Example analyzes

Contents

Installation

Krona can be installed according to these instructions. Alternatively, the KronaTools directory can be placed anywhere and its path specified manually (see below). The script *embed_krona.R* has to be downloaded and loaded with `source`.

Dependencies: Embedding interactive charts requires the `htmltools` package (should be present if rmarkdown is installed) and the `js` package. Taking snapshots requires the `webshot` package (also make sure to run `webshot::install_phantomjs()` after the installation).

Simple charts from phyloseq objects

In order to demonstrate `embed_krona`, we use the GlobalPatterns dataset from the phyloseq package, even though the phyloseq format is not the only way to provide data (see below).

```
library(phyloseq)
library(tibble)
# load the functions for plotting Krona charts
source('embed_krona.R')
# or with internet connection:
# source('https://raw.githubusercontent.com/markschl/embed_krona/master/embed_krona.R')
# load the phyloseq object
data(GlobalPatterns)
```

A krona chart can be directly produced from the phyloseq object with the following command (Figure 1):

```
plot_krona(GlobalPatterns)
```

Instead of installing Krona system-wide, the *KronaTools* directory can be manually extracted somewhere, and the location specified directly:

```
plot_krona(GlobalPatterns, kronatools_dir='KronaTools')
```

If the HTML Krona chart should be kept, the output path can be specified:

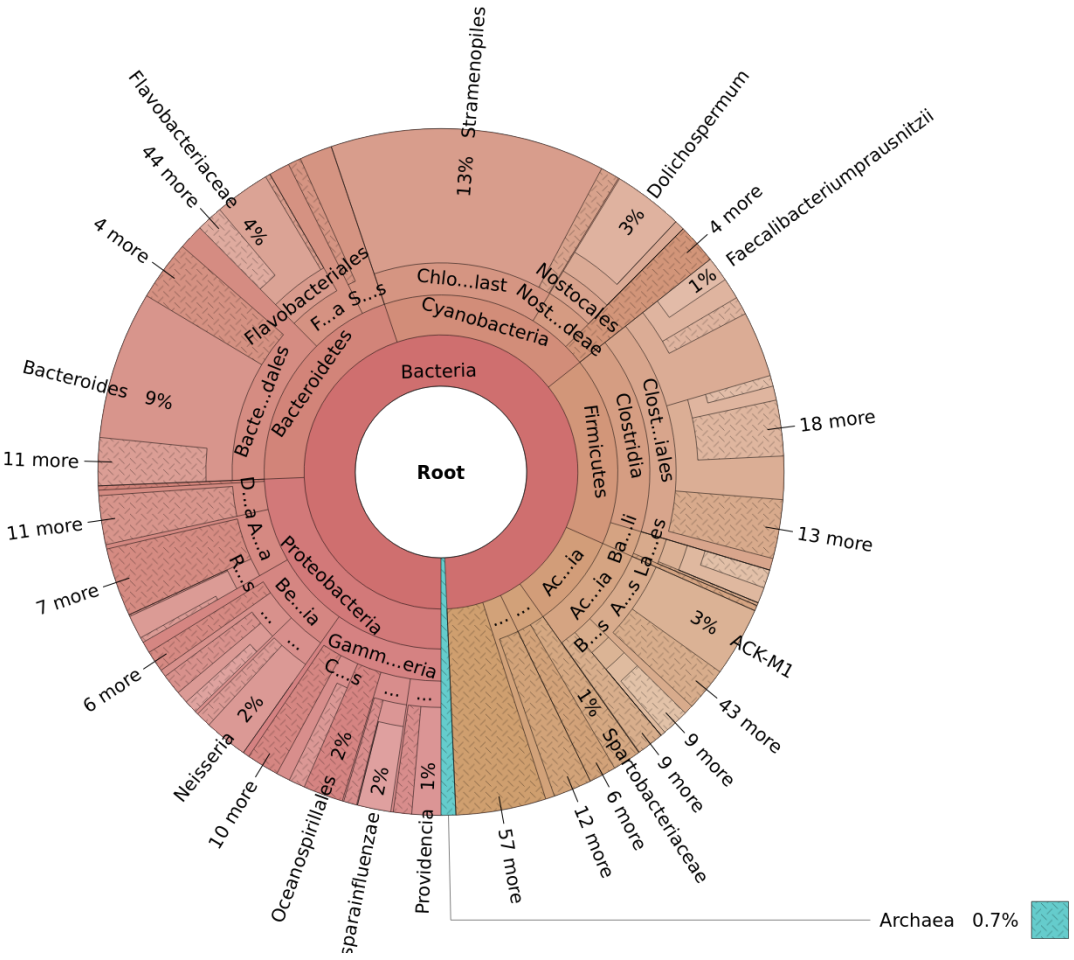

Figure 1: Taxonomic overview of the GlobalPatterns dataset

```
plot_krona(GlobalPatterns, output='krona/global_patterns.html')
```

It is also possible to produce a krona chart HTML file without actually displaying it, in the document:

```
plot_krona(GlobalPatterns, output='krona/global_patterns.html', display=F)
```

Formatting options

In R-Markdown documents displayed in Rstudio or in HTML output, an interactive chart is embedded. If `plot_krona` is issued in the R console, a browser window with an interactive chart opens. In static documents, a PNG snapshot is generated. The dimensions are controlled with *snapshot_dim* (dimensions in inches). Otherwise important settings are *snapshot_format* ('png' or 'pdf' for vector graphics), *snapshot_fontsize* (7 by default) or *snapshot_chart_size* controlling the size of the margins (default 0.7).

The following command embeds a PDF chart (Figure 2), which means that the chart is included as vector graphics in PDF documents. This looks sharper, usually produces smaller files, but on the downside makes lines look thicker and complex charts may slow down your PDF reader. PDFs may be displayed in HTML output depending on the browser (but require adjustments of `out.width` and `out.height` chunk options).

```
plot_krona(GlobalPatterns, snapshot_format='pdf', snapshot = T)
```

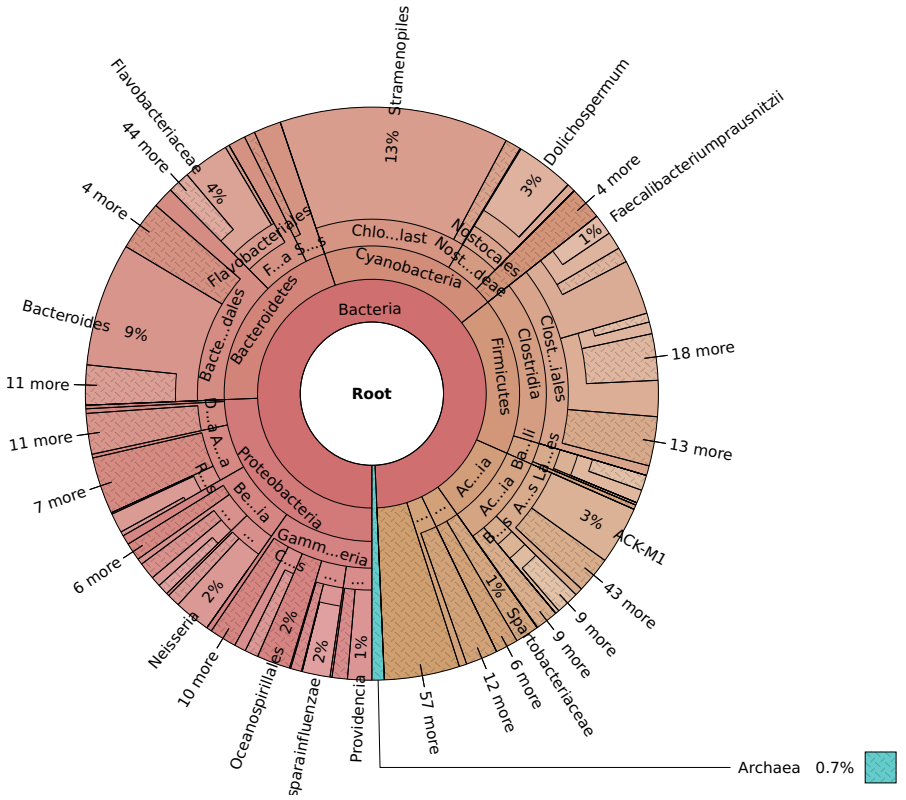

Figure 2: Static chart of the GlobalPatterns dataset embedded as vector graphics in PDF format (or just white area in HTML documents).

If the PDF chart should not be deleted, the output path can be specified (the format is recognized from the extension, and `snapshot = T` is automatically assumed):

```
plot_krona(GlobalPatterns, output='krona/global_patterns.pdf')
```

Multiple datasets

With phyloseq objects, it is also very simple to have different groups in the chart (Figure 3). This is of course only useful with interactive charts where differences between groups can be explored by manually selecting them. Static snapshots e.g. embedded in a PDF document shows only the first dataset.

```
plot_krona(GlobalPatterns, group_vars='SampleType',
           snapshot_dim=c(5, 4))
```

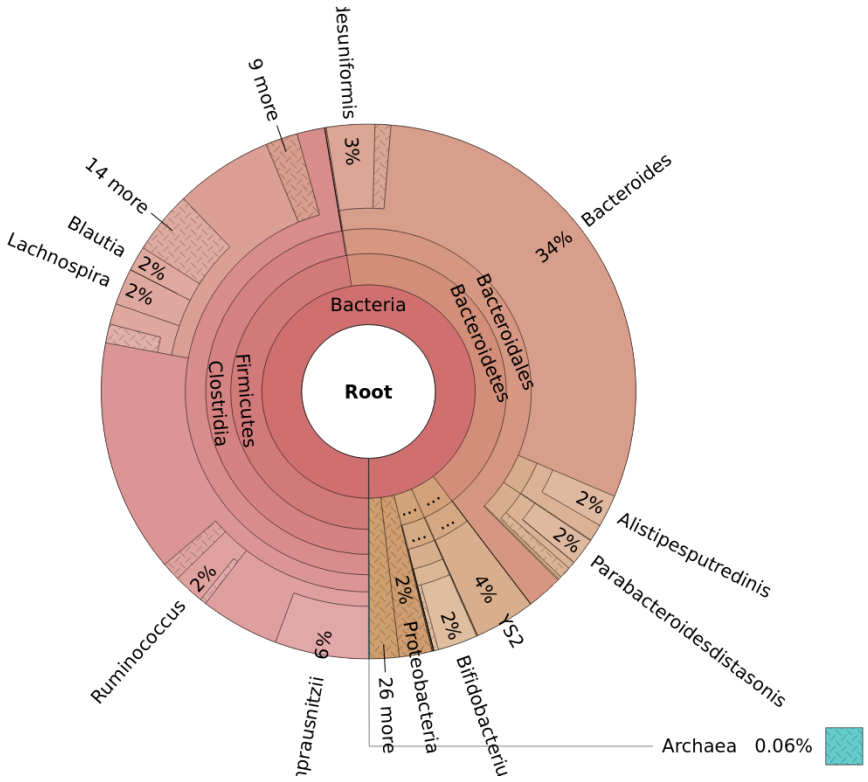

Figure 3: GlobalPatterns taxonomy, grouped by sample type. In static documents, only the first group (Feces) is displayed.

Other input data (without phyloseq)

As mentioned, phyloseq is not a requirement and charts can also be generated from any table with columns representing some hierarchy, along with a vector or data frame/matrix containing the dataset counts. Here we reproduce the Krona example (same values as in TSV or XML-formatted input files):

```
granola = tribble(
  ~cat1,           ~cat2,          ~cat3,                    ~`Brand X`, ~`Brand Y`,
  NA,              NA,              NA,                      3,          1,
  'Protein',       NA,              NA,                      5,          6,
  'Carbohydrates', NA,              NA,                      32,         28,
  'Carbohydrates', 'Sugars',        NA,                      3,          5,
  'Carbohydrates', 'Dietary fiber', NA,                      4,          8,
  'Fats',          'Saturated fat', NA,                      2,          1,
  'Fats',          'Unsaturated fat','Polyunsaturated fat',  3,          4,
```

```
  'Fats',          'Unsaturated fat','Monounsaturated fat',  3,          2
)

# hierarchical names component
hierarchy = granola[1:3]
# dataset abundances (grams)
abund = granola[4:5]
```

The result (Figure 4) is equivalent to the original chart. Note that the dataset abundances have to be supplied to the `dataset_abund` argument.

```
plot_krona(hierarchy, dataset_abund = abund,
           root_label = 'Granola serving', total_label = 'Grams',
           snapshot_dim = c(5, 4))
```

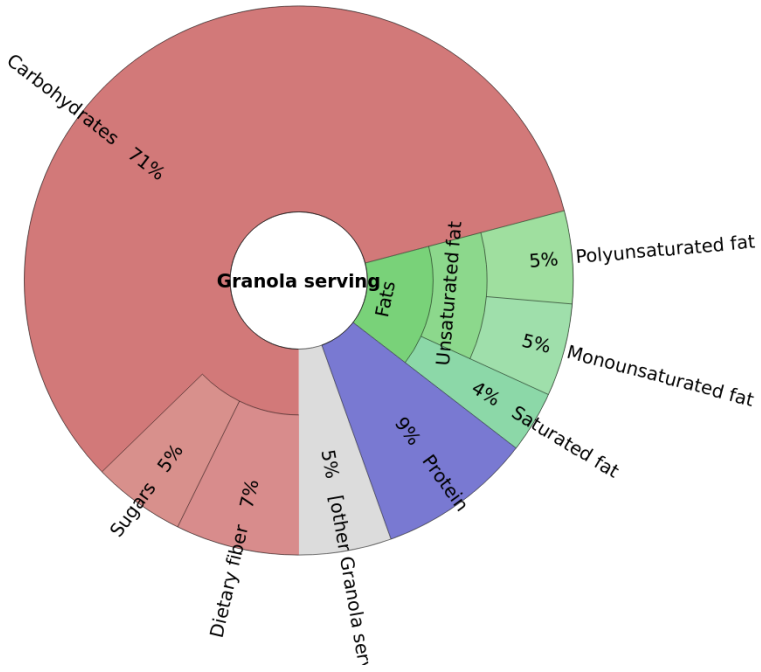

Figure 4: The Granola example reproduced

Alternatively, an abundance (OTU) table can be defined (`abund_tab` argument), which is then summarized using a grouping vector of length `ncol(abund_tab)` (`group` argument) to obtain the dataset abundances. For this example, the data is extracted from the phyloseq object before plotting. The result is equivalent to Figure 3.

```
tax = as(tax_table(GlobalPatterns), 'matrix')
otus = as(otu_table(GlobalPatterns), 'matrix')
sample_type = get_variable(GlobalPatterns, 'SampleType')

plot_krona(tax, abund_tab = otus, group = sample_type)
```

Coloring options

Another nice feature is the possibility to set the fill color depending on some property of the entries. We use data from the study by Kostic et al. (2012), which is shipped with the phyloseq package and also

used in a differential abundance analysis example. We color the taxa according to their association with colorectal carcinoma.

First, do the differential abundance testing using the *DESeq2* package:

```
library(DESeq2)

filepath = system.file('extdata', 'study_1457_split_library_seqs_and_mapping.zip',
                       package='phyloseq')
kostic = microbio_me_qiime(filepath)
kostic = subset_samples(kostic, DIAGNOSIS != 'None')

diagdds = phyloseq_to_deseq2(kostic, ~ DIAGNOSIS)
diagdds = DESeq(diagdds, sfType = 'poscounts', test = 'Wald', fitType = 'parametric')
res = results(diagdds, cooksCutoff = F)
```

After making sure that the taxa are still in the same order in the results data frame, we plot the phyloseq object and supply the log2 fold change as to the `color_values` argument (Figure 5).

```
stopifnot(taxa_names(kostic) == rownames(res))

plot_krona(kostic,
           color_values = res$log2FoldChange,
           color_label = 'log2 fold change',
           color_value_range = c(-1.8, 1.8)) # optional, to adjust the color scale
```

The interactive charts now have a checkbox on the left ("Color by log2 fold change"), which allows activating this color scheme. In the static snapshot, the color scheme is automatically activated (using `krona_opts=c(color='true')`). The strong association of *Fusobacterium* with the cancer can be clearly seen in this krona chart, although there actually isn't any indication of which associations were statistically significant. For example for species or "Unknown" names with multiple OTUs, the log fold change is calculated as the average of the log fold changes from each OTU, weighted by the OTU abundances. The same is done at higher taxonomic levels.

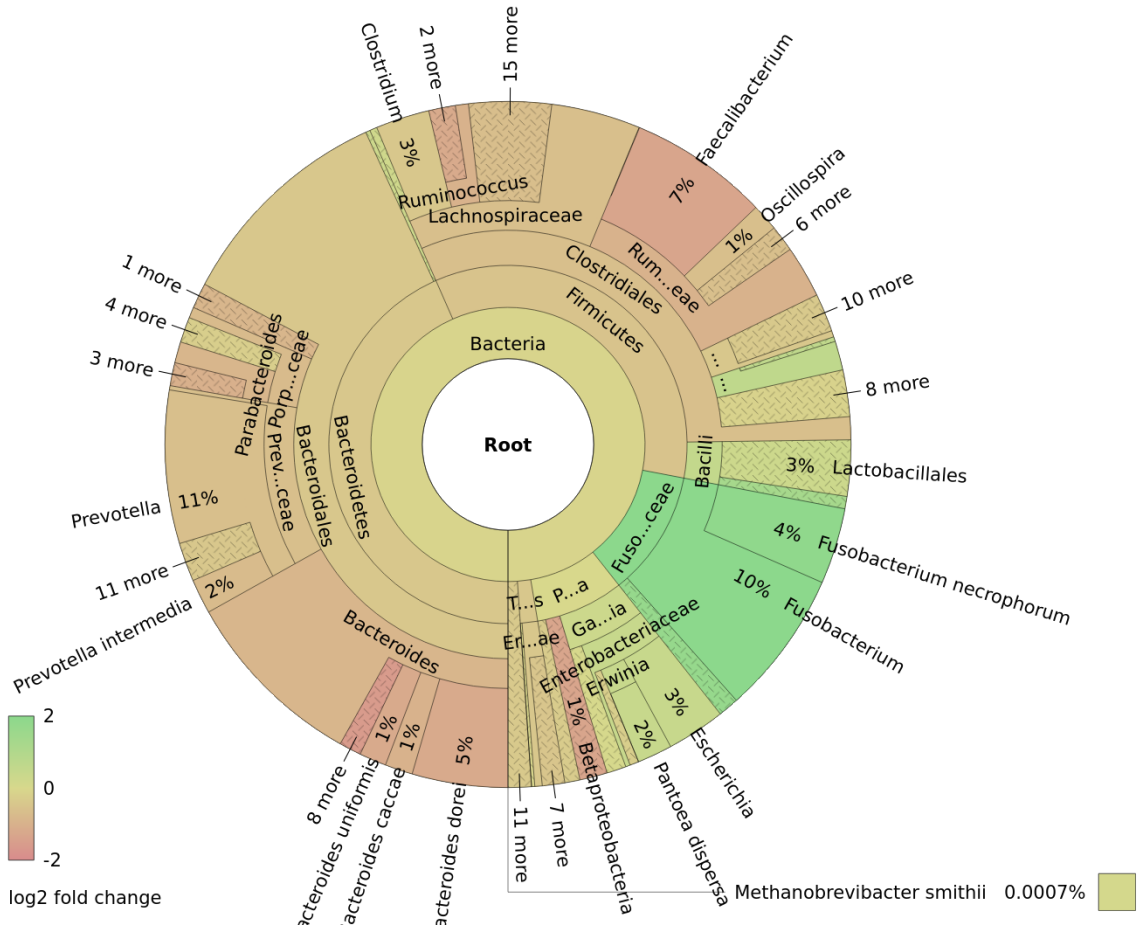

Figure 5: Taxonomic overview of the samples from the study by Kostic et al. (2012), colored by the strength of the association (positive or negative) with colorectal carcinoma. Information about the statistical significance of the association cannot be derived from this chart.