TF Classification

Arend Vancraeynest

2022-03-24

```
library(here)
```

Overview

Here, you must describe the goal of the analyses documented in this documented.

Proteome metadata

Ready the data for R manipulation

```
library(Biostrings)
proteomes <- list.files(here("data", "Proteomes"), full.names = TRUE)

seqs <- lapply(proteomes, readAAStringSet)

# A file containing the genome assembly versions and reference papers has been made manually
extra_data <- read.table(here("data", "Genome_references.txt"), header = FALSE, sep = "\t")
```

Generate a summary dataframe

```
# Initiate the dataframe based on the species names contained within the file names
# (genus abbreviated in 3 or 4 letters, species epithet in full, divided by a '.')
df <- data.frame(Species_name=gsub(".*/([a-z]\\.[a-zA-Z]+)_((cv|ssp)\\.[a-z0-9-]+)?.*", "\\1 \\2", prote
df$Species_name <- gsub("\\.", ". ", df$Species_name)

# Add the Genome Assembly version
df$Genome_assembly_version <- extra_data$V1

# Count the number of proteins and genes for each species and append to an initiated dataframe
check_isoforms <- function(seqlist = NULL) {
  combined_list <- lapply(seq_along(seqlist), function(x){
    all_seqs <- names(seqlist[[x]])
    proteins <- length(all_seqs)

    # Substring the unique gene names from the different types of annotation
    unique_seqs <- gsub("^.*gene=", "", all_seqs, ignore.case = TRUE)
    unique_seqs <- gsub("\\s.*$", "", unique_seqs)
    unique_seqs <- gsub("\\.t?[0-9]+(.p)?$", "", unique_seqs)

    # Only retain unique gene handles
    unique_seqs <- unique(unique_seqs)
    genes <- length(unique_seqs)

    return(c(proteins, genes))
  }
  )
  df$Coding_genes <<- sapply(combined_list, "[[", 2)
  df$Annotated_proteins <<- sapply(combined_list, "[[", 1)
}

check_isoforms(seqs)

# Add the reference papers
df$References <- extra_data$V2

# Tidy up the dataframe to make it more reader-friendly by replacing underscores by spaces in the heade
names(df) <- gsub("_", " ", names(df))
```

Clean up the protein FASTA files

```
library(tidyverse)

primary_transcripts <- function(seq_stringset = NULL){
  n <- names(seq_stringset)

  # For each transcript, find the encoding gene
  unique_seqs <- gsub("^.*gene=", "", n, ignore.case = TRUE)
  unique_seqs <- gsub("\\s.*$", "", unique_seqs)
  unique_seqs <- gsub("\\.t?[0-9]+(.p)?$", "", unique_seqs)

  # Make a dataframe with the transcript names, gene names and sequence lengths
  # group by the gene names and only keep the longest sequence (which should by the primary)
  df_clean <- data.frame(names = n, clean_names = unique_seqs, len = width(seq_stringset)) %>%
    group_by(clean_names) %>%
    filter(len == max(len))

  # In some cases transcripts are duplicated (different id, but same length & sequence) --> keep only t
  df_clean <- df_clean[!duplicated(df_clean$clean_names), ]

  # Return the AAStringset datatype filtered by the remaining names
  return(seq_stringset[df_clean$names])
}


# If the count of genes and proteins is not equal (aka more transcripts per gene instead of just the pr
# apply the primary_transcript function. If equal: just copy the object
clean_seqlist <- lapply(seq_along(seqs), function(x){
  if(df[x,]$'Coding genes' != df[x,]$'Annotated proteins'){
```

```r
    primary_transcripts(seqs[[x]])
  } else {seqs[[x]]
  }
})
```

Session info

This document was created under the following conditions:

```r
sessionInfo()
```

```
## R version 4.1.2 (2021-11-01)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 22000)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=Dutch_Belgium.1252  LC_CTYPE=Dutch_Belgium.1252
## [3] LC_MONETARY=Dutch_Belgium.1252 LC_NUMERIC=C
## [5] LC_TIME=Dutch_Belgium.1252
##
## attached base packages:
## [1] stats4    stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] forcats_0.5.1       stringr_1.4.0       dplyr_1.0.8
##  [4] purrr_0.3.4         readr_2.1.2         tidyr_1.2.0
##  [7] tibble_3.1.6        ggplot2_3.3.5       tidyverse_1.3.1
## [10] Biostrings_2.62.0   GenomeInfoDb_1.30.1 XVector_0.34.0
## [13] IRanges_2.28.0      S4Vectors_0.32.3    BiocGenerics_0.40.0
## [16] here_1.0.1
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.8.3           lubridate_1.8.0        assertthat_0.2.1
##  [4] rprojroot_2.0.2        digest_0.6.29          utf8_1.2.2
##  [7] R6_2.5.1               cellranger_1.1.0       backports_1.4.1
## [10] reprex_2.0.1           evaluate_0.15          httr_1.4.2
## [13] pillar_1.7.0           zlibbioc_1.40.0        rlang_1.0.2
## [16] readxl_1.3.1           rstudioapi_0.13        rmarkdown_2.13
## [19] RCurl_1.98-1.6         munsell_0.5.0          broom_0.7.12
## [22] compiler_4.1.2         modelr_0.1.8           xfun_0.30
## [25] pkgconfig_2.0.3        htmltools_0.5.2        tidyselect_1.1.2
## [28] GenomeInfoDbData_1.2.7 fansi_1.0.2            withr_2.5.0
## [31] crayon_1.5.0           tzdb_0.2.0             dbplyr_2.1.1
## [34] bitops_1.0-7           grid_4.1.2             jsonlite_1.8.0
## [37] gtable_0.3.0           lifecycle_1.0.1        DBI_1.1.2
## [40] magrittr_2.0.2         scales_1.1.1           cli_3.2.0
## [43] stringi_1.7.6          fs_1.5.2               xml2_1.3.3
## [46] ellipsis_0.3.2         generics_0.1.2         vctrs_0.3.8
## [49] tools_4.1.2            glue_1.6.2             hms_1.1.1
## [52] fastmap_1.1.0          yaml_2.3.5             colorspace_2.0-3
## [55] rvest_1.0.2            knitr_1.37             haven_2.4.3
```