Package: SRS (via r-universe)

September 3, 2024

Type Package

Title Scaling with Ranked Subsampling

Version 0.2.3

Description Analysis of species count data in ecology often requires normalization to an identical sample size. Rarefying (random subsampling without replacement), which is a popular method for normalization, has been widely criticized for its poor reproducibility and potential distortion of the community structure. In the context of microbiome count data, researchers explicitly advised against the use of rarefying. An alternative to rarefying is scaling with ranked subsampling (SRS). SRS consists of two steps. In the first step, the total counts for all OTUs (operational taxonomic units) or species in each sample are divided by a scaling factor chosen in such a way that the sum of the scaled counts Cscaled equals Cmin. In the second step, the non-integer Cscaled values are converted into integers by an algorithm that we dub ranked subsampling. The Cscaled value for each OTU or species is split into the integer part Cint (Cint = floor(Cscaled)) and the fractional part Cfrac (Cfrac = Cscaled - Cints). Since the sum of Cint is smaller or equal to Cmin, additional delta C = Cmin - the sum of Cint counts have to be added to the library to reach the total count of Cmin. This is achieved as follows. OTUs are ranked in the descending order of their Cfrac values. Beginning with the OTU of the highest rank, single count per OTU is added to the normalized library until the total number of added counts reaches delta C and the sum of all counts in the normalized library equals Cmin. When the lowest Cfrag involved in picking delta C counts is shared by several OTUs, the OTUs used for adding a single count to the library are selected in the order of their Cint values. This selection minimizes the effect of normalization on the relative frequencies of OTUs. OTUs with identical Cfrag as well as Cint are sampled randomly without replacement. See Beule & Karlovsky (2020) <doi:10.7717/peerj.9593> for details.

Depends R (>= 3.4.0), vegan (>= 2.5-6), shiny (>= 1.5.0), DT (>= 0.16), shinycssloaders (>= 1.0.0), shinybusy (>= 0.2.2)

License CC BY-SA 4.0

Encoding UTF-8

Author Lukas Beule [aut, cre], Vitor Heidrich [aut], Petr Karlovsky [aut]

Maintainer Lukas Beule <lukas.beule@julius-kuehn.de>

NeedsCompilation no

Date/Publication 2022-03-27 14:30:09 UTC

Repository https://lukasbeule.r-universe.dev

RemoteUrl https://github.com/cran/SRS

RemoteRef HEAD

RemoteSha 5020814288a7404b31ea8668fb2d23f0c0dada3d

Contents

`Scaling with ranked subsampling (SRS)`

Scaling with ranked subsampling (SRS)

Description

Scaling with ranked subsampling (SRS) for the normalization of ecological count data. It is recommended to use SRS.shiny.app for the determination of Cmin.

Usage

```
SRS(data, Cmin, set_seed = TRUE, seed = 1)
```

Arguments

`data`	Data frame (species count or OTU table) in which columns are samples and rows are the counts of species or OTUs. Only integers are accepted as data.
`Cmin`	The number of counts to which all samples will be normalized. Typically, the total OTU count of the sample with the lowest sequencing depth is chosen as Cmin. Samples with sequencing depth lower than the chosen Cmin will be discarded.

`set_seed`	Logical, if TRUE, a seed is set to enable reproducibility of SRS if OTUs with identical Cfrag as well as Cint are sampled randomly without replacement. See set.seed for details. Default is TRUE.
`seed`	Integer, specifying the seed. See set.seed for details. Default is 1.

Details

It is recommended to use SRS.shiny.app for the determination of Cmin. SRS consists of two steps. In the first step, the total counts for all OTUs (operational taxonomic units) or species in each sample are divided by a scaling factor chosen in such a way that the sum of the scaled counts Cscaled equals Cmin. In the second step, the non-integer Cscaled values are converted into integers by an algorithm that we dub ranked subsampling. The Cscaled value for each OTU or species is split into the integer part Cint ($Cint = floor(Cscaled)$) and the fractional part Cfrac ($Cfrac = Cscaled - Cint$). Since $\Sigma Cint \leq Cmin$, additional $\Delta C = Cmin - \Sigma Cint$ counts have to be added to the library to reach the total count of Cmin. This is achieved as follows. OTUs are ranked in the descending order of their Cfrac values. Beginning with the OTU of the highest rank, single count per OTU is added to the normalized library until the total number of added counts reaches ΔC and the sum of all counts in the normalized library equals Cmin. When the lowest Cfrag involved in picking ΔC counts is shared by several OTUs, the OTUs used for adding a single count to the library are selected in the order of their Cint values. This selection minimizes the effect of normalization on the relative frequencies of OTUs. OTUs with identical Cfrag as well as Cint are sampled randomly without replacement.

Value

Data frame normalized to Cmin.

Author(s)

Lukas Beule, Vitor Heidrich, Devon O'rourke, Petr Karlovsky

References

Beule L, Karlovsky P. 2020. Improved normalization of species count data in ecology by scaling with ranked subsampling (SRS): application to microbial communities. PeerJ 8:e9593

<https://doi.org/10.7717/peerj.9593>

Examples

```
##Samples should be arranged columnwise.
##Input data should not contain any categorial
##data such as taxonomic assignment or barcode sequences.
##An example of the input data can be found below:

example_input_data <- matrix(c(sample(1:20, 100, replace = TRUE),
sample(1:30, 100, replace = TRUE),sample(1:40, 100, replace = TRUE)), nrow = 100)
colnames(example_input_data) <- c("sample_1","sample_2","sample_3")
example_input_data <- as.data.frame(example_input_data)
example_input_data
```

```
##Selection of the desired number of counts
##(e.g., total OTU counts of the sample with the lowest sequencing depth):

Cmin <- min(colSums(example_input_data))
Cmin

##Running the SRS function
SRS_output <- SRS(example_input_data, Cmin)
SRS_output

##Samples that have a total number of counts < Cmin will be discarded:
SRS_output <- SRS(example_input_data, Cmin+1)
SRS_output
```

`Scaling with ranked subsampling (SRS) Shiny app`
Shiny app for scaling with ranked subsampling (SRS)

Description

Shiny app for the determination of Cmin for scaling with ranked subsampling (SRS).

Usage

```
SRS.shiny.app(data)
```

Arguments

`data`	Data frame (species count or OTU table) in which columns are samples and rows are the counts of species or OTUs. Only integers are accepted as data.

Details

Shiny app that generates a visualization of retained samples, summary statistics, SRS curves, and an interactive table in response to varying minimum sample size (Cmin).

Value

Launches Shiny app for SRS in the default web browser.

Author(s)

Vitor Heidrich, Devon O'rourke, Petr Karlovsky, Lukas Beule

References

Beule L, Karlovsky P. 2020. Improved normalization of species count data in ecology by scaling with ranked subsampling (SRS): application to microbial communities. PeerJ 8:e9593

<https://doi.org/10.7717/peerj.9593>

Examples

```
##Samples should be arranged columnwise.
##Input data should not contain any categorial
##data such as taxonomic assignment or barcode sequences.
##An example of the input data can be found below:

example_input_data <- matrix(c(sample(1:20, 100, replace = TRUE),
sample(1:30, 100, replace = TRUE),sample(1:40, 100, replace = TRUE)), nrow = 100)
colnames(example_input_data) <- c("sample_1","sample_2","sample_3")
example_input_data <- as.data.frame(example_input_data)
example_input_data

##Launching the SRS shiny app with example_input_data as input
if (interactive()) {SRS.shiny.app(example_input_data)}
```

`Scaling with ranked subsampling curve (SRScurve)`

Scaling with ranked subsampling curve (SRScurve)

Description

For each column of the input data, draws a line plot of alpha diversity indices (see metric) at different sample sizes (specified by step) normalized by scaling with ranked subsampling (using SRS). Minimum sample size (cutoff-level) can be evaluated by specifying sample. The function further allows to visualize trade-offs between cutoff-level and alpha diversity and enables direct comparison of SRS and repeated rarefying.

See Beule & Karlovsky (2020) <doi:10.7717/peerj.9593> for details regarding SRS.

Usage

```
SRScurve(data, metric = "richness", step = 50, sample = 0, max.sample.size = 0,
        rarefy.comparison = FALSE, rarefy.repeats = 10,
        rarefy.comparison.legend = FALSE, xlab = "sample size",
        ylab = "richness", label = FALSE, col, lty, ...)
```

Arguments

`data`	Data frame (species count or OTU table) in which columns are samples and rows are the counts of species or OTUs. Only integers are accepted as data.
`metric`	Character, "richness" (using specnumber) for species richness or "shannon", "simpson" or "invsimpson" (using diversity) for common diversity indices. Default is "richness".
`step`	Numeric, specifying the step used to vary the sample size. Default is 50.
`sample`	Numeric, specifying the cutoff-level to visualize trade-offs between cutoff-level and alpha diversity.

`max.sample.size`
: Numeric, specifying the maximum sample size to which SRS curves are drawn. Default is 0 which does not limit the maximum sample size.

`rarefy.comparison`
: Logical, if TRUE, median values of rarefy with n repeats (specified by rarefy.repeats) will be drawn for comparison. Default is FALSE.

`rarefy.repeats`
: Numeric, specifying the number of repeats used to obtain median values for rarefying. Default is 10.

`rarefy.comparison.legend`
: Logical, if TRUE, a legend for the comparison between SRS and rarefy is plotted. Default is FALSE.

`xlab`, `ylab`, `label`, `col`, `lty`, `...`
: Graphical parameters.

Details

See Beule & Karlovsky (2020) <doi:10.7717/peerj.9593> for details regarding scaling with ranked subsampling.

Value

Returns a line plot visualizing the change in alpha diversity indices with changing sample size.

Author(s)

Vitor Heidrich, Petr Karlovsky, Lukas Beule

References

Beule L, Karlovsky P. 2020. Improved normalization of species count data in ecology by scaling with ranked subsampling (SRS): application to microbial communities. PeerJ 8:e9593

<https://doi.org/10.7717/peerj.9593>

Examples

```
##Samples should be arranged columnwise.
##Input data should not contain any categorial
##data such as taxonomic assignment or barcode sequences.
##An example of the input data can be found below:

example_input_data <- matrix(c(sample(1:20, 100, replace = TRUE),
sample(1:30, 100, replace = TRUE),sample(1:40, 100, replace = TRUE)), nrow = 100)
colnames(example_input_data) <- c("sample_1","sample_2","sample_3")
example_input_data <- as.data.frame(example_input_data)
example_input_data

##Default settings of SRScurve.
SRScurve(example_input_data, metric = "richness", step = 50,
         ylab = "richness",
         col = c("#000000", "#E69F00", "#56B4E9"))
```

```
##Limit the compution of SRS curves to a sample size of 200.
SRScurve(example_input_data, metric = "richness", step = 50,
         max.sample.size = 200, ylab = "richness",
         col = c("#000000", "#E69F00", "#56B4E9"))

##SRScurve with comparison of SRS (solid lines) and repeated rarefying (dashed lines).
##Different colors correspond to indiviual samples. Cuttoff-level set to 200.
SRScurve(example_input_data, metric = "richness", step = 50,
         sample = 200, max.sample.size = 200,
         rarefy.comparison = TRUE, rarefy.repeats = 10, rarefy.comparison.legend = TRUE,
         ylab = "richness",
         col = c(rep(c("#000000", "#E69F00", "#56B4E9"),2)),
         lty = c(1,2))
```

Index