Basic usage of SPARKI!

Jacqueline M. Boccacino

2024-08-27

```
#library(SPARKI)
source("R/utilities.R")
source("R/helper.R")
source("R/plotting.R")
source("R/constants.R")
```

This tutorial will demonstrate how to run SPARKI on Kraken2 results. First of all, we will need Kraken2 reports in both standard and MPA-style formats. A standard report can be generated with the option –report when running Kraken2 on the command line; please note that the flag –report-minimizer-data must also be used. An MPA-style report can be generated when the options –report and –use-mpa-style are combined on the command line; alternatively, MPA-style reports can be generated from a standard report with the script kreport2mpa.py from the KrakenTools toolkit.

Loading Kraken2 results

The first step in the SPARKI workflow is to load the standard and MPA-style reports. In the example below, our files are located in the test directory. The functions load_MPAreports() and load_STDreports() will create dataframes containing all samples that are present in the directory we specified.

```
mpa_reports <- load_MPAreports("test/mpa", verbose = FALSE)
std_reports <- load_STDreports("test/reports", verbose = FALSE)
```

We can now inspect the dataframes that were created:

```
mpa_reports
```

```
## # A tibble: 73,290 x 12
##    sample   taxon_leaf rank  n_fragments_clade domain kingdom phylum class order
##    <chr>    <chr>      <chr>             <dbl> <chr>  <chr>   <chr>  <chr> <chr>
##  1 PR42171a Eukaryota  D              17329988 Eukar~ <NA>    <NA>   <NA>  <NA>
##  2 PR42171a Metazoa    K              16997144 Eukar~ Metazoa <NA>   <NA>  <NA>
##  3 PR42171a Chordata   P              16997144 Eukar~ Metazoa Chord~ <NA>  <NA>
##  4 PR42171a Mammalia   C              16997144 Eukar~ Metazoa Chord~ Mamm~ <NA>
##  5 PR42171a Primates   O              16997144 Eukar~ Metazoa Chord~ Mamm~ Prim~
##  6 PR42171a Hominidae  F              16997144 Eukar~ Metazoa Chord~ Mamm~ Prim~
##  7 PR42171a Homo       G              16997144 Eukar~ Metazoa Chord~ Mamm~ Prim~
##  8 PR42171a Homo sapi~ S              16997144 Eukar~ Metazoa Chord~ Mamm~ Prim~
##  9 PR42171a Fungi      K                    96 Eukar~ Fungi   <NA>   <NA>  <NA>
```

```
## 10 PR42171a Basidiomy~ P                    79 Eukar~ Fungi   Basid~ <NA>  <NA>
## # i 73,280 more rows
## # i 3 more variables: family <chr>, genus <chr>, species <chr>
```

```
std_reports
```

```
## # A tibble: 73,852 x 9
##    sample   pct_fragments_clade n_fragments_clade n_fragments_taxon n_minimisers
##    <chr>                  <dbl>             <dbl>             <dbl>        <dbl>
##  1 PR42171a                64.0          32926460          32926460            0
##  2 PR42171a                36.0          18513589             32699    202279022
##  3 PR42171a                33.7          17329988            326720    192969933
##  4 PR42171a                33.0          16997144                 0    189756889
##  5 PR42171a                33.0          16997144                 0    189756889
##  6 PR42171a                33.0          16997144                 0    189756889
##  7 PR42171a                33.0          16997144                 0    189756889
##  8 PR42171a                33.0          16997144                 0    189756889
##  9 PR42171a                33.0          16997144                 0    189756889
## 10 PR42171a                33.0          16997144          16997144    189756889
## # i 73,842 more rows
## # i 4 more variables: n_distinct_minimisers <dbl>, rank <chr>, ncbi_id <dbl>,
## #   taxon <chr>
```

Merging reports

Next, we will combine the information present in the standard and MPA-style dataframes into a single dataframe. Note that the Kraken2 results present in the different report formats are the same; however, they are represented in slightly different ways, and here we want to benefit from both types of representations. The function mergeReports() will do the merging task:

```
merged_reports <- mergeReports(std_reports, mpa_reports)
```

Let's have a look at the merged dataframe:

```
merged_reports
```

```
## # A tibble: 73,852 x 17
##    sample   pct_fragments_clade n_fragments_clade n_fragments_taxon n_minimisers
##    <chr>                  <dbl>             <dbl>             <dbl>        <dbl>
##  1 PR42171a                64.0          32926460          32926460            0
##  2 PR42171a                36.0          18513589             32699    202279022
##  3 PR42171a                33.7          17329988            326720    192969933
##  4 PR42171a                33.0          16997144                 0    189756889
##  5 PR42171a                33.0          16997144                 0    189756889
##  6 PR42171a                33.0          16997144                 0    189756889
##  7 PR42171a                33.0          16997144                 0    189756889
```

```
##  8 PR42171a                33.0          16997144                 0    189756889
##  9 PR42171a                33.0          16997144                 0    189756889
## 10 PR42171a                33.0          16997144          16997144    189756889
## # i 73,842 more rows
## # i 12 more variables: n_distinct_minimisers <dbl>, rank <chr>, ncbi_id <dbl>,
## #   taxon <chr>, domain <chr>, kingdom <chr>, phylum <chr>, class <chr>,
## #   order <chr>, family <chr>, genus <chr>, species <chr>
```

Loading metadata (optional)

Now that we have our merged dataframe, we can add sample metadata to it. This step is optional, but it can be very helpful to have additional sample information in our dataset when we interpret the final results. We can load metadata very easily by using the function loadMetadata().

```
mdata <- loadMetadata("test/metadata.csv")
```

To add metadata to our merged dataframe, we can simply use the function addMetadata(), specifying the columns that we want to add and the column that contains sample IDs in our metadata table:

```
mdata_sample_col <- "Tumour_RNA"
mdata_columns_to_add <- c("Diagnosis_short", "Site_group")

merged_reports <- addMetadata(
  merged_reports,
  mdata,
  mdata_sample_col,
  mdata_columns_to_add
)
```

If we inspect our merged dataframe again, we will see that it now contains sample metadata information:

```
merged_reports
```

```
## # A tibble: 73,852 x 19
##    sample   Diagnosis_short Site_group pct_fragments_clade n_fragments_clade
##    <chr>    <chr>           <chr>                    <dbl>             <dbl>
##  1 PR42171a SA              head&neck                 64.0          32926460
##  2 PR42171a SA              head&neck                 36.0          18513589
##  3 PR42171a SA              head&neck                 33.7          17329988
##  4 PR42171a SA              head&neck                 33.0          16997144
##  5 PR42171a SA              head&neck                 33.0          16997144
##  6 PR42171a SA              head&neck                 33.0          16997144
##  7 PR42171a SA              head&neck                 33.0          16997144
##  8 PR42171a SA              head&neck                 33.0          16997144
##  9 PR42171a SA              head&neck                 33.0          16997144
## 10 PR42171a SA              head&neck                 33.0          16997144
## # i 73,842 more rows
## # i 14 more variables: n_fragments_taxon <dbl>, n_minimisers <dbl>,
## #   n_distinct_minimisers <dbl>, rank <chr>, ncbi_id <dbl>, taxon <chr>,
## #   domain <chr>, kingdom <chr>, phylum <chr>, class <chr>, order <chr>,
## #   family <chr>, genus <chr>, species <chr>
```

Loading Kraken2's reference database information

Before we can start processing the Kraken2 results, the last thing we need to do is load the file inspect.txt from the Kraken2 reference database we used to generate our Kraken2 reports:

```
ref_db <- loadReference("test/inspect.txt")
```

Processing Kraken2 results

Now that all data is ready, we can start processing and visualising our Kraken2 results.

The initial processing of the data will be fairly simple; we will basically add a few columns to our merged dataframe:

- The function addSampleSize() will add a column with the total number of fragments that were analysed by Kraken2 per sample.
- The function addMinimiserData() will add columns with minimiser data from Kraken2's reference database.

```
merged_reports <- addSampleSize(merged_reports)
merged_reports <- addMinimiserData(merged_reports, ref_db)
```

Let's inspect the updated dataframe:

```
merged_reports
```

```
## # A tibble: 73,852 x 22
##    sample   sample_size Diagnosis_short Site_group pct_fragments_clade
##    <chr>          <dbl> <chr>           <chr>                     <dbl>
##  1 PR42171a    51440049 SA              head&neck                  64.0
##  2 PR42171a    51440049 SA              head&neck                  36.0
##  3 PR42171a    51440049 SA              head&neck                  33.7
##  4 PR42171a    51440049 SA              head&neck                  33.0
##  5 PR42171a    51440049 SA              head&neck                  33.0
##  6 PR42171a    51440049 SA              head&neck                  33.0
##  7 PR42171a    51440049 SA              head&neck                  33.0
##  8 PR42171a    51440049 SA              head&neck                  33.0
##  9 PR42171a    51440049 SA              head&neck                  33.0
## 10 PR42171a    51440049 SA              head&neck                  33.0
## # i 73,842 more rows
## # i 17 more variables: n_fragments_clade <dbl>, n_fragments_taxon <dbl>,
## #   n_minimisers <dbl>, n_distinct_minimisers <dbl>, rank <chr>, ncbi_id <dbl>,
## #   taxon <chr>, domain <chr>, kingdom <chr>, phylum <chr>, class <chr>,
## #   order <chr>, family <chr>, genus <chr>, species <chr>,
## #   db_n_minimisers_taxon <int>, db_n_minimisers_clade <dbl>
```

Note that the columns sample_size, db_n_minimisers_taxon, and db_n_minimisers_clade have been added.

At this stage, it will be interesting to visualise the Kraken2 results we are working on.

Read classification summary

We can start off by looking into the numbers of reads that Kraken2 was able to classify or not. The violin plot below shows, for each sample (connected dots), how many reads were classified and how many were not:

```
plotClassificationSummary_violin(
  merged_reports, return_plot = TRUE,
  outdir = "test/outputs/", prefix = "SebT"
)
```

Alternatively, if we want to look at each sample more closely, we can use a bar plot to visualise the proportions of classified/unclassified reads:

```
plotClassificationSummary_barplot(
  merged_reports, include_sample_names = FALSE, orientation = "horizontal",
  return_plot = TRUE, outdir = "test/outputs/", prefix = "SebT"
)
```

Finally, instead of looking at absolute numbers of classified/unclassified reads, we can also look at the proportion of reads classified relative to the sample sizes:

```
plotClassificationProportion(
  merged_reports, return_plot = TRUE,
  outdir = "test/outputs/", prefix = "SebT"
)
```

Distribution of classified reads

Next, for each rank, it is possible to visualise the distribution of classified reads:

```
plotDistribution_histogram(
  merged_reports,
  return_plot = TRUE,
  outdir = "test/outputs/",
  prefix = "SebT"
)
```

```
plotDistribution_violin(
  merged_reports,
  return_plot = TRUE,
```

```
  outdir = "test/outputs/",
  prefix = "SebT"
)
```

Read classification per domain

If we are interested in taxa from a particular domain (say, viruses), it can be useful to inspect the number of classified reads broken down by domain. The violin plot below shows this information to us:

```
plotDomainReads_violin(
  merged_reports, include_eukaryotes = FALSE, return_plot = TRUE,
  outdir = "test/outputs/", prefix = "SebT"
)
```

Alternatively, we can also make a bar plot to look at each sample more closely:

```
plotDomainReads_barplot(
  merged_reports, include_eukaryotes = FALSE, include_sample_names = FALSE,
  orientation = "horizontal", return_plot = TRUE,
  outdir = "test/outputs/", prefix = "SebT"
)
```

Note that in the plots above no eukaryotes were displayed - this happened because we set include_eukaryotes = FALSE. We can recreate the same plots now including taxa from the Eukaryota domain; however, you will see that the inclusion of eukaryotes will overwhelm the plots and the other domains will get harder to visualise.

```
plotDomainReads_violin(
  merged_reports, include_eukaryotes = TRUE, return_plot = TRUE,
  outdir = "test/outputs/", prefix = "SebT"
)
```

Alternatively, we can also make a bar plot to look at each sample more closely:

```
plotDomainReads_barplot(
  merged_reports, include_eukaryotes = TRUE, include_sample_names = FALSE,
  orientation = "horizontal", return_plot = TRUE,
  outdir = "test/outputs/", prefix = "SebT"
)
```

Statistical analysis

```
merged_reports <- subsetReports(merged_reports, species_to_remove = "Homo sapiens", verbose = TRUE)
merged_reports <- assessMinimiserRatio(merged_reports)
merged_reports <- assessStatistics(merged_reports, ref_db, verbose = TRUE)
```

```
## Calculating p-values...
## Adjusting p-values...
## Successfully completed.
```

```
plotSignificanceSummary(
  merged_reports,
  return_plot = TRUE,
  outdir = "test/outputs/",
  prefix = "SebT"
)

plotMinimisers_dotplot(
  merged_reports,
  domain = "Viruses",
  return_plot = TRUE,
  fig_width = 25,
  fig_height = 15,
  outdir = "test/outputs/",
  prefix = "SebT"
)
```