Package: BASSr (via r-universe)

May 18, 2026

Type Package

Title Benefit Applied Strategic Sampling

Version 0.3.0

Author David Hope

Maintainer David Hope <david.hope@ec.gc.ca>

Description Sampling design for generating a spatially dispersed sample that is representative across multiple variables.

License GPL (>= 3)

LinkingTo Rcpp, testthat

Depends R (>= 4.1.0),

Imports dplyr (>= 1.0), purrr (>= 1.0.1), tidyr (>= 1.0.0), rlang (>= 0.4.0), spsurvey (>= 5.0), sf (>= 0.8), glue, methods, Rcpp, magrittr, stringr (>= 1.5.0), nngeo (>= 0.4.7), units (>= 0.8.5)

Suggests ggplot2, knitr, rmarkdown, testthat (>= 3.0.0), withr, palmerpenguins, patchwork, terra, forcats, DiagrammeR, furrr, future, xml2

Encoding UTF-8

LazyData true

VignetteBuilder knitr

RoxygenNote 7.3.2

Roxygen list(markdown = TRUE)

URL https://github.com/dhope/BASSr, http://davidhope.ca/BASSr/

BugReports https://github.com/dhope/BASSr/issues

Config/testthat/edition 3

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev make libicu-dev libuv1-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev

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

Date/Publication 2025-12-19 22:15:10 UTC

RemoteUrl https://github.com/dhope/BASSr

RemoteRef HEAD

RemoteSha 3fadb7f4a5def956b2bbe0bbf0ea541f0e25e7e5

Contents

`all_study_areas` *All of Ontario's Study Areas*

Description

A simple features (package SF) object with all of Ontario's Study Areas and the associated Land Cover

Usage

```
all_study_areas
```

Format

A data frame with 746 rows and 19 variables:

StudyAreaID Unique Identifier for each study area

TOT_HA Total hectares in study area

D_CLC15 Dominant land cover in study area

LC.. Hectares covered by landcover type (00-18) ...

geometry SF geometry

Source

Hexagons generated in R, landcover extracted from National Land Cover 2015 (https://open.canada.ca/data/en/dataset/4e615eae-b90c-420b-adee-2ca35896caf6).

`allhexes` *Run speed bass on all hexagons and all samples*

Description

Run speed bass on all hexagons and all samples

Usage

```
allhexes(hexes, samples, total, w, printDets = FALSE)
```

Arguments

hexes	Matrix of hexagon land covers. Rows are hexagons, columns are land cover types
samples	Matrix of hexagon land covers from random sample. Rows are hexagons, columns are land cover types
total	Vector of total land cover. values are individual land cover types
w	A vector of weights for each land cover value
printDets	print details of function calculation. For debugging.

BASSr *BASSr: Benefit Applied Strategic Sampling in R*

Description

The BASSr package implements the Benefit Applied Strategic Sampling

See Also

Useful links:

- https://github.com/dhope/BASSr
- http://davidhope.ca/BASSr/
- Report bugs at https://github.com/dhope/BASSr/issues

BASSr-defunct *BASSr defunct functions*

Description

BASSr defunct functions

Usage

```
clean_forBass(...)

run_full_BASS_w_selection()
```

Arguments

...	Original function arguments

BASSr-deprecated *BASSr deprecated functions*

Description

Deprecated These functions are no longer useful.

Usage

```
subsample_grts_and_calc_benefit()

extract_habitat_cost()

genraster()

noGRTS_BASS_run()

getresults_BASS()
```

See Also

BASSr-defunct

calculate_benefit *Calculate the benefit of a hexagon from grts results.*

Description

Calculate the benefit of a hexagon from grts results.

Usage

```
calculate_benefit(
  land_hex,
  samples,
  hex_id = hex_id,
  stratum_id = NULL,
  non_random_set = NULL,
  land_cover_weights = NULL,
  crs = 4326,
  coords = c("lon", "lat"),
  quiet = FALSE
)
```

Arguments

land_hex	(Spatial) Data frame. Land Cover data by hexagon. If non-spatial, will be converted to spatial sf data frame using the crs and coords arguments. Requires columns identifying the Hex ID as well as the Stratum ID (see hex_id and stratum_id respectively).
samples	(Spatial) Data frame. Results from draw_random_samples().
hex_id	Column. Identifies hexagon IDs (e.g., default hex_id).
stratum_id	Column. Identifies larger area (e.g., StudyAreaID or Province).
non_random_set	Character vector. hex_ids of hexagons to include as a non randomly selected set.
land_cover_weights	Data frame. Proportional weights (weights column) for specific types of landcover (lc column). lc should correspond to the same landcover column names as the hex data.
crs	Numeric, character, or sf/sfc. Coordinate reference system. Must be valid input to sf::st_crs().
coords	Character vector. Names of the columns containing X and Y coordinates (default c("lon", "lat")).
quiet	Logical. Whether to suppress progress messages.

Value

Spatial data frame with benefits per hex

Examples

```
# Using example data psu_hexagons and psu_samples

calculate_benefit(
  land_hex = psu_hexagons,
  samples = psu_samples,
  non_random_set = c("SA_09", "SA_22", "SA_47"))

# Specify a non-random set

calculate_benefit(
 land_hex = psu_hexagons,
 samples = psu_samples,
 non_random_set = c("SA_09", "SA_22", "SA_47"))

# Without GRTS

non_grts_samples <- draw_random_samples(
  land_hex = psu_hexagons,
  num_runs = 3,
  n_samples = 10,
  use_grts = FALSE)
```

```
calculate_benefit(
 land_hex = psu_hexagons,
 samples = non_grts_samples)
```

`calculate_inclusion_probs` *BASS cost benefit calculation*

Description

Calculate the cost-benefits and inclusion probabilities.

Usage

```
calculate_inclusion_probs(
  benefits,
  costs,
  hex_id = hex_id,
  stratum_id = NULL,
  omit_flag = NULL,
  benefit_weight = 0.5
)
```

Arguments

`benefits`	Spatial Data frame. Benefits associated with each hexagon (output of `calculate_benefits()`).
`costs`	Data frame. Costs for each hexagon in a RawCost format.
`hex_id`	Column. Identifies hexagon IDs (e.g., default `hex_id`).
`stratum_id`	Column. Identifies larger area (e.g., `StudyAreaID` or `Province`).
`omit_flag`	Column identifying hexes to omit (e.g., water hexes).
`benefit_weight`	Numeric. Weight assigned to benefit in the selection probabilities. 0.5 is equal weighting of cost and benefits. 1.0 is zero weighting to cost. Default 0.5.

Value

A data frame with full inclusion probabilities for each hex.

Examples

```
b <- calculate_benefit(land_hex = psu_hexagons,
                       samples = psu_samples)

inc <- calculate_inclusion_probs(
  benefits = b,
```

```
  costs = psu_costs)

# Omit water hexes (identified by column `water`)

inc <- calculate_inclusion_probs(
  benefits = b,
  costs = psu_costs,
  omit_flag = water)
```

`calculate_PPS_hab_inlc_pr` *Calculate Propbability Proportional to Size (PPS) inclusion probabilities*

Description

See for more information: https://www150.statcan.gc.ca/n1/en/pub/12-001-x/2011001/article/11450-eng.pdf?st=0oyBln55 or https://en.wikipedia.org/wiki/Probability-proportional-to-size_sampling

Usage

```
calculate_PPS_hab_inlc_pr(
  land_hex,
  hex_id = hex_id,
  stratum_id = NULL,
  quiet = FALSE
)
```

Arguments

`land_hex`	(Spatial) Data frame. Land Cover data by hexagon. If non-spatial, will be converted to spatial sf data frame using the `crs` and `coords` arguments. Requires columns identifying the Hex ID as well as the Stratum ID (see `hex_id` and `stratum_id` respectively).
`hex_id`	Column. Identifies hexagon IDs (e.g., default `hex_id`).
`stratum_id`	Column. Identifies larger area (e.g., `StudyAreaID` or `Province`).
`quiet`	Logical. Whether to suppress progress messages.

Value

tibble with selection weights from PPS

Examples

```
calculate_PPS_hab_inlc_pr(land_hex = psu_hexagons)
```

calculate_z_scores *Calculate z-scores for each hexagon by sum of individual z scores*

Description

Calculate z-scores for each hexagon by sum of individual z scores

Usage

```
calculate_z_scores(land_hex, hex_id, stratum_id = NULL, quiet = FALSE)
```

Arguments

land_hex	(Spatial) Data frame. Land Cover data by hexagon. If non-spatial, will be converted to spatial sf data frame using the crs and coords arguments. Requires columns identifying the Hex ID as well as the Stratum ID (see hex_id and stratum_id respectively).
hex_id	Column. Identifies hexagon IDs (e.g., default hex_id).
stratum_id	Column. Identifies larger area (e.g., StudyAreaID or Province).
quiet	Logical. Whether to suppress progress messages.

Value

data frame

Examples

```
calculate_z_scores(psu_hexagons, hex_id)
```

clean_land_cover *Clean land cover habitat data*

Description

This is a general function to clean land cover columns.

Usage

```
clean_land_cover(land_raw, pattern = "CLC15_", append = "", quiet = FALSE)
```

Arguments

land_raw	(Spatial) data frame. Land Cover data to be cleaned.
pattern	Character. Pattern to match and replace with 'LC'
append	Character. Text to append to end of land cover code
quiet	Logical. Whether to suppress progress messages.

Value

(Spatial) Data frame with cleaned land cover column names.

Examples

```
psu_hex_clean <- clean_land_cover(psu_hex_dirty, pattern = "CLC0013_")
```

---

clrfile *rgb colour codes to plot 2015 National Land Cover.*

Description

A dataset containing red, blue, green values associated with all the land cover types found in the 2015 Canadian National Land Cover Classification

Usage

```
clrfile
```

Format

A data frame with 20 rows and 6 variables:

Value price, in US dollars

red weight of the diamond, in carats

green weight of the diamond, in carats

blue weight of the diamond, in carats

rgb weight of the diamond, in carats

LCC_NAME weight of the diamond, in carats

Source

https://open.canada.ca/data/en/dataset/4e615eae-b90c-420b-adee-2ca35896caf6

`cost_vars` *Variables for cost estimation*

Description

Cost variables for estimate_cost_study_area

Usage

```
cost_vars
```

Format

A list with 18 variables used to calculate the cost model for N Ontario:

truck_cost_per_day Cost of truck use per day per crew, in dollars

truck_n_crews Number of crews for truck surveys

truck_arus_per_crew_per_day number of arus deployed per crew per day

atv_cost_per_day Cost of ATV use per day per crew, in dollars

atv_n_crews Number of crews for ATV surveys

atv_arus_per_crew_per_day Number of arus deployed per crew per day

helicopter_cost_per_hour Cost of helicopter rental per hour, in dollars

helicopter_max_km_from_base Maximum range of helicopter from fuelling base, in kilometres

helicopter_base_setup_cost_per_km Cost of setting up base for helicopter use with distance from airport, in km

helicopter_l_per_hour Helicopter fuel usage per hour, in litres

helicopter_crew_size Helicopter crew size

helicopter_aru_per_person_per_day Number of arus deployed per day per person

helicopter_relocation_speed Speed of movement of helicopter when relocating, in km per hour

helicopter_airport_cost_per_l Cost of heli fuel from an airport, in dollars per litre

helicopter_base_cost_per_l Cost of heli fuel from a basecamp, in dollars per litre

helicopter_2nd_base_cost_per_l Cost of heli fuel from a a remote fuel cache, in dollars per litre

helicopter_hours_flying_within_sa_per_day Number of hours a helicopter spends flying in the study area per day. ...

Source

Advice from Rich Russell

`create_hexes` *Create Hexagonal grid*

Description

Function that takes a landscape as an sf object and returns a hexagonal grid of a given size. Allowed inputs are hectres ('ha'), metres squared ('m2'), metres or kilometres ('m', or 'km') as the diameter of each hexagon

Usage

```
create_hexes(
  land,
  hex_size,
  units = NULL,
  hex_prefix = "SA_",
  linear_type = "short_diagonal"
)
```

Arguments

`land`	sf Spatial. Area over which to create hexagonal grid. Must be have a valid CRS. If lat/lon (i.e. CRS 4326) will be projected to 3347 to ensure proper units.
`hex_size`	Numeric or Units. Size of hexagon in area or diagonal diameter. Can be a bear number (see `units`) or a `units` object with units embedded. See details for specifics.
`units`	Character. Units of `hex_size` in `"m"`, `"m2"`, `"km"`, `"km2"`, or `"ha"`. Ignored if `hex_size` is a `units` object.
`hex_prefix`	Character. Text to prefix to hexagon IDs. Default "SA_" results in hexagon ids of "SA_01", etc.
`linear_type`	Character. Type of diameter to use when specifying linear hexagonal grid sizes. One of `"short_diagonal"` (default; for the short diameter from side to side, or centroid to centroid), or `"long_diagonal"` (for the long diameter from vertex to vertex passing through the centre of the hex).

Details

If `hex_size` is provided as a units object (i.e., `units::set_units(100, km^2)`) then the units can be any area or length unit recognized by the `units` package and convertible to `m` or `m2`. Otherwise, if using a bare number in `hex_size` and providing the units as a character in `units`, they must be one of `"m"`, `"m2"`, `"km"`, `"km2"`, or `"ha"`.

For example, `hex_size = units::set_units(100, ft)` will work, but `hex_size = 100, units = "ft"` will not.

Value

Returns a sf polygon layer of hexagons with unique IDs

Examples

```
library(sf)
library(ggplot2)
plot <- st_polygon(list(cbind(c(-90,-90,-85,-85,-90),
                              c(50,55,55,50,50)))) |>
  st_sfc(crs = 4326) |>
  st_transform(3347)

ggplot() +
  geom_sf(data = plot, fill = "white")

# Create grid by area - 1000km2
grid <- create_hexes(plot, hex_size = 1000, units = "km2")

# Check the area
st_area(grid[1,]) |>
  units::set_units("km2")

# Check the visual
ggplot() +
  geom_sf(data = plot, fill = "white") +
  geom_sf(data = grid, fill = NA)

# Create grid by diameter - 33.98088 km from side to side
grid2 <- create_hexes(plot, hex_size = 33.98088, units = "km")

# Check the area - Hah! A hexagon with the diameter of 33.98088 km has an area of ~1,000km
st_area(grid2[1,]) |>
  units::set_units("km2")

# Check the visual - Identical
ggplot() +
  geom_sf(data = plot, fill = "white") +
  geom_sf(data = grid2, fill = NA)

# Diameter of a 1000 km2 hexagon is
area_km2 <- 1000
(sqrt(2 * area_km2 / (3 * sqrt(3)))) * sqrt(3)

# Create grid by hectare
grid <- create_hexes(plot, hex_size = 40000, units = "ha")
ggplot() +
  geom_sf(data = plot, fill = "white") +
  geom_sf(data = grid, fill = NA)

# Create grid with pre-set units
area <- units::set_units(1000, "km2", mode = "character")
grid <- create_hexes(plot, hex_size = area)
ggplot() +
  geom_sf(data = plot, fill = "white") +
  geom_sf(data = grid, fill = NA)
```

create_sites *Create sampling sites within hexagons*

Description

Creates a grid of sites within the hexagon grid cells created by `create_hexes()`. These sites can then be sampled with `sample_sites()`.

Usage

```
create_sites(hexes, spacing = NULL, n = NULL, hex_id = hex_id)
```

Arguments

`hexes`	Spatial Data frame. Hexagon grid across sampling region. Requires column identifying the hexagon IDs (see `hex_id`)
`spacing`	Numeric. Distance between sites. Units are assumed to be those of `hex` spatial data frame.
`n`	Numeric. *Approximate* number of sites to create within a hex grid.
`hex_id`	Column. Identifies hexagon IDs (e.g., default `hex_id`).

Value

Spatial data frame of sites as points.

Examples

```
# Get sites by exact within-hex distances
sites_sp <- create_sites(psu_hexagons, spacing = 5)

# Get sites by approximate number of points (but equal spacing among hexes)
sites_n <- create_sites(psu_hexagons, n = 61)

# Same number of sites, but in slightly different spots, because creating by
# n maximizes spacing, but creating by spacing using the exact spacing
# specified.

library(ggplot2)
ggplot() +
  geom_sf(data = psu_hexagons) +
  geom_sf(data = sites_sp, size = 0.5, colour = "red") +
  geom_sf(data = sites_n, size = 0.5, colour = "blue")
```

`downweight_selection_pr` *Adjust selection weighting*

Description

Adjust selection weighting

Usage

```
downweight_selection_pr(
  sample_locs,
  scalingFactor,
  sigma_value,
  selection_column = NULL,
  fun = "cauchy",
  existing_sampling = NULL,
  dmat = NULL
)
```

Arguments

`sample_locs`	sf object with the selection weighting column (polygons or points)
`scalingFactor`	scaling factor to downweight
`sigma_value`	sigma value of distribution effect for existing sampling. Larger value means sampling has wider effect
`selection_column`	Column with sampling weights to be adjusted. If null return only the weights.
`fun`	Type of decay function. Current options are 'cauchy', 'normal' or 'exp'
`existing_sampling`	exisint sampling to down weight around (points)
`dmat`	distance matrix from sample locations (rows) to existing sampling (columns)

Value

data.frame sample_locs with downweights and adjusted selection probabilities included. If selection column is null return a vector of the downweights alone.

Examples

```
# downweight_selection_pr(BASSr::all_study_areas[1:10,], BASSr::all_study_areas[20:30,], scalingFactor = 0.1, si
```

`draw_random_samples` *Draw random sample*

Description

Draw random sample

Usage

```
draw_random_samples(
  land_hex,
  num_runs,
  n_samples,
  use_grts = TRUE,
  crs = 4326,
  coords = c("lon", "lat"),
  return_grts = FALSE,
  seed = NULL,
  quiet = FALSE,
  ...
)
```

Arguments

`land_hex`	(Spatial) Data frame. Land Cover data by hexagon. If non-spatial, will be converted to spatial sf data frame using the `crs` and `coords` arguments. Requires columns identifying the Hex ID as well as the Stratum ID (see `hex_id` and `stratum_id` respectively).
`num_runs`	Numeric. Number of times to draw random samples.
`n_samples`	Numeric. Number of samples to draw in each run.
`use_grts`	Logical. Whether to use `spsurvey::grts()` or just sample randomly without spatial dispersion.
`crs`	Numeric, character, or sf/sfc. Coordinate reference system. Must be valid input to `sf::st_crs()`.
`coords`	Character vector. Names of the columns containing X and Y coordinates (default `c("lon", "lat")`).
`return_grts`	Logical. Return the `spsurvey` object(s).
`seed`	Numeric. Random seed to use for random sampling. Seed only applies to specific sampling events (does not change seed in the environment). `NULL` does not set a seed.
`quiet`	Logical. Whether to suppress progress messages.
`...`	Extra named arguments passed on to `spsurvey::grts()`.

Value

Spatial data frame of samples

Examples

```
draw_random_samples(psu_hexagons, num_runs = 1, n_samples = 10)
```

`estimate_cost_study_area` *Cost model estimate*

Description

Cost model estimate

Usage

```
estimate_cost_study_area(
  narus,
  StudyAreas,
  pr,
  sr,
  wr = 0,
  dist_base_sa,
  dist_airport_sa,
  dist2airport_base,
  AirportType,
  vars
)
```

Arguments

`narus`	The number of ARUs to deploy in the study area
`StudyAreas`	Tibble with study area information and distances
`pr`	Column with primary road buffer proportion of study area
`sr`	Column with secondary Road proportion of study area (should not include primary road area)
`wr`	Column with Winter Road proportion of study area (should not include primary or secondary road areas)
`dist_base_sa`	Column with distance between basecamp and study area
`dist_airport_sa`	Column with distance between airport and study area
`dist2airport_base`	Column with distance between airport and base camp
`AirportType`	Column with nearest airport type
`vars`	List containing parameters for cost estimation

Value

data frame

`extract_habitat_cost-deprecated` *Extract Habitat and Cost*

Description

Extract Habitat and Cost

Arguments

`number_iterations`	Number of iterations to draw samples from full GRTS
`n_samples_per_iter`	Number of samples to pull per iteration
`sample_hexes`	Sample hexagon file
`study_area_hexes`	Study area hexagon files
`id`	Id of the study area of interest
`id_col`	Column identifying study area
`hab_rast_location`	Location of raster for habitat
`shape_file_list`	named list of shapefiles used in cost calculation. The current iteration must include the following names: **primary_roads** sf polygon with primary roads buffer **secondary_roads** sf polygon with atv roads buffer **winter_roads** sf polygon with winter roads buffer **total_roads** sf polygon with all roads buffer **airport_locations** sf points with airport locations **camp_locations** sf points with camp locations
`return_all_`	Logical to return a full results or just the inclusion probabilities
`nARUs`	number of ARUs to deploy
`hexid_col`	hexagon identification column
`calc_cost`	Logical - calculate cost
`calc_hab`	Logical - calculate habitat
`write_hexes`	Logical - write hexagons
`load_hexes`	Logical load hexagons
`rds.loc`	RDS location
`sa.rast.loc`	Study area location
`quick`	Run using cpp

full_BASS_run *A full BASS run*

Description

A full BASS run

Usage

```
full_BASS_run(
  land_hex,
  num_runs,
  n_samples,
  costs = NULL,
  hex_id = hex_id,
  stratum_id = NULL,
  omit_flag = NULL,
  non_random_set = NULL,
  benefit_weight = 0.5,
  land_cover_weights = NULL,
  return_grts = FALSE,
  crs = 4326,
  coords = c("lon", "lat"),
  seed = NULL,
  quiet = FALSE,
  ...
)
```

Arguments

land_hex	(Spatial) Data frame. Land Cover data by hexagon. If non-spatial, will be converted to spatial sf data frame using the crs and coords arguments. Requires columns identifying the Hex ID as well as the Stratum ID (see hex_id and stratum_id respectively).
num_runs	Numeric. Number of times to draw random samples.
n_samples	Numeric. Number of samples to draw in each run.
costs	Data frame. Costs for each hexagon in a RawCost format.
hex_id	Column. Identifies hexagon IDs (e.g., default hex_id).
stratum_id	Column. Identifies larger area (e.g., StudyAreaID or Province).
omit_flag	Column identifying hexes to omit (e.g., water hexes).
non_random_set	Character vector. hex_ids of hexagons to include as a non randomly selected set.
benefit_weight	Numeric. Weight assigned to benefit in the selection probabilities. 0.5 is equal weighting of cost and benefits. 1.0 is zero weighting to cost. Default 0.5.

land_cover_weights
: Data frame. Proportional weights (weights column) for specific types of landcover (lc column). lc should correspond to the same landcover column names as the hex data.

return_grts
: Logical. Return the spsurvey object(s).

crs
: Numeric, character, or sf/sfc. Coordinate reference system. Must be valid input to sf::st_crs().

coords
: Character vector. Names of the columns containing X and Y coordinates (default c("lon", "lat")).

seed
: Numeric. Random seed to use for random sampling. Seed only applies to specific sampling events (does not change seed in the environment). NULL does not set a seed.

quiet
: Logical. Whether to suppress progress messages.

...
: Extra named arguments passed on to spsurvey::grts().

Value

Data frame of inclusion probabilities. Or, if return_grts = TRUE a list including the data frame of inclusion probabilities as well as the spsurvey grts sampling object.

Extra arguments

Extra named arguments for spsurvey::grts() can also be passed on via .... In particular, note that the default values for mindis (minimum distance between sites) is NULL, and maxtry (maximum attempts to try to obtain the minimum distance between sites) is 10.

Examples

```
# With example data psu_hexagons and psu_costs...

d <- full_BASS_run(
  land_hex = psu_hexagons,
  num_runs = 10,
  n_samples = 3,
  costs = psu_costs)

# Omit water hexes

d <- full_BASS_run(
  land_hex = psu_hexagons,
  num_runs = 10,
  n_samples = 3,
  costs = psu_costs,
  omit_flag = water)

# Keep grts objects

d <- full_BASS_run(
  land_hex = psu_hexagons,
```

```
  num_runs = 10,
  n_samples = 3,
  costs = psu_costs,
  return_grts = TRUE)

names(d)
d["inclusion_probs"]]
d["grts_output"]][[1]]

# Change spsurvey::grts() arguments

d <- full_BASS_run(
  land_hex = psu_hexagons,
  num_runs = 10,
  n_samples = 3,
  costs = psu_costs,
  mindis = 10, maxtry = 10)

d
```

`genraster-deprecated` *Generate Raster*

Description

Generate Raster

Arguments

`id_col`	Column to use
`studyareas`	Study area list
`id`	Id of study area to extract
`hab_rast_location`	Location of habitat file
`writeout`	Logical - write output
`outpath`	Location of output

`getresults_BASS-deprecated`
Get the results from a BASS grts run

Description

Get the results from a BASS grts run

Arguments

grts_output	Hypothetical sample set
study_area_results	Study area results of BASS
nARUs	Number of ARUs to deploy

---

lcc2015_codes *2015 National Landcover Classification Table*

Description

Key for land cover codes and names

Usage

```
lcc2015_codes
```

Format

A data frame with 19 rows and 2 variables:

LCC_CODE Land cover code

LCC_NAME Land cover name ...

Source

https://open.canada.ca/data/en/dataset/4e615eae-b90c-420b-adee-2ca35896caf6

---

noGRTS_BASS_run-deprecated *A calculate BASS from random samples*

Description

A calculate BASS from random samples

Arguments

samples	Hypothetical sample set
num_runs	The number of times to draw random samples from hexagons
n_samples	The number of samples to draw in each sample
costs	the cost table for each hexagon id

ontario *Polygon SF of Ontario*

Description

A poloygon with the outline of Ontario

Usage

```
ontario
```

Format

A simple feature data frame with 1 rows and 2 variables:

PROV Two letter code for Ontario

NAME Name of Province

geometry SF geometry

Source

Map of Canadian Jurisdictions

oppositeSigns *Opposite signs True or false*

Description

Opposite signs True or false

Usage

```
oppositeSigns(x, y)
```

Arguments

x	a double value
y	an integer

prepare_cost *Propare hexagons for cost calculations*

Description

Propare hexagons for cost calculations

Usage

```
prepare_cost(
  truck_roads,
  atv_roads,
  winter_roads,
  all_roads,
  airports,
  basecamps,
  hexagons,
  idcol_,
  calc_roads = T,
  airport_cols = c("NAME", "AIRPORT_TY", "OGF_ID"),
  basecamp_cols = c("OFFICIAL_N", "OGF_ID", "CLASS_SUBT"),
  ...
)
```

Arguments

truck_roads	Primary roads (truck roads) buffer. Should be a polygon layer. Only used if calculating road estimates.
atv_roads	Secondary (atv roads) buffer. Should be a polygon layer. Only used if calculating road estimates.
winter_roads	Winter roads buffer. Should be a polygon layer. Not currently in use.
all_roads	Total roads buffer. Should be polygon. Not currently in use.
airports	Airport locations. Should be a polygon layer.
basecamps	Basecamp locations. Should be a polygon layer.
hexagons	Hexagon layer
idcol_	Column with hexagon ids
calc_roads	Logical. Should you calculate roads or are they already included in hexagon layer
airport_cols	Columns to use extract airport info. See examples. Should be length of 3.
basecamp_cols	Columns to use extract basecamp info. See examples. Should be length of 3.
...	You can include multisession with the furrr package. Needs to include Multicor=T & Cores = int

Value

data frame

Examples

```
## Not run:
prepare_cost(
 truck_roads = NA, atv_roads = NA, winter_roads = NA, all_roads = NA, airports = airports_official, basecamps = tour
  sf::left_join(road_info, by = c("StudyAreaID" = "StudyArea")), idcol_ = StudyAreaID, calc_roads = F, airport_col
 basecamp_cols = c("OFFICIAL_N", "OGF_ID", "CLASS_SUBT")
)

## End(Not run)
```

---

psu_costs *Dummy costs data*

Description

Dummy costs data

Usage

```
psu_costs
```

Format

A data frame with 33 rows and 27 columns

hex_id ID of the hex

province Province code for that hex

water Whether that hex is in water or not

area Area of the hex in m2

pr - total_heli_cost Specific costs for each hex (see ?`estimate_cost_study_area`)

narus Number of ARUs to be deployed

RawCost Total raw cost of sampling this hex

Source

Data generated in data-raw/data_create_study_area.R

psu_hex_dirty *Dummy hex data to be cleaned*

Description

Dummy hex data to be cleaned

Usage

```
psu_hex_dirty
```

Format

A spatial data frame with 33 features and 9 fields

hex_id ID of the hex

province Province code for that hex

water Whether that hex is in water or not

CLC... Land cover columns

x Geometry

Source

Data generated in data-raw/data_create_study_area.R

psu_hexagons *Dummy hex data*

Description

Dummy hex data

Usage

```
psu_hexagons
```

Format

A spatial data frame with 33 features and 9 fields

hex_id ID of the hex

province Province code for that hex

water Whether that hex is in water or not

LC... Land cover columns

x Geometry

Source

Data generated in data-raw/data_create_study_area.R

`psu_samples` *Dummy sampled hexes*

Description

Dummy sampled hexes

Usage

```
psu_samples
```

Format

A spatial data frame with 30 features and 21 fields

siteID-caty Sampling output (see `spsurvey::grts()`)

hex_id ID of the hex

province Province code for that hex

water Whether that hex is in water or not

LC... Land cover columns

x Geometry

run Run number

num_runs Total number of runs performed

n_samples Total number of samples drawn in a run

Source

Data generated in data-raw/data_create_study_area.R

`run_full_BASS_w_selection-defunct` *Full BASSr run with sample selection*

Description

Running this function will run BASS on both study areas and sample units within those study areas. It will return a selection of sample units with associated costs and habitats.

Arguments

`study_areas_hab_cost`	A list of study area hexagons, costs, and habitat characteristics. See naming above.
`Number_of_Study_areas`	Number of Study Areas to select
`Number_of_sample_units`	Number of Sample Units to select
`Size_of_HSS`	Size of they Hypothetical Sample Set
`Number_of_HSS`	Number of iterations in the hypothetical sample set
`Weight_of_benfit`	Weight of benefit in selection probabilties.
`LandCoverType`	String with identifer for both the land cover hexagons and their code within that tibble
`RemovedLayers_`	Layers to remove from benefit calculation must be in format of c(-var1, -var2, -var3)
`Area_of_interest`	Area of interest in tibble with StudyAreaID
`RandomSeed`	Random seet to use in GRTS
`calculate_benefits`	Should you calculate benefits or are they included in benefit_df
`only_calculate_benefits`	Do you only want to calculate benefits or complete the full run
`benefit_dfs`	If calculate_benefits is TRUE need some data.frames with benefits.
`returnALL`	do you return the GRTS object, as well as the selection probs. (Can be used later to calculate spatial balance)
`oversample`	Proportion of sites to oversample, used for both study areas and sample units
`weighted_benefits_df`	list of data frames with weight and land covers for benefit calculation
`Non`	random set of study areas and sample units in form of named list.

Value

List of sample units and a summary comparing land cover to the local area

`run_grts_on_BASS` *Run grts sampling on BASSr results*

Description

Sample sites based on the cost/benefit probabilities calculated in previous steps. Sites can be sampled with or without stratification.

Usage

```
run_grts_on_BASS(
  probs,
  nARUs,
  os = NULL,
  num_runs = 1,
  hex_id = NULL,
  stratum_id = NULL,
  remove_hexes = NULL,
  selection_weighting = inclpr,
  seed = NULL,
  ...
)
```

Arguments

`probs`	Data frame. Output of `calculate_inclusion_probs()` or `full_BASS_run()`.
`nARUs`	Numeric, Data frame, Vector, or List. Number of base samples to choose. For stratification, a named vector/list of samples per stratum, or a data frame with columns `n` for samples, `n_os` for oversamples and the column matching `stratum_id`.
`os`	Numeric, Vector, or List. Over sample size (proportional) or named vector/list of number of samples per stratum. Ignored if `nARUs` is a data frame.
`num_runs`	Numeric. Number of times to draw random samples.
`hex_id`	Column. Identifies hexagon IDs (e.g., default `hex_id`).
`stratum_id`	Column. Identifies larger area (e.g., `StudyAreaID` or `Province`).
`remove_hexes`	Character Vector. Ids of hexagons to remove prior to sampling.
`selection_weighting`	Column. Identifies selection weightings used by the `aux_var` argument in `spsurvey::grts()`. Default is `inclpr`.
`seed`	Numeric. Random seed to use for random sampling. Seed only applies to specific sampling events (does not change seed in the environment). `NULL` does not set a seed.
`...`	Extra named arguments passed on to `spsurvey::grts()`.

Value

If num_runs is 1, a single spsurvey object, otherwise a list of spsurvey objects.

Extra arguments

Extra named arguments for spsurvey::grts() can also be passed on via .... In particular, note that the default values for mindis (minimum distance between sites) is NULL, and maxtry (maximum attempts to try to obtain the minimum distance between sites) is 10.

Examples

```
d <- full_BASS_run(
  land_hex = psu_hexagons,
  num_runs = 10,
  n_samples = 3,
  costs = psu_costs)

# Simple selection
sel <- run_grts_on_BASS(
  probs = d,
  nARUs = 5,
  os = 0.2)

# Stratify
d <- dplyr::mutate(d, Province = c(rep("ON", 16), rep("MB", 17))) # Add Strata

# With lists...
sel <- run_grts_on_BASS(
  probs = d,
  nARUs = list("ON" = 5, "MB" = 2),
  stratum_id = Province,
  os = 0.2)

# With data frame...
sel <- run_grts_on_BASS(
  probs = d,
  nARUs = data.frame(Province = c("ON", "MB"),
                     n = c(5, 2),
                     n_os = c(1, 1)),
  stratum_id = Province)
```

---

select_sites *Select sites for sampling*

Description

Selection methods for processing site selection using GRTS, random sampling, clustering, or shortest path methods.

Usage

```
select_sites(
  sites,
  type,
  n_samples,
  min_dist,
  cluster_size = NULL,
  min_dist_cluster = NULL,
  os = NULL,
  hex_id = hex_id,
  site_id = site_id,
  ARUonly = FALSE,
  useGRTS = TRUE,
  progress = TRUE,
  seed = NULL
)
```

Arguments

sites	Spatial Data frame. Site points created in `create_sites()`. Requires columns identifying the Hex ID as well as the Site ID (see `hex_id` and `site_id` respectively).
type	String. Method to select sites. Must be one of • "cluster" - Clustered sampling. Sample a single point, then `cluster_size` samples around that point. • "path" - Shortest Path sampling. Sample a single point, then `cluster_size` samples in a path from that point. • "Random" - Random sampling. Sample a random set of points.
n_samples	Numeric. Number of samples to draw for each hex.
min_dist	Numeric. Minimum distance between points, or if Clusters, between cluster centres.
cluster_size	Integer. For *Clusters*, number of points per cluster. For *Shortest Paths*, number of points per path. Only applies to Clusters and Paths.
min_dist_cluster	Numeric. Minimum distance between ARU samples within clusters. Only applies to Clusters.
os	Numeric. Over sample size (proportional). Only applies to Clusters and Random.
hex_id	Column. Identifies hexagon IDs (e.g., default `hex_id`).
site_id	Column. Identifies site IDs (default `site_id`).
ARUonly	Logical. Return only ARU locations. If `FALSE` Clusters return point count locations as well. Only applies to Clusters and Random sampling.
useGRTS	Logical. Should the program be run using GRTS? Only applies to Clusters or Random samples.

progress	Logical. Show progress bars if applicable.
seed	Numeric. Random seed to use for random sampling. Seed only applies to specific sampling events (does not change seed in the environment). NULL does not set a seed.

Value

- If Clustered, returns a data frame of clustered points selected from sites.
- If Random, returns a data frame of sampled points selected from sites.
- If Shortest Path, returns a list of the points on the path and the original points selected to create the path.

Examples

```
library(dplyr)
library(ggplot2)

sites <- psu_hexagons |>
    slice_sample(n = 7) |>
    create_sites(spacing = 5) |>
    mutate(scaled_benefit = 1, benefit = 0.95)

# Basic clusters
s <- select_sites(sites = sites, hex_id = hex_id, site_id = site_id,
                  type = "cluster", os = 0.75, n_samples = 7, cluster_size = 5,
                  ARUonly = FALSE, seed = 1234, useGRTS = TRUE,
                  min_dist = 25, min_dist_cluster = 9)
ggplot() +
  geom_sf(data = psu_hexagons) +                  # Hex grid
  geom_sf(data = sites, alpha = 0.4) +      # Sites on selected Hex grids
  geom_sf(data = s, aes(colour = aru)) + # Selected sites
  scale_colour_viridis_d()

# Random samples

s <- select_sites(sites = sites, hex_id = hex_id, site_id = site_id,
                  type = "random", os = 1.0, n_samples = 2,
                  ARUonly = FALSE, seed = 1234, min_dist = 10)

ggplot() +
  geom_sf(data = psu_hexagons) +                  # Hex grid
  geom_sf(data = sites, alpha = 0.4) +      # Sites on selected Hex grids
  geom_sf(data = s, aes(colour = siteuse)) + # Selected sites
  scale_colour_viridis_d()

# Shortest Path

s <- select_sites(sites = sites, hex_id = hex_id, site_id = site_id,
                  type = "path", n_samples = 8, cluster_size = 4,
                  ARUonly = FALSE, seed = 1234, useGRTS = TRUE,
                  min_dist = 10, progress = FALSE)
```

```
ggplot() +
  geom_sf(data = sites, alpha = 0.4) + # Sites on selected Hex grid
  geom_sf(data = s$routes, aes(colour = factor(route))) + # Selected sites
  scale_colour_viridis_d()
```

`speedbass` *The internal BASSr benefit algorithm*

Description

The internal BASSr benefit algorithm

Usage

```
speedbass(hex, w, sample, total, printDets = FALSE)
```

Arguments

`hex`	A vector of land cover values
`w`	A vector of weights for each land cover value
`sample`	a vector of land cover values from random sample
`total`	a vector of land cover values from total of study area
`printDets`	logical - should you print the details - messy for now.

`ssu_land_cover` *Dummy SSU land cover*

Description

Dummy SSU land cover

Usage

```
ssu_land_cover
```

Format

A data frame with 3003 rows and 10 columns

hex_id ID of the hex

ssuID ID of subsampling unit (hex)

HexArea Area of the SSU hex

LC... Land cover columns

province Province code for that hex

Source

Data generated in data-raw/data_create_study_area.R

---

ssu_points *Dummy SSU points*

Description

Dummy SSU points

Usage

```
ssu_points
```

Format

A spatial data frame with 3003 features and 3 fields

geometry Geometry

hex_id ID of the hex

ssuID ID of subsampling unit (hex)

province Province code for that hex

Source

Data generated in data-raw/data_create_study_area.R

StudyArea_hexes *BASSr data needed for example study area*

Description

A list containing the cost, landcover, and id for an example study area

Usage

```
StudyArea_hexes
```

Format

A list of 2 dataframes and one character string:

cost cost estimates for each sample unit

landcover SF polygon with land cover in percentages for each sample unit

study_area Unique identifier for Study Area

Source

BASSr analysis for N. Ontario

`subsample_grts_and_calc_benefit-deprecated` *Subsample GRTS and calculate benefit*

Description

Subsample GRTS and calculate benefit

Arguments

`n_samples`	Number of Samples
`num_runs`	Number of iterations
`grts_file`	grts file to run
`land_hex`	Att frame
`quick`	run it using CPP quick calc

`sumC` *sum of vector*

Description

sum of vector

Usage

```
sumC(x)
```

Arguments

`x`	A vector

sumH *Add a number to a sum of vector*

Description

Add a number to a sum of vector

Usage

```
sumH(x, h)
```

Arguments

x	A vector
h	a nuber to add

Index