

Package: LandWebUtils (via r-universe)

October 10, 2026

Type Package

Title Helper functions for the LandWeb project

Version 1.0.3.9048

Date 2026-08-17

Description Additional utilities for LandWeb analyses.

License GPL-3

URL <http://LandWebUtils.predictiveecology.org>,
<https://github.com/PredictiveEcology/LandWebUtils>

Depends R (>= 4.4)

Imports data.table, dplyr, fs, future.apply, ggplot2, googledrive,
graphics, grDevices, map, ngeo, methods, pemisc, purrr,
rasterVis, reproducible, scales, sf, SpaDES.tools (>=
2.1.2.9000), spatialutils, stats, terra, tibble, tools, utils,
withr, workflowtools

Suggests bcddata, covr, DEoptim, landscapemetrics, parallel,
parallelly, patchwork, testthat (>= 3.1.9), tidyterra, VGAM

Remotes FOR-CAST/spatialutils, FOR-CAST/workflowtools@development,
PredictiveEcology/map@development,
PredictiveEcology/pemisc@development,
PredictiveEcology/reproducible@development,
PredictiveEcology/SpaDES.tools@development

Additional_repositories <https://predictiveecology.r-universe.dev>

Config/testthat/edition 3

Encoding UTF-8

Language en-CA

LazyData true

Roxygen list(markdown = TRUE)

Collate 'CASFRiv4.R' 'LandWebUtils-package.R' 'misc.R' 'boxplots.R'
'caribou.R' 'current-conditions.R' 'equivalentName.R'
'histograms.R' 'landmine-calibration-plot.R'

```
'landmine-calibration.R' 'landmine-events.R'
'landmine-fri-diagnostics.R' 'landmine-helpers.R'
'landmine-plot.R' 'landmine-ros.R' 'landmine.R'
'landweb-area.R' 'landweb-growth-curves.R'
'landweb-reporting.R' 'landweb-species-units.R'
'landweb-species.R' 'largePatches.R' 'leading.R' 'plotting.R'
'polygon-clean.R' 'rasterListByPoly.R'
'reportingPolygonCrossings.R' 'reportingPolygons.R'
'shortNames.R' 'simfiles.R' 'species-groups.R'
'studyAreaGroups.R' 'studyAreas.R' 'zzz.R'
```

Config/roxygen2/version 8.1.0

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev git make default-
jdk libarchive-dev libgit2-dev libicu-dev libjpeg-dev libpng-dev libuv1-dev libssl-dev libproj-
dev python3 libsqlite3-dev libudunits2-dev libx11-dev

Repository <https://predictiveecology.r-universe.dev>

Date/Publication 2026-10-09 21:43:20 UTC

RemoteUrl <https://github.com/PredictiveEcology/LandWebUtils>

RemoteRef development

RemoteSha 1e0867afa98dc5f2c89bdb2dcddcc5d942dd142

Contents

.cleanLandWebStudyArea	4
analysesOutputsTimes	5
build_studyarea_crosswalk	5
buildCaribouRanges	7
buildCrossedReportingPolygons	8
buildLandbasePolygons	9
buildReportingPolygons	10
caribouRangeLayers	11
CASFRItToSpRasts	13
cc_age_composite	14
cc_age_pct_missing	15
cleanAreaName	16
extractFMA	17
extractFMU	17
findSimFile	18
joinReportingPolygons	18
landbaseLayers	19
landmine_area_burned_by_zone	20
landmine_area_share	21
landmine_attach_identity	22
landmine_burn1	23
landmine_calibration_plots	25
landmine_draw_num_fires	26
landmine_estimate_kBest	27

landmine_fire_attainment	28
landmine_fire_ros	29
landmine_fri_metrics	31
landmine_fri_summary	34
landmine_fri_verdict	35
landmine_fuel_types	36
landmine_ignition_budget	36
landmine_known_species	38
landmine_optim_burnFun	39
landmine_optim_calibrate	40
landmine_optim_clusterExport	41
landmine_optim_clusterSetup	42
landmine_optim_clusterWrap	43
landmine_optim_fireSizes	43
landmine_optim_fitAndison	44
landmine_optim_fitSN	46
landmine_optim_islands	47
landmine_optim_islandTarget	48
landmine_optim_islandWeight	48
landmine_optim_landscape	49
landmine_optim_shapeIndex	50
landmine_optim_shapeTarget	51
landmine_optim_unpack	51
landmine_plot_fri_drivers	52
landmine_plot_fri_zones	53
landmine_plots	54
landmine_reburn_budget	55
landmine_reburn_ceiling	56
landmine_ros_table	57
landmine_sizes_to_pixels	57
landweb_add_report_column	58
landweb_anpp_area	59
landweb_area	59
landweb_bec_zones	60
landweb_damage_codes	60
landweb_dominant_sppEquiv	61
landweb_growth_sppEquiv	62
landweb_growth_trait_frequency	62
landweb_growth_traits	63
landweb_lthfc	63
landweb_member_cover	64
landweb_report_colours	64
landweb_report_groups	65
landweb_report_map	66
landweb_require_species	66
landweb_resample_psp	67
landweb_species_map	68
landweb_species_sppEquiv	68

landweb_species_units	69
landweb_sppEquiv	71
landweb_sum_layers	72
landweb_unit_growth_traits	73
LandWebCRS	73
LandWebStudyAreas	74
LargePatches	75
lcc2020_classes	76
lcc2020_remap_table	76
LeadingVegTypeByAgeClass	78
loadCASFRI	79
meanTruncPareto	80
plotFMA	80
polygonClean	81
prepFMAs	81
prepSpeciesLayers_CASFRI	82
prepStudyArea	82
rasterListByPoly	83
recode_cohorts	84
refCodeFor	84
reportingPolygonLayers	85
runBoxPlotsVegCover	86
runHistsLargePatches	86
runHistsVegCover	87
shortNameFor	87
sppEquiv_groups	88
studyAreaCrosswalk	88
tenureShortNames	89
validateShortNames	90
Index	91

.cleanLandWebStudyArea

Clean up the LandWeb LTHFCs

Description

Clean up the LandWeb LTHFCs

Usage

```
.cleanLandWebStudyArea(poly, minFRI = 40)
```

Arguments

poly	A polygon or character string identifying the path to polygon
minFRI	Numeric or integer, the minimum fire return interval kept. Intervals strictly below this become NA; an interval equal to minFRI is kept.

analysesOutputsTimes *Simulation timesteps for analyses*

Description

Simulation timesteps for analyses

Usage

```
analysesOutputsTimes(period, interval)
```

Arguments

period	numeric vector of length 2 corresponding to the start and end times to use for analyses.
interval	numeric indicating the interval between timesteps for analyses

Value

numeric vector of timesteps for which to run analyses

build_studyarea_crosswalk
Build the LandWeb study-area grouping crosswalk

Description

Groups the LandWeb FMA/TSA/FML boundary polygons into ecologically-coherent study areas by assigning each to its **dominant ecological unit** (an ecoregion by default) via spatial overlap, then names the group after that unit. Small, scattered forest tenures (e.g. a single small FMA) are thereby run *inside* a larger, ecologically homogeneous area, so the downstream studyAreaANPP PSP catchment is both large enough for trait estimation and ecologically appropriate (species traits can differ across ecoregions, so PSP pools should not span drastically different ecology).

Usage

```
build_studyarea_crosswalk(
  fmas,
  eco,
  eco_field = "REGION_NAM",
  res_m = 240,
  min_area_km2 = 100,
  shortNames = tenureShortNames()
)
```

Arguments

fmas	sf of the combined LandWeb FMA boundaries (e.g. prepFMAs()).
eco	sf of ecological units to group by (e.g. an ecoregions layer). Supply an eco-province or ecodistrict layer to change the grouping granularity.
eco_field	character naming the column in eco that holds each unit's name (default "REGION_NAM", the ecoregion name field).
res_m	numeric simulation pixel size in metres, used only for the mpix size estimate (default 240).
min_area_km2	numeric; member tenures smaller than this are dropped as sliver artifacts (with a message listing them), since they cannot form a meaningful study area. Default 100. Set to 0 to keep everything.
shortNames	the curated tenure short-name table (default tenureShortNames()). Every retained member must appear in it; an uncurated member is an error , since falling back to the long name is exactly the output-key collision the curated codes exist to prevent.

Details

The grouping unit is a parameter: pass a coarser layer (ecoprovinces) for fewer, larger groups, or a finer one (ecodistricts) to subdivide groups that prove too large computationally. Membership is by **verified spatial overlap against the current FMA layer** — never by company/tenure labels, which drift over time (a polygon labelled for one licensee in an older file may belong to another now).

Value

A data.frame crosswalk, one row per member tenure, with columns: group (study-area group token), fma_name (the member's coalesced FMA/TSA/FML identity — the join key back to fmas), name_short (the curated partner-facing tenure code, see [tenureShortNames\(\)](#)), company (licensee, for company-driven reruns), province, eco_unit (the dominant ecological unit), area_km2, and mpix (approx. pixels at res_m). Sort/split by group to get each study area's member set.

buildCaribouRanges	<i>Assemble the caribou range reporting layer</i>
--------------------	---

Description

Fetch each jurisdictional source in [caribouRangeLayers\(\)](#), drop locally extirpated herds, label it from its labelCols, tag its juris and boreal/mountain ecotype, clip it to studyArea with [spatialutils::prep_vector\(\)](#), and bind the intersecting pieces into one layer. Sources that do not reach the study area are dropped, so this works unchanged for a study area in any jurisdiction.

Usage

```
buildCaribouRanges(  
  studyArea,  
  destinationPath,  
  targetCRS = LandWebCRS,  
  layers = caribouRangeLayers()  
)
```

Arguments

studyArea	A SpatVector (or anything terra::vect() accepts).
destinationPath	Directory for downloads/extraction.
targetCRS	Target CRS (default LandWebCRS).
layers	Candidate-source table (default caribouRangeLayers()).

Details

Range names are normalised via an explicit spelling table (see the source of [caribouRangeLayers\(\)](#)): a range that two provinces spell differently would otherwise appear as two reporting units, since the summaries group by **name** rather than by feature. That same name-grouping is what stitches a cross-border range whose two provincial shares carry the same name — Chinchaga and Narraway — back into a single reporting unit.

Value

A SpatVector with columns Name, juris and ecotype, or NULL if no source intersects studyArea.

See Also

[caribouRangeLayers\(\)](#), [buildReportingPolygons\(\)](#)

 buildCrossedReportingPolygons

Build the tenure x sub-region crossed reporting polygons

Description

Restores the v2 per-tenure reporting units on top of v3's study-area-clipped layers. Each tenure in the tenure layer (dissolved to one geometry per tenure, see [reportingPolygonLayers\(\)](#)\$isTenure) is crossed — via [joinReportingPolygons\(\)](#) — with:

Usage

```
buildCrossedReportingPolygons(
  polys,
  landbases = list(),
  layers = reportingPolygonLayers(),
  members = NULL,
  min_area_km2 = 1
)
```

Arguments

polys	Named list of study-area-clipped reporting layers, keyed by NAME_SHORT (the output of buildReportingPolygons()); must contain the tenure layer.
landbases	Named list of per-tenure landbase coverages keyed "<tenure> <NAME_SHORT>" (the output of buildLandbasePolygons()); may be empty.
layers	Candidate-layer table (default reportingPolygonLayers()), supplying isTenure and cross.
members	optional character vector of the study area's member tenure short names (a study-area group's name_shorts from studyAreaCrosswalk()). When supplied, only members are crossed. Tenure polygons overlap in the v10 layer, so clipping to a group boundary also catches neighbouring tenures that belong to a <i>different</i> group — and since the crosswalk assigns each tenure to exactly one group, crossing them here would report a fragment of a tenure that is reported in full elsewhere. NULL (the default) crosses every tenure present.
min_area_km2	numeric; tenures contributing less than this to the study area are not crossed (default 1). A neighbouring tenure that merely grazes the study-area boundary would otherwise mint a full set of reporting units over a handful of pixels. Crossed units (all features sharing a Name) smaller than this are merged into the neighbouring unit of the same layer that they share the longest border with (see spatialutils::eliminate_slivers()); a small unit with no neighbour, or a layer made up only of small units, is dropped. 0 disables both.

Details

1. every sub-region layer marked cross in layers (Caribou ranges, Alberta Natural Subregions, BC biogeoclimatic zones, NWT/national ecoregions, national ecozones);
2. its own active/passive landbase coverage(s), if any ([buildLandbasePolygons\(\)](#));
3. and, where a tenure has both, the landbase **and** Caribou crossing together — the triple (tenure x landbase x Caribou) v2 produced for the West Fraser CLS blocks.

Products are named "<tenure> <layer NAME_SHORT>" (e.g. "SprayLake ANSR", "Tolko_Vand_WF_SL LB_S17 Caribou"), so each is a distinct reportingPolygons entry with its own refCode, parquet aggregate directory, and figure folder. Their features are named "<sub-region> <tenure>" by the concatenation in [joinReportingPolygons\(\)](#). Crossings that do not intersect are dropped.

Value

A named list of crossed layers (SpatVectors), ready to be appended to the reportingPolygons input to NRV_summary. The tenure-crossed landbases are included here, so the raw landbases list should **not** also be appended.

See Also

[buildReportingPolygons\(\)](#), [buildLandbasePolygons\(\)](#), [joinReportingPolygons\(\)](#)

buildLandbasePolygons *Build the active/passive landbase-status reporting polygons*

Description

For each candidate landbase source ([landbaseLayers\(\)](#)) whose tenure is present in tenures, download it (idempotent workflowtools *_once helpers), then dissolve it by landbase status and clip it to studyArea with [spatialutils::prep_landbase\(\)](#) – which pushes the column and spatial filters down to the read, so a large coverage is reduced to the study area before any geometry work. Sources that do not intersect the study area (or whose tenure is absent) are skipped; the result is a named list of SpatVectors keyed "<tenure> <NAME_SHORT>", each with a Name column holding the landbase status, suitable for merging into the reportingPolygons input to NRV_summary.

Usage

```
buildLandbasePolygons(
  studyArea,
  tenures,
  destinationPath,
  targetCRS = LandWebCRS,
  layers = landbaseLayers()
)
```

Arguments

studyArea	A SpatVector (or a source readable by <code>terra::vect()</code>).
tenures	Character vector of the curated tenure short names present in this study area (e.g. <code>reportingPolygons[["FMA"]]</code> \$Name, or the study-area group's name_shorts from <code>studyAreaCrosswalk()</code>). Only landbases belonging to one of these are fetched.
destinationPath	Directory for downloads/extraction.
targetCRS	Target CRS (default <code>LandWebCRS</code>).
layers	Candidate-layer table (default <code>landbaseLayers()</code>).

Value

A named list of SpatVectors, one per applicable, intersecting landbase; empty when none apply.

See Also

`buildReportingPolygons()`, `landbaseLayers()`

`buildReportingPolygons`

Build the reporting-polygons named list

Description

Fetch each candidate reporting layer (`reportingPolygonLayers()`), clip it to studyArea with `spatialutils::prep_vector()`, and keep only those that intersect – a layer that does not overlap the study area is dropped (`nrow == 0`), which replaces the former per-study-area hardcoding. Downloads use the idempotent workflowtools `*_once` helpers (bypassing `reproducible::prepInputs`). The result is a named list keyed by each layer's curated NAME_SHORT (which is what refCode is slugged from, so the on-disk key matches the label in the figures), suitable as the `reportingPolygons` input to `NRV_summary`; each layer's `labelCols` label is copied to a common Name column. The "CC SAM"/"CC TSF" (current condition) and "ecoregionLayer" entries that `NRV_summary` also expects come from the simulation, not from here, and are merged in separately.

Usage

```
buildReportingPolygons(
  studyArea,
  destinationPath,
  targetCRS = LandWebCRS,
  layers = reportingPolygonLayers(),
  min_area_km2 = 1
)
```

Arguments

studyArea	A SpatVector (or source readable by <code>terra::vect()</code>).
destinationPath	Directory for downloads/extraction.
targetCRS	Target CRS (default <code>LandWebCRS</code>).
layers	Candidate-layer table (default <code>reportingPolygonLayers()</code>).
min_area_km2	numeric; reporting units (all features sharing a Name) smaller than this within the study area are merged into the neighbouring unit of the same layer that they share the longest border with (see <code>spatialutils::eliminate_slivers()</code>), so an edge sliver does not become a reporting unit of its own. A small unit with no neighbour, or a layer made up only of small units, is dropped. Default 1; 0 disables it.

Details

The **tenure** layer (`isTenure`) is labelled differently: its Name is the curated tenure short name (`tenureShortNames()`) resolved from the coalesced v10 member identity, not the raw `labelCols`. That both keeps the long (up to 97-character) `FMA_NAMEs` out of the output keys and gives the non-Alberta members — whose `FMA_NAME` is NA — a label at all.

Value

A named list of SpatVectors, one per intersecting layer.

caribouRangeLayers	<i>Jurisdictional caribou range source layers</i>
--------------------	---

Description

The six provincial/territorial sources assembled into LandWeb’s caribou reporting layer by `buildCaribouRanges()`. LandWeb reports to **jurisdictional** partners, who each want their own management-unit names, so these are authoritative for reporting rather than any national layer.

Usage

`caribouRangeLayers()`

Details

This replaces two earlier approaches. Julie Duval’s pre-combined v10 layer (`CaribouRange_Regional_ltfc_v10_LCC1`) was assembled by hand, and the combine had **dropped the Northwest Territories ranges** — 0 of its 74 features intersect either NWT tenure — while leaving range names scattered across five per-jurisdiction columns. A GNWT-only supplement then patched the NWT hole. Assembling from the sources here removes both problems and the whole class of defect they came from: a bad merge becomes a one-row fix instead of a re-delivery.

source uses the `reportingPolygonLayers()` transports ("drive", "url", "geojson"). `labelCols` is a comma-separated priority list coalesced left to right, as there. `statusCols` names the candidate column(s) carrying herd status and extirpated the regular expression matching values to **exclude**; both NA where a source ships no status field.

Value

A tibble with columns `juris`, `key`, `source`, `id`, `labelCols`, `statusCols` and `extirpated`.

Why the ECCC national layer is not here

Measured against the assembled layer, ECCC adds no coverage: with designatable-unit scope matched (boreal + mountain), the two footprints agree to **0.7%** — the assembly divides the same ground into 74 units where the federal layer has 52. The difference is granularity and naming, not extent. ECCC's Southern Mountain units are aggregate Local Population Units that *merge* the finer provincial herds, and for ranges wholly inside one province the provincial polygon is geometrically identical to ECCC's (AB IoU = 1.0000 for 9 of 12 boreal ranges). It is retained as a documented comparison in `scripts/make_caribou_reference.R`, not as a reporting layer.

Per-source notes

- **AB** — labelled on SUBUNIT before LOCALRANGE, i.e. at **subunit granularity throughout**. This is a deliberate choice, settled 2026-08-17, and it is not a like-for-like reproduction of v2 — no single setting is. v2 reported Alberta's *mountain* ranges at subunit level (A La Pêche, Narraway and Redrock-Prairie Creek were three separate v2 units, but all three are subunits of the one LOCALRANGE "West Central") while reporting its *boreal* ranges whole — because v2 took Alberta's boreal ranges from ECCC's national layer, at range level, and only its mountain ranges from Alberta. SUBUNIT therefore matches v2 for the mountain ranges and reports **finer than v2** for exactly one boreal range: East Side Athabasca River resolves into its 7 subunits (Agnes, Algar, Bohn, Christina, Egg-Pony, Wandering, Wiau). Every other Alberta range is 1:1 with both v2 and ECCC, so this is the only place the two granularities differ; the finer breakdown is kept as an improvement. See `outputs/_reference/caribouAlberta3way.csv` for the unit-by-unit comparison, and report 03 for the v2 crosswalk.
Alberta's East Side Athabasca / West Side Athabasca are its own official domain values and are kept as-is (ECCC appends River); Bistcho is Alberta's misspelling of Bistcho and is corrected. See `.caribouNameFixes()`.
- **BC** — fetched live from the BC Data Catalogue WFS rather than a cached export. The WFS release is **newer and materially different** from the v2-era shapefile: 55 features against 72, and it flags **5** extirpated herds where the old export flagged
 1. NB the WFS uses unabbreviated field names (HERD_STATUS, RISK_STATUS) while a shapefile export truncates them to 10 characters (HERD_STAT), hence the candidate list in `statusCols`.
- **SK** — CONUNIT first, falling back to RNGEUNIT: SK1 carries the literal "Not Applicable" in CONUNIT, which `buildReportingPolygons()`'s placeholder handling treats as missing. Same footprint as ECCC (IoU 0.9983), subdivided finer.
- **MB** — by-request from the Government of Manitoba, so Drive-hosted; there is no public endpoint. Provided in **2018** and **current** — Manitoba has published nothing since, so this is the authoritative version, not a stale copy (the `_2015` in the filename is the delineation year, not

the delivery). It is also **newer than any ECCC boundary set**, and 11.9% of it falls outside every ECCC range, so it is not substitutable.

- **NWT** — GNWT MapServer layer 97, which subdivides NT1 into 6 named planning regions the way LOCALRANGE/HERD_NAME subdivide AB/BC. Labelled on REGION, not NAME: both exist, but NAME carries diacritics and apostrophes (Sahtú, Wek'èezhìi, Gwich'in) while REGION is ASCII-safe, and this label is slugged into the refCode that keys parquet aggregates and figure filenames.
- **ON** — the LIO direct download. Required, and easy to overlook: RANGE_NAME is populated for exactly the ON and MB features of the old combined layer, so omitting Ontario silently loses every ON tenure x Caribou reporting unit.

See Also

```
buildCaribouRanges(), reportingPolygonLayers()
```

CASFRtoSpRasts	CASFRtoSpRasts
----------------	----------------

Description

TODO: description and title needed

Usage

```
CASFRtoSpRasts(  
  CASFRIRas,  
  CASFRIattrLong,  
  CASFRIdt,  
  sppEquiv,  
  sppEquivCol,  
  destinationPath  
)
```

Arguments

CASFRIRas	TODO: description needed
CASFRIattrLong	TODO: description needed
CASFRIdt	TODO: description needed
sppEquiv	table with species name equivalencies between the kNN and final naming formats. See data("sppEquivalencies_CA", "LandR"). For functions that have mixedType e.g., vegTypeMapGenerator, this only necessary if mixedType == 2. If not provided and mixedType == 2, will attempt to use data("sppEquivalencies_CA", "LandR").
sppEquivCol	the column name to use from sppEquiv. For functions that have mixedType e.g., vegTypeMapGenerator, Only necessary if mixedType == 2. If not provided and mixedType == 2, will attempt to use "Boreal".
destinationPath	path to data directory where objects will be downloaded or saved to

Value

TODO: description needed

cc_age_composite	<i>Composite current-condition stand age, filling gaps from a second source</i>
------------------	---

Description

Crops the primary age raster to the study area, converts its NoData sentinel to NA, and optionally fills the cells it leaves empty from a second age raster aged forward to the same epoch. Promoted from LandWeb_preamble, where base is fRI Research's age_in2025 and fill is NTEMS forest age.

Usage

```
cc_age_composite(
  base,
  studyArea,
  fill = NULL,
  fillYear = NULL,
  epoch = 2025L,
  baseNAflag = 65535,
  fillNAflag = 255
)
```

Arguments

base	SpatRaster of stand age at epoch; authoritative wherever it has a value.
studyArea	SpatVector or sf polygon(s). Both rasters are cropped to it before any other work (see Details).
fill	optional SpatRaster of stand age at fillYear, in any CRS.
fillYear	the year fill represents. Required with fill.
epoch	the year base represents; fill is aged forward by epoch - fillYear.
baseNAflag, fillNAflag	the NoData sentinel of each source (age_in2025: 65535; NTEMS: 255, meaning non-treed). NA if the source has none.

Value

A SpatRaster on base's grid, cropped and masked to studyArea.

Fill, never minimum

base stays authoritative: fill only supplies cells base left empty. A cell-wise minimum would instead let a *modelled* age drag *inventory* ages younger, and because a minimum is one-directional that is a systematic young bias into exactly the areas where the data is best.

Sentinels

A sentinel left unconverted is read as a real age – 65,535 years, or a 255-year stand on every non-treed pixel – and silently poisons every downstream aggregate. terra does not read age_in2025's NoData tag, so the sentinel must be supplied. It is converted with `terra::classify()` after cropping rather than with `NAflag<-`: for an in-memory raster, `NAflag<-` converts the values but `NAflag()` still reports NaN, so an assertion on the flag fails on correct data, and whether the flag "took" depends on how the raster is stored.

fill's sentinel is converted **before** it is reprojected and aged, so it can never surface as sentinel + (epoch - fillYear) (e.g. 258).

Cost

Cropping first matters: age_in2025 alone is about 6.5e9 cells, so aligning a fill at full extent would move billions of cells to produce one study area's output. fill is reprojected with `method = "near"`, so the result contains only the source's own ages.

See Also

[cc_age_pct_missing\(\)](#) to check the result is complete enough to use.

cc_age_pct_missing	<i>Percent of forest with no current-condition stand age</i>
--------------------	--

Description

The completeness check for a current-condition age raster: of the forest inside the study area, what share has no age? Promoted from LandWeb_preamble's ccAgeMaxMissing gate.

Usage

```
cc_age_pct_missing(
  age,
  studyArea,
  landcover,
  forestClasses = lcc2020_classes()$forest
)
```

Arguments

age	SpatRaster of stand age, e.g. from cc_age_composite() .
studyArea	SpatVector or sf polygon(s).
landcover	SpatRaster of land cover, in any CRS and at any resolution; it is cropped and reprojected onto age's grid with <code>method = "near"</code> .
forestClasses	the landcover codes that count as forest. Defaults to the LCC 2020 forest classes from lcc2020_classes() .

Details

The denominator is the part of the study area that is forest. It took three attempts to get there, and each wrong version is pinned by a test:

- 1. **Not the raster.** Cropping with `mask = TRUE` sets every cell outside the polygon to NA, indistinguishable from a cell that genuinely lacks an age, so `mean(is.na(age))` measures the shape of the bounding box. LandWeb’s study-area groups fill only 30-44% of theirs: WesternAlber-taUpland read 71.4% missing against a true 3.9%.
- 2. **Not the whole study area.** Stand age is only defined for forest, and a treed-only source such as NTEMS leaves water and wetland empty by design, so the polygon-wide figure mostly measures lakes. ChurchillRiverUpland read 39.4% missing after filling; 76% of that was water, barren or wetland.
- 3. **Not on a coarser grid.** Aggregating with `average` fills a coarse cell from any one of its children, so a cell only counts as missing when *all* of them are. At 240 m that loosened the figure 3-20x and let an unfilled study area pass.

Counting uses `terra::global()`, which works in chunks, so no full-resolution vector is held in memory.

Value

A single number, the percent (0-100) of forest cells with no age, with attributes `nForest` and `nAged` giving the counts behind it.

<code>cleanAreaName</code>	<i>Extract study area name from run name</i>
----------------------------	--

Description

Extract study area name from run name

Usage

`cleanAreaName(area)`

Arguments

<code>area</code>	Simulated area (i.e., run) name
-------------------	---------------------------------

extractFMA	<i>Extract boundary polygon(s) for LandWeb forest management area(s)</i>
------------	--

Description

Extract boundary polygon(s) for LandWeb forest management area(s)

Usage

```
extractFMA(fmas, name)
```

Arguments

fmas	sf object with LandWeb FMA boundaries
name	A character (regex) string to match.

Value

sf polygons object

extractFMU	<i>Extract boundary polygon(s) for Alberta forest management unit(s)</i>
------------	--

Description

Extract boundary polygon(s) for Alberta forest management unit(s)

Usage

```
extractFMU(fmus, name)
```

Arguments

fmus	sf object with Alberta FMU boundaries
name	A character (regex) string to match.

Value

sf polygons object

findSimFile	<i>Find LandWeb simulation output files</i>
-------------	---

Description

Find LandWeb simulation output files

Usage

```
findSimFile(outputDir = NULL, rep = NULL)

findOldSimFiles(outputDir = NULL, rep = NULL, before = "2024-01-01")
```

Arguments

- outputDir path to LandWeb output directory.
- rep integer giving the replicate id, or character string in the form of "rep01".
- before date before which files are considered outdated (default "2024-01-01").

Value

path to the file

Examples

```
## Not run:
outDir <- file.path("~/GitHub/LandWeb/outputs/Tolko_AB_S_aspenDispersal_logROS")
if (dir.exists(outDir)) {
  oldFiles <- findOldSimFiles(outDir)      ## search all reps
  oldFiles <- findOldSimFiles(outDir, 1:5) ## search specific reps
  # fs::file_delete(oldFiles) ## double check before deleting
}

## End(Not run)
```

joinReportingPolygons	<i>Join reporting polygons and intersect their features</i>
-----------------------	---

Description

Join two reporting polygons, preserving their features; concatenate the Name.* fields into a single Name field. E.g., if x and y each contain 2 features, the resulting object will contain 4 features (corresponding to x1.y1, x1.y2, x2.y1, and x2.y2).

Usage

```
joinReportingPolygons(x, y)
```

Arguments

`x, y` an sf or SpatVector (anything `terra::vect()` accepts).

Details

The concatenated Name is what makes the crossings work downstream: both `nrvttools::calculateLandWebMetrics()` and `nrvttools::subregion_forested_area()` group by **name**, not by feature, so a sub-region split into several disjoint parts inside a tenure is tallied as one unit for free.

Value

a polygons object of the same class as `x` (sf unless `x` was a SpatVector).

Geometry repair and the intersection engine

Repair goes through `spatialutils::repair_geoms()` (`terra::makeValid()` on just the invalid subset), **not** `sf::st_make_valid()`: the latter collapses the reversed ring-winding-order polygons present in the v10 FMA layer (e.g. Prince Albert, -31,600 km² -> a 4 km² sliver) that terra correctly re-orientes.

The intersection itself uses `terra::intersect()` rather than v2's `sf::st_intersection()`. On the real v10 layers `st_intersection()` throws GEOS TopologyException: side location conflict on some already-repaired tenure x Caribou pairs (e.g. Dawson Creek TSA), which silently cost those units their reporting polygons; `terra::intersect()` handles the same geometries. Using one engine (the same one the repair already runs through) also keeps the result deterministic, rather than depending on which pairs happen to trip GEOS.

landbaseLayers

Candidate active/passive landbase-status source layers

Description

The per-FMA landbase-status ("active/passive", a.k.a. "contributing/non-contributing") coverages, each fetched, dissolved by status, and clipped to the study area by `buildLandbasePolygons()`. This generalises the per-study-area landbase processing formerly hard-coded across the LandWeb_preamble study-area helpers (SprayLake.R, WestFraser.R, Tolko.R, SundreFP.R, Manning.R).

Usage

```
landbaseLayers()
```

Details

A landbase coverage belongs to **one tenure** (not every tenure has one), so tenure holds that tenure's curated short name (`tenureShortNames()`): `buildLandbasePolygons()` fetches a source **only** when its tenure is present in the study area, avoiding the download of large (multi-GB) coverages that could not intersect it anyway. `status_col` is the source attribute holding the landbase status; `layer` names the layer for multi-layer File Geodatabases (NA = the source's only/first layer). `NAME_SHORT` discriminates a tenure's several coverages (West Fraser's three CLS supply blocks); it need only be unique **within** a tenure, since the reporting-unit name is "<tenure> <NAME_SHORT>".

Value

A tibble with columns `key`, `NAME_SHORT`, `tenure`, `source`, `id`, `layer`, and `status_col`.

Note

Gating was formerly a regex (`applies_to`) matched against the *study-area name*, which silently stopped matching anything once study areas became ecoregion groups ("WesternAlbertaUpland" matches no company name), so no landbase was ever fetched. Gating on the tenures actually present restores it.

landmine_area_burned_by_zone
Area burned per FRI zone in a single year

Description

Counts burned pixels of `currentBurn` within each FRI zone and converts to hectares.

Usage

```
landmine_area_burned_by_zone(  
  currentBurn,  
  fireReturnInterval,  
  time,  
  pixelAreaHa  
)
```

Arguments

<code>currentBurn</code>	SpatRaster of this year's burn; burned pixels are 1, unburned 0.
<code>fireReturnInterval</code>	SpatRaster of per-pixel FRI values (the zone labels).
<code>time</code>	numeric, the simulation time recorded in the returned table.
<code>pixelAreaHa</code>	numeric, the area of one pixel in hectares.

Details

Fixes a bug. This previously counted burned pixels with `unname(table(currBurn[ids])[2])` – positional indexing into a `table()`, which assumes the levels are always `{0, 1}` in that order. When a zone burns **completely** there is only one level, so `[2]` is `NA`, and the following `npix[is.na(npix)] <- 0` (whose comment says it handles the *no-burn* case) silently converted it to its exact opposite: a fully-burned zone was recorded as **zero** hectares burned. Counting with `sum(vals == 1, na.rm = TRUE)` is correct in all three cases.

Value

A `data.frame` with columns `time`, `nPixelsBurned`, `haBurned`, `FRI` (a factor).

landmine_area_share	<i>Share of total area carried by the largest fires</i>
---------------------	---

Description

Share of total area carried by the largest fires

Usage

```
landmine_area_share(sizes, topQuantile = 0.95)
```

Arguments

<code>sizes</code>	numeric vector of fire sizes.
<code>topQuantile</code>	numeric; 0.95 selects the largest 5%.

Details

Strictly greater-than, so fires exactly at the quantile are excluded. With a heavily tied distribution – which a rounded truncated Pareto is, since most draws round to 1 – that matters: the selected set can be much smaller than `1 - topQuantile` of the fires.

Value

A numeric scalar between 0 and 1.

Examples

```
landmine_area_share(c(1, 1, 1, 1, 100), 0.5)
```

landmine_attach_identity

Attach fire identity to a burn result

Description

Joins fireID/attempt/targetSize onto the cluster table spread2() returns, keyed on the fire's start cell.

Usage

```
landmine_attach_identity(fa, idDT)
```

Arguments

fa	the clusterDT from a burn – needs initialPixels.
idDT	a data.table with initialPixels, fireID, attempt and targetSize, one row per fire ignited this round.

Details

spread2() returns clusters in its own order, which is not the order start cells were handed to it, so identity must be joined rather than assigned by position. Row order of fa is preserved.

Every burned cluster must match an ignited fire. An unmatched one would carry NA identity into fireSizes, silently corrupting per-fire attainment while leaving sum(size) == sum(maxSize) intact – so nothing downstream would catch it. It is an error.

Value

fa with fireID, attempt and targetSize added.

Examples

```
fa <- data.table::data.table(initialPixels = c(12L, 11L), size = c(10, 500))
idDT <- data.table::data.table(
  initialPixels = c(11L, 12L), fireID = c(7L, 3L),
  attempt = c(1L, 2L), targetSize = c(500, 40)
)
landmine_attach_identity(fa, idDT)
```

landmine_burn1

*Core Burn function for Andison's LandMine Fire Module***Description**

The main function for the Andison Fire Module. See details.

Usage

```
landmine_burn1(
  landscape,
  startCells,
  fireSizes = 5,
  nActiveCells1 = c(10, 36),
  spawnNewActive = c(0.46, 0.2, 0.26, 0.11),
  maxRetriesPerID = 10L,
  sizeCutoffs = c(8000, 20000),
  spreadProbRel = NA_real_,
  spreadProb = 0.77,
  omitPixels = NULL
)

landmine_burn(
  landscape,
  startCells,
  fireSizes = 5,
  nActiveCells1 = c(10, 36),
  spawnNewActive = c(0.46, 0.2, 0.26, 0.11),
  sizeCutoffs = c(8000, 20000),
  spreadProbRel = 0.23
)
```

Arguments

landscape	A SpatRaster. This only provides the extent and resolution for the fire spread algorithm.
startCells	A numeric vector indicating the indices on the landscape where the fires will start with 100%% certainty.
fireSizes	A numeric vector indicating the final size of each of the fires. Must be same length as startCells.
nActiveCells1	A numeric vector of length 2. These are cutoffs above and below each of which different values of spawnNewActive are used. See details.
spawnNewActive	A numeric vector of length 4. These are the probabilities of creating spreading to 2 neighbours instead of the 1 default neighbour, each time step. The 4 values are for 4 different fire size conditions. See details.

maxRetriesPerID	Integer. Maximum number of retry attempts per firelet ID.
sizeCutoffs	A numeric vector of length 2. These are 2 size (in hectares) thresholds that affect which spawnNewActive probabilities are used. See details.
spreadProbRel	Relative probabilities, with non-flammable pixels NA: either a raster layer, or a numeric vector of length terra::ncell(landscape). For landmine_burn1(), prefer the numeric vector, for the same reason as spreadProb (needs SpaDES.tools >= 2.1.2.9000).
spreadProb	Spread probabilities, with non-flammable pixels NA: either a raster layer, or a numeric vector of length terra::ncell(landscape). For landmine_burn1(), prefer the numeric vector: spread2() re-materialises a raster spreadProb on every spread step, which is O(ncell) per step.
omitPixels	An optional vector of pixel IDs to omit from fire size calculations. Can be used if spreadProb or spreadProbRel do not designate non-flammable pixels as NA, to allow fires to move through these pixels, while excluding these pixels from burn area calculations. Useful in study areas with discontinuous fuels, which would otherwise result in fires getting 'stuck' too often, and not reach their target size.

Details

This algorithm is a modified contagious cellular automaton.

Value

A data.table with 4 columns (initialPixels, pixels, state, order).

Algorithm

Core: Each fire starts at a single pixel, startCells and will spread, i.e., visit and convert from a 0 to the fire id number. It will iteratively spread until the number of cells visited is equal to floor(fireSizes).

Adjustments due to current fire size and number of active pixels: That can vary too, but it gets a bit complicated, so use that for now. Spawning probability was originally set at 13%, but created problems with very large and very small fires, so over time has been adjusted to vary depending on: a) number of active "firelets" (NF); and b) fire size (FS), such that:

- If nActiveCells1[1] <= NF < nActiveCells1[2] and FS < sizeCutoffs[2] -> spawnNewActive[2];
- If NF >= nActiveCells1[2] and FS < sizeCutoffs[1] -> spawnNewActive[4];
- If NF < nActiveCells1[2] and FS > sizeCutoffs[2] -> spawnNewActive[3];
- otherwise (including NF < nActiveCells1[1]) -> spawnNewActive[1].

These rules create more heterogeneity in the pattern of burning.

Thresholds are given symbolically because the fitted values differ from the figures this documentation used to quote: sizeCutoffs is calibrated, and is currently 1629 / 52016 ha rather than the 8,000 / 20,000 originally described.

The second rule uses >=: with >, the NF == nActiveCells1[2] case matched no rule.

A gap remains, and is deliberately *not* patched silently: when NF >= nActiveCells1[2] and FS >= sizeCutoffs[1], no rule applies and spawnNewActive[1] is used. The deprecated landmine_burn()

set that region to zero instead, noting it is "undescribed in Andison" and that fires "look too circular" without it. Deciding between those requires a re-fit, so it is left as a documented open question.

Fire jumping: If the fire has not reached its target size, it will try to pick new neighbours among the 8 immediate neighbours up to 4 times. If it still did not find enough neighbours, then it will jump or "spot" up to 4 pixels away. It will then repeat the previous 2 stages again once (i.e., 4 neighbours, 1 jump, repeat), then it will stop, unable to achieve the desired fireSize.

Note

The original version (`landmine_burn()`) is deprecated and should not be used. Use `landmine_burn1()` instead.

landmine_calibration_plots

LandMine calibration diagnostic plots

Description

Figures for reporting what a fire-spread calibration achieved, and whether the objective can actually tell good parameters from bad. They take plain data frames so the same functions serve the module vignette, the project reports, and ad-hoc checks.

Usage

```
landmine_plot_calibShape(components, shapeRangeHa = c(10, 3000))
```

```
landmine_plot_calibIslands(components)
```

```
landmine_plot_calibDiscrimination(scores)
```

```
landmine_plot_calibParams(params, lower, upper)
```

```
landmine_plot_calibConvergence(deoptim)
```

Arguments

components	The "components" attribute of a <code>landmine_optim_fitAndison()</code> result: one row per fire size (and replicate) with areaHa, shapeObs, shapeExp, pctIslandsObs, pctIslandsExp.
shapeRangeHa	Range Andison fitted the shape relationship over; shaded on the plot, because the $\log_{10}(\text{AREA})^4$ term extrapolates sharply outside it.
scores	A data.frame with group (e.g. "fitted" / "random") and objective.
params	A matrix or data.frame of fitted parameter sets, one row per calibration run (as returned by <code>landmine_optim_params_read()</code>).
lower, upper	Search bounds used for the fit, in the same parameter order.
deoptim	A DEoptim result object (or its member\$bestvalit vector).

Details

- `landmine_plot_calibShape()` – simulated shape index against Anderson’s fitted curve;
- `landmine_plot_calibIslands()` – simulated remnant-island fraction against Anderson Table 3.5;
- `landmine_plot_calibDiscrimination()` – objective values for fitted vs random parameters;
- `landmine_plot_calibParams()` – spread of repeat fits within the search bounds;
- `landmine_plot_calibConvergence()` – DEoptim’s best value against iteration, showing whether the run had converged or was still improving when it stopped.

Value

a ggplot object.

`landmine_draw_num_fires`

Draw the number of fires per FRI zone for one period

Description

Draw the number of fires per FRI zone for one period

Usage

```
landmine_draw_num_fires(numFiresPerYear, fireTimestep = 1, size = 1.3765)
```

Arguments

<code>numFiresPerYear</code>	numeric, expected fires per year for each zone.
<code>fireTimestep</code>	numeric, years per burn event.
<code>size</code>	numeric, the negative-binomial dispersion. The default of 1.3765 is Land-Mine’s calibrated value (lowered from 1.8765 in 2018 because the draw was otherwise too constant); it is deliberately overdispersed, so a zone with a small <code>numFiresPerYear</code> sees long runs of zero-fire years punctuated by bursts.

Value

An integer vector, one count per element of `numFiresPerYear`, with names preserved.

landmine_estimate_kBest

Fit the truncated-Pareto shape for the fire-size distribution

Description

Chooses the shape parameter k so that a given share of total burned area falls in a given share of the largest fires.

Usage

```
landmine_estimate_kBest(
  biggestPossibleFireSizeHa,
  interval = c(0.05, 0.99),
  nDraws = 1e+06,
  topFireQuantile = 0.95,
  targetAreaShare = 0.95
)
```

Arguments

biggestPossibleFireSizeHa	numeric, the distribution's upper bound, in hectares.
interval	length-2 numeric, the search interval for k .
nDraws	integer, draws per objective evaluation. The objective is stochastic, so this trades run time against the precision of the fit.
topFireQuantile	numeric, the quantile defining "the largest fires" – 0.95 means the largest 5%.
targetAreaShare	numeric, the share of total burned area those fires should carry.

Details

This is a **rule of thumb, deliberately not a fit to fire data**: Dave Andison's "95% of the area is in 5% of the fires". The observational record (NBAC, NFDB) is used elsewhere, in the burn summaries, but not here – see the LandMine documentation for why that was decided.

The rule has been restated several times, and the module carried each earlier version as commented-out code. They are recorded here instead, since they are history rather than options:

- "90% of area in 5% of fires" – the original rule of thumb;
- "95% of area in 5% of fires" – Eliot's adjustment, because each year came out too constant and the regime needed more variation. This is what the defaults reproduce;
- "95% of area in 10% of fires" (2018-10-23) – a further adjustment for still more variation, which was written but never adopted.

Because the objective draws $nDraws$ random fire sizes on every evaluation, the result varies from call to call. Set a seed for reproducibility.

Value

A numeric scalar, the fitted shape parameter.

Examples

```
set.seed(1)
k <- landmine_estimate_kBest(1e6, nDraws = 1e4)
```

```
landmine_fire_attainment
```

Per-fire target attainment from a LandMine fireSizes table

Description

Collapses the per-burn rows to one row per fire and reports whether that fire ever reached the target it was ignited for.

Usage

```
landmine_fire_attainment(fireSizes, by = "fireID")
```

Arguments

<code>fireSizes</code>	a <code>data.table</code> (or coercible) with <code>fireID</code> , <code>attempt</code> , <code>targetSize</code> , <code>size</code> and <code>maxSize</code> – i.e. <code>sim\$fireSizes</code> from <code>LandMine</code> , row-bound across years.
<code>by</code>	character; the columns that identify one fire. <code>fireID</code> is issued per burn YEAR, so it is unique only within a year: a table spanning years needs <code>c("year", "fireID")</code> , and one spanning replicates <code>c("rep", "year", "fireID")</code> . Grouping too coarsely merges unrelated fires and reports a target no single fire ever had.

Details

`size == maxSize` holds for **every** row by construction, because a fire that stalls is either discarded and retried at full target, or has its `maxSize` rewritten to the area that did burn while the shortfall is re-issued as new fires. So `maxSize` cannot express a shortfall, and attainment is only meaningful per `fireID` against `targetSize`: a fire is short only when it exhausted `maxReburns[2]` and was abandoned.

`attempts` is the highest attempt index reached, so it answers "did this fire reach its target on the first try, or the fourth?".

Value

A `data.table` keyed by `fireID` with `burned` (pixels), `target` (pixels), `attempts` and `reached`.

Examples

```
fs <- data.table::data.table(
  fireID = c(1L, 2L, 2L), attempt = c(1L, 3L, 4L),
  targetSize = c(100, 200, 200), size = c(100, 60, 90), maxSize = c(100, 60, 90)
)
landmine_fire_attainment(fs)
```

landmine_fire_ros	<i>Assign a rate of spread to every pixel</i>
-------------------	---

Description

Combines a vegetation type map, a stand-age (time since fire) map and a table of rates of spread by fuel type and age class into one rate-of-spread vector, then fills the remaining flammable and non-flammable pixels with their defaults.

Usage

```
landmine_fire_ros(
  vegTypeMap,
  rstTimeSinceFire,
  flammableMap,
  ROSTable,
  sppEquiv = NULL,
  sppEquivCol = NULL,
  ROStoher,
  knownSpecies = landmine_known_species(),
  fuelTypes = NULL,
  ROStype = c("default", "burny"),
  ageCutoffs = c(young = 40, immature = 120),
  youngFilter = c("always", "legacy"),
  assertions = getOption("LandR.assertions", TRUE)
)
```

Arguments

vegTypeMap	SpatRaster of leading vegetation type, with a raster attribute table.
rstTimeSinceFire	SpatRaster of time since fire, or a numeric vector with one element per pixel of vegTypeMap.
flammableMap	SpatRaster (or numeric vector) in which 1 marks flammable pixels and 0 or NA marks non-flammable ones.
ROStable	data.table with columns age, leading, ros. The age values may be "mature", "immature", "young", or compound versions such as "immature_young" where two classes share one rate.

sppEquiv	data.table of species equivalencies, with (at least) columns LandMine, LandWeb, and sppEquivCol. Not used with fuelTypes.
sppEquivCol	character; the sppEquiv column whose values appear as the labels of vegTypeMap's attribute table. Not used with fuelTypes.
ROSother	integer; the rate of spread for flammable pixels that are not forested (grassland, lichen, shrub). Must lie within the range of ROSTable\$ros and within 5% of mature spruce.
knownSpecies	named character vector mapping species codes to fuel types; defaults to <code>landmine_known_species()</code> . Not used with fuelTypes.
fuelTypes	named character vector giving the fuel type of each label of vegTypeMap's attribute table, e.g. <code>landmine_fuel_types()</code> for a map of reporting groups. When given, labels are looked up in it directly instead of being matched to species names, and sppEquiv, sppEquivCol and knownSpecies are not used. It is an error for a label to have no fuel type.
ROStype	character; "default" leaves non-flammable pixels as NA, "burny" gives them the young-deciduous rate so that fire can cross discontinuous fuels.
ageCutoffs	length-2 numeric giving the young/immature and immature/mature stand-age boundaries, in years. These are the only place the ROSTable's age labels acquire a numeric meaning – the table itself carries no ages.
youngFilter	character; whether the young age class is always restricted to the vegetation types that have a young rate ("always"), or only under the module's original condition ("legacy", the default). See Details.
assertions	logical; run <code>.ros_self_test()</code> and the attribute-table check.

Value

An integer vector of rates of spread, one element per pixel of vegTypeMap.

The youngFilter argument

The module applied the species filter to the mature and immature classes when `!all(pixelValue %in% vegTypeIDs)`, but to the young class when `all(pixelValue %in% vegTypeIDs)` – the opposite condition. The git history shows this is an unfinished copy-paste: the line was introduced (ad5ddb3) as a copy of the mature guard that still read `sppEquiv["mature"]` in its condition while assigning to the young class, and the follow-up (b6d600c) repaired the subset key but not the dropped `!`.

Restoring the `!` is nonetheless **not** the fix. Because every `pixelValue` in the lookup comes from the attribute table, the condition is only ever asking whether the class has any rows at all, so the correct behaviour – for all three classes – is to apply the filter unconditionally. That is what "always" does.

"legacy" and "always" agree except in one case: when no vegetation type present in the landscape has a "young"-classified age row. Then "legacy" skips the filter, every young pixel is assigned NA from an empty lookup, and that NA overwrites the immature rate the pixel had already been given – so young stands silently fall through to ROSother instead of their immature_young rate. With LandMine's default ROSTable, "young" appears only for pine, so this bites on any landscape with no pine in it, and not otherwise.

The default is "always". It makes no difference on a landscape containing pine (0 of 4,458,168 pixels differ on WesternAlbertaUpland), and prevents a pine-free study area from being silently wrong. "legacy" remains available to reproduce older runs.

Attribute-table row order

The lookup is built from `which()` and `grep()` over the attribute table, which give row *positions*; those are translated into raster *values* via the table's own value column. Row position and raster value coincide as `LandR: :vegTypeMapGenerator` builds the map, but not necessarily after a `terra::writeRaster()/terra::readRaster()` round trip, which may return the rows in a different order. Code that reloads a saved `vegTypeMap` and indexes by position – as this function's predecessor in the `LandMine` module did – assigns whole species the wrong fuel type with no error. Measured on a saved `WesternAlbertaUpland` `vegTypeMap`, that meant aspen and pine, 36% of the landscape between them, taking each other's and softwood's rates of spread.

Mixedwood, and a related defect that is NOT fixed

`knownSpecies` maps species onto spruce, pine, decid and softwood only. The fifth fuel type, mixed, is invented for attribute-table entries matching no species, so it exists on one side of the species join and not the other: the join leaves its leading value NA, and the subsequent join onto `ROSTable` (keyed on leading) dropped it. Mixedwood therefore used to fall through to `ROSother` – 30 with the default table, the mature-spruce rate, at every age – instead of 12 (young/immature) or 17 (mature), leaving both mixed rows dead. leading is now filled in for that row, which is unambiguous because `knownSpecies` can never produce "mixed".

A related defect is **deliberately left alone**: the pattern-based fuel typing can disagree with `knownSpecies`. `Lari_spp` matches none of `Pice/Pinu/Popu`, so it is typed softwood, while `knownSpecies` calls larch decid; the two disagree, the join finds no match, and larch also falls through to `ROSother`. It is not swept up by the fix above because the pattern-derived value would override the authoritative mapping, which is the wrong direction. Larch is 3 pixels on `WesternAlbertaUpland`. A test pins the behaviour so a future change is deliberate. With `fuelTypes` the labels are mapped explicitly, so neither defect can arise there: larch is a spruce fuel within its reporting group, and "Mixed" is listed.

landmine_fri_metrics *Per-zone fire-return-interval diagnostics*

Description

Extends `landmine_fri_summary()` with the structural measurements needed to tell *why* a fire-return-interval zone misses its target, and classifies each zone against a tolerance.

Usage

```
landmine_fri_metrics(
  lthfc,
  flammableMap,
  meanAnnualCumulBurnMap,
```

```

    studyAreaName,
    pixelAreaHa,
    studyArea = NULL,
    vegTypeMap = NULL,
    vegTypeMapInit = NULL,
    lccMap = NULL,
    treeClasses = NULL,
    tolerance = c(0.8, 1.25),
    severe = c(0.5, 2),
    minIgnitions = 20L,
    nYears = NA_real_
  )

```

Arguments

lthfc	SpatRaster of target fire return intervals (the "LTHFC").
flammableMap	SpatRaster; 1 is flammable, 0 and NA are not.
meanAnnualCumulBurnMap	SpatRaster of mean annual burn counts per pixel.
studyAreaName	character, recorded in the returned table.
pixelAreaHa	area of one pixel, in hectares.
studyArea	optional SpatVector/sf polygon of the area the fire model was actually allowed to burn. Supplying this is strongly recommended ; see Details.
vegTypeMap	optional SpatRaster of leading vegetation type. When supplied, a pctNoCohorts column reports the share of each zone's flammable pixels that carry no cohorts at all – non-forest land, not young forest. See Details.
vegTypeMapInit	optional SpatRaster of leading vegetation type at the START of the run. Supplying it separates land that never had cohorts from land that lost them, via pctCohortsInit and pctCohortsLost.
lccMap	optional SpatRaster of land cover class, on the same grid as lthfc.
treeClasses	integer vector of lccMap values that count as treed. Required with lccMap.
tolerance	length-2 numeric; achieved/target ratios outside this range are flagged. Symmetric in log space by default.
severe	length-2 numeric; ratios outside this range are flagged "severe".
minIgnitions	integer; a zone expected to see fewer ignitions than this over the whole run is reported but not judged, because its achieved FRI is dominated by sampling noise. Set NA to judge every zone.
nYears	length of the run in years, used with minIgnitions. NA skips the noise check.

Details

ratio is **achieved / target**, so a value above 1 means the zone burns *less* often than intended (under-burning) and below 1 means more often. That orientation is what the diverging fill in [landmine_plot_fri_zones\(\)](#) is centred on.

Value

A data.table, one row per zone, sorted by LTHFC, with columns studyArea, LTHFC, FRI, ratio, status, areaHa, nTotal, nFlam, pctFlam, nPatches, largestPatchPct, pctNoCohorts, pctCohortsInit, pctCohortsLost, pctTreedLCC, cohortGap, pctInStudyArea, FRIunmasked, ratioUnmasked and expIgnitions. The columns depending on an optional input are NA when it is not supplied.

Why studyArea matters

LandMine masks both its rate-of-spread map and its spread probability to `sim$studyArea`, so pixels outside that polygon cannot be ignited or spread into. The fire-return-interval raster is not similarly masked, so it commonly extends past the polygon – and `landmine_fri_summary()` divides burns by **every** flammable pixel carrying an interval, unburnable ones included. Each such pixel inflates a zone's achieved interval without contributing any opportunity to burn.

This is not a small correction. On WesternAlbertaUpland, 988,437 of 3,361,240 flammable zone pixels (29.4%) lay outside the polygon and burned at a mean rate of 0.51 against 16.59 inside. Uncorrected, two zones appeared to under-burn by 3.5x and 12.2x; with the denominator restricted to the burnable area, every zone came in between 0.90 and 1.07 – there was no attainment problem at all. Supply `studyArea` and both figures are reported, `ratio` from the burnable area and `ratioUnmasked` the way the module's own summary computes it, so the discrepancy is visible rather than mistaken for a fire-model failure.

The structural columns exist because the failure modes are not distinguishable from the ratio alone:

- `pctFlam` – the fraction of the zone that can carry fire at all. `spread2()` treats a non-flammable neighbour as a hard mask, so a porous zone loses perimeter growth every iteration and its fires stall short of their target size. On WesternAlbertaUpland the two failing zones were the only ones below 93% flammable (54.6% and 57.2%), which no previously-reported metric would have shown – the landscape-wide figure was 89%.
- `nPatches / largestPatchPct` – these separate *porous* from *fragmented*. A zone can be half non-flammable and still be one connected sheet (zone 170's largest patch held 86.8% of its flammable pixels), which is a very different problem from a zone broken into islands smaller than the fire-size draw.
- `pctNoCohorts` – on WesternAlbertaUpland this tracked the shortfall more closely than anything else: every zone below 5% hit its target (ratios 0.96-1.00), while the two worst zones were at 87.0% and 99.9%. Note it means *non-forest land*, not young forest: those same zones had the OLDEST stands in the study area (median time since fire 777 and 1067 years), so a young-fuel explanation is ruled out – high stand age is a consequence of under-burning, not its cause.
- `cohortGap = pctTreedLCC - pctCohortsInit` – treed land that never received cohorts when the run was initialised. This separates "the zone genuinely is not forest" from "the zone is forest that was never populated", which the other columns cannot tell apart, and which imply completely different fixes. On WesternAlbertaUpland the gap was 0.1-2.1 points in every zone that met its target and 10.7-33.1 points in every zone that did not; cohort *loss* over 1000 years was at most 13.4% anywhere and ~0% in the two worst zones, so the shortfall is an initialisation property, not regeneration failure.

See Also

`landmine_fri_verdict()`, `landmine_plot_fri_zones()`

landmine_fri_summary *Summarize achieved fire return interval by FRI zone*

Description

For each distinct value of lthfc (the target fire return interval, held as a raster of per-pixel FRI values), computes the achieved FRI as the reciprocal of the mean annual burn rate over that zone's **flammable** pixels.

Usage

```
landmine_fri_summary(
  lthfc,
  flammableMap,
  meanAnnualCumulBurnMap,
  studyAreaName,
  studyArea = NULL
)
```

Arguments

lthfc	SpatRaster of target fire return intervals (the "LTHFC").
flammableMap	SpatRaster; pixels that are NA or 0 are non-flammable and are excluded from both the numerator and the denominator.
meanAnnualCumulBurnMap	SpatRaster of mean annual burn counts per pixel.
studyAreaName	character, recorded in the returned table.
studyArea	optional SpatVector/sf polygon of the area the fire model was permitted to burn. Supply this. See Details.

Details

This was duplicated verbatim in the LandMine module's single- and multi-mode summary functions, so the two could silently diverge.

Value

A data.table with columns studyArea, LTHFC, FRI, sorted by LTHFC.

studyArea is not optional in practice

Without it, the achieved interval is computed over every flammable pixel carrying a fire return interval – including pixels outside the polygon that LandMine masks ROSmap and mod\$spreadProb to, which therefore cannot burn at all. On WesternAlbertaUpland, 988,437 of 3,361,240 flammable zone pixels (29.4%) were outside, burning at a mean rate of 0.51 against 16.59 inside, and one zone had 73 burnable pixels out of 146,627. That made two zones look as though they under-burned by 3.5x and 12.2x; with the denominator restricted to the burnable area every zone came in between

0.90 and 1.07, i.e. there was no shortfall to explain. The argument defaults to NULL only so that existing callers keep their previous (wrong) numbers until they are updated deliberately.

Two edge cases are **preserved as-is** rather than "fixed", because changing them would change published outputs:

- a zone in which nothing burned yields $1/0 = \text{Inf}$, not NA and not a large finite number;
- the burn total is a plain `sum()` with no `na.rm`, so an NA burn pixel inside a zone makes that whole zone's achieved FRI NA. Masking all three rasters by `flammableMap` is what normally prevents this – an implicit contract that the NA masks of `lthfc` and `meanAnnualCumulBurnMap` agree. Both behaviours are covered by tests so a future change to either is deliberate.

landmine_fri_verdict *One-line verdict on fire-return-interval attainment*

Description

Summarises `landmine_fri_metrics()` as a single sentence, for the end of a run's log and for a figure caption.

Usage

```
landmine_fri_verdict(metrics, maxListed = 5L)
```

Arguments

<code>metrics</code>	the <code>data.table</code> from <code>landmine_fri_metrics()</code> .
<code>maxListed</code>	maximum number of failing zones to name.

Details

Failing zones are named worst-first, each with its ratio and its flammable fraction, because those two numbers together are usually enough to tell a porosity-limited zone from one that is genuinely mis-parameterised. When `pctNoCohorts` is available it is named too, since it has been the better predictor of the two.

Value

A length-1 character vector.

landmine_fuel_types	<i>Reporting group to fuel type, for LandMine</i>
---------------------	---

Description

The LandMine fuel type of each LandWeb reporting group ([landweb_report_groups\(\)](#)) and of "Mixed". With species simulated separately, LandMine types fuel from the leading reporting group: biomass is summed within a group before the leading type is decided, as for the merged groups before.

Usage

```
landmine_fuel_types()
```

Details

White spruce and black spruce (with the larches) are spruce fuels, pines are pine, the true firs and Douglas-fir are softwood, and broadleaves are deciduous. For the merged groups LandWeb simulated before, this matches [landmine_known_species\(\)](#), except for larch: that table calls it deciduous, but larch never reached that rate (see [landmine_fire_ros\(\)](#)) and was given the mature-spruce rate at every age. It is now a spruce fuel, with black spruce.

Value

A named character vector; names are reporting-group codes and "Mixed", values are fuel types of [landmine_ros_table\(\)](#).

See Also

[landmine_fire_ros\(\)](#), whose fuelTypes takes it.

Examples

```
landmine_fuel_types()[["Bl_Spruce"]] ## "spruce"
```

landmine_ignition_budget	<i>Ignition budget per FRI zone</i>
--------------------------	-------------------------------------

Description

Masks ineligible pixels out of the fire-return-interval raster, tabulates pixels per FRI zone, and converts that to an expected number of fires per year per zone.

Usage

```
landmine_ignition_budget(
  fireReturnInterval,
  flammableMap,
  kBest,
  biggestPossibleFireSizeHa,
  studyArea = NULL
)
```

Arguments

fireReturnInterval	SpatRaster whose pixel values are the target fire return interval. Zero-valued and non-flammable pixels are set to NA.
flammableMap	SpatRaster; pixels that are NA or 0 are non-flammable.
kBest	numeric, the fitted truncated-Pareto shape for the fire-size distribution.
biggestPossibleFireSizeHa	numeric, the distribution's upper bound, in hectares.
studyArea	optional SpatVector/sf polygon of the area the fire model can actually burn. Supply this. See Details.

Details

The budget is (zone area / mean fire size) / FRI, so the expected area burned per year in a zone equals area / FRI by construction – provided each fire ignites in the zone, reaches its drawn size, and burns inside the zone.

This function creates an ordering contract that the reburn loop then consumes positionally. Downstream, numFiresThisPeriod[.GRP] indexes the per-zone fire counts by the *group position* of a polygonNumeric-keyed table, which is valid only because:

1. terra::freq() returns rows sorted ascending by value;
2. the value = NA row is appended **last**;
3. dividing by the NA return interval makes that entry's **value** NA, so a later na.omit() drops it – an NA *name* alone would not.

Break any one and zones are handed each other's fire counts, with no error. All three are pinned by tests here, at the point where the contract is created.

Value

A list with:

fireReturnInterval the masked raster.

fireReturnIntervalsByPolygonNumeric FRI per zone, ascending, with a trailing NA.

numFiresPerYear expected fires per year per zone, named by FRI, in the same order.

studyArea is not optional in practice

LandMine masks both its rate-of-spread map and spreadProb to sim\$studyArea, so a fire ignited outside that polygon burns its own start cell and can spread no further. The fire return interval raster is not similarly masked and routinely overhangs the polygon, so without studyArea those pixels inflate each zone's area and therefore its expected fires per year – and, because the start-cell pool is built from this same raster, they are drawn as ignition points too.

On WesternAlbertaUpland this allocated **2,918 fires per year against a correct 2,015 – 31% of every year's ignitions aimed at ground the model cannot burn**. One zone, 99.95% outside the polygon, was over-allocated by a factor of **2,008** (45.6 fires per year against 0.023).

Non-flammable pixels are masked out of the raster *before* tabulation, so the budget is a flammable-area budget and matches how achieved FRI is measured in [landmine_fri_summary\(\)](#). It also means such pixels can never be ignition locations, since the start-cell pool is built from this same masked raster.

landmine_known_species

Leading-species to fuel-type lookup used by LandMine

Description

The mapping from LandWeb-column species codes to the four LandMine fuel types (spruce, pine, decid, softwood) that [landmine_fire_ros\(\)](#) joins the ROSTable on.

Usage

```
landmine_known_species()
```

Details

The LandMine module previously defined this inline in `init()` and used it in two places, one of them inside `fireROS()`. Both now call this, so the fuel-type assignment cannot drift between them.

Note the keys are LandWeb-column codes, **not** LandR-column codes.

Value

A named character vector; names are species codes, values are fuel types.

Examples

```
landmine_known_species()[["Pice_mar"]] ## "spruce"
```

landmine_optim_burnFun

LandMine burn optimization function

Description

LandMine burn optimization function

Usage

```
landmine_optim_burnFun(
  ros,
  centreCell,
  fireSize,
  spawnNewActive,
  sizeCutoffs,
  spreadProb
)
```

Arguments

ros	SpatRaster of LandMine Rate Of Spread values.
centreCell	See startCells in landmine_burn1() .
fireSize	See fireSizes in landmine_burn1() .
spawnNewActive	A numeric vector of length 4. These are the probabilities of creating spreading to 2 neighbours instead of the 1 default neighbour, each time step. The 4 values are for 4 different fire size conditions. See details.
sizeCutoffs	A numeric vector of length 2. These are 2 size (in hectares) thresholds that affect which spawnNewActive probabilities are used. See details.
spreadProb	Spread probabilities, with non-flammable pixels NA: either a raster layer, or a numeric vector of length <code>terra::ncell(landscape)</code> . For <code>landmine_burn1()</code> , prefer the numeric vector: <code>spread2()</code> re-materialises a raster <code>spreadProb</code> on every spread step, which is $O(ncell)$ per step.

Value

named list of length 2 containing:

- `burnedMap`: SpatRaster of burned pixels;
- `LM`: `data.frame` of patch statistics from `landscapemetrics`.

landmine_optim_calibrate

Run the LandMine fire-spread calibration

Description

Wraps the DEoptim fit that was previously written inline in LandMine.Rmd. Defaults reproduce that fit exactly.

Usage

```
landmine_optim_calibrate(
  pixelSize = 240,
  n = 1000L,
  file = tempfile(fileext = ".tif"),
  fireSizesHa = c(50, 200, 600, 1500, 3000),
  fireSizes = NULL,
  desiredPerimeterArea = 0.003,
  lower = c(-2, -3, -3, -3, 1, 3.5, 0.75),
  upper = c(-0.1, -0.5, -0.5, -1, 3.5, 5, 1),
  NP = NULL,
  itermax = 200L,
  VTR = 0.001,
  strategy = 6L,
  cl = NULL,
  nodes = NULL,
  seed = NULL,
  objective = c("andison", "sn"),
  crnSeed = 1L,
  replicates = 1L,
  paramsFile = NULL
)
```

Arguments

pixelSize, n, file	Passed to landmine_optim_landscape().
fireSizesHa	Target fire sizes in hectares , converted to pixels for the current pixelSize by landmine_optim_fireSizes() . Defaults sit inside the 10–3,000 ha range Andison fitted.
fireSizes	Target fire sizes in pixels . Normally NULL; supply it only to reproduce a historical fit that was specified in pixels, and note that a pixel-count vector means a different physical fire at each resolution.
desiredPerimeterArea	Target perimeter-to-area ratio (m/m ²); objective = "sn" only.
lower, upper	Search bounds for the 7 parameters.

NP	DEoptim population size. Defaults to the number of cluster nodes.
itermax, VTR, strategy	Passed to <code>DEoptim::DEoptim.control()</code> .
cl	An existing cluster, or NULL to build one from nodes.
nodes	Node specification for <code>landmine_optim_clusterSetup()</code> .
seed	Optional integer seed, set before cluster creation so the per-worker RNG streams are reproducible. Note that reproducibility also requires NP to be held fixed, since the stream count follows the number of workers.
objective	Which objective to optimise. "andison" (default) uses <code>landmine_optim_fitAndison()</code> , which fits the shape-vs-area relationship and the remnant island fractions Andison (1996) reported. "sn" uses the original <code>landmine_optim_fitSN()</code> , retained so historical fits can be reproduced.
crnSeed	Passed to <code>landmine_optim_fitAndison()</code> as common random numbers. Strongly recommended: the objective is stochastic, and without it the optimiser compares parameter vectors evaluated against <i>different</i> random draws.
replicates	Passed to <code>landmine_optim_fitAndison()</code> : evaluations to average per fire size. The island statistic is highly variable between draws, so replicates > 1 buys discrimination at linear cost. objective = "andison" only.
paramsFile	Optional path to a calibration CSV; if given, the best parameters found are appended to it.

Value

The DEoptim result object.

Note

The objective is stochastic, so treat any single fit as one draw rather than a point estimate, and pass crnSeed. The original objective (`landmine_optim_fitSN()`) did not identify the parameters at all: in 43 repeat fits recorded in `LandMine_DEoptim_params.csv`, four of seven scattered over more than 90% of their search interval, and repeat solutions proved statistically indistinguishable from one another – and from some randomly drawn parameter vectors. objective = "andison" adds constraints to address that; verify identifiability again after any change to the objective rather than assuming it.

landmine_optim_clusterExport

Export objects used for optimization and load packages on cluster

Description

Export objects used for optimization and load packages on cluster

Usage

```
landmine_optim_clusterExport(cl = NULL, objs = NULL, pkgs = NULL)
```

Arguments

cl	a cluster object or NULL
objs	character vector of names of objects to export
pkgs	character vector of packages to pre-load on the cluster nodes

Value

NULL. Invoked for its side effects.

landmine_optim_clusterSetup
Setup a cluster for LandMine optimization

Description

Setup a cluster for LandMine optimization

Usage

```
landmine_optim_clusterSetup(nodes = NULL, seed = NULL)
```

Arguments

nodes	positive integer of length 1 specifying the number of threads to use on the current machine (localhost), or a character vector of hostnames on which to run worker copies.
seed	Optional integer passed to <code>parallel::clusterSetRNGStream()</code> , so the per-worker RNG streams are reproducible. Without it the seed is drawn from ambient RNG state, which made calibration runs irreproducible unless the caller happened to call <code>set.seed()</code> first. Note reproducibility also requires a fixed worker count, since one stream is assigned per worker.

Value

a cluster object

landmine_optim_clusterWrap

Wrapper function to setup cluster, export objects and load packages

Description

Wrapper function to setup cluster, export objects and load packages

Usage

```
landmine_optim_clusterWrap(cl = NULL, nodes, reps, objs, pkgs)
```

Arguments

cl	a cluster object or NULL
nodes	positive integer of length 1 specifying the number of threads to use on the current machine (localhost), or a character vector of hostnames on which to run worker copies.
reps	integer. number of replicates to run.
objs	character vector of names of objects to export
pkgs	character vector of packages to pre-load on the cluster nodes

Value

named list of length 2 containing: cl, a cluster object; out, a list of burn maps (aka burnMapList).

landmine_optim_fireSizes

Fire sizes for calibration, in pixels, from areas in hectares

Description

The calibration targets are defined in **hectares** (Andison fitted the shape relationship over 10–3,000 ha), but the fire model takes sizes in **pixels**. Specifying pixel counts directly therefore means different physical fires at every resolution: c(10, ..., 1e5) pixels is 10–100,000 ha at 100 m but 57.6–576,000 ha at 240 m. That also defeats the point of calibrating at several pixel sizes, which is to check the fitted parameters transfer across resolutions – only meaningful if the fires compared are the same physical size.

Usage

```
landmine_optim_fireSizes(
  pixelSize,
  areaHa = c(50, 200, 600, 1500, 3000),
  shapeRangeHa = c(10, 3000)
)
```

Arguments

pixelSize	Pixel resolution, in metres.
areaHa	Fire sizes, in hectares.
shapeRangeHa	Range over which the shape target is supported; a warning is issued for areas outside it, since the $\log_{10}(\text{AREA})^4$ term extrapolates sharply.

Value

Integer vector of fire sizes in pixels, with areaHa and haPerPixel attributes.

Examples

```
landmine_optim_fireSizes(240)          ## v3 resolution
landmine_optim_fireSizes(100)         ## the resolution originally calibrated at
```

```
landmine_optim_fitAndison
```

Andison-constrained objective for the LandMine fire-spread calibration

Description

A replacement for `landmine_optim_fitSN()` that fits against the validation statistics Andison (1996) actually reported, rather than a single constant perimeter-to-area target.

Usage

```
landmine_optim_fitAndison(
  par,
  ros,
  centreCell,
  fireSizes = 10^(2:5),
  shapeIntercept = 1.77,
  shapeSlope = 0.041,
  minIslandSizeHa = 2,
  islandRangeHa = c(100, Inf),
  replicates = 1L,
  wShape = 1,
  wSize = 1,
  wIsland = NULL,
  crnSeed = NULL
)
```

Arguments

par	Numeric vector of length 7; see landmine_optim_unpack() .
ros	Path to the landscape raster (a path, not a SpatRaster: these are not serializable to cluster workers).
centreCell	Ignition cell.
fireSizes	Target fire sizes, in pixels.
shapeIntercept, shapeSlope	Coefficients for landmine_optim_shapeTarget() .
minIslandSizeHa	Passed to landmine_optim_islands() .
islandRangeHa	Length-2 range of fire areas over which the island term is applied. Outside it the term is dropped, because the target is not supported there: Andison "did not bother simulating many fires less than 100 hectares since disturbances below this size are relatively constant and simple in shape, and generally do not produce remnant islands", and the empirical island data (DeLong and Tanner 1995) covers 159–2,239 ha. A small fire cannot satisfy the target at all – a 9-pixel fire has no room for an interior hole, so it scores the maximum penalty regardless of the parameters, which is noise rather than signal.
replicates	Number of independent evaluations to average per fire size. The island statistic in particular is highly variable between draws, so replicates > 1 buys discrimination at linear cost.
wShape, wSize, wIsland	Weights on the three components. Set one to 0 to drop it. wIsland defaults to landmine_optim_islandWeight() for the landscape's own resolution, because a coarse grid cannot form the small islands that carry most of Andison's island area. Pass an explicit value to override.
crnSeed	Optional integer. If given, the RNG is seeded as crnSeed + i before the i th fire, so every parameter vector is evaluated against the same random draws (common random numbers). This is variance reduction, not determinism: it makes comparisons between parameter vectors far less noisy.

Details

[landmine_optim_fitSN\(\)](#) penalises departure from one constant desiredPerimeterArea at *every* fire size. Andison's central result, though, is that shape complexity **increases with fire area** – that is what the $\log_{10}(\text{AREA})^4$ term describes – so the old objective works against the relationship it exists to reproduce. Combined with a size penalty that never fired, it reduced to a single statistic constraining seven parameters, which left the fitted values unidentifiable: across 43 repeat fits, four of seven scattered over more than 90% of their search range, and repeat solutions were statistically indistinguishable from one another (and from some randomly drawn parameter vectors).

This objective sums three squared relative deviations per fire size:

- **shape** – observed vs [landmine_optim_shapeTarget\(\)](#);
- **size** – relative shortfall against the target size, continuous rather than a step, so the optimiser gets a direction instead of a cliff;
- **islands** – percent of fire area in remnant islands vs [landmine_optim_islandTarget\(\)](#).

Value

A single numeric objective value, with the per-fire components attached as the "components" attribute for diagnostics.

See Also

[landmine_optim_fitSN\(\)](#) for the original objective, retained so historical fits can be reproduced.

landmine_optim_fitSN *LandMine objective functions*

Description

landmine_fitSN() is used for the module.

Usage

```
landmine_optim_fitSN(
  sna,
  ros,
  centreCell,
  fireSizes = 10^(2:5),
  desiredPerimeterArea = 0.004
)

landmine_optim_fitSN2(
  par,
  ros,
  centreCell,
  fireSizes = 10^(2:5),
  desiredPerimeterArea = 0.003,
  spreadProb = 0.9
)
```

Arguments

sna	(i.e., spawnNewActive) A numeric vector of length 4. These are the probabilities of creating spreading to 2 neighbours instead of the 1 default neighbour, each time step. The 4 values are for 4 different fire size conditions. See details in landmine_optim_burnFun() .
ros	Character, specifying the file path to raster of LandMine Rate Of Spread values.
centreCell	Integer id of the centre (start) cell of ros raster. See startCells in landmine_burn1() .
fireSizes	A numeric vector indicating the final size of each of the fires. See landmine_burn1() .
desiredPerimeterArea	Numeric target perimeter-area ratio.
par	parameter vector of length 5
spreadProb	spread probability

Details

landmine_fitSN2() is an alternative version tries the optimization using fewer parameters, to test whether a simpler version gets better/different results. Although this version was not used for the final module, we preserve it here for posterity.

Value

Summary data.table of fit results.

See Also

[landmine_burn1\(\)](#), [landmine_optim_burnFun\(\)](#)

landmine_optim_islands

Remnant islands within a simulated fire

Description

Unburned patches fully enclosed by the fire. Their formation is an emergent by-product of the spread algorithm rather than something the model targets directly, which is why Andison (1996, §3.3.2) used them as a validation statistic.

Usage

```
landmine_optim_islands(burnedMap, minIslandSizeHa = 2)
```

Arguments

burnedMap	A SpatRaster in which burned cells are non-NA (as returned by landmine_optim_burnFun()).
minIslandSizeHa	Islands smaller than this are ignored, matching Andison, who dropped islands below 2 ha because the model could not represent them and the inventory data did not resolve them.

Details

Unburned patches are found with 4-connectivity, the topological complement of the 8-connected spread used to grow the fire. A patch touching the raster edge is exterior, not an island; if the fire reaches the edge, no enclosure can be determined for that side.

Value

A list with nIslands, islandAreaHa (total), fireAreaHa (burned plus enclosed islands, i.e. the area within the fire perimeter), percentIslands, and sizesHa.

landmine_optim_islandTarget
Andison island-remnant target

Description

Percent of total fire area expected to be in remnant islands, by disturbance size class (Andison 1996, Table 3.5, **empirical** column from DeLong and Tanner 1995). The model column of that table (3.3 / 6.2 / 10.5) is the previous model's output, not a target.

Usage

```
landmine_optim_islandTarget(fireAreaHa)
```

Arguments

fireAreaHa Fire area, in hectares.

Value

Expected percent of fire area in remnant islands.

landmine_optim_islandWeight
Resolution-dependent weight for the remnant-island term

Description

Andison (1996) rasterised at **50 m (quarter-hectare) pixels** – his spatial database was 260 x 422 = 114,920 quarter-hectare pixels. LandWeb runs much coarser, and remnant islands are the statistic that suffers most, because a coarse grid cannot form small islands at all:

Usage

```
landmine_optim_islandWeight(pixelSize, refPixelSize = 50)
```

Arguments

pixelSize Pixel resolution, in metres.
 refPixelSize Resolution the target statistics were measured at; defaults to Andison's 50 m.

Details

resolution	ha/pixel	smallest possible island
Andison 50 m	0.25	0.25 ha
LandWeb 100 m	1.00	1 ha
LandWeb 120 m	1.44	1.44 ha
LandWeb 240 m	5.76	5.76 ha

Andison's Table 3.6 puts **65%** of island area in the 2–5 ha class for fires under 1,000 ha (24% for larger fires). At 240 m a single pixel is already 5.76 ha, so that entire class is unreachable by construction and the island target cannot be met however the parameters are set. Weighting the term down with coarsening resolution stops the optimiser chasing a target the grid forbids, at the expense of statistics it can actually reproduce.

Value

A weight in (0, 1]: $\min(1, \text{refPixelSize} / \text{pixelSize})$. 1.0 at 50 m, 0.42 at 120 m, 0.21 at 240 m.

Note

This is a deliberate, documented judgement, not a derived quantity. The linear ratio is a pragmatic choice; the representability argument above justifies *down-weighting*, not this particular functional form. Record the weight used alongside any fit.

Examples

```
landmine_optim_islandWeight(c(50, 100, 120, 240))
```

```
landmine_optim_landscape
```

Build the synthetic landscape used for LandMine calibration

Description

The calibration is deliberately run on a uniform, fully-flammable square rather than a real study area, so that the fitted shape statistics are not confounded by a particular landscape's fuel pattern. It is run at several pixel sizes to confirm the parameters transfer across resolutions.

Usage

```
landmine_optim_landscape(
  pixelSize,
  n = 1000L,
  file = tempfile(fileext = ".tif")
)
```

Arguments

pixelSize	Pixel resolution, in map units (m).
n	Number of pixels per side. Default 1000, i.e. a 1e6-cell landscape.
file	Optional path to write the raster to. The objective function takes the landscape as a <i>file path</i> , because SpatRaster objects cannot be serialized to cluster workers.

Value

A list with file (path), ros (the SpatRaster), and centreCell (the cell index at the centre of the landscape, used as the ignition point).

Examples

```
lscape <- landmine_optim_landscape(pixelSize = 240, n = 100)
terra::ncell(lscape$ros)
```

landmine_optim_shapeIndex

Andison's shape index

Description

The dimensionless shape index used by Andison (1996, §3.3.1): the patch perimeter relative to that of a circle of the same area, so a circle scores 1 and more convoluted shapes score higher.

Usage

```
landmine_optim_shapeIndex(perimeter, area)
```

Arguments

perimeter	Patch perimeter, in metres.
area	Patch area, in square metres.

Value

Numeric shape index.

Note

Perimeters measured on a raster follow cell edges, so they exceed the perimeter of the equivalent vector polygon and inflate this index relative to the GIS-derived values Andison fitted. The bias is systematic, so it shifts the intercept rather than the slope.

Examples

```
## a circle scores 1
landmine_optim_shapeIndex(perimeter = 2 * pi * 100, area = pi * 100^2)
```

landmine_optim_shapeTarget

Andison's expected shape index for a given fire area

Description

SHAPE = intercept + slope * log10(AREA)^4, fitted by Andison (1996) over fires of 10–3,000 ha. The default intercept is the one fitted to the **empirical** data (1.770, $R^2 = 0.670$, $n = 928$). The value 1.881 that also appears in the thesis is the fit to the *model's own* output, and targeting it would be circular.

Usage

```
landmine_optim_shapeTarget(areaHa, intercept = 1.77, slope = 0.041)
```

Arguments

areaHa	Fire area, in hectares.
intercept, slope	Coefficients; defaults are Andison's empirical fit.

Value

Expected shape index.

Examples

```
landmine_optim_shapeTarget(c(10, 100, 1000)) ## ~1.8, ~2.1, ~5.1
```

landmine_optim_unpack *Unpack LandMine fire-spread calibration parameters*

Description

The DEoptim calibration fits seven parameters on a transformed scale: the four spawnNewActive probabilities and the two sizeCutoffs are fitted as base-10 logarithms, while spreadProb is fitted directly. This function is the single definition of that convention – it was previously repeated at eight call sites.

The calibration results live in a CSV (LandMine_DEoptim_params.csv) whose schema is date, pixelSize, then par1..par7. These accessors are the single definition of that schema; previously it was written in one place and read in two others, and the two writers disagreed (one appended, one overwrote).

Usage

```
landmine_optim_unpack(par)

landmine_optim_params_read(file, rowID = NULL)

landmine_optim_params_append(
  file,
  par,
  pixelSize,
  objective = NA_character_,
  date = format(Sys.time(), "%Y-%m-%d")
)
```

Arguments

par	Numeric vector of length 7 to append (see <code>landmine_optim_unpack()</code>).
file	Path to the calibration CSV.
rowID	Optional row number to extract. If NULL (default), all rows are returned.
pixelSize	Pixel resolution (m) the calibration was run at.
objective	Name of the objective the fit was made against. Recorded because parameter values are only comparable within one objective: fits made against <code>landmine_optim_fitSN()</code> before its size penalty was repaired, or before fire sizes were specified in hectares, are not comparable to anything fitted since.
date	Date of the run, as a string. Defaults to today.

Value

A named list with elements `spawnNewActive` (length 4), `sizeCutoffs` (length 2), and `spreadProb` (length 1), all on the natural scale.

`landmine_optim_params_read()` returns a named numeric vector of length 7 when `rowID` is given, otherwise a matrix with one row per calibration run.

`landmine_optim_params_append()` invisibly returns the full table as written.

Examples

```
landmine_optim_unpack(c(-0.22, -1.55, -0.65, -1.48, 3.21, 4.72, 0.856))
```

```
landmine_plot_fri_drivers
```

Plot what separates the failing zones from the rest

Description

Achieved/target ratio against each structural measurement that is available, one point per zone. This is the figure that makes a porosity-limited zone, a genuinely non-forest one, and one that is forest but was never populated with cohorts distinguishable at a glance – three failure modes the ratio alone cannot separate.

Usage

```
landmine_plot_fri_drivers(metrics, studyAreaName)
```

Arguments

`metrics` the data.table from `landmine_fri_metrics()`.
`studyAreaName` character, used in the title.

Value

A ggplot.

landmine_plot_fri_zones

Map fire-return-interval zones and their attainment

Description

Two panels side by side: the study area's zones filled by their **target** fire return interval, and the same geometry filled by **achieved/target ratio** on a diverging scale centred on 1.

Usage

```
landmine_plot_fri_zones(  
  lthfc,  
  metrics,  
  studyAreaName,  
  caption = NULL,  
  maxCells = 5e+05  
)
```

Arguments

`lthfc` SpatRaster of target fire return intervals.
`metrics` the data.table from `landmine_fri_metrics()`.
`studyAreaName` character, used in the title.
`caption` optional character; defaults to `landmine_fri_verdict()`.
`maxCells` display resampling budget passed to `tidyterra::geom_spatraster()`; the default keeps a full-resolution study area readable without rendering every cell.

Details

The two panels encode different things and therefore carry **separate fill scales**: panel A is a magnitude (one hue, light to dark), panel B a polarity (two hues either side of a neutral midpoint). Putting both in one faceted plot would force a single shared scale, on which the log-ratios – which span about one unit – collapse entirely against target intervals spanning 170 years, rendering panel B a flat block of colour.

Panel B's fill is $\log_2(\text{ratio})$ so that burning twice as often and half as often sit equally far from centre, with the legend labelled in plain ratios.

Value

A patchwork object (which prints and ggsaves like a ggplot).

See Also

[landmine_fri_metrics\(\)](#)

landmine_plots	<i>LandMine diagnostic plots</i>
----------------	----------------------------------

Description

- `landmine_plot_areaBurnedOverTime()` plots the area burned over time by LTHFC polygon;
- `landmine_plot_LTHFC()` produces a [rasterVis::levelplot\(\)](#) a map of the LTHFC polygons;
- `landmine_plot_FRI()` plots

Usage

```
landmine_plot_areaBurnedOverTime(areaBurnedOverTime)
```

```
landmine_plot_LTHFC(lthfc, studyAreaName, ...)
```

```
landmine_plot_FRI(friSummary)
```

Arguments

<code>areaBurnedOverTime</code>	Summary data.frame of area burned over time, containing the following columns: time (numeric) gives the simulation time (year); haBurned (numeric) gives the burned area in hectares; FRI (numeric) identifies the fire return interval polygon.
<code>lthfc</code>	long-term historic fire cycle map (raster).
<code>studyAreaName</code>	study area name (character).
<code>...</code>	additional arguments passed to rasterVis::levelplot() .

`friSummary` Summary `data.frame` of simulated fire return intervals (FRI) vs. the long-term historic fire cycles (LTHFC), containing the following columns: `simArea` (character) gives the study area name; `LTHFC` (numeric) is the expected (i.e., historic) FRI; `FRI` (numeric) is the simulated FRI.

Value

a `ggplot2` or `rasterVis` object; invoked for side effect of creating plots.

`landmine_reburn_budget`

Rebuild the per-zone fire budget for the next reburn round

Description

After a burn round, decides how many fires each FRI zone still needs and at what target sizes, for both reburn phases.

Usage

```
landmine_reburn_budget(tooSmallByPoly, friByPolygon, remainingSize = NULL)
```

Arguments

`tooSmallByPoly` a `data.table` of the fires that did not reach their target, joined to their ignition zone: needs `polygonNumeric`, `maxSize`, and (optionally) `pixel`. May also carry the fire-identity columns `fireID`, `attempt` and `targetSize`, which are threaded through to the returned vectors so a re-ignited fire stays linked to the one it replaces.

`friByPolygon` the per-zone FRI vector from `landmine_ignition_budget()` – ascending, with a trailing NA. Its order **defines** the output order.

`remainingSize` optional numeric; phase 2 passes the *remaining shortfall* per fire, which replaces `maxSize`. Phase 1 leaves it NULL, keeping each fire’s full original target.

Details

Three positional contracts live here, none of which errors when broken:

- the right join is driven by `friByPolygon`, so the result – and hence `numFiresThisPeriod` – is in **that** order (zones ascending, NA row last). The caller indexes it with `[.GRP]` against a `polygonNumeric`-keyed table whose groups are also ascending, and the trailing NA row is never reached because that table has no NA zone.
- `na.omit()` on the whole table is load-bearing: it drops both the zero-shortfall zones (`maxSize` is NA where the join found no match) and the NA-FRI row, leaving `fireSizesInPixels` paired **positionally** with the start cells the caller then draws by `= polygonNumeric`. It would also silently drop a row if any other column went NA. The identity vectors come from that **same** `na.omit()`, so they cannot drift out of step with the sizes – but an NA in any identity column would drop that fire entirely.

- remainingSize is assigned **by position** onto the too-small rows; both descend from the same `fa[tooSmall]` subset, so they agree – but nothing enforces it.

remainingSize can be 0 for a fire that burned exactly its target; the caller's `fireSizesInPixels > 0` filter is what removes those.

Unlike the inline code this replaces, the input table is **not** modified by reference.

Value

A list with `polysNeedMoreFires` (the joined table), `numFiresThisPeriod` (fires still needed per zone, in `friByPolygon` order), `fireSizesInPixels` (target sizes, positionally paired with the start cells to be drawn), and `fireIDs/attempts/targetSizes` – the identity of those same fires, paired the same way, or `NULL` when the input carried none.

landmine_reburn_ceiling

Summarize FRI zones still short of target at the reburn ceiling

Description

Summarize FRI zones still short of target at the reburn ceiling

Usage

```
landmine_reburn_ceiling(polysNeedMoreFires, year)
```

Arguments

<code>polysNeedMoreFires</code>	a <code>data.table</code> with columns <code>polygonNumeric</code> (the FRI zone), <code>N</code> (fires still needed) and <code>maxSize</code> (pixels still to burn).
<code>year</code>	numeric, the simulation year.

Details

Emitted only when the reburn loop exits on the `maxReburns` ceiling rather than because every fire reached its target – the two are otherwise indistinguishable in a log, and the second means the year's area-burned target was abandoned unmet.

`pixelsShort` means different things in the two reburn phases: in phase 1 `maxSize` is a fire's *full* target, in phase 2 it is the *remaining shortfall*. The caller is responsible for passing the table from the phase it means.

Value

A `data.table` with columns `year`, `FRI`, `nFires`, `pixelsShort`, containing only zones with at least one outstanding fire; zero rows when nothing is short.

landmine_ros_table	<i>LandMine's rate-of-spread table</i>
--------------------	--

Description

Rates of spread by stand age class and fuel type, from Table 3.2 of Anderson (1996): the ROSTable that [landmine_fire_ros\(\)](#) joins onto each pixel.

Usage

```
landmine_ros_table()
```

Details

Anderson's water, non-productive brush and non-productive black spruce classes are omitted. His young mixed wood = 6 is a typo for young hardwood = 6, and is entered as immature_young decid.

LandWeb_preamble and LandMine each defined this table inline; both now call this. The table is the same for the "default" and "burny" ROSTypes, which differ only in whether non-flammable pixels can spread fire.

Value

A data.table with columns age (age class), leading (fuel type) and ros (integer rate of spread), one row per age class and fuel type.

See Also

[landmine_known_species\(\)](#) for the species behind each fuel type.

Examples

```
landmine_ros_table()
```

landmine_sizes_to_pixels	<i>Convert fire sizes in hectares to whole pixels, preserving the expectation</i>
--------------------------	---

Description

Convert fire sizes in hectares to whole pixels, preserving the expectation

Usage

```
landmine_sizes_to_pixels(sizesHa, pixelAreaHa)
```

Arguments

sizesHa numeric, fire sizes in hectares.
 pixelAreaHa numeric, the area of one pixel in hectares.

Details

Truncating to whole pixels would systematically lose every fire smaller than one pixel – at a 6.25 ha pixel, a 3.25 ha fire is "not detectable". Rounding the fractional part **probabilistically** instead keeps $E[\text{pixels}] == \text{sizesHa}/\text{pixelAreaHa}$, so the fire-size distribution's mean survives the discretisation even though any individual small fire may round to zero. Fires that do round to zero are dropped by the caller.

Exact multiples of pixelAreaHa never round up, because the fractional part is 0 and $0 > \text{runif}(1)$ is FALSE.

Value

A numeric vector of whole pixel counts, the same length as sizesHa.

landweb_add_report_column

Add the LandWeb reporting groups to a species equivalency table

Description

Adds a column giving each row's reporting group, looked up from the code it is simulated as.

Usage

```
landweb_add_report_column(
  sppEquiv,
  col = "LandWeb",
  reportCol = "LandWebReport"
)
```

Arguments

sppEquiv data.table species equivalency table with column col, e.g. from [landweb_sppEquiv\(\)](#). It is not modified.
 col character; the column holding the simulated codes.
 reportCol character; the name of the new column.

Value

A copy of sppEquiv with column reportCol. It is an error for a row's simulated code to have no reporting group, or for reportCol to exist already.

See Also

[landweb_report_map\(\)](#)

landweb_anpp_area	<i>Ecological units that touch an area, for fitting growth curves</i>
-------------------	---

Description

The polygons of the national ecological framework, at ecozone or ecoprovince level, that intersect area, kept whole. LandWeb_preamble builds each study area's studyAreaANPP this way from the buffered study area; the shared growth-curve fit uses the ecoprovinces that touch the LandWeb area.

Usage

```
landweb_anpp_area(
  area,
  ecoLevel = c("ecoprovince", "ecozone"),
  destinationPath
)
```

Arguments

area	sf, sfc or SpatVector polygons.
ecoLevel	"ecoprovince" or "ecozone".
destinationPath	directory the framework's shapefile is downloaded to.

Value

sf of the intersecting polygons, in the coordinate reference system of area.

landweb_area	<i>The outline of the LandWeb area</i>
--------------	--

Description

The outline of the LandWeb area

Usage

```
landweb_area(lthfc)
```

Arguments

lthfc	the cleaned LTHFC polygons: polygonClean() of landweb_lthfc() .
-------	---

Value

sfc polygon: the union of lthfc, made valid, without holes.

landweb_bec_zones	<i>BC BEC zones over the BC plots of a growth-curve fitting area</i>
-------------------	--

Description

Fetches the biogeoclimatic (BEC) zone polygons from the BC Data Catalogue over the BC provincial plots inside area, for Biomass_speciesParameters' hybrid spruce relabelling (its input BECzonesBC). The module fetches them itself when they are not supplied, but a failed fetch there only warns and skips the relabelling, which changes the spruce curves; here it stops.

Usage

```
landweb_bec_zones(PSPgis, PSPmeasure, area)
```

Arguments

PSPgis	sf of plot locations, with OrigPlotID1 (from PSPclean::getPSP()).
PSPmeasure	data.table of tree measurements, with OrigPlotID1 and source.
area	the fitting area (sf or sfc).

Value

sf of BEC polygons with a ZONE column, or NULL if no BC plot lies in area.

landweb_damage_codes	<i>Damage agent codes to exclude from growth-curve plots, by plot source</i>
----------------------	--

Description

LandWeb simulates neither bark beetles nor defoliators, so a growth curve fitted on trees they damaged would read their damage as growth and senescence. BC and the NFI code damage agents alike: a leading I for insects, then B for bark beetles and D for defoliators; BC lists each species (IBM mountain pine beetle, IDE spruce budworm, ...) and the budworm genus as CHX. The BC codes are those of BC's damage agent table as PSPclean reads it (BCForestry_DamageAgentCodes.csv). Alberta records damage causes as numbers (1 spruce budworm, 2 defoliator, 3 mountain pine beetle); Saskatchewan records only the cause of a tree's death, with all insects as 3, so either class excludes every tree that insects killed there.

Usage

```
landweb_damage_codes(agents = c("barkBeetles", "defoliators"))
```

Arguments

agents damage agent classes: any of "barkBeetles" and "defoliators".

Value

list named by plot source (BC, AB, SK, NFI), for PSPclean::getPSP()'s codesToExclude.

landweb_dominant_sppEquiv

Give each merged LandWeb species group its dominant member's traits

Description

Where a LandWeb species group merges several species, keeps only the trait code (LANDIS_traits) of the member with the most cover, and blanks the others, so that LandR::prepSpeciesTable() and LandR::speciesTableUpdate() take the group's traits from that one member.

Usage

```
landweb_dominant_sppEquiv(sppEquiv, cover)
```

Arguments

sppEquiv data.table from [landweb_sppEquiv\(\)](#), with SCANFI, LANDIS_traits and LandWeb columns. It is not modified.

cover named numeric vector of total cover by SCANFI code, as from [landweb_member_cover\(\)](#).

Details

Both LandR functions give a group the minimum of each numeric trait across its members' LANDIS_traits codes. For LandWeb's Pice_gla (white, Engelmann and hybrid spruce) that meant Engelmann spruce's 30 m effective seed dispersal for a group that is mostly white spruce (100 m). A member's cover is summed over the rows that share its LANDIS_traits code (the hybrid spruce shares Engelmann spruce's), so rows with the dominant code keep it. A group with one trait code, or with no cover at all in the study area, is left unchanged. Rows without a SCANFI code (see [landweb_sppEquiv\(\)](#)) contribute no cover, and keep their code only if it is the dominant one.

Value

A copy of sppEquiv with the non-dominant members' LANDIS_traits set to NA. Attribute "dominant" is a data.table of each group's dominant trait code and its share of the group's cover.

landweb_growth_sppEquiv

Species table for the shared growth-curve fit

Description

[landweb_species_sppEquiv\(\)](#) with the hybrid spruce counted as Engelmann spruce, so that `Biomass_speciesParameters` relabels BC provincial plot records of white spruce outside the boreal BEC zones as the hybrid (its `relabelHybridSprucePSP()` requires the hybrid row's `LandR` code to be `Pice_eng`). The hybrid row also takes Engelmann spruce's `LANDIS_traits` code, so that Engelmann spruce's traits do not change by merging it.

Usage

```
landweb_growth_sppEquiv(sppEquiv)
```

Arguments

`sppEquiv` data.table from [landweb_species_sppEquiv\(\)](#).

Value

A copy of `sppEquiv`.

landweb_growth_trait_frequency

How often refits on resampled plots chose each species' growth-curve traits

Description

Summarises the refits of the shared growth curves on bootstrap samples of plots ([landweb_resample_psp\(\)](#)): for each species, each distinct set of the five traits a growth curve sets, with how many refits chose it and whether the fit to all plots chose it too.

Usage

```
landweb_growth_trait_frequency(resampled, fitted)
```

Arguments

`resampled` data.table of [landweb_growth_traits\(\)](#) from every refit, with the refit's number in `resample`.

`fitted` [landweb_growth_traits\(\)](#) of the fit to all plots.

Value

data.table with one row per species and set of traits: species, the five traits, growthCurveSource, n (refits that chose the set), share (n over all refits) and fullFit (TRUE for the set the fit to all plots chose). Sorted by species, then by decreasing n.

landweb_growth_traits	<i>The growth-curve traits of each species, from the shared fit</i>
-----------------------	---

Description

The growth-curve traits of each species, from the shared fit

Usage

```
landweb_growth_traits(species)
```

Arguments

species	the shared growth-curve stage’s species table: one row per species, after Biomass_speciesParameters has fitted the species it can and given the others their hardwood or softwood means (growthCurveSource "imputed").
---------	--

Value

data.table with species, the five traits a growth curve sets (growthcurve, mortalityshape, mANPPproportion, inflationFactor, longevity) and growthCurveSource.

landweb_lthfc	<i>The LandWeb long-term historic fire cycle (LTHFC) map</i>
---------------	--

Description

Downloads the v10 LTHFC map once, reads it, projects it, and keeps its fire cycle as LTHFC and each polygon’s area in m^2 as area. Prepare it with [polygonClean\(\)](#); [landweb_area\(\)](#) takes its outline.

Usage

```
landweb_lthfc(destinationPath, targetCRS = LandWebCRS)
```

Arguments

destinationPath	directory under which the archive is downloaded and extracted (in lthfc/).
targetCRS	coordinate reference system to project to.

Details

The download needs Google Drive credentials (the LandWeb service account in a pipeline session).

Value

SpatVector.

landweb_member_cover	<i>Total SCANFI cover of each member species in a study area</i>
----------------------	--

Description

Sums the per-species SCANFI cover layers that the speciesData stage writes for a study area (SCANFI_spsCC_<CODE>_<year>_...tif), one value per SCANFI code.

Usage

```
landweb_member_cover(dir)
```

Arguments

dir directory holding the SCANFI_spsCC_*.tif layers.

Value

A named numeric vector of total cover (summed cover percent over the layer's cells); names are SCANFI codes. It is an error for dir to hold no such layer.

See Also

[landweb_dominant_sppEquiv\(\)](#)

landweb_report_colours	<i>Colours for the LandWeb reporting groups</i>
------------------------	---

Description

Colours for the LandWeb reporting groups

Usage

```
landweb_report_colours(groups = landweb_report_groups()[["code"]])
```

Arguments

groups character; reporting-group codes to include ([landweb_report_groups\(\)](#)).

Value

A named character vector of colours for groups and "Mixed", in the order of [landweb_report_groups\(\)](#): the form `LandR::vegTypeMapGenerator()` takes as colors, whose names must be exactly the groups in its `sppEquiv` plus "Mixed".

Examples

```
landweb_report_colours(c("Pine", "Decid"))
```

```
landweb_report_groups    LandWeb reporting groups
```

Description

The species groups LandWeb reports leading vegetation and large patches by.

Usage

```
landweb_report_groups()
```

Details

These are the species groups of the original LandWeb current-condition data (Silvacom, 2018), as listed in the "Notes" of each "Percent Cover by" table of its data dictionary: White Spruce (white, Engelmann and hybrid spruce), Black Spruce (black spruce and the larches), Pine (all pines), Fir (the true firs) and Deciduous (all broadleaves). Douglas-fir, which the dictionary puts under Fir, is reported on its own, as in the v2 Spray Lake runs. Western redcedar and western hemlock, which the dictionary does not list, are reported as Fir: LandWeb simulates them with the firs (see [landweb_species_map\(\)](#)).

label is the v2 output label of each group. Type is the group's conifer or broadleaf class, which `LandR::vegTypeMapGenerator()` reads to find mixedwood stands (`mixedType = 2`); every member of a group has the same Type in `LandR::sppEquivalencies_CA`. The colours are those of the LandWeb figures, checked for colour-vision deficiency; see [landweb_report_colours\(\)](#).

Value

A data.table with one row per group and columns code, label, Type and colour.

See Also

[landweb_report_map\(\)](#), [landweb_add_report_column\(\)](#), [landweb_report_colours\(\)](#)

Examples

```
landweb_report_groups()
```

landweb_report_map	<i>Simulated species and groups to LandWeb reporting groups</i>
--------------------	---

Description

The reporting group (`landweb_report_groups()`) of every code LandWeb simulates: single species, by their LandR code, and the merged species groups of `landweb_species_map()`.

Usage

```
landweb_report_map()
```

Value

A named character vector; names are simulated codes, values are reporting-group codes.

See Also

`landweb_add_report_column()`

Examples

```
landweb_report_map()[c("Lari_lar", "Pinu_spp")] ## "Bl_Spruce" "Pine"
```

landweb_require_species	<i>Stop when a species input is empty</i>
-------------------------	---

Description

Checks that a species input handed between LandWeb pipeline stages is not empty, and returns it unchanged so the check can wrap the reference inline.

Usage

```
landweb_require_species(x, name)
```

Arguments

x	the input to check: a <code>data.frame</code> (e.g. <code>sppEquiv</code> , <code>cohortData</code>), a <code>SpatRaster</code> (e.g. <code>speciesLayers</code>), or a named list holding one of those as element name (e.g. the result of <code>SpaDES.targets::sim_objects()</code>).
name	character; the object's name, used to find it in a list and in the error message.

Details

The upstream Biomass_* modules and LandR treat a study area with no tree species as valid: an empty species table, NULL species layers or an empty cohortData make them skip their work and finish without error. For LandWeb every study area is forested, so an empty species input can only mean a broken species mapping, and a run would otherwise "succeed" with no vegetation dynamics. This turns that into an error at the stage boundary.

Value

x, unchanged.

Examples

```
landweb_require_species(data.frame(species = "Pice_mar"), "sppEquiv")
```

landweb_resample_psp *Resample the plots of a growth-curve fit, whole plots with replacement*

Description

A bootstrap sample of the plots inside the fitting area, for refitting the shared growth curves to see how much the fitted traits depend on which plots happen to have been measured. As many plots as lie inside area are drawn with replacement, each with all its measurements and trees. A plot drawn more than once is copied, and each copy's plot and measurement IDs get a suffix (_r1, _r2, ...), so that Biomass_speciesParameters counts the copies as separate plots. area selects plots as the module does (PSPgis[studyAreaANPP,]), so plots the fit would not use are not drawn.

Usage

```
landweb_resample_psp(psp, area, seed)
```

Arguments

psp	list with PSPmeasure, PSPplot and PSPgis, from PSPclean::getPSP().
area	the fitting area (sf or sfc), given to Biomass_speciesParameters as studyAreaANPP.
seed	integer seed for the draw. The global random number generator is left as it was.

Value

psp, with the drawn plots only.

landweb_species_map *SCANFI species to LandWeb species groups*

Description

The mapping from SCANFI species codes to the seven species groups LandWeb simulates: Abie_spp, Lari_spp, Pice_gla, Pice_mar, Pinu_spp, Popu_spp and Pseu_men.

Usage

```
landweb_species_map()
```

Details

Every SCANFI species found in the LandWeb study areas is mapped, except POPU_GRA, whose SCANFI layer is unreliable.

Western redcedar (THUJ_PLI) and western hemlock (TSUG_HET) merge into Abie_spp, the closest shade-tolerant softwood, rather than being simulated as distinct species. Neither is among the original Silvacom current-condition species groups, and in Alberta they look like SCANFI over-attribution (western hemlock is about 25% of the Spray Lake FMA, well outside its range). Engelmann spruce and its hybrid with white spruce merge into Pice_gla, and paper birch into Popu_spp.

Value

A named character vector; names are SCANFI codes, values are LandWeb species groups.

See Also

[landweb_sppEquiv\(\)](#), which adds these groups to a species equivalency table.

Examples

```
landweb_species_map()[["TSUG_HET"]] ## "Abie_spp"
```

landweb_species_sppEquiv
 SCANFI species to LandWeb species, one code per species

Description

Like [landweb_sppEquiv\(\)](#), but the simulation column LandWeb holds each species' own LandR code, and the merged group goes to LandWebGroup. [landweb_species_units\(\)](#) then decides, per study area, which species are common enough to simulate on their own.

Usage

```
landweb_species_sppEquiv(sppEquiv)
```

Arguments

sppEquiv data.table species equivalency table with SCANFI and LandR columns, normally LandR::sppEquivalencies_CA. It is not modified.

Details

The rows are those [landweb_sppEquiv\(\)](#) keeps, in the same order. Hybrid white x Engelmann spruce keeps its own code, Pice_eng_gla; [landweb_species_units\(\)](#) puts it with whichever parent has more cover in the study area.

Every row of a species carries the labels and colour of its first row, so LandR names it the same way whichever row it looks up. Every row of *Pinus contorta* – shore pine, var. *latifolia* and the generic row – takes interior lodgepole pine’s (PINU_CON_LAT) labels, colour and LANDIS_traits code. LandR lists shore pine first, so its labels would otherwise name the species, and shore pine’s trait code has no rows for the ecozones LandWeb reads. Western larch shares tamarack’s colour in LandR and gets its own here, since LandR::sppColors() falls back to a generated palette when two species share a colour.

Value

A copy of the rows of sppEquiv that map to a LandWeb species, with columns LandWeb (the species’ LandR code), LandWebGroup (its group in [landweb_species_map\(\)](#)) and LandWebReport (its reporting group, [landweb_report_map\(\)](#)).

See Also

[landweb_species_units\(\)](#)

landweb_species_units *Which species LandWeb simulates on its own in a study area*

Description

Splits a species out of its merged LandWeb group where it holds at least threshold percent of the study area’s tree cover; the group’s other species stay together. Returns everything a study area’s simulation needs to use the result: its species table, colours, and which species layers make up each simulated unit.

Usage

```
landweb_species_units(
  sppEquiv,
  layers,
  studyArea = NULL,
  threshold = 1,
  floor = 5,
  neverSplit = c("Thuj_pli", "Tsug_het")
)
```

Arguments

sppEquiv	data.table from landweb_species_sppEquiv() .
layers	SpatRaster of percent cover, one layer per LandWeb code: the speciesData stage's speciesLayers for the study area.
studyArea	SpatVector or sf polygon(s) the shares are measured over, normally the reporting study area; NULL uses every cell of layers.
threshold	numeric; percent of the study area's tree cover a species (or a group's remainder) needs to be simulated on its own.
floor	numeric; Biomass_borealDataPrep's minCoverThreshold.
neverSplit	character; species never simulated on their own (western redcedar and western hemlock, whose traits are unreviewed).

Details

Shares are measured as Biomass_borealDataPrep will see the layers. A candidate unit's layer is the sum of its species' layers, rounded; pixels whose total cover is at most floor are left out; cells of a unit with at most floor cover are dropped; and each pixel is rescaled to 100. A unit's share is then its mean percent of a pixel's tree cover over studyArea.

The rule, for each merged group (LandWebGroup):

1. hybrid spruce (Pice_eng_gla) joins whichever parent, white or Engelmann spruce, has more cover in studyArea, before the floor;
2. a member with a share of at least threshold is simulated on its own;
3. the members left form the group's remainder: one unit if its share is at least threshold, otherwise merged into the group's largest split-out member, and otherwise (none split out) dropped, its cover going to the other species of each pixel;
4. species in neverSplit take no part in 2-3: they join the largest unit of their group at the end, or are dropped if their group has none.

A unit is named by its species' code, or by its group's code when it holds several species. It takes its labels and colour from that species or group, and the traits of its member with most cover: the other members' LANDIS_traits codes are blanked (see [landweb_dominant_sppEquiv\(\)](#)), and hybrid rows take their parent's code.

A species unit's growth curve is fitted on that species' permanent sample plot trees only; a group unit's on all its members'. Biomass_speciesParameters assigns plot trees to units by Latin_full,

so the rows of a species unit's merged minor members (hybrid spruce included), and of species in neverSplit, have it blanked: their trees still count in plot biomass but do not shape the curve. On WesternAlbertaUpland, merged Engelmann and hybrid spruce plots from the montane ecozone made white spruce's pooled curve unidentifiable.

Value

A list:

- sppEquiv: the study area's species table, LandWeb holding each row's unit; rows of dropped species are removed;
- sppColorVect: named colours of the units and "Mixed";
- sppColorVectReport: named colours of the reporting groups present and "Mixed";
- layerMap: named character vector, each layer of layers to its unit (NA: dropped), for [landweb_sum_layers\(\)](#);
- units: data.table of each unit's members, dominant member and share (percent), and ranked, a list column of its members in the order it takes growth-curve traits from (most cover first, species in neverSplit last; see [landweb_unit_growth_traits\(\)](#));
- hybridParent: the species hybrid spruce joined.

See Also

[landweb_sum_layers\(\)](#)

landweb_sppEquiv	<i>Add the LandWeb species groups to a species equivalency table</i>
------------------	--

Description

Adds a LandWeb column to a LandR-style species equivalency table, giving each SCANFI species its LandWeb species group, and relabels the groups that merge several species.

Usage

```
landweb_sppEquiv(sppEquiv)
```

Arguments

sppEquiv	data.table species equivalency table with SCANFI and LandR columns, normally LandR::sppEquivalencies_CA. It is not modified.
----------	--

Details

Groups that merge species take a group label in EN_generic_short, EN_generic_full and Leading: Abie_spp, Lari_spp, Pice_gla, Pinu_spp, Popu_spp and Pseu_men. Abie_spp is labelled "Fir", although western redcedar and western hemlock are merged into it too (see [landweb_species_map\(\)](#)). Pice_mar holds one species, so its row keeps its own labels.

Consumers such as Biomass_core's leading-vegetation maps look a group's label up from its first row, so a group without a label would be named after whichever of its species comes first.

A row without a SCANFI code is kept when another row of the same species (the same LandR code) maps, and joins that row's group. LandR lists some species under a generic name as well as by variety, and gives only one of them a SCANFI code: the generic *Pinus contorta* row has none, while var. *latifolia* and var. *contorta* do. Generic *Pinus contorta* takes its group and colour from var. *latifolia* (PINU_CON_LAT, interior lodgepole pine), not from whichever variety comes first. Biomass_speciesParameters assigns PSP trees to groups by their Latin_full in this table and discards trees whose name is missing, and the NFI records lodgepole pine as *Pinus contorta* and black cottonwood as *Populus trichocarpa*. Without these rows neither contributed to the growth curves fitted for Pinu_spp and Popu_spp. A row with a SCANFI code that LandWeb does not use (coastal Douglas-fir, PSEU_MEN_MEN) stays out.

These rows add no SCANFI layer, and they come after the rows mapped by SCANFI code, so the first row of each group is unchanged. One with no colorHex takes the colour of its species' mapped row: LandR: : sppColors() uses the table's colours only when every row has one.

Value

A copy of sppEquiv restricted to the rows that map to a LandWeb species group, by SCANFI code or through their species (see Details), with a new LandWeb column. Rows mapped by SCANFI code come first. It is an error for no row to map (see [landweb_require_species\(\)](#)).

See Also

[landweb_species_map\(\)](#)

landweb_sum_layers	<i>Sum species cover layers into the units LandWeb simulates</i>
--------------------	--

Description

Sum species cover layers into the units LandWeb simulates

Usage

```
landweb_sum_layers(layers, layerMap)
```

Arguments

- | | |
|----------|---|
| layers | SpatRaster of percent cover, one layer per species (named by its LandWeb code). |
| layerMap | named character vector from landweb_species_units() : each layer to its unit; NA drops the layer. |

Value

A `SpatRaster` with one layer per unit, the sum of its species' layers, named by unit. Cells that are NA in every layer stay NA. Summing these layers is exactly what `Biomass_speciesData` does when it merges species (it resamples and thresholds each SCANFI layer, then sums them, with no cap or rescaling).

landweb_unit_growth_traits	<i>Give each simulated unit the growth-curve traits of a member species</i>
----------------------------	---

Description

Each unit of a study area takes, from the shared fit, the traits of its member with most cover among those whose curve was fitted (`growthCurveSource` "estimated"); a unit none of whose members was fitted takes its dominant member's traits, which are then hardwood or softwood means. Members follow the unit's ranked order (species in `neverSplit` last); without it, the dominant member is used. A merged fir unit dominated by unfitted balsam fir thus takes subalpine fir's fitted curve rather than the conifer mean. The traits' source is recorded in `growthTraitSource`, e.g. "estimated (Abie_las)". Apply the result to speciesEcoregion with `LandR::modifySpeciesAndSpeciesEcoregionTable()`.

Usage

```
landweb_unit_growth_traits(species, traits, units)
```

Arguments

- species a study area's species table (one row per unit), from data preparation.
- traits from `landweb_growth_traits()`.
- units the study area's units, from `landweb_species_units()`.

Value

A copy of species with the traits set.

LandWebCRS	<i>Target CRS (projection) to use with LandWeb</i>
------------	--

Description

As of September 2025, this corresponds to the SCANFI's `NAD_1983_Canada_Lambert` projection.

Usage

```
LandWebCRS
```

Note

needs to be character, not CRS class, for downstream use with `data.table`

LandWebStudyAreas	<i>Study areas available for LandWeb simulations</i>
-------------------	--

Description

Study areas available for LandWeb simulations

Usage

LandWebStudyAreas

`studyAreaProv(name)`

`studyAreaIn(name, prov)`

Arguments

name	character specifying the study area name. Must be one of <code>LandWebStudyAreas\$Name</code> .
prov	character specifying the province abbreviation (i.e., "AB", "BC", "MB", "NWT", or "SK").

Value

LandWebStudyAreas is a `data.frame` with 4 columns:

- Name specifies the shortname of the study area for use with simulation setup;
- Province identifies the location of the study area;
- ID corresponds to the polygon ID of the Updated FMA Boundaries polygons;
- Description provides the full study area name and additional context (if any);

`studyAreaProv` returns a character indicating the province the study area is in.

`studyAreaIn` returns a logical indicating whether study area name is in prov.

LargePatches

Calculate proportion of large patches in NRV

Description

TODO: needs description

Usage

```
LargePatches(
  tsf,
  vtm,
  poly,
  labelColumn,
  id,
  ageClassCutOffs,
  ageClasses,
  sppEquivCol,
  sppEquiv,
  crop2poly = FALSE
)
```

Arguments

tsf	A single filename, relative or absolute, pointing to a Time Since Fire raster. Can be any format that terra can use.
vtm	A single filename, relative or absolute, pointing to a Vegetation Type Map raster. Can be any format that terra can use.
poly	A single sf object or a factor SpatRaster. This layer MUST have a column labelled shinyLabel
labelColumn	TODO: description needed
id	TODO: description needed
ageClassCutOffs	A numeric vector with the endpoints for the ageClasses. Should be length(ageClasses) + 1. See .ageClassCutOffs.
ageClasses	A character vector with labels for age classes to bin the tsf times, e.g., c("Young", "Immature", "Mature", "Old"). See .ageClasses.
sppEquivCol	Character giving the column name to use in sppEquiv.
sppEquiv	Species equivalency table, e.g., derived from LandR::sppEquivalencies_CA.
crop2poly	logical indicating whether to crop/mask vtm and tsf rasters to poly. Default FALSE for backwards compatibility.

lcc2020_classes	<i>Canada LCC 2020 class groups used by LandWeb</i>
-----------------	---

Description

The single definition of which Land Cover of Canada 2020 (NALCMS level II, 30 m) codes LandWeb treats as forest, non-flammable, urban, and so on. The same codes are needed by the land-cover remap ([lcc2020_remap_table\(\)](#)), the flammability map, and the current-condition age check ([cc_age_pct_missing\(\)](#)); defining them once keeps those from drifting apart.

Usage

```
lcc2020_classes()
```

Value

A named list of integer vectors:

forest 1, 2, 5, 6 – temperate/sub-polar needleleaf, sub-polar taiga needleleaf, temperate/sub-polar broadleaf deciduous, mixed forest (LandWeb v2 CC classes 0-2).

shrubGrassMoss 8, 10, 11, 12, 13 – shrubland, grassland and the sub-polar lichen/moss classes: no vegetation dynamics, but they burn (v2 CC 3).

wetland 14 – burns (v2 CC 3).

cropland 15 – treated as grassland for fire (v2 CC 5).

nonFlammable 16, 18, 19 – barren, water, snow/ice (v2 CC 4).

urban 17 – no v2 equivalent; reclassified to the nearest type for simulation.

The groups are disjoint.

Examples

```
lcc2020_classes()$forest
```

lcc2020_remap_table	<i>Remap table for overlaying LCC 2020 onto the simulation land cover</i>
---------------------	---

Description

Builds the remapTable that `LandR::overlayLCCs()` uses to decide, for every combination of a simulation land-cover code (LCC, SCANFI-derived) and a current-condition LCC 2020 code (CC), which code the pixel ends up with. Promoted verbatim from `LandWeb_preamble`, which mirrors LandWeb v2's rule ordering with LCC 2020 codes substituted for v2's CC 0-5.

Usage

```
lcc2020_remap_table(
  lccClasses,
  ccClasses,
  treeClassesToReplace,
  classes = lcc2020_classes()
)
```

Arguments

<code>lccClasses</code>	integer vector of the simulation land-cover codes present (NA is added).
<code>ccClasses</code>	integer vector of the LCC 2020 codes present; NAs are dropped, then NA is added back as its own row.
<code>treeClassesToReplace</code>	integer vector of LCC codes that must be reclassified (set to 99).
<code>classes</code>	class groups, as returned by <code>lcc2020_classes()</code> .

Details

Rule order matters: later rules overwrite earlier ones. In order:

1. LCC 0, 20, 30 -> NA;
2. CC missing or cropland -> keep LCC;
3. CC non-flammable (barren/water/snow-ice) -> NA;
4. CC forest, shrub/grass/moss or wetland -> keep LCC;
5. LCC missing but CC forest -> 99;
6. CC urban -> 99;
7. LCC in `treeClassesToReplace` -> 99.

CC does **not** override the simulation's forest determination: every CC class except barren/water/snow-ice defers to LCC, so LCC 2020 calling a treed wetland "wetland" does not strip forest from the simulation.

Note that rule 1 is overwritten by rules 2-6 for **every** code in the LCC 2020 legend, so it only takes effect for a CC value outside that legend. It is kept for fidelity with v2 rather than removed, and the behaviour is pinned by a test.

Value

A data.table with integer columns LCC, CC and newLCC, one row per combination.

Examples

```
lcc2020_remap_table(
  lccClasses = c(20L, 30L, 210L, 220L),
  ccClasses = c(1L, 14L, 17L, 18L),
  treeClassesToReplace = 240L
)
```

LeadingVegTypeByAgeClass

Calculate proportion of landscape occupied by each vegetation class

Description

This function is recursive. If poly is a SpatialPolygon, then the function will enter once, and convert this to a rasterized version, and pass that into the function replacing poly. It is also recursive if passed a vector of filenames for tsf and vtm.

Usage

```
LeadingVegTypeByAgeClass(
  tsf,
  vtm,
  poly,
  ageClassCutOffs,
  ageClasses,
  sppEquivCol,
  sppEquiv
)
```

Arguments

tsf	A single filename, relative or absolute, pointing to a Time Since Fire raster. Can be any format that terra can use.
vtm	A single filename, relative or absolute, pointing to a Vegetation Type Map raster. Can be any format that terra can use.
poly	A single sf object or a factor SpatRaster.
ageClassCutOffs	A numeric vector with the endpoints for the ageClasses. Should be length(ageClasses) + 1. See .ageClassCutOffs.
ageClasses	A character vector with labels for age classes to bin the tsf times, e.g., c("Young", "Immature", "Mature", "Old"). See .ageClasses.
sppEquivCol	Character giving the column name to use in sppEquiv.
sppEquiv	Species equivalency table, e.g., derived from LandR::sppEquivalencies_CA.

Value

A data.table with proportion of the pixels in each vegetation class, for each given age class within each polygon.

loadCASFRI	<i>Load CASFRI data for LandWeb</i>
------------	-------------------------------------

Description

Using CASFRI v4 data prepared for LandWeb.

Usage

```
loadCASFRI(
  CASFRIRas,
  attrFile,
  headerFile,
  sppEquiv,
  sppEquivCol,
  type = c("cover", "age")
)
```

Arguments

CASFRIRas	TODO: description needed
attrFile	TODO: description needed
headerFile	TODO: description needed
sppEquiv	table with species name equivalencies between the kNN and final naming formats. See <code>data("sppEquivalencies_CA", "LandR")</code> . For functions that have <code>mixedType</code> e.g., <code>vegTypeMapGenerator</code> , this only necessary if <code>mixedType == 2</code> . If not provided and <code>mixedType == 2</code> , will attempt to use <code>data("sppEquivalencies_CA", "LandR")</code> .
sppEquivCol	the column name to use from <code>sppEquiv</code> . For functions that have <code>mixedType</code> e.g., <code>vegTypeMapGenerator</code> , Only necessary if <code>mixedType == 2</code> . If not provided and <code>mixedType == 2</code> , will attempt to use "Boreal".
type	Character string. Either "cover" or "age".

Value

TODO: description needed

meanTruncPareto	<i>Calculate the mean of a truncated Pareto distribution</i>
-----------------	--

Description

Calculate the mean of a truncated Pareto distribution

Usage

```
meanTruncPareto(k, lower, upper, alpha)
```

Arguments

k	TODO: description needed
lower	TODO: description needed
upper	TODO: description needed
alpha	TODO: description needed

Value

TODO: description needed

plotFMA	<i>Plot boundary polygon(s) for forest management areas</i>
---------	---

Description

Plot boundary polygon(s) for forest management areas

Usage

```
plotFMA(x, provs, caribou = NULL, xsr = NULL, title = NULL, png = NULL)

plotLandWeb(x, provs, caribou = NULL, xsr = NULL, title = NULL, png = NULL)

plotGG(x, provs, caribou = NULL, png = NULL, title = NULL)
```

Arguments

x	sf object corresponding to the FMA to be plotted
provs	sf object corresponding to the provincial (or territorial) boundaries to plot
caribou	Optional sf object corresponding to caribou boundaries
xsr	Optional sf object corresponding to a buffered studyArea
title	Character string to use for plot title
png	Optional. If non-NULL, must be a valid file path to write a png

polygonClean	<i>Do an arbitrary set of operations on a polygon</i>
--------------	---

Description

Do an arbitrary set of operations on a polygon

Usage

```
polygonClean(poly, fn = NULL, type = NULL, ...)
```

Arguments

poly	A polygon object, or a character string identifying the shapefile path to load, and clean.
fn	A function identifying the type of cleaning to do.
type	If fn is not known, an character string can be specified to identify which fn to use. This must be a known type for this function.
...	Passed to fn

prepFMAs	<i>Prepare reporting polygons</i>
----------	-----------------------------------

Description

- LandWeb FMA boundaries (FMAs);
- Alberta FMU boundaries (FMUs);
- Alberta Natural Subregions (ANSRs);

Usage

```
prepFMAs(destinationPath, targetCRS = LandWebCRS)
prepFMUs(destinationPath, targetCRS = LandWebCRS)
prepANSRs(destinationPath, targetCRS = LandWebCRS)
prepEcoregionLayer(destinationPath, targetCRS = LandWebCRS)
prepEcoprovinceLayer(destinationPath, targetCRS = LandWebCRS)
```

Arguments

destinationPath	character specifying path to destination directory
targetCRS	character specifying the CRS to reproject polygons to

prepSpeciesLayers_CASFRI	<i>Prepare species layers from CASFRI v4</i>
--------------------------	--

Description

Prepare species layers from CASFRI v4

Usage

```
prepSpeciesLayers_CASFRI(  
  destinationPath,  
  outputPath,  
  url = NULL,  
  studyArea,  
  rasterToMatch,  
  sppEquiv,  
  sppEquivCol,  
  ...  
)
```

Arguments

destinationPath	
outputPath	character, directory for downloaded/extracted inputs.
url	optional character, source URL for the CASFRI data.
studyArea	study-area polygon used to crop/mask the inputs.
rasterToMatch	template raster defining CRS, extent, and resolution.
sppEquiv	species-equivalency table (e.g. LandR : sppEquivalencies_CA).
sppEquivCol	character, column of sppEquiv naming the species set.
...	additional arguments passed to lower-level functions.

prepStudyArea	<i>Extract boundary polygon(s) for a LandWeb study area</i>
---------------	---

Description

Resolves a study-area name to an sf boundary. The primary case is an **ecoregion study-area group** (see [build_studyarea_crosswalk\(\)](#)): its member FMA/TSA/FML polygons are looked up in crosswalk and unioned into one boundary. Alberta FMU names and the "random" test area are handled specially, and legacy single-FMA registry names ([LandWebStudyAreas](#)) still resolve for back-compatibility.

Usage

```

prepStudyArea(name, destinationPath, targetCRS = LandWebCRS, crosswalk = NULL)

prepTestStudyArea(destinationPath, targetCRS = LandWebCRS, .seed = 867)

```

Arguments

name	A study-area group token (e.g. "WesternAlbertaUpland"), an Alberta FMU name, "random", or a legacy LandWebStudyAreas name.
destinationPath	character specifying path to destination directory
targetCRS	character specifying the CRS to reproject polygons to
crosswalk	the study-area grouping crosswalk (see studyAreaCrosswalk()). If NULL (default) it is built/loaded via studyAreaCrosswalk() ; pass a precomputed one (e.g. a cached pipeline target) to avoid rebuilding it.
.seed	integer specifying the random seed to use to generate study area boundary

Value

sf polygons object.

rasterListByPoly	<i>Create a list of rasters in each rep, at each timestep, for each polygon area</i>
------------------	--

Description

Create a list of rasters in each rep, at each timestep, for each polygon area

Usage

```
rasterListByPoly(files, polys, names, col, filter)
```

Arguments

files	character vector giving paths to raster files
polys	polygon object of class sf
names	character vector giving the names of each of the subpolygons in poly
col	character string giving the column name in poly to use
filter	regex string giving partial filename in files to be filtered out (i.e., when extracting reps, times, etc. from filenames) (e.g., 'rstTimeSinceFire_', 'vegTypeMap_')

Value

list of SpatRaster objects with attributes reps, times, polyNames

Note

uses `future_lapply` internally to loop across files; set e.g., `option future.availableCores.fallback` appropriately for your system.

<code>recode_cohorts</code>	<i>Recode cohorts' species to their groups</i>
-----------------------------	--

Description

Recode cohorts' species to their groups

Usage

```
recode_cohorts(cohortData, map)
```

Arguments

- | | |
|-------------------------|---|
| <code>cohortData</code> | <code>data.table</code> of cohorts with columns <code>pixelGroup</code> , <code>speciesCode</code> and <code>B</code> . It is not modified. |
| <code>map</code> | named character vector from species code to group, as <code>sppEquiv_groups()</code> \$map. |

Value

A new `data.table` with columns `pixelGroup`, `speciesCode` (now the group) and `B`, ready for `LandR::vegTypeMapGenerator()` with `sppEquiv_groups()`\$groups. It is an error for a cohort's species to have no group.

See Also

[sppEquiv_groups\(\)](#)

<code>refCodeFor</code>	<i>Output key (refCode) for one analysis kind and reporting layer</i>
-------------------------	---

Description

The filesystem key an analysis writes under: it names the parquet aggregate directory (`_aggregates/<refCode>/`) and the figure/CSV filenames. Built by slugging the layer's name — which is its curated `NAME_SHORT`, or "`<tenure> <NAME_SHORT>`" for a crossed unit — so the on-disk key is exactly the label shown in the figures, and two units can only collide if their curated short names do (which [validateShortNames\(\)](#) forbids).

Usage

```
refCodeFor(kind, layer)
```

Arguments

kind	character analysis kind ("lm", "pm", "lw", "sspm", ...).
layer	character reporting-layer name (a reportingPolygons list name).

Details

This replaces the former `abbreviate(layer, minlength = 8)` keying, which gave no cross-unit uniqueness guarantee: 68 of the 399 crossed FMA x ANSR names abbreviated onto a name already taken, silently overwriting each other's aggregates and figures.

Value

a character refCode.

reportingPolygonLayers

Candidate reporting-polygon source layers

Description

The reporting-polygon layers considered for LandWeb/NRV summaries. Each is fetched, clipped to the study area with `spatialutils::prep_vector()`, and kept only if it intersects (see `buildReportingPolygons()`). source is one of "drive" (a Google Drive file id), "url" (a direct URL to a zipped shapefile), or "geojson" (a URL returning GeoJSON directly, e.g. an ArcGIS REST query endpoint – fetched with no archive-extraction step); labelCols names the attribute(s) holding the sub-region label.

Usage

```
reportingPolygonLayers()
```

Details

labelCols is a **comma-separated priority list**, coalesced left to right, because the v10 layers were assembled by merging per-jurisdiction sources and several name their features in different columns depending on the province. The Caribou ranges are the sharp case: RANGE_NAME is populated for only 21 of the 74 ranges (Ontario and Manitoba) — British Columbia uses HERD_NAME, Alberta LOCALRANGE, Saskatchewan CONUNIT/RNGEUNIT — so labelling on RANGE_NAME alone left 53 ranges unnamed. Unnamed features are dropped downstream, which silently removed every tenure x Caribou reporting unit outside ON/MB.

NAME_SHORT is the curated short token that names the layer everywhere it is keyed: it is the name of the layer's entry in the `buildReportingPolygons()` list, the second half of a crossed unit's name ("`<tenure NAME_SHORT> <layer NAME_SHORT>`", see `buildCrossedReportingPolygons()`), and — via `.slug()` — the refCode that keys the parquet aggregates and figure files. key is retained as the human title for reports. See `validateShortNames()` for the invariants enforced.

isTenure marks the **tenure** layer — the forest-management boundaries every other layer is crossed against (`buildCrossedReportingPolygons()`). Exactly one layer may carry it.

cross marks the layers crossed with each tenure by `buildCrossedReportingPolygons()`, reproducing the v2 per-tenure reporting units. Every non-tenure layer is crossed: reporting is `studyAreaReporting` intersected by tenure, so a stand-alone layer covering ground *outside* the tenures has nothing to report against.

A reporting layer may be assembled from **several sources**: rows sharing a `NAME_SHORT` are built independently and then merged into one layer (reduced to the common `Name` column). `where` optionally keeps only the features whose label matches a regular expression, so a supplementary source can contribute just the part the primary source is missing. Removing a supplement later is a one-row deletion.

Value

A tibble with columns `key`, `NAME_SHORT`, `isTenure`, `cross`, `source`, `id`, `labelCols`, and `where`.

<code>runBoxPlotsVegCover</code>	<i>Generate box and whisker plots for leading vegetation cover</i>
----------------------------------	--

Description

Generate box and whisker plots for leading vegetation cover

Usage

```
runBoxPlotsVegCover(map, functionName, analysisGroups, dPath)
```

Arguments

- | | |
|-----------------------------|---|
| <code>map</code> | A map object. |
| <code>functionName</code> | TODO: description needed |
| <code>analysisGroups</code> | TODO: description needed |
| <code>dPath</code> | Destination path for the resulting PNG files. |

<code>runHistsLargePatches</code>	<i>Generate histograms for large patches</i>
-----------------------------------	--

Description

TODO: description needed

Usage

```
runHistsLargePatches(map, functionName, analysisGroups, dPath)
```

Arguments

map	A map object.
functionName	TODO: description needed
analysisGroups	TODO: description needed
dPath	Destination path for the resulting PNG files.

runHistsVegCover	<i>Generate histograms for leading vegetation cover</i>
------------------	---

Description

Generate histograms for leading vegetation cover

Usage

```
runHistsVegCover(map, functionName, analysisGroups, dPath)
```

Arguments

map	A map object.
functionName	Character specifying the name of the function to apply.
analysisGroups	Character specifying the analysis groups to run.
dPath	Destination path for the resulting PNG files.

shortNameFor	<i>Look up curated short names</i>
--------------	------------------------------------

Description

Vectorised lookup of NAME_SHORT for a set of names, against a two-column curated table (e.g. [tenureShortNames\(\)](#)).

Usage

```
shortNameFor(x, table = tenureShortNames())
```

Arguments

x	character vector of long names to look up.
table	a two-column curated table; the first column holds the long names and NAME_SHORT the short ones (default tenureShortNames()).

Value

character vector the same length as x; NA where x has no curated code.

sppEquiv_groups	<i>Group the simulated species of a species table</i>
-----------------	---

Description

Pairs each simulated species code with its group, and builds the one-row-per-group table that `LandR::vegTypeMapGenerator()` takes as `sppEquiv` for cohorts recoded to their groups (`recode_cohorts()`).

Usage

```
sppEquiv_groups(sppEquiv, sppEquivCol, groupCol)
```

Arguments

- `sppEquiv` `data.table` species equivalency table with columns `sppEquivCol`, `groupCol` and `Type`. It is not modified.
- `sppEquivCol` character; the column of simulated species codes.
- `groupCol` character; the column of groups.

Details

It is an error for a simulated code to have no group, or more than one, and for a group to hold both conifers and broadleaves (`Type`). `LandR::vegTypeMapGenerator()` reads `Type`, one row per code, to find mixedwood stands (`mixedType = 2`), so a group of both would be counted twice there.

Value

A list: `map`, a named character vector from simulated code to group; `groups`, a `data.table` with one row per group (sorted) and columns `groupCol` and `Type`; and `col`, `groupCol`.

See Also

`recode_cohorts()`, `landweb_add_report_column()`

studyAreaCrosswalk	<i>Build (or load a cached) LandWeb study-area grouping crosswalk</i>
--------------------	---

Description

Convenience wrapper that assembles `build_studyarea_crosswalk()`'s inputs from the packaged prep helpers — `prepFMAs()` for the FMA boundaries and, by default, `prepEcoregionLayer()` for the grouping layer — and caches the result under `destinationPath` so repeated study-area setups within a run do not rebuild it (the overlay is the expensive part).

Usage

```
studyAreaCrosswalk(  
  destinationPath,  
  targetCRS = LandWebCRS,  
  eco = prepEcoregionLayer,  
  eco_field = "REGION_NAM",  
  cache = TRUE  
)
```

Arguments

destinationPath	character specifying path to destination directory
targetCRS	character specifying the CRS to reproject polygons to
eco	a function of (destinationPath, targetCRS) returning the ecological grouping layer as sf (default prepEcoregionLayer); pass prepEcoprovinceLayer for coarser groups or a custom prep for a different eco* layer.
eco_field	character naming the grouping-unit column in that layer (default "REGION_NAM", the ecoregion name).
cache	logical; reuse/save file.path(destinationPath, "studyAreaCrosswalk.rds").

Value

the crosswalk data.frame (see [build_studyarea_crosswalk\(\)](#)).

tenureShortNames	<i>Curated short names for the LandWeb tenures</i>
------------------	--

Description

One row per member tenure of the v10 combined boundary layer (FMAs_LandWeb1tfc_map_v10_LCC1), mapping its **coalesced identity** — the same FMA_NAME/FMU_NAME/TSA_NUMB_1/FOREST_NAM/FML_NAME/Name coalesce that [build_studyarea_crosswalk\(\)](#) uses as fma_name — to a curated NAME_SHORT.

Usage

```
tenureShortNames()
```

Details

NAME_SHORT is the partner-facing label: it names the tenure in every crossed reporting unit ("**<NAME_SHORT> <sub-region>**"), in the figure titles and facet strips, and (via `.slug()`) in the refCode that keys the parquet aggregates and figure files. Codes are derived from the **v10 identity**, not from the legacy v2 study-area registry ([LandWebStudyAreas](#)), because tenures change hands and the v10 layer is the authority. Codes are hand-curated rather than generated so that adding a reporting layer, or a new member to the boundary layer, can never silently rename an existing unit's outputs. Adding a member to the v10 layer therefore requires adding its code here — an uncurated member is an error, not a silent fallback (see [build_studyarea_crosswalk\(\)](#)).

Value

A tibble with columns fma_name (the coalesced v10 identity) and NAME_SHORT.

See Also

[validateShortNames\(\)](#), [build_studyarea_crosswalk\(\)](#)

validateShortNames	<i>Validate a set of curated short names</i>
--------------------	--

Description

Enforces the invariants the output keying depends on: every short name is present, filesystem-safe (unchanged by `.slug()`, so refCode round-trips), no longer than `max_len`, and unique — globally, or within each level of `by` (used for the per-tenure landbase codes, which need only be unique inside their tenure).

Usage

```
validateShortNames(  
  x,  
  what = "NAME_SHORT",  
  max_len = .maxShortNameLength,  
  by = NULL  
)
```

Arguments

x	character vector of candidate short names.
what	character used in error messages to name the table being validated.
max_len	integer maximum permitted length (default 16).
by	optional grouping vector the same length as x; uniqueness is then required only within each group.

Value

x, invisibly, if every check passes; otherwise an error.

Index

.cleanLandWebStudyArea, 4

analysesOutputsTimes, 5

build_studyarea_crosswalk, 5

build_studyarea_crosswalk(), 82, 88–90

buildCaribouRanges, 7

buildCaribouRanges(), 11, 13

buildCrossedReportingPolygons, 8

buildCrossedReportingPolygons(), 85, 86

buildLandbasePolygons, 9

buildLandbasePolygons(), 8, 9, 19, 20

buildReportingPolygons, 10

buildReportingPolygons(), 7–10, 12, 85

caribouRangeLayers, 11

caribouRangeLayers(), 7

CASFRItoSpRasts, 13

cc_age_composite, 14

cc_age_composite(), 15

cc_age_pct_missing, 15

cc_age_pct_missing(), 15, 76

cleanAreaName, 16

extractFMA, 17

extractFMU, 17

findOldSimFiles (findSimFile), 18

findSimFile, 18

joinReportingPolygons, 18

joinReportingPolygons(), 8, 9

landbaseLayers, 19

landbaseLayers(), 9, 10

landmine_area_burned_by_zone, 20

landmine_area_share, 21

landmine_attach_identity, 22

landmine_burn (landmine_burn1), 23

landmine_burn1, 23

landmine_burn1(), 39, 46, 47

landmine_calibration_plots, 25

landmine_draw_num_fires, 26

landmine_estimate_kBest, 27

landmine_fire_attainment, 28

landmine_fire_ros, 29

landmine_fire_ros(), 36, 38, 57

landmine_fri_metrics, 31

landmine_fri_metrics(), 35, 53, 54

landmine_fri_summary, 34

landmine_fri_summary(), 31, 33, 38

landmine_fri_verdict, 35

landmine_fri_verdict(), 33, 53

landmine_fuel_types, 36

landmine_fuel_types(), 30

landmine_ignition_budget, 36

landmine_ignition_budget(), 55

landmine_known_species, 38

landmine_known_species(), 30, 36, 57

landmine_optim_burnFun, 39

landmine_optim_burnFun(), 46, 47

landmine_optim_calibrate, 40

landmine_optim_clusterExport, 41

landmine_optim_clusterSetup, 42

landmine_optim_clusterWrap, 43

landmine_optim_fireSizes, 43

landmine_optim_fireSizes(), 40

landmine_optim_fitAndison, 44

landmine_optim_fitAndison(), 25, 41

landmine_optim_fitSN, 46

landmine_optim_fitSN(), 41, 44–46, 52

landmine_optim_fitSN2
(landmine_optim_fitSN), 46

landmine_optim_islands, 47

landmine_optim_islands(), 45

landmine_optim_islandTarget, 48

landmine_optim_islandTarget(), 45

landmine_optim_islandWeight, 48

landmine_optim_islandWeight(), 45

landmine_optim_landscape, 49

- landmine_optim_params_append
(landmine_optim_unpack), 51
- landmine_optim_params_read
(landmine_optim_unpack), 51
- landmine_optim_params_read(), 25
- landmine_optim_shapeIndex, 50
- landmine_optim_shapeTarget, 51
- landmine_optim_shapeTarget(), 45
- landmine_optim_unpack, 51
- landmine_optim_unpack(), 45
- landmine_plot_areaBurnedOverTime
(landmine_plots), 54
- landmine_plot_calibConvergence
(landmine_calibration_plots),
25
- landmine_plot_calibDiscrimination
(landmine_calibration_plots),
25
- landmine_plot_calibIslands
(landmine_calibration_plots),
25
- landmine_plot_calibParams
(landmine_calibration_plots),
25
- landmine_plot_calibShape
(landmine_calibration_plots),
25
- landmine_plot_FRI (landmine_plots), 54
- landmine_plot_fri_drivers, 52
- landmine_plot_fri_zones, 53
- landmine_plot_fri_zones(), 32, 33
- landmine_plot_LTHFC (landmine_plots), 54
- landmine_plots, 54
- landmine_reburn_budget, 55
- landmine_reburn_ceiling, 56
- landmine_ros_table, 57
- landmine_ros_table(), 36
- landmine_sizes_to_pixels, 57
- landweb_add_report_column, 58
- landweb_add_report_column(), 65, 66, 88
- landweb_anpp_area, 59
- landweb_area, 59
- landweb_area(), 63
- landweb_bec_zones, 60
- landweb_damage_codes, 60
- landweb_dominant_sppEquiv, 61
- landweb_dominant_sppEquiv(), 64, 70
- landweb_growth_sppEquiv, 62
- landweb_growth_trait_frequency, 62
- landweb_growth_traits, 63
- landweb_growth_traits(), 62, 73
- landweb_lthfc, 63
- landweb_lthfc(), 59
- landweb_member_cover, 64
- landweb_member_cover(), 61
- landweb_report_colours, 64
- landweb_report_colours(), 65
- landweb_report_groups, 65
- landweb_report_groups(), 36, 65, 66
- landweb_report_map, 66
- landweb_report_map(), 59, 65, 69
- landweb_require_species, 66
- landweb_require_species(), 72
- landweb_resample_psp, 67
- landweb_resample_psp(), 62
- landweb_species_map, 68
- landweb_species_map(), 65, 66, 69, 72
- landweb_species_sppEquiv, 68
- landweb_species_sppEquiv(), 62, 70
- landweb_species_units, 69
- landweb_species_units(), 68, 69, 72, 73
- landweb_sppEquiv, 71
- landweb_sppEquiv(), 58, 61, 68, 69
- landweb_sum_layers, 72
- landweb_sum_layers(), 71
- landweb_unit_growth_traits, 73
- landweb_unit_growth_traits(), 71
- LandWebCRS, 7, 10, 11, 73
- LandWebStudyAreas, 74, 82, 83, 89
- LargePatches, 75
- lcc2020_classes, 76
- lcc2020_classes(), 15, 77
- lcc2020_remap_table, 76
- lcc2020_remap_table(), 76
- LeadingVegTypeByAgeClass, 78
- loadCASFRI, 79
- meanTruncPareto, 80
- parallel::clusterSetRNGStream(), 42
- plotFMA, 80
- plotGG (plotFMA), 80
- plotLandWeb (plotFMA), 80
- polygonClean, 81
- polygonClean(), 59, 63
- prepANSRs (prepFMAs), 81
- prepEcoprovinceLayer, 89

prepEcoprovinceLayer (prepFMAs), 81
prepEcoregionLayer, 89
prepEcoregionLayer (prepFMAs), 81
prepEcoregionLayer(), 88
prepFMAs, 81
prepFMAs(), 6, 88
prepFMUs (prepFMAs), 81
prepSpeciesLayers_CASFRI, 82
prepStudyArea, 82
prepTestStudyArea (prepStudyArea), 82

rasterListByPoly, 83
rasterVis::levelplot(), 54
recode_cohorts, 84
recode_cohorts(), 88
refCodeFor, 84
reportingPolygonLayers, 85
reportingPolygonLayers(), 8, 10–13
runBoxPlotsVegCover, 86
runHistsLargePatches, 86
runHistsVegCover, 87

sf::st_intersection(), 19
sf::st_make_valid(), 19
shortNameFor, 87
spatialutils::prep_landbase(), 9
spatialutils::prep_vector(), 7, 10, 85
spatialutils::repair_geoms(), 19
sppEquiv_groups, 88
sppEquiv_groups(), 84
studyAreaCrosswalk, 88
studyAreaCrosswalk(), 8, 10, 83
studyAreaIn (LandWebStudyAreas), 74
studyAreaProv (LandWebStudyAreas), 74

tenureShortNames, 89
tenureShortNames(), 6, 11, 20, 87
terra::intersect(), 19
terra::vect(), 7, 10, 11, 19

validateShortNames, 90
validateShortNames(), 84, 85, 90