Package: LakeGeoid (via r-universe)

November 4, 2026

Title Lake geoid correction grids

Version 0.0.0.9002

Maintainer Karina Nielsen <karni@space.dtu.dk>

Description What the package does (one paragraph).

LazyData true

Imports RTMB, fmesher, sf, terra, viridis, fields, httr2, ncdf4, Matrix

License `use_mit_license()`, `use_gpl3_license()` or friends to pick a license

Encoding UTF-8

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.1.1

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

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

Date/Publication 2026-11-04 16:22:38 UTC

RemoteUrl https://github.com/cavios/LakeGeoid

RemoteRef HEAD

RemoteSha 14bf0674a316dd38bc3d6a929b282380fdcd8855

RemoteSubdir LakeGeoid

Contents

addCrossTrack 2
C2kivu.RData 3
clipToLake 3
downsample 4
flagOutliers 5
getGeoidCorr 7
getGeoidGrid 7

getLakeGeoid 8
getLakeLevel 10
getPLDLake 11
getSWOTRaster 12
kivu.RData 13
myshape.RData 14
openLake 14
overpassSummary 15
plot.LakeGeoid 15
plotOverpass 16
print.LakeGeoid 17
summary.LakeGeoid 18

Index **19**

`addCrossTrack` *Add the cross-track distance to SWOT pixels*

Description

The function addCrossTrack adds the column "cross_track" (signed cross-track distance in m, zero at nadir), needed by getLakeGeoid(overpassRoll=TRUE), to pixels downloaded before getSWOTRaster() kept it. The ground track of a pass repeats to within about a kilometre, so for each pass one granule (the overpass with most pixels) is downloaded again, the cross-track distance is fitted as a quadratic in UTM coordinates over the lake, and the fit is applied to the pixels of every overpass of that pass. Pixels that already have a cross-track distance keep it.

Usage

```
addCrossTrack(
  dat,
  lake,
  netrc = "~/.netrc",
  tmpDir = tempdir(),
  product = "SWOT_L2_HR_Raster_250m_D"
)
```

Arguments

`dat`	A data.frame with SWOT pixels (columns "lon", "lat" and "granule"), e.g. from getSWOTRaster()
`lake`	The lake polygon (sf), as used for the download
`netrc`	Path to the .netrc file with the Earthdata login
`tmpDir`	Folder for the temporary granules (deleted after use)
`product`	Short name of the SWOT raster product

Value

dat with the column "cross_track"; the fits per pass are in the attribute "xtrack" (with the RMSE of each fit, typically a few metres)

Examples

```
# dat <- addCrossTrack(dat, lake, netrc = "~/.netrc")
# fit <- getLakeGeoid(dat[dat$flag == "ok", ], maxEdge, lake, overpassRoll = TRUE)
```

C2kivu.RData *CryoSat-2 water level observations for lake Kivu*

Description

The CryoSat-2 L1b waveforms were retracked by the Narrow Primary Peak retracker (Villadsen, H., Andersen, O.B., Stenseng, L., Nielsen, K. and Knudsen, P., 2015. CryoSat-2 altimetry for river level monitoring—Evaluation in the Ganges–Brahmaputra River basin. Remote Sensing of Environment, 168, pp.80-89.). CryoSat-2 is available from

- L1b SAR Precise Orbit. Baseline E.https://doi.org/10.5270/CR2-fbae3cd
- L1b LRM Precise Orbit, Baseline E.https://doi.org/10.5270/CR2-41ad749
- L1b SARin Precise Orbit. Baseline E.https://doi.org/10.5270/CR2-6afef01

Author(s)

Karina Nielsen (karni@space.dtu.dk)

References

Paper in prep

---

clipToLake *Keep the SWOT pixels inside a lake polygon*

Description

The function clipToLake drops the pixels outside the polygon, e.g. after removing river channels with openLake() from data downloaded with the original polygon.

Usage

```
clipToLake(dat, lake)
```

Arguments

dat	A data.frame with SWOT pixels (columns "lon" and "lat"), e.g. from getSWOTRaster()
lake	A lake polygon of the class "sf", e.g. from openLake()

Value

"dat" without the pixels outside the polygon

---

downsample *Downsample water level observations*

Description

The function downsample reduces the number of observations to speed up getLakeGeoid(). The thinning is done within each epoch (unique value of time), so all epochs are kept. With "cellSize" the observations are thinned spatially on a regular grid in UTM coordinates, which keeps an even spatial coverage. A cell size well below the "maxEdge" of the mesh preserves the resolution of the estimated field. Observations are selected, not averaged, so the observation error model in getLakeGeoid() is unchanged. "fraction" and "maxPerEpoch" are applied after the spatial thinning.

Usage

```
downsample(dat, cellSize = NULL, fraction = NULL, maxPerEpoch = NULL, seed = 1)
```

Arguments

dat	Input data set; must at least contain the columns time, lon and lat
cellSize	Size (in meters) of the grid cells used for spatial thinning; one randomly chosen observation is kept per grid cell and epoch (NULL means no spatial thinning)
fraction	Fraction (between 0 and 1) of the observations kept within each epoch (NULL means all)
maxPerEpoch	Maximum number of observations kept within each epoch (NULL means no maximum)
seed	Seed for the random number generator, for reproducible results (NULL means no seed). The random number state of the session is restored afterwards

Value

The data set with a subset of the rows

Examples

```
# datSmall <- downsample(kivu, cellSize = 750, seed = 1)
```

`flagOutliers` *Flag outliers in SWOT raster water level data*

Description

The function flagOutliers flags outliers in SWOT raster data. The tests are applied in the order listed above, and each test only considers the observations not flagged by the previous tests. A test is skipped if its argument is NULL. The reference field is the median, on a grid, of the observations relative to their overpass median, i.e. a first estimate of the static (geoid) signal. It is removed before the tilt, level and MAD tests, so the geoid signal itself is not flagged. The level test catches overpasses that are biased as a whole (e.g. affected by dark water or rain) but internally consistent, which the pixel tests do not detect.

Usage

```
flagOutliers(
  dat,
  lake = NULL,
  maxDarkFrac = 0.1,
  maxIceClim = 1,
  maxIceDyn = 0,
  maxIceShare = 0.1,
  maxWseUncert = NULL,
  minNPIXC = NULL,
  maxDev = 1,
  shoreBuffer = NULL,
  minPerEpoch = 100,
  minCoverage = 0.2,
  maxTilt = 0.5,
  maxLevelJump = 0.3,
  levelNmad = 5,
  levelWindow = 3,
  nmad = 5,
  cellSize = 1000
)
```

Arguments

`dat`	Data set returned by getSWOTRaster(); must contain the columns time, lon, lat and wse, and the columns used by the selected tests (darkwater_frac, wse_uncert, N_PIXC). If the column granule is present an overpass is defined by cycle and pass, otherwise by time
`lake`	Lake polygon of the class "sf"; only needed for "shoreBuffer"
`maxDarkFrac`	Flag pixels with darkwater_frac above this value ("dark")
`maxIceClim`	Flag pixels with ice_clim_flag (climatological ice cover: 0 none, 1 uncertain, 2 full) above this value ("ice"); NULL means no test. The test is skipped when the column is missing (data downloaded with older versions)

maxIceDyn	Flag pixels with ice_dyn_flag (observed ice cover: 0 none, 1 partial, 2 full) above this value ("ice"); NULL means no test
maxIceShare	Flag whole overpasses in which more than this share of the pixels is flagged "ice" ("ice"); partly frozen overpasses also have unreliable open-water pixels. NULL means no test
maxWseUncert	Flag pixels with wse_uncert above this value ("uncert")
minNPIXC	Flag pixels with fewer than this number of pixel cloud points ("npixc")
maxDev	Flag pixels more than this value (meters) from the median of their overpass ("range"). This removes errors that repeat at the same place in every overpass (e.g. layover from steep terrain), which the reference field would otherwise absorb. It must be larger than the expected range of the geoid signal over the lake
shoreBuffer	Flag pixels closer than this distance (meters) to the shore ("shore")
minPerEpoch	Flag whole overpasses with fewer than this number of remaining observations ("sparse")
minCoverage	Flag whole overpasses whose coverage (number of 1 km cells with remaining observations) is less than this fraction of the coverage of the best covered overpass ("coverage"). The default 0.2 removes overpasses that only see a small part of the lake, whose pass bias and level trade off against the field there. NULL means no test
maxTilt	Flag whole overpasses where the plane fitted to the residuals (relative to a reference field) varies more than this value in meters across the observations ("tilt")
maxLevelJump	Flag whole overpasses whose water level (median of the observations minus the reference field) differs from the median level of the neighbouring overpasses in time by more than this value in meters and by more than "levelNmad" times the MAD of these differences over all overpasses ("level")
levelNmad	See "maxLevelJump"; makes the level test adapt to the overpass-to-overpass scatter of the lake (NULL: only "maxLevelJump" is used)
levelWindow	Number of overpasses on each side used as neighbours in the level test
nmad	Flag pixels where the residual (relative to the overpass median and the reference field) is more than "nmad" times the MAD of the residuals of the overpass ("mad")
cellSize	Grid cell size (meters) of the reference field

Value

The data set with the additional column "flag" ("ok" or the reason for flagging). A table with statistics per overpass is attached as attribute "overpass"

Examples

```
# dat <- flagOutliers(dat); fit <- getLakeGeoid(dat[dat$flag == "ok", ], maxEdge = 3000, myshape = lake)
```

getGeoidCorr *Predict the geoid correction error*

Description

Predict the geoid correction error

Usage

```
getGeoidCorr(fit, newdat, UTM = TRUE)
```

Arguments

fit	Object returned by getLakeGeoid()
newdat	a data set at least including the columns named "lon" and "lat" with the coordinates where the corretion should be predicted.
UTM	logic variable to indicate if the estimated field is in a UTM projection (FALSE means the field is in degrees)

getGeoidGrid *Static field of a fit on a regular grid*

Description

The function getGeoidGrid evaluates the estimated static field (the geoid correction) on a regular grid, using the same linear interpolation within the mesh triangles as the model. Cells outside the mesh are NA. The result can be saved with terra::writeRaster(), e.g. as GeoTIFF.

Usage

```
getGeoidGrid(fit, res = 250)
```

Arguments

fit	Object returned by getLakeGeoid()
res	Grid resolution in the units of the mesh (meters for UTM)

Value

A terra SpatRaster in the projection of the fit (UTM unless getLakeGeoid() was called with UTM=FALSE)

Examples

```
# r <- getGeoidGrid(fit); terra::writeRaster(r, "geoidcorr.tif")
```

getLakeGeoid *Reconstruct missing geoid signal and a water level time series*

Description

The function getLakeGeoid reconstruct a static spatial water level signal and a water level time series. The static spatial signal is modeled as a Gaussian Markov Random Field on a triangular mesh where the nodes specifies the neighbor structure. The water level time series is modeled with an Random Walk as the underlying process and the observation error is here assumed to be Gaussian. The Static spatial field is here intended to models potential missing geoid model signals. Optionally a constant bias per SWOT pass is estimated (passBias=TRUE); the estimates are returned in the element "passBias" of the output. Optionally a random tilt per overpass (overpassTilt=TRUE) and a random level offset per overpass (overpassError=TRUE) are estimated.

Usage

```
getLakeGeoid(
  dat,
  maxEdge,
  myshape,
  UTM = TRUE,
  cellSize = NULL,
  fraction = NULL,
  maxPerEpoch = NULL,
  seed = 1,
  passBias = FALSE,
  overpassTilt = FALSE,
  overpassError = TRUE,
  overpassRoll = FALSE,
  rollSd = NULL,
  rollPrior = NULL,
  fieldMeanZero = TRUE,
  tiltCenter = c("overpass", "all")
)
```

Arguments

dat	Input data set; must at least contain columns names: time (decimal years), lon (longitude in decimal degrees), lat (latitude in decimal degrees), height (meters)
maxEdge	Maximum side length of the triangles in the mesh (created by the function "fm_mesh_2d" from the package "fmesher").
myshape	a shapefile/polygon of the class "sf" defining the boundary of the lake where the model will be reconstructed.
UTM	logic variable to specify if the coordinates should be projected to UTM coordinates (the default is TRUE)

`cellSize` Optional downsampling: keep one observation per grid cell of this size (meters) and epoch, see downsample()

`fraction` Optional downsampling: fraction of the observations kept within each epoch, see downsample()

`maxPerEpoch` Optional downsampling: maximum number of observations kept within each epoch, see downsample()

`seed` Seed used for the random downsampling

`passBias` Logical; if TRUE a constant bias is estimated for each SWOT pass (the biases sum to zero, so the water level is the average over passes). The pass is taken from the column "pass" or derived from the column "granule"

`overpassTilt` Logical; if TRUE a tilt (an east and a north slope) is estimated for each overpass (cycle and pass from the column "granule", otherwise each epoch) as a random effect with mean zero, to absorb e.g. SWOT roll errors. The standard deviations of the slopes (per 10 km for UTM coordinates) are estimated; the slopes are returned in the element "tilt" of the output. A tilt common to all epochs remains in the static field

`overpassError` Logical; if TRUE a level offset is estimated for each overpass as a random effect with mean zero and estimated standard deviation ("SdOpErr"). It represents errors common to all observations of an overpass (e.g. residual atmospheric corrections), so these are separated from the water level, which then becomes smoother and gets a more realistic uncertainty. The offsets are returned in the element "overpassError" of the output. TRUE by default since version 0.0.0.9001 (FALSE before). Set it to FALSE where the lake level changes fast compared with the spacing of the SWOT epochs (e.g. alpine lakes with flashy inflows seen every one to two weeks): the random walk then cannot follow the real jumps and the overpass error absorbs them

`overpassRoll` Logical; if TRUE a roll is estimated for each overpass as a random effect with mean zero: a height error proportional to the signed cross-track distance of the pixel (column "cross_track", m, see addCrossTrack()), so it is zero along the nadir track. Unlike the tilt, a roll also shifts the mean level of a lake that lies to one side of nadir, which is the signature of the residual KaRIn roll error. It helps for lakes that are wide across the swath and seen far from nadir, but should not be used for shallow, wind-exposed lakes, whose real wind tilts it would turn into level errors, or for small lakes. Tested at 32 gauged lakes, fixing its standard deviation (rollSd=0.012) worked better than estimating it or giving it a prior. The rolls (m per 10 km) are returned in the element "roll" of the output

`rollSd` Fixed standard deviation of the roll (m per 10 km, e.g. 0.015); NULL (default) estimates it

`rollPrior` Optional prior on the standard deviation of the roll when it is estimated: c(median, sd of its logarithm), e.g. c(0.012, 0.3); with many overpasses the data outweigh it

`fieldMeanZero` Logical; if TRUE (default) the area-weighted mean of the static field is constrained to zero (soft constraint, sd 1 mm). The overall level then belongs to the water level, which removes a trade-off between the two that can make the fit slow or degenerate when the estimated range of the field is much longer than the

	lake, and makes the uncertainty of the water level meaningful. FALSE gives the behaviour of earlier versions
tiltCenter	Where the tilt of each overpass is centered: "overpass" (default) at the mean position of that overpass's observations, so the tilt does not change the overpass's mean level (important for overpasses that only cover part of the lake); "all" at the mean position of all observations

---

`getLakeLevel` *Water level time series without a static field*

Description

The function getLakeLevel fits only the temporal part of getLakeGeoid(): a random walk water level L and a random error per overpass, to the median (or mean) of each overpass. The sampling error of each overpass summary is computed from its pixels (1.25*1.4826*MAD/sqrt(n) for the median) and treated as known. This is much faster than getLakeGeoid(), and gives the same water level where the static field is small (below about 2 cm) or where every pass covers the whole lake. It does not correct partial passes over a lake with a large field. Passing the output of overpassSummary() avoids reading the pixels again. With robust=TRUE, \"sdOpErr\" is the scale of the Student-t distribution (its standard deviation is sdOpErr*sqrt(df/(df-2))), and the column \"weight\" of \"overpass\" gives the Student-t weight of each overpass (1 for a typical overpass, small for an outlier).

Usage

```
getLakeLevel(
  dat,
  stat = c("median", "mean"),
  minPix = 10,
  robust = "auto",
  df = 4,
  autoThreshold = 2,
  p = NULL
)
```

Arguments

dat	A data.frame with SWOT pixels (columns "wse", "time" and, to separate overpasses on the same date, "granule"), e.g. the pixels flagged "ok" by flagOutliers(), or an overpass table from overpassSummary()
stat	Summary of each overpass when "dat" holds pixels: "median" (default, robust) or "mean"
minPix	Minimum number of pixels for an overpass to be used
robust	If TRUE, the overpass error follows a Student-t distribution with "df" degrees of freedom instead of a normal distribution, so that single overpasses far off (outliers that the pixel tests did not catch) pull the water level less. With "auto",

	both are fitted and the robust fit is returned unless the normal fit has a negative log-likelihood lower by more than "autoThreshold"; the element "auto" of the output records the choice (\"auto\" is the default since version 0.0.0.9001; FALSE before)
df	Degrees of freedom of the Student-t overpass error when robust=TRUE (default 4)
autoThreshold	Log-likelihood margin used by robust="auto" (default 2)
p	Mixing probability of the Cauchy component when robust="mixture"; estimated if NULL (default)

Value

A list of the class "LakeLevel" with the water level "wl", its standard error "wlSD", the epochs "time", the overpass table "overpass" (level, sampling error, estimated overpass error) and the optimizer output "fit"

Examples

```
# lev <- getLakeLevel(dat[dat$flag=="ok",])
# ov <- overpassSummary(dat[dat$flag=="ok",]); lev <- getLakeLevel(ov)
# lev_robust <- getLakeLevel(ov, robust = TRUE)
# lev_auto <- getLakeLevel(ov, robust = "auto"); lev_auto$auto
```

---

getPLDLake *Extract a lake polygon from the SWOT Prior Lake Database (PLD)*

Description

The function getPLDLake returns the polygon of a single lake from the SWOT Prior Lake Database. When "pld" is a file path only the requested lake is read from the file. The lake id field may be named "lakeid" or "lake_id".

Usage

```
getPLDLake(lakeid, pld, opening = NULL)
```

Arguments

lakeid	The PLD lake id (numeric or character), e.g. 1320035532 for lake Kivu
pld	Either a path to a PLD shapefile (or any file readable by sf::st_read) or an object of the class "sf" containing the PLD
opening	NULL (default) or a width in m: remove river channels and narrow bays narrower than about twice this width with openLake()

Value

An object of the class "sf" with the lake polygon in longitude/latitude (EPSG:4326). Invalid polygons (e.g. self-intersecting ones) are repaired

Examples

```
# lake <- getPLDLake(1320035532, "SWOTPLDv2.2_simple/lakesub_simple.shp")
```

`getSWOTRaster` *Download SWOT Level 2 Water Mask Raster Image 250m data for a lake*

Description

The function getSWOTRaster searches NASA's Common Metadata Repository (CMR) for SWOT raster granules overlapping the lake polygon, downloads them from PO.DAAC using an Earthdata login, and extracts the pixels inside the lake polygon. All pixels from one granule get the same time (the date of the granule in decimal years), so each overpass is one epoch in getLakeGeoid(). Files already present in "outDir" are not downloaded again.

Usage

```
getSWOTRaster(
  lake,
  pld = NULL,
  startDate = NULL,
  endDate = NULL,
  shortName = "SWOT_L2_HR_Raster_250m_D",
  outDir = file.path(tempdir(), "SWOT"),
  netrc = "~/.netrc",
  maxDarkFrac = NULL,
  minWaterFrac = NULL,
  maxWseQual = NULL,
  keep = FALSE,
  tries = 3
)
```

Arguments

`lake`	Either a lake polygon of the class "sf" or a PLD lake id (then "pld" must be given)
`pld`	Path to a PLD shapefile or an "sf" object with the PLD; only used when "lake" is a lake id
`startDate`	Start date ("YYYY-MM-DD"); NULL means from the start of the mission
`endDate`	End date ("YYYY-MM-DD"); NULL means until today
`shortName`	CMR short name of the collection. The default "SWOT_L2_HR_Raster_250m_D" is version D; use "SWOT_L2_HR_Raster_250m_2.0" for version C
`outDir`	Directory where the NetCDF files are downloaded
`netrc`	Path to a .netrc file with the Earthdata login (machine urs.earthdata.nasa.gov login password). If the file does not exist the environment variables EARTHDATA_USER and EARTHDATA_PASS are used

maxDarkFrac	Remove pixels with dark_frac above this value (NULL means no filtering)
minWaterFrac	Remove pixels with water_frac below this value (NULL means no filtering)
maxWseQual	Remove pixels with wse_qual above this value (0=good, 1=suspect, 2=degraded, 3=bad; NULL means no filtering)
keep	Logical; keep the downloaded NetCDF files (the default FALSE deletes them after reading)
tries	Number of attempts per granule. A transfer that stalls for 2 minutes is aborted and retried. A granule that fails all attempts is skipped with a warning; the skipped granules are listed in attr(, "failed") of the result

Value

A data.frame with the columns time (decimal years), lon, lat, wse (meters above EGM2008), wse_qual, wse_uncert, water_frac, darkwater_frac, N_PIXC (number of pixel cloud points), ice_clim_flag and ice_dyn_flag (0 no ice, 1 uncertain/partial, 2 full ice cover), cross_track (signed cross-track distance, m) and granule. It can be used directly as input to getLakeGeoid()

Examples

```
# dat <- getSWOTRaster(1320035532, pld = "lakesub_simple.shp", startDate = "2024-01-01", endDate = "2024-06-30", ma
```

---

kivu.RData *Level 2 Water Mask Raster Image 250m Data Product, Version C extracted for lake Kivu*

Description

Selected columns of the Level 2 Water Mask Raster Image 250m Data Product, Version C. The data has been filtered by "darkwater_frac", large outliers were removed, and the data has been down sampled to reduce the volume.

Author(s)

SWOT Level 2 Water Mask Raster Image 250m Data Product, Version C (10.5067/SWOT-RASTER-2.0)

References

Surface Water Ocean Topography (SWOT). 2024. SWOT Level 2 Water Mask Raster Image Data Product, Version C. Ver. C. PO.DAAC, CA, USA. Dataset accessed 2025-04-01 at https://doi.org/10.5067/SWOT-RASTER-2.0

myshape.RData — *Shapefile of the African lake Kivu*

Description

A shapefile of the class "sf" for the African lake Kivu. The shapefile is taken from the SWOT Prior Lake Database.

Author(s)

Wang et al.

References

Wang, J., Pottier, C., Cazals, C., Battude, M., Sheng, Y., Song, C., Sikder, M.S., Yang, X., Ke, L., Gosset, M. and Oliveira, R.R.A., 2023. The Surface Water and Ocean Topography Mission (SWOT) Prior Lake Database (PLD): Lake mask and operational auxiliaries. Authorea Preprints.https://www.authorea.com/doi/full/10.22541/au.170258987.72387777

openLake — *Remove narrow parts (river channels, canals, narrow bays) from a lake polygon*

Description

The function openLake applies a morphological opening to the lake polygon in UTM coordinates: the polygon is shrunk by "width", only the main part is kept, and the result is grown back by "width" and intersected with the original polygon, so the shoreline of the main lake is unchanged. River channels and narrow bays often sit at a different water level (e.g. below a dam or along a river with a gradient) and would otherwise appear as a static signal in the estimated field.

Usage

```
openLake(lake, width = 300, keep = c("largest", "all"), minArea = 1)
```

Arguments

lake	A lake polygon of the class "sf" (e.g. from getPLDLake())
width	Half the width (m) of the parts to remove: parts narrower than about 2*width are removed
keep	"largest" keeps only the largest part after the inward buffer; "all" keeps every part larger than "minArea"
minArea	Minimum area (km^2) of a part kept with keep="all"

Value

An object of the class "sf" with the opened polygon in longitude/latitude (EPSG:4326); the attributes "width" and "removed_km2" give the opening width and the removed area

Examples

```
# lake <- openLake(getPLDLake(7250051493, "lakesub_simple.shp"), width = 300)
```

---

overpassSummary *Summary of each overpass*

Description

The function overpassSummary reduces the pixels to one row per overpass: the epoch (column "time", as used by getLakeGeoid()), cycle, pass, the start time of the granule (UTC), the number of pixels, the median (or mean) water surface elevation and its sampling error (1.2533*1.4826*MAD/sqrt(n) for the median, SD/sqrt(n) for the mean). The table is small, so it can be saved once and used instead of the pixels by getLakeLevel() and for comparisons with gauges.

Usage

```
overpassSummary(dat, stat = c("median", "mean"), minPix = 10)
```

Arguments

dat	A data.frame with SWOT pixels (columns "wse", "time" and, to separate overpasses on the same date, "granule"), e.g. the pixels flagged "ok" by flagOutliers()
stat	Summary of each overpass: "median" (default, robust) or "mean"
minPix	Minimum number of pixels for an overpass to be kept

Value

A data.frame with one row per overpass, sorted by time

---

plot.LakeGeoid *Plot an object returned by the function getLakeGeoid()*

Description

The static field is evaluated on a regular grid with the same linear interpolation within the mesh triangles as used in the model, and drawn as an image. The right panel shows the estimated water level time series.

Usage

```
## S3 method for class 'LakeGeoid'
plot(fit, zlim = NULL, dat = NULL, doSave = FALSE, res = 250, showMesh = FALSE)
```

Arguments

fit	Object returned by getLakeGeoid()
zlim	vector with the zlim interval c(low,high); if not specified the 0.1% and 99.9% quantiles of the field are used (but at least 5 grid cells at each end, for small lakes), so a few extreme grid cells (e.g. at the shore or an inlet) do not stretch the color scale. Values outside zlim are drawn with the end colors
dat	The raw water level data can be added to the plot
doSave	plot is saved to a pdf file
res	Resolution (in the units of the mesh, i.e. meters for UTM) of the grid the field is evaluated on
showMesh	Logical; draw the triangles of the mesh on top of the field

Examples

```
# plot(fit)
```

---

plotOverpass *Map of the observations from a single SWOT overpass*

Description

The function plotOverpass draws a map of the water surface elevation from one SWOT overpass (one cycle and pass, which may consist of several raster tiles). The title shows cycle|pass, the time of the first tile and the number of observations.

Usage

```
plotOverpass(
  dat,
  lake = NULL,
  cycle = NULL,
  pass = NULL,
  nsd = 2,
  robust = TRUE,
  zlim = NULL,
  cex = 0.6,
  showFlagged = TRUE
)
```

Arguments

dat	Data set returned by getSWOTRaster() (must contain the columns lon, lat, wse and granule)
lake	Optional lake polygon of the class "sf" drawn as outline
cycle	Cycle number; if NULL (and "pass" is NULL) the first overpass in "dat" is plotted
pass	Pass number
nsd	The color scale covers the center plus/minus "nsd" standard deviations of the wse of the overpass; values outside get the end colors
robust	Logical; if TRUE (default) the median and the MAD-based standard deviation are used for the color scale, so outliers do not stretch it. If FALSE the mean and standard deviation are used
zlim	Optional fixed color limits c(low,high); overrides "nsd"
cex	Point size
showFlagged	Logical; if "dat" has a column "flag" (from flagOutliers()), observations not flagged "ok" are drawn as gray crosses and are not used for the color scale. If FALSE flagged observations are not drawn

Examples

```
# plotOverpass(dat, lake, cycle = 7, pass = 249)
```

---

print.LakeGeoid *Simple print of convergence statutus*

Description

This function presents a summary of the output

Usage

```
## S3 method for class 'LakeGeoid'
print(x)
```

Arguments

x	An object of class "LakeGeoid"

Value

Print the objective function and state convergence

`summary.LakeGeoid` *Summary of output*

Description

This function presents a summary of the output

Usage

```
## S3 method for class 'LakeGeoid'
summary(x)
```

Arguments

`x`	An object of class "LakeGeoid"

Value

Summary of output

Index

∗ **data**
C2kivu.RData, 3
kivu.RData, 13
myshape.RData, 14
∗ **plot**
plot.LakeGeoid, 15
∗ **predict**
getGeoidCorr, 7
2025-04-01, *13*

addCrossTrack, 2

C2kivu.RData, 3
clipToLake, 3

downsample, 4

flagOutliers, 5

getGeoidCorr, 7
getGeoidGrid, 7
getLakeGeoid, 8
getLakeLevel, 10
getPLDLake, 11
getSWOTRaster, 12

kivu.RData, 13

myshape.RData, 14

openLake, 14
overpassSummary, 15

plot.LakeGeoid, 15
plotOverpass, 16
print.LakeGeoid, 17

summary.LakeGeoid, 18