Basic SPARKI usage!

Jacqueline M. Boccacino

2025-06-24

```
#library(SPARKI)
devtools::load_all()
logger::log_threshold(logger::OFF)
```

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("tests/testthat/testdata/mpa_reports/valid_reports/")
std_reports <- load_STDreports("tests/testthat/testdata/std_reports/valid_reports/")
```

We can now inspect the dataframes that were created:

```
mpa_reports
```

```
## # A tibble: 1,198 x 12
##    sample  taxon_leaf  rank  n_fragments_clade domain kingdom phylum class order
##    <chr>   <chr>       <chr>             <int> <chr>  <chr>   <chr>  <chr> <chr>
##  1 sample1 Eukaryota   D              19638197 Eukar~ <NA>    <NA>   <NA>  <NA>
##  2 sample1 Metazoa     K              19274919 Eukar~ Metazoa <NA>   <NA>  <NA>
##  3 sample1 Chordata    P              19274919 Eukar~ Metazoa Chord~ <NA>  <NA>
##  4 sample1 Mammalia    C              19274919 Eukar~ Metazoa Chord~ Mamm~ <NA>
##  5 sample1 Primates    O              19274919 Eukar~ Metazoa Chord~ Mamm~ Prim~
##  6 sample1 Hominidae   F              19274919 Eukar~ Metazoa Chord~ Mamm~ Prim~
##  7 sample1 Homo        G              19274919 Eukar~ Metazoa Chord~ Mamm~ Prim~
##  8 sample1 Homo sapie~ S              19274919 Eukar~ Metazoa Chord~ Mamm~ Prim~
##  9 sample1 Fungi       K                    47 Eukar~ Fungi   <NA>   <NA>  <NA>
## 10 sample1 Basidiomyc~ P                    40 Eukar~ Fungi   Basid~ <NA>  <NA>
## # i 1,188 more rows
## # i 3 more variables: family <chr>, genus <chr>, species <chr>
```

```
std_reports
```

```
## # A tibble: 832 x 9
##    sample  pct_fragments_clade n_fragments_clade n_fragments_taxon n_minimisers
##    <chr>                 <dbl>             <int>             <int>        <int>
```

```
##  1 sample1                64.6          37743453          37743453           0
##  2 sample1                35.4          20655056             30751    227608519
##  3 sample1                32.7          19095410            347078    214835364
##  4 sample1                32.1          18740115                 0    211074103
##  5 sample1                32.1          18740115                 0    211074103
##  6 sample1                32.1          18740115                 0    211074103
##  7 sample1                32.1          18740115                 0    211074103
##  8 sample1                32.1          18740115                 0    211074103
##  9 sample1                32.1          18740115                 0    211074103
## 10 sample1                32.1          18740115          18740115    211074103
## # i 822 more rows
## # i 4 more variables: n_distinct_minimisers <int>, rank <chr>, ncbi_id <chr>,
## #   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: 832 x 17
##    sample  pct_fragments_clade n_fragments_clade n_fragments_taxon n_minimisers
##    <chr>                 <dbl>             <int>             <int>        <int>
##  1 sample1                64.6          37743453          37743453           0
##  2 sample1                35.4          20655056             30751    227608519
##  3 sample1                32.7          19095410            347078    214835364
##  4 sample1                32.1          18740115                 0    211074103
##  5 sample1                32.1          18740115                 0    211074103
##  6 sample1                32.1          18740115                 0    211074103
##  7 sample1                32.1          18740115                 0    211074103
##  8 sample1                32.1          18740115                 0    211074103
##  9 sample1                32.1          18740115                 0    211074103
## 10 sample1                32.1          18740115          18740115    211074103
## # i 822 more rows
## # i 12 more variables: n_distinct_minimisers <int>, rank <chr>, ncbi_id <chr>,
## #   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("tests/testthat/testdata/metadata.txt")
```

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 <- "sample"
mdata_columns_to_add <- c("type", "date", "status")

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: 832 x 20
##    sample  type   date       status pct_fragments_clade n_fragments_clade
##    <chr>   <chr>  <date>     <chr>                <dbl>             <int>
##  1 sample1 tumour 2024-11-15 ok                    64.6          37743453
##  2 sample1 tumour 2024-11-15 ok                    35.4          20655056
##  3 sample1 tumour 2024-11-15 ok                    32.7          19095410
##  4 sample1 tumour 2024-11-15 ok                    32.1          18740115
##  5 sample1 tumour 2024-11-15 ok                    32.1          18740115
##  6 sample1 tumour 2024-11-15 ok                    32.1          18740115
##  7 sample1 tumour 2024-11-15 ok                    32.1          18740115
##  8 sample1 tumour 2024-11-15 ok                    32.1          18740115
##  9 sample1 tumour 2024-11-15 ok                    32.1          18740115
## 10 sample1 tumour 2024-11-15 ok                    32.1          18740115
## # i 822 more rows
## # i 14 more variables: n_fragments_taxon <int>, n_minimisers <int>,
## #   n_distinct_minimisers <int>, rank <chr>, ncbi_id <chr>, 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("tests/testthat/testdata/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)
merged_reports <- add_nTaxaInRank(merged_reports)
```

Let's inspect the updated dataframe:

```
merged_reports
```

```
## # A tibble: 832 x 24
##    sample  sample_size type   date       status pct_fragments_clade
##    <chr>         <dbl> <chr>  <date>     <chr>                <dbl>
##  1 sample1    58398509 tumour 2024-11-15 ok                    64.6
##  2 sample1    58398509 tumour 2024-11-15 ok                    35.4
##  3 sample1    58398509 tumour 2024-11-15 ok                    32.7
##  4 sample1    58398509 tumour 2024-11-15 ok                    32.1
##  5 sample1    58398509 tumour 2024-11-15 ok                    32.1
##  6 sample1    58398509 tumour 2024-11-15 ok                    32.1
##  7 sample1    58398509 tumour 2024-11-15 ok                    32.1
##  8 sample1    58398509 tumour 2024-11-15 ok                    32.1
##  9 sample1    58398509 tumour 2024-11-15 ok                    32.1
## 10 sample1    58398509 tumour 2024-11-15 ok                    32.1
## # i 822 more rows
## # i 18 more variables: n_fragments_clade <int>, n_fragments_taxon <int>,
## #   n_minimisers <int>, n_distinct_minimisers <int>, rank <chr>,
## #   n_taxa_in_rank <dbl>, ncbi_id <chr>, 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>
```

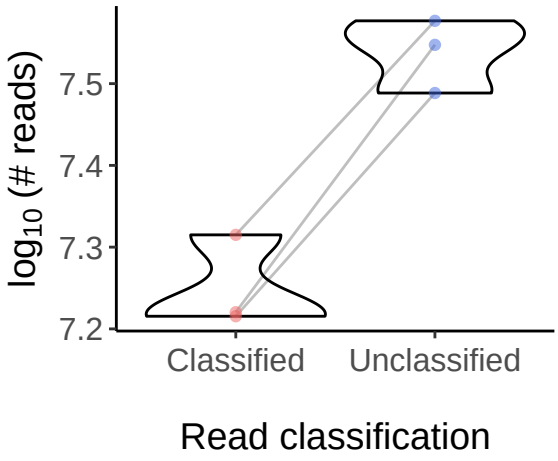
Note that the columns `sample_size`, `db_n_minimisers_taxon`, and `db_n_minimisers_clade` have been added.

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

Read classification summary

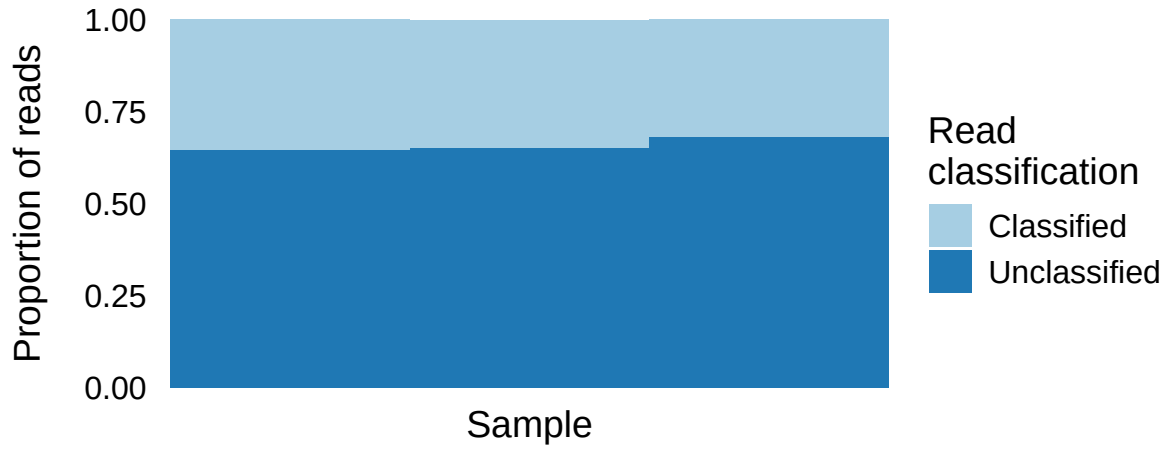
We can start off by looking into the numbers of reads that Kraken2 was able to classify or not. The violin plot below shows, for each sample (connected dots), how many reads were classified and how many were not:

```
plotClassificationSummary_violin(merged_reports)
```



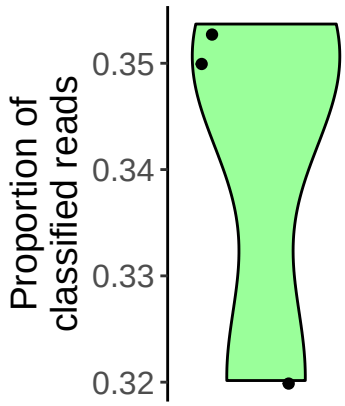
Alternatively, if we want to look at each sample more closely, we can use a bar plot to visualise the proportions of classified/unclassified reads:

```
plotClassificationSummary_barplot(
  merged_reports,
  include_sample_names = FALSE,
  orientation = "horizontal"
)
```



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:

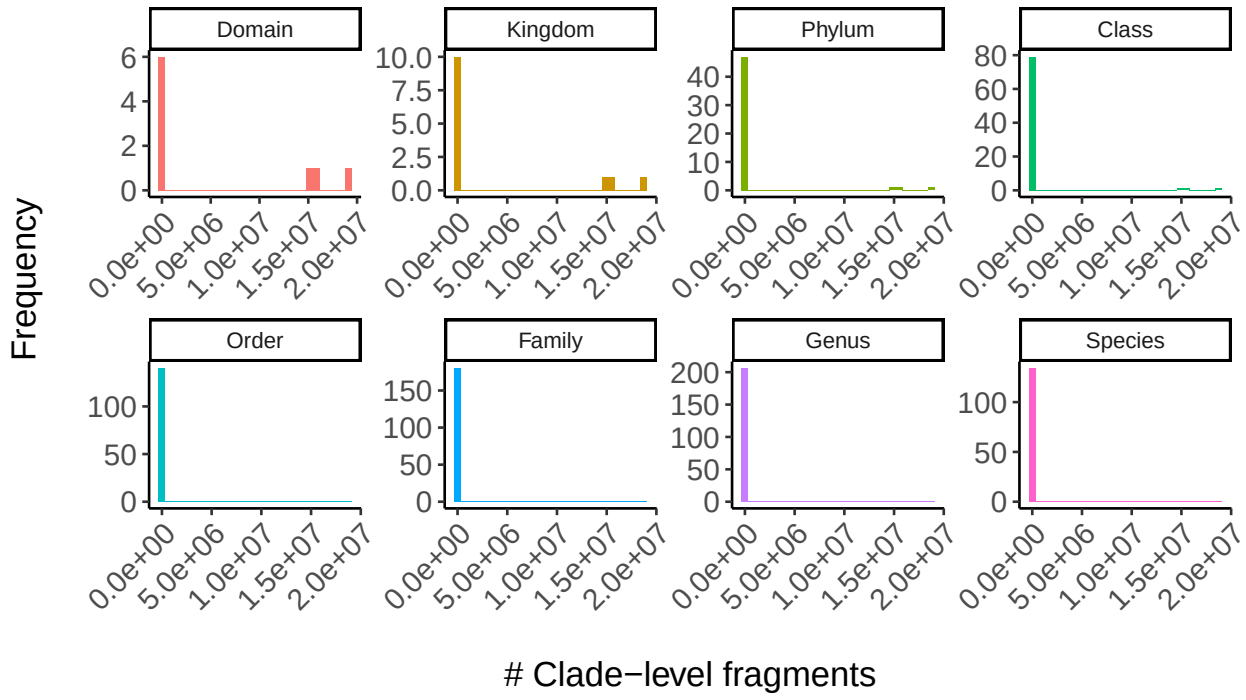
```
plotClassificationProportion(merged_reports)
```



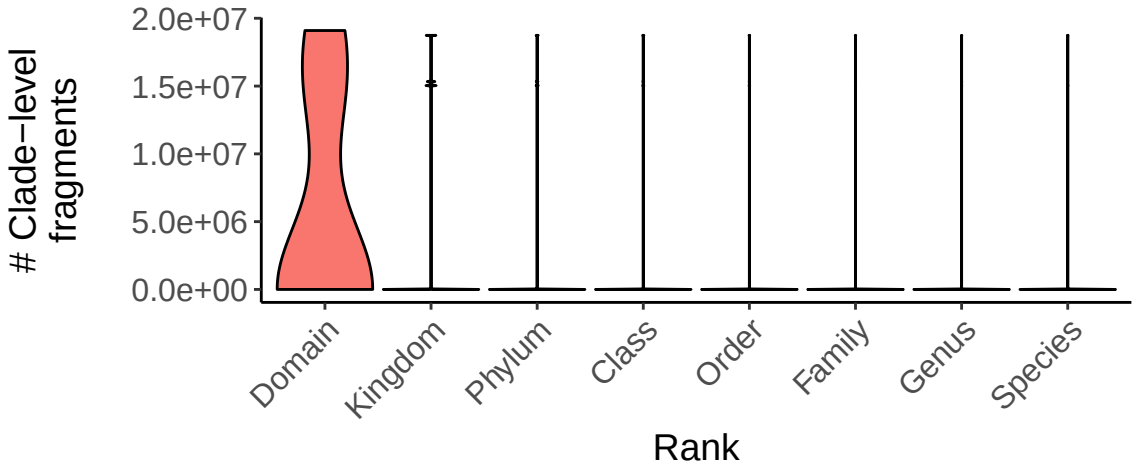
Distribution of classified reads

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

```
plotDistribution_histogram(merged_reports)
```



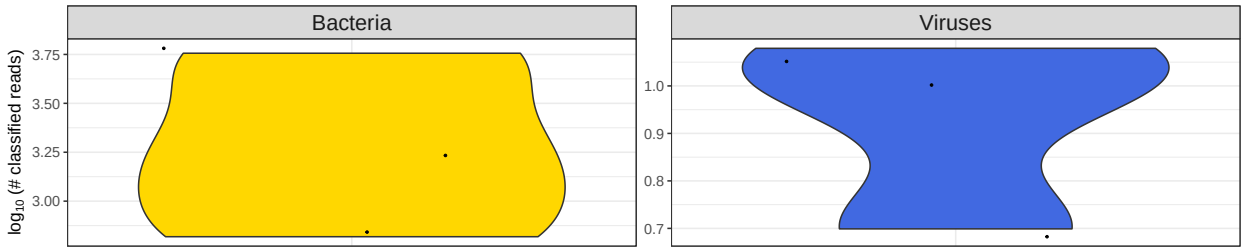

```
plotDistribution_violin(merged_reports)
```



Read classification per domain

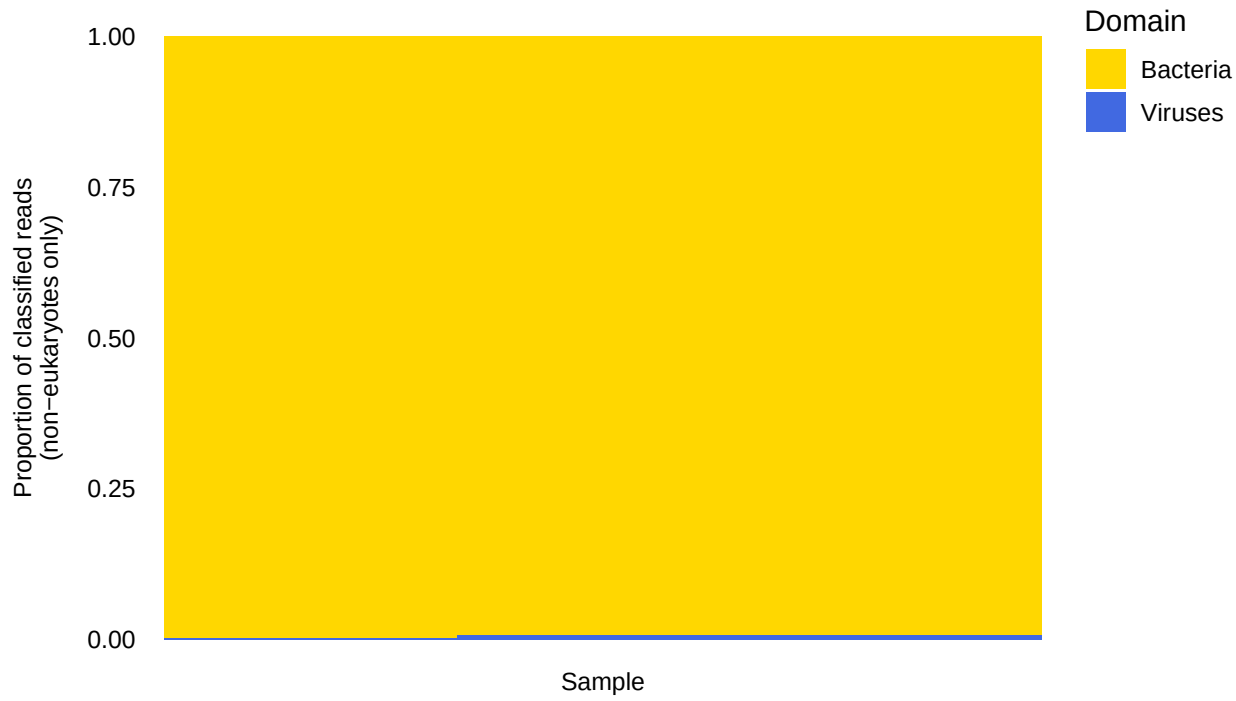
If we are interested in taxa from a particular domain (say, viruses), it can be useful to inspect the number of classified reads broken down by domain. The violin plot below shows this information to us:

```
plotDomainReads_violin(merged_reports, include_eukaryotes = FALSE)
```



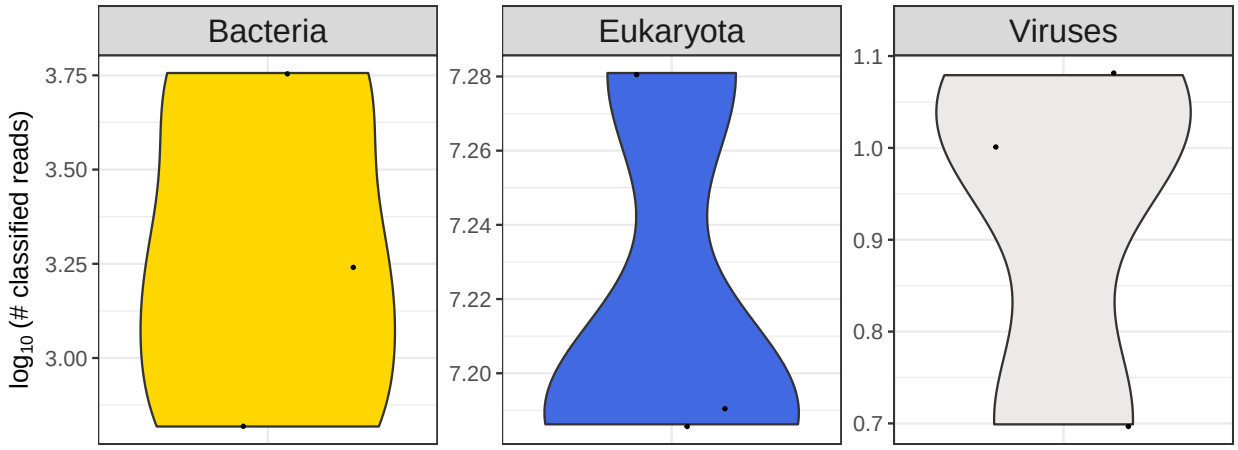
Alternatively, we can also make a bar plot to look at each sample more closely:

```
plotDomainReads_barplot(
  merged_reports,
  include_eukaryotes = FALSE,
  include_sample_names = FALSE,
  orientation = "horizontal"
)
```



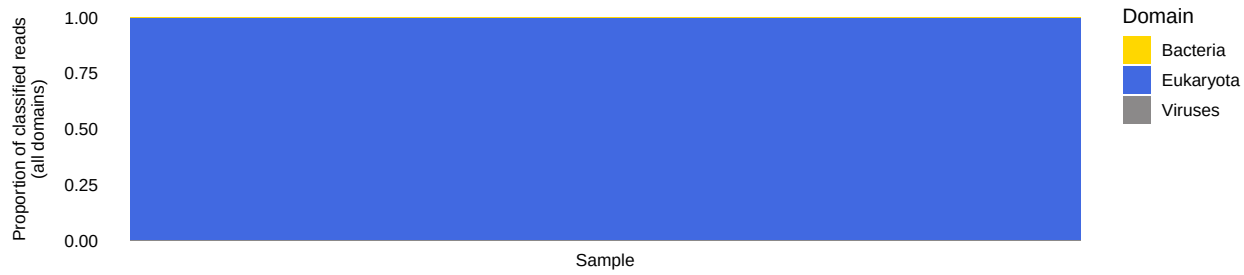
Note that in the plots above no eukaryotes were displayed - this happened because we set `include_eukaryotes = FALSE`. We can recreate the same plots now including taxa from the Eukaryota domain; however, you will see that the inclusion of eukaryotes will overwhelm the plots and the other domains will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

```
plotDomainReads_violin(merged_reports, include_eukaryotes = TRUE)
```



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

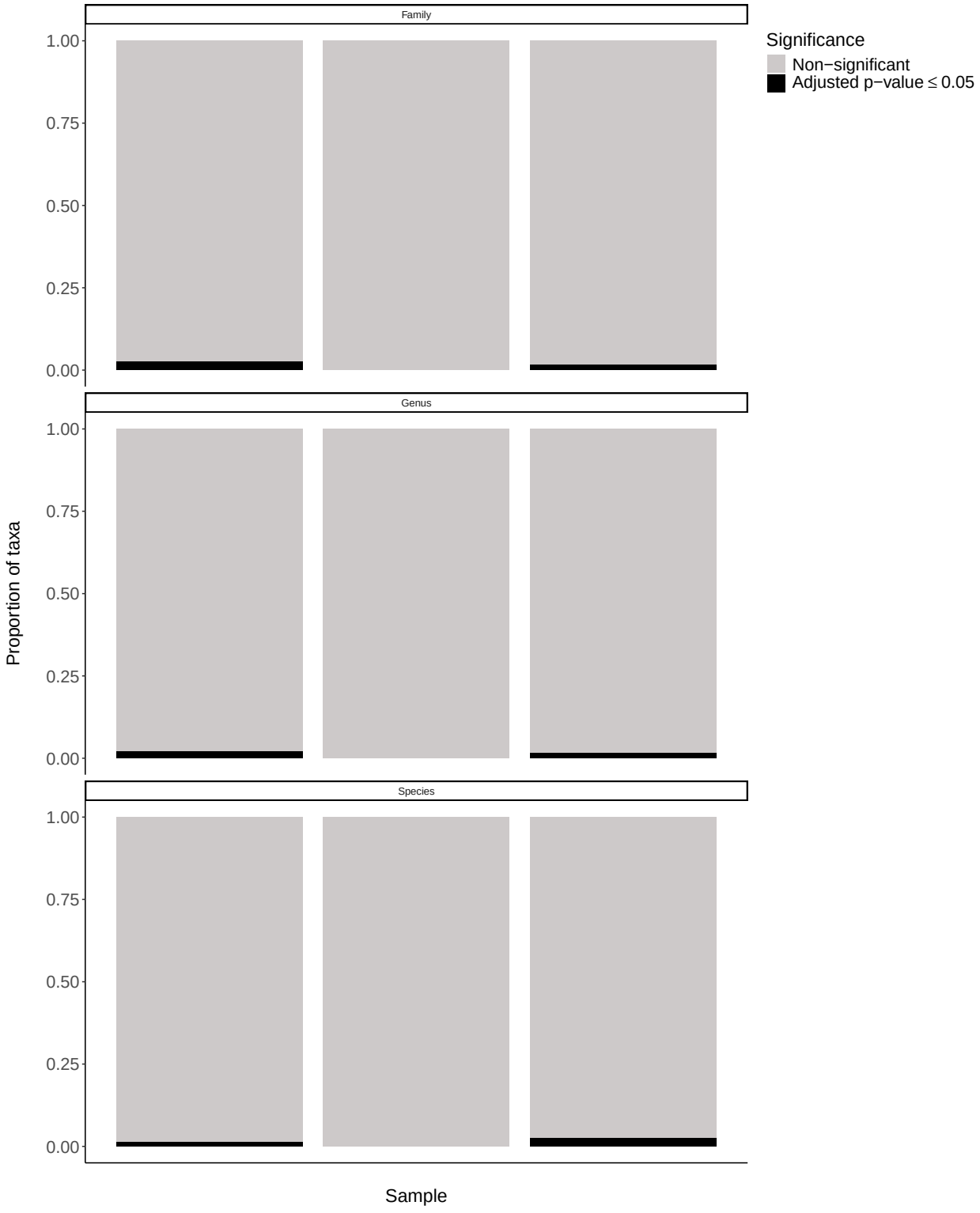
```
plotDomainReads_barplot(
  merged_reports,
  include_eukaryotes = TRUE,
  include_sample_names = FALSE,
  orientation = "horizontal"
)
```



Statistical analysis

```
merged_reports <- subsetReports(merged_reports, species = "Homo sapiens")
merged_reports <- assessMinimiserRatio(merged_reports)
merged_reports <- assessStatistics(merged_reports, ref_db)
```

```
plotSignificanceSummary(merged_reports)
```




```
plotMinimisers_dotplot(
  merged_reports,
  domain = "Viruses",
  fig_width = 12,
```

```
  fig_height = 15
)
```