Package: fieldopt (via r-universe)

October 22, 2026

Title Area-Frame and Dual-Frame Survey Design under Real Field Cost

Version 0.1.0.9001

Description Design of agricultural and other area-frame and dual-frame (area plus list) sample surveys that go to the field, with the real cost of field work (travel between sampling units plus interviews) next to the variance of the design-based estimator. Travel matrices from coordinates or a road network; GRASP routing of the field work for one or several teams; stratified spatially balanced (local pivotal) or spatially ordered systematic selection of segments with interpenetrating replicates; points inside segments with multiplicity (expected-hits) estimation; closed, open and weighted segment estimators; Horvitz-Thompson estimation with local-mean or replicate variances; cost-aware allocation across strata, Hartley's dual-frame allocation with an overlap, and two-stage allocation with a cost function; and the cost-variance frontier of a design by simulation. The computational engine is the Rust crate 'fieldopt-core'. The methods follow Grafström, Lundström and Schelin (2012) <doi:10.1111/j.1541-0420.2011.01699.x> for the local pivotal method, Grafström and Schelin (2014) <doi:10.1111/sjos.12016> for the local-mean variance, Hartley (1962) for the dual-frame design, and Beardwood, Halton and Hammersley (1959) <doi:10.1017/S0305004100034095> for the square-root law of travel behind the cost curve.

License GPL (>= 3)

URL https://castlaboratory.github.io/fieldopt/, https://github.com/castlaboratory/fieldopt

BugReports https://github.com/castlaboratory/fieldopt/issues

Depends R (>= 4.2)

Imports cli, generics, ggplot2, rlang, stats, tibble

Suggests arrow, BalancedSampling, dplyr, highs, knitr, osrm, purrr, Matrix, rmarkdown, sf, survey, testthat (>= 3.0.0), tidyr, vrpr

VignetteBuilder knitr

Config/Needs/website rmarkdown

Config/rextendr/version 0.5.0

Config/testthat/edition 3

Encoding UTF-8

Language en-GB

Roxygen list(markdown = TRUE)

RoxygenNote 8.0.0

SystemRequirements Cargo (Rust's package manager), rustc >= 1.70.0, xz

Config/pak/sysreqs xz-utils libclang-dev

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

Date/Publication 2026-10-22 20:14:24 UTC

RemoteUrl https://github.com/castlaboratory/fieldopt

RemoteRef HEAD

RemoteSha 115376dac3d6d0f2ddbcc65deac40444925eff5c

Contents

as_frame . . . 3
as_sf . . . 4
as_svydesign . . . 4
autoplot.fieldopt_frontier . . . 5
autoplot.fieldopt_routes . . . 6
autoplot.fieldopt_sample . . . 7
autoplot.fieldopt_schedule . . . 7
core_version . . . 8
cost_variance_frontier . . . 8
design_variance . . . 10
dual_frame_allocation . . . 11
dual_frame_design . . . 13
dual_frame_estimator . . . 15
field_cost_model . . . 16
frame_allocation . . . 17
frame_overlap . . . 18
multivariate_allocation . . . 19
point_design_effect . . . 20
point_estimator . . . 22
point_expected_hits . . . 23
ratio_estimator . . . 23
read_areaframe . . . 24
route_fieldwork . . . 25
routed_cost_curve . . . 27
routed_unit_cost . . . 29
routes-methods . . . 31
sampled . . . 32

schedule_calendar 32
schedule_fieldwork 34
segment_estimator 35
select_points 36
select_units 37
spatial_balance 39
tidy.fieldopt_multi_allocation 40
travel_matrix 41
two_stage_allocation 43
two_stage_design 44

Index **47**

`as_frame` *Turn a spatial object into a frame*

Description

Converts an `sf` object (points or polygons) into the plain data frame that `travel_matrix()` and `select_units()` expect: a column `unit`, the coordinates `lat`/`lon` (centroids for polygons, in WGS 84) and, for polygons, their `area` in square kilometres, next to the other columns of the object. Nothing spatial is kept: use `as_sf()` to put the results back on the geometries.

Usage

```
as_frame(x, unit = NULL)
```

Arguments

`x`	An `sf` object (the `sf` package must be installed), or a data frame returned unchanged.
`unit`	Name of the column with the unit names; made from the row numbers when `NULL` and no `unit` column exists.

Value

A tibble with `unit`, `lat`, `lon`, `area` (polygons only) and the attribute columns.

Examples

```
if (requireNamespace("sf", quietly = TRUE)) {
 sq <- sf::st_sfc(sf::st_polygon(list(rbind(c(0, 0), c(0, 1), c(1, 1), c(1, 0), c(0, 0)))),
    crs = 4326
  )
  cells <- sf::st_sf(crop = 3, geometry = sf::st_make_grid(sq, n = 3))
  as_frame(cells)
}
```

as_sf — *Put results back on the geometries of a spatial frame*

Description

Joins a sample, a frontier allocation or any data frame with a `unit` column to the geometries of the `sf` object the frame came from, for maps.

Usage

```
as_sf(x, sf, unit = NULL)
```

Arguments

`x`	A data frame with a `unit` column (a `fieldopt_sample`, for example).
`sf`	The `sf` object that `as_frame()` was applied to.
`unit`	Name of the column with the unit names in `sf`, as in `as_frame()`.

Value

An `sf` object with the columns of `x`.

Examples

```
if (requireNamespace("sf", quietly = TRUE)) {
 sq <- sf::st_sfc(sf::st_polygon(list(rbind(c(0, 0), c(0, 1), c(1, 1), c(1, 0), c(0, 0)))),
    crs = 4326
  )
  cells <- sf::st_sf(geometry = sf::st_make_grid(sq, n = 4))
  s <- select_units(as_frame(cells), n = 4, seed = 1)
  as_sf(s, cells)
}
```

as_svydesign — *Hand a sample to the survey package*

Description

Builds a `survey` design object from a `select_units()` sample so that the estimators of that package (calibration, post-stratification, domains, ratios, regression) can be used on the field data. Units not sampled are dropped; the weights are the inverse inclusion probabilities, and the strata of the sample are the strata of the design.

Usage

```
as_svydesign(sample, variance = c("brewer", "srs"))
```

Arguments

sample	A select_units() sample.
variance	"brewer" or "srs" (ignored for replicated samples).

Details

- Samples with interpenetrating replicates (method = "systematic", replicates > 1) become a replicate-weight design (survey::svrepdesign()) whose variance is the variance between the replicate estimates, the one design_variance() uses.
- Other samples become a survey::svydesign(): with variance = "brewer" (default) the without-replacement probability-proportional-to- size approximation of Brewer, which needs no joint probabilities; with variance = "srs" the finite-population correction of simple random sampling within strata. Neither uses the spatial balance of the local pivotal method, so for the variance of a total of a spatially balanced sample prefer design_variance(); use the survey object for what it adds.

Value

A survey.design2 or svyrep.design object.

Examples

```
if (requireNamespace("survey", quietly = TRUE)) {
  cells <- expand.grid(x = 1:10, y = 1:10)
  cells$unit <- paste0("c", 1:100)
  cells$crop <- 10 + cells$x + rnorm(100)
  s <- select_units(cells, n = 20, seed = 1)
  d <- as_svydesign(s)
  survey::svytotal(~crop, d)
  design_variance(s, "crop")[, c("total", "se")]
}
```

---

autoplot.fieldopt_frontier

Plot the cost-variance frontier

Description

Plot the cost-variance frontier

Usage

```
## S3 method for class 'fieldopt_frontier'
autoplot(object, ...)
```

Arguments

`object`	A `fieldopt_frontier`.
`...`	Unused.

Value

A ggplot of mean cost against mean variance (or against `n` when no study variable was given), labelled by sample size.

`autoplot.fieldopt_routes` *Plot the routes*

Description

Plot the routes

Usage

```
## S3 method for class 'fieldopt_routes'
autoplot(object, ...)
```

Arguments

`object`	A `fieldopt_routes`.
`...`	Unused.

Value

A ggplot with the units, the depot and the routes in the coordinate space of the travel matrix.

Examples

```
set.seed(1)
pts <- data.frame(
  unit = c("depot", paste0("s", 1:12)),
  x = c(0, runif(12, 0, 10)), y = c(0, runif(12, 0, 10))
)
r <- route_fieldwork(travel_matrix(pts, method = "euclidean"), paste0("s", 1:12),
  depot = "depot", max_stops = 5, iterations = 50
)
autoplot(r)
```

autoplot.fieldopt_sample

Plot a sample of units

Description

Plot a sample of units

Usage

```
## S3 method for class 'fieldopt_sample'
autoplot(object, ...)
```

Arguments

object	A fieldopt_sample.
...	Unused.

Value

A ggplot of the frame with the sampled units highlighted, coloured by stratum when the sample is stratified, with longitude on the x axis for geographic coordinates.

Examples

```
cells <- expand.grid(x = 1:12, y = 1:12)
cells$h <- ifelse(cells$x <= 6, "west", "east")
autoplot(select_units(cells, n = 20, strata = "h", seed = 3))
```

autoplot.fieldopt_schedule

Plot a field schedule

Description

Plot a field schedule

Usage

```
## S3 method for class 'fieldopt_schedule'
autoplot(object, ...)
```

Arguments

object	A fieldopt_schedule.
...	Unused.

Value

A ggplot of the routes of every base, coloured by team, with the bases marked.

---

core_version *Version of the Rust engine*

Description

Version of the Rust engine

Usage

```
core_version()
```

Value

The version string of the `fieldopt-core` crate compiled into the package.

Examples

```
core_version()
```

---

cost_variance_frontier *Cost-variance frontier of a design by simulation*

Description

For each sample size in `n_grid`, draws `n_rep` spatially balanced samples from the frame, routes the field work from the depot, prices it with the cost model and, when a study variable is given, estimates the variance of the total. The result is the trade-off a survey planner faces: how much the field work costs and how precise the estimate is at each sample size.

Usage

```
cost_variance_frontier(
  frame,
  depot,
  cost_model,
  n_grid,
  y = NULL,
  size = NULL,
  strata = NULL,
  selection = c("lpm", "systematic", "srs"),
  replicates = 1,
```

```
  matrix = NULL,
  method = c("haversine", "euclidean", "osrm"),
  max_length = Inf,
  max_stops = Inf,
  n_rep = 20,
  iterations = 100,
  seed = 1
)
```

Arguments

frame	Data frame of units as in select_units(), including the depot row named in depot.
depot	Name of the depot unit in frame$unit.
cost_model	A field_cost_model().
n_grid	Sample sizes to evaluate: a numeric vector of total sizes (allocated across strata as in select_units()), or a list of vectors named by stratum, one allocation per grid point.
y	Optional column of the study variable for the whole frame (a planning value); when given, the variance of the estimated total is computed with design_variance() on each simulated sample.
size	Optional size column for probability-proportional-to-size selection.
strata	Optional stratum column, passed to select_units().
selection	Selection method of select_units(): "lpm", "systematic" or "srs".
replicates	Replicates of a "systematic" selection.
matrix	Optional travel_matrix() of the frame (computed from the coordinates otherwise).
method	Distance method of travel_matrix() when matrix is NULL ("osrm" queries the road network once for the whole frame).
max_length, max_stops	Route limits passed to route_fieldwork().
n_rep	Replications per sample size.
iterations	GRASP iterations per routing.
seed	Seed.

Value

A tibble of class fieldopt_frontier with one row per sample size: n, cost_mean, cost_sd, cost_per_unit (the planning value for c1 in two_stage_allocation() or cost_a in dual_frame_allocation()), travel_mean, routes_mean, variance_mean and cv_mean (when y is given), n_rep, and allocation (a list column with the stratum sizes) when strata are used.

Examples

```
set.seed(3)
frame <- data.frame(
  unit = c("depot", paste0("s", 1:40)),
  x = c(5, runif(40, 0, 10)), y = c(5, runif(40, 0, 10))
)
frame$crop <- c(NA, 20 + 3 * frame$x[-1] + rnorm(40))
model <- field_cost_model(
  per_travel = 2, per_unit = 30, per_interview = 10,
  interviews_per_unit = 4
)
cost_variance_frontier(frame,
  depot = "depot", cost_model = model, n_grid = c(8, 16),
  y = "crop", method = "euclidean", n_rep = 3, iterations = 20
)
```

---

design_variance *Design-based estimate of a total with its variance*

Description

Horvitz-Thompson estimate of the population total of `y` from a `select_units()` sample, with the variance estimator that matches the design: the local-mean estimator of Grafström and Schelin (2014) for spatially balanced samples (it needs no joint inclusion probabilities), the variance among replicate estimates for replicated systematic samples, and the simple-random-sampling formula for equal-probability simple random samples. A systematic sample drawn as a single replicate has no unbiased design-based variance estimator; the local-mean estimator is used as the customary approximation. The simple-random-sampling variance of the same sample is always reported as a reference.

Usage

```
design_variance(sample, y, neighbours = 3L)
```

Arguments

`sample`	A `fieldopt_sample`.
`y`	Column of the study variable (observed on the sampled units; other rows may be `NA`).
`neighbours`	Number of nearest sampled units in the neighbourhood of the local-mean estimator.

Details

The local-mean estimator averages over each sampled unit and its `neighbours` nearest sampled units (ties included), as `BalancedSampling::vsb()` does with its default `k = 3`. In the package's experiments (three frames, Gaussian fields at four spatial ranges) it is close to unbiased when the

variable has no spatial structure and conservative when it is smooth; the replicate estimator is unbiased but has only -1 degrees of freedom, so intervals should use the t quantile with `df` (reported) rather than the normal one.

Value

A one-row tibble: `total`, `variance`, `se`, `cv`, `variance_srs`, `n`, `N`, `variance_method` and `df` (degrees of freedom of the variance estimate: -1 with `r` replicates, the number of sampled units minus the number of strata otherwise).

References

Grafström, A. and Schelin, L. (2014). How to select representative samples. *Scandinavian Journal of Statistics*, 41(2), 277–290.

Examples

```
set.seed(2)
frame <- data.frame(x = runif(80), y = runif(80))
frame$crop <- 10 + 20 * frame$x + rnorm(80)
s <- select_units(frame, n = 20)
design_variance(s, y = "crop")
r <- select_units(frame, n = 20, method = "systematic", replicates = 4)
design_variance(r, y = "crop")
```

`dual_frame_allocation` *Dual-frame allocation (area frame plus list frame with overlap)*

Description

Hartley's (1962, 1974) design for two frames whose union covers the population: frame A (typically the area frame, complete but expensive) and frame B (the list, cheap but incomplete). The population splits into the domains `a` (A only), `ab` (both) and `b` (B only), and the total is estimated by `Y_a + theta * Y_ab(A) + (1 - theta) * Y_ab(B) + Y_b`: the overlap is estimated from both samples and mixed with weight `theta`. With simple random sampling in each frame (other designs enter through design effects) the variance is a sum of two frame terms, so for each `theta` the cost-optimal `n_A`, `n_B` follow the cost-weighted Neyman rule; `theta` is then optimised. `theta = 0` is the **screening** design, in which overlap units found in the area sample are not used (the list covers them), the usual practice when the list holds the large producers.

Usage

```
dual_frame_allocation(
  domains,
  cost_a,
  cost_b,
  deff_a = 1,
  deff_b = 1,
```

```
  theta = NULL,
  target_variance = NULL,
  target_cv = NULL,
  budget = NULL
)
```

Arguments

- `domains` — Data frame with one row per domain `a`, `ab`, `b` (column `domain`) and columns `size` (units), `mean` and `sd` (standard deviation with divisor `size - 1`) of the study variable.
- `cost_a`, `cost_b` — Cost per sampled unit in frame A and frame B.
- `deff_a`, `deff_b` — Design effects of the two samples relative to simple random sampling (an area sample of segments or points usually has `deff_a > 1`).
- `theta` — `NULL` to optimise, a number in `[0, 1]`, or `"screening"` (zero).
- `target_variance`, `target_cv`, `budget` — Exactly one: target variance of the estimated total, target coefficient of variation (relative to the population total implied by `domains`), or budget.

Details

The cost per area unit should include the field work: take it from the routing, for example `cost_mean / n` of a `cost_variance_frontier()` at the sample size under consideration, and iterate once or twice.

Value

A list of class `fieldopt_dual_frame`: `n_a`, `n_b`, `theta`, `cost`, `variance`, `se`, `cv`, `total` (implied population total), `variance_a`, `variance_b`, `overlap_a`, `overlap_b` (expected overlap units in each sample), `bounded`, and the inputs.

References

Hartley, H. O. (1962). Multiple frame surveys. *Proceedings of the Social Statistics Section, ASA*, 203–206. Hartley, H. O. (1974). Multiple frame methodology and selected applications. *Sankhya C*, 36, 99–118. Lohr, S. L. and Rao, J. N. K. (2006). Estimation in multiple-frame surveys. *JASA*, 101(475), 1019–1030.

Examples

```
domains <- data.frame(
  domain = c("a", "ab", "b"), size = c(8000, 1500, 500),
  mean = c(5, 40, 60), sd = c(6, 30, 50)
)
dual_frame_allocation(domains, cost_a = 200, cost_b = 40, budget = 1e5)
dual_frame_allocation(domains, cost_a = 200, cost_b = 40, theta = "screening", target_cv = 0.05)
```

dual_frame_design *Dual-frame allocation with the area-frame cost taken from the routing*

Description

dual_frame_allocation() needs the cost per unit of the area frame, which is not a constant: the routed cost G(n_a) of visiting n_a area units grows like c0 + a n_a + b sqrt(n_a). As two_stage_design(), this function measures that curve by routing samples at a few sizes, fits it, and re-allocates with the **marginal** cost G'(n_a) as cost_a, which sets the split between the frames and theta; under a budget the sample sizes then come from the fitted curve, keeping that split, so the design spends the budget at its real routed cost. The cost model prices the area units (visits, interviews and travel); cost_b is the list cost per unit.

Usage

```
dual_frame_design(
  frame,
  depot,
  cost_model,
  domains,
  cost_b,
  deff_a = 1,
  deff_b = 1,
  theta = NULL,
  target_variance = NULL,
  target_cv = NULL,
  budget = NULL,
  interviews_per_unit = NULL,
  cost_a_start = NULL,
  max_iter = 8,
  size = NULL,
  strata = NULL,
  selection = c("lpm", "systematic", "srs"),
  matrix = NULL,
  method = c("haversine", "euclidean", "osrm"),
  max_length = Inf,
  max_stops = Inf,
  n_rep = 5,
  iterations = 100,
  seed = 1
)
```

Arguments

frame	Data frame of units as in select_units(), including the depot row named in depot.
depot	Name of the depot unit in frame$unit.

cost_model — A field_cost_model().

domains — Data frame with one row per domain a, ab, b (column domain) and columns size (units), mean and sd (standard deviation with divisor size - 1) of the study variable.

cost_b — Cost per sampled unit of the list frame.

deff_a, deff_b — Design effects of the two samples relative to simple random sampling (an area sample of segments or points usually has deff_a > 1).

theta — NULL to optimise, a number in [0, 1], or "screening" (zero).

target_variance, target_cv, budget — Exactly one: target variance of the estimated total, target coefficient of variation (relative to the population total implied by domains), or budget.

interviews_per_unit — Expected interviews per visited area unit (for the cost model; the model's own value is used when NULL).

cost_a_start — Starting value of the cost per area unit.

max_iter — Maximum number of rounds.

size — Optional size column for probability-proportional-to-size selection.

strata — Optional stratum column, passed to select_units().

selection — Selection method of select_units(): "lpm", "systematic" or "srs".

matrix — Optional travel_matrix() of the frame (computed from the coordinates otherwise).

method — Distance method of travel_matrix() when matrix is NULL ("osrm" queries the road network once for the whole frame).

max_length, max_stops — Route limits passed to route_fieldwork().

n_rep — Routed samples per sample size of the cost curve.

iterations — GRASP iterations per routing.

seed — Seed.

Value

The final fieldopt_dual_frame allocation, with cost the routed cost G(n_a) + cost_b n_b, cost_a the marginal cost used, cost_a_average (G(n_a)/n_a), cost_curve, cost_coef, history (one row per round: round, cost_a, cost_a_average, n_a, n_b, theta, cost, variance) and converged.

Examples

```
set.seed(3)
cells <- expand.grid(x = 1:12, y = 1:12)
cells$unit <- paste0("c", seq_len(nrow(cells)))
frame <- rbind(data.frame(unit = "depot", x = 6, y = 6), cells)
model <- field_cost_model(
  per_travel = 3, per_unit = 40, per_interview = 20,
  interviews_per_unit = 3
```

```
)
domains <- data.frame(
  domain = c("a", "ab", "b"), size = c(400, 60, 20),
  mean = c(6, 30, 45), sd = c(5, 20, 35)
)
dual_frame_design(frame, "depot", model, domains,
  cost_b = 45, deff_a = 1.5,
  target_cv = 0.05, method = "euclidean", n_rep = 1, iterations = 10
)
```

`dual_frame_estimator` *Dual-frame estimate of a total*

Description

Combines the sample of frame A (the area frame) and the sample of frame B (the list) into one estimate of the population total of `y`. Each sampled unit carries its domain: `"a"` or `"ab"` in the A sample, `"b"` or `"ab"` in the B sample. Two estimators:

Usage

```
dual_frame_estimator(
  sample_a,
  sample_b,
  y,
  domain = "domain",
  theta = NULL,
  estimator = c("hartley", "fuller-burmeister")
)
```

Arguments

`sample_a`, `sample_b`	`select_units()` samples of frame A and frame B with columns `y` and `domain` (unsampled rows may hold `NA`).
`y`	Column of the study variable.
`domain`	Column with the domain of each unit.
`theta`	Mixing weight of the overlap for Hartley: a number in `[0, 1]`, `"screening"`, or `NULL` to estimate it.
`estimator`	`"hartley"` or `"fuller-burmeister"`.

Details

- **Hartley (1962)**: `Y_a + theta * Y_ab(A) + (1 - theta) * Y_ab(B) + Y_b`, with `theta` fixed, `"screening"` (`theta = 0`: overlap units found in the area sample are dropped, as when the list is enumerated completely) or estimated as the value that minimises the variance;

- **Fuller and Burmeister (1972)**: adds a second term in the difference between the two estimates of the overlap size, with coefficients chosen to minimise the variance.

Variances and covariances of the domain totals come from the variance estimator of each sample's design (`design_variance()`), the two samples being independent. The estimated `theta` and the Fuller-Burmeister coefficients make the variances slightly optimistic in small samples.

Value

A one-row tibble: `total`, `variance`, `se`, `cv`, `estimator`, `theta` (Hartley) or `beta_1`, `beta_2` (Fuller-Burmeister), the domain totals `y_a`, `y_ab_a`, `y_ab_b`, `y_b`, and `n_a`, `n_b`.

References

Hartley, H. O. (1962). Multiple frame surveys. *Proceedings of the Social Statistics Section, ASA*, 203-206. Fuller, W. A. and Burmeister, L. F. (1972). Estimators for samples selected from two overlapping frames. *Proceedings of the Social Statistics Section, ASA*, 245-249. Lohr, S. L. (2009). Multiple-frame surveys. In *Handbook of Statistics 29A*, 71-88.

Examples

```
set.seed(7)
cells <- expand.grid(x = 1:12, y = 1:12)
cells$unit <- paste0("c", 1:144)
cells$domain <- ifelse(runif(144) < 0.2, "ab", "a")
cells$y <- ifelse(cells$domain == "ab", 80, 10) + rnorm(144, sd = 3)
# the list holds the overlap units (same values) plus units of its own
ab <- cells[cells$domain == "ab", ]
list <- data.frame(
  unit = c(paste0("l", seq_len(nrow(ab))), paste0("b", 1:11)),
  x = runif(nrow(ab) + 11), y = c(ab$y, 120 + rnorm(11, sd = 5)),
  domain = rep(c("ab", "b"), c(nrow(ab), 11))
)
list$lon <- runif(nrow(list))
list$lat <- runif(nrow(list))
sa <- select_units(cells, n = 30, seed = 1)
sb <- select_units(list, n = 15, coords = c("lat", "lon"), method = "srs", seed = 2)
dual_frame_estimator(sa, sb, y = "y", domain = "domain")
dual_frame_estimator(sa, sb, y = "y", domain = "domain", theta = "screening")
dual_frame_estimator(sa, sb, y = "y", domain = "domain", estimator = "fuller-burmeister")
```

`field_cost_model` *Field cost model*

Description

Converts a routing solution and a sample into money (or time): a cost per unit of travel, a fixed cost per route (a team-day: allowances, vehicle, lodging), a fixed cost per visited unit (access, setup) and a cost per interview.

Usage

```
field_cost_model(
  per_travel = 1,
  per_unit = 0,
  per_interview = 0,
  interviews_per_unit = 1,
  per_route = 0,
  currency = "cost"
)
```

Arguments

`per_travel`	Cost per unit of the travel matrix (per km, per minute).
`per_unit`	Fixed cost of visiting one unit.
`per_interview`	Cost of one interview.
`interviews_per_unit`	Expected interviews per visited unit: a single number, a vector with one value per row of the travel matrix (depot included, in the same order), or a function of the row indices of the visited units.
`per_route`	Fixed cost of one route (one team for one day).
`currency`	Label for reports.

Value

An object of class `field_cost_model`.

Examples

```
field_cost_model(per_travel = 1.2, per_unit = 50, per_interview = 15, interviews_per_unit = 8)
```

`frame_allocation` *Cost-aware allocation across strata*

Description

Classical optimum allocation with a cost per unit in each stratum (Cochran 1977, Section 5.5): minimum cost for a target variance of the estimated total, or minimum variance for a budget. For two frames with an overlap use `dual_frame_allocation()` instead.

Usage

```
frame_allocation(strata, target_variance = NULL, budget = NULL)
```

Arguments

- `strata` — Data frame with columns `size` (population units), `sd` (standard deviation of the study variable) and `cost` (cost per unit, interview plus expected access), and optionally `stratum` with names.
- `target_variance` — Target variance of the estimated total (give this or `budget`).
- `budget` — Total cost allowed.

Value

The strata with a column `n`, plus attributes `cost`, `variance` and `bounded` (whether `2 <= n_h <= N_h` was active), of class `fieldopt_allocation`.

Examples

```
strata <- data.frame(
  stratum = c("list", "area"), size = c(2000, 800),
  sd = c(12, 30), cost = c(40, 180)
)
frame_allocation(strata, target_variance = 4e6)
frame_allocation(strata, budget = 20000)
```

`frame_overlap` *Overlap between a list frame and the area frame*

Description

Finds, for every establishment of a list frame, the cell of the area frame it falls in, and marks its domain: `"ab"` when it lies in a cell of the frame (it can be reached through the area sample as well), `"b"` otherwise. The cells get the count and the names of the listed establishments inside them, which is what field teams need to screen the overlap out of the area sample (the screening design) or what `dual_frame_estimator()` needs as domains.

Usage

```
frame_overlap(list, cells, cell_size = 1, coords = NULL, pairs = NULL)
```

Arguments

- `list` — Data frame of the list frame with a column `unit` and the coordinate columns (or an `sf` object of points).
- `cells` — The area frame as in `select_units()` (or an `sf` object of polygons with a `unit` column).
- `cell_size` — Side of the square cells, in the unit of the coordinates (ignored for polygons).
- `coords` — Names of the coordinate columns in both tables.
- `pairs` — Optional data frame with columns `unit` (list) and `cell`.

Details

Locations are matched with coordinates: square cells of side `cell_size` around the cell centres (the columns `coords`), or exact polygons when `cells` and `list` are `sf` objects (the `sf` package is then used). A table of `pairs` from record linkage (for example from the `reclin2` package) can add or override matches: rows with the list unit and the cell it belongs to.

Value

A list with `list` (the list frame with columns `cell` and `domain`) and `cells` (the area frame with columns `listed`, the number of listed establishments inside, and `listed_units`, their names).

Examples

```
cells <- expand.grid(x = 1:5, y = 1:5)
cells$unit <- paste0("c", 1:25)
farms <- data.frame(unit = c("f1", "f2", "f3"), x = c(1.2, 3.4, 9), y = c(1.1, 2.6, 9))
ov <- frame_overlap(farms, cells, cell_size = 1)
ov$list
ov$cells[ov$cells$listed > 0, ]
```

---

multivariate_allocation

Cost-aware allocation for several study variables

Description

Minimum cost `sum c_h n_h` such that the variance of the estimated total of every study variable is at most its target (Bethel 1989; Chromy 1987). This is `frame_allocation()` with one constraint per variable: a survey that must estimate several crops, or a crop and the number of farms, with stated precisions. The multipliers tell which constraints bind.

Usage

```
multivariate_allocation(
  strata,
  sd,
  target_variance = NULL,
  target_cv = NULL,
  totals = NULL
)
```

Arguments

`strata`	Data frame with columns `size`, `cost`, optionally `stratum`, and one standard-deviation column per study variable.
`sd`	Names of the standard-deviation columns, one per variable.

target_variance
: Target variance of each variable's total, in the order of sd (give this or target_cv).

target_cv
: Target coefficient of variation of each total; needs totals.

totals
: Population totals of the variables, for target_cv.

Value

The strata with a column n, of class fieldopt_multi_allocation, with attributes cost, targets, attained (variance of each variable), multipliers, bounded.

References

Bethel, J. (1989). Sample allocation in multivariate surveys. *Survey Methodology*, 15, 47-57. Chromy, J. R. (1987). Design optimization with multiple objectives. *Proceedings of the Survey Research Methods Section, ASA*, 194-199.

Examples

```
strata <- data.frame(
  stratum = c("high", "mid", "low"), size = c(300, 800, 2000),
  cost = c(250, 180, 150), sd_corn = c(40, 25, 8), sd_cattle = c(15, 30, 12)
)
multivariate_allocation(strata,
  sd = c("sd_corn", "sd_cattle"),
  target_cv = c(0.05, 0.08), totals = c(90000, 60000)
)
```

---

point_design_effect *Design effect and bias of point sampling by simulation*

Description

Repeats the whole area-frame design, selecting cells with select_units(), placing points with select_points(), letting each point hit the establishment whose land it falls on (with probability equal to the establishment's share of the cell), and estimating the total with point_estimator(). Returns the empirical variance of the estimator, its bias, the mean number of interviews, and the design effect relative to a simple random sample of establishments with the same number of interviews: the deff_a that dual_frame_allocation() asks for.

Usage

```
point_design_effect(
  cells,
  areas,
  n,
  points_per_cell,
  cell_size,
  size = NULL,
```

```
  strata = NULL,
  method = c("lpm", "systematic", "srs"),
  replicates = 1,
  n_sim = 200,
  seed = 1
)
```

Arguments

cells	Data frame of cells as in select_units() (with unit, coordinates and, when used, strata and size columns).
areas	Data frame with one row per (establishment, cell) pair: establishment, unit, area (land of the establishment inside the cell, in the square units of cell_size) and y (the study variable of the whole establishment, repeated across its cells).
n	Sample size: a single number (allocated to strata in proportion to the sum of size, or to the number of units when size is NULL), or a vector named by stratum.
points_per_cell	Points per sampled cell: a single number, a vector named by stratum (for example more points where agricultural use is more intense), or the name of a column of the sample.
cell_size	Side of the square cells, in the units of the coordinates (one number, or c(dx, dy)).
size	Column of the size measure for probability-proportional-to-size selection, or NULL for equal probabilities within stratum.
strata	Column with the stratum of each unit, or NULL.
method	"lpm", "systematic", "srs" or "cube" (balanced sampling by the cube method of Deville and Tillé 2004, with the fast flight phase and landing by suppression of variables).
replicates	Number of independent interpenetrating replicates for "systematic" (each a systematic sample of about n / replicates units; replicates may share units). Ignored by the other methods.
n_sim	Number of simulated surveys.
seed	Seed.

Details

Hits are attributed by the establishments' shares of the cell area, not by the geometry of their fields, so the simulator does not distinguish the random and systematic point layouts; it captures the cell design, the point intensity and the multiplicity, which is what drives the design effect at the cell level.

Value

A one-row tibble of class fieldopt_point_deff: n_sim, total, mean_estimate, bias, variance, cv, mean_variance_estimate (the mean of the point_estimator() variance estimates, to be compared with variance), interviews, variance_srs, deff; attribute estimates holds the simulated values.

Examples

```
set.seed(4)
cells <- expand.grid(x = 1:10, y = 1:10)
cells$unit <- paste0("c", 1:100)
areas <- rbind(
  data.frame(establishment = paste0(cells$unit, "-1"), unit = cells$unit, area = 0.5),
  data.frame(establishment = paste0(cells$unit, "-2"), unit = cells$unit, area = 0.2)
)
areas$y <- round(rgamma(nrow(areas), 3, 3 / (10 * areas$area)), 1)
point_design_effect(cells, areas, n = 15, points_per_cell = 6, cell_size = 1, n_sim = 20)
```

---

point_estimator *Multiplicity (point) estimator of a total from an area sample of points*

Description

Each point that hits an establishment contributes `y / expected_hits` of that establishment; the sum over hits is unbiased for the population total whatever the number of times an establishment is hit (the more points fall on an establishment, the larger its contribution, matching its larger chance of selection). The variance is estimated at the cell level (ultimate-cluster approximation): the cell contributions are treated as cell totals and the variance estimator of the cell design is applied (local-mean for `"lpm"`, replicates for replicated systematic samples, simple random sampling otherwise). With replicated systematic samples the expected hits refer to the union of the replicates, so the replicate variance is an approximation.

Usage

```
point_estimator(hits, sample)
```

Arguments

`hits`	Data frame with one row per point that hit an establishment: columns `unit` (cell), `establishment`, `y` (value of the whole establishment) and `expected_hits` (from `point_expected_hits()`).
`sample`	The `fieldopt_sample` of cells.

Value

A one-row tibble as in `design_variance()`, with `n_hits` and `n_establishments`.

point_expected_hits *Expected number of point hits of each establishment*

Description

An establishment with agricultural area `area` inside cell `unit` is hit by a point of that cell with expected count `density * area`; summed over the cells it spans, this is the multiplicity factor that `point_estimator()` divides by. Cells outside the sample contribute `pi * points / area` as well, so the expectation is over the whole design; supply `points_per_cell` for them.

Usage

```
point_expected_hits(areas, sample, points_per_cell, cell_size)
```

Arguments

`areas`	Data frame with columns `establishment`, `unit` (cell) and `area` (agricultural area of the establishment inside that cell, in the square units of `cell_size`).
`sample`	The `fieldopt_sample` of cells.
`points_per_cell`	As in `select_points()`, applied to every cell of the frame.
`cell_size`	As in `select_points()`.

Value

A tibble with `establishment` and `expected_hits`.

ratio_estimator *Ratio estimate of a total with an auxiliary variable*

Description

The ratio estimator `X * Y_hat / X_hat`, where `X` is the known population total of the auxiliary (the area of the segments, the number of farms from a census), with the variance of the residuals `y - R x` under the design, estimated as in `design_variance()`. The gain over the Horvitz-Thompson estimate is reported.

Usage

```
ratio_estimator(sample, y, x, x_total = NULL)
```

Arguments

sample	A fieldopt_sample.
y	Column of the study variable (observed on the sampled units; other rows may be NA).
x	Column of the auxiliary variable.
x_total	Known population total of x; the frame total of the column when NULL.

Value

A one-row tibble as in design_variance() plus ratio, x_total and variance_ht (the variance of the plain Horvitz-Thompson estimate).

Examples

```
cells <- expand.grid(x = 1:12, y = 1:12)
cells$unit <- paste0("c", 1:144)
cells$farmland <- runif(144, 20, 100)
cells$crop <- 0.4 * cells$farmland + rnorm(144, sd = 3)
s <- select_units(cells, n = 24, seed = 2)
ratio_estimator(s, y = "crop", x = "farmland")
```

---

read_areaframe *Read the cells of an area frame exported by the areaframe service*

Description

Reads a frame of grid cells as exported by the area-frame service (a Parquet file with cell_index, centroid_lon, centroid_lat, area_m2 and the geometry as WKB, plus any stratum or covariate columns) or any spatial file sf can read (GeoPackage, GeoJSON, shapefile), and returns the plain frame select_units() works on: unit, lat, lon, area in square kilometres (also used as size), and the other columns. With sf = TRUE the geometries are kept and an sf object is returned instead (see as_frame() and as_sf()).

Usage

```
read_areaframe(path, sf = FALSE)
```

Arguments

path	Path to the file.
sf	Return an sf object with the geometries.

Value

A tibble, or an sf object.

Examples

```
## Not run:
cells <- read_areaframe("cells.parquet")
s <- select_units(cells, n = 300, size = "area", strata = "stratum")

## End(Not run)
```

---

route_fieldwork *Route the field work*

Description

Finds low-cost routes from a depot through the selected units: one route when the limits allow it, several otherwise (one per team or per day). Instances of up to 13 units are solved exactly (dynamic programming over subsets). Larger ones go to a hybrid genetic search (Vidal 2022): a population of giant tours, order crossover, the optimal split of Prins (2004) under the limits, and a local search with granular neighbourhoods (2-opt, Or-opt, relocate, swap, 2-opt*). The `gap` to a lower bound is reported: the Held-Karp bound for a single tour on a symmetric matrix (usually within a few percent of the optimum), a loose bound otherwise. For a single tour, a branch-and-bound on Held-Karp 1-trees then tries to close the gap within `certify` seconds; `optimal` says when the solution is proven optimal (most tours of up to about 100 units are, within a few seconds).

Usage

```
route_fieldwork(
  matrix,
  units,
  depot,
  max_length = Inf,
  max_stops = Inf,
  service_time = 0,
  demand = 0,
  capacity = Inf,
  iterations = 200,
  time_limit = NULL,
  certify = 1,
  alpha = 0.3,
  seed = 1,
  cost_model = NULL,
  engine = c("fieldopt", "vrpr")
)
```

Arguments

`matrix`	A `travel_matrix()`.
`units`	Names (or indices in the matrix) of the units to visit, or a `select_units()` sample, whose sampled units are visited.

depot	Name (or index) of the depot.
max_length	Maximum length of a route, depot to depot: travel plus the service time of its units when service_time is given; Inf for none.
max_stops	Maximum units per route; Inf for none.
service_time	Time spent at each unit (interviews, measurements), in the unit of the travel matrix: a single number, or a vector named by unit. It counts towards max_length and is reported in durations.
demand	Load each unit adds to its route (interviews to make, samples to carry): a single number or a vector named by unit.
capacity	Maximum load of a route; Inf for none.
iterations	Offspring of the genetic search (ignored when the instance is solved exactly or when time_limit is given).
time_limit	Seconds of search instead of iterations (NULL for none); the only stopping rule of the "vrpr" engine (default 5 s).
certify	Seconds allowed to the branch-and-bound on Held-Karp 1-trees that tries to prove a single tour optimal (symmetric matrix, no limits forcing several routes); 0 switches it off. It also improves the tour when it finds a better one.
alpha	Greediness of the constructions that seed the population, in [0, 1] (0 greedy, 1 random).
seed	Seed of the solver.
cost_model	Optional field_cost_model() to price the solution.
engine	"fieldopt" (the built-in solver) or "vrpr", the PyVRP solver through the vrpr package (in Suggests), with the same inputs and outputs, except that it takes one of max_stops and capacity and gives no lower bound. Both engines charge a cost model's per_route inside the objective (as per_route / per_travel travel units per route), so the number of routes is itself optimised when a cost model is given.

Value

An object of class fieldopt_routes: routes (a tibble with columns route, stop, unit), lengths (travel per route), durations (travel plus service), loads, total, lower_bound, gap, optimal, best_iteration, cost (when a model is given), the inputs and the travel matrix.

References

Vidal, T. (2022). Hybrid genetic search for the CVRP: open-source implementation and SWAP* neighborhood. *Computers & Operations Research*, 140, 105643. Prins, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, 31, 1985-2002. Held, M. and Karp, R. M. (1971). The traveling-salesman problem and minimum spanning trees: part II. *Mathematical Programming*, 1, 6-25.

Examples

```
set.seed(1)
pts <- data.frame(
  unit = c("depot", paste0("s", 1:12)),
  x = c(0, runif(12, 0, 10)), y = c(0, runif(12, 0, 10))
)
m <- travel_matrix(pts, method = "euclidean")
r <- route_fieldwork(m,
  units = paste0("s", 1:12), depot = "depot", max_stops = 5,
  iterations = 50
)
r
```

`routed_cost_curve` *Routed cost curve of a design: the travel constant of Hansen, Hurwitz and Madow*

Description

Selects samples of each size in `n_grid`, routes each one from the depot (or bases), prices it with the cost model, and fits the expected cost as `G(n) = c0 + a n + b sqrt(n)`. This is the cost function of Hansen, Hurwitz and Madow (1953, Vol. I, Chapter 6, Section 12), `C0 sqrt(m) + C1 m`, with `b` in the role of their travel constant `C0` and `a` of the cost per primary unit `C1`. They derived `C0` from points on a regular grid and called it a rough approximation; here it is measured on the frame, with its roads (when the matrix comes from OSRM), its depots and its route limits. The travel of a tour through `n` scattered points grows like `sqrt(n)` (Beardwood, Halton and Hammersley 1959), and fixed costs per visit and per route add the linear term.

Usage

```
routed_cost_curve(
  frame,
  depot,
  cost_model,
  n_grid,
  size = NULL,
  strata = NULL,
  selection = c("lpm", "systematic", "srs"),
  replicates = 1,
  matrix = NULL,
  method = c("haversine", "euclidean", "osrm"),
  max_length = Inf,
  max_stops = Inf,
  n_rep = 10,
  iterations = 100,
  seed = 1
)
```

```
## S3 method for class 'fieldopt_cost_curve'
predict(object, newdata, type = c("total", "average", "marginal"), ...)

## S3 method for class 'fieldopt_cost_curve'
tidy(x, ...)

## S3 method for class 'fieldopt_cost_curve'
glance(x, ...)

## S3 method for class 'fieldopt_cost_curve'
autoplot(object, ...)
```

Arguments

`frame`	Data frame of units as in `select_units()`, including the depot row named in `depot`.
`depot`	Name of the depot unit in `frame$unit`.
`cost_model`	A `field_cost_model()`.
`n_grid`	Sample sizes to route (at least three distinct values for the three coefficients).
`size`	Optional size column for probability-proportional-to-size selection.
`strata`	Optional stratum column, passed to `select_units()`.
`selection`	Selection method of `select_units()`: `"lpm"`, `"systematic"` or `"srs"`.
`replicates`	Replicates of a `"systematic"` selection.
`matrix`	Optional `travel_matrix()` of the frame (computed from the coordinates otherwise).
`method`	Distance method of `travel_matrix()` when `matrix` is `NULL` (`"osrm"` queries the road network once for the whole frame).
`max_length`, `max_stops`	Route limits passed to `route_fieldwork()`.
`n_rep`	Routed samples per sample size.
`iterations`	GRASP iterations per routing.
`seed`	Seed.
`newdata`	Sample sizes (a numeric vector) for `predict()`.
`type`	`"total"` for `G(n)`, `"average"` for `G(n)/n`, `"marginal"` for `G'(n)`.
`...`	Unused.
`x`, `object`	A `fieldopt_cost_curve`.

Details

The fit is constrained to be increasing and concave (`a >= 0`, `b >= 0`), so the marginal cost `G'(n) = a + b / (2 sqrt(n))` is positive and falls with `n`. The marginal cost is the one the classical allocation formulas need (see `two_stage_design()`).

Value

An object of class `fieldopt_cost_curve`: a list with `points` (a tibble with `n`, `cost_mean`, `cost_sd`, `travel_mean`, `routes_mean` per routed size), `coef` (`c0`, `a`, `b`), `r_squared`, `currency` and the settings. `predict()` gives the total, average or marginal cost at any `n`; `tidy()`, `glance()` and `autoplot()` are available.

References

Hansen, M. H., Hurwitz, W. N. and Madow, W. G. (1953). *Sample Survey Methods and Theory*, Vol. I, Chapter 6, Sections 11-15, and Vol. II, Chapter 6, Section 11. Wiley. Beardwood, J., Halton, J. H. and Hammersley, J. M. (1959). The shortest path through many points. *Proc. Cambridge Phil. Soc.*, 55, 299-327.

Examples

```
set.seed(2)
cells <- expand.grid(x = 1:15, y = 1:15)
cells$unit <- paste0("c", seq_len(nrow(cells)))
frame <- rbind(data.frame(unit = "depot", x = 8, y = 8), cells)
model <- field_cost_model(per_travel = 3, per_unit = 40)
curve <- routed_cost_curve(frame, "depot", model,
  n_grid = c(10, 20, 40, 80), method = "euclidean", n_rep = 2,
  iterations = 20
)
curve
predict(curve, c(30, 60), type = "marginal")
```

---

routed_unit_cost *Field cost per sampled unit at a given sample size, from the routing*

Description

Selects `n` units from the frame `n_rep` times, routes each sample from the depot, prices it with the cost model and returns the mean cost per sampled unit. This is the planning value of the cost of a primary unit (`c1` in `two_stage_allocation()`) or of an area-frame unit (`cost_a` in `dual_frame_allocation()`); it depends on `n` because a larger sample is denser and cheaper to reach per unit.

Usage

```
routed_unit_cost(
  frame,
  depot,
  cost_model,
  n,
  size = NULL,
  strata = NULL,
  selection = c("lpm", "systematic", "srs"),
```

```
  replicates = 1,
  matrix = NULL,
  method = c("haversine", "euclidean", "osrm"),
  max_length = Inf,
  max_stops = Inf,
  n_rep = 10,
  iterations = 100,
  seed = 1
)
```

Arguments

frame	Data frame of units as in select_units(), including the depot row named in depot.
depot	Name of the depot unit in frame$unit.
cost_model	A field_cost_model().
n	Sample size (a number, or a vector named by stratum).
size	Optional size column for probability-proportional-to-size selection.
strata	Optional stratum column, passed to select_units().
selection	Selection method of select_units(): "lpm", "systematic" or "srs".
replicates	Replicates of a "systematic" selection.
matrix	Optional travel_matrix() of the frame (computed from the coordinates otherwise).
method	Distance method of travel_matrix() when matrix is NULL ("osrm" queries the road network once for the whole frame).
max_length, max_stops	Route limits passed to route_fieldwork().
n_rep	Replications per sample size.
iterations	GRASP iterations per routing.
seed	Seed.

Value

A one-row tibble: n, cost_per_unit, travel_per_unit, cost_mean, routes_mean, n_rep.

Examples

```
set.seed(1)
frame <- data.frame(
  unit = c("depot", paste0("s", 1:40)),
  x = c(5, runif(40, 0, 10)), y = c(5, runif(40, 0, 10))
)
model <- field_cost_model(
  per_travel = 2, per_unit = 30, per_interview = 10,
  interviews_per_unit = 4
)
```

```
routed_unit_cost(frame, "depot", model,
  n = 12, method = "euclidean", n_rep = 3,
  iterations = 20
)
```

---

routes-methods *Tidy and glance methods for routes and frontiers*

Description

`tidy()` of a `fieldopt_routes` returns one row per stop (`route`, `stop`, `unit`, and `leg`, the travel from the previous stop or the depot); `glance()` one row with the total travel, the lower bound, the gap, the number of routes and the cost when a model was given. `glance()` of a `fieldopt_frontier` returns one row with the range of sample sizes, costs and variances covered, the selection method and the replicates.

Usage

```
## S3 method for class 'fieldopt_routes'
tidy(x, ...)

## S3 method for class 'fieldopt_routes'
glance(x, ...)

## S3 method for class 'fieldopt_frontier'
glance(x, ...)

## S3 method for class 'fieldopt_schedule'
tidy(x, ...)

## S3 method for class 'fieldopt_schedule'
glance(x, ...)
```

Arguments

`x`	A `fieldopt_routes` or `fieldopt_frontier`.
`...`	Unused.

Value

A tibble.

Examples

```
set.seed(1)
pts <- data.frame(unit = c("depot", paste0("s", 1:8)), x = c(0, runif(8)), y = c(0, runif(8)))
r <- route_fieldwork(travel_matrix(pts, method = "euclidean"), paste0("s", 1:8), "depot",
  max_stops = 4, iterations = 20
```

```
)
tidy(r)
glance(r)
```

---

sampled *The sampled units of a sample*

Description

The rows of a select_units() sample that were selected, as a plain tibble, for pipelines: `frame |> select_units(n) |> sampled()`.

Usage

```
sampled(sample)
```

Arguments

sample	A select_units() sample.

Value

A tibble with the sampled rows (coordinates, `pi` and the other columns).

Examples

```
cells <- expand.grid(x = 1:6, y = 1:6)
cells |>
  select_units(n = 5, seed = 1) |>
  sampled()
```

---

schedule_calendar *Redistribute the routes of a schedule over teams and days with constraints*

Description

schedule_fieldwork() hands the routes of each base to its teams with a simple rule (longest route first, one route per team-day). This function solves that distribution exactly as a small integer programme, through the `highs` package (in Suggests), which allows the constraints of real field work: teams unavailable on some days, routes fixed to a team or a day, routes that must precede others, a daily limit per team that lets two short routes share one team-day. The objective finishes as early as possible: it minimises the last working day of the schedule and, among schedules with the same end, the sum of the days on which routes are done.

Usage

```
schedule_calendar(
  schedule,
  availability = NULL,
  fixed = NULL,
  before = NULL,
  daily_limit = NULL,
  time_limit = 10
)
```

Arguments

schedule	A fieldopt_schedule from schedule_fieldwork().
availability	Optional data frame with columns team, day and available (logical): the team-days that are not listed are available. Team names are those of the calendar ("base-1", "base-2", ...).
fixed	Optional data frame with columns base, route and team and/or day: routes that must go to that team and/or that day.
before	Optional data frame with columns base, route, base_after and route_after: the first route must be done on an earlier day than the second.
daily_limit	Maximum duration of a team-day (in the unit of the travel matrix); the routes already respect the max_length of the schedule, so this is only useful to let several short routes share a day. NULL keeps one route per team-day.
time_limit	Seconds allowed to the solver.

Value

The schedule with its calendar, summary and fits recomputed, plus solver ("highs"), objective and status. An error when no calendar satisfies the constraints within days.

Examples

```
set.seed(5)
pts <- data.frame(
  unit = c("base1", "base2", paste0("s", 1:30)),
  x = c(2, 8, runif(30, 0, 10)), y = c(2, 8, runif(30, 0, 10))
)
m <- travel_matrix(pts, method = "euclidean")
sch <- schedule_fieldwork(m,
  units = paste0("s", 1:30), teams = c(base1 = 2, base2 = 1),
  days = 5, max_stops = 4, iterations = 30
)
if (requireNamespace("highs", quietly = TRUE)) {
  off <- data.frame(team = "base1-1", day = 1, available = FALSE)
  schedule_calendar(sch, availability = off)
}
```

schedule_fieldwork *Schedule the field work of several teams from several bases*

Description

Solves the assignment of the units to bases (depots) and their routing jointly, as one multi-depot problem: every route (one team-day) leaves from and returns to one base, each base may send at most `teams * days` routes, and the objective is the total travel plus the `per_route` cost of the cost model for every route opened. The routes of each base are then distributed over its teams and the days available, longest routes first, so that the teams finish as evenly as possible. The result is a calendar: which team visits which units on which day, and whether the work fits.

Usage

```
schedule_fieldwork(
  matrix,
  units,
  teams,
  days,
  bases = NULL,
  max_length = Inf,
  max_stops = Inf,
  service_time = 0,
  cost_model = NULL,
  iterations = 200,
  time_limit = NULL,
  alpha = 0.3,
  seed = 1,
  engine = c("fieldopt", "vrpr"),
  region = NULL
)
```

Arguments

`matrix`	A `travel_matrix()` holding the units and the bases.
`units`	Names of the units to visit, or a `select_units()` sample.
`teams`	Named vector: number of teams per base (names are units of the matrix that act as bases). A single unnamed number with one base is accepted when `bases` is given.
`days`	Number of working days available.
`bases`	Optional names of the bases when `teams` is unnamed.
`max_length`, `max_stops`, `service_time`	Daily limits of a route, as in `route_fieldwork()`.
`cost_model`	Optional `field_cost_model()`; its `per_route` cost is charged once per team-day and enters the objective.

`iterations`, `time_limit`, `alpha`, `seed`, `engine`
: Solver settings of `route_fieldwork()`; both engines solve the multi-depot problem jointly.

`region`
: Optional data frame with columns `unit` and `region` covering every unit and base: the schedule is then solved one region at a time (its units with its bases) and the calendars are combined. This is how a national survey is scheduled state by state without a single matrix over everything; `matrix` may then hold only the pairs within each region (other entries are ignored).

Details

When the team-days of the bases are not enough, the routes are solved without the limit per base and the calendar reports how many days each base would need (`fits` is `FALSE`).

Value

An object of class `fieldopt_schedule`: `calendar` (a tibble with one row per route: `base`, `team`, `day`, `route`, `stops`, `travel`, `duration`, and `units` as a list column), `assignment` (unit, base), `routes` (the `route_fieldwork()` solution of each base), `summary` (per base: units, teams, routes, days needed, days available, whether it fits), `fits` and `cost`.

Examples

```
set.seed(5)
pts <- data.frame(
  unit = c("base1", "base2", paste0("s", 1:30)),
  x = c(2, 8, runif(30, 0, 10)), y = c(2, 8, runif(30, 0, 10))
)
m <- travel_matrix(pts, method = "euclidean")
sch <- schedule_fieldwork(m,
  units = paste0("s", 1:30), teams = c(base1 = 2, base2 = 1),
  days = 4, max_stops = 4, iterations = 30
)
sch
sch$calendar
```

---

segment_estimator *Segment estimators of a total from an area sample of segments*

Description

The three classical ways to attribute establishments to the segments of an area frame (Houseman 1975; Nealon 1984):

Usage

```
segment_estimator(tracts, sample, type = c("closed", "open", "weighted"))
```

Arguments

tracts	Data frame with one row per (segment, establishment) pair: `unit` (segment), `establishment`, and, as needed, `y_tract` (value inside the segment, for `"closed"`), `y_total` (value of the whole establishment, for `"open"` and `"weighted"`), `headquarters` (logical, headquarters inside the segment, for `"open"`) and `share` (fraction of the establishment's land inside the segment, for `"weighted"`).
sample	The `fieldopt_sample` of segments.
type	`"closed"`, `"open"` or `"weighted"`.

Details

- `"closed"`: each segment reports what lies inside it (the tracts);
- `"open"`: each establishment is attributed entirely to the segment that holds its headquarters;
- `"weighted"`: each establishment is attributed to every segment it touches in proportion to the share of its land inside the segment.

The segment totals are then expanded with the segment design.

Value

A one-row tibble as in `design_variance()` with a column `type`.

References

Nealon, J. P. (1984). Review of the multiple and area frame estimators. USDA Statistical Reporting Service, Staff Report 80.

---

select_points *Select points inside sampled grid cells*

Description

For an area frame of square cells (segments) selected with `select_units()`, draws `points_per_cell` uniform random points inside each sampled cell. In the field, each point identifies the field (and so the establishment) it falls on; an establishment can be hit several times. The point density of a cell, `pi * points / area`, is what `point_expected_hits()` and `point_estimator()` need.

Usage

```
select_points(
  sample,
  points_per_cell,
  cell_size,
  layout = c("random", "systematic"),
  seed = 1
)
```

Arguments

`sample`	A `fieldopt_sample` of cells with coordinate columns giving the cell centres.
`points_per_cell`	Points per sampled cell: a single number, a vector named by stratum (for example more points where agricultural use is more intense), or the name of a column of the sample.
`cell_size`	Side of the square cells, in the units of the coordinates (one number, or `c(dx, dy)`).
`layout`	`"random"` (independent uniform points) or `"systematic"` (a regular grid of `k1 x k2 = m` points with one random offset per cell, which spreads the points inside the cell; each point is still uniform over the cell, so the density is the same).
`seed`	Seed.

Value

A tibble of class `fieldopt_points` with one row per point: `unit` (the cell), `point`, the two coordinates, `pi_cell`, `points_in_cell` and `density` (expected points per unit area of the cell).

Examples

```
cells <- expand.grid(x = 1:10, y = 1:10)
cells$unit <- paste0("c", 1:100)
s <- select_units(cells, n = 12, seed = 2)
p <- select_points(s, points_per_cell = 9, cell_size = 1, layout = "systematic")
nrow(p)
head(p)
```

`select_units` *Select units with a spatially balanced or systematic probability sample*

Description

Draws a probability sample of the units of a frame (segments, grid cells, establishments) with equal or size-proportional inclusion probabilities, within strata when given, by one of three methods:

Usage

```
select_units(
  frame,
  n,
  size = NULL,
  coords = NULL,
  strata = NULL,
  method = c("lpm", "systematic", "srs", "cube"),
  replicates = 1,
```

```
  seed = 1,
  balance = NULL
)
```

Arguments

`frame`	Data frame with one row per unit: coordinate columns (`lat`, `lon` or `x`, `y`, or those named in `coords`), an optional `size` column, an optional `unit` column with names and an optional stratum column.
`n`	Sample size: a single number (allocated to strata in proportion to the sum of `size`, or to the number of units when `size` is `NULL`), or a vector named by stratum.
`size`	Column of the size measure for probability-proportional-to-size selection, or `NULL` for equal probabilities within stratum.
`coords`	Names of the coordinate columns used for spatial balance (any number of columns for `"lpm"`; exactly two for `"systematic"`).
`strata`	Column with the stratum of each unit, or `NULL`.
`method`	`"lpm"`, `"systematic"`, `"srs"` or `"cube"` (balanced sampling by the cube method of Deville and Tillé 2004, with the fast flight phase and landing by suppression of variables).
`replicates`	Number of independent interpenetrating replicates for `"systematic"` (each a systematic sample of about `n / replicates` units; replicates may share units). Ignored by the other methods.
`seed`	Seed.
`balance`	For `"cube"`: names of the columns to balance on (the inclusion probabilities are always included, so the sample size is fixed). The Horvitz-Thompson estimates of these columns match their population totals as closely as the landing allows.

Details

- `"lpm"`: the local pivotal method (Grafström, Lundström and Schelin, 2012), which spreads the sample over the coordinate space so that nearby units are rarely selected together;
- `"systematic"`: systematic sampling with probabilities proportional to size along a Hilbert curve through the coordinates (a spatially ordered systematic sample), drawn as `replicates` independent interpenetrating systematic samples so that a design-based variance can be estimated;
- `"srs"`: simple random sampling without replacement.

Spatial balance reduces the variance of totals of spatially structured variables and, at the same time, spreads the field work; the cost model and the routing make that tension explicit.

Value

The frame as a tibble with columns `pi` (inclusion probability in the union of the replicates) and `sampled`, of class `fieldopt_sample`, with attributes `n`, `coords`, `strata`, `method`, `seed` and, for replicated systematic samples, `replicates` (a logical matrix, one column per replicate) and `pi_replicate` (inclusion probability within one replicate).

References

Grafström, A., Lundström, N. L. P. and Schelin, L. (2012). Spatially balanced sampling through the pivotal method. *Biometrics*, 68(2), 514–520.

Examples

```
set.seed(1)
cells <- expand.grid(x = 1:20, y = 1:20)
cells$unit <- paste0("c", seq_len(nrow(cells)))
cells$intensity <- cut(
  cells$x + rnorm(400, sd = 3), c(-Inf, 7, 14, Inf),
  c("low", "mid", "high")
)
cells$size <- c(low = 1, mid = 2, high = 4)[cells$intensity]
s <- select_units(cells,
  n = c(low = 10, mid = 15, high = 25), size = "size",
  strata = "intensity"
)
table(s$intensity, s$sampled)
r <- select_units(cells, n = 40, method = "systematic", replicates = 4)
dim(attr(r, "replicates"))
```

---

spatial_balance *Spatial balance of a sample*

Description

The Voronoi measure of Stevens and Olsen (2004): every unit of the frame is assigned to its nearest sampled unit, the inclusion probabilities assigned to each sampled unit are summed, and the balance is the mean squared deviation of those sums from one. Zero is perfect balance; simple random samples give values near one. With strata the measure is computed within each stratum and pooled.

Usage

```
spatial_balance(sample, by_stratum = FALSE)
```

Arguments

`sample`	A `select_units()` sample.
`by_stratum`	Return one value per stratum instead of the pooled one.

Value

A number, or a named vector with `by_stratum = TRUE`.

References

Stevens, D. L. and Olsen, A. R. (2004). Spatially balanced sampling of natural resources. *Journal of the American Statistical Association*, 99, 262-278.

Examples

```
cells <- expand.grid(x = 1:10, y = 1:10)
spatial_balance(select_units(cells, n = 10, method = "lpm", seed = 1))
spatial_balance(select_units(cells, n = 10, method = "srs", seed = 1))
```

---

tidy.fieldopt_multi_allocation

Tidy and glance methods for allocations

Description

`tidy()` returns one row per stratum, frame or stage; `glance()` one row with the totals.

Usage

```
## S3 method for class 'fieldopt_multi_allocation'
tidy(x, ...)

## S3 method for class 'fieldopt_multi_allocation'
glance(x, ...)

## S3 method for class 'fieldopt_allocation'
tidy(x, ...)

## S3 method for class 'fieldopt_allocation'
glance(x, ...)

## S3 method for class 'fieldopt_dual_frame'
tidy(x, ...)

## S3 method for class 'fieldopt_dual_frame'
glance(x, ...)

## S3 method for class 'fieldopt_two_stage'
tidy(x, ...)

## S3 method for class 'fieldopt_two_stage'
glance(x, ...)
```

Arguments

`x`	A `fieldopt_allocation`, `fieldopt_dual_frame` or `fieldopt_two_stage`.
`...`	Unused.

Value

A tibble.

Examples

```
strata <- data.frame(
  stratum = c("list", "area"), size = c(2000, 800),
  sd = c(12, 30), cost = c(40, 180)
)
a <- frame_allocation(strata, budget = 20000)
tidy(a)
glance(a)
```

`travel_matrix` *Travel matrix between the units of a frame*

Description

Builds the dense matrix of travel distances or times between all units, depot included, in one of three ways:

Usage

```
travel_matrix(
  coords,
  method = c("haversine", "euclidean", "osrm"),
  matrix = NULL,
  unit = NULL,
  detour = 1,
  measure = c("duration", "distance"),
  server = getOption("osrm.server"),
  profile = getOption("osrm.profile"),
  block = 100,
  bases = NULL,
  speed = NULL
)
```

Arguments

`coords`	Data frame or matrix with two coordinate columns: `lat`, `lon` (degrees) for `"haversine"` and `"osrm"`, or `x`, `y` (planar, any unit) for `"euclidean"`. Row names, or a column `unit`, name the units. A `select_units()` sample is accepted and only its sampled units are used, so that `frame |> select_units(n) |> travel_matrix(bases = towns)` builds the matrix of the field work.
`method`	`"haversine"`, `"euclidean"` or `"osrm"`.
`matrix`	Optional square numeric matrix of travel costs already computed; when given, `method` is ignored except for the coordinate columns it implies.
`unit`	Label of the travel unit (`"km"`, `"min"`, ...), kept for reports; set automatically unless a matrix is supplied.

`detour`	Multiplier applied to straight-line distances (`>= 1`). Ignored, with a warning, for `"osrm"` and supplied matrices.
`measure`	For `"osrm"`: `"duration"` (minutes) or `"distance"` (kilometres on the network).
`server, profile`	For `"osrm"`: the base URL of the routing server and the profile (`"car"`, `"bike"`, `"foot"`); default to the `osrm` package options.
`block`	For `"osrm"`: number of origins and of destinations per request (`block^2` cells each).
`bases`	Optional data frame of depots (a `unit` column and the same coordinate columns) put in front of `coords`.
`speed`	Optional speed in coordinate units per hour (kilometres per hour with `"haversine"`): the distances are turned into minutes and `unit` becomes `"min"`. Ignored for `"osrm"` durations.

Details

- from coordinates, as straight-line distances: great-circle (`"haversine"`, kilometres) with `lat/lon`, or Euclidean with planar `x/y`. A `detour` factor (the ratio of road to straight-line distance, typically 1.2 to 1.4 on rural road networks) turns them into a planning estimate of road distances when no network is available;
- from a road network, with `method = "osrm"`: durations (minutes) or distances (kilometres) from an OSRM server through the `osrm` package, requested in blocks so that the server's table limit is respected (the public demo server allows 10 000 cells per request, and is meant for small tests only: run your own server for a frame);
- from a matrix computed elsewhere (`matrix`), with the coordinates used only for names and plots.

Road matrices need not be symmetric (one-way streets, different durations by direction); the routing solver prices every leg in the direction travelled.

Value

A square numeric matrix of class `fieldopt_matrix` with unit names as dimnames and attributes `coords`, `method`, `unit`, `detour` and, for `"osrm"`, `snap` (the distance in kilometres from each unit to the point of the network it was snapped to: large values flag units far from any road).

Examples

```
segments <- data.frame(
  unit = c("depot", "s1", "s2", "s3"),
  lat = c(-8.05, -8.10, -8.00, -8.12),
  lon = c(-34.90, -34.95, -34.85, -34.80)
)
travel_matrix(segments)
travel_matrix(segments, detour = 1.3)
## Not run:
# road durations from the OSRM demo server (small frames only)
travel_matrix(segments, method = "osrm", measure = "duration")

## End(Not run)
```

two_stage_allocation *Two-stage allocation with a field cost function*

Description

n primary units (segments, grid cells) are visited and m secondary units (points, tracts, establishments) are observed in each, at a cost c1 * n + c2 * n * m. With s2_between the variance among primary-unit means and s2_within the variance within primary units, the variance of the estimated mean per secondary unit is (1 - n/N) s2_between / n + (1 - m/M) s2_within / (n m) and the cost-optimal m is sqrt(c1 s2_within / (c2 (s2_between - s2_within / M))) (Cochran 1977, Section 10.6). The cost c1 of visiting a primary unit is where the routing enters: it is not a constant but depends on how spread the sample is; take it from cost_variance_frontier() at the candidate n and iterate.

Usage

```
two_stage_allocation(
  n_primary,
  m_secondary,
  s2_between,
  s2_within,
  c1,
  c2,
  target_variance = NULL,
  target_cv = NULL,
  budget = NULL,
  mean = NULL
)
```

Arguments

n_primary, m_secondary
: Primary units in the population and secondary units per primary unit (an average when they differ; need not be a whole number). The allocation tries every whole m up to m_secondary and keeps the cheapest pair meeting the target (or the least variance within the budget), so small populations are handled exactly.

s2_between, s2_within
: Variance among primary-unit means and within primary units.

c1, c2
: Cost per primary unit visited and per secondary unit observed.

target_variance, target_cv, budget
: Exactly one: target variance of the estimated **total**, target coefficient of variation (needs mean), or budget.

mean
: Population mean per secondary unit, for target_cv.

Value

A list of class `fieldopt_two_stage`: `n`, `m`, `m_optimal`, `cost`, `variance_mean`, `variance_total`, `se_total`, `cv` (when `mean` is given), `bounded`.

Examples

```
two_stage_allocation(
  n_primary = 5000, m_secondary = 20, s2_between = 4, s2_within = 25,
  c1 = 300, c2 = 20, budget = 1e5
)
two_stage_allocation(5000, 20, 4, 25, c1 = 300, c2 = 20, target_cv = 0.03, mean = 12)
```

---

two_stage_design *Two-stage allocation with the primary-unit cost taken from the routing*

Description

`two_stage_allocation()` needs the cost `c1` of visiting a primary unit, which is not a constant: a tour shares its travel among the units it visits, so the routed cost `G(n)` of visiting `n` units grows like `c0 + a n + b sqrt(n)` (Beardwood, Halton and Hammersley). This function measures `G(n)` by routing samples at a few sizes around the current allocation, fits that curve, and re-allocates with the **marginal** cost `G'(n)` in place of `c1`; with a non-linear cost the optimal number of secondary units is Cochran's formula with the marginal, not the average, cost per primary unit, as in Hansen, Hurwitz and Madow's travel-cost model. Under a budget the number of primary units then comes from the fitted curve, so the design spends the budget at its real routed cost. It stops when `n` no longer changes (or after `max_iter` rounds; a two-cycle stops with the better of the two designs).

Usage

```
two_stage_design(
  frame,
  depot,
  cost_model,
  m_secondary,
  s2_between,
  s2_within,
  target_variance = NULL,
  target_cv = NULL,
  budget = NULL,
  mean = NULL,
  c1_start = NULL,
  max_iter = 8,
  size = NULL,
  strata = NULL,
  selection = c("lpm", "systematic", "srs"),
  matrix = NULL,
  method = c("haversine", "euclidean", "osrm"),
```

```
  max_length = Inf,
  max_stops = Inf,
  n_rep = 5,
  iterations = 100,
  seed = 1
)
```

Arguments

frame	Data frame of units as in select_units(), including the depot row named in depot.
depot	Name of the depot unit in frame$unit.
cost_model	A field_cost_model().
m_secondary	Secondary units per primary unit in the population.
s2_between, s2_within	Variance among primary-unit means and within primary units.
target_variance, target_cv, budget	Exactly one: target variance of the estimated **total**, target coefficient of variation (needs mean), or budget.
mean	Population mean per secondary unit, for target_cv.
c1_start	Starting value of the cost per primary unit (default: the per-visit cost plus the travel cost of the median trip from the depot).
max_iter	Maximum number of rounds.
size	Optional size column for probability-proportional-to-size selection.
strata	Optional stratum column, passed to select_units().
selection	Selection method of select_units(): "lpm", "systematic" or "srs".
matrix	Optional travel_matrix() of the frame (computed from the coordinates otherwise).
method	Distance method of travel_matrix() when matrix is NULL ("osrm" queries the road network once for the whole frame).
max_length, max_stops	Route limits passed to route_fieldwork().
n_rep	Routed samples per sample size of the cost curve.
iterations	GRASP iterations per routing.
seed	Seed.

Details

Using the average cost G(n)/n instead takes too few primary units: in the package's experiments the marginal rule was within 0.2 percent of the cost of the best design found by exhaustive search, the average rule up to 3 percent above it, and a constant c1 guessed from depot distances 7 to 19 percent above.

Value

The final `fieldopt_two_stage` allocation, with `cost` the routed cost `G(n) + c2 n m`, and the extra elements `c1` (the marginal cost used), `c1_average` (`G(n)/n`), `cost_curve` (the routed points: `n`, `cost_mean`, `cost_sd`), `cost_coef` (`c0`, `a`, `b`), `history` (one row per round: `round`, `c1`, `c1_average`, `n`, `m`, `cost`, `variance_total`) and `converged`.

References

Hansen, M. H., Hurwitz, W. N. and Madow, W. G. (1953). *Sample Survey Methods and Theory*, Vol. II, Chapter 6, Section 11 (two-stage optimum with a travel cost proportional to `sqrt(m)`). Cochran, W. G. (1977). *Sampling Techniques*, 3rd ed., Section 10.6. Beardwood, J., Halton, J. H. and Hammersley, J. M. (1959). The shortest path through many points. *Proc. Cambridge Phil. Soc.*, 55, 299-327.

Examples

```
set.seed(2)
cells <- expand.grid(x = 1:12, y = 1:12)
cells$unit <- paste0("c", seq_len(nrow(cells)))
frame <- rbind(data.frame(unit = "depot", x = 6, y = 6), cells)
model <- field_cost_model(
  per_travel = 3, per_unit = 40, per_interview = 20,
  interviews_per_unit = 1
)
two_stage_design(frame, "depot", model,
  m_secondary = 12, s2_between = 9, s2_within = 40,
  target_cv = 0.05, mean = 10, method = "euclidean", n_rep = 2,
  iterations = 20
)
```

Index

allocation-methods
(tidy.fieldopt_multi_allocation), 40
as_frame, 3
as_frame(), *4*, *24*
as_sf, 4
as_sf(), *3*, *24*
as_svydesign, 4
autoplot.fieldopt_cost_curve
(routed_cost_curve), 27
autoplot.fieldopt_frontier, 5
autoplot.fieldopt_routes, 6
autoplot.fieldopt_sample, 7
autoplot.fieldopt_schedule, 7

core_version, 8
cost_variance_frontier, 8
cost_variance_frontier(), *12*, *43*

design_variance, 10
design_variance(), *5*, *9*, *16*, *22–24*, *36*
dual_frame_allocation, 11
dual_frame_allocation(), *9*, *13*, *17*, *20*, *29*
dual_frame_design, 13
dual_frame_estimator, 15
dual_frame_estimator(), *18*

expected_hits (point_expected_hits), 23

field_cost_model, 16
field_cost_model(), *9*, *14*, *26*, *28*, *30*, *34*, *45*
frame_allocation, 17
frame_allocation(), *19*
frame_overlap, 18

glance.fieldopt_allocation
(tidy.fieldopt_multi_allocation), 40
glance.fieldopt_cost_curve
(routed_cost_curve), 27
glance.fieldopt_dual_frame
(tidy.fieldopt_multi_allocation), 40
glance.fieldopt_frontier
(routes-methods), 31
glance.fieldopt_multi_allocation
(tidy.fieldopt_multi_allocation), 40
glance.fieldopt_routes
(routes-methods), 31
glance.fieldopt_schedule
(routes-methods), 31
glance.fieldopt_two_stage
(tidy.fieldopt_multi_allocation), 40

multivariate_allocation, 19

point_design_effect, 20
point_estimator, 22
point_estimator(), *20, 21*, *23*, *36*
point_expected_hits, 23
point_expected_hits(), *22*, *36*
predict.fieldopt_cost_curve
(routed_cost_curve), 27

ratio_estimator, 23
read_areaframe, 24
route_fieldwork, 25
route_fieldwork(), *9*, *14*, *28*, *30*, *34, 35*, *45*
routed_cost_curve, 27
routed_unit_cost, 29
routes-methods, 31

sampled, 32
schedule_calendar, 32
schedule_fieldwork, 34
schedule_fieldwork(), *32, 33*
segment_estimator, 35
select_points, 36

select_points(), *20*, *23*
select_units, 37
select_units(), *3–5*, *9, 10*, *13–15*, *18*, *20, 21*, *24, 25*, *28*, *30*, *32*, *34*, *36*, *39*, *41*, *45*
spatial_balance, 39
survey::svrepdesign(), *5*
survey::svydesign(), *5*

tidy.fieldopt_allocation *(*tidy.fieldopt_multi_allocation*)*, 40
tidy.fieldopt_cost_curve *(*routed_cost_curve*)*, 27
tidy.fieldopt_dual_frame *(*tidy.fieldopt_multi_allocation*)*, 40
tidy.fieldopt_multi_allocation, 40
tidy.fieldopt_routes *(*routes-methods*)*, 31
tidy.fieldopt_schedule *(*routes-methods*)*, 31
tidy.fieldopt_two_stage *(*tidy.fieldopt_multi_allocation*)*, 40
travel_matrix, 41
travel_matrix(), *3*, *9*, *14*, *25*, *28*, *30*, *34*, *45*
two_stage_allocation, 43
two_stage_allocation(), *9*, *29*, *44*
two_stage_design, 44
two_stage_design(), *13*, *28*