Creating a directory with data for the Shiny app

Fabricio Almeida-Silva

2023-04-28

```r
set.seed(123) # for reproducibility

# Load required packages
library(here)
library(SummarizedExperiment)
library(arrow)
library(tidyverse)
library(rvest)
```

Overview

Here, we will describe the code to create the files that will be required to run the Shiny app. These files will be stored in a directory named `app_data`.

.parquet files in `parquet_dir`

Gene-level transcript abundances in TPM and bias-corrected counts will be stored in a partitioned *.parquet* directory, so that expression data can be accessed in the app back-end with Apache Arrow via the `BiocStyle::CRANpkg("arrow")` package.

This directory contains partitioned *.parquet* files with a gene expression data frame in long format with the following variables:

1. `Gene`: character, gene ID.
2. `Sample`: character, sample name.
3. `TPM`: numeric, gene-level transcript abundances in TPM.
4. `Count`: numeric, gene-level transcript abundances in bias-corrected counts.
5. `BioProject`: factor, BioProject IDs.
6. `Part`: character, plant part.

```r
# Load SummarizedExperiment object
load(here("products", "result_files", "se_atlas_gene.rda"))

# Get expression data in long format
## TPM
exp_tpm <- assay(se_atlas_gene, "gene_TPM") |>
    reshape2::melt() |>
    mutate(
        Gene = as.character(Var1),
        Sample = as.character(Var2),
        TPM = as.numeric(value)
    ) |>
    dplyr::select(Gene, Sample, TPM)
```

```r
hist(log_sorted_tpm)

## Counts
exp_counts <- assay(se_atlas_gene, "gene_counts") |>
    reshape2::melt() |>
    mutate(
        Gene = as.character(Var1),
        Sample = as.character(Var2),
        Count = as.numeric(value)
    ) |>
    dplyr::select(Gene, Sample, Count)

names(exp_counts) <- c("Gene", "Sample", "Count")

## Combine data frames
identical(exp_counts$Sample, exp_tpm$Sample)
identical(exp_counts$Gene, exp_tpm$Gene)
exp_final <- cbind(exp_tpm, exp_counts[, "Count", drop = FALSE])

# Export data with BioProject and Part info
sample_metadata <- colData(se_atlas_gene) |>
    as.data.frame() |>
    tibble::rownames_to_column("BioSample")

sample_and_additional_info <- data.frame(
    Sample = sample_metadata$BioSample,
    BioProject = sample_metadata$BioProject,
    Part = sample_metadata$Part
)

exp_final2 <- left_join(
    exp_final,
    sample_and_additional_info
) |>
    mutate(
        BioProject = as.factor(BioProject),
        Sample = as.factor(Sample),
        BioProject = as.factor(BioProject),
        Part = as.factor(Part)
    )

parquet_dir_partitioned <- here("app_data", "parquet_dir")
fs::dir_create(parquet_dir_partitioned)

arrow::write_dataset(
    exp_final2,
    path = parquet_dir_partitioned,
    format = "parquet",
    partitioning = c("BioProject", "Part")
)
```

R objects

The following R objects are small enough to be stored as .rda files, so that they can be directly loaded when the app starts without compromising performance. These `.rda` objects will be stored in the `data/` directory of the app.

sample_metadata.rda

This file contains a data frame of sample metadata with the following fields:

1. BioProject
2. BioSample
3. Part
4. Pubmed
5. Study_title
6. Study_abstract

This file was generated with the following code:

```r
# Get a data frame of sample metadata
sample_metadata <- as.data.frame(colData(se_atlas_gene)) |>
    tibble::rownames_to_column("BioSample") |>
    dplyr::select(
        BioProject, BioSample, Part, Pubmed,
        Study_title, Study_abstract
    )

# Save to file
save(
    sample_metadata, compress = "xz",
    file = here("app_data", "sample_metadata.rda")
)
```

project_metadata.rda

This object is similar to `sample_metadata.rda`, but it stores metadata at the BioProject level.

```r
#' Create a project table to display in the "Search by project" tab
#'
#' @param metadata Data frame of sample metadata.
#'
#' @return A data frame with the variables:
#' \itemize{
#'   \item
#'   \item
#' }
#' @importFrom dplyr add_count select rename distinct group_by filter
#' summarise arrange
#' @importFrom stringr str_c
#' @noRd
create_project_table <- function(metadata = NULL) {
    table <- metadata %>%
        dplyr::filter(startsWith(BioProject, "PRJ")) %>%
        dplyr::add_count(BioProject) %>%
        dplyr::select(BioProject, n, Study_title, Study_abstract) %>%
        dplyr::rename(
            N = n,
            `Study title` = Study_title,
            `Study abstract` = Study_abstract
        ) %>%
        dplyr::distinct()

    tissue_count <- metadata %>%
        dplyr::filter(startsWith(BioProject, "PRJ")) %>%
        group_by(BioProject, Part) %>%
        summarise(n = n()) %>%
        ungroup() %>%
        arrange(-n) %>%
        group_by(BioProject) %>%
        summarise(part_count = stringr::str_c(
            Part, ": ", n, collapse = " | ")
        )

    final_table <- dplyr::inner_join(
        table, tissue_count, by = "BioProject"
    ) %>%
        dplyr::rename(Part = part_count) %>%
        dplyr::select(
            BioProject, N, Part, `Study title`, `Study abstract`
        )
    return(final_table)
}

# Combine sample metadata into project metadata
project_metadata <- create_project_table(sample_metadata)

# Create a data frame with PMID and DOI of publications associated with projects
all_bioprojects <- unique(project_metadata$BioProject)
pub_info <- Reduce(rbind, lapply(all_bioprojects, function(x) {
    message(x)
    pubs <- read_html(
        paste0("https://www.ncbi.nlm.nih.gov/bioproject/?term=", x)
    ) |>
        html_nodes(".RegularLink") |>
        html_attr("href")

    # Get PMID
    pmid <- pubs[grepl("/pubmed/", pubs)]
    pmid <- unique(gsub("/pubmed/", "", pmid))

    id_table <- NULL
    if(length(pmid) != 0) {
        # Use PMID to extract DOI
        doi <- sapply(pmid, function(y) {
            d <- read_html(
                paste0("https://pubmed.ncbi.nlm.nih.gov/", y)
            ) |>
                html_nodes("a") |>
                html_attr("href")
```

```r
            d <- unique(d[grepl("doi\\.org/", d)])[1]
            return(d)
        })

        id_table <- data.frame(
            BioProject = x,
            PMID = pmid,
            DOI = doi
        )
    }

    return(id_table)
}))

pub_table <- pub_info |>
    mutate(DOI = str_replace_all(DOI, "https://doi.org/", "")) |>
    group_by(BioProject) |>
    summarise(
        DOI = paste0(DOI, collapse = ", "),
        PMID = paste0(PMID, collapse = ", ")
    ) |>
    mutate(
        DOI = as.factor(DOI),
        PMID = as.factor(PMID)
    )

pmeta <- left_join(project_metadata, pub_table) |>
    dplyr::select(
        BioProject, N, Part, `Study title`, `Study abstract`, DOI, PMID
    )

project_metadata <- pmeta

# Save object
save(
    project_metadata, compress = "xz",
    file = here("app_data", "project_metadata.rda")
)
```

genes.rda

This object contains a character vector of all genes in the Soybean Expression Atlas. Not all genes in the genome are included here, as genes with no detectable expression were not included in the expression matrix.

```r
genes <- rownames(se_atlas_gene)

save(
    genes, compress = "xz",
    file = here("app_data", "genes.rda")
)
```

gene_descriptions.rda

This file contains a 2-column data frame with genes and their short descriptions. Descriptions will be obtained from PLAZA Dicots 5.0.

```r
# Create a data frame of all genes
genes_df <- data.frame(
    Gene = sort(genes)
)

# Get descriptions from PLAZA Dicots 5.0
gene_descriptions <- read_tsv(
    file.path(
        "https://ftp.psb.ugent.be/pub/plaza/",
        "plaza_public_dicots_05/Descriptions/gene_description.gma.csv.gz"
    ),
    show_col_types = FALSE, skip = 8
) |>
    select(
        Gene = `#gene_id`, Description = id
    ) |>
    mutate(Description = str_replace(Description, ".* - ", "")) |>
    right_join(genes_df) |>
    arrange(Gene)

# Save object
save(
    gene_descriptions, compress = "xz",
    file = here("app_data", "gene_descriptions.rda")
)
```

tsne_coordinates.rda

This object contains t-SNE coordinates in a data frame with the following variables:

1. `tSNE1`: numeric, x-axis coordinates.
2. `tSNE2`: numeric, y-axis coordinates.
3. `BioSample`: factor, BioSample ID.
4. `Part`: factor, plant part.

```r
# Load tSNE plot
load(here("products", "plots", "p_tsne_optimal_perplexity.rda"))

# Create data frame
tsne_coordinates <- p_tsne_optimal_perplexity$data |>
    tibble::rownames_to_column("BioSample") |>
    rename(
        tSNE1 = X,
        tSNE2 = Y,
        Part = colour_by
    ) |>
    select(tSNE1, tSNE2, BioSample, Part) |>
    mutate(
        Part = str_to_title(Part),
        Part = as.factor(Part),
        BioSample = as.factor(BioSample)
    )

# Save object
save(
    tsne_coordinates, compress = "xz",
    file = here("app_data", "tsne_coordinates.rda")
)
```

Session information

This document was created under the following conditions:

```r
sessioninfo::session_info()
```

```
## - Session info ---------------------------------------------------------------
##  setting  value
##  version  R version 4.3.0 (2023-04-21)
##  os       Ubuntu 20.04.5 LTS
##  system   x86_64, linux-gnu
##  ui       X11
##  language (EN)
##  collate  en_US.UTF-8
##  ctype    en_US.UTF-8
##  tz       Europe/Brussels
##  date     2023-04-28
##  pandoc   2.19.2 @ /usr/lib/rstudio/resources/app/bin/quarto/bin/tools/ (via rmarkdown)
##
## - Packages -------------------------------------------------------------------
##  package     * version date (UTC) lib source
##  cli           3.6.1   2023-03-23 [1] CRAN (R 4.3.0)
##  digest        0.6.31  2022-12-11 [1] CRAN (R 4.3.0)
##  evaluate      0.20    2023-01-17 [1] CRAN (R 4.3.0)
##  fastmap       1.1.1   2023-02-24 [1] CRAN (R 4.3.0)
##  htmltools     0.5.5   2023-03-23 [1] CRAN (R 4.3.0)
##  knitr         1.42    2023-01-25 [1] CRAN (R 4.3.0)
##  rlang         1.1.0   2023-03-14 [1] CRAN (R 4.3.0)
##  rmarkdown     2.21    2023-03-26 [1] CRAN (R 4.3.0)
##  rstudioapi    0.14    2022-08-22 [1] CRAN (R 4.3.0)
##  sessioninfo   1.2.2   2021-12-06 [1] CRAN (R 4.3.0)
##  xfun          0.39    2023-04-20 [1] CRAN (R 4.3.0)
##  yaml          2.3.7   2023-01-23 [1] CRAN (R 4.3.0)
##
##  [1] /home/faalm/R/x86_64-pc-linux-gnu-library/4.3
##  [2] /usr/local/lib/R/site-library
##  [3] /usr/lib/R/site-library
##  [4] /usr/lib/R/library
##
## ------------------------------------------------------------------------------
```