

Package: rcef (via r-universe)

October 22, 2026

Type Package

Title Discard Raising and Length/Age Catch Allocation over CEF (RDBES)
Data

Version 0.1.0

Description Implements the ICES discard-raising and length/age catch allocation workflows over CEF (RDBES) long-format data: discard-rate and sampling-coverage tables with the plots that draw them, plots of the numbers and mean weight at length and age, and plots of the raised and allocated catch, with one colour palette for all of them. Wraps the RDBESStockCoord engines catchRaising() and catchAllocationAoL(), and writes what they give in the format of the stock coordination (Catch Combined and Distributions Combined).

License file LICENSE

URL <https://github.com/DVillagra26/rcef>

BugReports <https://github.com/DVillagra26/rcef/issues>

Encoding UTF-8

Roxygen list(markdown = FALSE)

RoxygenNote 7.3.3

Depends R (>= 4.1)

Imports dplyr, tidyr, rlang, stringr, data.table, ggplot2 (>= 3.5.0),
ggalluvial, ggrepel, FactoMineR, scales (>= 1.3.0), stats,
utils, tools, grDevices, grid

VignetteBuilder knitr

Suggests knitr, rmarkdown, icesVocab, icesSD, maps, units, pkgload,
remotes, testthat (>= 3.0.0)

Config/pak/sysreqs cmake libfreetype6-dev make libicu-dev libpng-dev libuv1-dev zlib1g-dev

Repository <https://ices-tools-prod.r-universe.dev>

Date/Publication 2026-10-22 09:05:06 UTC

RemoteUrl <https://github.com/DVillagra26/rcef>

RemoteRef HEAD

RemoteSha c3bccadd5866dbd07711e2321919f2cb1d46f1f6

Contents

build_coverage_strata	2
build_discard_clusters	6
catchAllocation	7
catchAllocationAoL	10
catchAtAoLAggregation	13
catchCombined	15
catchRaising	17
check_group_conditions	20
coverage-plots	21
derive_fields	24
distributionsCombined	26
funICoutCEF	30
funIntercatchCEF	31
funMakeRelation	32
meanWoLatAoLAggregation	32
plot_catches	34
plot_compare_raised	36
plot_coverage_map	38
plot_discard_cluster_variables	40
plot_discard_clusters	41
plot_DR	42
plot_MWaLoA	45
plot_NaLoA	48
plot_recommended_groups	51
plot_river	53
raised_overview_table	54
recommend_groups	56
saving-plots	61
Index	63

build_coverage_strata *Sampling coverage and discard rate per stratum*

Description

One row per stratum: what was landed and discarded, the discard rate, and how well the stratum is sampled - a table you can read, sort and filter, at whatever aggregation you choose. Give it the catch and the columns that define a stratum; for length and age, also the distributions.

This is the table behind [plot_coverage_grid\(\)](#), [plot_coverage_bar\(\)](#), [plot_coverage_map\(\)](#) and [plot_DR\(\)](#). They build it themselves from raw catch and take it directly when you pass it, so every number in a figure can be found in this table, and a table can always be looked at as a figure.

Usage

```
build_coverage_strata(
  catch,
  dims = NULL,
  distributions = NULL,
  what = "discard",
  catchCategory = NULL
)
```

Arguments

catch	CEF catch data (long format) carrying catchCategory, total and, for discards, domainCatchDis, plus every column named in dims. Those can be columns of your own, or the ones <code>derive_fields()</code> adds (metier4, mesh, metier4_mesh, the ICES area levels): run it once on the raw catch and pass that - the table reads the columns the catch has and derives nothing itself.
dims	Character vector of the columns that define a stratum, for example <code>c("metier4", "vesselFlagCountry")</code> , or the finer <code>c("fleetValue", "areaValue", "seasonValue")</code> . Default NULL: nothing is pooled, and the table is built at the finest a stratum naturally gets (fleetValue, areaValue, vesselFlagCountry, seasonValue - whichever are present in catch). For length and age catchCategory is never a stratum column (it is a column of the result, one row per catch part) and is ignored if named.
distributions	A CEF distributions table. Required for what = "length" and what = "age", ignored for discards.
what	"discard" (default), "length" or "age". The plural and mixed case are accepted ("discards", "Length").
catchCategory	For what = "length" / "age": "All" (default - landings and discards, one row each), "Lan", "Dis" or "BMS". Not valid for what = "discard" (an error).

Details**What the table leaves out:**

It reports numbers and does not judge them: there is no status, category or decision column. The thresholds (`thresholds = list(cov = 50, trips = NA, fish = NA)`) are the choice of the functions that judge these numbers: `plot_coverage_grid()` applies them as it draws and `recommend_groups()` to choose the groups. This function takes none.

Column names and units:

Columns follow the final CAR / CAS output tables wherever those have a counterpart (total, weightingTotal, rate, numTrips, catchCategory, variableUnit), so a stratum here reads like its row there. Weights are in tonnes; the CAR / CAS tables themselves keep the CEF's kilograms. Both ways of expressing the discard rate are in the table - rate (D/L) and discardRate (D/(L+D)) - so either can be read, sorted or plotted.

Discard tables (what = "discard"):

One row per stratum, with the discard estimate on the Dis side and the landings that weight it on the Lan side. Only landings and discards are counted: BMS is a third catch part with its own raising, not a discard.

Column	Meaning
the dims columns	The stratum, as you named it.
catchCategory, variableUnit	"Dis" and "t": what total is.
total	Discards. NA when the stratum has no discard estimate at all (nobody looked), or when a ra
weightingCategory	"Lan".
weightingTotal	The landings that carry a discard estimate: the weight rate is weighted by. A landing does
rate	$\text{total} / \text{weightingTotal}$ (D/L): discards per tonne of covered landings, as the CAS output
discardRate	$\text{total} / (\text{weightingTotal} + \text{total})$ (D/(L+D)): the discards as a share of the covered lan
landings	All landings in the stratum, covered or not.
coverage	$\text{weightingTotal} / \text{landings}$, in percent; NA when there are no landings.
numTrips	The trips behind the stratum; NA when the data does not report them. The largest count amon
importedOrRaised	Only when catch is a raised catch (the output of <code>catchRaising()</code>): "raised" when the st

Length and age tables (what = "length", what = "age"):

One row per stratum **and catch part**, so catchCategory = "All" (the default) gives a Lan and a Dis row for every stratum.

Column	Meaning
the dims columns	The stratum, as you named it.
catchCategory	The catch part the row describes: "Lan", "Dis" or "BMS".
distributionType	"LengthTotal" or "Age".
variableUnit	"t".
total	The weight of that catch part.
coverage	The share of total, in percent, whose domainBiology matches a distribution of that type and catch part
numMeasurements	The measurements behind those distributions (0 when there are none).
numTrips	As above.

Choosing the aggregation:

dims is the aggregation: name fewer columns for a coarser table (each row pools everything else), more for a finer one. Weights add up, so a table built finer can be pooled to coarser plotted columns later without loss - coverage and rate are re-derived from the summed weights, never averaged. Leave dims out and nothing is pooled at all: the table is built at the finest a stratum naturally gets - fleetValue, areaValue, vesselFlagCountry, seasonValue, whichever are actually present in catch - so a stratum here is as fine as the data itself, and only naming dims yourself asks for anything coarser.

The other derived columns are not lost:

A column `derive_fields()` added but you left out of dims (mesh when dims is just "metier4", say) still rides along on each row, *when* that stratum pins it to a single value - so the table can be filtered on it afterwards without having named it as part of the aggregation. A stratum coarser than the column - several meshes inside one metier4 - reports NA there instead of an arbitrary pick, since no single value would be true of the whole row.

Build once, draw many:

For length and age the expensive step is linking the catch to the distributions. Build the table once and pass it to as many plots as you like, rather than passing the catch to each. The same goes for `derive_fields()`: run it once, on the raw catch, and keep building on that catch - the table and the plots never derive anything again.

Value

A data frame with the columns described above, or NULL when there is nothing to describe (no landings or discards, or no catch rows for that catchCategory). Distributions that link to nothing are not "nothing": every stratum is then reported with coverage 0.

See Also

`coverage-plots` and `plot_DR()`, which draw this table.

Other coverage plots: `coverage-plots`, `plot_coverage_map()`

Examples

```
## Not run:

# discard rate and coverage per metier x country
cov <- build_coverage_strata(catch, c("metier4", "vesselFlagCountry"))
head(cov[order(-cov$landings), ])

# the same table, coarser or finer
build_coverage_strata(catch, "seasonValue")
nrow(build_coverage_strata(catch,
                           c("fleetValue", "areaValue", "seasonValue")))

# dims left out: nothing pooled - one row per fleetValue x areaValue x
# vesselFlagCountry x seasonValue combination the catch actually has
nrow(build_coverage_strata(catch))

# length coverage of the landings: weight, share with a distribution,
# and the measurements behind it
len <- build_coverage_strata(catch, c("metier4", "vesselFlagCountry"),
                           distributions, what = "length",
                           catchCategory = "Lan")
head(len[order(-len$total), ])

# the plots draw the same table, from the raw catch or from the table
plot_coverage_grid(cov, y = "metier4", x = "vesselFlagCountry",
                  thresholds = list(cov = 50))
plot_DR(cov, x = "metier4", colour = "vesselFlagCountry")

## End(Not run)
```

build_discard_clusters

Which strata share a discard rate, and which variables predict that

Description

Groups sampled strata into k clusters of similar discard rate (k-means), places every stratum on a map of dims built by MCA (Multiple Correspondence Analysis - the categorical equivalent of PCA, since dims is ordinarily categorical: metier4, mesh, vesselFlagCountry...), and scores how strongly each of dims lines up with the resulting clusters. The table this returns is what [plot_discard_clusters\(\)](#) and [plot_discard_cluster_variables\(\)](#) draw; nothing here needs a distribution or a raised catch.

Usage

```
build_discard_clusters(catch, dims, k = 3, seed = 1)
```

Arguments

catch	Raw CEF catch data.
dims	Categorical columns to place on the map and to score for importance - typically fleet/area/country/season columns (derive_fields() output), e.g. c("metier4", "mesh", "vesselFlagCountry").
k	Number of discard-rate clusters.
seed	Seed for the (otherwise global-state-free) k-means run, so the same data and k always cluster the same way.

Details

The two steps are independent on purpose. Clustering runs on the discard rate alone (stats::kmeans()), not on the MCA coordinates, so a cluster always means "similar discard rate", never "similar categories that happen to also have a similar rate". The MCA map is only where those clusters are then shown, so [plot_discard_clusters\(\)](#) can answer "does a cluster occupy its own part of variable space, or is it scattered through it" - the map's two dimensions rarely capture more than a fifth of the total structure (eig), so treat it as a sketch, not the full picture.

Clustering uses discardRate (bounded 0-100%) rather than rate, which can run past 100% for a stratum with very little covered landings and would otherwise dominate the clustering on its own.

Value

A list, or NULL when no stratum has a discard rate:

- strata - one row per sampled stratum: dims, rate, discardRate, landings, cluster (low to high mean rate) and Dim1/Dim2 (its MCA coordinates).
- importance - one row per dims column, its Cramer's V against cluster, sorted strongest first.
- eig - percent of variance the MCA's first dimensions explain.

See Also

[build_coverage_strata\(\)](#) for the table this clusters; [plot_discard_clusters\(\)](#), [plot_discard_cluster_variables\(\)](#)

catchAllocation	<i>Catch Allocation (CAR and CAS): how the discards were raised, in the format of the stock coordination</i>
-----------------	--

Description

The raised catch written down stratum by stratum, the way the allocation setup of the stock coordination format wants it: the **Catch Allocation Results (CAR)**, one row for every stratum of the catch - imported or raised - with the group it was raised for or fed, and the **Catch Allocation Settings (CAS)**, one row for every donor stratum of every group, with the discard ratio it gave and the weight it counted for. Give it what [catchRaising\(\)](#) was given and made: the raised catch and the conditions of the groups.

Usage

```
catchAllocation(
  catch_data,
  condition_raising_st_list,
  condition_matched_data_list = condition_raising_st_list,
  originType = c("WGEstimate", "Official"),
  variableType = c("WeightLive", "Number"),
  catchAllocationSchemeName = NULL,
  unit = NULL,
  round = NULL,
  workingGroupYear,
  version = 1,
  versionDescription = NA_character_,
  calendarYear = "Y",
  startDateYear = NA_character_
)
```

Arguments

catch_data	The raised catch: the result of catchRaising() , with its importedOrRaised, DrGroup and the catch's own columns.
condition_raising_st_list, condition_matched_data_list	The conditions the catch was raised with: the strata each group raises and the data each takes its discard ratio from, as in catchRaising() (the same names and order; quosures or text, not logical vectors, which refer to the rows of the catch before the raising). The matched data default to the strata. A scheme of recommend_groups() has both in strataCond and matchedDataCond. A condition that uses an object of the session (a list of strata, say) rather than columns of the catch is evaluated where it was made: unquote the object into it - <code>rlang::quo(Stratum %in% !!high)</code> - to keep the condition for another session (<code>saveRDS()</code>).

originType, variableType	The origin and the variable of the rows the catch was raised for, as in <code>catchRaising()</code> : "WGEstimate" and "WeightLive" by default. The rows of other origins and variables are not part of the tables.
catchAllocationSchemeName	The name of the scheme the discards were allocated by, a string; default the versionDescription.
unit	The unit to convert the weights to before the tables are made: "t" or "kg" (for weights; the raising works in kg). NULL (default) leaves them as they are.
round	Digits to round total and weightingTotal to; NULL (default) does not round. The rates are not rounded.
workingGroupYear, version, versionDescription, calendarYear, startDateYear	The header of <code>catchCombined()</code> ; workingGroupYear has no default.

Details

The CAR:

Every stratum of the raised catch is a row: the landings, the discards that were reported and the discards the raising made, whichever of the origin and the variable the raising was run for. imported says which: "N" for the discards the raising made (importedOrRaised == "raised"), "Y" for all the others. The two names of the group say how the strata took part in the raising:

- catchAllocationGroupNameTarget: the group the stratum was **raised for**, on the discards the raising made (imported = "N") and on those only. It is the group of condition_raising_st_list the stratum falls in.
- catchAllocationGroupNameSource: the group(s) the stratum **feeds**, on the donor strata (imported = "Y") and on those only - the landings and the discards of the strata whose discard ratio went into a group's. A stratum that feeds several groups has them all, separated by "; ". A group's donors are the strata of its condition_matched_data_list condition that have a discard estimate (and landings), and the other rows of the same domain, as `catchRaising()` picks them.
- catchAllocationSchemeName is written on the raised discards only.

A group that raised nothing (it had no landings without an estimate, or no donor to take a ratio from) is neither a target nor a source: it is not in the tables. A raised discard with no total - the ratio of a group with no donor is NaN - is not a result and is left out, with a warning.

The other fields are the catch's: total is added up for the strata that are one row of the CAR (the format has no metier6, so a stratum the CEF splits further is added up), and so are the counts of the sampling - PSU, numPSUs and numTrips, NA for the raised discards, which have no sample of their own - as in `catchCombined()`, whose totals the CAR adds up to. The format asks for them on every imported discard (and BMS) stratum, and a stratum without them is counted in a warning. variance is left empty. vesselFlagCountry is the ISO 3166 code the format wants: a country the converted InterCatch file name ("Belgium", "UK(Scotland)") is written as its code (BE, GB-SCT), and one with no code is warned about and kept.

The CAS:

One row for each donor discard stratum of each group: rate, the discards over the landings of the donor (the discard ratio, which can exceed 1), and weightingTotal, the landings that

weight it. This is how `catchRaising()` combines them: the ratio of a group is the mean of the donors' ratios weighted by their landings - averagingMethod "Weighted", weightingCategory "Lan" and weightingParameter the variable of the raising ("WeightLive") - and so $\text{sum}(\text{rate} * \text{weightingTotal}) / \text{sum}(\text{weightingTotal})$ of a group is the ratio the raised discards of the group were made with. That is checked: a group whose donors do not give the ratio its raised discards have is warned about - the conditions are not those of the raising, or the donors are not all in the catch given. The CAS strata are rows of the CAR (imported = "Y", Dis, with the group in its source), by the country, year, season, area and fleet.

The header:

workingGroupYear, version, versionDescription, calendarYear and startDateYear are those of `catchCombined()` - give them the same values - and are written on every row. catchAllocationSchemeName is the name of the scheme the discards were allocated by; it is the versionDescription when not given.

Value

A list of two tibbles, with the fields of the SCF in their order: CAR (recordType, workingGroupYear, version, versionDescription, catchAllocationSchemeName, catchAllocationGroupNameTarget, catchAllocationGroupNameSource, imported, vesselFlagCountry, calendarYear, startDateYear, year, workingGroup, stock, speciesCode, catchCategory, seasonType, seasonValue, areaType, areaValue, fisheriesManagementUnit, fleetType, fleetValue, variableType, variableUnit, total, variance, PSU, numPSUs, numTrips, comment) and CAS (recordType, workingGroupYear, version, versionDescription, catchAllocationSchemeName, catchAllocationGroupName, vesselFlagCountry, calendarYear, startDateYear, year, workingGroup, stock, speciesCode, catchCategory, seasonType, seasonValue, areaType, areaValue, fisheriesManagementUnit, fleetType, fleetValue, rate, averagingMethod, weightingCategory, weightingParameter, weightingTotal, comment). A code the SCF does not accept is warned about.

See Also

`catchRaising()` for the raising, `recommend_groups()` for groups to give it, `catchCombined()` for the catch added up.

Other stock coordination format: `catchCombined()`, `distributionsCombined()`

Examples

```
## Not run:
strataCond <- list(TBB = rlang::quo(metier4 == "TBB"), REST = rlang::quo(metier4 != "TBB"))
raised <- catchRaising(catch, strataCond, logFile = tempfile(fileext = ".txt"))

alloc <- catchAllocation(raised, strataCond,
                        workingGroupYear = 2026, version = 1,
                        versionDescription = "WGNSSK 2026, first run")

alloc$CAR
alloc$CAS

# the discards of every group can be read back from its donors
with(alloc$CAS[alloc$CAS$catchAllocationGroupName == "TBB", ],
     sum(rate * weightingTotal) / sum(weightingTotal))
```

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

catchAllocationAoL	<i>Allocate length or age distributions to the catch</i>
--------------------	--

Description

Gives every stratum of the catch a length (or age) distribution. Where the catch of a stratum has a sampled distribution it is kept; where it has none, one is estimated from the sampled distributions of a group of matching strata and marked as estimated. The result holds, for each stratum, the numbers and the mean weight at every length or age class - what [catchAtAoLAggregation\(\)](#) and [meanWolAtAoLAggregation\(\)](#) add up into catch-at-length / catch-at-age tables, and what [plot_NaLoA\(\)](#) and [plot_MWLoA\(\)](#) draw.

Usage

```
catchAllocationAoL(
  catch_data,
  distribution_data,
  condition_alloc_st_list,
  condition_matched_data_list = condition_alloc_st_list,
  distributionType = c("Age", "LengthTotal"),
  originType_catch = c("WGEstimate", "Official"),
  variableType_catch = c("WeightLive", "Number"),
  variableType_mean = c("WeightLive"),
  weighting = c("NumberAtAoL", "CATON"),
  verbose = TRUE,
  assembled_output = TRUE,
  append = FALSE,
  ...
)
```

Arguments

catch_data	The catch, a CEF long-format table, with the stratum columns the conditions refer to (see derive_fields()).
distribution_data	The CEF distributions table: distributionType, distributionClass, variableType ("Number" or "WeightLive"), value, domainBiology and the keys above.
condition_alloc_st_list	The conditions that select the strata that are allocated together, one per group - see <i>Groups</i> .
condition_matched_data_list	The conditions that select the matched data of each group; same length and names as condition_alloc_st_list. Defaults to the allocation strata themselves.

distributionType	"LengthTotal", "Age" or both (the default): which distributions are allocated. These are CEF's own values.
originType_catch	"WGEstimate" (default) or "Official": the origin of the catch rows that are allocated. One at a time.
variableType_catch	"WeightLive" (default) or "Number": the variable of the catch rows that are allocated. One at a time.
variableType_mean	The variable whose mean at length or age is allocated; only "WeightLive".
weighting	How the mean weights at length or age are weighted: "NumberAtAoL" (default), by the numbers at that class, or "CATON", by the catch.
verbose	Report what is done, group by group.
assembled_output	TRUE (default): add the rows that were not allocated to the result; FALSE: only the allocated rows.
append	Whether to append to the log file (FALSE: overwrite it).
...	Passed on to <code>check_group_conditions()</code> , e.g. <code>logFile</code> , where the check of the conditions is written.

Details

Groups:

The allocation is done group by group. A group is a pair of conditions, one from `condition_alloc_st_list` and one from `condition_matched_data_list`: the first selects the catch rows that are allocated together, the second the catch rows whose sampled distributions are pooled to estimate the distribution of those that have none. The two lists have the same length and the same names - a group's name, GA1, GA2, ... when the lists have none - and each condition is written as in `dplyr::filter()`: a quosure (`rlang::quo(metier4 == "TBB")`), a text that is turned into one, or a logical vector with a value for every row of `catch_data`.

The allocation strata must not overlap - a catch row in two of them is an error, which `check_group_conditions()` reports before anything is allocated. The matched data may overlap, and usually do: a group can draw on a wider pool than its own strata. Every group is allocated within one year and one stock.

How a distribution is estimated:

The numbers at length (or age) of a stratum without a sampled distribution are the pooled sampled numbers of the matched data, scaled by the weight of the stratum's catch over the weight of the catch that was sampled. Its mean weight at length (or age) is the mean of the sampled mean weights, weighted by the numbers at that class (`weighting = "NumberAtAoL"`) or by the catch ("CATON"). A stratum that has a sampled distribution keeps it as it is.

Catch and distributions are linked on `vesselFlagCountry`, `year`, `workingGroup`, `stock`, `speciesCode`, `catchCategory` and `domainBiology`. Units are harmonised first - catch weight to kg and thousands, distribution values to g and numbers (N) - so what comes back is in those.

What comes back:

A list of two tables:

- **distribution:** every distribution row - the sampled ones as they came, and the estimated ones - with `sampledOrEstimated("sampled" or "estimated")` and `allocGroup`, the group they were allocated in. Numbers are `variableType == "Number"` rows, mean weights `"WeightLive"` rows.
- **catch:** the allocated catch rows, with `allocGroup`. A stratum without a `domainBiology` is given one (`"[alloc]_<season>_<area>_<unit>_<fleet>_<category>"`), and its estimated distributions carry it, so the two tables join on `domainBiology` - which is why anything that reads an allocation takes both tables from this result, not the catch that went in.

With `assembled_output = TRUE` the rows that were not allocated (another origin or variable type, or in no group) are added, with `allocGroup NA`.

Value

A list with `distribution` and `catch`, see *What comes back*.

Author(s)

Yves Reecht

See Also

`catchAtAoLAggregation()` and `meanWoLatAoLAggregation()` to add the allocation up; `check_group_conditions()` to check the groups; `catchRaising()` for the discards.

Examples

```
## Not run:
# one group: every stratum, matched against every stratum
alloc <- catchAllocationAoL(catch, distributions,
                           condition_alloc_st_list = list(ALL = rlang::quo(TRUE)),
                           distributionType = "LengthTotal",
                           logFile = tempfile(fileext = ".txt"))
table(alloc$distribution$sampledOrEstimated)

# two groups: beam trawls are filled from beam trawls, the rest from every stratum
alloc <- catchAllocationAoL(catch, distributions,
                           condition_alloc_st_list = list(TBB = rlang::quo(metier4 == "TBB"),
                                                            REST = rlang::quo(metier4 != "TBB")),
                           condition_matched_data_list = list(TBB = rlang::quo(metier4 == "TBB"),
                                                                REST = rlang::quo(TRUE)),
                           distributionType = "LengthTotal", logFile = tempfile(fileext = ".txt"))

## End(Not run)
```

catchAtAoLAggregation *Catch at length or age: the numbers of an allocation, added up*

Description

Adds the numbers at length (or age) of an allocation - or of any CEF distributions linked to a catch - up into a table with one column per length or age class, for each combination of the variables in grouping: the catch at length, or at age.

Usage

```
catchAtAoLAggregation(
  distribution_data,
  catch_data,
  grouping = c("catchCategory", "attributeType", "attributeValue"),
  originType_catch = c("WGEstimate", "Official"),
  variableType_catch = c("WeightLive", "Number"),
  distributionType = "Age",
  unit = NULL,
  round = NULL,
  minAoL = 0,
  maxAoL = NA,
  plusGroup = TRUE
)
```

Arguments

distribution_data	The distributions: a CEF distributions table, or the distribution table of a catchAllocationAoL() result.
catch_data	The catch they belong to: a CEF catch table, the catch table of the same allocation, or the raised catch the allocation was run on. A distribution and a catch row are linked by domainBiology. The strata an allocation estimated a distribution for have none in the raised catch: the allocation made one up for the distribution and for its own catch, "[alloc]_<season>_<area>_<unit>_<fleet>_<category>", and a catch row with no domain is given the same, so the raised catch finds the estimated distributions as the catch of the same allocation does. A catch weight with no distribution, or a distribution with fish and no catch weight, is warned about.
grouping	The variables to add up by: columns of the catch rows (catchCategory, seasonValue, metier4, ...) or of the distributions (attributeType, attributeValue).
originType_catch	"WGEstimate" or "Official": the origin of the catch rows to link. Pass one.
variableType_catch	"WeightLive" or "Number": the variable of the catch rows to link. Pass one.


```
# catch at length, in thousands of fish, by catch category and season,
# with every class from 400 up as one plus group
catchAtAoLAggregation(alloc$distribution, alloc$catch,
                      grouping = c("catchCategory", "seasonValue"),
                      originType_catch = "WGEstimate",
                      variableType_catch = "WeightLive",
                      distributionType = "LengthTotal",
                      unit = "NE3", round = 0, maxAoL = 400)

## End(Not run)
```

catchCombined	<i>Catch Combined (CC): the catch of a stock, added up, in the format of the stock coordination</i>
---------------	---

Description

The catch of a stock, the way the stock coordination format (SCF v4) wants it: one row per year, working group, stock and catch category - and per variable and unit - with the catch added up. Give it the result of `catchRaising()`; every column the format has and the catch does not fill is there too, NA, and every column the format does not have is gone.

Usage

```
catchCombined(
  catch_data,
  grouping = "catchCategory",
  originType_catch = c("WGEstimate", "Official"),
  variableType_catch = c("WeightLive", "Number"),
  unit = NULL,
  round = NULL,
  workingGroupYear,
  version = 1,
  versionDescription = NA_character_,
  calendarYear = "Y",
  startDateYear = NA_character_
)
```

Arguments

catch_data	The catch to add up: the result of <code>catchRaising()</code> , or a CEF catch table.
grouping	The variables to add up by, on top of the year, the working group and the stock: "catchCategory" (default) and, if wanted, "seasonValue", "areaValue", "fleetValue" (with seasonType and areaType, which go with them).
originType_catch	"WGEstimate" or "Official": the origin of the catch rows to add up. Pass one.

variableType_catch	"WeightLive" or "Number": the variable of the catch rows to add up; both are reported, each in its own rows.
unit	The unit to convert the catch to before adding it up: "t" or "kg" for weights, "N" or "NE3" for numbers. NULL (default) leaves them as they are.
round	Digits to round the totals to; NULL (default) does not round.
workingGroupYear	The year of the working group the data are processed for, as a four-digit year (2026). Required.
version	The version of the file, a whole number (default 1).
versionDescription	What the version is, a string, or NA (default).
calendarYear	"Y" (default) or "N".
startDateYear	With calendarYear = "N", the start of the model year as "MM-DD"; left out otherwise.

Details

The rows are added up by year, workingGroup, stock, the variables in grouping and the catch's variableType and variableUnit. grouping always holds the catch category, and takes the other columns of the SCF that a stock can be split by - seasonValue, areaValue and fleetValue; a season or an area is reported with its type. Usually the catch is not split, which is the default; ask for more when the stock assessment is. To report the catch by fleets of your own, put their names in fleetValue first.

Take one originType_catch: the default takes the catch rows of both origins, and a stratum that has rows of both is counted twice (a warning says so). "WGEstimate" is the working group's own catch - the raised discards among it. A catch of NaN (a stratum the raising had no rate for) adds up as nothing. The rows are added up whatever their domainBiology: the raised catch, and the catch of the allocation made from it (which has the domain the allocation made up for the strata it estimated), give the same CC.

The distributions of the same stock go with it: use the same grouping in `distributionsCombined()`, since each distribution needs the catch it belongs to in the CC.

The header:

workingGroupYear is the year of the working group the data are processed for - the 2026 of a WGNSSK 2026 - and has no default, since it is not in the data. version and versionDescription name the run, so more than one can be kept (a benchmark, a sensitivity test). calendarYear is "Y" when year is the calendar year, which it is for most stocks, and "N" when it is a model year that starts on another date, startDateYear ("07-01"), the same for every row of the stock. They are written on every row; every other field comes from the catch.

The sampling:

PSU, numPSUs and numTrips say how much sampling stands behind the total. They are taken from the columns of the catch with the same names, as the CEF has them: the counts of the rows added up are summed, and are NA when none of the rows reports one (nothing was counted, which is not 0), and PSU, the type of the sampling unit ("FishingTrip"), is that of the rows that report a number of them - a ; separated list when there are several types. A row the raising only estimated

(importedOrRaised is "raised") has no sample of its own, its rate was borrowed from a group, so whatever count it carries is left out: the format asks for the strata that actually contributed to the estimate. variance is left empty.

Value

A tibble with the CC fields of the SCF: recordType, workingGroupYear, version, versionDescription, calendarYear, startDateYear, year, workingGroup, stock, catchCategory, seasonType, seasonValue, areaType, areaValue, fleetValue, variableType, variableUnit, total, variance, PSU, numPSUs and numTrips. A code the SCF does not accept is warned about.

See Also

[distributionsCombined\(\)](#) for the distributions that go with it; [catchRaising\(\)](#) for the catch it is made of.

Other stock coordination format: [catchAllocation\(\)](#), [distributionsCombined\(\)](#)

Examples

```
## Not run:
raised <- catchRaising(catch, strataCond, matchedDataCond,
  logFile = tempfile(fileext = ".txt"))

# the catch of the stock by catch category, in tonnes
catchCombined(raised, originType_catch = "WGEstimate",
  variableType_catch = "WeightLive", unit = "t",
  workingGroupYear = 2026,
  version = 1, versionDescription = "WGNSSK 2026, first run")

# by quarter as well
catchCombined(raised, grouping = c("catchCategory", "seasonValue"),
  originType_catch = "WGEstimate", unit = "t",
  workingGroupYear = 2026)

# a model year that starts on 1 July
catchCombined(raised, originType_catch = "WGEstimate", unit = "t",
  workingGroupYear = 2026,
  calendarYear = "N", startDateYear = "07-01")

## End(Not run)
```

catchRaising

Raise the discards of each group from its donor pool

Description

Gives the landings that have no discard estimate a raised one, group by group. The discard ratio comes from a donor pool of matched data that can be wider than the group itself. The result is the catch with the raised discards added, every row flagged "imported" or "raised" and tagged with its group.

Usage

```

catchRaising(
  catch_data,
  condition_raising_st_list,
  condition_matched_data_list = condition_raising_st_list,
  originType = c("WGEstimate", "Official"),
  variableType = c("WeightLive", "Number"),
  type = c("discards", "BMS"),
  verbose = TRUE,
  assembled_output = TRUE,
  append = FALSE,
  ...
)

```

Arguments

catch_data	The catch, a CEF long-format table, with the columns the conditions refer to (see derive_fields()).
condition_raising_st_list	The conditions that select the raising strata, one per group - see <i>Groups</i> .
condition_matched_data_list	The conditions that select the matched data of each group; same length and names as condition_raising_st_list. Defaults to the raising strata themselves.
originType	"WGEstimate" (default) or "Official": the origin of the rows that are raised. One at a time.
variableType	"WeightLive" (default) or "Number": the variable of the rows that are raised. One at a time.
type	"discards" (default) or "BMS". Only discards are implemented; "BMS" stops with an error.
verbose	Print messages while raising.
assembled_output	TRUE (default): the raised discards together with the imported rows of the raising strata; FALSE: only the raised discards.
append	Whether to append to the log file (FALSE: overwrite it).
...	Passed on to check_group_conditions() , e.g. logFile, where the check of the conditions is written.

Details**Groups:**

A group is a pair of conditions, one from condition_raising_st_list and one from condition_matched_data_list: the first selects the catch rows that are raised together (the raising stratum), the second the rows whose landings and discards give the ratio (the donor pool, or matched data). The two lists have the same length and the same names - a group's name, G1, G2, ... when the lists have none - and

each condition is written as in `dplyr::filter()`: a quosure (`rlang::quo(metier4 == "TBB")`), a text that is turned into one, or a logical vector with a value for every row of `catch_data`.

The raising strata must not overlap - a catch row in two of them is an error, which `check_group_conditions()` reports before anything is raised. The matched data may overlap, and usually do: a group can draw on a wider pool than its own strata. Every group is raised within one year and one stock.

What is raised:

In a raising stratum the landings that have no discard estimate are the ones raised: those whose `domainCatchDis` has no discards row of the chosen `originType` and `variableType`. Each gets a discards row of its landings times the group's discard ratio. The ratio is the mean of discards over landings of the matched data, weighted by the landings - the total discards over the total landings of the matched strata that have a discard estimate.

A raised row has `catchCategory` "Dis" and `importedOrRaised` "raised"; its `domainCatchDis` is the landings' own, or "[raised]_<season>_<area>_<management unit>_<fleet>" when they had none, and `domainBiology` and `domainCatchBMS` are NA. Every other row of the stratum - landings, the discards that were reported, BMS - is kept as it is and flagged "imported".

`total` is converted first, to kg for weights and to thousands (NE3) for numbers. `catch_data` is a CEF long-format table whose columns run in CEF order - `vesselFlagCountry`, `year`, `workingGroup`, `stock`, `speciesCode`, `catchCategory` one after another - as they do in a table read from CEF files.

Value

A tibble with the columns of `catch_data` plus `importedOrRaised` ("imported" or "raised") and `DrGroup`, the name of the group, on every row. Rows of `catch_data` that fall in no raising stratum are not part of it.

Author(s)

Yves Reecht

See Also

`check_group_conditions()` to check the groups before raising; `plot_catches()`, `plot_DR()` and `raised_overview_table()` to look at the result; `catchAllocationAoL()` for the lengths and ages.

Examples

```
## Not run:
# one group per fleet, each raised from its own discard ratio
strataCond <- list(TBB = rlang::quo(metier4 == "TBB"),
                  OTB = rlang::quo(metier4 == "OTB"))
raised <- catchRaising(catch, strataCond, verbose = FALSE,
                      logFile = tempfile(fileext = ".txt"))
table(raised$DrGroup, raised$importedOrRaised)

# the same groups, both drawing their ratio on every beam and otter trawl
donors <- rlang::quo(metier4 %in% c("TBB", "OTB"))
raised <- catchRaising(catch, strataCond,
```

```

condition_matched_data_list = list(TBB = donors, OTB = donors),
verbose = FALSE, logFile = tempfile(fileext = ".txt"))

## End(Not run)

```

check_group_conditions

Function to check groups consistency

Description

Function to check groups consistency.

Usage

```

check_group_conditions(
  catch_data,
  condition_list,
  conditionType = c("strata", "matched_data"),
  domain = c("domainCatchDis", "domainCatchBMS", "domainBiology"),
  originType = c("unspecified", "WGEstimate", "Official"),
  variableType = c("unspecified", "WeightLive", "Number"),
  logFile = NULL,
  append = FALSE
)

```

Arguments

catch_data	census (or assembled census+estimated+raised) data.
condition_list	A (named) list of conditions, with one condition per group. Supported formats are logical (same length as number of rows in census data), expressions, quosures, calls. A character strings describing the condition (as in filter(...), but with quotes) will be tentatively converted to quosure.
conditionType	raising/allocation "strata" or "matched_data" (the latter is allowed to have overlap among groups).
domain	The domain, used to match records with extra info.
originType	Origin of the records the groups are for: "unspecified" (default), "WGEstimate" or "Official". Not used for now: the groups are checked on every row of catch_data, whatever its origin.
variableType	Variable type (for info in the log-file only).
logFile	Log-file path.
append	Whether to append to the log file (FALSE: override it).

Value

Group conditions as a named list of logicals (1 per group).

Author(s)

Yves Reecht

coverage-plots*Coverage: which strata are sampled well enough*

Description

Three plots of how well a set of strata is sampled: a grid of cells drawn as a heatmap or as weighted bubbles (`plot_coverage_grid()`, `type = "heat"` or `"bubble"`) and a sampled-versus-unsampled weight bar chart (`plot_coverage_bar()`), both documented here, plus a map (`plot_coverage_map()`) on its own page.

All take the CEF catch (and distributions for length and age) or the table `build_coverage_strata()` returns from it. Given the catch they build that table at exactly the plotted columns; given the table they pool it to them. So the numbers behind a figure can always be read as a table, at the same aggregation. The plotted columns are read off the catch as it is - `derive_fields()` run once on the raw data, and any column of your own - and nothing is derived again.

Usage

```
plot_coverage_grid(  
  catch,  
  distributions = NULL,  
  y,  
  x = NULL,  
  facet = NULL,  
  what = "discard",  
  catchCategory = NULL,  
  type = c("heat", "bubble"),  
  thresholds = list(cov = 50, trips = NA, fish = NA),  
  palette = c("default", "colorblind", "gradual"),  
  save = NULL  
)  
  
plot_coverage_bar(  
  catch,  
  distributions = NULL,  
  y,  
  fill = NULL,  
  facet = NULL,  
  what = "discard",  
  catchCategory = NULL,  
  palette = c("default", "colorblind", "gradual"),  
  save = NULL  
)
```

Arguments

catch	CEF catch data carrying the columns you plot by (derive_fields() output, or columns of your own), or a table from build_coverage_strata() .
distributions	A CEF distributions table - needed for what = "length" / "age" with raw catch, ignored otherwise.
y	Row category: a column of the catch (or of the table). Required.
x	Column category, the second axis of the grid. NULL collapses to a single column.
facet	Optional facet column.
what	"discard" (default), "length" or "age". Taken from the table when catch is a coverage table.
catchCategory	For what = "length" / "age": "All" (default), "Lan", "Dis" or "BMS". Sets which catch part the distribution is linked to and the weight word in the plot title. With a coverage table it selects among the parts the table has. Not valid for what = "discard" (an error).
type	"heat" (default) or "bubble" (grid only): the cells as tiles that carry their weight as a number, or as bubbles sized by it. The same cells, status, legend, thresholds box and labels either way.
thresholds	A list (grid): cov, the minimum coverage in percent (default 50; always in use, coverage is never dropped from the classification); trips, the minimum number of sampling trips; fish, the minimum number of measurements for what = "length" / "age". trips and fish default to NA (or NULL): that question is left out of the classification altogether. trips = 0 is not the same as NA: it still checks trips, and a stratum with no reported trip count still fails that check. fish is not used with what = "discard" (a warning: no fish are measured for a discard estimate). Leave out of the list what you do not set; recommend_groups() takes the same list.
palette	"default", "colorblind" or "gradual". Recolours the bar fill, or the status colours of the heatmap and the bubbles.
save	Where to write the plot as well, or NULL (default) to only return it: a folder, a file name, or a list of options - see saving-plots .
fill	Bar fill category (bars only). NULL uses y.

Details**What "sampled" means:**

what chooses the question:

- "discard" (default): does the stratum have a discard estimate? A landing has one when a discard row with a weight carries its domainCatchDis (same country, origin and variable type), as [catchRaising\(\)](#) pairs them.
- "length" / "age": is a length / age distribution linked to the stratum's catch? Needs distributions. catchCategory chooses the catch part - "Lan", "Dis", "BMS", or "All" (the default: landings and discards together, titled "catch").

The plural and mixed case are accepted ("discards", "Length"). A coverage table brings its own what and catch parts, so with one neither needs to be given; a what that contradicts the table is an error.

The status scale (heatmap and bubbles):

The colour of a cell is its sampling status against the thresholds:

Colour (default palette)	Status
dark green	100% covered (and, when trips are used, more than <code>thresholds\$trips</code> trips)
green	covered at least <code>thresholds\$cov</code> , and passing every threshold in use
yellow	one threshold in use missed
orange	two thresholds in use missed
dark orange	three missed: only for length / age, with both <code>trips</code> and <code>fish</code> in use
red	no discard estimate / no linked distribution at all - never a threshold miss
blue	discards only, no landings (discard coverage)

The other palettes keep the same scale, from best to worst sampled.

A cell with no weight at all draws nothing. The legend always lists every reachable status, including those no cell falls in, so the same figure drawn on two subsets has the same key.

The thresholds are one list, `thresholds = list(cov = 50, trips = NA, fish = NA)`

- the list `recommend_groups()` takes, so the figure and the groups made from it are judged by the same values. Coverage (`cov`, in percent) is always in use. `trips` (discard, length, age) and `fish` (length, age) default to `NA` and are each optional: `NA` (or `NULL`) drops that question from the classification and the legend, a number turns it on; leave out of the list what you do not set. The label says which threshold(s) a cell missed, so "< 50% covered" and ">= 50% covered, < 2 trips" are both yellow but read differently.

`trips = 0` is not the same as `NA`: it is a threshold in use (if a lenient one), and a stratum with no reported trip count fails a `trips` test however low it is set - unknown is not treated as adequate. A fully covered stratum with no trip count reads "safe" at `trips = NA` but stays a warning at `trips = 0`.

`fish` counts the fish measured for a length or age distribution, which a discard estimate has none of: with `what = "discard"` it is not used (a warning), so one list can be given to the discards and to the lengths alike.

What the title and caption say:

The title names the weight a figure is judged against first, then what is sampled: "Landings - Sampling of Length", "Discards - Sampling of Age". The weight word comes from `catchCategory` (landed, discarded, BMS, or the catch as a whole). Discard coverage drops it entirely - "Sampling of Discards" - since it is always judged against the landings, so naming that weight would only repeat itself.

The caption's first line says what a stratum is here: the plotted columns, grouped over every other stratum column ("`grouping = metier4 x vesselFlagCountry x seasonValue`, grouped over finer fleet levels (`fleetValue`) and area (`areaValue`)"), so the same figure drawn at other columns has other strata behind it. The rest of the caption spells out how the numbers on the figure are worked out - the value in a tile, what a bubble's size or a bar means - leaving out anything a legend already shows. On the heatmap and bubble grid the thresholds in use move into their own boxed guide, bottom right, under the in-panel status legend - a detail section of it, not a caption about the plot itself.

Bars and palettes:

The bar chart shows sampled versus unsampled weight, which no threshold changes, so it takes none: passing thresholds is an error. palette recolours the bars, and the status scale of the heatmap and the bubbles.

Argument order:

What data (catch, distributions, what, catchCategory), then which columns are the strata (y, x or fill, facet), then the thresholds, then palette.

Value

A ggplot object, or NULL when there is nothing to draw.

See Also

[build_coverage_strata\(\)](#) for the table, [plot_coverage_map\(\)](#), [plot_DR\(\)](#).

Other coverage plots: [build_coverage_strata\(\)](#), [plot_coverage_map\(\)](#)

Examples

```
## Not run:

# discard coverage: which metier x country strata carry a discard estimate?
plot_coverage_grid(catch, y = "metier4", x = "vesselFlagCountry")
plot_coverage_grid(catch, y = "metier4", x = "vesselFlagCountry",
                    type = "bubble", thresholds = list(cov = 50, trips = 2))
plot_coverage_bar(catch, y = "metier4", fill = "vesselFlagCountry")

# length coverage of the landings, with a floor on the measurements
plot_coverage_grid(catch, distributions, y = "metier4",
                    x = "vesselFlagCountry", what = "length", catchCategory = "Lan",
                    thresholds = list(cov = 50, fish = 100))

# the same figure from the table behind it
cov <- build_coverage_strata(catch, c("metier4", "vesselFlagCountry"))
plot_coverage_grid(cov, y = "metier4", x = "vesselFlagCountry",
                    thresholds = list(cov = 50))

## End(Not run)
```

derive_fields

Derive the grouping columns from a CEF catch table - or from its distributions

Description

Adds the columns an assessment groups by, none of which the CEF format carries directly: the nested metier levels ('metier4' to 'metier6'), 'target' and 'mesh' from 'fleetValue'; 'metier4_mesh', the gear and the mesh joined ("TBB_70-99"; just the gear when the fleet code has no mesh part); the three ICES area levels from 'areaValue'; and 'Stratum', the four native keys joined. Country and season are not among them - 'vesselFlagCountry' and 'seasonValue' already name them, so a 'Country' / 'Season' alias would only be a second name for the same column.

Usage

```
derive_fields(
  catch_data,
  distribution_data = NULL,
  what = c("catch", "distribution")
)
```

Arguments

catch_data	A CEF catch table (data frame).
distribution_data	A CEF distributions table. Only with 'what = "distribution"', which needs it.
what	"catch" (default): 'catch_data' with the derived columns. "distribution": 'distribution_data' with them.

Details

Run it once, on the raw CEF data, and build on the result. The coverage table and the plots read the columns your catch has and derive nothing themselves, so a column you add (a fleet grouping of your own, say) or regroup (your own 'metier4') is used as it stands. A column the catch already has is left as it is - an edit of yours, or a column from an earlier call - so calling it again, or deriving the distributions from a catch you have edited, never undoes the edit; drop a column first to have it derived afresh.

An empty field of a CEF csv arrives as "" in a text column when the file is read with 'read.csv()' or 'read.taf()' (and as 'NA' when it is read with readr). Empty text is missing in the CEF format, so it comes back as 'NA' - on the catch and, with 'what = "distribution"', on the distributions - and the raising, the allocation and the link between catch and distributions all see the same missing id. Left as "", every catch row with no 'domainBiology' would share the domain "" and be taken for one distribution.

An ICES area code is a nested path ('27.4.b.1'), so it is split into subarea, division and subdivision. A code already coarser than the level asked for REPEATS rather than becoming 'NA': '27.4' is its own division and subdivision as far as the data goes, so a stratum reported at that resolution is never silently dropped from a finer grouping.

With 'what = "distribution"' the same fields come back on the distributions table instead: 'derive_fields(catch_data, distribution_data, what = "distribution")'. 'derive_fields(catch_data)' is the catch, as it always was.

A distribution carries none of the stratum columns - the fleet, the area and the season belong to the catch - so deriving its fields needs both tables. Each distribution row is linked to the catch rows it describes through 'domainBiology', together with the catch category and every other column both tables carry among the ones the allocation itself matches on ('vesselFlagCountry', 'year', 'workingGroup', 'stock', 'speciesCode': a 'domainBiology' names no country, so two countries can use the same one). The row takes the 'fleetValue', 'areaValue' and 'seasonValue' of those catch rows and every field derived from them. The derivation is the one described above, run once on the catch: the distributions only look its results up, so the two tables can never disagree about what a field means.

A 'domainBiology' can be used by catch rows of several strata (a distribution covers every catch row that names it). The fields those rows share are kept and the ones they differ on are 'NA',

with a message, rather than one of them being picked at random. A distribution row with no catch row behind it gets 'NA' throughout, with a warning: the two files disagree. The rows of 'distribution_data' come back as they went in (empty text apart), same number and same order; columns of the same name already on it (an earlier call, say) are replaced.

Value

With 'what = "catch"', 'catch_data' with the derived columns added. With 'what = "distribution"', 'distribution_data' with 'fleetValue', 'areaValue', 'seasonValue' (and 'vesselFlagCountry', when the distributions have none of their own) and the derived columns added.

See Also

[build_coverage_strata()]

Examples

```
## Not run:

catch <- derive_fields(catch)
head(catch[, c("fleetValue", "metier4", "metier4_mesh", "ICESDivision")], 3)

# the distributions get the same fields, read off the catch on domainBiology
dist <- derive_fields(catch, distributions, what = "distribution")
head(dist[, c("domainBiology", "fleetValue", "metier4_mesh", "ICESDivision")], 3)

## End(Not run)
```

distributionsCombined *Distributions Combined (DC): the numbers and mean weights at length or age, in the format of the stock coordination*

Description

The length or age distributions of a stock, the way the stock coordination format (SCF v4) wants them: the catch's numbers at length (or age) and the mean weight at length (or age), one row per class, added up by year, working group, stock and catch category - and whatever else grouping adds. Give it the two tables of [catchAllocationAoL\(\)](#); every column the format has and the distributions do not fill is there too, NA.

Usage

```
distributionsCombined(
  distribution_data,
  catch_data,
  grouping = c("catchCategory", "attributeType", "attributeValue"),
  weighting = c("NumberAtAoL", "CATON"),
  originType_catch = c("WGEstimate", "Official"),
```

```

variableType_catch = c("WeightLive", "Number"),
distributionType = "Age",
unit = c("NE3", "g"),
round = NULL,
minAoL = 0,
maxAoL = NA,
plusGroup = TRUE,
workingGroupYear,
version = 1,
versionDescription = NA_character_,
calendarYear = "Y",
startDateYear = NA_character_
)

```

Arguments

distribution_data	The distributions: a CEF distributions table, or the distribution table of a catchAllocationAoL() result.
catch_data	The catch they belong to: a CEF catch table, the catch table of the same allocation, or the raised catch the allocation was run on. A distribution and a catch row are linked by domainBiology. The strata an allocation estimated a distribution for have none in the raised catch: the allocation made one up for the distribution and for its own catch, "[alloc]_<season>_<area>_<unit>_<fleet>_<category>", and a catch row with no domain is given the same, so the raised catch finds the estimated distributions as the catch of the same allocation does. A catch weight with no distribution, or a distribution with fish and no catch weight, is warned about.
grouping	The variables to add up by, on top of the year, the working group, the stock and the catch category (always reported by): the sex of the distribution - "attributeType" and "attributeValue", the default - and, as in catchCombined() , "seasonValue", "areaValue" and "fleetValue" (with seasonType and areaType, which go with them).
weighting	How the mean weights are weighted: "NumberAtAoL" (default), by the numbers at length or age, or "CATON", by the weight of the catch row a distribution belongs to. See meanWoLatAoLAggregation() .
originType_catch	"WGEstimate" or "Official": the origin of the catch rows to link. Pass one.
variableType_catch	"WeightLive" or "Number": the variable of the catch rows to link. Pass one.
distributionType	"Age" (default), "LengthTotal" or both.
unit	The units to convert to before adding up: the numbers in "N" or "NE3" (thousands), the mean weights in "g" or "kg". Default c("NE3", "g"); NULL leaves them as they are.
round	Digits to round the numbers and the mean weights to; NULL (default) does not round.

minAoL, maxAoL, plusGroup	As in <code>catchAtAoLAggregation()</code> ; named by distribution type when there are several.
workingGroupYear	The year of the working group the data are processed for, as a four-digit year (2026). Required.
version	The version of the file, a whole number (default 1).
versionDescription	What the version is, a string, or NA (default).
calendarYear	"Y" (default) or "N".
startDateYear	With <code>calendarYear = "N"</code> , the start of the model year as "MM-DD"; left out otherwise.

Details

This is what `catchAtAoLAggregation()` (the numbers) and `meanWoLatAoLAggregation()` (the mean weights) add up - they are given the same arguments here, and add up the same way - in the long format of the SCF instead of one column per class: a Number row (valueType Total) and a WeightLive row (valueType Mean) for each class. A class no fish fall in is left out, since it has no mean weight.

The distributions are linked to the catch through domainBiology, the catch category and the country, and the grouping is read off the catch they are linked to. A sampled distribution has the domain of its catch. The strata an allocation estimated a distribution for have none in the raised catch, and the allocation made one up for the distribution and for the catch it returns, "[alloc]_<season>_<area>_<unit>_<fleet>_<catch>". A catch row with no domain is given the same, so every stratum - raised or not, sampled or allocated - finds its distribution whether the raised catch the allocation was run on or the catch of its result is handed in (`distributionsCombined(alloc$distribution, alloc$catch, ...)`). It has to be that raised catch: another raising has other strata and weights. A catch weight with no distribution, or a distribution with fish and no catch weight, is warned about. Take one `originType_catch` and one `variableType_catch` too, as in `catchAtAoLAggregation()`.

The grouping is the one of `catchCombined()` (and of the same file: a distribution needs a catch of the same year, stock, category, season, area and fleet in the CC), with the sex of the distribution on top - `attributeType` and `attributeValue`, the default - which is NA when the distributions do not have one. The header - `workingGroupYear` and the rest - is the one of `catchCombined()` too, and so is the sampling's reading.

The sampling:

PSU, numPSUs, numTrips and numMeasurements say how much sampling stands behind each class. They are taken from the columns of the distributions with the same names, as the CEF has them. A row of the table is made of the distributions of the catch it is added up from: each counts once, however many classes it has (the CEF repeats its counts on every class, and a plus group puts several of them in one row), and the counts of the distributions are summed. They are NA when none of the distributions reports one - the allocation leaves them empty for the distributions it estimated, which were borrowed: the format asks for the strata that actually contributed

- and PSU, the type of the sampling unit, is that of the distributions that report a number of them, a ; separated list when there are several types. `numMeasurements` is the count of the whole distribution, which is all the CEF has, though the format asks for the measurements at the class: do not sum it over the classes. `variance` is left empty.

Classes:

Lengths are reported in mm, in the class the distributions give. minAoL drops the classes below it; maxAoL and plusGroup make a plus group of the ages from maxAoL up, written ageGroupPlus on every row of the distribution. The SCF has no plus group for lengths: with them maxAoL only cuts the classes off (plusGroup = FALSE). distributionType can be several (c("Age", "LengthTotal")), and minAoL, maxAoL and plusGroup then one value for each, named: maxAoL = c(Age = 10).

Value

A tibble with the DC fields of the SCF: recordType, workingGroupYear, version, versionDescription, calendarYear, startDateYear, year, workingGroup, stock, catchCategory, seasonType, seasonValue, areaType, areaValue, fleetValue, distributionType, distributionUnit, distributionClass, ageGroupPlus, attributeType, attributeValue, variableType, variableUnit, valueType, value, variance, PSU, numPSUs, numTrips and numMeasurements.

See Also

[catchCombined\(\)](#) for the catch that goes with it; [catchAllocationAoL\(\)](#) for the allocation it is made of; [catchAtAoLAggregation\(\)](#) and [meanWoLatAoLAggregation\(\)](#) for the same numbers and mean weights with one column per class.

Other stock coordination format: [catchAllocation\(\)](#), [catchCombined\(\)](#)

Examples

```
## Not run:
alloc <- catchAllocationAoL(raised, distributions,
                           condition_alloc_st_list = list(ALL = rlang::quo(TRUE)),
                           distributionType = "LengthTotal",
                           logFile = tempfile(fileext = ".txt"))

# numbers in thousands and mean weights in g, by catch category
distributionsCombined(alloc$distribution, alloc$catch,
                     originType_catch = "WGEstimate",
                     variableType_catch = "WeightLive",
                     distributionType = "LengthTotal",
                     workingGroupYear = 2026,
                     version = 1, versionDescription = "WGNSSK 2026, first run")

# ages, with every age from 10 up in a plus group, and the lengths as they are
distributionsCombined(alloc$distribution, alloc$catch,
                     originType_catch = "WGEstimate",
                     variableType_catch = "WeightLive",
                     distributionType = c("Age", "LengthTotal"),
                     maxAoL = c(Age = 10), plusGroup = c(Age = TRUE, LengthTotal = FALSE),
                     workingGroupYear = 2026)

## End(Not run)
```

funICoutCEF

InterCatch output to CEF format

Description

InterCatch output to CEF format

Usage

```
funICoutCEF(
  dat_path,
  years,
  stock_relation = NULL,
  output_format = c("to_environment", "to_file", "to_upload_file"),
  out_path = getwd(),
  keep_temp_file = FALSE,
  file_prefix = "",
  include_BMS_Caton = TRUE,
  patch_missing_DIS_PSU = FALSE,
  metier6 = NULL
)
```

Arguments

dat_path	Path to the folder in which IC data is stored by year
years	year for which IC data will be loaded
stock_relation	The stock / area / species lookup that gives every record its stock code and working group, as made by funMakeRelation(). NULL (default): made here, by funMakeRelation(max(years)).
output_format	What to do with the result, any of "to_environment" (the tables catches, distributions and effort are assigned in the global environment), "to_file" (they are written as csv files in out_path), "to_upload" (they are written as the upload files, without a header) and "to_list" (they are returned, invisibly). The default is the whole vector, which does the first two; it lists "to_upload_file", which the code does not act on.
out_path	Folder the files are written to (default: the working directory).
keep_temp_file	Keep the folder tmp that is made inside dat_path, which holds the IC data as the exchange format files (hi.csv, si.csv and sd.csv) that funInterCatchCEF() reads? Default FALSE: it is removed.
file_prefix	Prefix of the names of the files written (default: none), as in pok_2022_catches_CEF_v17.1.csv.
include_BMS_Caton	Default TRUE. FALSE leaves out the BMS rows of the working group's estimate (Caton); the official BMS landings stay.

patch_missing_DIS_PSU	Default FALSE. TRUE gives a discard domain that has no sampling unit the PSU "FishingTrip" and 0 sampled trips (with a warning).
metier6	NULL (default) writes the metier6 as MIS_MIS_0_0_0. "Fleet" copies the fleet into it and writes the unit k of the numbers as NE3.

funIntercatchCEF	<i>InterCatch exchange format to CEF v17.1</i>
------------------	--

Description

InterCatch exchange format to CEF v17.1

Usage

```
funIntercatchCEF(
  dat_path = getwd(),
  stock_relation = stock_relation,
  output_format = c("to_environment", "to_file", "to_upload_file"),
  out_path = getwd(),
  file_prefix = "",
  file_suffix = paste0("_CEF_v", getOption("CEF_version")),
  include_BMS_Caton = TRUE,
  patch_missing_DIS_PSU = FALSE,
  metier6 = NULL
)
```

Arguments

dat_path	folder with intercatch exchange format files
stock_relation	stock/area/species lookup, as built by funMakeRelation()
output_format	where should the output go: any of "to_environment" (the tables catches, distributions and effort are assigned in the global environment), "to_file" (they are written as csv files in out_path), "to_upload" (they are written as the upload files, without a header) and "to_list" (they are returned). The default is the whole vector, which does the first two; it lists "to_upload_file", which the code does not act on.
out_path	if it should be in files, where
file_prefix	Prefix of the names of the files written (default: none), as in pok_2022_catches_CEF_v17.1.csv.
file_suffix	End of the names of the files written (default: _CEF_v17.1, the version of the format).
include_BMS_Caton	Default TRUE. FALSE leaves out the BMS rows of the working group's estimate (Caton); the official BMS landings stay.

patch_missing_DIS_PSU	Default FALSE. TRUE gives a discard domain that has no sampling unit the PSU "FishingTrip" and 0 sampled trips (with a warning).
metier6	NULL (default) writes the metier6 as MIS_MIS_0_0_0. "Fleet" copies the fleet into it and writes the unit k of the numbers as NE3.

funMakeRelation	<i>Function to create stock_relation from ICES databases</i>
-----------------	--

Description

Function to create stock_relation from ICES databases

Usage

```
funMakeRelation(year)
```

Arguments

year	Year from which to extract the ICES stock databases
------	---

meanWoLatAoLAggregation	<i>Mean weight at length or age: the mean weights of an allocation, averaged</i>
-------------------------	--

Description

Averages the mean weights at length (or age) of an allocation - or of any CEF distributions linked to a catch - into a table with one column per length or age class, for each combination of the variables in grouping: the mean weight at length, or at age.

Usage

```
meanWoLatAoLAggregation(
  distribution_data,
  catch_data,
  grouping = c("catchCategory", "attributeType", "attributeValue"),
  weighting = c("NumberAtAoL", "CATON"),
  originType_catch = c("WGEstimate", "Official"),
  variableType_catch = c("WeightLive", "Number"),
  variableType_dist = "WeightLive",
  distributionType = "Age",
  unit = NULL,
  round = NULL,
```

```

    minAoL = 0,
    maxAoL = NA,
    plusGroup = TRUE,
    samplesOnly = TRUE
  )

```

Arguments

distribution_data	The distributions: a CEF distributions table, or the distribution table of a <code>catchAllocationAoL()</code> result.
catch_data	The catch they belong to: a CEF catch table, the catch table of the same allocation, or the raised catch the allocation was run on. A distribution and a catch row are linked by domainBiology. The strata an allocation estimated a distribution for have none in the raised catch: the allocation made one up for the distribution and for its own catch, "[alloc]_<season>_<area>_<unit>_<fleet>_<category>", and a catch row with no domain is given the same, so the raised catch finds the estimated distributions as the catch of the same allocation does. A catch weight with no distribution, or a distribution with fish and no catch weight, is warned about.
grouping	The variables to add up by: columns of the catch rows (catchCategory, seasonValue, metier4, ...) or of the distributions (attributeType, attributeValue).
weighting	How the mean weights are weighted: "NumberAtAoL" (default), by the numbers at length or age of each distribution, or "CATON", by the total weight of the catch row each distribution belongs to.
originType_catch	"WGEstimate" or "Official": the origin of the catch rows to link. Pass one.
variableType_catch	"WeightLive" or "Number": the variable of the catch rows to link. Pass one.
variableType_dist	The variable of the distributions that is averaged; only "WeightLive".
distributionType	"Age" (default) or "LengthTotal", CEF's own values. One at a time.
unit	The unit to convert the weights to before averaging: "g", "kg" or "t". NULL (default) leaves them as they are.
round	Digits to round the weights to; NULL (default) does not round.
minAoL	The lowest class kept.
maxAoL	The class from which the plus group starts, or the highest class kept without one; NA (default) for the highest class found.
plusGroup	Pool the classes from maxAoL up into a plus group (TRUE, default) or leave them out (FALSE).
samplesOnly	Not used.

Details

The distributions are linked to the catch rows as in [catchAtAoLAggregation\(\)](#), which has the same warning about taking both tables from one allocation and passing a single `originType_catch` and `variableType_catch`. The mean weights of the "WeightLive" distributions are averaged within each group and class, weighted by the numbers at that class (`weighting = "NumberAtAoL"`). The numbers come from the "Number" distributions of the same domain, so a mean weight without numbers is left out of the average.

`minAoL`, `maxAoL` and `plusGroup` cut and pool the classes exactly as they do in [catchAtAoLAggregation\(\)](#).

Value

A tibble with the grouping columns, `unit` (the unit of the weights) and one column per class, named `W_<distributionType>_<class>`: `W_LengthTotal_300`, `W_Age_3`, `W_Age_7+`.

Author(s)

Yves Reecht

See Also

[catchAtAoLAggregation\(\)](#) for the numbers; [distributionsCombined\(\)](#) for the same mean weights, one row per class, in the format of the stock coordination; [catchAllocationAoL\(\)](#) for the allocation; [plot_MWaLoA\(\)](#) to look at it.

Examples

```
## Not run:
alloc <- catchAllocationAoL(catch, distributions,
                           condition_alloc_st_list = list(ALL = rlang::quo(TRUE)),
                           distributionType = "LengthTotal",
                           logFile = tempfile(fileext = ".txt"))

# mean weight at length in kg, by catch category
meanWoLatAoLAggregation(alloc$distribution, alloc$catch,
                        grouping = "catchCategory",
                        originType_catch = "WGEstimate",
                        variableType_catch = "WeightLive",
                        distributionType = "LengthTotal",
                        unit = "kg", round = 3)

## End(Not run)
```

Description

A stacked bar of catch weight at any aggregation: bars are landings, discards (and BMS), or just the one, via `catchCategory`. Given the raised catch (the output of `catchRaising()`) it shows what the raising added - importedOrRaised, imported from the data versus added by the raising - as the mark, as the fill or as a facet, as you choose. Given raw catch it is the weight as it was reported. The arguments are the same either way, and so is the plot; the title says which it is.

Usage

```
plot_catches(
  catch,
  x = NULL,
  facet = NULL,
  fill = NULL,
  alpha = "importedOrRaised",
  catchCategory = NULL,
  ncol = 2,
  palette = c("default", "colorblind", "gradual"),
  save = NULL
)
```

Arguments

<code>catch</code>	CEF catch data, or the raised catch (<code>catchRaising()</code>): <code>catchCategory</code> and <code>total</code> , and for the mark <code>importedOrRaised</code> .
<code>x</code>	Column on the x axis: one bar (or bar segment) per level; <code>"importedOrRaised"</code> included.
<code>facet</code>	Optional facet column; <code>"importedOrRaised"</code> included.
<code>fill</code>	Column mapped to the bar fill. <code>NULL</code> (default) colours the bars by catch category - landings, discards, BMS; <code>"importedOrRaised"</code> is allowed.
<code>alpha</code>	<code>"importedOrRaised"</code> (default) marks what the raising added: solid bars for imported weight, transparent and hatched ones for raised. <code>NULL</code> draws no mark. Left out when <code>importedOrRaised</code> is the x, the fill or the facet, and when nothing was raised.
<code>catchCategory</code>	Optional filter to just <code>"Lan"</code> , <code>"Dis"</code> or <code>"BMS"</code> (matched case-insensitively), so discards are readable on their own when landings would otherwise dominate the scale. <code>NULL</code> (default) keeps all the catch parts, stacked.
<code>ncol</code>	Facet columns, when facet is set.
<code>palette</code>	<code>"default"</code> , <code>"colorblind"</code> or <code>"gradual"</code> : the bar colours.
<code>save</code>	Where to write the plot as well, or <code>NULL</code> (default) to only return it: a folder, a file name, or a list of options - see saving-plots .

Details

`importedOrRaised` is a variable like any other. By default it is the mark (`alpha = "importedOrRaised"`): solid for what was imported, transparent and hatched for what raising added. Name it in `fill` to

give the two a colour each (the bars are then no longer coloured landings / discards: say which they are with catchCategory, or with facet = "catchCategory"), in facet to give each its own panel, or in x for a bar each; it is then not drawn as a mark as well. alpha = NULL draws no mark, and unless it is named elsewhere the imported and the raised weights are added together.

The title says what the rows drawn are: *imported + raised* when the raising added some of them, *only imported* when it did not - a raw catch, a raised catch in which nothing needed raising, or the landings of a raised catch, which are always imported - and then there is no mark, because there is nothing to tell apart.

Value

A ggplot object.

See Also

[plot_DR\(\)](#) for the discard rate of the same catch, before or after the raising; [raised_overview_table\(\)](#) for the weights as a table.

Other raising plots: [plot_compare_raised\(\)](#), [plot_recommended_groups\(\)](#), [plot_river\(\)](#)

Examples

```
## Not run:

# the catch as reported, by metier
plot_catches(catch, x = "metier4")

# the same of the raised catch: solid = imported, hatched = raised
raised <- catchRaising(catch, strataCond, logFile = tempfile(fileext = ".txt"))
plot_catches(raised, x = "metier4", facet = "vesselFlagCountry")

# only the discards, imported and raised side by side
plot_catches(raised, x = "metier4", catchCategory = "Dis",
              fill = "importedOrRaised")

## End(Not run)
```

plot_compare_raised	<i>Raised catch of several scenarios, side by side</i>
---------------------	--

Description

[plot_catches\(\)](#) for two or more raised catches at once - two ways of defining the groups, say. For each level of x (and each panel of facet) one bar per scenario: its colour says which scenario, its mark whether the weight came with the data (imported, solid) or was added by the raising (raised, transparent and hatched). The bars are catch weight, or only the landings or only the discards with catchCategory.

Usage

```
plot_compare_raised(
  raised_catch_scenarios,
  scenarios_names = NULL,
  x = NULL,
  facet = NULL,
  catchCategory = NULL,
  ncol = 2,
  palette = c("default", "colorblind", "gradual"),
  save = NULL
)
```

Arguments

<code>raised_catch_scenarios</code>	Two or more raised catches (the output of <code>catchRaising()</code>), written <code>list(a, b)</code> or <code>c(a, b)</code> .
<code>scenarios_names</code>	Optional names for the scenarios, in the same order. Default: the names given inside <code>c()</code> / <code>list()</code> , or else the object names.
<code>x</code>	Column on the x axis: one group of bars per level. NULL (default): one group for everything.
<code>facet</code>	Optional facet column.
<code>catchCategory</code>	Optional filter to just "Lan" or "Dis" (matched case-insensitively), so discards are readable on their own when landings would otherwise dominate the scale. NULL (default) keeps every category, summed.
<code>ncol</code>	Facet columns, when facet is set.
<code>palette</code>	"default", "colorblind" or "gradual": the scenario colours.
<code>save</code>	Where to write the plot as well, or NULL (default) to only return it: a folder, a file name, or a list of options - see saving-plots .

Details

Only the raised catches are needed: the plot sums each of them itself, at whatever x and facet name, so changing either never needs a table built first. They are written `list(a, b)` or `c(a, b)`; a scenario without a name of its own is called what its object is called, and `scenarios_names` overrides either. Every raised catch must carry the columns x and facet name.

Catch is every catch category: landings, discards and anything else the raised catch carries (BMS), which counts toward the catch but is neither landings nor discards. [raised_overview_table\(\)](#) gives the same comparison as a table, category by category.

Value

A ggplot object, or NULL when there is nothing to draw.

See Also

[raised_overview_table\(\)](#) for the same comparison as a table; [plot_catches\(\)](#) for one raised catch.

Other raising plots: [plot_catches\(\)](#), [plot_recommended_groups\(\)](#), [plot_river\(\)](#)

Examples

```
## Not run:
cef <- subset(catch, year == 2023)
log <- tempfile(fileext = ".txt")

# the groups, and the data each takes its discard ratio from: its own, or everything
strataCond <- list(TBB = rlang::quo(metier4 == "TBB"),
                  REST = rlang::quo(metier4 != "TBB"))
own <- strataCond
all <- list(TBB = rlang::quo(TRUE), REST = rlang::quo(TRUE))
raised_own <- catchRaising(cef, strataCond, own, logFile = log)
raised_all <- catchRaising(cef, strataCond, all, logFile = log)

# the scenarios are named after the objects ...
plot_compare_raised(c(raised_own, raised_all), x = "metier4")

# ... or after you; discards only, one panel per quarter
plot_compare_raised(c(own = raised_own, everything = raised_all),
                    x = "metier4", facet = "seasonValue", catchCategory = "Dis")

## End(Not run)
```

plot_coverage_map

Coverage map: one pie per ICES area

Description

A map of the fishing area with one small pie per ICES area: the pie's radius is the area's landed (or caught) weight, and the pie is split into the sampled and the not-sampled share of that weight. With group (for example "seasonValue", "vesselFlagCountry" or "metier4") each area gets a labelled box holding a tidy grid of pies, one per group level; overlapping boxes are nudged apart and a thin leader keeps each box tied to its area's true position.

Usage

```
plot_coverage_map(
  catch,
  distributions = NULL,
  area = "areaValue",
  group = NULL,
  what = "discard",
  catchCategory = NULL,
```

```

    scale = 1,
    palette = c("default", "colorblind", "gradual"),
    save = NULL
  )

```

Arguments

catch	CEF catch data (with areaValue and the area / group columns on it), or a build_coverage_strata() table (its own what is then used).
distributions	A CEF distributions table - needed for what = "length" / "age" with raw catch, ignored otherwise.
area	Column holding the ICES area code: "areaValue" (default), or a coarser level such as "ICESSubArea" or "ICESDivision", which needs areaValue in catch to be placed where the data is (see <i>Where a pie is placed</i>).
group	Optional column: one boxed grid of pies per area, one pie per level. A group with a single level warns and draws one pie per area.
what, catchCategory	As in coverage-plots . The weight word in the title comes from what and catchCategory.
scale	Multiplies every pie radius (and, when grouped, the size of the box's grid cell).
palette	"default", "colorblind" or "gradual": the colour of the sampled slice.
save	Where to write the plot as well, or NULL (default) to only return it: a folder, a file name, or a list of options - see saving-plots .

Details

"Sampled" means what it means in the other coverage plots (see [coverage-plots](#)): the weight carries a discard estimate (what = "discard") or a linked length / age distribution. Like them, the map takes raw catch (plus distributions for length and age) or a [build_coverage_strata\(\)](#) table that has the area column (and the group column) among its dims.

Where a pie is placed:

Areas are placed at approximate centroids of the ICES divisions and subdivisions. A coarser area column ("ICESSubArea", "ICESDivision") is placed where its data is: at the mean of the centroids of the areaValues that have data in it. The pie of subarea 27.7 sits around 7.d and 7.e when nothing comes from 7.a or 7.f, not in the middle of the whole of 7, and 27.3 sits between 27.3.a.20 and 27.3.a.21 when those are the only two areas with data. Only a level with no areaValue behind it - a table aggregated to the subarea alone, say - takes the centroid of the area itself, and a subarea or division the list leaves out (3, 3.d, 5 ...) has one too, the mean of the listed areas inside it. An area with no centroid at all is warned about by name and left off the map. The map needs the suggested maps package.

Value

A ggplot object, or NULL (with a message) when nothing can be mapped.

See Also

[coverage-plots](#) for the heatmap, the bubbles and the bars; [build_coverage_strata\(\)](#) for the table behind the pies.

Other coverage plots: [build_coverage_strata\(\)](#), [coverage-plots](#)

Examples

```
## Not run:
if (requireNamespace("maps", quietly = TRUE)) {
  plot_coverage_map(catch)
  plot_coverage_map(catch, group = "seasonValue")

  # one pie per subarea, placed at the mean of the areaValue centroids with data
  plot_coverage_map(catch, area = "ICESSubArea", group = "fleetGroup")
}

## End(Not run)
```

plot_discard_cluster_variables

Which variable best predicts the discard-rate cluster

Description

Cramer's V of each grouping column against the cluster it helped describe ([build_discard_clusters\(\)](#))
 - 0 means unrelated to the clustering, 1 means it alone determines it.

Usage

```
plot_discard_cluster_variables(
  x,
  palette = c("default", "colorblind", "gradual"),
  save = NULL
)
```

Arguments

x	Output of build_discard_clusters() .
palette	"default", "colorblind" or "gradual": the bar colour.
save	Where to write the plot as well, or NULL (default) to only return it: a folder, a file name, or a list of options - see saving-plots .

Value

A ggplot object.

See Also

[build_discard_clusters\(\)](#), [plot_discard_clusters\(\)](#).

`plot_discard_clusters` *Discard-rate clusters, shown on the map of dims*

Description

Every sampled stratum, coloured by its discard-rate cluster ([build_discard_clusters\(\)](#)), on the MCA map of dims - an ellipse per cluster so a reader can see at a glance whether it occupies its own part of the map or overlaps the rest.

Usage

```
plot_discard_clusters(  
  x,  
  palette = c("default", "colorblind", "gradual"),  
  save = NULL  
)
```

Arguments

<code>x</code>	Output of build_discard_clusters() .
<code>palette</code>	"default", "colorblind" or "gradual": the cluster colours. Which cluster is "1" depends on this run's data and <code>k</code> alone, so a cluster's colour means nothing outside its own plot.
<code>save</code>	Where to write the plot as well, or NULL (default) to only return it: a folder, a file name, or a list of options - see saving-plots .

Value

A ggplot object.

See Also

[build_discard_clusters\(\)](#), [plot_discard_cluster_variables\(\)](#).

plot_DR

*Discard rate per stratum, before or after the raising***Description**

One point per stratum, its discard rate on the y axis and its landings as the point size - the finest a stratum gets, never an average of finer ones. Given raw catch it shows the rates as they were reported, with a thin line behind the points marking the mean rate of each x level, weighted by landings, and an optional reference that draws a dashed line and names the strata above it. Given the raised catch (the output of [catchRaising\(\)](#)) it shows what the raising did: a stratum whose discards came with the data (*imported*) is drawn as a circle, one whose discards the raising added (*raised*) as a cross. The arguments are the same either way, and so is the plot; the title says which it is.

Usage

```
plot_DR(
  catch,
  x,
  colour = NULL,
  shape = NULL,
  facet = NULL,
  alpha = "importedOrRaised",
  rate = c("ratio", "rate"),
  reference = NULL,
  jitter = TRUE,
  palette = c("default", "colorblind", "gradual"),
  save = NULL
)
```

Arguments

catch	Raw CEF catch data, the raised catch (catchRaising()), or a discard coverage table from build_coverage_strata() .
x	Column on the x axis: one point per level (times colour, shape and facet). Can be "DrGroup" or "importedOrRaised". Required.
colour	Column mapped to the point fill; "importedOrRaised" included.
shape	Column mapped to the point shape; "importedOrRaised" included.
facet	Optional facet column; "importedOrRaised" included.
alpha	"importedOrRaised" (default) marks what the raising added: circles for imported strata, crosses for raised ones (transparent circles when shape is a column). NULL draws no mark. Left out when importedOrRaised is the x, colour, shape or facet, and when nothing was raised.
rate	"ratio" (default, discards / landings) or "rate" (discards / catch): which discard rate is the y variable. See <i>The rate</i> .

reference	Optional single rate, as a fraction ($0.3 = 30\%$), on the scale of rate. Draws a dashed line and, before the raising, labels the strata above it.
jitter	Jitter the points horizontally. Cosmetic only - one point is still one stratum.
palette	"default", "colorblind" or "gradual": the point colours.
save	Where to write the plot as well, or NULL (default) to only return it: a folder, a file name, or a list of options - see saving-plots .

Details

What a point is:

A stratum is one row of the discard coverage table (`build_coverage_strata()`). Given a discard coverage table, its rows are drawn exactly as they are - one point per row. Given catch, raw or raised, the table is built at the finest resolution the catch carries (`fleetValue`, `areaValue`, `vesselFlagCountry`, `seasonValue`, whichever are present) *together with* `x` / colour / shape / facet, so a point is always an individual, un-pooled stratum: several points can share one `x` level. `strata_vars` (the columns behind a point's label, see below) is exactly this set. Strata with no discard estimate have no rate and are not drawn: a rate is a property of sampled strata only.

Before and after the raising:

The raising adds `importedOrRaised` to the catch, and with it the plot knows what the raising did. `importedOrRaised` is a variable like any other. By default it is the mark (`alpha = "importedOrRaised"`): imported strata are circles, raised ones crosses (in the colour a circle would be filled with). When shape is already a column of its own the symbol is taken, and raised strata are transparent circles instead. Name it in `x`, colour, shape or facet to draw it there instead - a colour each, a panel each, two columns of points - and it is then not drawn as a mark as well. `alpha = NULL` draws no mark, and unless it is named elsewhere imported and raised strata are drawn alike. Whichever way, it is worked out per stratum (a stratum is raised when its discard rows were), never from the landings rows next to them, which are always imported; a stratum that holds both kinds of discard row is raised.

The title says what the strata drawn are: *imported + raised* when the raising added some of them, *only imported* when it did not - a raw catch, or a raised catch in which nothing needed raising - and then there is no mark, because there is nothing to tell apart.

Put the raising group on `x` ("`DrGroup`"): a group's crosses all sit on one line, the rate it was raised at, and its circles show how far the strata it was built from scatter around that line - a group whose circles are spread far from it is a guess dressed up as a group. Only the strata of the raised catch are drawn, so the donors of a group wider than its own strata (a REST group raised from every gear, say) appear under the groups they themselves belong to.

A raised stratum whose group had no donor to draw a rate from comes out of the raising as `NaN`: it has no rate to plot and is left out, with a message saying how many.

Once strata are raised the crosses are themselves the reference, so the weighted-mean line, the reference line's label and the per-point labels, which help to read the rates as they were reported, are left out; the dashed reference line stays.

The rate:

rate chooses what is plotted, and both are columns of the coverage table:

- "ratio" (the default, the "discard ratio") is $\text{rate} = \text{total} / \text{weightingTotal}$: the stratum's discards `D` over the landings `L` that carry a discard estimate - the same number the CAS output reports. It is a ratio, so it can exceed 100%.

- "rate" (the "discard rate") is $\text{discardRate} = \text{total} / (\text{weightingTotal} - \text{total})$: the discards as a share of the catch - those landings plus the discards. It stays between 0 and 100%.

The weighted mean:

The line on each x level (in each panel, when faceted) is the mean of the plotted rates of the strata at that level, weighted by the landings that carry a discard estimate (`weightingTotal`, the CAS weight): a big stratum counts for more than a small one. For "ratio" it equals the rate of the x level built directly, `build_coverage_strata(catch, x)`. It sits behind the points, thin and muted, so it reads as a reference, not the headline. Before the raising only.

The reference line:

reference is a rate on the same scale as the axis, as a fraction (0.3 is 30%; with `rate = "rate"` it must lie between 0 and 1). A dashed line marks it and, before the raising, the strata above it are labelled with their identity - the finest column of each stratum family present (so a coarser alias already determined by a finer one you also plotted, `metier4` alongside `fleetValue` say, is not repeated), bare values joined by "|" ("`PS_SPF_0_0_0|87.1.2|Chile|1`") - a point is an individual stratum, so its label identifies it fully, not just by the columns you happened to plot.

The y axis:

For "ratio" the axis is pseudo-log: linear near zero (zero stays zero) and logarithmic above about 5%. A few tiny strata (a few kilograms of covered landings) give ratios of many hundreds of percent that would flatten everything else on a plain axis. Add `+ ggplot2::scale_y_continuous()` to the returned plot to replace it with an ordinary one. "rate" is bounded, so its axis is an ordinary 0-100% one.

Value

A ggplot object, or NULL when no stratum has a discard rate.

See Also

`build_coverage_strata()` for the table behind the points; `coverage-plots` for how well those strata are sampled; `plot_catches()` for the weight itself, before or after the raising.

Examples

```
## Not run:

plot_DR(catch, x = "metier4", colour = "vesselFlagCountry")
plot_DR(catch, x = "vesselFlagCountry", colour = "metier4",
        facet = "seasonValue")

# discards as a share of the catch, with a reference line: the strata
# above 20% are named, in full
plot_DR(catch, x = "metier4", colour = "vesselFlagCountry",
        rate = "rate", reference = 0.2)

# the same plot of the raised catch: imported are circles, raised crosses
raised <- catchRaising(catch, strataCond, logFile = tempfile(fileext = ".txt"))
```

```

plot_DR(raised, x = "DrGroup", colour = "metier4")

# imported against raised as the panels instead of the crosses
plot_DR(raised, x = "metier4", colour = "vesselFlagCountry",
        facet = "importedOrRaised")

# the table behind the points
cov <- build_coverage_strata(catch, c("metier4", "vesselFlagCountry"))
head(cov[!is.na(cov$rate), c("metier4", "vesselFlagCountry", "total", "rate",
                             "discardRate")])

## End(Not run)

```

plot_MWaLoA

Mean weight at length or age, before or after the allocation

Description

The number-weighted mean weight in each length (or age) class, as a line with points. Weight is in grams, length in cm and age in years. When the strata are averaged (no shape) and a spread is available, a +/- 1 sd band is drawn as well. Two options help to judge how far to trust the curve: a plus group pools the oldest ages into one class, and semi-transparent bars behind the lines show how many fish stand behind each point. Given the result of the allocation ([catchAllocationAoL\(\)](#)) it shows what the allocation did: `samledOrEstimated` says whether a mean weight was sampled or estimated for a stratum nobody sampled, and it is drawn as the mark (a solid line with circles = sampled, a dotted line with crosses = estimated), as the colour or as a facet, as you choose. The arguments are the same before and after, and so is the plot; the title says which it is.

Usage

```

plot_MWaLoA(
  data,
  distributions = NULL,
  distributionType = c("LengthTotal", "Age"),
  catchCategory = "All",
  colour = NULL,
  facet = NULL,
  shape = NULL,
  alpha = "samledOrEstimated",
  plus_group = NULL,
  bin_width = NULL,
  show_samples = FALSE,
  scales = "fixed",
  palette = c("default", "colorblind", "gradual"),
  save = NULL
)

```

Arguments

data	CEF catch data carrying the columns you group by (derive_fields() output, or your own); after the allocation, its \$catch.
distributions	A CEF distributions table, linked to the catch through domainBiology; after the allocation, its \$distribution.
distributionType	"LengthTotal" (default) or "Age".
catchCategory	"All" (default: landings and discards together, titled "catch"), "Lan", "Dis" or "BMS". Only used with distributions.
colour	Column mapped to the line colour; "sampledOrEstimated" included.
facet	Up to two facet columns: "col" or c("row", "col"); "sampledOrEstimated" included.
shape	Column drawn as separate series with their own point shapes; opens the averaged group back up and drops the band.
alpha	"sampledOrEstimated" (default) marks what the allocation estimated: a solid line with circles for sampled mean weights, a dotted line with crosses for estimated ones (transparent circles when shape is a column). NULL draws no mark. Left out when sampledOrEstimated is the colour, the shape or a facet, and when nothing was estimated.
plus_group	The first age of the plus group: 7 pools ages 7 and older into one class, shown as 7+. NULL (default): no plus group. Ages only. See <i>The plus group</i> .
bin_width	Width, in cm, of the length bins every class is pooled into. NULL (default): no binning. Lengths only. See <i>Length bins</i> .
show_samples	TRUE draws bars of the fish measured at each class behind the lines, on a second axis. Default FALSE. See <i>The fish behind each point</i> .
scales	Facet scales: "fixed" (default), "free_x", "free_y" or "free".
palette	"default", "colorblind" or "gradual".
save	Where to write the plot as well, or NULL (default) to only return it: a folder, a file name, or a list of options - see saving-plots .

Details

data is the catch and distributions the CEF distributions: they are linked through domainBiology and the country and summarised inside, with catchCategory choosing the landings, the discards, BMS or the catch as a whole. The catch part goes in the plot title. A distributions row that repeats another exactly is dropped, with a warning: a repeat is not a second sample.

A mean weight is not additive, so strata that are pooled are averaged weighted by the numbers measured at that class: a stratum that measured five fish cannot outweigh one that measured five hundred. The band is the number-weighted standard deviation across the strata pooled into each point; a point built from a single stratum has none.

colour groups the lines. shape opens the averaged group back up - one series per level, each drawn separately - and drops the band. facet takes one column (a wrapped set of panels) or two (c("row", "col"), a grid). Live weight and a line are the only options, so there is no variableType or type argument. The legend sits inside the panel like the other plots', except with show_samples, plus_group or the mark, where it moves outside so it cannot cover the right-hand end of the curve.

After the allocation:

As in `plot_NaLoA()`: give both tables of the allocation's result, its `$catch` as data and its `$distribution` as distributions. The allocation adds `sampldOrEstimated` to the distributions, and it is a variable like any other: by default the mark (`alpha = "sampldOrEstimated"`), or named in colour (a colour each) or in facet (a panel each), where it is not drawn as a mark as well. With the mark, every combination of the plotted columns is one line per sampled / estimated, so the sampled curve (solid, with circles) and the curve the allocation estimated for the unsampled strata (dotted, with crosses) sit side by side. When `shape` is already a column of its own the crosses are not available, and the estimated points are transparent instead. `alpha = NULL` draws no mark, and unless it is also the colour or a facet the sampled and the estimated weights are averaged together, weighted by their numbers.

The title says what the distributions drawn are: *sampld + estimated* when the allocation estimated some of them, *only sampld* when it did not, and then there is no mark, because there is nothing to tell apart. A distribution the allocation estimated has no measured fish to count, so it has no bar with `show_samples`.

The plus group:

`plus_group = 7` pools every age from 7 upward into one class, labelled 7+ on the axis and in the caption. Inside each distribution the numbers of those ages are added and their mean weights averaged, weighted by those numbers; the pooled classes are then averaged over strata like any other point, so the band is still a spread across strata, not across ages. It shows what a plus group does to the curve: the ragged tail of old ages, each on a handful of fish, becomes one point with many. It applies to ages only.

Length bins:

`bin_width = 5` is the length counterpart: every length class is grouped into 5cm bins (10-14 becomes one class labelled 10-15), pooled the same way a plus group is - numbers added, mean weights averaged weighted by those numbers, inside each distribution first, then strata averaged as usual. It applies to lengths only (`distributionType = "LengthTotal"`).

The fish behind each point:

`show_samples = TRUE` draws, behind the lines, semi-transparent bars of the number of fish measured at each class, read on a second axis on the right. A mean weight that rests on a few fish is uncertain however smooth the line looks; a short bar under a point says so. The bars are stacked by colour, one stack per class in each panel, and follow the plus group when there is one.

The distributions record `numMeasurements` once for a whole distribution, not per class, so the count at a class is that number shared out over the distribution's classes in proportion to its numbers there. This is exact for a length distribution raised from a length sample by one factor, and an estimate for ages read off an age-length key. A distribution counts once however many strata it is linked to, and one that reports no count (`numMeasurements 0`) adds none - a point with no bar is a point whose sample size is unknown, not zero, and the caption says so.

Value

A ggplot object, or NULL when there is nothing to draw.

See Also

`plot_NaLoA()` for the numbers in the same classes; [coverage-plots](#) for whether the strata have a distribution at all; `meanWoLatAoLAggregation()` for the same mean weights as a table.

Other distribution plots: `plot_NaLoA()`

Examples

```
## Not run:

# mean weight at length of the landings, one line per country
plot_MWLoA(catch, distributions,
            catchCategory = "Lan", colour = "vesselFlagCountry")

# at age, one panel per quarter
plot_MWLoA(catch, distributions,
            distributionType = "Age",
            colour = "vesselFlagCountry", facet = "seasonValue")

# what a plus group does: every age from 3 upward pooled into "3+"
plot_MWLoA(catch, distributions,
            distributionType = "Age", colour = "vesselFlagCountry",
            plus_group = 3)

# length classes pooled into 2cm bins
plot_MWLoA(catch, distributions, colour = "vesselFlagCountry",
            bin_width = 2)

# the fish measured at each age, as bars behind the lines
plot_MWLoA(catch, distributions,
            distributionType = "Age", colour = "vesselFlagCountry",
            show_samples = TRUE)

# after the allocation: both tables of its result. Landings by country,
# solid line = sampled, dotted line with crosses = estimated
plot_MWLoA(alloc$catch, alloc$distribution,
            catchCategory = "Lan", colour = "vesselFlagCountry")

# sampled and estimated as the colour instead
plot_MWLoA(alloc$catch, alloc$distribution, catchCategory = "Lan",
            colour = "sampledOrEstimated")

## End(Not run)
```

Description

The catch numbers spread over the length (or age) classes, as a stacked or dodged histogram. Numbers are in thousands, length in cm and age in years. Given the result of the allocation ([catchAllocationAol\(\)](#)) it shows what the allocation did: `sampldOrEstimated` says whether a number was sampled or estimated for a stratum nobody sampled, and it is drawn as the mark (solid bars = sampled, transparent and hatched bars = estimated), as the fill or as a facet, as you choose. The arguments are the same before and after, and so is the plot; the title says which it is.

Usage

```
plot_NaLoA(
  data,
  distributions = NULL,
  distributionType = c("LengthTotal", "Age"),
  catchCategory = "All",
  fill = NULL,
  facet = NULL,
  alpha = "sampldOrEstimated",
  stacked = TRUE,
  plus_group = NULL,
  bin_width = NULL,
  scales = "fixed",
  palette = c("default", "colorblind", "gradual"),
  save = NULL
)
```

Arguments

<code>data</code>	CEF catch data carrying the columns you group by (derive_fields() output, or your own); after the allocation, its <code>\$catch</code> .
<code>distributions</code>	A CEF distributions table, linked to the catch through <code>domainBiology</code> ; after the allocation, its <code>\$distribution</code> .
<code>distributionType</code>	"LengthTotal" (default) or "Age".
<code>catchCategory</code>	"All" (default: landings and discards together, titled "catch"), "Lan", "Dis" or "BMS". Only used with distributions.
<code>fill</code>	Column mapped to the bar fill; "sampldOrEstimated" included.
<code>facet</code>	Up to two facet columns: "col" or <code>c("row", "col")</code> ; "sampldOrEstimated" included.
<code>alpha</code>	"sampldOrEstimated" (default) marks what the allocation estimated: solid bars for sampled numbers, transparent and hatched ones for estimated. NULL draws no mark. Left out when <code>sampldOrEstimated</code> is the fill or a facet, and when nothing was estimated.
<code>stacked</code>	Stack the bars on fill (default) rather than dodge them.
<code>plus_group</code>	The first age of the plus group: 7 pools ages 7 and older into one class, shown as 7+. NULL (default): no plus group. Ages only.

bin_width	Width, in cm, of the length bins every class is pooled into. NULL (default): no binning. Lengths only.
scales	Facet scales: "fixed" (default), "free_x", "free_y" or "free".
palette	"default", "colorblind" or "gradual".
save	Where to write the plot as well, or NULL (default) to only return it: a folder, a file name, or a list of options - see saving-plots .

Details

data is the catch and distributions the CEF distributions: they are linked through domainBiology and the country and summed inside, with catchCategory choosing the landings, the discards, BMS or the catch as a whole. The catch part goes in the plot title. A distributions row that repeats another exactly is dropped, with a warning: a repeat is not a second sample.

After the allocation:

Give both tables of the allocation's result: its \$catch as data and its \$distribution as distributions, plot_NaLoA(alloc\$catch, alloc\$distribution). The distributions it estimated for a stratum with no biological domain carry a domain the allocation made up, and only its own \$catch knows it - the catch that went into the allocation cannot find them, which the plot warns about.

The allocation adds sampledOrEstimated to the distributions, and from then on it is a variable like any other. By default it is the mark (alpha = "sampledOrEstimated"): solid bars where the numbers were sampled, transparent and hatched ones where the allocation estimated them. Name it in fill to give the two a colour each, or in facet to give each its own panel; it is then not drawn as a mark as well. alpha = NULL draws no mark, and unless it is also the fill or a facet the sampled and the estimated numbers are added together - the numbers the allocation finally reports.

The title says what the distributions drawn are: *sampled + estimated* when the allocation estimated some of them, *only sampled* when it did not - distributions as they were sampled, or an allocation that estimated none of the ones drawn - and then there is no mark, because there is nothing to tell apart.

Like the allocation itself, the numbers count a distribution once for every stratum of the catch that names its domainBiology, so they assume the catch has one row per domain, catch category and country - which CEF files built from InterCatch have. Where there are several, the numbers come out inflated, though the shapes still read.

Fill, facets and classes:

fill colours (and stacks) the bars by a column of the catch. facet takes one column (a wrapped set of panels) or two (c("row", "col"), a grid). plus_group (ages) and bin_width (lengths) pool the classes exactly as in plot_MWLoA() - see that function's *The plus group* and *Length bins* - except numbers simply add rather than being averaged, so there is no per-distribution weighting step to speak of.

Value

A ggplot object, or NULL when there is nothing to draw.

See Also

[plot_MWaLoA\(\)](#) for the mean weight in the same classes; [coverage-plots](#) for whether the strata have a distribution at all; [catchAtAoLAggregation\(\)](#) for the same numbers as a table; [plot_catches\(\)](#) for the same sampled / raised contrast on the weights.

Other distribution plots: [plot_MWaLoA\(\)](#)

Examples

```
## Not run:

# numbers at length of the landings, stacked by country
plot_NaLoA(catch, distributions,
            catchCategory = "Lan", fill = "vesselFlagCountry")

# numbers at age, one panel per quarter, dodged
plot_NaLoA(catch, distributions,
            distributionType = "Age", fill = "vesselFlagCountry",
            facet = "seasonValue", stacked = FALSE)

# every age from 3 upward pooled into "3+"
plot_NaLoA(catch, distributions,
            distributionType = "Age", fill = "vesselFlagCountry", plus_group = 3)

# length classes pooled into 2cm bins
plot_NaLoA(catch, distributions, fill = "vesselFlagCountry", bin_width = 2)

# after the allocation: both tables of its result. Landings by country,
# solid = sampled, hatched = estimated
alloc <- catchAllocationAoL(raised, distributions,
                           condition_alloc_st_list = list(ALL = rlang::quo(TRUE)),
                           distributionType = "LengthTotal", verbose = FALSE,
                           logFile = tempfile(fileext = ".txt"))
plot_NaLoA(alloc$catch, alloc$distribution,
            catchCategory = "Lan", fill = "vesselFlagCountry")

# sampled and estimated as the fill instead, and the countries as panels
plot_NaLoA(alloc$catch, alloc$distribution, catchCategory = "Lan",
            fill = "sampledOrEstimated", facet = "vesselFlagCountry")

## End(Not run)
```

plot_recommended_groups

The level each stratum is raised from

Description

Draws the result of [recommend_groups\(\)](#): for the landings of the discard raising - or the landings and the discards of a length or age allocation - the level their discards (their lengths, their ages)

come from. The weight that has an estimate or a distribution of its own is the solid part of L1, the stratum as it was reported; the weight that has none is hatched, and its colour is the level of the pool it is raised (allocated) from. With *x* the same is broken down by a column of the strata - a variable of the hierarchy, say - so that it shows where the coarse pools are needed: which areas, meshes or countries can stand on their own data and which have to borrow.

Usage

```
plot_recommended_groups(
  rec,
  x = NULL,
  facet = NULL,
  share = FALSE,
  palette = c("default", "colorblind", "gradual"),
  save = NULL
)
```

Arguments

<i>rec</i>	The result of <code>recommend_groups()</code> .
<i>x</i>	Column of the strata for the bars, e.g. "areaValue"; NULL (default) for one bar for each level. Any column of <i>rec\$strata</i> : a variable of the hierarchy, group, or catchCategory for lengths and ages.
<i>facet</i>	Optional column of <i>rec\$strata</i> for panels. Only with <i>x</i> .
<i>share</i>	TRUE: shares instead of tonnes (see "Details"). Default FALSE.
<i>palette</i>	"default", "colorblind" or "gradual": the colours of the levels.
<i>save</i>	Where to write the plot as well, or NULL (default) to only return it: a folder, a file name, or a list of options - see saving-plots .

Details

It is the *landings* that are drawn, and the discards that are raised: a bar is the weight of the landings whose discards come from the level. The title says which grouping it is - "Discard raising groups", "Length allocation groups" or "Age allocation groups" - and there is no argument for it.

Without *x*, one bar for each level, the narrowest on top. L1 is the stratum as it is reported: its solid bar is the landings that have a discard estimate and are imported as reported. L2 is the reported strata summed into the strata of the hierarchy (the own stratum, aggregated - it is not there when the hierarchy is the reported stratum already), and each level below pools across one more variable, from the last of the hierarchy up, to All by All. The hatched part of a bar is the landings that have no estimate of their own and are raised from that level. The number at the end of a bar is how many strata it holds. For lengths and ages the landings and the discards are two panels.

With *x*, one bar for each value of *x*, stacked by the level its strata take. *share* = TRUE draws the weights as shares: percent of the landings (of the catch part, for lengths and ages) in the figure of the levels, and bars of 100 % with *x*, so that the values can be compared whatever their weight. `plot_river()` draws the same flows as ribbons.

A level is a colour, the same in every figure of the groups. A tall solid bar at L1 means most of the stock stands on its own data; a tall hatched bar further down means a lot of it is raised from a wide pool.

Value

A ggplot object.

See Also

[recommend_groups\(\)](#); [plot_river\(\)](#) for the flows from a variable to the levels.

Other raising groups: [recommend_groups\(\)](#)

Other raising plots: [plot_catches\(\)](#), [plot_compare_raised\(\)](#), [plot_river\(\)](#)

Examples

```
# the small catch (kg) of ?recommend_groups
catch <- data.frame(
  catchCategory = c("Lan", "Lan", "Lan", "Lan", "Lan", "Lan", "Dis", "Dis"),
  areaValue     = c("27.4", "27.4", "27.4", "27.7.d", "27.7.d", "27.7.d", "27.4", "27.7.d"),
  mesh         = c("70-99", "70-99", "100-119", "70-99", "70-99", "100-119", "70-99", "70-99"),
  total        = c(3000, 1000, 2000, 800, 200, 500, 200, 40),
  domainCatchDis = c("d1", NA, NA, "d2", NA, NA, "d1", "d2"),
  variableUnit  = "kg", stringsAsFactors = FALSE)
rec <- recommend_groups(catch, c("areaValue", "mesh"), list(cov = 50))
plot_recommended_groups(rec)                # which level the landings are raised from
plot_recommended_groups(rec, share = TRUE)   # ... as a share of the landings
plot_recommended_groups(rec, x = "areaValue") # ... and where

## Not run:
rec <- recommend_groups(catch_data,
  c("areaValue", "mesh", "vesselFlagCountry", "seasonValue"))
plot_recommended_groups(rec)                # which level the landings are raised from
plot_recommended_groups(rec, x = "areaValue") # ... and where
plot_recommended_groups(rec, x = "vesselFlagCountry", share = TRUE)

## End(Not run)
```

plot_river

River plot: which strata are raised from which group

Description

An alluvial plot with one ribbon per stratum, running from the stratum on the left to the donor group on the right, its width the stratum's weight. It answers the question a table cannot: how much of the stock is being raised from each pool, and which strata were swept into the fallback.

Usage

```
plot_river(
  df,
  left_var,
  right_var = "raise_level",
```

```

value_var = "landings",
title = "Which strata are raised from which group",
left_label = NULL,
right_label = NULL,
stratum_axis = "Stratum",
palette = c("default", "colorblind", "gradual"),
save = NULL
)

```

Arguments

df	A stratum table: one row per stratum, with the columns named by 'left_var', 'right_var' and 'value_var'. Or the result of [recommend_groups()]: its strata, which run from the value of 'left_var' (any column of them - a variable of the hierarchy, say) to the level they are raised at, the weight they raise there the width.
left_var	Column on the left axis, usually the stratum label.
right_var	Column on the right axis, usually the donor group (the level, for the result of [recommend_groups()]).
value_var	Column giving each ribbon its width ('raised_t', the weight that is raised, for the result of [recommend_groups()]).
title	Plot title.
left_label, right_label	Axis labels.
stratum_axis	Label for the stratum axis.
palette	"default", "colorblind" or "gradual": the colours of the groups.
save	Where to write the plot as well, or NULL (default) to only return it: a folder, a file name, or a list of options - see saving-plots .

Value

A ggplot object, or 'NULL' when there is nothing to draw.

See Also

Other raising plots: [plot_catches\(\)](#), [plot_compare_raised\(\)](#), [plot_recommended_groups\(\)](#)

raised_overview_table *Imported against raised catch, by catch category: one raised catch or several*

Description

Each catch category's weight in tonnes - how much of it came with the data (imported) and how much the raising added (raised) - and each part as a share of its category, closed by a Total row. Give it two or more raised catches, say two ways of defining the groups, and they are laid out side by side, scenario by scenario. Give it by and every figure is split further by those columns - a season, a metier, an area, the raising group - each split with its own Total.

Usage

```
raised_overview_table(
  raised_catch_scenarios,
  scenarios_names = NULL,
  by = NULL
)
```

Arguments

<code>raised_catch_scenarios</code>	One raised catch (the output of <code>catchRaising()</code>), or two or more written list (a, b) or c(a, b).
<code>scenarios_names</code>	Optional names for the scenarios, in the same order. Default: the names given inside <code>c()</code> / <code>list()</code> , or else the object names.
<code>by</code>	Optional columns to split every figure by, in addition to the catch category and imported / raised: "seasonValue", c("metier4", "areaValue"), "DrGroup". Every raised catch must have them.

Details

The raised catches are written list(a, b) or c(a, b); one without a name of its own is called what its object is called, and scenarios_names overrides either. A single raised catch needs no list and comes back without the Scenario column.

Within each scenario and by split the rows are catch category, then imported before raised, and the scenarios of one split sit next to each other so the same figure can be read across them. Percent of category is the share of a part within its category in that split, so imported and raised add up to 100.

A raised row whose group had no donor to draw a rate from comes out of the raising as NaN: it raises to nothing and counts as no weight. The No donor (rows) column, shown only when there are any, counts them where they fall - by raising group, say - so they are not mistaken for a group that simply raised little.

Value

A tibble with Scenario (for several raised catches), Catch category, the by columns, Raised/Imported, Caton (t), Percent of category and, when any, No donor (rows).

See Also

`plot_compare_raised()` for the same comparison as a figure; `plot_catches()` for one raised catch.

Examples

```
## Not run:
cef <- subset(catch, year == 2023)
log <- tempfile(fileext = ".txt")
```

```

# the groups, and the data each takes its discard ratio from: its own, or everything
strataCond <- list(TBB = rlang::quo(metier4 == "TBB"),
                  REST = rlang::quo(metier4 != "TBB"))
own <- strataCond
all <- list(TBB = rlang::quo(TRUE), REST = rlang::quo(TRUE))
raised_own <- catchRaising(cef, strataCond, own, logFile = log)
raised_all <- catchRaising(cef, strataCond, all, logFile = log)

# one raised catch
raised_overview_table(raised_own)

# two, side by side, named after the objects ...
raised_overview_table(c(raised_own, raised_all))

# ... or after you, split by season and metier
raised_overview_table(c(own = raised_own, everything = raised_all),
                      by = c("seasonValue", "metier4"))

## End(Not run)

```

recommend_groups

Groups for the raising and the allocation, from a hierarchy of variables

Description

Chooses the groups of [catchRaising\(\)](#) (discards) or [catchAllocationAoL\(\)](#) (lengths and ages) from the data, instead of writing them by hand. You name the dimensions the groups are made by, from the one that matters most to the one that matters least - `c("areaValue", "mesh", "vesselFlagCountry", "seasonValue")`; the cluster plot of [build_discard_clusters\(\)](#) ranks them - and how well a pool has to be sampled. What has no estimate of its own is raised from the narrowest pool that is sampled well enough: first from its own stratum, then from the pool that is one variable wider, and so on up to All by All, so that the variable that matters most is the last one to be pooled across. What comes back holds the two lists of conditions the two functions take, and tables that show how the strata fell into the levels - see [plot_recommended_groups\(\)](#) and [plot_river\(\)](#) to draw them.

Usage

```

recommend_groups(
  catch_data,
  hierarchy,
  thresholds = list(cov = 50, trips = NA, fish = NA),
  distribution_data = NULL,
  distributionType = c("LengthTotal", "Age"),
  drop_catchCategory = TRUE
)

## S3 method for class 'rcef_groups'
print(x, ...)

```

Arguments

catch_data	The catch, a CEF long-format table that has the hierarchy columns (see derive_fields()): for discards the catch as it comes, for lengths and ages the raised catch. One year and one stock.
hierarchy	The dimensions the groups are made by, from the one that matters most to the one that matters least: c("areaValue", "mesh", "vesselFlagCountry", "seasonValue") (the area matters more than the mesh, the mesh more than the country ...). The first is kept the longest, the last is pooled across first. Columns of catch_data; what is not named is summed over from the start.
thresholds	A list: cov (the coverage a pool has to reach, in percent; default 50, cannot be empty), trips (the least number of trips) and fish (the least number of fish measured; lengths and ages), the last two NA (default) when not used. Leave out what you do not set; NULL is the same as NA. plot_coverage_grid() takes the same list.
distribution_data	The CEF distributions, for the groups of the lengths or the ages. NULL (default): the groups of the discards.
distributionType	"LengthTotal" (default) or "Age": the distributions the coverage is judged on. Only with distribution_data.
drop_catchCategory	Lengths and ages only: TRUE (default) makes the catch part the last variable to be pooled across, after the thresholds have been tested; FALSE keeps it at every level, the last one included. See "Discards, lengths and ages".
x	The result of recommend_groups().
...	Not used.

Details**The levels, from the narrowest to the widest:**

Nothing is pooled until it has to be. Take c("areaValue", "mesh", "vesselFlagCountry", "seasonValue") on a catch whose strata are reported as area, season, country and fleet. The levels are numbered from the narrowest, and a stratum is raised from the first of them whose pool reaches the thresholds:

Level	The pool a stratum is raised from
L1	own stratum (raw) : the strata as they are reported
L2	own stratum (aggregated) : the reported strata summed into strata of the dimensions
L3	pooled across seasonValue
L4	pooled across vesselFlagCountry and seasonValue
L5	pooled across mesh, vesselFlagCountry and seasonValue
L6	All by All

1. **L1, the strata as they are reported.** A landing that has a discard estimate (a discard row of its domain, as [catchRaising\(\)](#) pairs them) needs no raising, and the raising leaves it as it

is. Nothing is chosen for it: it is the part of the stock that stands on its own data. Only the landings without an estimate are raised, and what `recommend_groups()` chooses is the data their discard ratio is taken from. A stratum that has nothing left to raise gets the group L1. Printing the result says how much of the landings that is.

2. **L2, the strata of the dimensions.** The reported strata are summed into strata made of the variables you named - here the area, the mesh, the country and the season: what you did not name (the metier, say) is summed over. Each of them is tested against the thresholds *before anything else is pooled*: a stratum whose landings with an estimate reach them is raised from itself. If the dimensions are the reported strata already (the area, the season, the country and the fleet) there is nothing to sum and this level is skipped: the own stratum is L1, and the numbers that follow move up by one.
3. **L3 and on, across the last variable first.** The strata that fall short are pooled across the season - the same area, mesh and country in every season - and tested again; then across the country as well, then across the mesh. The pool that reaches the thresholds raises them.
4. **The last level, All by All.** Everything pooled across all the variables. It is the last resort and is always accepted - a warning says when it does not reach the thresholds - so that nothing is left without a group.

```

reported strata: area + season + country + fleet    (landings with an estimate stay as they are)
| L1 own stratum (raw)
| summed over what is not named, e.g. the metier
v
L2 area + mesh + country + season                  tested first
| pooled across the season
v
L3 area + mesh + country
| pooled across the country
v
L4 area + mesh
| pooled across the mesh
v
L5 area
| pooled across the area
v
L6 All by All                                     the last resort

```

So the variable that matters most, the area, is kept the longest and the one that matters least, the season, is pooled across first. The order you write them in is the hierarchy - one variable to a line, the most important first, shows it in a script:

```

hierarchy <- c("areaValue",      # matters most: the last one to be pooled across
              "mesh",
              "vesselFlagCountry",
              "seasonValue")      # matters least: the first one to be pooled across

```

Printing the result, and the figures of `plot_recommended_groups()`, spell it out as `areaValue > mesh > vesselFlagCountry`

A variable that is not in the hierarchy is not kept apart at any level - from the first step on, the strata are summed over it. If the metier matters for the discards, name it: `c("areaValue", "mesh", "metier4", "vesselFlagCountry")`. A variable and a coarser version of it can both be named,

the coarser one first: `c("ICESDivision", "areaValue", "mesh")` pools the strata of a division across their areas only when the pool of the area is not enough.

The thresholds:

`thresholds` is a list with up to three values; leave out what you do not use. A pool reaches the thresholds when it reaches every one that is used and has some sampled weight in it (a pool with nothing sampled has no ratio, or distribution, to give).

- `cov`, the coverage in percent: the share of the pool's weight that has been sampled - landings with a discard estimate, as `catchRaising()` counts them (see `weightingTotal` of `build_coverage_strata()`), or `catch` with a length (or age) distribution. It cannot be empty; the default is 50.
- `trips`, the number of trips behind the pool (`numTrips`, added up over its strata). NA (default): not used.
- `fish`, the number of fish measured behind the pool (`numMeasurements`, added up over its strata): lengths and ages only. NA (default): not used.

These are the numbers `plot_coverage_grid()` draws, and that function takes the same thresholds list - so the figure of where the sampling falls short and the groups made from it are judged by the same values. They are pooled the same way too: weights and counts add up, and the coverage is worked out again from the summed weights. A pool takes the rows of every stratum in it as its data - also those of the strata that were raised from their own, which are the best-sampled ones.

Discards, lengths and ages:

For discards give the `catch` as it is (after `derive_fields()`): the coverage is the share of the landings that have a discard estimate, and the groups go to `catchRaising()`. For lengths or ages also give the `distribution_data` and the `catch` the allocation is run on - the raised `catch`, the result of `catchRaising()`: the coverage is the share of the weight that has a distribution, and the groups go to `catchAllocationAoL()`. Landings and discards are two catch parts with their own coverage, and the conditions say which (`catchCategory == "Lan"`).

`drop_catchCategory` says what the catch part is among the variables. TRUE (default): it is the *last* variable to be pooled across - every level above is made within the landings or within the discards, the pool of everything in the catch part ("All by All (per catch part)") is tested with the thresholds, and only the strata that still fall short are pooled across the catch parts as well, in an All by All of landings and discards together. FALSE: the catch part is always kept - the landings are never allocated from the discards, nor the other way round - and the last level is All by All within the catch part, accepted whether or not it reaches the thresholds. It does not apply to discards, where the landings are the stratum and the discards its estimate.

The groups:

A group is named after its level and a number: L3_02 is the second pool of level 3, L1 the strata with nothing to raise, L6 the pool of All by All (Lan_L3_02 for the landings, when there are lengths). `strataCond` selects the strata a group raises - every row of them, so the groups share the catch out without overlap - and `matchedDataCond` the rows of the pool its discard ratio (or its distribution) is taken from, which are not only the strata it raises: strata that took a narrower pool give their rows to the wider pools too. Nothing is raised in L1, so it draws on everything (of its catch part): a landing of no weight and no estimate would otherwise have no donor, and come out as NaN.

Value

A list of class `rcef_groups`. Printing it says how much of the landings already has a discard estimate (a length or age distribution) and is kept as it is, and how many strata, groups and landings fall in each level. It holds

- `strataCond` and `matchedDataCond`: named lists of quosures, one per group, for the arguments of `catchRaising()` (`condition_raising_st_list`, `condition_matched_data_list`) or `catchAllocationAoL()` (`condition_alloc_st_list`, `condition_matched_data_list`);
- `strata`: one row per stratum of the hierarchy (and catch part, for lengths and ages): the hierarchy columns, `weight_t` (its landings, or its catch, in tonnes), `covered_t` (the part of it that has an estimate or a distribution of its own: what is not raised), `raised_t` (the rest), coverage (in percent), `numTrips` (and `numMeasurements`), the level, `level_label` and group it falls in - the level its `raised_t` comes from; a stratum with nothing to raise is level 1 - and what the pool it was given stands on: `pool_coverage`, `pool_trips` (and `pool_fish`) and `pool_strata`, how many strata it holds;
- `groups`: one row per group: group, level, `level_label`, its pool in words, `n_strata`, `weight_t` and `raised_t` of what it raises, and the coverage, `numTrips` (and `numMeasurements`) of its pool;
- `levels`: one row per level (and catch part): `n_groups`, `n_strata` (the strata that take the level for what they raise; at level 1 those with nothing to raise), `covered_t` (at level 1: the weight of all the strata that has an estimate of its own), `raised_t` (what the level's pools raise), `weight_t` (the two together: the weight that rests on the level) and `pct_weight`, `pct_raised` and `cum_pct_raised`: the share of the weight, the share of everything that has to be raised that the level settles, and the running total of that from the narrowest level;
- `what`, `hierarchy`, `thresholds` and `drop_catchCategory`, as they were used.

See Also

`plot_recommended_groups()` and `plot_river()` to look at how the strata fell into the levels; `build_coverage_strata()` for the numbers behind them; `build_discard_clusters()` to choose the variables; `catchRaising()` and `catchAllocationAoL()`, which take the groups.

Other raising groups: `plot_recommended_groups()`

Examples

```
# a small catch (kg): a landing has a discard estimate when it has a domain
# (domainCatchDis) that a discard row carries too
catch <- data.frame(
  catchCategory = c("Lan", "Lan", "Lan", "Lan", "Lan", "Lan", "Dis", "Dis"),
  areaValue     = c("27.4", "27.4", "27.4", "27.7.d", "27.7.d", "27.7.d", "27.4", "27.7.d"),
  mesh         = c("70-99", "70-99", "100-119", "70-99", "70-99", "100-119", "70-99", "70-99"),
  total        = c(3000, 1000, 2000, 800, 200, 500, 200, 40),
  domainCatchDis = c("d1", NA, NA, "d2", NA, NA, "d1", "d2"),
  variableUnit  = "kg", stringsAsFactors = FALSE)

# the area matters more than the mesh: the mesh is pooled across first
rec <- recommend_groups(catch, c("areaValue", "mesh"), list(cov = 50))
rec                                     # what needs no raising, and how the strata fell into the levels
```

```

rec$strata          # 70-99 stands on its own; 100-119 is raised from its area
rec$strataCond      # what catchRaising() takes
rec$matchedDataCond

## Not run:
hierarchy <- c("areaValue",      # matters most: the last one to be pooled across
              "mesh",
              "vesselFlagCountry",
              "seasonValue")      # matters least: the first one to be pooled across
thresholds <- list(cov = 50, trips = NA, fish = NA)

# the groups of the discards, and the raising they go to
rec <- recommend_groups(catch_data, hierarchy, thresholds)
rec          # how the strata fell into the levels
raised <- catchRaising(catch_data, rec$strataCond, rec$matchedDataCond,
                      type = "discards", logFile = tempfile(fileext = ".txt"))

# the groups of the lengths of the raised catch: at least 100 fish measured, the
# landings never allocated from the discards
rec_len <- recommend_groups(raised, hierarchy,
                           list(cov = 50, trips = NA, fish = 100),
                           distribution_data = distributions_data,
                           drop_catchCategory = FALSE)
alloc <- catchAllocationAoL(raised, distributions_data,
                           rec_len$strataCond, rec_len$matchedDataCond,
                           distributionType = "LengthTotal",
                           logFile = tempfile(fileext = ".txt"))

## End(Not run)

```

saving-plots

Saving a plot as it is drawn

Description

Every plot function takes a `save` argument. Left as `NULL` the plot is only returned. Give it a place to write to and the plot is written to a file as well, with the size and resolution you ask for, and returned all the same - so it still prints and can be added to.

Arguments

<code>save</code>	Where to write the plot as well, or <code>NULL</code> (default) to only return it: a folder, a file name, or a list of options - see saving-plots .
-------------------	---

Details

`save` is one of:

- a **folder**, `save = "report/figures"`: the plot is written there, named after its title plus a short code that tells apart plots with the same title (a different fill or facet, say). Drawing the same plot again writes the same file;

- a **file name** with an extension, `save = "report/figures/rates.png"`: that file. The extension (png, pdf, svg, jpeg, tiff, bmp) chooses the format;
- a **list** of options, named as in `ggplot2::ggsave()`: `path` (a folder or a file name, as above), `filename`, `width`, `height`, `units`, `dpi`, `device` and `bg`.

Unless you say otherwise the plot is 10 x 7 inches at 300 dpi on a white background, as a png. `units` is "in", "cm", "mm" or "px": with "px", width and height are the pixels of the image itself, whatever the dpi. The folder is created when it does not exist, and a file that is already there is overwritten. A folder whose name contains a dot is read as a file name: give it as `list(path = "report.v2", filename = "rates.png")`.

See Also

`ggplot2::ggsave()`, which does the writing.

Examples

```
## Not run:
fig <- file.path(tempdir(), "figures")

# into a folder, named after the plot
plot_DR(catch, x = "metier4", save = fig)

# to a file, 6 x 4 inches at 150 dpi
plot_DR(catch, x = "metier4",
        save = list(path = file.path(fig, "rates.png"),
                    width = 6, height = 4, dpi = 150))

# exactly 1600 x 900 pixels
plot_DR(catch, x = "metier4",
        save = list(path = fig, filename = "rates_wide.png",
                    width = 1600, height = 900, units = "px"))

list.files(fig)

## End(Not run)
```

Index

- * **coverage plots**
 - build_coverage_strata, 2
 - coverage-plots, 21
 - plot_coverage_map, 38
- * **discard-rate plots**
 - plot_DR, 42
- * **distribution plots**
 - plot_MWaLoA, 45
 - plot_NaLoA, 48
- * **raising groups**
 - plot_recommended_groups, 51
 - recommend_groups, 56
- * **raising plots**
 - plot_catches, 34
 - plot_compare_raised, 36
 - plot_recommended_groups, 51
 - plot_river, 53
- * **stock coordination format**
 - catchAllocation, 7
 - catchCombined, 15
 - distributionsCombined, 26
- build_coverage_strata, 2, 24, 40
- build_coverage_strata(), 7, 21, 22, 24, 39, 40, 42–44, 59, 60
- build_discard_clusters, 6
- build_discard_clusters(), 40, 41, 56, 60
- catchAllocation, 7, 17, 29
- catchAllocationAoL, 10
- catchAllocationAoL(), 13, 14, 19, 26, 27, 29, 33, 34, 45, 49, 56, 59, 60
- catchAtAoLAggregation, 13
- catchAtAoLAggregation(), 10, 12, 28, 29, 34, 51
- catchCombined, 9, 15, 29
- catchCombined(), 8, 9, 27–29
- catchRaising, 17
- catchRaising(), 4, 7–9, 12, 15, 17, 22, 35, 37, 42, 55–57, 59, 60
- check_group_conditions, 20
- check_group_conditions(), 11, 12, 18, 19
- coverage-plots, 5, 21, 39, 40, 44, 48, 51
- derive_fields, 24
- derive_fields(), 3–5, 10, 18, 21, 22, 46, 49, 57, 59
- distributionsCombined, 9, 17, 26
- distributionsCombined(), 14, 16, 17, 34
- funICoutCEF, 30
- funIntercatchCEF, 31
- funMakeRelation, 32
- ggplot2::ggsave(), 62
- meanWoLatAoLAggregation, 32
- meanWoLatAoLAggregation(), 10, 12, 14, 27–29, 48
- plot_catches, 34, 38, 53, 54
- plot_catches(), 19, 36, 38, 44, 51, 55
- plot_compare_raised, 36, 36, 53, 54
- plot_compare_raised(), 55
- plot_coverage_bar (coverage-plots), 21
- plot_coverage_bar(), 2
- plot_coverage_grid (coverage-plots), 21
- plot_coverage_grid(), 2, 3, 57, 59
- plot_coverage_map, 5, 24, 38
- plot_coverage_map(), 2, 21, 24
- plot_discard_cluster_variables, 40
- plot_discard_cluster_variables(), 6, 7, 41
- plot_discard_clusters, 41
- plot_discard_clusters(), 6, 7, 41
- plot_DR, 42
- plot_DR(), 2, 5, 19, 24, 36
- plot_MWaLoA, 45, 51
- plot_MWaLoA(), 10, 34, 50, 51
- plot_NaLoA, 48, 48
- plot_NaLoA(), 10, 14, 47, 48

`plot_recommended_groups`, [36](#), [38](#), [51](#), [54](#),
 [60](#)
`plot_recommended_groups()`, [56](#), [58](#), [60](#)
`plot_river`, [36](#), [38](#), [53](#), [53](#)
`plot_river()`, [52](#), [53](#), [56](#), [60](#)
`print.rcef_groups` (`recommend_groups`), [56](#)

`raised_overview_table`, [54](#)
`raised_overview_table()`, [19](#), [36–38](#)
`recommend_groups`, [53](#), [56](#)
`recommend_groups()`, [3](#), [7](#), [9](#), [22](#), [23](#), [51–53](#)

`saving-plots`, [22](#), [35](#), [37](#), [39–41](#), [43](#), [46](#), [50](#),
 [52](#), [54](#), [61](#), [61](#)