

# Package ‘sgsR’

November 19, 2022

**Type** Package

**Version** 1.3.2

**License** GPL (>= 3)

**Title** Structurally Guided Sampling

**Description** Structurally guided sampling (SGS) approaches for airborne laser scanning (ALS; LiDAR). Primary functions provide means to generate data-driven stratifications & methods for allocating samples. Intermediate functions for calculating and extracting important information about input covariates and samples are also included. Processing outcomes are intended to help forest and environmental management practitioners better optimize field sample placement as well as assess and augment existing sample networks in the context of data distributions and conditions. ALS data is the primary intended use case, however any rasterized remote sensing data can be used, enabling data-driven stratifications and sampling approaches.

**URL** <https://github.com/tgoodbody/sgsR>,  
<https://tgoodbody.github.io/sgsR/>

**BugReports** <https://github.com/tgoodbody/sgsR/issues>

**Depends** R (>= 3.5.0), methods

**Imports** dplyr, ggplot2, sf, terra, tidyr, clhs, SamplingBigData,  
BalancedSampling, spatstat.geom

**Suggests** knitr, rmarkdown, Rfast, testthat (>= 3.0.0), doParallel,  
doSNOW, snow, foreach, entropy, roxygen2, covr, RANN

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**Encoding** UTF-8

**RoxygenNote** 7.2.0

**NeedsCompilation** no

**Author** Tristan RH Goodbody [aut, cre, cph]

(<<https://orcid.org/0000-0002-6894-7925>>),

Nicholas C Coops [aut] (<<https://orcid.org/0000-0002-0151-9037>>),

Martin Queinnec [aut] (<<https://orcid.org/0000-0002-2741-1032>>)

**Maintainer** Tristan RH Goodbody <goodbody.t@gmail.com>

**Repository** CRAN

**Date/Publication** 2022-11-19 12:40:02 UTC

## R topics documented:

calculate_allocation . . . . .	2
calculate_coobs . . . . .	4
calculate_distance . . . . .	6
calculate_lhsOpt . . . . .	7
calculate_pcomp . . . . .	9
calculate_pop . . . . .	10
calculate_representation . . . . .	12
calculate_sampsize . . . . .	13
extract_metrics . . . . .	15
extract_strata . . . . .	16
sample_ahels . . . . .	17
sample_balanced . . . . .	20
sample_clhs . . . . .	21
sample_existing . . . . .	23
sample_nc . . . . .	25
sample_srs . . . . .	27
sample_strat . . . . .	29
sample_systematic . . . . .	31
strat_breaks . . . . .	34
strat_kmeans . . . . .	35
strat_map . . . . .	37
strat_poly . . . . .	39
strat_quantiles . . . . .	40
<b>Index</b>	<b>43</b>

---

calculate\_allocation *Sample allocation type and count*

---

### Description

Determine how many samples to allocate within strata.

### Usage

```
calculate_allocation(
  sraster,
  nSamp,
  allocation = "prop",
  weights = NULL,
```

```

    mraster = NULL,
    existing = NULL,
    force = FALSE
  )

```

### Arguments

sraster	spatRaster. Stratification raster to be used for sampling.
nSamp	Numeric. Number of desired samples.
allocation	Character. Allocation algorithm to be used. Either prop (default) for proportional allocation, optim for optimal allocation (equal sampling cost), equal for equal number of samples (defined by nSamp) for each strata, or "manual" for user defined strata weights defined using weights.
weights	Numeric. Only applicable when allocation = "manual". Vector of weights where sum(weights) == 1. Vector length must be equal to the number of unique strata where the first numeric value corresponds to stratum 1, second stratum 2 etc.
mraster	spatRaster. ALS metric raster. Required when allocation = optim.
existing	sf 'POINT' or data.frame. Existing plot network.
force	Logical. Default = FALSE - force nSamp to be exactly the user defined value in cases where nSamp and sraster strata count are not equally divisible. Additional samples often need to be allocated or removed based on rounding differences resulting from proportional differences between nSamp and strata coverages in sraster. In these instances samples are either added to strata with the lowest number of samples or are removed from strata with the highest number of samples. Has no effect when existing is provided.

### Value

Returns a data.frame of:

- strata - Strata ID.
- total - Number of samples to be allocated. Values correspond to under representation (samples needed; positive value) or over representation (too many samples; negative value) based on the nSamp provided.
- need - Required samples per strata based on allocation method. Rounded.

### Author(s)

Tristan R.H. Goodbody

### References

Gregoire, T.G., & Valentine, H.T. (2007). Sampling Strategies for Natural Resources and the Environment (1st ed.). Chapman and Hall/CRC. <https://doi.org/10.1201/9780203498880>

**See Also**

Other calculate functions: [calculate\\_coobs\(\)](#), [calculate\\_distance\(\)](#), [calculate\\_pcomp\(\)](#), [calculate\\_pop\(\)](#), [calculate\\_representation\(\)](#), [calculate\\_sampsiz\(\)](#)

**Examples**

```
#--- Load strata raster and existing samples---#
r <- system.file("extdata", "sraster.tif", package = "sgsR")
sr <- terra::rast(r)

e <- system.file("extdata", "existing.shp", package = "sgsR")
e <- sf::st_read(e)

#--- proportional allocation ---#
calculate_allocation(
  sraster = sr,
  nSamp = 10
)

#--- equal allocation ---#
calculate_allocation(
  sraster = sr,
  allocation = "equal",
  nSamp = 10
)

#--- manual allocation ---#
#--- define user-defined weights ---#

weights <- c(0.2, 0.2, 0.5, 0.1)

calculate_allocation(
  sraster = sr,
  allocation = "manual",
  weights = weights,
  nSamp = 10
)
```

---

calculate\_coobs

*coobs algorithm sampling*

---

**Description**

Perform the COunt of OBServations (coobs) algorithm using existing site data and raster metrics. This algorithm aids the user in determining where additional samples could be located by comparing existing samples to each pixel and associated covariates. The output coobs raster could be used to constrain clhs sampling to areas that are underrepresented.

**Usage**

```
calculate_coobs(  
  mraster,  
  existing,  
  cores = 1,  
  threshold = 0.95,  
  plot = FALSE,  
  filename = NULL,  
  overwrite = FALSE  
)
```

**Arguments**

mraster	spatRaster. ALS metrics raster. Requires at least 2 layers to calculate covariance matrix.
existing	sf 'POINT'. Existing plot network.
cores	Numeric. Number of cores to use for parallel processing. default = 1.
threshold	Numeric. Proxy maximum pixel quantile to avoid outliers. default = 0.95.
plot	Logical. Plots output strata raster and visualized strata with boundary dividers.
filename	Character. Path to write stratified raster to disc.
overwrite	Logical. Specify whether filename should be overwritten on disc.

**Value**

output raster with coobs and classified coobs layers.

**Note**

Special thanks to Dr. Brendan Malone for the original implementation of this algorithm.

**Author(s)**

Tristan R.H. Goodbody

**References**

Malone BP, Minansy B, Brungard C. 2019. Some methods to improve the utility of conditioned Latin hypercube sampling. PeerJ 7:e6451 DOI 10.7717/peerj.6451

**See Also**

Other calculate functions: [calculate\\_allocation\(\)](#), [calculate\\_distance\(\)](#), [calculate\\_pcomp\(\)](#), [calculate\\_pop\(\)](#), [calculate\\_representation\(\)](#), [calculate\\_sampsize\(\)](#)

**Examples**

```
## Not run:
#--- Load raster and existing plots---#
r <- system.file("extdata", "mraster.tif", package = "sgsR")
mr <- terra::rast(r)

e <- system.file("extdata", "existing.shp", package = "sgsR")
e <- sf::st_read(e)

calculate_coobs(
  mraster = mr,
  existing = e,
  cores = 4
)

## End(Not run)
```

---

calculate\_distance      *Distance to access layer*

---

**Description**

Per pixel distance to nearest access vector. Intended to be used as a ‘cost’ constraint within the [sample\\_clhs](#) function

**Usage**

```
calculate_distance(
  raster,
  access,
  plot = FALSE,
  filename = NULL,
  overwrite = FALSE
)
```

**Arguments**

raster	spatRaster. Raster to be used to calculate pixel level distance to access layer.
access	sf 'LINESTRING' or 'MULTILINESTRING'. Access network.
plot	Logical. Plots output strata raster with samples.
filename	Character. Path to write output samples.
overwrite	Logical. Choice to overwrite existing filename if it exists.

**Value**

Input raster with dist2access layer appended.

**Author(s)**

Tristan R.H. Goodbody

**See Also**

Other calculate functions: [calculate\\_allocation\(\)](#), [calculate\\_coobs\(\)](#), [calculate\\_pcomp\(\)](#), [calculate\\_pop\(\)](#), [calculate\\_representation\(\)](#), [calculate\\_sampsize\(\)](#)

**Examples**

```
#--- Load raster and access files ---#
r <- system.file("extdata", "mraster_small.tif", package = "sgsR")
mr <- terra::rast(r)

a <- system.file("extdata", "access.shp", package = "sgsR")
ac <- sf::st_read(a)

calculate_distance(
  raster = mr,
  access = ac,
)
```

---

calculate\_lhsOpt

*Analyze optimal Latin hypercube sample number*

---

**Description**

Population level analysis of metric raster data to determine optimal Latin Hypercube sample size

**Usage**

```
calculate_lhsOpt(
  mats,
  PCA = TRUE,
  quant = TRUE,
  KLdiv = TRUE,
  minSamp = 10,
  maxSamp = 100,
  step = 10,
  rep = 10,
  iter = 10000
)
```

**Arguments**

mats	List. Output from <code>calculate_pop</code> function.
PCA	Logical. Calculates principal component loadings of the population for PCA similarity factor testing. <code>default = FALSE</code> .
quant	Logical. Perform quantile comparison testing.
KLdiv	Logical. Perform Kullback–Leibler divergence testing.
minSamp	Numeric. Minimum sample size to test. <code>default = 10</code> .
maxSamp	Numeric. Maximum sample size to test. <code>default = 100</code> .
step	Numeric. Sample step size for each iteration. <code>default = 10</code> .
rep	Numeric. Internal repetitions for each sample size. <code>default = 10</code> .
iter	Positive Numeric. The number of iterations for the Metropolis-Hastings annealing process. Defaults to 10000. Internal to <code>clhs</code> .

**Value**

data.frame with summary statistics.

**Note**

Special thanks to Dr. Brendan Malone for the original implementation of this algorithm.

**Author(s)**

Tristan R.H. Goodbody

**References**

Malone BP, Minansy B, Brungard C. 2019. Some methods to improve the utility of conditioned Latin hypercube sampling. PeerJ 7:e6451 DOI 10.7717/peerj.6451

**Examples**

```
## Not run:
#--- Load raster and access files ---#
r <- system.file("extdata", "mraster.tif", package = "sgsR")
mr <- terra::rast(r)

#--- calculate lhsPop details ---#
mats <- calculate_pop(mraster = mr)

calculate_lhsOpt(mats = mats)

calculate_lhsOpt(
  mats = mats,
  PCA = FALSE,
  iter = 200
)
```



```
## End(Not run)
```

---

calculate_pcomp	<i>Raster principal components</i>
-----------------	------------------------------------

---

### Description

Calculate and rasterize principal components from a metric raster

### Usage

```
calculate_pcomp(
  mraster,
  nComp,
  center = TRUE,
  scale = TRUE,
  plot = FALSE,
  details = FALSE,
  filename = NULL,
  overwrite = FALSE,
  ...
)
```

### Arguments

mraster	spatRaster. ALS metrics raster.
nComp	Numeric. Value indicating number of principal components to be rasterized.
center	Logical. Value indicating whether the variables should be shifted to be zero centered.
scale	Logical. Value indicating whether the variables should be scaled to have unit variance.
plot	Logical. Plots output strata raster with samples.
details	Logical. If FALSE (default) output is only stratification raster. If TRUE return a list where \$details is additional stratification information and \$raster is the output stratification spatRaster.
filename	Character. Path to write output samples.
overwrite	Logical. Choice to overwrite existing filename if it exists.
...	Additional arguments to be passed to <a href="#">prcomp</a> .

### Value

Output raster with specified number of principal components as layers.

**Author(s)**

Tristan R.H. Goodbody

**See Also**

Other calculate functions: [calculate\\_allocation\(\)](#), [calculate\\_coobs\(\)](#), [calculate\\_distance\(\)](#), [calculate\\_pop\(\)](#), [calculate\\_representation\(\)](#), [calculate\\_sampsize\(\)](#)

**Examples**

```
#--- Load raster ---#
r <- system.file("extdata", "mraster.tif", package = "sgsR")
mr <- terra::rast(r)

calculate_pcomp(
  mraster = mr,
  nComp = 2
)

pcomp <- calculate_pcomp(
  mraster = mr,
  nComp = 3,
  details = TRUE
)

#--- Display principal component details ---#
#pcomp$pca

#--- Display importance of components ---#
#summary(pcomp$pca)
```

---

calculate\_pop

*Population descriptors*

---

**Description**

Population matrices and descriptions of metric raster data

Calculates population level statistics including principal components, quantile matrix, and covariance matrix needed necessary for [calculate\\_lhsOpt](#). Outputs can also be used as an input for [sample\\_ahels](#).

**Usage**

```
calculate_pop(mraster, PCA = FALSE, matQ = TRUE, nQuant = 10, matCov = TRUE)
```

**Arguments**

mraster	spatRaster. ALS metrics raster.
PCA	Logical. Calculates principal component loadings of the population for PCA similarity factor testing. default = FALSE.
matQ	Logical. Calculates quantile matrix of the population for quantile comparison testing. default = TRUE.
nQuant	Numeric. Number of quantiles to divide the population into for matQ. default = 10.
matCov	Logical. Calculates covariate matrix of the population. Needed for Kullback–Leibler divergence testing. default = TRUE. Requires matQ = TRUE.

**Value**

List of matrices to be used as input for [calculate\\_lhsOpt](#).

**Note**

Special thanks to Dr. Brendan Malone for the original implementation of this algorithm.

**Author(s)**

Tristan R.H. Goodbody

**References**

Malone BP, Minansy B, Brungard C. 2019. Some methods to improve the utility of conditioned Latin hypercube sampling. PeerJ 7:e6451 DOI 10.7717/peerj.6451

**See Also**

Other calculate functions: [calculate\\_allocation\(\)](#), [calculate\\_coobs\(\)](#), [calculate\\_distance\(\)](#), [calculate\\_pcomp\(\)](#), [calculate\\_representation\(\)](#), [calculate\\_sampsize\(\)](#)

**Examples**

```
### Load raster and access files ###
r <- system.file("extdata", "mraster.tif", package = "sgsR")
mr <- terra::rast(r)

calculate_pop(mraster = mr)
```

---

calculate\_representation

*Compare sample representation within sraster strata*

---

### Description

Compare sample representation within sraster strata

### Usage

```
calculate_representation(sraster, existing, plot = FALSE)
```

### Arguments

sraster	spatRaster. Stratification raster.
existing	sf 'POINT'. Existing plot network.
plot	Logical. Plot frequency of strata coverage and sampling coverage for sraster and existing. Will return a list if TRUE.

### Details

Calculate how sraster strata are represented in existing samples

### Value

Returns a tibble where:

- strata - sraster strata ID.
- srasterFreq - sraster coverage frequency.
- sampleFreq - Sampling frequency within sraster strata.
- diffFreq - Difference between srasterFreq & sampleFreq. Positive values indicate over representation.
- nSamp - Number of samples within each strata in existing.
- need -  $\text{srasterFreq} * \text{sum}(\text{nSamp})$ . Total theoretical number of required samples to be representative of strata coverage. This values is rounded. It is important for the user to consider diffFreq. A small difference - e.g. 1 in sampleFreq vs. srasterFreq frequency could make the algorithm allocate or remove samples that could likely be ignored.

### Author(s)

Tristan R.H. Goodbody, Martin Queinnec

### See Also

Other calculate functions: [calculate\\_allocation\(\)](#), [calculate\\_coobs\(\)](#), [calculate\\_distance\(\)](#), [calculate\\_pcomp\(\)](#), [calculate\\_pop\(\)](#), [calculate\\_sampsize\(\)](#)

**Examples**

```
### --- generate example stratification ---###

#--- load ALS metrics from sgsR internal data ---#
r <- system.file("extdata", "mraster.tif", package = "sgsR")

#--- read ALS metrics using the terra package ---#
mraster <- terra::rast(r)

#--- perform stratification ---#
sraster <- strat_kmeans(
  mraster = mraster$zq90,
  nStrata = 6
)

### --- create existing sample network ---###

#--- simple random sampling ---#
existing <- sample_srs(
  raster = mraster$zq90,
  nSamp = 100
)

#--- calculate representation ---#

calculate_representation(
  sraster = sraster,
  existing = existing
)
```

---

calculate\_sampsize      *Sample size determination*

---

**Description**

Determine a samples size for simple random sampling using relative standard error

**Usage**

```
calculate_sampsize(
  mraster,
  rse = NULL,
  start = 0.01,
  end = 0.05,
  increment = 0.001,
  plot = FALSE
)
```

**Arguments**

mraster	spatRaster. Metrics raster. All values must be numeric.
rse	Numeric. Desired relative standard error (coefficient of variation of the mean) threshold to determine sample size.
start	Numeric. First rse value to begin rse sequence. default = 0.01.
end	Numeric. Final rse value to end rse sequence. default = 0.05.
increment	Numeric. Value to increment between start & end. default = 0.001.
plot	Logical. if TRUE output graphical representation of estimated sample size vs. rse.

**Value**

A data.frame of sample size and rse by raster variable.

**Note**

$$rse = (100 * SE) / mean$$

Where:

- SE - Standard error of the mean
- s - Standard deviation of the observations
- n - Number of observations

**Author(s)**

Tristan R.H. Goodbody

**References**

Benedetti, R., Piersimoni, F., & Postiglione, P. (2015). Sampling spatial units for agricultural surveys. pp 202-203. Berlin: Springer.

**See Also**

Other calculate functions: [calculate\\_allocation\(\)](#), [calculate\\_coobs\(\)](#), [calculate\\_distance\(\)](#), [calculate\\_pcomp\(\)](#), [calculate\\_pop\(\)](#), [calculate\\_representation\(\)](#)

**Examples**

```
#--- Load raster ---#
r <- system.file("extdata", "mraster.tif", package = "sgsR")
mr <- terra::rast(r)

calculate_sampsize(
  mraster = mr,
```

```

    rse = 0.01
  )

  calculate_sampsize(
    mraster = mr
  )

  calculate_sampsize(
    mraster = mr,
    rse = 0.025,
    start = 0.01,
    end = 0.08,
    increment = 0.01
  )

  #--- higher variance leads to need for more samples ---#

```

---

extract_metrics	<i>Extract metrics</i>
-----------------	------------------------

---

## Description

Extract metric values to existing samples

## Usage

```

extract_metrics(
  mraster,
  existing,
  quiet = FALSE,
  data.frame = FALSE,
  filename = NULL,
  overwrite = FALSE
)

```

## Arguments

mraster	spatRaster. Metrics Raster.
existing	sf 'POINT'. Existing plot network.
quiet	Logical. If TRUE the user will not get messages about samples with NA values.
data.frame	Logical. Output as data.frame if TRUE
filename	Character. Path to write output samples.
overwrite	Logical. Choice to overwrite existing filename if it exists.

## Value

An sf or data.frame object of samples with metrics attributes.

**Note**

If `data.frame = TRUE` output will be written using `write.table`

**Author(s)**

Tristan R.H. Goodbody

**See Also**

Other extract functions: `extract_strata()`

**Examples**

```
#--- Load mraster ---#
r <- system.file("extdata", "mraster.tif", package = "sgsR")
mr <- terra::rast(r)

#' #--- load existing samples ---#
e <- system.file("extdata", "existing.shp", package = "sgsR")
e <- sf::st_read(e)

extract_metrics(mraster = mr,
                existing = e)
```

---

extract\_strata

*Extract strata*

---

**Description**

Extract stratum values to existing samples

**Usage**

```
extract_strata(
  sraster,
  existing,
  quiet = FALSE,
  data.frame = FALSE,
  filename = NULL,
  overwrite = FALSE
)
```



**Arguments**

sraster	spatRaster. Stratification raster.
existing	sf 'POINT'. Existing plot network.
quiet	Logical. If TRUE the user will not get messages about samples with NA values.
data.frame	Logical. Output as data.frame if TRUE
filename	Character. Path to write output samples.
overwrite	Logical. Choice to overwrite existing filename if it exists.

**Value**

An sf or data.frame object of samples with strata attribute.

**Note**

If data.frame = TRUE output will be written using [write.table](#)

**Author(s)**

Tristan R.H. Goodbody

**See Also**

Other extract functions: [extract\\_metrics\(\)](#)

**Examples**

```
#--- Load sraster ---#
r <- system.file("extdata", "sraster.tif", package = "sgsR")
sr <- terra::rast(r)

#--- load existing samples ---#
e <- system.file("extdata", "existing.shp", package = "sgsR")
e <- sf::st_read(e)

extract_strata(sraster = sr,
              existing = e)
```

**Description**

Perform the adapted Hypercube Evaluation of a Legacy Sample (ahels) algorithm using existing site data and raster metrics. New samples are allocated based on quantile ratios between the existing sample and covariate dataset.

**Usage**

```
sample_ahels(
  mraster,
  existing,
  nQuant = 10,
  nSamp = NULL,
  threshold = 0.9,
  tolerance = 0,
  matrices = NULL,
  plot = FALSE,
  details = FALSE,
  filename = NULL,
  overwrite = FALSE
)
```

**Arguments**

mraster	spatRaster. ALS metrics raster.
existing	sf 'POINT'. Existing plot network.
nQuant	Numeric. Number of quantiles to divide covariates and samples into. Quantiles that do not cover at least 1 percent of the area of interest will be excluded and be returned as NA.
nSamp	Numeric. Maximum number of new samples to allocate.
threshold	Numeric. Sample quantile ratio threshold. After the threshold default = 0.9 is reached, no additional samples will be added. Values close to 1 can cause the algorithm to continually loop.
tolerance	Numeric. Allowable tolerance ( $\leq 0.1$ (10 added until the $1 - \text{tolerance}$ density is reached. If threshold is used, samples will be added until the threshold - tolerance value is reached. This parameter allows the user to define a buffer around desired quantile densities to permit the algorithm to not add additional samples if quantile density is very close to 1, or user-defined threshold.
matrices	List. Quantile and covariance matrices generated from <code>calculate_pop(mraster = mraster, nQuant = nQuant)</code> . Both <code>mraster</code> & <code>nQuant</code> inputs must be the same to supply the covariance matrix. Supplying the matrix allows users with very large <code>mrasters</code> to pre-process the covariance matrix to avoid longer sampling processing times.
plot	Logical. Plots samples of type 'existing' (if provided; crosses) and 'new' (circles) along with <code>mraster</code> .
details	Logical. If FALSE (default) output is sf object of systematic samples. If TRUE returns a list of sf objects where <code>tessellation</code> is the tessellation grid for sampling, and <code>samples</code> are the systematic samples.
filename	Character. Path to write output samples.
overwrite	Logical. Choice to overwrite existing filename if it exists.

**Value**

Returns sf point object with existing samples and supplemental samples added by the ahels algorithm.

**Note**

Messages in the algorithm will state that samples have been added to under-represented quantiles. The number between square brackets that follow represent the matrix row and column respectively that can be printed using `details = TRUE`.

In some cases, generally when a single metric is used as `mraster`, sampling ratios all be  $\geq 1$  before the `nSamp` number of samples are allocated. The algorithm will stop in this scenario.

Special thanks to Dr. Brendan Malone for the original implementation of this algorithm.

**Author(s)**

Tristan R.H. Goodbody

**References**

Malone BP, Minansy B, Brungard C. 2019. Some methods to improve the utility of conditioned Latin hypercube sampling. PeerJ 7:e6451 DOI 10.7717/peerj.6451

**See Also**

Other sample functions: [sample\\_balanced\(\)](#), [sample\\_clhs\(\)](#), [sample\\_existing\(\)](#), [sample\\_nc\(\)](#), [sample\\_srs\(\)](#), [sample\\_strat\(\)](#), [sample\\_systematic\(\)](#)

**Examples**

```
## Not run:
#--- Load raster and existing plots---#
r <- system.file("extdata", "mraster.tif", package = "sgsR")
mr <- terra::rast(r)

e <- system.file("extdata", "existing.shp", package = "sgsR")
e <- sf::st_read(e)

sample_ahels(
  mraster = mr,
  existing = e,
  plot = TRUE
)

#--- supply quantile and covariance matrices ---#
mat <- calculate_pop(mraster = mr)

sample_ahels(
  mraster = mr,
  existing = e,
  matrices = mat,
```

```

    nSamp = 300
  )

  ## End(Not run)

```

---

 sample\_balanced

*Balanced sampling*


---

## Description

Balanced raster sampling using `lcube` and `lpm2_kdtree` methods

## Usage

```

sample_balanced(
  mraster,
  nSamp,
  algorithm = "lpm2_kdtree",
  p = NULL,
  access = NULL,
  buff_inner = NULL,
  buff_outer = NULL,
  plot = FALSE,
  filename = NULL,
  overwrite = FALSE
)

```

## Arguments

<code>mraster</code>	spatRaster. ALS metrics raster.
<code>nSamp</code>	Numeric. Number of desired samples.
<code>algorithm</code>	Character. One of <code>lpm2_kdtree</code> , <code>lcube</code> , <code>lcubestratified</code> .
<code>p</code>	Numeric. Vector with length equal to the number of cells in <code>mraster</code> representing the inclusion probability for each candidate sample. Default = $nSamp / N$ , where $N$ is the number of cells.
<code>access</code>	sf 'LINESTRING' or 'MULTILINESTRING'. Access network.
<code>buff_inner</code>	Numeric. Inner buffer boundary specifying distance from access where plots cannot be sampled.
<code>buff_outer</code>	Numeric. Outer buffer boundary specifying distance from access where plots can be sampled.
<code>plot</code>	Logical. Plots output strata raster and visualized strata with boundary dividers.
<code>filename</code>	Character. Path to write stratified raster to disc.
<code>overwrite</code>	Logical. Specify whether filename should be overwritten on disc.

**Value**

An sf object with nSamp samples.

**Author(s)**

Tristan R.H. Goodbody

**References**

Anton Grafstrom and Jonathan Lisic (2019). *BalancedSampling: Balanced and Spatially Balanced Sampling*. R package version 1.5.5. <https://CRAN.R-project.org/package=BalancedSampling>

Jonathan Lisic and Anton Grafstrom (2018). *SamplingBigData: Sampling Methods for Big Data*. R package version 1.0.0. <https://CRAN.R-project.org/package=SamplingBigData>

Grafström, A. Lisic, J (2018). *BalancedSampling: Balanced and Spatially Balanced Sampling*. R package version 1.5.4. <http://www.antongrafstrom.se/balancedsampling>

**See Also**

Other sample functions: [sample\\_ahels\(\)](#), [sample\\_clhs\(\)](#), [sample\\_existing\(\)](#), [sample\\_nc\(\)](#), [sample\\_srs\(\)](#), [sample\\_strat\(\)](#), [sample\\_systematic\(\)](#)

**Examples**

```
#--- Load raster and existing plots---#
r <- system.file("extdata", "mraster.tif", package = "sgsR")
mr <- terra::rast(r)

a <- system.file("extdata", "access.shp", package = "sgsR")
ac <- sf::st_read(a)

sample_balanced(
  mraster = mr,
  nSamp = 200
)
```

---

sample\_clhs

*Conditioned Latin Hypercube Sampling*

---

**Description**

Conditioned Latin Hypercube Sampling using [clhs](#) functionality.

**Usage**

```

sample_clhs(
  mraster,
  nSamp,
  iter = 10000,
  cost = NULL,
  existing = NULL,
  access = NULL,
  buff_inner = NULL,
  buff_outer = NULL,
  plot = FALSE,
  details = FALSE,
  filename = NULL,
  overwrite = FALSE,
  ...
)

```

**Arguments**

<code>mraster</code>	spatRaster. ALS metrics raster.
<code>nSamp</code>	Numeric. Number of desired samples.
<code>iter</code>	Numeric. Value giving the number of iterations within the Metropolis-Hastings process.
<code>cost</code>	Numeric/Character. Index or name of covariate within <code>mraster</code> to be used to constrain cLHS sampling. If default (NULL), a cost constraint is not used.
<code>existing</code>	sf 'POINT'. Existing plot network.
<code>access</code>	sf 'LINESTRING' or 'MULTILINESTRING'. Access network.
<code>buff_inner</code>	Numeric. Inner buffer boundary specifying distance from access where plots cannot be sampled.
<code>buff_outer</code>	Numeric. Outer buffer boundary specifying distance from access where plots can be sampled.
<code>plot</code>	Logical. Plots output strata raster with samples.
<code>details</code>	Logical. If FALSE (default) output is only stratification raster. If TRUE return a list where <code>\$details</code> is additional stratification information and <code>\$raster</code> is the output stratification spatRaster.
<code>filename</code>	Character. Path to write output samples.
<code>overwrite</code>	Logical. Choice to overwrite existing <code>filename</code> if it exists.
<code>...</code>	Additional arguments for <code>clhs</code> sampling. See <a href="#">clhs</a> .

**Value**

An sf object with `nSamp` stratified samples.

**Author(s)**

Tristan R.H. Goodbody

## References

Minasny, B. and McBratney, A.B. 2006. A conditioned Latin hypercube method for sampling in the presence of ancillary information. *Computers and Geosciences*, 32:1378-1388.

Minasny, B. and A. B. McBratney, A.B.. 2010. Conditioned Latin Hypercube Sampling for Calibrating Soil Sensor Data to Soil Properties. In: *Proximal Soil Sensing, Progress in Soil Science*, pages 111-119.

Roudier, P., Beaudette, D.E. and Hewitt, A.E. 2012. A conditioned Latin hypercube sampling algorithm incorporating operational constraints. In: *Digital Soil Assessments and Beyond. Proceedings of the 5th Global Workshop on Digital Soil Mapping*, Sydney, Australia.

## See Also

Other sample functions: [sample\\_ahels\(\)](#), [sample\\_balanced\(\)](#), [sample\\_existing\(\)](#), [sample\\_nc\(\)](#), [sample\\_srs\(\)](#), [sample\\_strat\(\)](#), [sample\\_systematic\(\)](#)

## Examples

```

#--- Load raster and existing plots---#
r <- system.file("extdata", "mraster.tif", package = "sgsR")
mr <- terra::rast(r)

e <- system.file("extdata", "existing.shp", package = "sgsR")
e <- sf::st_read(e)

a <- system.file("extdata", "access.shp", package = "sgsR")
ac <- sf::st_read(a)

sample_clhs(
  mraster = mr,
  nSamp = 200,
  plot = TRUE,
  iter = 100
)

sample_clhs(
  mraster = mr,
  nSamp = 400,
  existing = e,
  iter = 250,
  details = TRUE
)

```

---

sample\_existing

*Sample existing*

---

## Description

Sub-sample an existing sample using [clhs](#) functionality.

**Usage**

```

sample_existing(
  existing,
  nSamp,
  raster = NULL,
  cost = NULL,
  access = NULL,
  buff_inner = NULL,
  buff_outer = NULL,
  iter = 10000,
  plot = FALSE,
  details = FALSE,
  filename = NULL,
  overwrite = FALSE,
  ...
)

```

**Arguments**

existing	sf 'POINT'. Existing plot network.
nSamp	Numeric. Number of desired samples.
raster	spatRaster. Raster used to define population distributions.
cost	Numeric/Character. Index or name of covariate within mraster to be used to constrain cLHS sampling. If default (NULL), a cost constraint is not used.
access	sf. Road access network - must be lines.
buff_inner	Numeric. Inner buffer boundary specifying distance from access where plots cannot be sampled.
buff_outer	Numeric. Outer buffer boundary specifying distance from access where plots can be sampled.
iter	Numeric. Value giving the number of iterations within the Metropolis-Hastings process.
plot	Logical. Plots output strata raster with samples.
details	Logical. If FALSE (default) output is sf object of systematic samples. If TRUE returns a list of sf objects where tessellation is the tessellation grid for sampling, and samples are the systematic samples.
filename	Character. Path to write output samples.
overwrite	Logical. Choice to overwrite existing filename if it exists.
...	Additional arguments for clhs sampling. See <a href="#">clhs</a> .

**Value**

An sf object of samples



**Note**

If providing only existing - all attributes will be used for sampling. Remove attributes not indented for sampling prior to using this algorithm.

**Author(s)**

Tristan R.H. Goodbody

**See Also**

Other sample functions: [sample\\_ahels\(\)](#), [sample\\_balanced\(\)](#), [sample\\_clhs\(\)](#), [sample\\_nc\(\)](#), [sample\\_srs\(\)](#), [sample\\_strat\(\)](#), [sample\\_systematic\(\)](#)

**Examples**

```
#--- Load raster ---#
r <- system.file("extdata", "mraster.tif", package = "sgsR")
mr <- terra::rast(r)

#--- generate an existing sample ---#
e <- sample_systematic(raster = mr, cellsize = 200)

#--- perform sub-sampling ---#
sample_existing(existing = e,
                raster = mr,
                nSamp = 50)

#--- extract metrics to sample ---#
e <- extract_metrics(mr, e)

#--- perform sub-sampling ---#
sample_existing(existing = e,
                nSamp = 30,
                plot = TRUE)
```

---

sample\_nc

*Nearest centroid (NC) sampling*

---

**Description**

Sampling using the nearest centroid (NC) approach described in Melville & Stone (2016).

**Usage**

```
sample_nc(
  mraster,
  nSamp,
  k = 1,
```

```

iter = 500,
algorithm = "Lloyd",
access = NULL,
buff_inner = NULL,
buff_outer = NULL,
plot = FALSE,
details = FALSE,
filename = NULL,
overwrite = FALSE
)

```

### Arguments

mraster	spatRaster. ALS metrics raster.
nSamp	Numeric. Number of desired samples.
k	Numeric. The number of nearest neighbours to take for each k-means center. When $k = 1$ (default), the output number of samples will match nSamp. Increases to $k$ results in a multiplicative result total number of samples $nSamp * k$ .
iter	Numeric. The maximum number of kmeans iterations allowed.
algorithm	Character. Lloyd (default) or MacQueen kmeans algorithms.
access	sf 'LINESTRING' or 'MULTILINESTRING'. Access network.
buff_inner	Numeric. Inner buffer boundary specifying distance from access where plots cannot be sampled.
buff_outer	Numeric. Outer buffer boundary specifying distance from access where plots can be sampled.
plot	Logical. Plot mraster with allocated samples.
details	Logical. If FALSE (default) output is only stratification raster. If TRUE return a list where \$details is additional stratification information and \$raster is the output stratification spatRaster.
filename	Character. Path to write stratified raster to disc.
overwrite	Logical. Specify whether filename should be overwritten on disc.

### Value

An sf object with nSamp randomly sampled points.

### Note

When details = TRUE, a list is returned where:

1. samples output nearest centroid samples with 'kcenter' attribute linking to associated kmeans centers.
2. kmeans is a list output of the [kmeans](#) function
3. centers Un-scaled kmeans center values for each layer in mraster with 'kcenter' attribute to link with the same attribute in samples.
4. kplot is a ggplot scatterplot object visualizing the kmeans centers and associated nearest neighbor samples.

**Author(s)**

Tristan R.H. Goodbody

**References**

G. Melville & C. Stone (2016) Optimising nearest neighbour information—a simple, efficient sampling strategy for forestry plot imputation using remotely sensed data, *Australian Forestry*, 79:3, 217-228, DOI: 10.1080/00049158.2016.1218265

**See Also**

Other sample functions: [sample\\_ahels\(\)](#), [sample\\_balanced\(\)](#), [sample\\_clhs\(\)](#), [sample\\_existing\(\)](#), [sample\\_srs\(\)](#), [sample\\_strat\(\)](#), [sample\\_systematic\(\)](#)

**Examples**

```
#--- Load raster and access files ---#
r <- system.file("extdata", "mraster.tif", package = "sgsR")
mr <- terra::rast(r)

#--- perform simple random sampling ---#
sample_nc(
  mraster = mr,
  nSamp = 5,
)
```

---

sample\_srs

*Simple random sampling*

---

**Description**

Randomly sample within a stratification raster extent.

**Usage**

```
sample_srs(
  raster,
  nSamp,
  mindist = NULL,
  access = NULL,
  buff_inner = NULL,
  buff_outer = NULL,
  plot = FALSE,
  filename = NULL,
  overwrite = FALSE
)
```

**Arguments**

raster	spatRaster. Raster to be used for random sampling.
nSamp	Numeric. Number of desired samples.
mindist	Numeric. Minimum allowable distance between selected samples. Default = NULL.
access	sf 'LINESTRING' or 'MULTILINESTRING'. Access network.
buff_inner	Numeric. Inner buffer boundary specifying distance from access where plots cannot be sampled.
buff_outer	Numeric. Outer buffer boundary specifying distance from access where plots can be sampled.
plot	Logical. Plots output strata raster with samples.
filename	Character. Path to write output samples.
overwrite	Logical. Choice to overwrite existing filename if it exists.

**Value**

An sf object with nSamp randomly sampled points.

**Author(s)**

Tristan R.H. Goodbody & Martin Queinnec

**See Also**

Other sample functions: [sample\\_ahels\(\)](#), [sample\\_balanced\(\)](#), [sample\\_clhs\(\)](#), [sample\\_existing\(\)](#), [sample\\_nc\(\)](#), [sample\\_strat\(\)](#), [sample\\_systematic\(\)](#)

**Examples**

```
#--- Load raster and access files ---#
r <- system.file("extdata", "sraster.tif", package = "sgsR")
sr <- terra::rast(r)

a <- system.file("extdata", "access.shp", package = "sgsR")
ac <- sf::st_read(a)

#--- perform simple random sampling ---#
sample_srs(
  raster = sr,
  nSamp = 200,
)
```

---

sample_strat	<i>Stratified sampling</i>
--------------	----------------------------

---

### Description

Sampling based on a stratified raster.

### Usage

```
sample_strat(
  sraster,
  nSamp,
  allocation = "prop",
  method = "Queinnec",
  weights = NULL,
  force = FALSE,
  mraster = NULL,
  mindist = NULL,
  existing = NULL,
  include = FALSE,
  remove = FALSE,
  access = NULL,
  buff_inner = NULL,
  buff_outer = NULL,
  wrow = 3,
  wcol = 3,
  plot = FALSE,
  details = FALSE,
  filename = NULL,
  overwrite = FALSE
)
```

### Arguments

sraster	spatRaster. Stratification raster to be used for sampling.
nSamp	Numeric. Number of desired samples. existing, include and force influence this value.
allocation	Character. Allocation algorithm to be used. Either prop (default) for proportional allocation, optim for optimal allocation (equal sampling cost), equal for equal number of samples (defined by nSamp) for each strata, or "manual" for user defined strata weights defined using weights.
method	Character. Sampling design approach "Queinnec" (default) or "random". "Queinnec" method is described in notes below. "random" performs traditional stratified random sampling where probability to sample each cell within each stratum is equal assuming default parameters for mindist. existing, include, remove are ignored when method = "random".

weights	Numeric. Only applicable when allocation = "manual". Vector of weights where $\text{sum}(\text{weights}) == 1$ . Vector length must be equal to the number of unique strata where the first numeric value corresponds to stratum 1, second stratum 2 etc.
force	Logical. Default = FALSE - force nSamp to be exactly the user defined value in cases where nSamp and sraster strata count are not equally divisible. Additional samples often need to be allocated or removed based on rounding differences resulting from proportional differences between nSamp and strata coverages in sraster. In these instances samples are either added to strata with the lowest number of samples or are removed from strata with the highest number of samples. Has no effect when existing is provided.
mraster	spatRaster. ALS metric raster. Required when allocation = optim.
mindist	Numeric. Minimum allowable distance between selected samples. Default = NULL.
existing	sf 'POINT' or data.frame. Existing plot network.
include	Logical. If TRUE include existing plots in nSamp total.
remove	Logical. If TRUE randomly remove samples from over represented strata to meet allocated sample numbers. Used only when existing and include are both TRUE.
access	sf 'LINESTRING' or 'MULTILINESTRING'. Access network.
buff_inner	Numeric. Inner buffer boundary specifying distance from access where plots cannot be sampled.
buff_outer	Numeric. Outer buffer boundary specifying distance from access where plots can be sampled.
wrow	Numeric. Number of row in the focal window (default = 3).
wcol	Numeric. Number of columns in the focal window (default = 3).
plot	Logical. Plots samples of type 'existing' (if provided; crosses) and 'new' (circles) along with sraster.
details	Logical. If FALSE (default) output is sf object of stratified samples. If TRUE return a list where \$details additional sampling information and \$raster is an sf object of stratified samples.
filename	Character. Path to write output samples.
overwrite	Logical. Choice to overwrite existing filename if it exists.

**Value**

An sf object with nSamp stratified samples.

**Note**

The sampling is performed in 2 stages when method = "Queinnee":

1. Rule 1 - Sample within grouped stratum pixels defined within the wrow, wcol parameters
2. Rule 2 - If no samples exist to satisfy Rule 1 individual stratum pixels are sampled.

The rule applied to allocate each sample is defined in the rule attribute of output samples.

existing may contain samples that fall in sraster cells that are 'NA'. If this occurs and include = TRUE, 'NA' samples are separated during sampling and re-appended at the end of the sampling process.

If the sraster provided contains factor values, the algorithm will automatically convert these into the numeric factor levels and perform sampling using those values. The categories (factor values) will be extracted and appended to the algorithm output as the 'category' attribute.

### Author(s)

Tristan R.H. Goodbody & Martin Queinnec

### References

Queinnec, M., White, J. C., & Coops, N. C. (2021). Comparing airborne and spaceborne photon-counting LiDAR canopy structural estimates across different boreal forest types. *Remote Sensing of Environment*, 262 (August 2020), 112510. <https://doi.org/10.1016/j.rse.2021.112510>

### See Also

Other sample functions: [sample\\_ahels\(\)](#), [sample\\_balanced\(\)](#), [sample\\_clhs\(\)](#), [sample\\_existing\(\)](#), [sample\\_nc\(\)](#), [sample\\_srs\(\)](#), [sample\\_systematic\(\)](#)

### Examples

```
#--- Load raster and access files ---#
r <- system.file("extdata", "sraster.tif", package = "sgsR")
sr <- terra::rast(r)

#--- perform stratified sampling random sampling ---#
sraster <- sample_strat(
  sraster = sr,
  nSamp = 50
)
```

---

sample\_systematic      *Systematic sampling*

---

### Description

Systematic sampling with random start point and translation within a square or hexagonal tessellation.

**Usage**

```

sample_systematic(
  raster,
  cellsize,
  square = TRUE,
  location = "centers",
  force = FALSE,
  access = NULL,
  buff_inner = NULL,
  buff_outer = NULL,
  plot = FALSE,
  filename = NULL,
  overwrite = FALSE,
  details = FALSE,
  ...
)

```

**Arguments**

raster	spatRaster. Raster used to define extent of fishnet grid.
cellsize	Numeric. Desired cellsize for tessellation.
square	Logical. Tessellation shape. Default is regular square grid, if FALSE hexagons are used.
location	Character. Sample location within tessellation. Default = "centers") returns samples at tessellation centers, "corners" - corners of tessellation are returned, "random" - samples are randomly located within tessellations.
force	Logical. Only applies when location = "random". If TRUE, random samples are forced to fall in areas where raster does not have NA values. This will considerably slow processing.
access	sf. Road access network - must be lines.
buff_inner	Numeric. Inner buffer boundary specifying distance from access where plots cannot be sampled.
buff_outer	Numeric. Outer buffer boundary specifying distance from access where plots can be sampled.
plot	Logical. Plots output strata raster with samples.
filename	Character. Path to write output samples.
overwrite	Logical. Choice to overwrite existing filename if it exists.
details	Logical. If FALSE (default) output is sf object of systematic samples. If TRUE returns a list of sf objects where tessellation is the tessellation grid for sampling, and samples are the systematic samples.
...	Additional arguments for <code>st_make_grid</code> . Options include <code>offset</code> to offset grid by providing lower left coordinates.

**Value**

An sf object with sampled points over a tessellation.



**Note**

Specifying `location = "random"` can result in tessellations with no samples. This results from raster have NA values at the random location chosen. Using `force = TRUE` removes areas of NA from sampling entirely, but considerably slows processing speeds.

**Author(s)**

Tristan R.H. Goodbody, Lukas Winiwarter

**See Also**

Other sample functions: [sample\\_ahels\(\)](#), [sample\\_balanced\(\)](#), [sample\\_clhs\(\)](#), [sample\\_existing\(\)](#), [sample\\_nc\(\)](#), [sample\\_srs\(\)](#), [sample\\_strat\(\)](#)

**Examples**

```
#--- Load raster and access files ---#
r <- system.file("extdata", "sraster.tif", package = "sgsR")
sr <- terra::rast(r)

a <- system.file("extdata", "access.shp", package = "sgsR")
ac <- sf::st_read(a)

e <- system.file("extdata", "existing.shp", package = "sgsR")
e <- sf::st_read(e)

#--- perform grid sampling ---#
sample_systematic(
  raster = sr,
  cellsize = 1000
)

sample_systematic(
  raster = sr,
  cellsize = 1000,
  square = FALSE,
  location = "corners",
  plot = TRUE
)

sample_systematic(
  raster = sr,
  cellsize = 1000,
  square = FALSE,
  location = "random"
)
```

---

 strat\_breaks

*Breaks stratification*


---

### Description

Stratify metrics raster using user defined breaks

### Usage

```
strat_breaks(
  mraster,
  mraster2 = NULL,
  breaks,
  breaks2 = NULL,
  plot = FALSE,
  details = FALSE,
  filename = NULL,
  overwrite = FALSE
)
```

### Arguments

mraster	Spatraster. Primary covariate raster to stratify.
mraster2	Spatraster. Secondary covariate raster to stratify.
breaks	Numeric. Vector of breakpoints for mraster.
breaks2	Numeric. Vector of breakpoints for mraster2 (if provided).
plot	Logical. Plots output strata raster and visualized strata with boundary dividers.
details	Logical. If FALSE (default) output is only stratification raster. If TRUE return a list where \$details is additional stratification information and \$raster is the output stratification spatRaster.
filename	Character. Path to write stratified raster to disc.
overwrite	Logical. Specify whether filename should be overwritten on disc.

### Value

Returns an output stratification spatRaster or a list when details = TRUE.

When a list is returned:

1. details is a list output of the [prcomp](#) function
2. raster is a stratified spatRaster based on quantiles
3. plot is a ggplot histogram object showing distribution and break points.

### Author(s)

Tristan R.H. Goodbody

**See Also**

Other stratify functions: [strat\\_kmeans\(\)](#), [strat\\_map\(\)](#), [strat\\_poly\(\)](#), [strat\\_quantiles\(\)](#)

**Examples**

```
#--- Load raster ---#
r <- system.file("extdata", "mraster.tif", package = "sgsR")
mr <- terra::rast(r)

#--- create vector breaks ---#
br.max <- c(3, 5, 11, 18)
br.sd <- c(1, 2, 5)

strat_breaks(
  mraster = mr$zq90,
  breaks = br.max,
  plot = TRUE,
  details = TRUE
)

strat_breaks(
  mraster = mr$zq90,
  mraster2 = mr$zsd,
  breaks = br.max,
  breaks2 = br.sd
)
```

---

strat\_kmeans

*k-means stratification*

---

**Description**

Stratify metrics raster using [kmeans](#) algorithm

**Usage**

```
strat_kmeans(
  mraster,
  nStrata,
  iter = 500,
  algorithm = "Lloyd",
  center = TRUE,
  scale = TRUE,
  plot = FALSE,
  details = FALSE,
  filename = NULL,
  overwrite = FALSE,
  ...
)
```

**Arguments**

mraster	spatRaster. ALS metrics raster.
nStrata	Character. Number of desired strata.
iter	Numeric. The maximum number of iterations allowed.
algorithm	Character. Lloyd (default) or MacQueen algorithms.
center	Logical. Value indicating whether the variables should be shifted to be zero centered.
scale	Logical. Value indicating whether the variables should be scaled to have unit variance.
plot	Logical. Plots output strata raster and visualized strata with boundary dividers.
details	Logical. If FALSE (default) output is only stratification raster. If TRUE return a list where \$details is additional stratification information and \$raster is the output stratification spatRaster.
filename	Character. Path to write stratified raster to disc.
overwrite	Logical. Specify whether filename should be overwritten on disc.
...	Additional arguments to be passed to <a href="#">kmeans</a> function.

**Value**

output stratification spatRaster, or a list when details = TRUE.

**Author(s)**

Tristan R.H. Goodbody

**See Also**

Other stratify functions: [strat\\_breaks\(\)](#), [strat\\_map\(\)](#), [strat\\_poly\(\)](#), [strat\\_quantiles\(\)](#)

**Examples**

```

#--- Load raster and access files ---#
r <- system.file("extdata", "mraster.tif", package = "sgsR")
mr <- terra::rast(r)

#--- perform stratification using k-means ---#
kmeans <- strat_kmeans(
  mraster = mr,
  nStrata = 5
)

kmeans <- strat_kmeans(
  mraster = mr,
  nStrata = 5,
  iter = 1000,
  algorithm = "MacQueen",
  details = TRUE

```

```
)
```

---

```
strat_map
```

```
Map 2 stratified rasters
```

---

## Description

Map stratified rasters to a combined stratification.

## Usage

```
strat_map(
  sraster,
  sraster2,
  stack = FALSE,
  filename = NULL,
  overwrite = FALSE,
  plot = FALSE,
  details = FALSE
)
```

## Arguments

<code>sraster</code>	spatRaster. Primary stratification raster.
<code>sraster2</code>	spatRaster. Secondary stratification raster.
<code>stack</code>	Logical. Default = FALSE. If TRUE, output raster will be 3 layers: <code>strata</code> , <code>strata2</code> , <code>stratamapped</code> .
<code>filename</code>	Character. Path to write stratified raster to disc.
<code>overwrite</code>	Logical. Specify whether filename should be overwritten on disc.
<code>plot</code>	Logical. Plots output <code>strata</code> raster and visualized <code>strata</code> with boundary dividers.
<code>details</code>	Logical. If FALSE (default) output is a mapped stratified spatRaster object. If TRUE return a list where <code>\$outRaster</code> is the mapped stratified raster, and <code>\$lookup</code> is the lookup table for the stratification.

## Value

A spatRaster object.

## Mapping

The mapping algorithm will take the stratification from `sraster` and combine it with overlying `strata` values in `sraster2`. This will result in a `stratamapped` attribute where the values from both inputs are combined.

i.e.

If `strata = 1` and `strata2 = 1` then `stratamapped = 11`.

If `strata = 2` and `strata2 = 14` then `stratamapped = 214`.

**Author(s)**

Tristan R.H. Goodbody, Robert Hijmans

**See Also**

Other stratify functions: [strat\\_breaks\(\)](#), [strat\\_kmeans\(\)](#), [strat\\_poly\(\)](#), [strat\\_quantiles\(\)](#)

**Examples**

```
#--- load input metrics raster ---#
raster <- system.file("extdata", "sraster.tif", package = "sgsR")
sraster <- terra::rast(raster)

#--- read polygon coverage ---#
poly <- system.file("extdata", "inventory_polygons.shp", package = "sgsR")
fri <- sf::st_read(poly)

#--- stratify polygon coverage ---#
#--- specify polygon attribute to stratify ---#

attribute <- "NUTRIENTS"

#--- specify features within attribute & how they should be grouped ---#
#--- as a single vector ---#

features <- c("poor", "rich", "medium")

srasterfri <- strat_poly(
  poly = fri,
  attribute = attribute,
  features = features,
  raster = sraster
)

#--- map srasters ---#
strat_map(
  sraster = srasterfri,
  sraster2 = sraster
)

strat_map(
  sraster = srasterfri,
  sraster2 = sraster,
  stack = TRUE,
  details = TRUE
)
```

---

strat_poly	<i>Stratify using polygons</i>
------------	--------------------------------

---

**Description**

Stratify based on polygon coverage attributes and features.

**Usage**

```
strat_poly(
  poly,
  attribute,
  features,
  raster,
  filename = NULL,
  overwrite = FALSE,
  plot = FALSE,
  details = FALSE
)
```

**Arguments**

poly	sf. Input polygon coverage. e.g. - forest resources inventory coverage.
attribute	Character. Name of attribute within poly that will be stratified.
features	Vector / list of vectors. Features within attribute to guide stratification.
raster	spatRaster. Raster template to enable polygon to raster conversion.
filename	Character. Path to write stratified raster to disc.
overwrite	Logical. Specify whether filename should be overwritten on disc.
plot	Logical. Plots output spatRaster.
details	Logical. If FALSE (default) output is spatRaster object of stratified polygon attributes. If TRUE return a list where \$outRaster is the stratified attributes, \$lookUp is the lookup table for the stratification, and poly is the defined polygon attribute with corresponding features / strata

**Value**

A spatRaster object.

**Author(s)**

Tristan R.H. Goodbody

**See Also**

Other stratify functions: [strat\\_breaks\(\)](#), [strat\\_kmeans\(\)](#), [strat\\_map\(\)](#), [strat\\_quantiles\(\)](#)

## Examples

```
#--- load input metrics raster ---#
raster <- system.file("extdata", "sraster.tif", package = "sgsR")
sraster <- terra::rast(raster)

#--- read polygon coverage ---#
poly <- system.file("extdata", "inventory_polygons.shp", package = "sgsR")
fri <- sf::st_read(poly)

#--- stratify polygon coverage ---#
#--- specify polygon attribute to stratify ---#

attribute <- "NUTRIENTS"

#--- specify features within attribute & how they should be grouped ---#
#--- as a single vector ---#

features <- c("poor", "rich", "medium")

srasterpoly <- strat_poly(
  poly = fri,
  attribute = attribute,
  features = features,
  raster = sraster
)

#--- or as multiple lists ---#

g1 <- "poor"
g2 <- c("rich", "medium")

features <- list(g1, g2)

srasterpoly <- strat_poly(
  poly = fri,
  attribute = attribute,
  features = features,
  raster = sraster,
  details = TRUE
)
```

---

strat\_quantiles

*Quantiles stratification*

---

## Description

Stratify metric raster using metric quantiles.



**Usage**

```
strat_quantiles(
  mraster,
  mraster2 = NULL,
  nStrata,
  nStrata2 = NULL,
  plot = FALSE,
  details = FALSE,
  samp = 0.2,
  filename = NULL,
  overwrite = FALSE
)
```

**Arguments**

<code>mraster</code>	spatRaster. ALS metrics raster.
<code>mraster2</code>	Spatraster. Secondary covariate raster to stratify.
<code>nStrata</code>	Numeric. Number of quantiles to stratify <code>mraster</code> .
<code>nStrata2</code>	Numeric. Number of quantiles to stratify <code>mraster2</code> .
<code>plot</code>	Logical. Plots output strata raster and visualized strata with boundary dividers.
<code>details</code>	Logical. If FALSE (default) output is only stratification raster. If TRUE return a list where <code>\$details</code> is additional stratification information and <code>\$raster</code> is the output stratification spatRaster.
<code>samp</code>	Numeric. Determines proportion of cells to plot in scatterplot (default = 0.2). Lower values reduce visualization time.
<code>filename</code>	Character. Path to write stratified raster to disc.
<code>overwrite</code>	Logical. Specify whether filename should be overwritten on disc.

**Value**

Returns an output stratification spatRaster or a list when `details = TRUE`.

When a list is returned:

1. `details` generates is a strata lookUp table for the stratification.
2. `raster` is a stratified spatRaster based on quantiles
3. `plot` is a ggplot histogram / scatter plot object (depends on whether `metric2` was supplied). Histogram shows distribution and break points while scatter plot shows colour coded and strata boundaries.

**Author(s)**

Tristan R.H. Goodbody

**See Also**

Other stratify functions: [strat\\_breaks\(\)](#), [strat\\_kmeans\(\)](#), [strat\\_map\(\)](#), [strat\\_poly\(\)](#)

**Examples**

```
#--- Load raster and existing plots---#
r <- system.file("extdata", "mraster.tif", package = "sgsR")
mr <- terra::rast(r)

strat_quantiles(
  mraster = mr$q90,
  nStrata = 4,
  plot = TRUE
)

strat_quantiles(
  mraster = mr$q90,
  mraster2 = mr$zsd,
  nStrata = 3,
  nStrata2 = 4
)
```

# Index

- \* **analyze functions**
    - calculate\_lhsOpt, 7
  - \* **calculate functions**
    - calculate\_allocation, 2
    - calculate\_coobs, 4
    - calculate\_distance, 6
    - calculate\_pcomp, 9
    - calculate\_pop, 10
    - calculate\_representation, 12
    - calculate\_sampsize, 13
  - \* **extract functions**
    - extract\_metrics, 15
    - extract\_strata, 16
  - \* **sample functions**
    - sample\_ahels, 17
    - sample\_balanced, 20
    - sample\_clhs, 21
    - sample\_existing, 23
    - sample\_nc, 25
    - sample\_srs, 27
    - sample\_strat, 29
    - sample\_systematic, 31
  - \* **stratify functions**
    - strat\_breaks, 34
    - strat\_kmeans, 35
    - strat\_map, 37
    - strat\_poly, 39
    - strat\_quantiles, 40
- calculate\_allocation, 2, 5, 7, 10–12, 14
- calculate\_coobs, 4, 4, 7, 10–12, 14
- calculate\_distance, 4, 5, 6, 10–12, 14
- calculate\_lhsOpt, 7, 10, 11
- calculate\_pcomp, 4, 5, 7, 9, 11, 12, 14
- calculate\_pop, 4, 5, 7, 8, 10, 10, 12, 14
- calculate\_representation, 4, 5, 7, 10, 11, 12, 14
- calculate\_sampsize, 4, 5, 7, 10–12, 13
- clhs, 8, 21–24
- extract\_metrics, 15, 17
- extract\_strata, 16, 16
- kmeans, 26, 35, 36
- lcube, 20
- lpm2\_kdtree, 20
- prcomp, 9, 34
- sample\_ahels, 10, 17, 21, 23, 25, 27, 28, 31, 33
- sample\_balanced, 19, 20, 23, 25, 27, 28, 31, 33
- sample\_clhs, 6, 19, 21, 21, 25, 27, 28, 31, 33
- sample\_existing, 19, 21, 23, 23, 27, 28, 31, 33
- sample\_nc, 19, 21, 23, 25, 25, 28, 31, 33
- sample\_srs, 19, 21, 23, 25, 27, 27, 31, 33
- sample\_strat, 19, 21, 23, 25, 27, 28, 29, 33
- sample\_systematic, 19, 21, 23, 25, 27, 28, 31, 31
- st\_make\_grid, 32
- strat\_breaks, 34, 36, 38, 39, 41
- strat\_kmeans, 35, 35, 38, 39, 41
- strat\_map, 35, 36, 37, 39, 41
- strat\_poly, 35, 36, 38, 39, 41
- strat\_quantiles, 35, 36, 38, 39, 40
- write.table, 16, 17