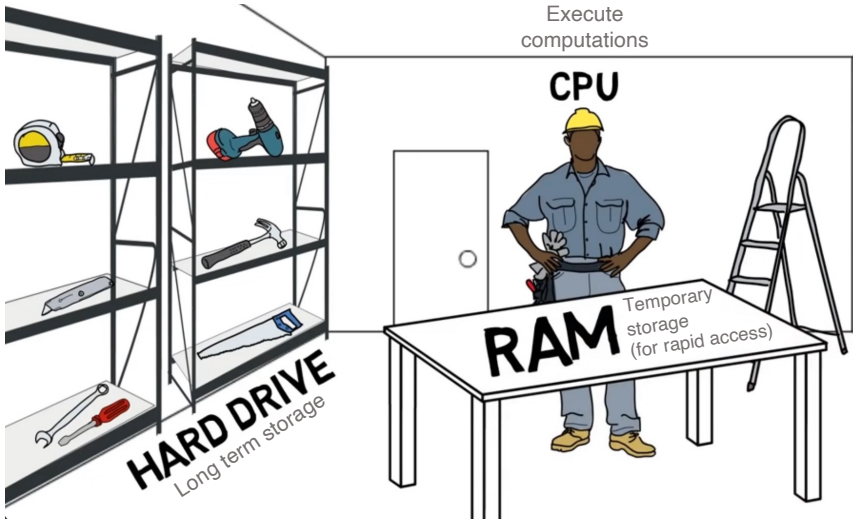
PTEN_vs_WT	list [3]	List of length 3
seuratObj	S4 [17010 x 16555] (SeuratOl	S4 object of class Seurat
assays	list [1]	List of length 1
meta.data	list [16555 x 6] (S3: data.fran	A data.frame with 16555 rows and 6...
active.assay	character [1]	'RNA'
active.ident	factor	Factor with 10 levels: "Luminal A", "L...
graphs	list [0]	List of length 0
neighbors	list [0]	List of length 0
reductions	list [3]	List of length 3
images	list [0]	List of length 0
project.name	character [1]	'SeuratProject'
misc	list [0]	List of length 0
version	list [1] (S3: package_version,	List of length 1
commands	list [9]	List of length 9
tools	list [1]	List of length 1
markers	list [2]	List of length 2
CellType markers	list [2]	List of length 2
table	list [9797 x 7] (S3: data.frame	A data.frame with 9797 rows and 7 c...
aggrexpression	list [1]	List of length 1
orig.ident markers	list [2]	List of length 2
comparison	list [1]	List of length 1
orig.ident in Lumin...	list [2]	List of length 2

Content of the downloadable R object

Deployment to a local server

With the help of the IT team, we now have a server with:

- 8 Go hard disk drive
- 4 Go RAM
- 1 vCPU
- Admin rights to Gilles and myself



Analogy of CPU, RAM, hard drive

Currently deployed to IGBMC's internal network + VPN

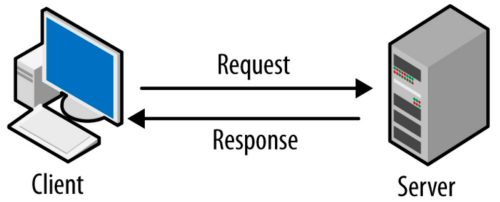
This also means that we could create other apps, or a static website if we wish ☺

The app is at the address: https://metzger-chambon.igbmc.science/SingleCellViz/

Everything that the app needs to run is on this server: the datasets, the R scripts, the R libraries…

Cache

Requesting a database, and parsing the data takes a couple of seconds…

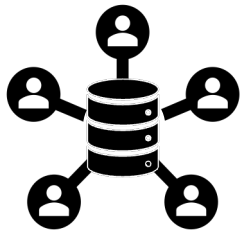


Some of the data are always needed (e.g. UMAP), or can often be requested (genes of interest).

→ Use cache to reduce waiting time!

Records output without recomputing, if it had the same keys (input, date)
In practice, it saves a R object and reloads it when needed

- Shared across users
- Fixed total size (default 200 Mo)
- Automatically removes least recently used data if needed



Server limitations

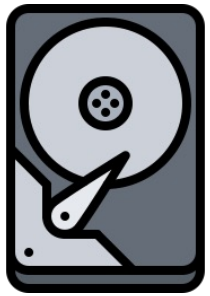
Server's RAM: 4 Go

No user: ~ 600 Mo
Launch Shiny: ~ 1 Go

+ Data per user: ~ 200 to 400 Mo

→ Limited to ~ 5 to 10 **simultaneous** users

Server's hard disk: 8 Go

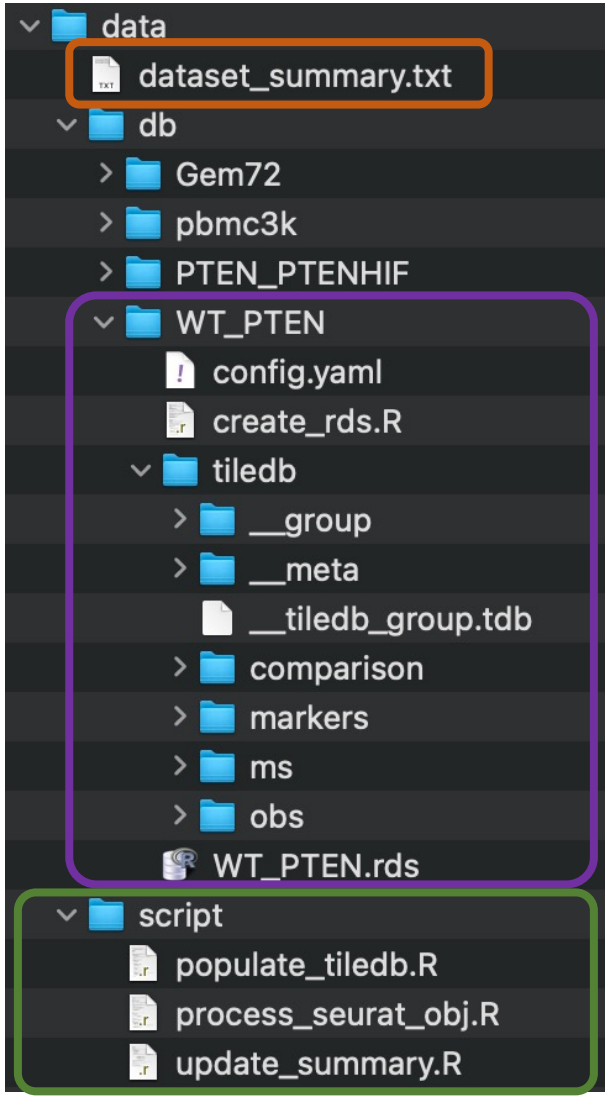


app scripts ~ 500 Ko
cache ~ 200 Mo
R libs ~ 1 Go

1 dataset:
~ 15 000 cells
~ 15 000 features
~ 400 Mo

→ Limited to ~15 datasets of similar size

Structure of the database



App data folder

Text file summarizing basic information about datasets
(~ what is in the “general information” tab, and paths to each dataset objects)

A dataset in the database

- Config file (more details later)
- Script ran to filter/clean the original Seurat object

- Contains differential expression table and aggrexpression
- Contains markers table and aggrexpression
- Contains individual measurements (~ Seurat’s RNA assay + reductions)
- Contains primary annotation (~ Seurat’s metadata)
- R object ready-to-download

Useful scripts to manage the database:

- add datasets to the database
- process a Seurat object to be added to the database
- update the dataset_summary.txt file

TileDB-SOMA

[tile]DB

- TileDB = engine for **storing** and **accessing** dense and sparse multi-dimensional array
- SOMA (Stack Of Matrices, Annotated) = specific implementation for single-cell genomic data
- Python and **R APIs**

Doc was not very user friendly… but the package is great as it allows to not load the full object, but only parts of it.

```
 gene_annotation <- reactive({
      var_query <- tiledbsoma::SOMAAxisQuery$new(
        value_filter = paste("var_id %in% c('",
paste(input$gene_annotation, collapse = "', '", sep
= ''))

      expt_query <-
tiledbsoma::SOMAExperimentAxisQuery$new(
        experiment, group, # group = RNA, SCT,
integrated…
        var_query = var_query
      )

      expt_query <- expt_query$to_sparse_matrix(
        collection = "X",
        layer_name = array, # array = scale_data,
data, counts…
        obs_index =  "obs_id",
        var_index = "var_id"
      )

      gene_annotation <- expt_query %>% as.matrix()
%>% as.data.frame()

      return(gene_annotation)
    })
```

Example of how to retrieve expression of a gene for all cells

Add a dataset to the database

- Step 1: Clean your Seurat object
  - Metadata: convert characters to factors, give clear label names, reorder levels if needed, remove unnecessary metadata
  - Lighten the object: remove unused assays, remove unnecessary reductions, remove scale.data

- Step 2: Create markers and differential expression of interest
  - Run a script with parameters of interest

- Step 3: Add the dataset to the database
  - Save a R object containing the Seurat object, markers and differential expression (ready-to-download object)
  - Add a config.yaml for the dataset
  - Run a script to add this dataset to the database
  - Update the dataset_summary.txt file

Save all that you did in a R script to be able to reproduce it, and remember the parameters.

```yaml
title: PTEN HIF vs PTEN
rds: data/db/PTEN_PTENHIF/PTEN_PTENHIF.rds
output_folder: data/db/PTEN_PTENHIF/tiledb
description: 'Hypoxia-mediated stabilization
of HIF1A in prostatic intraepithelial
neoplasia promotes cell plasticity and
malignant progression.'
doi: 10.1126/sciadv.abo2295
date: 22/07/2022
force: true
```

Example of a config file of a dataset

Version control with git

Git = version control **system**, i.e. **track changes** made to the code over time

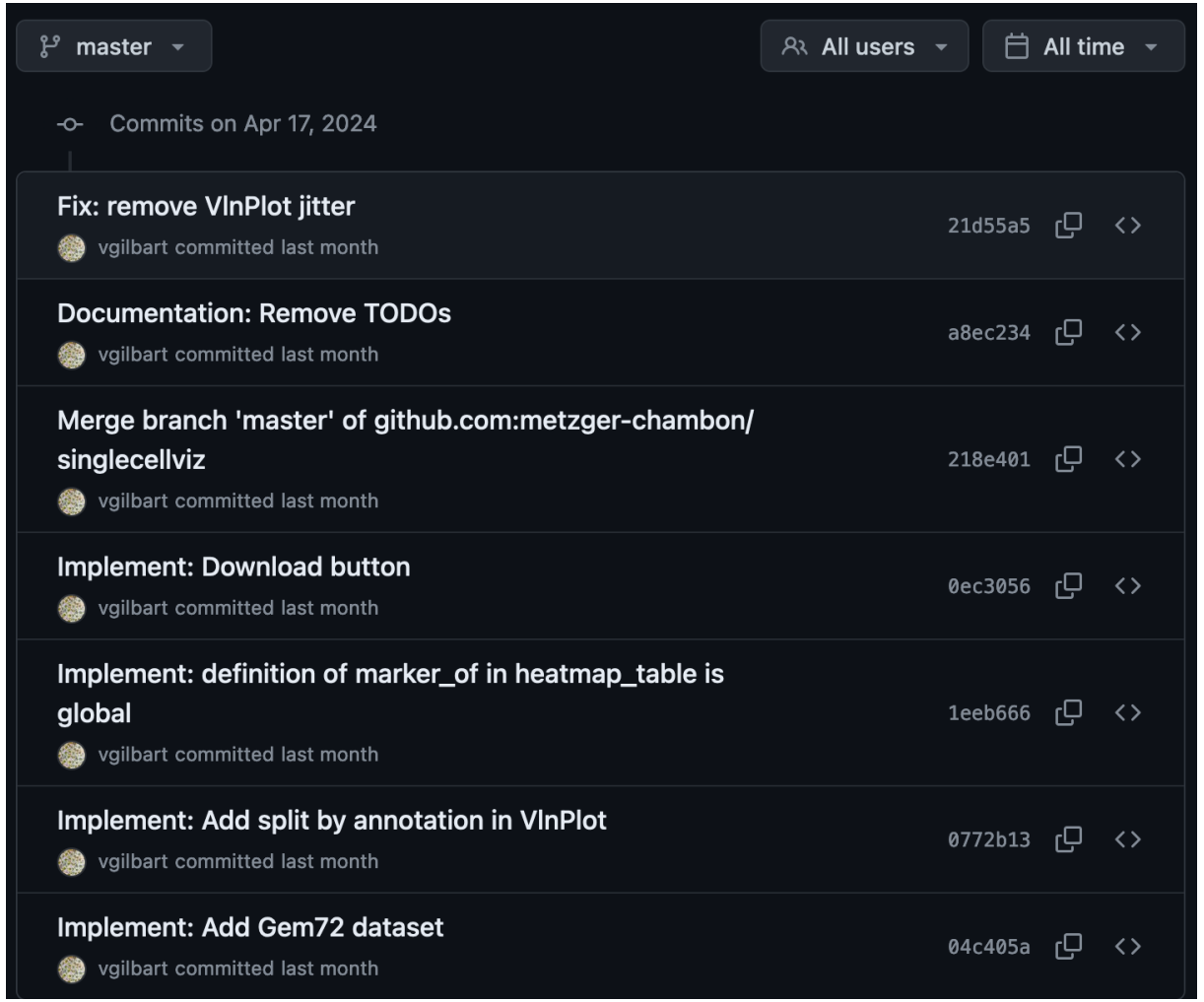
Advantages:

- Revert/compare to **previous versions**
- Understand **who** made changes and **why** (with commit messages)
- **Managing conflicts** when different changes were made to the code

Github = git **platform** service to create, store and manage code

Advantages:

- Create, assign, and track **issues, bugs, and feature requests**
- **Share** code to collaborate

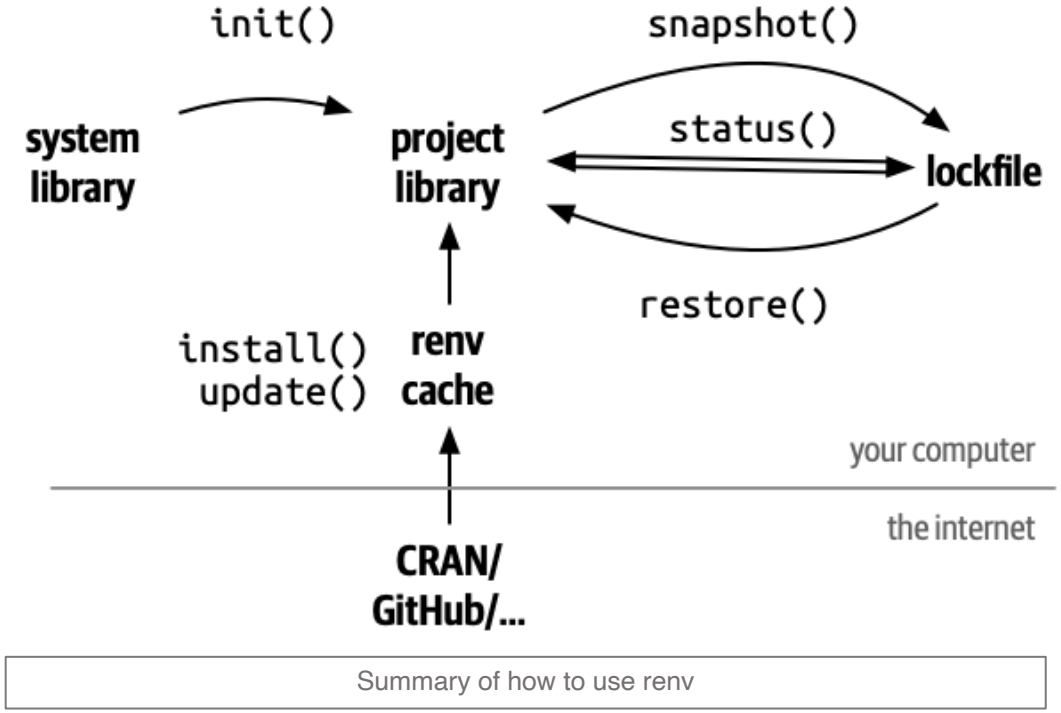


App github history

renv

renv package = create reproducible environments for R projects

Project library = each project has its own independent collection of packages at specific versions



Summary of how to use renv

App developed locally on my Mac...
But deployed on a Linux server
→ renv eased the download of R packages with the same versions!

Can also be used to:

- facilitate collaborations
- reproduce your analysis in a couple of years
- Work on different projects, with different R packages versions

Limitation: does not deal with R versions

Unit testing

testthat package = unit testing package for R

Unit testing = systemic method to test isolated snippet of

How it works?

- You create functions to quickly in/validate expected behaviours
- When developing, you automatically detect bugs and breaking changes by running existing tests

Why?

- Create robust code (for future you, or for collaborations)
- Modify functions without testing edge cases manually

```
test_that("clean_result works", {
  table_FindMarkers <- # some example data.table
  table_FindAllMarkers <- # some other example of
data.table

  # Rune the functions that we are testing
  clean_FindMarkers <-
clean_findmarkers_result(table_FindMarkers)
  clean_FindAllMarkers <-
clean_findmarkers_result(table_FindAllMarkers)

  # Check if the returned object is of expected class
  expect_s3_class(clean_FindMarkers, "tbl_df")
  expect_s3_class(clean_FindAllMarkers, "tbl_df")

  # Check that we have the expected table columns
  expect_equal(colnames(clean_FindMarkers), c("p_val",
"avg_log2FC", "pct.1", "pct.2", "p_val_adj", "gene"))
  expect_equal(colnames(clean_FindAllMarkers), c("p_val",
"avg_log2FC", "pct.1", "pct.2", "p_val_adj", "cluster",
"gene"))

  # Check that we have the expected table dimensions
  expect_equal(dim(clean_FindMarkers), c(6, 6))
  expect_equal(dim(clean_FindAllMarkers), c(6, 7))
})
```

Example of a unit testing

Conclusion, limitations and perspectives

CONCLUSION:

- Available internally at: https://metzger-chambon.igbmc.science/SingleCellViz/
- Quickly download, visualize, explore and highlight results of publicated dataset

LIMITATIONS:

- Limited simultaneous users (RAM and CPU)
- Limited number of datasets to explore (hard storage)
- Not an analysis tool!

PERSPECTIVES:

- Write documentation for app and server usage
- Deploy externally (after internal beta-testing)
- Add (publicated?) datasets

Thank you ☺

Suppl: golem

4 Introduction to {golem}

https://engineering-shiny.org/

https://thinkr-open.github.io/golem/

golem is a **toolkit for simplifying the creation, development and deployment of a shiny application**. It focuses on building applications that will be sent to production, but of course starting with golem from the very beginning is also possible, even recommended: it is easier to start with golem than to refactor your codebase to fit into the framework.

```
golex
├── DESCRIPTION
├── NAMESPACE
├── R
│   ├── app_config.R
│   ├── app_server.R
│   ├── app_ui.R
│   └── run_app.R
├── dev
│   ├── 01_start.R
│   ├── 02_dev.R
│   ├── 03_deploy.R
│   └── run_dev.R
├── inst
│   ├── app
│   │   └── www
│   │       └── favicon.ico
│   └── golem-config.yml
└── man
    └── run_app.Rd
```

Suppl: Shiny best practices

Table of contents

https://mastering-shiny.org/

Suppl: