Package 'glacierSMBM'

September 28, 2017

Type Package

Title Glacier Surface Mass Balance Model

Version 0.1

Date 2017-09-26

Author Alexander R. Groos [cre, aut], Christoph Mayer [ctb]

Maintainer Alexander R. Groos <alexander.groos@giub.unibe.ch>

Description A fully distributed glacier surface mass balance model developed for the simulation of accumulation and ablation processes on debris-free as well as debris-covered glaciers.

License GPL (>= 3)

Depends R (>= 2.10), methods, raster, sp, udunits2

LazyData true

NeedsCompilation no

Repository CRAN

Date/Publication 2017-09-28 20:52:27 UTC

R topics documented:

glacierSMBM-package *Package: Glacier Surface Mass Balance Model*

Description

A fully distributed glacier surface mass balance model developed for the simulation of accumulation and ablation processes on debris-free as well as debris-covered glaciers.

Details

Package:	glacierSMBM
Type:	Package
Version:	0.1
Date:	2017-09-26
License:	GPL (>= 3)
Depends:	methods, raster, sp, udunits2

Author(s)

Alexander R. Groos (<alexander.groos@giub.unibe.ch>)

Christoph Mayer

References

Groos, A.R., Mayer, C., Smiraglia, C., Diolaiuti, G., and Lambrecht A. (submitted). A first attempt to model region-wide glacier surface mass balances in the Karakoram: findings and future challenges. Geografia Fisica e Dinamica Quaternaria.

See Also

glacierSMBM

inputGlacierSMBM-class

glacialMelt, snowMelt, iceMelt,

debrisCoveredIceMelt, snowFall,

Examples

```
# Load the provided RasterLayer objects as exemplary
# input for the function
data(airTemperature_30m_daily, airDensity_30m_daily,
    netRad_30m_daily, glacierMask_30m, iceMask_30m, firnMask_30m,
    debrisMask_30m, debrisThickness_30m, precipTuningFactor_30m,
    snowFall_30m_daily, package = "glacierSMBM")
```

```
# Individual RasterLayer objects should be loaded or
# created using the function raster()

# create a three-day virtual meteorological data set
AirT <- stack(airTemperature_30m_daily,
    airTemperature_30m_daily * 0.99,
    airTemperature_30m_daily * 1.01)
NetRad <- stack(netRad_30m_daily,netRad_30m_daily * 0.99,
    netRad_30m_daily * 1.01)
Snowfall <- stack(snowFall_30m_daily, snowFall_30m_daily * 2,
    snowFall_30m_daily * 0.3)

# create a new object of class "inputGlacierSMBM" which is
# requested as input for the glacier surface mass balance model
InputGlacierSMBM <- new("inputGlacierSMBM")

# Add the required data and information to the respective
# slots of the new object (for additional setting options read
# the help section of class "inputGlacierSMBM")
# Create a numeric vector containing date and time of
# the meteorological input data
InputGlacierSMBM@date <- seq.POSIXt(ISOdate(2011,8,15),
    ISOdate(2011,8,17), "days")
InputGlacierSMBM@decimalPlaces <- 4
InputGlacierSMBM@airT <- AirT
InputGlacierSMBM@airDensity <- stack(airDensity_30m_daily)
InputGlacierSMBM@netRad <- NetRad
InputGlacierSMBM@snowfall <- Snowfall
InputGlacierSMBM@glacierMask <- stack(glacierMask_30m)
InputGlacierSMBM@iceMask <- stack(iceMask_30m)
InputGlacierSMBM@firnMask <- stack(firnMask_30m)
InputGlacierSMBM@debrisMask <- stack(debrisMask_30m)
InputGlacierSMBM@debrisThickness <- stack(debrisThickness_30m)
InputGlacierSMBM@disTuningFacPrecip <- stack(precipTuningFactor_30m)

# Calculate glacier surface mass balance using standard settings,
# but suppress to write any output
InputGlacierSMBM@writeOutput <- rep(0, 17)

## Not run:
output <- glacierSMBM(inputGlacierSMBM = InputGlacierSMBM)

# Plot output
plot(output, main = "glacier surface mass balance",
    legend.args=list(text='Mass balance (m d-1)', side=3,
    line=1.5), col = colorRampPalette(c("darkred", "red",
    "blue"))(100))


## End(Not run)

# Calculate glacier surface mass balance using modified settings
# Change thermal conductivity and wind speed applied in the
```

```
# implemented function "debrisCoveredIceMelt"
InputGlacierSMBM@thermalConductivity <- 1.5
InputGlacierSMBM@windSpeed <- 5

## Not run:
output <- glacierSMBM(inputGlacierSMBM = InputGlacierSMBM)

# Plot output
plot(output, main = "glacier surface mass balance",
    legend.args=list(text='Mass balance (m d-1)', side=3,
    line=1.5), col = colorRampPalette(c("darkred", "red",
    "blue"))(100))

## End(Not run)
```

`airDensity_30m_daily` *Data: Air density (30m, daily)*

Description

Distributed air density at the Liligo Glacier (Karakoram, Pakistan)

Usage

```
data(airDensity_30m_daily)
```

Format

An object of class `'RasterLayer'`.

Details

Dataset: The High Asia Refined Analysis (TU Berlin, Chair of Climatology)

Date: 2011-08-15

Temporal resolution: daily

Pixel resolution: 10 km

Unit: kg m-3

Projection: UTM 43 N

Note: The original dataset was resampled to a spatial resolution of 30 m using the function `resample`.

Source

High Asia Refined Analysis

References

Groos, A.R., Mayer, C., Smiraglia, C., Diolaiuti, G., and Lambrecht A. (submitted). A first attempt to model region-wide glacier surface mass balances in the Karakoram: findings and future challenges. Geografia Fisica e Dinamica Quaternaria.

Maussion, F., Scherer, D., Moelg, T., Collier, E., Curio, J., and Finkelnburg, R. (2014). Precipitation Seasonality and Variability over the Tibetan Plateau as Resolved by the High Asia Reanalysis. Journal of Climate 27, 1910-1927.

Examples

```
data(airDensity_30m_daily)
plot(airDensity_30m_daily)
```

`airPressure_10km_daily`

Data: Air pressure (10km, daily)

Description

Distributed air pressure at the Liligo Glacier (Karakoram, Pakistan)

Usage

```
data(airPressure_10km_daily)
```

Format

An object of class `'RasterLayer'`.

Details

Dataset: The High Asia Refined Analysis (TU Berlin, Chair of Climatology)

Date: 2011-08-15

Temporal resolution: daily

Pixel resolution: 10 km

Unit: Pa

Projection: UTM 43 N

Source

High Asia Refined Analysis

References

Groos, A.R., Mayer, C., Smiraglia, C., Diolaiuti, G., and Lambrecht A. (submitted). A first attempt to model region-wide glacier surface mass balances in the Karakoram: findings and future challenges. Geografia Fisica e Dinamica Quaternaria.

Maussion, F., Scherer, D., Moelg, T., Collier, E., Curio, J., and Finkelnburg, R. (2014). Precipitation Seasonality and Variability over the Tibetan Plateau as Resolved by the High Asia Reanalysis. Journal of Climate 27, 1910-1927.

Examples

```
data(airPressure_10km_daily)
plot(airPressure_10km_daily)
```

---

airPressure_30m_hourly

Data: Air pressure (30m, daily)

Description

Distributed air pressure at the Liligo Glacier (Karakoram, Pakistan)

Usage

```
data(airPressure_30m_hourly)
```

Format

An object of class `'RasterLayer'`.

Details

Dataset: The High Asia Refined Analysis (TU Berlin, Chair of Climatology)

Date: 2011-08-10

Temporal resolution: daily

Pixel resolution: 10 km

Unit: Pa

Projection: UTM 43 N

Note: The original dataset was interpolated to a spatial resolution of 30 m using the function `interpolateAirP`.

Source

High Asia Refined Analysis

References

Groos, A.R., Mayer, C., Smiraglia, C., Diolaiuti, G., and Lambrecht A. (submitted). A first attempt to model region-wide glacier surface mass balances in the Karakoram: findings and future challenges. Geografia Fisica e Dinamica Quaternaria.

Maussion, F., Scherer, D., Moelg, T., Collier, E., Curio, J., and Finkelnburg, R. (2014). Precipitation Seasonality and Variability over the Tibetan Plateau as Resolved by the High Asia Reanalysis. Journal of Climate 27, 1910-1927.

Examples

```
data(airPressure_30m_hourly)
plot(airPressure_30m_hourly)
```

`airTemperature_10km_daily`

Data: Air temperature (10km, daily)

Description

Distributed air temperature at the Liligo Glacier (Karakoram, Pakistan)

Usage

```
data(airTemperature_10km_daily)
```

Format

An object of class `'RasterLayer'`.

Details

Dataset: The High Asia Refined Analysis (TU Berlin, Chair of Climatology)

Date: 2011-08-15

Temporal resolution: daily

Pixel resolution: 10 km

Unit: K

Note: Projection: UTM 43 N

Source

High Asia Refined Analysis

References

Groos, A.R., Mayer, C., Smiraglia, C., Diolaiuti, G., and Lambrecht A. (submitted). A first attempt to model region-wide glacier surface mass balances in the Karakoram: findings and future challenges. Geografia Fisica e Dinamica Quaternaria.

Maussion, F., Scherer, D., Moelg, T., Collier, E., Curio, J., and Finkelnburg, R. (2014). Precipitation Seasonality and Variability over the Tibetan Plateau as Resolved by the High Asia Reanalysis. Journal of Climate 27, 1910-1927.

Examples

```
data(airTemperature_10km_daily)
plot(airTemperature_10km_daily)
```

---

airTemperature_30m_daily

Data: Air temperature (30m, daily)

Description

Distributed air temperature at the Liligo Glacier (Karakoram, Pakistan)

Usage

```
data(airTemperature_30m_daily)
```

Format

An object of class `'RasterLayer'`.

Details

Dataset: The High Asia Refined Analysis (TU Berlin, Chair of Climatology)

Date: 2011-08-15

Temporal resolution: daily

Pixel resolution: 10 km

Unit: K

Projection: UTM 43 N

Note: The original dataset was interpolated to a spatial resolution of 30 m using the function `interpolateAirT`.

Source

High Asia Refined Analysis

References

Groos, A.R., Mayer, C., Smiraglia, C., Diolaiuti, G., and Lambrecht A. (submitted). A first attempt to model region-wide glacier surface mass balances in the Karakoram: findings and future challenges. Geografia Fisica e Dinamica Quaternaria.

Maussion, F., Scherer, D., Moelg, T., Collier, E., Curio, J., and Finkelnburg, R. (2014). Precipitation Seasonality and Variability over the Tibetan Plateau as Resolved by the High Asia Reanalysis. Journal of Climate 27, 1910-1927.

Examples

```
data(airTemperature_30m_daily)
plot(airTemperature_30m_daily)
```

---

airTemperature_30m_hourly

Data: Air temperature (30m, hourly)

Description

Distributed air temperature at the Liligo Glacier (Karakoram, Pakistan)

Usage

```
data(airTemperature_30m_hourly)
```

Format

An object of class `'RasterLayer'`.

Details

Dataset: The High Asia Refined Analysis (TU Berlin, Chair of Climatology)

Date: 2011-08-15

Temporal resolution: hourly

Pixel resolution: 10 km

Unit: K

Projection: UTM 43 N

Note: The original dataset was interpolated to a spatial resolution of 30 m using the function `interpolateAirT`.

Source

High Asia Refined Analysis

References

Groos, A.R., Mayer, C., Smiraglia, C., Diolaiuti, G., and Lambrecht A. (submitted). A first attempt to model region-wide glacier surface mass balances in the Karakoram: findings and future challenges. Geografia Fisica e Dinamica Quaternaria.

Maussion, F., Scherer, D., Moelg, T., Collier, E., Curio, J., and Finkelnburg, R. (2014). Precipitation Seasonality and Variability over the Tibetan Plateau as Resolved by the High Asia Reanalysis. Journal of Climate 27, 1910-1927.

Examples

```
data(airTemperature_30m_hourly)
plot(airTemperature_30m_hourly)
```

---

debrisCoveredIceMelt *Function: Sub-debris ice melt model*

Description

An energy balance model to calculate glacial melt under a porous debris layer.

Usage

```
debrisCoveredIceMelt(airT, airDensity, glacierMask, debrisMask,
    debrisThickness, inRadSW = stack(), inRadLW = stack(),
    netRad = stack(), tUnit = "K", tuningFacAirT = 1,
    disTuningFacAirT = stack(), tmpRes = "d", measurementHeight=2,
    relativeHumidity = 0.73, disRelativeHumidity = stack(),
    windSpeed = 2, disWindSpeed = stack(), debrisAlbedo = 0.07,
    disDebrisAlbedo = stack(), thermalConductivity = 0.585,
    disThermalConductivity = stack(), thermalEmissivity = 0.95,
    disThermalEmissivity = stack(), surfaceRoughnessHeight = 0.01,
    disSurfaceRoughnessHeight = stack(), frictionVelocity = 0.16,
    disFrictionVelocity = stack(), volumeFractionDebrisInIce = 0.01,
    disVolumeFractionDebrisInIce = stack(), debrisAirRatio = 188,
    disDebrisAirRatio = stack(),dragCoefficient = 5,
    disDragCoefficient = stack(), iceDensity = 900,
    disIceDensity = stack(), decimalPlaces = 4, outType = "mean",
    writeOutput = FALSE, outputName = "dcIceMelt", tmpCreate =
    FALSE, tmpDir = "", outDir = "", ... )
```

Arguments

airT	An object of class 'RasterStack'. Distributed air temperature (Kelvin or degree Celsius). For every time step.
airDensity	An object of class 'RasterStack'. Distributed air density (kg m-3). Stationary or for every time step.

`glacierMask` An object of class `'RasterStack'`. Glacier area (1 = glacier, 0 = no glacier). Stationary or for every time step.

`debrisMask` An object of class `'RasterStack'`. Area of debris covered glacier ice (1 = debris, 0 = no debris). Stationary or for every time step.

`debrisThickness`
An object of class `'RasterStack'`. Distributed supraglacial debris thickness (m). Stationary or for every time step.

`inRadSW` An object of class `'RasterStack'`. Distributed incoming shortwave radiation (W m-2). For every time step.

`inRadLW` An object of class `'RasterStack'`. Distributed incoming longwave radiation (W m-2). For every time step.

`netRad` An object of class `'RasterStack'`. Distributed net radiation (W m-2). For every time step. Optional instead of `'inRadSW'` and `'inRadLW'`.

`tUnit` An object of class `'character'`. Unit ("K" = Kelvin, "C" = degree Celsius) of air temperature (default = "K").

`tuningFacAirT` An object of class `'numeric'`. General air temperature tuning factor (<1 = temperature decrease, 1 = default, >1 = temperature increase).

`disTuningFacAirT`
An object of class `'RasterStack'`. Distributed air temperature tuning factor (`tuningFacAirT`). Stationary or for every time step.

`tmpRes` An object of class `'character'`. Time aggregation (temporal resolution) of the input variables (default = "d"). "y" = year, "w" = week, "d" = day, "h" = hour, "s" = second.

`measurementHeight`
An object of class `'numeric'`. Height (m) of meteorological measurements (default = 2).

`relativeHumidity`
An object of class `'numeric'`. Relative humidity (0-1) at measurement height (default = 0.73).

`disRelativeHumidity`
An object of class `'RasterStack'`. Distributed relative humidity (0-1) at measurement height. Stationary or for every time step.

`windSpeed` An object of class `'numeric'`. Wind speed (m s-1) at measurement height (default = 2).

`disWindSpeed` An object of class `'RasterStack'`. Distributed wind speed (m s-1) at measurement height. Stationary or for every time step.

`debrisAlbedo` An object of class `'numeric'`. Albedo (0-1) of the debris (default = 0.07).

`disDebrisAlbedo`
An object of class `'RasterStack'`. Distributed albedo (0-1) of the debris. Stationary or for every time step.

`thermalConductivity`
An object of class `'numeric'`. Thermal conductivity (W m-1 K-1) of the debris layer (default = 0.585).

`disThermalConductivity`
: An object of class `'RasterStack'`. Distributed thermal conductivity (W m-1 K-1) of the debris layer. Stationary or for every time step.

`thermalEmissivity`
: An object of class `'numeric'`. Thermal emissivity (0-1) of the debris layer (default = 0.95).

`disThermalEmissivity`
: An object of class `'RasterStack'`. Distributed thermal emissivity (0-1) of the debris layer. Stationary or for every time step.

`surfaceRoughnessHeight`
: An object of class `'numeric'`. Surface roughness height (m) of the debris layer (default = 0.01).

`disSurfaceRoughnessHeight`
: An object of class `'RasterStack'`. Distributed surface roughness height (m) of the debris layer. Stationary or for every time step.

`frictionVelocity`
: An object of class `'numeric'`. Friction velocity (m s-1) of the debris layer (default = 0.16).

`disFrictionVelocity`
: An object of class `'RasterStack'`. Distributed friction velocity (m s-1) of the debris layer. Stationary or for every time step.

`volumeFractionDebrisInIce`
: An object of class `'numeric'`. Volume fraction (0-1) of debris in the ice body (default = 0.01).

`disVolumeFractionDebrisInIce`
: An object of class `'RasterStack'`. Distributed volume fraction (0-1) of debris in the ice body. Stationary or for every time step.

`debrisAirRatio`
: An object of class `'numeric'`. Ratio of the debris surface area to the volume of air (m-1) in the debris layer (default = 188).

`disDebrisAirRatio`
: An object of class `'RasterStack'`. Distributed ratio of the debris surface area to the volume of air (m-1) in the debris layer. Stationary or for every time step.

`dragCoefficient`
: An object of class `'numeric'`. Drag coefficient (m-1) of the debris layer (default = 5).

`disDragCoefficient`
: An object of class `'RasterStack'`. Distributed drag coefficient (m-1) of the debris layer. Stationary or for every time step.

`iceDensity`
: An object of class `'numeric'`. Density (kg m-3) of ice (default = 900).

`disIceDensity`
: An object of class `'RasterStack'`. Distributed density (kg m-3) of ice. Stationary or for every time step.

`decimalPlaces`
: An object of class `'numeric'`. Number of decimal places (default = 4).

`outType`
: An object of class `'character'`. Type of output to be returned by the function: "mean" (default) sub-debris ice melt or "sum".

writeOutput	An object of class 'logical'. Determines whether the ouput shall be exported as RasterLayer (TRUE) or not (FALSE, default).
outputName	An object of class 'character'. File name for the output RasterLayer(s) (default = "dcIceMelt").
tmpCreate	An object of class 'logical'. Determines whether a temporary directory should be used (TRUE) or not (FALSE, default). Recommendend if large datasets are processed.
tmpDir	An object of class 'character'. Directory where processing files can be temporarily stored if 'tmpCreate' = TRUE.
outDir	An object of class 'character'. Directory for the output files if 'writeOutput' = TRUE.
...	Further arguments.

Details

The impact of supraglacial debris on the melting process of underlying ice depends on the thickness of the debris layer itself. A thin dust layer of several centimeters enhances ice melt due to increased radiative absorption, whereas a thick debris cover (>4-5 cm) isolates and reduces ablation (e.g. Mihalcea et al. 2006, 2008; Mayer et al., 2010). Since the relationship between debris thickness and ice melt is non-linear, simple melting factors (also known as "degree-day factors") are not applicable. An energy-balance model developed by Evatt et al. (2015, Equations 41-46) is therefore applied to calculate glacial melt under a porous debris layer (for detailed information please refer to Groos et al. (submitted, Equations 11-16).

The sub-debris ice melt model consideres the following energy fluxes:

- shortwave energy flux
- longwave energy flux
- sensible heat flux
- heat flux due to evaporation at the debris-ice interface
- latent heat flux due to melting
- heat flux within the debris layer

Value

An object of class 'RasterLayer' returning the calculated spatial distribution of sub-debris ice melt (e.g. in m d-1, depending on 'tmpRes').

Note

The following input variables are the requested minimum to run the model:

- 'airT' (for every time step)
- 'inRadSW' and 'inRadLW' or 'netRad' (for every time step)
- 'airDensity' (stationary or for every time step)
- 'glacierMask' (stationary or for every time step)

- 'debrisMask' (stationary or for every time step)
- 'debrisThickness' (stationary or for every time step)

If 'inRadSW' and 'inRadLW' are provided instead of 'netRad', the energy-balance at the atmosphere-debris interface is calculated taking the 'debrisAlbedo' and 'thermalEmissivity' of the debris layer into account.

A default value (constant in space and time) is given for each additional argument like 'windSpeed', 'relativeHumidity' and 'thermalConductivity'. If desired, the default parameters can be modified. Furthermore, there is the option to pass distributed values (stationary or for every time step) instead of general values using the related 'dis*'-arguments like 'disWindSpeed', 'disRelativeHumidity' and 'disThermalConductivity'. In this case, the general parameter is ignored.

File format of written ouput: GeoTIFF.

Author(s)

Alexander R. Groos (<alexander.groos@giub.unibe.ch>)

References

Evatt, G.W., Abrahams, D., Heil, M., Mayer, C., Kingslake, J., Mitchell, S.L., Fowler, A.C., and Clark, C.D. (2015). Glacial melt under a porous debris layer. Journal of Glaciology 61, 825-836.

Groos, A.R., Mayer, C., Smiraglia, C., Diolaiuti, G., and Lambrecht A. (submitted). A first attempt to model region-wide glacier surface mass balances in the Karakoram: findings and future challenges. Geografia Fisica e Dinamica Quaternaria.

Mayer, C., Lambrecht, A., Mihalcea, C., Belo, M., Diolaiuti, G., Smiraglia, C., and Bashir, F. (2010). Analysis of Glacial Meltwater in Bagrot Valley, Karakoram. Mountain Research and Development 30, 169-177.

Mihalcea, C., Mayer, C., Diolaiuti, G., Lambrecht, A., Smiraglia, C., and Tartari, G. (2006). Ice ablation and meteorological conditions on the debris-covered area of Baltoro glacier, Karakoram, Pakistan. Annals of Glaciology 43, 292-300.

Mihalcea, C., Mayer, C., Diolaiuti, G., D'Agata, C., Smiraglia, C., Lambrecht, A., Vuillermoz, E., and Tartari, G. (2008). Spatial distribution of debris thickness and melting from remote-sensing and meteorological data, at debris-covered Baltoro glacier, Karakoram, Pakistan. Annals of Glaciology 48, 49-57.

See Also

glacialMelt, snowMelt, iceMelt

Examples

```
# Load the provided RasterLayer objects
# as exemplary input for the function
data(glacierMask_30m, debrisThickness_30m, debrisMask_30m,
    airTemperature_30m_daily, airDensity_30m_daily,
    netRad_30m_daily, package = "glacierSMBM")
# Individual RasterLayer objects should be loaded
# or created using the function raster()
```

```
# Include RasterLayers in RasterStack
GlacierMask <- stack(glacierMask_30m)
DebrisThickness <- stack(debrisThickness_30m)
DebrisMask <- stack(debrisMask_30m)
AirTemperature <- stack(airTemperature_30m_daily)
AirDensity <- stack(airDensity_30m_daily)
NetRad <- stack(netRad_30m_daily)

# Calculate ice melt under a porous debris layer
# using standard settings
output <- debrisCoveredIceMelt(airT = AirTemperature,
    netRad = NetRad, airDensity = AirDensity,
    glacierMask = GlacierMask, debrisMask = DebrisMask,
    debrisThickness = DebrisThickness)

# Plot output
plot(output, main = "debris covered ice melt",
    legend.args=list(text='Ice melt (m d-1)', side=3, line=1.5))

# Calculate ice melt under a porous debris layer using modified
# settings (e.g. change numeric values for thermal conductivity
# and temporal resolution)
output <- debrisCoveredIceMelt(airT = AirTemperature,
    netRad = NetRad, airDensity = AirDensity,
    glacierMask = GlacierMask, debrisMask = DebrisMask,
    debrisThickness = DebrisThickness, thermalConductivity = 1.5,
    tmpRes = "h")

# Plot output
plot(output, main = "debris covered ice melt",
    legend.args=list(text='Ice melt (m h-1)', side=3, line=1.5))
```

`debrisCoveredIceMelt-method`

Method: Sub-debris ice melt model

Description

An energy balance model to calculate glacial melt under a porous debris layer.

Details

The impact of supraglacial debris on the melting process of underlying ice depends on the thickness of the debris layer itself. A thin dust layer of several centimeters enhances ice melt due to increased radiative absorption, whereas a thick debris cover (>4-5 cm) isolates and reduces ablation (e.g. Mihalcea et al. 2006, 2008; Mayer et al., 2010). Since the relationship between debris thickness and ice melt is non-linear, simple melting factors (also known as "degree-day factors") are not applicable. An energy-balance model developed by Evatt et al. (2015, Equations 41-46) is therefore

applied to calculate glacial melt under a porous debris layer (for detailed information please refer to Groos et al. (submitted, Equations 11-16).

The sub-debris ice melt model consideres the following energy fluxes:

- shortwave energy flux
- longwave energy flux
- sensible heat flux
- heat flux due to evaporation at the debris-ice interface
- latent heat flux due to melting
- heat flux within the debris layer

Value

An object of class `'RasterLayer'` returning the calculated spatial distribution of sub-debris ice melt (e.g. in m d-1, depending on `'tmpRes'`).

Note

The following input variables are the requested minimum to run the model:

- `'airT'` (for every time step)
- `'inRadSW'` and `'inRadLW'` or `'netRad'` (for every time step)
- `'airDensity'` (stationary or for every time step)
- `'glacierMask'` (stationary or for every time step)
- `'debrisMask'` (stationary or for every time step)
- `'debrisThickness'` (stationary or for every time step)

If `'inRadSW'` and `'inRadLW'` are provided instead of `'netRad'`, the energy-balance at the atmosphere-debris interface is calculated taking the `'debrisAlbedo'` and `'thermalEmissivity'` of the debris layer into account.

A default value (constant in space and time) is given for each additional argument like `'windSpeed'`, `'relativeHumidity'` and `'thermalConductivity'`. If desired, the default parameters can be modified. Furthermore, there is the option to pass distributed values (stationary or for every time step) instead of general values using the related 'dis*'-arguments like `'disWindSpeed'`, `'disRelativeHumidity'` and `'disThermalConductivity'`. In this case, the general parameter is ignored.

File format of written ouput: GeoTIFF.

Author(s)

Alexander R. Groos (<alexander.groos@giub.unibe.ch>)

References

Evatt, G.W., Abrahams, D., Heil, M., Mayer, C., Kingslake, J., Mitchell, S.L., Fowler, A.C., and Clark, C.D. (2015). Glacial melt under a porous debris layer. Journal of Glaciology 61, 825-836.

Groos, A.R., Mayer, C., Smiraglia, C., Diolaiuti, G., and Lambrecht A. (submitted). A first attempt to model region-wide glacier surface mass balances in the Karakoram: findings and future challenges. Geografia Fisica e Dinamica Quaternaria.

Mayer, C., Lambrecht, A., Mihalcea, C., Belo, M., Diolaiuti, G., Smiraglia, C., and Bashir, F. (2010). Analysis of Glacial Meltwater in Bagrot Valley, Karakoram. Mountain Research and Development 30, 169-177.

Mihalcea, C., Mayer, C., Diolaiuti, G., Lambrecht, A., Smiraglia, C., and Tartari, G. (2006). Ice ablation and meteorological conditions on the debris-covered area of Baltoro glacier, Karakoram, Pakistan. Annals of Glaciology 43, 292-300.

Mihalcea, C., Mayer, C., Diolaiuti, G., D'Agata, C., Smiraglia, C., Lambrecht, A., Vuillermoz, E., and Tartari, G. (2008). Spatial distribution of debris thickness and melting from remote-sensing and meteorological data, at debris-covered Baltoro glacier, Karakoram, Pakistan. Annals of Glaciology 48, 49-57.

See Also

glacialMelt, snowMelt, iceMelt

Examples

```
# Load the provided RasterLayer objects
# as exemplary input for the function
data(glacierMask_30m, debrisThickness_30m, debrisMask_30m,
    airTemperature_30m_daily, airDensity_30m_daily,
    netRad_30m_daily, package = "glacierSMBM")
# Individual RasterLayer objects should be loaded
# or created using the function raster()

# Include RasterLayers in RasterStack
GlacierMask <- stack(glacierMask_30m)
DebrisThickness <- stack(debrisThickness_30m)
DebrisMask <- stack(debrisMask_30m)
AirTemperature <- stack(airTemperature_30m_daily)
AirDensity <- stack(airDensity_30m_daily)
NetRad <- stack(netRad_30m_daily)

# Calculate ice melt under a porous debris layer
# using standard settings
output <- debrisCoveredIceMelt(airT = AirTemperature,
    netRad = NetRad, airDensity = AirDensity,
    glacierMask = GlacierMask, debrisMask = DebrisMask,
    debrisThickness = DebrisThickness)

# Plot output
plot(output, main = "debris covered ice melt",
    legend.args=list(text='Ice melt (m d-1)', side=3, line=1.5))
```

```
# Calculate ice melt under a porous debris layer using modified
# settings (e.g. change numeric values for thermal conductivity
# and temporal resolution)
output <- debrisCoveredIceMelt(airT = AirTemperature,
    netRad = NetRad, airDensity = AirDensity,
    glacierMask = GlacierMask, debrisMask = DebrisMask,
    debrisThickness = DebrisThickness, thermalConductivity = 1.5,
    tmpRes = "h")

# Plot output
plot(output, main = "debris covered ice melt",
    legend.args=list(text='Ice melt (m h-1)', side=3, line=1.5))
```

`debrisMask_30m` *Data: Debris mask (30m)*

Description

Debris cover at the Liligo Glacier (Karakoram, Pakistan)

Usage

```
data(debrisMask_30m)
```

Format

An object of class `'RasterLayer'`.

Details

Dataset: Landsat 5

Date: 2011-08-10

Pixel resolution: 30 m

1 = debris, 0 = no debris

Projection: UTM 43 N

Note: The debris cover distribution was derived from a Landsat 5 image (for more information see Groos et al., submitted).

Source

USGS EarthExplorer

References

Groos, A.R., Mayer, C., Smiraglia, C., Diolaiuti, G., and Lambrecht A. (submitted). A first attempt to model region-wide glacier surface mass balances in the Karakoram: findings and future challenges. Geografia Fisica e Dinamica Quaternaria.

Examples

```
data(debrisMask_30m)
plot(debrisMask_30m)
```

`debrisThicknessEmp` *Function: Empirical debris thickness model*

Description

A simple empirical model to derive supraglacial debris thickness from land surface temperature.

Usage

```
debrisThicknessEmp(fittingParameters, surfaceTemperature = c(),
    disSurfaceTemperature = stack(), decimalPlaces = 4)
```

Arguments

`fittingParameters`	An object of class `'numeric'`. Two fitting parameters (output of `debrisThicknessFit`).
`surfaceTemperature`	An object of class `'numeric'`. Point information (as vector) of the glacier surface temperature (K).
`disSurfaceTemperature`	An object of class `'RasterStack'`. Spatial distribution of the glacier surface temperature (K). Optional instead of `surfaceTemperature`.
`decimalPlaces`	An object of class `'numeric'`. Number of decimal places (default = 4).

Details

The spatial distribution and thickness of supraglacial debris can be derived from remotely sensed surface temperatures based on an empirical relationship as shown by Mihalcea et al. (2006, 2008a, 2008b). High surface temperatures are correlated with thick debris, whereas surface temperatures closer to or below the melting point indicate a thin or absent debris layer. An exponential function with two fitting parameters (fp) was found to be most suitable to predict debris thickness from surface temperature (Minora et al., 2015; Groos et al., submitted, Equations 3-4):

debrisThickness = exp(fp_1 * surfaceTemperature - fp_2)

A prerequisite for the application of the empirical model is the availability of at least some (in-situ) debris thickness measurements from the study area, since they are required for the calibration of the model (`debrisThicknessFit`).

Value

An object of class `'RasterLayer'` or `'numeric'` (depending on the input) returning the calculated debris thickness (m).

Note

File format of written ouput: GeoTIFF.

Author(s)

Alexander R. Groos (<alexander.groos@giub.unibe.ch>)

References

Groos, A.R., Mayer, C., Smiraglia, C., Diolaiuti, G., and Lambrecht A. (submitted). A first attempt to model region-wide glacier surface mass balances in the Karakoram: findings and future challenges. Geografia Fisica e Dinamica Quaternaria.

Mihalcea, C., Mayer, C., Diolaiuti, G., Lambrecht, A., Smiraglia, C., and Tartari, G. (2006). Ice ablation and meteorological conditions on the debris-covered area of Baltoro glacier, Karakoram, Pakistan. Annals of Glaciology 43, 292-300.

Mihalcea, C., Brock, B.W., Diolaiuti, G., D'Agata, C., Citterio, M., Kirkbride, M.P., Cutler, M.E.J., and Smiraglia, C. (2008a). Using ASTER satellite and ground-based surface temperature measurements to derive supraglacial debris cover and thickness patterns on Miage Glacier (Mont Blanc Massif, Italy). Cold Regions Science and Technology 52, 341-354.

Mihalcea, C., Mayer, C., Diolaiuti, G., D'Agata, C., Smiraglia, C., Lambrecht, A., Vuillermoz, E., and Tartari, G. (2008b). Spatial distribution of debris thickness and melting from remote-sensing and meteorological data, at debris-covered Baltoro glacier, Karakoram, Pakistan. Annals of Glaciology 48, 49-57.

Minora, U., Senese, A., Bocchiola, D., Soncini, A., D'Agata, C., Ambrosini, R., Mayer, C., Lambrecht, A., Vuillermoz, E., Smiraglia, C., et al. (2015). A simple model to evaluate ice melt over the ablation area of glaciers in the Central Karakoram National Park, Pakistan. Annals of Glaciology 56, 202-216.

See Also

debrisThicknessFit, debrisThicknessPhy

Examples

```
# Load the provided data set and RasterLayer as exemplary
# input for the function. The values of the data set do not
# represent real field measurements and were only created for
# demonstration purposes
data(debrisThickness_measured, lst_measured, lst_30m_hourly,
    package = "glacierSMBM")
# Individual data sets or RasterLayers should be loaded using
# the functions read.*() or raster(), respectively

# Calculate the required fitting parameters for the
# function debrisThicknessEmp()
Fitting_Parameters <- debrisThicknessFit(surfaceTemperature =
    lst_measured, debrisThickness = debrisThickness_measured,
    plotOutput = FALSE)
```

```
# Derive debris thickness from land surface temperature using
# an empirical model
output <- debrisThicknessEmp(disSurfaceTemperature =
    lst_30m_hourly, fittingParameters = Fitting_Parameters)

# Plot output
plot(output, main = "debris thickness",
    legend.args=list(text='Debris thickness (m)',
    side=3, line=1.5))
```

`debrisThicknessEmp-method`

Method: Empirical debris thickness model

Description

A simple empirical model to derive supraglacial debris thickness from land surface temperature.

Details

The spatial distribution and thickness of supraglacial debris can be derived from remotely sensed surface temperatures based on an empirical relationship as shown by Mihalcea et al. (2006, 2008a, 2008b). High surface temperatures are correlated with thick debris, whereas surface temperatures closer to or below the melting point indicate a thin or absent debris layer. An exponential function with two fitting parameters (fp) was found to be most suitable to predict debris thickness from surface temperature (Minora et al., 2015; Groos et al., submitted, Equations 3-4):

debrisThickness = exp(fp_1 * surfaceTemperature - fp_2)

A prerequisite for the application of the empirical model is the availability of at least some (in-situ) debris thickness measurements from the study area, since they are required for the calibration of the model (`debrisThicknessFit`).

Value

An object of class `'RasterLayer'` or `'numeric'` (depending on the input) returning the calculated debris thickness (m).

Note

File format of written ouput: GeoTIFF.

Author(s)

Alexander R. Groos (<alexander.groos@giub.unibe.ch>)

References

Groos, A.R., Mayer, C., Smiraglia, C., Diolaiuti, G., and Lambrecht A. (submitted). A first attempt to model region-wide glacier surface mass balances in the Karakoram: findings and future challenges. Geografia Fisica e Dinamica Quaternaria.

Mihalcea, C., Mayer, C., Diolaiuti, G., Lambrecht, A., Smiraglia, C., and Tartari, G. (2006). Ice ablation and meteorological conditions on the debris-covered area of Baltoro glacier, Karakoram, Pakistan. Annals of Glaciology 43, 292-300.

Mihalcea, C., Brock, B.W., Diolaiuti, G., D'Agata, C., Citterio, M., Kirkbride, M.P., Cutler, M.E.J., and Smiraglia, C. (2008a). Using ASTER satellite and ground-based surface temperature measurements to derive supraglacial debris cover and thickness patterns on Miage Glacier (Mont Blanc Massif, Italy). Cold Regions Science and Technology 52, 341-354.

Mihalcea, C., Mayer, C., Diolaiuti, G., D'Agata, C., Smiraglia, C., Lambrecht, A., Vuillermoz, E., and Tartari, G. (2008b). Spatial distribution of debris thickness and melting from remote-sensing and meteorological data, at debris-covered Baltoro glacier, Karakoram, Pakistan. Annals of Glaciology 48, 49-57.

Minora, U., Senese, A., Bocchiola, D., Soncini, A., D'Agata, C., Ambrosini, R., Mayer, C., Lambrecht, A., Vuillermoz, E., Smiraglia, C., et al. (2015). A simple model to evaluate ice melt over the ablation area of glaciers in the Central Karakoram National Park, Pakistan. Annals of Glaciology 56, 202-216.

See Also

debrisThicknessFit, debrisThicknessPhy

Examples

```
# Load the provided data set and RasterLayer as exemplary
# input for the function. The values of the data set do not
# represent real field measurements and were only created for
# demonstration purposes
data(debrisThickness_measured, lst_measured, lst_30m_hourly,
    package = "glacierSMBM")
# Individual data sets or RasterLayers should be loaded using
# the functions read.*() or raster(), respectively

# Calculate the required fitting parameters for the
# function debrisThicknessEmp()
Fitting_Parameters <- debrisThicknessFit(surfaceTemperature =
    lst_measured, debrisThickness = debrisThickness_measured,
    plotOutput = FALSE)

# Derive debris thickness from land surface temperature using
# an empirical model
output <- debrisThicknessEmp(disSurfaceTemperature =
    lst_30m_hourly, fittingParameters = Fitting_Parameters)

# Plot output
plot(output, main = "debris thickness",
    legend.args=list(text='Debris thickness (m)',
```

```
    side=3, line=1.5))
```

`debrisThicknessFit` *Function: Debris thickness fitting*

Description

A function to fit remotely sensed surface temperatures to measured debris thickness.

Usage

```
debrisThicknessFit(surfaceTemperature, debrisThickness,
    plotOutput = FALSE)
```

Arguments

`surfaceTemperature`	An object of class `'numeric'`. Remotely sensed (or in-situ measured) surface temperature (in K) for the respective sites of debris thickness measurements.
`debrisThickness`	An object of class `'numeric'`. Measured debris thickness (m).
`plotOutput`	An object of class `'logical'`. Determines whether to plot the results (`TRUE`) or not (`FALSE`, default).

Details

The spatial distribution and thickness of supraglacial debris can be derived from remotely sensed surface temperatures based on an empirical relationship as shown by Mihalcea et al. (2006, 2008a, 2008b). High surface temperatures are correlated with thick debris, whereas surface temperatures closer to or below the melting point indicate a thin or absent debris layer. An exponential function with two fitting parameters (fp) was found to be most suitable to predict debris thickness from surface temperature (Minora et al., 2015). To derive debris thickness from surface temperature, an empirical non-linear model (`debrisThicknessEmp`) is applied (Groos et al., submitted). The two fitting parameters of the model are obtained by iteratively comparing measured and modelled debris thickness using varying starting values. Calculated non-linear (weighted) least-squares (`nls`) serve for the selection of the optimal fitting parameters.

Value

An object of class `'numeric'` containing the two fitting parameters.

Note

The function `extractRasterValues` may help to extract the respective surface temperature values from a `'RasterLayer'` at the locations of debris thickness measurements.

Author(s)

Alexander R. Groos (<alexander.groos@giub.unibe.ch>)

References

Groos, A.R., Mayer, C., Smiraglia, C., Diolaiuti, G., and Lambrecht A. (submitted). A first attempt to model region-wide glacier surface mass balances in the Karakoram: findings and future challenges. Geografia Fisica e Dinamica Quaternaria.

Mihalcea, C., Mayer, C., Diolaiuti, G., Lambrecht, A., Smiraglia, C., and Tartari, G. (2006). Ice ablation and meteorological conditions on the debris-covered area of Baltoro glacier, Karakoram, Pakistan. Annals of Glaciology 43, 292-300.

Mihalcea, C., Brock, B.W., Diolaiuti, G., D'Agata, C., Citterio, M., Kirkbride, M.P., Cutler, M.E.J., and Smiraglia, C. (2008a). Using ASTER satellite and ground-based surface temperature measurements to derive supraglacial debris cover and thickness patterns on Miage Glacier (Mont Blanc Massif, Italy). Cold Regions Science and Technology 52, 341-354.

Mihalcea, C., Mayer, C., Diolaiuti, G., D'Agata, C., Smiraglia, C., Lambrecht, A., Vuillermoz, E., and Tartari, G. (2008b). Spatial distribution of debris thickness and melting from remote-sensing and meteorological data, at debris-covered Baltoro glacier, Karakoram, Pakistan. Annals of Glaciology 48, 49-57.

Minora, U., Senese, A., Bocchiola, D., Soncini, A., D'Agata, C., Ambrosini, R., Mayer, C., Lambrecht, A., Vuillermoz, E., Smiraglia, C., et al. (2015). A simple model to evaluate ice melt over the ablation area of glaciers in the Central Karakoram National Park, Pakistan. Annals of Glaciology 56, 202-216.

See Also

debrisThicknessEmp, nls

Examples

```
# Load the provided data set as exemplary input for the function
# The values of the data set do not represent real field
# measurements and were only created for demonstration purposes
data(debrisThickness_measured, lst_measured,
    package = "glacierSMBM")
# Individual data sets should be loaded using the
# functions read.*()

# Calculate the required fitting parameters for the function
# debrisThicknessEmp() and plot the results
output <- debrisThicknessFit(surfaceTemperature = lst_measured,
    debrisThickness = debrisThickness_measured, plotOutput = TRUE)
```

`debrisThicknessFit-method`

Method: Debris thickness fitting function

Description

A function to fit remotely sensed surface temperatures to measured debris thickness.

Details

The spatial distribution and thickness of supraglacial debris can be derived from remotely sensed surface temperatures based on an empirical relationship as shown by Mihalcea et al. (2006, 2008a, 2008b). High surface temperatures are correlated with thick debris, whereas surface temperatures closer to or below the melting point indicate a thin or absent debris layer. An exponential function with two fitting parameters (fp) was found to be most suitable to predict debris thickness from surface temperature (Minora et al., 2015). To derive debris thickness from surface temperature, an empirical non-linear model (`debrisThicknessEmp`) is applied (Groos et al., submitted). The two fitting parameters of the model are obtained by iteratively comparing measured and modelled debris thickness using varying starting values. Calculated non-linear (weighted) least-squares (`nls`) serve for the selection of the optimal fitting parameters.

Value

An object of class `'numeric'` containing the two fitting parameters.

Note

The function `extractRasterValues` may help to extract the respective surface temperature values from a `'RasterLayer'` at the locations of debris thickness measurements.

Author(s)

Alexander R. Groos (<alexander.groos@giub.unibe.ch>)

References

Groos, A.R., Mayer, C., Smiraglia, C., Diolaiuti, G., and Lambrecht A. (submitted). A first attempt to model region-wide glacier surface mass balances in the Karakoram: findings and future challenges. Geografia Fisica e Dinamica Quaternaria.

Mihalcea, C., Mayer, C., Diolaiuti, G., Lambrecht, A., Smiraglia, C., and Tartari, G. (2006). Ice ablation and meteorological conditions on the debris-covered area of Baltoro glacier, Karakoram, Pakistan. Annals of Glaciology 43, 292-300.

Mihalcea, C., Brock, B.W., Diolaiuti, G., D'Agata, C., Citterio, M., Kirkbride, M.P., Cutler, M.E.J., and Smiraglia, C. (2008a). Using ASTER satellite and ground-based surface temperature measurements to derive supraglacial debris cover and thickness patterns on Miage Glacier (Mont Blanc Massif, Italy). Cold Regions Science and Technology 52, 341-354.

Mihalcea, C., Mayer, C., Diolaiuti, G., D'Agata, C., Smiraglia, C., Lambrecht, A., Vuillermoz, E., and Tartari, G. (2008b). Spatial distribution of debris thickness and melting from remote-sensing and meteorological data, at debris-covered Baltoro glacier, Karakoram, Pakistan. Annals of Glaciology 48, 49-57.

Minora, U., Senese, A., Bocchiola, D., Soncini, A., D'Agata, C., Ambrosini, R., Mayer, C., Lambrecht, A., Vuillermoz, E., Smiraglia, C., et al. (2015). A simple model to evaluate ice melt over the ablation area of glaciers in the Central Karakoram National Park, Pakistan. Annals of Glaciology 56, 202-216.

See Also

debrisThicknessEmp, nls

Examples

```
# Load the provided data set as exemplary input for the function
# The values of the data set do not represent real field
# measurements and were only created for demonstration purposes
data(debrisThickness_measured, lst_measured,
    package = "glacierSMBM")
# Individual data sets should be loaded using the
# functions read.*()

# Calculate the required fitting parameters for the function
# debrisThicknessEmp() and plot the results
output <- debrisThicknessFit(surfaceTemperature = lst_measured,
    debrisThickness = debrisThickness_measured, plotOutput = TRUE)
```

---

debrisThicknessPhy *Function: Physical debris thickness model*

Description

An energy balance model to derive supraglacial debris thickness from surface temperature.

Usage

```
debrisThicknessPhy(surfaceTemperature, airT, netRad, airP,
    tUnit = "K", measurementHeight = 2, windSpeed = 2,
    disWindSpeed = stack(), surfaceRoughnessLength = 0.016,
    disSurfaceRoughnessLength = stack(),
    thermalConductivity = 0.96, disThermalConductivity = stack(),
    gRatio = 2.7, decimalPlaces = 4, writeOutput = FALSE,
    outputName = "debrisThickness", tmpCreate = FALSE,
    tmpDir = "", outDir = "" )
```

Arguments

`surfaceTemperature`
: An object of class `'RasterLayer'`. Distributed glacier surface temperature (Kelvin or degree Celsius).

`airT`
: An object of class `'RasterStack'`. Distributed air temperature (Kelvin or degree Celsius).

`netRad`
: An object of class `'RasterLayer'`. Distributed net radiation (W m-2).

`airP`
: An object of class `'RasterLayer'`. Distributed air pressure (Pa).

`tUnit`
: An object of class `'character'`. Unit ("K" = Kelvin, "C" = degree Celsius) of air temperature (default = "K").

`measurementHeight`
: An object of class `'numeric'`. Height (m) of meteorological measurements (default = 2).

`windSpeed`
: An object of class `'numeric'`. Wind speed (m s-1) at measurement height (default = 2).

`disWindSpeed`
: An object of class `'RasterStack'`. Distributed wind speed (m s-1) at measurement height. Stationary or for every time step.

`surfaceRoughnessLength`
: An object of class `'numeric'`. Surface roughness length (m) of the debris layer (default = 0.016).

`disSurfaceRoughnessLength`
: An object of class `'RasterStack'`. Distributed surface roughness length (m) of the debris layer.

`thermalConductivity`
: An object of class `'numeric'`. Effective thermal conductivity (W m-1 K-1) of the debris layer (default = 0.96).

`disThermalConductivity`
: An object of class `'RasterStack'`. Distributed effective thermal conductivity (W m-1 K-1) of the debris layer.

`gRatio`
: An object of class `'numeric'`. Ratio that represents the temperature profile within the debris layer (default = 2.7).

`decimalPlaces`
: An object of class `'numeric'`. Number of decimal places (default = 4).

`writeOutput`
: An object of class `'logical'`. Determines whether the ouput shall be exported as `RasterLayer` (TRUE) or not (FALSE, default).

`outputName`
: An object of class `'character'`. File name for the output `RasterLayer` (default = "debrisThickness").

`tmpCreate`
: An object of class `'logical'`. Determines whether a temporary directory should be used (TRUE) or not (FALSE, default). Recommendend if large datasets are processed.

`tmpDir`
: An object of class `'character'`. Directory where processing files can be temporarily stored if `'tmpCreate'` = TRUE.

`outDir`
: An object of class `'character'`. Directory for the output files if `'writeOutput'` = TRUE.

Details

Ice melt rates below a porous debris layer are not only controlled by meteorological conditions, but also the thickness and thermal properties of the layer (e.g. Evatt et al., 2015). For the modelling of sub-debris ice melt, information on the thickness and spatial variability of the debris layer are necessary. In-situ measurements of debris thickness are labour-intensive and represent only a small fraction of the glacier surface. To compensate for that, different energy balance models, which derive debris thickness from remotely sensed glacier surface temperatures, have been developed (e.g. Foster et al., 2012; Rounce & McKinney, 2014; Schauwecker et al. 2015). The steady-state surface energy balance model of Rounce & McKinney (2014, 1-7) is applied to calculate the supraglacial debris thickness distribution from land surface and meteorological data. An approximation factor (`gRatio`) is used to take the non-linear temperature variation in the debris layer into account. For more details of the approach please refer to Groos et al. (submitted, Equations 5-7).

Value

An object of class `'RasterLayer'` returning the calculated debris thickness distribution (m).

Note

The following input variables are the requested minimum to run the model:

- `'surfaceTemperature'`
- `'airT'`
- `'netRad'`
- `'airP'`

A default value (constant in space and time) is given for each additional argument like `'windSpeed'` or `'thermalConductivity'`. If desired, the default parameters can be modified. Furthermore, there is the option to pass distributed values (stationary or for every time step) instead of general values using the related 'dis*'-arguments like `'disWindSpeed'` or `'disThermalConductivity'`. In this case, the general parameter is ignored.

File format of written ouput: GeoTIFF.

Author(s)

Alexander R. Groos (<alexander.groos@giub.unibe.ch>)

References

Evatt, G.W., Abrahams, D., Heil, M., Mayer, C., Kingslake, J., Mitchell, S.L., Fowler, A.C., and Clark, C.D. (2015). Glacial melt under a porous debris layer. Journal of Glaciology 61, 825-836.

Foster, L.A., Brock, B.W., Cutler, M.E.J., and Diotri, F. (2012). A physically based method for estimating supraglacial debris thickness from thermal band remote-sensing data. Journal of Glaciology 58, 677-691.

Groos, A.R., Mayer, C., Smiraglia, C., Diolaiuti, G., and Lambrecht A. (submitted). A first attempt to model region-wide glacier surface mass balances in the Karakoram: findings and future challenges. Geografia Fisica e Dinamica Quaternaria.

Rounce, D.R., and McKinney, D.C. (2014). Debris thickness of glaciers in the Everest area (Nepal Himalaya) derived from satellite imagery using a nonlinear energy balance model. The Cryosphere 8, 1317-1329.

Schauwecker, S., Rohrer, M., Huggel, C., Kulkarni, A., Ramanathan, A., Salzmann, N., Stoffel, M., and Brock, B. (2015). Remotely sensed debris thickness mapping of Bara Shigri Glacier, Indian Himalaya. Journal of Glaciology 61, 675-688.

See Also

debrisThicknessEmp, debrisCoveredIceMelt

Examples

```
# Load the provided RasterLayer objects as exemplary input for
# the function
data(lst_30m_hourly, airTemperature_30m_hourly, netRad_30m_hourly,
    airPressure_30m_hourly, package = "glacierSMBM")
# Individual RasterLayer objects should be loaded or
# created using the function raster()

# Derive debris thickness from land surface temperature using
# standard settings
output <- debrisThicknessPhy(surfaceTemperature = lst_30m_hourly,
    airT = airTemperature_30m_hourly, netRad = netRad_30m_hourly,
    airP = airPressure_30m_hourly)

# Plot output
plot(output, main = "debris thickness",
    legend.args=list(text='Debris thickness (m)',
    side=3, line=1.5))

# Derive debris thickness from land surface temperature using
# modified settings (e.g. change numeric values for effective
# thermal conductivity and wind speed)

output <- debrisThicknessPhy(surfaceTemperature = lst_30m_hourly,
    airT = airTemperature_30m_hourly, netRad = netRad_30m_hourly,
    airP = airPressure_30m_hourly, windSpeed = 6,
    thermalConductivity = 0.86)

# Plot output
plot(output, main = "debris thickness",
    legend.args=list(text='Debris thickness (m)',
    side=3, line=1.5))
```

---

debrisThicknessPhy-method

Method: Physical debris thickness model

Description

An energy balance model to derive supraglacial debris thickness from surface temperature.

Details

Ice melt rates below a porous debris layer are not only controlled by meteorological conditions, but also the thickness and thermal properties of the layer (e.g. Evatt et al., 2015). For the modelling of sub-debris ice melt, information on the thickness and spatial variability of the debris layer are necessary. In-situ measurements of debris thickness are labour-intensive and represent only a small fraction of the glacier surface. To compensate for that, different energy balance models, which derive debris thickness from remotely sensed glacier surface temperatures, have been developed (e.g. Foster et al., 2012; Rounce & McKinney, 2014; Schauwecker et al. 2015). The steady-state surface energy balance model of Rounce & McKinney (2014, 1-7) is applied to calculate the supraglacial debris thickness distribution from land surface and meteorological data. An approximation factor (`gRatio`) is used to take the non-linear temperature variation in the debris layer into account. For more details of the approach please refer to Groos et al. (submitted, Equations 5-7).

Value

An object of class `'RasterLayer'` returning the calculated debris thickness distribution (m).

Note

The following input variables are the requested minimum to run the model:

- `'surfaceTemperature'`
- `'airT'`
- `'netRad'`
- `'airP'`

A default value (constant in space and time) is given for each additional argument like `'windSpeed'` or `'thermalConductivity'`. If desired, the default parameters can be modified. Furthermore, there is the option to pass distributed values (stationary or for every time step) instead of general values using the related 'dis*'-arguments like `'disWindSpeed'` or `'disThermalConductivity'`. In this case, the general parameter is ignored.

File format of written ouput: GeoTIFF.

Author(s)

Alexander R. Groos (<alexander.groos@giub.unibe.ch>)

References

Evatt, G.W., Abrahams, D., Heil, M., Mayer, C., Kingslake, J., Mitchell, S.L., Fowler, A.C., and Clark, C.D. (2015). Glacial melt under a porous debris layer. Journal of Glaciology 61, 825-836.

Foster, L.A., Brock, B.W., Cutler, M.E.J., and Diotri, F. (2012). A physically based method for estimating supraglacial debris thickness from thermal band remote-sensing data. Journal of Glaciology 58, 677-691.

Groos, A.R., Mayer, C., Smiraglia, C., Diolaiuti, G., and Lambrecht A. (submitted). A first attempt to model region-wide glacier surface mass balances in the Karakoram: findings and future challenges. Geografia Fisica e Dinamica Quaternaria.

Rounce, D.R., and McKinney, D.C. (2014). Debris thickness of glaciers in the Everest area (Nepal Himalaya) derived from satellite imagery using a nonlinear energy balance model. The Cryosphere 8, 1317-1329.

Schauwecker, S., Rohrer, M., Huggel, C., Kulkarni, A., Ramanathan, A., Salzmann, N., Stoffel, M., and Brock, B. (2015). Remotely sensed debris thickness mapping of Bara Shigri Glacier, Indian Himalaya. Journal of Glaciology 61, 675-688.

See Also

debrisThicknessEmp, debrisCoveredIceMelt

Examples

```
# Load the provided RasterLayer objects as exemplary input for
# the function
data(lst_30m_hourly, airTemperature_30m_hourly, netRad_30m_hourly,
    airPressure_30m_hourly, package = "glacierSMBM")
# Individual RasterLayer objects should be loaded or
# created using the function raster()

# Derive debris thickness from land surface temperature using
# standard settings
output <- debrisThicknessPhy(surfaceTemperature = lst_30m_hourly,
    airT = airTemperature_30m_hourly, netRad = netRad_30m_hourly,
    airP = airPressure_30m_hourly)

# Plot output
plot(output, main = "debris thickness",
    legend.args=list(text='Debris thickness (m)',
    side=3, line=1.5))

# Derive debris thickness from land surface temperature using
# modified settings (e.g. change numeric values for effective
# thermal conductivity and wind speed)

output <- debrisThicknessPhy(surfaceTemperature = lst_30m_hourly,
    airT = airTemperature_30m_hourly, netRad = netRad_30m_hourly,
    airP = airPressure_30m_hourly, windSpeed = 6,
    thermalConductivity = 0.86)

# Plot output
plot(output, main = "debris thickness",
    legend.args=list(text='Debris thickness (m)',
    side=3, line=1.5))
```

`debrisThickness_30m` *Data: Debris thickness (30m)*

Description

Distributed debris thickness at the Liligo Glacier (Karakoram, Pakistan)

Usage

```
data(debrisThickness_30m)
```

Format

An object of class `'RasterLayer'`.

Details

Dataset: Landsat 5

Date: 2011-08-10

Pixel resolution: 30 m

Unit: m

Projection: UTM 43 N

Note: The debris thickness distribution was derived from satellite-based land surface temperatures and meteorological data using the function `debrisThicknessPhy` (for more information see Groos et al., submitted).

Source

USGS EarthExplorer

References

Groos, A.R., Mayer, C., Smiraglia, C., Diolaiuti, G., and Lambrecht A. (submitted). A first attempt to model region-wide glacier surface mass balances in the Karakoram: findings and future challenges. Geografia Fisica e Dinamica Quaternaria.

Examples

```
data(debrisThickness_30m)
plot(debrisThickness_30m)
```

debrisThickness_measured *Data: Debris thickness (in-situ)*

Description

Debris thickness "measurements" at the Liligo Glacier (Karakoram, Pakistan)

Usage

```
data(debrisThickness_measured)
```

Format

An object of class `'numeric'`.

Details

Unit: m

Note: The values of the dataset do not represent real field measurements.

Examples

```
data(debrisThickness_measured)
print(debrisThickness_measured)
```

dem_30m *Data: DEM (30m)*

Description

Digital elevation model of the Liligo Glacier (Karakoram, Pakistan)

Usage

```
data(dem_30m)
```

Format

An object of class `'RasterLayer'`.

Details

Dataset: The High Asia Refined Analysis (TU Berlin, Chair of Climatology)

Pixel resolution: 10 km

Unit: m

Projection: UTM 43 N

Note: The original dataset was resampled to a spatial resolution of 30 m using the function `resample`.

Source

High Asia Refined Analysis

References

Groos, A.R., Mayer, C., Smiraglia, C., Diolaiuti, G., and Lambrecht A. (submitted). A first attempt to model region-wide glacier surface mass balances in the Karakoram: findings and future challenges. Geografia Fisica e Dinamica Quaternaria.

Examples

```
data(dem_30m)
plot(dem_30m)
```

---

extractRasterValues *Function: Extract raster values*

Description

A function to extract cell values from a RasterLayer.

Usage

```
extractRasterValues(rasterLayer, selectedCoordinates)
```

Arguments

`rasterLayer`	An object of class `'RasterLayer'` from which the requested cell values are extracted.
`selectedCoordinates`	An object of class `'matrix'`. Coordinates (x = first column, y = second column) for each location where point values from the `RasterLayer` are requested.

Note

The `projection` of the given coordinates and `RasterLayer` must be the same.

Author(s)

Alexander R. Groos (<alexander.groos@giub.unibe.ch>)

See Also

cellFrom, extract

Examples

```
# Load the provided data set and RasterLayer as exemplary
# input for the function
data(selectedCoordinates, lst_30m_hourly,
    package = "glacierSMBM")
# Individual data sets should be loaded using the
# functions read.*() or raster(), respectively

# Extract the requested cell values of a RasterLayer based
# on the given coordinates
output <- extractRasterValues(rasterLayer = lst_30m_hourly,
    selectedCoordinates = selectedCoordinates)
```

---

extractRasterValues-method

Method: Extract raster values

Description

A function to extract cell values from a RasterLayer.

Note

The projection of the given coordinates and RasterLayer must be the same.

Author(s)

Alexander R. Groos (<alexander.groos@giub.unibe.ch>)

See Also

cellFrom, extract

Examples

```
# Load the provided data set and RasterLayer as exemplary
# input for the function
data(selectedCoordinates, lst_30m_hourly,
    package = "glacierSMBM")
# Individual data sets should be loaded using the
# functions read.*() or raster(), respectively
```

```
# Extract the requested cell values of a RasterLayer based
# on the given coordinates
output <- extractRasterValues(rasterLayer = lst_30m_hourly,
    selectedCoordinates = selectedCoordinates)
```

`firnMask_30m` *Data: Firn mask (30m)*

Description

Firn and snow cover at the Liligo Glacier (Karakoram, Pakistan)

Usage

```
data(firnMask_30m)
```

Format

An object of class `'RasterLayer'`.

Details

Dataset: Landsat 5

Date: 2011-08-10

Pixel resolution: 30 m

1 = firn, 0 = no firn

Projection: UTM 43 N

Note: The firn and snow cover distribution was derived from a Landsat 5 image (for more information see Groos et al., submitted).

Source

USGS EarthExplorer

References

Groos, A.R., Mayer, C., Smiraglia, C., Diolaiuti, G., and Lambrecht A. (submitted). A first attempt to model region-wide glacier surface mass balances in the Karakoram: findings and future challenges. Geografia Fisica e Dinamica Quaternaria.

Examples

```
data(firnMask_30m)
plot(firnMask_30m)
```

glacialMelt *Function: Ablation model*

Description

A model to calculate melting of supraglacial snow and firn, bare glacier ice and debris covered glacier ice.

Usage

```
glacialMelt(airT, airDensity, netRad, glacierMask, iceMask,
    snowMask, debrisMask, debrisThickness, inRadSW = stack(),
    inRadLW = stack(), disTuningFacAirT = disTuningFacAirT,
    disIceTMF = stack(), disSnowTMF = stack(),
    disIceRMF = stack(), disSnowRMF = stack(),
    disRelativeHumidity = stack(), disWindSpeed = stack(),
    disDebrisAlbedo = stack(), disThermalConductivity = stack(),
    disThermalEmissivity = stack(),
    disSurfaceRoughnessHeight = stack(),
    disFrictionVelocity = stack(),
    disVolumeFractionDebrisInIce = stack(),
    disDebrisAirRatio = stack(), disDragCoefficient = stack(),
    disIceDensity = stack(), outType = "mean",
    writeOutput = c(0, 0, 0, 0), outputName = "glacialMelt",
    tmpCreate = FALSE, tmpDir = "", outDir = "", ...)
```

Arguments

airT	An object of class 'RasterStack'. Distributed air temperature (Kelvin or degree Celsius). For every time step.
airDensity	An object of class 'RasterStack'. Distributed air density (kg m-3). Stationary or for every time step.
netRad	An object of class 'RasterStack'. Distributed net radiation (W m-2). For every time step.
glacierMask	An object of class 'RasterStack'. Glacier area (1 = glacier, 0 = no glacier). Stationary or for every time step.
iceMask	An object of class RasterStack. Area of bare glacier ice (1 = bare ice, 0 = no bare ice). Stationary or for every time step.
snowMask	An object of class RasterStack. Area of supraglacial snow or firn (1 = snow or firn, 0 = no snow or firn). Stationary or for every time step.
debrisMask	An object of class 'RasterStack'. Area of debris covered glacier ice (1 = debris, 0 = no debris). Stationary or for every time step.
debrisThickness	An object of class 'RasterStack'. Distributed supraglacial debris thickness (m). Stationary or for every time step.

inRadSW An object of class 'RasterStack'. Distributed incoming shortwave radiation (W m-2). Optional in the function 'debrisCoveredIceMelt' instead of 'netRad'. For every time step.

inRadLW An object of class 'RasterStack'. Distributed incoming longwave radiation (W m-2). Optional in the function 'debrisCoveredIceMelt' instead of 'netRad'. For every time step.

disTuningFacAirT
An object of class 'RasterStack'. Distributed air temperature tuning factor ('tuningFacAirT'). Stationary or for every time step.

disIceTMF An object of class RasterStack. Distributed temperature melting factor (m K-1 timestep-1) of ice. Stationary or for every time step.

disSnowTMF An object of class RasterStack. Distributed temperature melting factor (m K-1 timestep-1) of snow. Stationary or for every time step.

disIceRMF An object of class RasterStack. Distributed radiative temperature melting factor (m K-1 timestep-1) of ice. Stationary or for every time step.

disSnowRMF An object of class RasterStack. Distributed radiative temperature melting factor (m K-1 timestep-1) of snow. Stationary or for every time step.

disRelativeHumidity
An object of class 'RasterStack'. Distributed relative humidity (0-1) at measurement height. Stationary or for every time step.

disWindSpeed An object of class 'RasterStack'. Distributed wind speed (m s-1) at measurement height. Stationary or for every time step.

disDebrisAlbedo
An object of class 'RasterStack'. Distributed albedo (0-1) of the debris. Stationary or for every time step.

disThermalConductivity
An object of class 'RasterStack'. Distributed thermal conductivity (W m-1 K-1) of the debris layer. Stationary or for every time step.

disThermalEmissivity
An object of class 'RasterStack'. Distributed thermal emissivity (0-1) of the debris layer. Stationary or for every time step.

disSurfaceRoughnessHeight
An object of class 'RasterStack'. Distributed surface roughness height (m) of the debris layer. Stationary or for every time step.

disFrictionVelocity
An object of class 'RasterStack'. Distributed friction velocity (m s-1) of the debris layer. Stationary or for every time step.

disVolumeFractionDebrisInIce
An object of class 'RasterStack'. Distributed volume fraction (0-1) of debris in the ice body. Stationary or for every time step.

disDebrisAirRatio
An object of class 'RasterStack'. Distributed ratio of the debris surface area to the volume of air (m-1) in the debris layer. Stationary or for every time step.

disDragCoefficient
An object of class 'RasterStack'. Distributed drag coefficient (m-1) of the debris layer. Stationary or for every time step.

disIceDensity	An object of class 'RasterStack'. Distributed density (kg m-3) of ice. Stationary or for every time step.
outType	An object of class 'character'. Type of output to be returned by the function: "mean" (default) snow melt or "sum".
writeOutput	An object of class 'logical'. Determines which of the four outputs (default = none) shall be exported as 'RasterLayer' (1) and which not (0). For more information see section values.
outputName	An object of class 'character'. File name for the output 'RasterLayer(s)' (default = "glacialMelt").
tmpCreate	An object of class 'logical'. Determines whether a temporary directory should be used (TRUE) or not (FALSE, default). Recommendend if large datasets are processed.
tmpDir	An object of class 'character'. Directory where processing files can be temporarily stored if 'tmpCreate' = TRUE.
outDir	An object of class 'character'. Directory for the output files if 'writeOutput' = 1.
...	Further arguments passed to the embedded functions iceMelt, snowMelt and debrisCoveredIceMelt.

Details

Glacier ablation comprises melting of bare and debris-covered ice as well as melting of supraglacial snow and firn. Depending on the glacier surface classification as defined in 'iceMask', 'snowMask' and 'debrisMask', the melt rate for each pixel is quantified using the functions iceMelt, snowMelt and debrisCoveredIceMelt. For more information please refer to the applied melting functions, the examples below or the original publication (Groos et al., submitted, Equations 10-16).

Value

An object of class 'RasterStack' consisting of four individual 'RasterLayers':

1. Melt rate of supraglacial snow and firn as calculated by snowMelt
2. Melt rate of bare ice as calculated by iceMelt
3. Melt rate of debris covered ice as calculated by debrisCoveredIceMelt
4. Sum of 1-3

To access the individual 'RasterLayers', use subset.

If more than one 'RasterLayer' is stored in the input 'RasterStacks', the output 'RasterStack' returns the "mean" or "sum" (depending on 'outType') of snow melt, ice melt and debris covered ice melt.

Note

The following input variables are the requested minimum to run the model:

- 'airT' (for every time step)

- 'netRad' (for every time step)
- 'airDensity' (stationary or for every time step)
- 'glacierMask' (stationary or for every time step)
- 'iceMask' (stationary or for every time step)
- 'snowMask' (stationary or for every time step)
- 'debrisMask' (stationary or for every time step)
- 'debrisThickness' (stationary or for every time step)

If 'inRadSW' and 'inRadLW' are provided instead of 'netRad', the energy-balance at the atmosphere-debris interface is calculated taking the 'debrisAlbedo' and 'thermalEmissivity' of the debris layer into account.

A default value (constant in space and time) is given for each additional argument like 'windSpeed', 'relativeHumidity' and 'thermalConductivity'. If desired, the default parameters can be modified. Furthermore, there is the option to pass distributed values (stationary or for every time step) instead of general values using the related 'dis*'-arguments like 'disWindSpeed', 'disRelativeHumidity' and 'disThermalConductivity'. In this case, the general parameter is ignored.

File format of written ouput: GeoTIFF.

Author(s)

Alexander R. Groos (<alexander.groos@giub.unibe.ch>)

References

Collier, E., Moelg, T., Maussion, F., Scherer, D., Mayer, C., and Bush, A.B.G. (2013). High-resolution interactive modelling of the mountain glacier-atmosphere interface: an application over the Karakoram. The Cryosphere Discussions 7, 103-144.

Evatt, G.W., Abrahams, D., Heil, M., Mayer, C., Kingslake, J., Mitchell, S.L., Fowler, A.C., and Clark, C.D. (2015). Glacial melt under a porous debris layer. Journal of Glaciology 61, 825-836.

Groos, A.R., Mayer, C., Smiraglia, C., Diolaiuti, G., and Lambrecht A. (submitted). A first attempt to model region-wide glacier surface mass balances in the Karakoram: findings and future challenges. Geografia Fisica e Dinamica Quaternaria.

Hock, R. (2003). Temperature index melt modelling in mountain areas. Journal of Hydrology 282, 104-115.

Hock, R. (2005). Glacier melt: a review of processes and their modelling. Progress in Physical Geography 29, 362-391.

See Also

snowMelt, iceMelt, debrisCoveredIceMelt

Examples

```
# Load the provided RasterLayer objects as exemplary
# input for the function
data(airTemperature_30m_daily, netRad_30m_daily,
    airDensity_30m_daily, glacierMask_30m, iceMask_30m,
    firnMask_30m, debrisMask_30m, debrisThickness_30m,
    package = "glacierSMBM")
# Individual RasterLayer objects should be loaded or
# created using the function raster()

# Include RasterLayer in RasterStack
AirTemperature_30m_daily <- stack(airTemperature_30m_daily * 1.01)
NetRad_30m_daily <- stack(netRad_30m_daily)
AirDensity_30m_daily <- stack(airDensity_30m_daily)
GlacierMask_30m <- stack(glacierMask_30m)
IceMask_30m <- stack(iceMask_30m)
FirnMask_30m <- stack(firnMask_30m)
DebrisThickness_30m <- stack(debrisThickness_30m)
DebrisMask_30m <- stack(debrisMask_30m)

## Not run:
# Calculate ablation using standard settings
output <- glacialMelt(airT = AirTemperature_30m_daily,
    netRad = NetRad_30m_daily, airDensity = AirDensity_30m_daily,
    glacierMask = GlacierMask_30m, iceMask = IceMask_30m,
    snowMask = FirnMask_30m, debrisMask = DebrisMask_30m,
    debrisThickness = DebrisThickness_30m)

# Plot output
plot(output, main = c("snow melt", "bare ice melt",
    "debris covered ice melt","total ablation"),
    legend.args=list(text='Melt (m d-1)', side=3, line=1.5))

## End(Not run)

# Calculate ablation using modified setting (e.g. input
# temperature in celsius instead of kelvin and
# increased thermal conductivity)

# Therefore exemplarily convert temperature from
# kelvin to degree celsius
airTcelsius <- AirTemperature_30m_daily - 273.15

# Include RasterLayer in RasterStack
airTcelsius <- stack(airTcelsius)

## Not run:
output <- glacialMelt(airT = airTcelsius, netRad =
    NetRad_30m_daily, airDensity <- AirDensity_30m_daily,
    glacierMask = GlacierMask_30m, iceMask = IceMask_30m,
    snowMask = FirnMask_30m, debrisMask = DebrisMask_30m,
    debrisThickness = DebrisThickness_30m, tUnit = "C",
```

```
    thermalConductivity = 1.5)

# Plot output
plot(output, main = c("snow melt", "bare ice melt",
    "debris covered ice melt","total ablation"),
    legend.args=list(text='Melt (m d-1)', side=3, line=1.5))

## End(Not run)
```

`glacialMelt-method` *Method: Ablation model*

Description

A model to calculate melting of supraglacial snow and firn, bare glacier ice and debris covered glacier ice.

Details

Glacier ablation comprises melting of bare and debris-covered ice as well as melting of supraglacial snow and firn. Depending on the glacier surface classification as defined in `'iceMask'`, `'snowMask'` and `'debrisMask'`, the melt rate for each pixel is quantified using the functions `iceMelt`, `snowMelt` and `debrisCoveredIceMelt`. For more information please refer to the applied melting functions, the examples below or the original publication (Groos et al., submitted, Equations 10-16).

Value

An object of class `'RasterStack'` consisting of four individual `'RasterLayers'`:

1. Melt rate of supraglacial snow and firn as calculated by `snowMelt`
2. Melt rate of bare ice as calculated by `iceMelt`
3. Melt rate of debris covered ice as calculated by `debrisCoveredIceMelt`
4. Sum of 1-3

To access the individual `'RasterLayers'`, use `subset`.

If more than one `'RasterLayer'` is stored in the input `'RasterStacks'`, the output `'RasterStack'` returns the "mean" or "sum" (depending on `'outType'`) of snow melt, ice melt and debris covered ice melt.

Note

The following input variables are the requested minimum to run the model:

- `'airT'` (for every time step)
- `'netRad'` (for every time step)
- `'airDensity'` (stationary or for every time step)

- 'glacierMask' (stationary or for every time step)
- 'iceMask' (stationary or for every time step)
- 'snowMask' (stationary or for every time step)
- 'debrisMask' (stationary or for every time step)
- 'debrisThickness' (stationary or for every time step)

If 'inRadSW' and 'inRadLW' are provided instead of 'netRad', the energy-balance at the atmosphere-debris interface is calculated taking the 'debrisAlbedo' and 'thermalEmissivity' of the debris layer into account.

A default value (constant in space and time) is given for each additional argument like 'windSpeed', 'relativeHumidity' and 'thermalConductivity'. If desired, the default parameters can be modified. Furthermore, there is the option to pass distributed values (stationary or for every time step) instead of general values using the related 'dis*'-arguments like 'disWindSpeed', 'disRelativeHumidity' and 'disThermalConductivity'. In this case, the general parameter is ignored.

File format of written ouput: GeoTIFF.

Author(s)

Alexander R. Groos (<alexander.groos@giub.unibe.ch>)

References

Collier, E., Moelg, T., Maussion, F., Scherer, D., Mayer, C., and Bush, A.B.G. (2013). High-resolution interactive modelling of the mountain glacier-atmosphere interface: an application over the Karakoram. The Cryosphere Discussions 7, 103-144.

Evatt, G.W., Abrahams, D., Heil, M., Mayer, C., Kingslake, J., Mitchell, S.L., Fowler, A.C., and Clark, C.D. (2015). Glacial melt under a porous debris layer. Journal of Glaciology 61, 825-836.

Groos, A.R., Mayer, C., Smiraglia, C., Diolaiuti, G., and Lambrecht A. (submitted). A first attempt to model region-wide glacier surface mass balances in the Karakoram: findings and future challenges. Geografia Fisica e Dinamica Quaternaria.

Hock, R. (2003). Temperature index melt modelling in mountain areas. Journal of Hydrology 282, 104-115.

Hock, R. (2005). Glacier melt: a review of processes and their modelling. Progress in Physical Geography 29, 362-391.

See Also

snowMelt, iceMelt, debrisCoveredIceMelt

Examples

```
# Load the provided RasterLayer objects as exemplary
# input for the function
data(airTemperature_30m_daily, netRad_30m_daily,
    airDensity_30m_daily, glacierMask_30m, iceMask_30m,
    firnMask_30m, debrisMask_30m, debrisThickness_30m,
    package = "glacierSMBM")
```

```
# Individual RasterLayer objects should be loaded or
# created using the function raster()

# Include RasterLayer in RasterStack
AirTemperature_30m_daily <- stack(airTemperature_30m_daily * 1.01)
NetRad_30m_daily <- stack(netRad_30m_daily)
AirDensity_30m_daily <- stack(airDensity_30m_daily)
GlacierMask_30m <- stack(glacierMask_30m)
IceMask_30m <- stack(iceMask_30m)
FirnMask_30m <- stack(firnMask_30m)
DebrisThickness_30m <- stack(debrisThickness_30m)
DebrisMask_30m <- stack(debrisMask_30m)

## Not run:
# Calculate ablation using standard settings
output <- glacialMelt(airT = AirTemperature_30m_daily,
    netRad = NetRad_30m_daily, airDensity = AirDensity_30m_daily,
    glacierMask = GlacierMask_30m, iceMask = IceMask_30m,
    snowMask = FirnMask_30m, debrisMask = DebrisMask_30m,
    debrisThickness = DebrisThickness_30m)

# Plot output
plot(output, main = c("snow melt", "bare ice melt",
    "debris covered ice melt","total ablation"),
    legend.args=list(text='Melt (m d-1)', side=3, line=1.5))

## End(Not run)

# Calculate ablation using modified setting (e.g. input
# temperature in celsius instead of kelvin and
# increased thermal conductivity)

# Therefore exemplarily convert temperature from
# kelvin to degree celsius
airTcelsius <- AirTemperature_30m_daily - 273.15

# Include RasterLayer in RasterStack
airTcelsius <- stack(airTcelsius)

## Not run:
output <- glacialMelt(airT = airTcelsius, netRad =
    NetRad_30m_daily, airDensity <- AirDensity_30m_daily,
    glacierMask = GlacierMask_30m, iceMask = IceMask_30m,
    snowMask = FirnMask_30m, debrisMask = DebrisMask_30m,
    debrisThickness = DebrisThickness_30m, tUnit = "C",
    thermalConductivity = 1.5)

# Plot output
plot(output, main = c("snow melt", "bare ice melt",
    "debris covered ice melt","total ablation"),
    legend.args=list(text='Melt (m d-1)', side=3, line=1.5))

## End(Not run)
```

`glacierMask_30m` *Data: Glacier mask (30m)*

Description

Glacier outlines of the Liligo Glacier (Karakoram, Pakistan)

Usage

```
data(glacierMask_30m)
```

Format

An object of class `'RasterLayer'`.

Details

Dataset: Landsat 5

Date: 2011-08-10

Pixel resolution: 30 m

1 = glacier, 0 = no glacier

Projection: UTM 43 N

Note: The glacier outlines were derived from a set of high-resolution satellite images (for more information see Minora et al., 2013; Groos et al., submitted).

Source

USGS EarthExplorer

References

Minora, U., Bocchiola, D., D'Agata, C., Maragno, D., Mayer, C., Lambrecht, A., Mosconi, B., Vuillermoz, E., Senese, A., Compostella, C., et al. (2013). 2001-2010 glacier changes in the Central Karakoram National Park: a contribution to evaluate the magnitude and rate of the "Karakoram anomaly." The Cryosphere Discussions 7, 2891-2941.

Groos, A.R., Mayer, C., Smiraglia, C., Diolaiuti, G., and Lambrecht A. (submitted). A first attempt to model region-wide glacier surface mass balances in the Karakoram: findings and future challenges. Geografia Fisica e Dinamica Quaternaria.

Examples

```
data(glacierMask_30m)
plot(glacierMask_30m)
```

`glacierSMBM` *Function: Glacier surface mass balance model*

Description

A model consisting of physical and statistical functions to calculate the surface mass balance of individual or multiple debris-free and debris-covered glaciers.

Usage

```
glacierSMBM(inputGlacierSMBM, ...)
```

Arguments

`inputGlacierSMBM`	An object of class `'inputGlacierSMBM'`. Required parameters and variables to calculate glacier surface mass balances. Fore more information see `inputGlacierSMBM-class`.
`...`	Optional arguments should be also addressed in `'inputGlacierSMBM'`.

Details

Glacier surface mass balances are quantified by calculating the difference between the glacier mass gain ascribed to deposited snow and the mass loss associated with melting of bare an debris-covered ice as well as supraglacial snow and firn. The initial glacier outlines and surface conditions can be defined in `'glacierMask'`, `'iceMask'`, `'firnMask'`, `'snowHeight'`, `'debrisMask'` and `'debrisThickness'` (for more information see `inputGlacierSMBM-class`). Accumulation and ablation is iteratively modelled for every discrete time step (e.g. month, day or hour). The function `snowFall` is applied to quantify accumulation for every pixel and time step (note: mass input from redistributed snow has not yet been implemented). Ablation does only occur in the model at air temperatures above freezing since the complex process of percolation, retention and refreezing of meltwater is (so far) neglected. The extent and height of supraglacial snow changes with time due to the interplay of snow accumulation (`snowFall`) and snow melt (`snowMelt`). Ablation of debris-free (`iceMelt`) or debris-covered glacier ice (`debrisCoveredIceMelt`) sets in as soon as the overlying snow pack is completely melted and takes place as long as a pixel is snow-free. For more information please refer to the applied functions, the examples below or the original publication (Groos et al., submitted).

Value

The potential model output consists of sixteen individual `'RasterLayers'` and a summarising table:

1. Accumulation rate (for every time step).
2. Accumulation sum (for the modelling period).
3. Glacial ablation rate (for every time step).
4. Glacial ablation sum (for the modelling period).

5. Ice melt rate (for every time step).
6. Ice melt sum (for the modelling period).
7. Debris ice melt rate (for every time step).
8. Debris ice melt sum (for the modelling period).
9. Firn melt rate (for every time step).
10. Firn melt sum (for the modelling period).
11. Snow melt rate (for every time step).
12. Snow melt sum (for the modelling period).
13. Snow cover (for every time step).
14. Snow height (for the modelling period).
15. Glacier surface mass balance (for every time step).
16. Glacier surface mass balance (for the modelling period).
17. Output table consisting of the spatial average of the fourteen variables above for every time step.

Note

The following input variables in `inputGlacierSMBM-class` are the requested minimum to run the model:

- `'date'` (date and time of each `RasterLayer` in the following `RasterStacks`.)
- `'airT'` (for every time step)
- `'netRad'` (for every time step)
- `'snowfall'` or `'precip'` (for every time step)
- `'airDensity'` (stationary or for every time step)
- `'glacierMask'` (stationary or for every time step)
- `'iceMask'` (stationary or for every time step)
- `'firnMask'` (stationary or for every time step)
- `'debrisMask'` (stationary or for every time step)
- `'debrisThickness'` (stationary or for every time step)

If the investigated glacier or glacier area is debris-free or snow-free, pass a `'RasterLayer'` with the extent of `'glacierMask'` and all values = 0 (zero means no snow or no debris, respectively) to the respective `'RasterStack'` (e.g. `'debrisMask'` or `'firnMask'`).

If `'inRadSW'` and `'inRadLW'` are provided instead of `'netRad'`, the energy-balance at the atmosphere-debris interface is calculated taking the `'debrisAlbedo'` and `'thermalEmissivity'` of the debris layer into account.

A default value (constant in space and time) is given for each additional argument in `inputGlacierSMBM-class` like `'windSpeed'`, `'relativeHumidity'` and `'thermalConductivity'`. If desired, the default parameters can be modified. Furthermore, there is the option to pass distributed values (stationary or for every time step) instead of general values using the related 'dis*'-arguments like `'disWindSpeed'`, `'disRelativeHumidity'` and `'disThermalConductivity'`. In this case, the general parameter is ignored.

File format of written ouput: GeoTIFF: GeoTIFF (1-16) and text file (17).

Author(s)

Alexander R. Groos (<alexander.groos@giub.unibe.ch>)

References

Groos, A.R., Mayer, C., Smiraglia, C., Diolaiuti, G., and Lambrecht A. (submitted). A first attempt to model region-wide glacier surface mass balances in the Karakoram: findings and future challenges. Geografia Fisica e Dinamica Quaternaria.

See Also

glacialMelt, snowMelt, iceMelt, debrisCoveredIceMelt, snowFall, inputGlacierSMBM-class

Examples

```
# Load the provided RasterLayer objects as exemplary
# input for the function
data(airTemperature_30m_daily, airDensity_30m_daily,
    netRad_30m_daily, glacierMask_30m, iceMask_30m, firnMask_30m,
    debrisMask_30m, debrisThickness_30m, precipTuningFactor_30m,
    snowFall_30m_daily, package = "glacierSMBM")
# Individual RasterLayer objects should be loaded or
# created using the function raster()

# create a three-day virtual meteorological data set
AirT <- stack(airTemperature_30m_daily,
    airTemperature_30m_daily * 0.99,
    airTemperature_30m_daily * 1.01)
NetRad <- stack(netRad_30m_daily,netRad_30m_daily * 0.99,
    netRad_30m_daily * 1.01)
Snowfall <- stack(snowFall_30m_daily, snowFall_30m_daily * 2,
    snowFall_30m_daily * 0.3)

# create a new object of class "inputGlacierSMBM" which is
# requested as input for the glacier surface mass balance model
InputGlacierSMBM <- new("inputGlacierSMBM")

# Add the required data and information to the respective
# slots of the new object (for additional setting options read
# the help section of class "inputGlacierSMBM")
# Create a numeric vector containing date and time of
# the meteorological input data
InputGlacierSMBM@date <- seq.POSIXt(ISOdate(2011,8,15),
    ISOdate(2011,8,17), "days")
InputGlacierSMBM@decimalPlaces <- 4
InputGlacierSMBM@airT <- AirT
InputGlacierSMBM@airDensity <- stack(airDensity_30m_daily)
InputGlacierSMBM@netRad <- NetRad
InputGlacierSMBM@snowfall <- Snowfall
InputGlacierSMBM@glacierMask <- stack(glacierMask_30m)
InputGlacierSMBM@iceMask <- stack(iceMask_30m)
InputGlacierSMBM@firnMask <- stack(firnMask_30m)
```

```
InputGlacierSMBM@debrisMask <- stack(debrisMask_30m)
InputGlacierSMBM@debrisThickness <- stack(debrisThickness_30m)
InputGlacierSMBM@disTuningFacPrecip <- stack(precipTuningFactor_30m)

# Calculate glacier surface mass balance using standard settings,
# but suppress to write any output
InputGlacierSMBM@writeOutput <- rep(0, 17)

## Not run:
output <- glacierSMBM(inputGlacierSMBM = InputGlacierSMBM)

# Plot output
plot(output, main = "glacier surface mass balance",
    legend.args=list(text='Mass balance (m d-1)', side=3,
    line=1.5), col = colorRampPalette(c("darkred", "red",
    "blue"))(100))


## End(Not run)

# Calculate glacier surface mass balance using modified settings
# Change thermal conductivity and wind speed applied in the
# implemented function "debrisCoveredIceMelt"
InputGlacierSMBM@thermalConductivity <- 1.5
InputGlacierSMBM@windSpeed <- 5

## Not run:
output <- glacierSMBM(inputGlacierSMBM = InputGlacierSMBM)

# Plot output
plot(output, main = "glacier surface mass balance",
    legend.args=list(text='Mass balance (m d-1)', side=3,
    line=1.5), col = colorRampPalette(c("darkred", "red",
    "blue"))(100))

## End(Not run)
```

`glacierSMBM-method` *Method: Glacier surface mass balance model*

Description

A model consisting of physical and statistical components to calculate the surface mass balance of individual or multiple glaciers.

Details

Glacier surface mass balances are quantified by calculating the difference between the glacier mass gain ascribed to deposited snow and the mass loss associated with melting of bare an debris-covered ice as well as supraglacial snow and firn. The initial glacier outlines and surface conditions

can be defined in 'glacierMask', 'iceMask', 'firnMask', 'snowHeight', 'debrisMask' and 'debrisThickness' (for more information see inputGlacierSMBM-class). Accumulation and ablation is iteratively modelled for every discrete time step (e.g. month, day or hour). The function snowFall is applied to quantify accumulation for every pixel and time step (note: mass input from redistributed snow has not yet been implemented). Ablation does only occur in the model at air temperatures above freezing since the complex process of percolation, retention and refreezing of meltwater is (so far) neglected. The extent and height of supraglacial snow changes with time due to the interplay of snow accumulation (snowFall) and snow melt (snowMelt). Ablation of debris-free (iceMelt) or debris-covered glacier ice (debrisCoveredIceMelt) sets in as soon as the overlying snow pack is completely melted and takes place as long as a pixel is snow-free. For more information please refer to the applied funtions, the examples below or the original publication (Groos et al., submitted).

Value

The potential model output consists of sixteen individual 'RasterLayers' and a summarising table:

1. Accumulation rate (for every time step).
2. Accumulation sum (for the modelling period).
3. Glacial ablation rate (for every time step).
4. Glacial ablation sum (for the modelling period).
5. Ice melt rate (for every time step).
6. Ice melt sum (for the modelling period).
7. Debris ice melt rate (for every time step).
8. Debris ice melt sum (for the modelling period).
9. Firn melt rate (for every time step).
10. Firn melt sum (for the modelling period).
11. Snow melt rate (for every time step).
12. Snow melt sum (for the modelling period).
13. Snow cover (for every time step).
14. Snow height (for the modelling period).
15. Glacier surface mass balance (for every time step).
16. Glacier surface mass balance (for the modelling period).
17. Output table consisting of the spatial average of the fourteen variables above for every time step.

Note

The following input variables in inputGlacierSMBM-class are the requested minimum to run the model:

- 'date' (date and time of each RasterLayer in the following RasterStacks.)
- 'airT' (for every time step)

- 'netRad' (for every time step)
- 'snowfall' or 'precip' (for every time step)
- 'airDensity' (stationary or for every time step)
- 'glacierMask' (stationary or for every time step)
- 'iceMask' (stationary or for every time step)
- 'firnMask' (stationary or for every time step)
- 'debrisMask' (stationary or for every time step)
- 'debrisThickness' (stationary or for every time step)

If the investigated glacier or glacier area is debris-free or snow-free, pass a 'RasterLayer' with the extent of 'glacierMask' and all values = 0 (zero means no snow or no debris, respectively) to the respective 'RasterStack' (e.g. 'debrisMask' or 'firnMask').

If 'inRadSW' and 'inRadLW' are provided instead of 'netRad', the energy-balance at the atmosphere-debris interface is calculated taking the 'debrisAlbedo' and 'thermalEmissivity' of the debris layer into account.

A default value (constant in space and time) is given for each additional argument in inputGlacierSMBM-class like 'windSpeed', 'relativeHumidity' and 'thermalConductivity'. If desired, the default parameters can be modified. Furthermore, there is the option to pass distributed values (stationary or for every time step) instead of general values using the related 'dis*'-arguments like 'disWindSpeed', 'disRelativeHumidity' and 'disThermalConductivity'. In this case, the general parameter is ignored.

File format of written ouput: GeoTIFF: GeoTIFF (1-16) and text file (17).

Author(s)

Alexander R. Groos (<alexander.groos@giub.unibe.ch>)

References

Groos, A.R., Mayer, C., Smiraglia, C., Diolaiuti, G., and Lambrecht A. (submitted). A first attempt to model region-wide glacier surface mass balances in the Karakoram: findings and future challenges. Geografia Fisica e Dinamica Quaternaria.

See Also

glacialMelt, snowMelt, iceMelt, debrisCoveredIceMelt, snowFall, inputGlacierSMBM-class

Examples

```
# Load the provided RasterLayer objects as exemplary
# input for the function
data(airTemperature_30m_daily, airDensity_30m_daily,
    netRad_30m_daily, glacierMask_30m, iceMask_30m, firnMask_30m,
    debrisMask_30m, debrisThickness_30m, precipTuningFactor_30m,
    snowFall_30m_daily, package = "glacierSMBM")
# Individual RasterLayer objects should be loaded or
# created using the function raster()
```

```
# create a three-day virtual meteorological data set
AirT <- stack(airTemperature_30m_daily,
    airTemperature_30m_daily * 0.99,
    airTemperature_30m_daily * 1.01)
NetRad <- stack(netRad_30m_daily,netRad_30m_daily * 0.99,
    netRad_30m_daily * 1.01)
Snowfall <- stack(snowFall_30m_daily, snowFall_30m_daily * 2,
    snowFall_30m_daily * 0.3)

# create a new object of class "inputGlacierSMBM" which is
# requested as input for the glacier surface mass balance model
InputGlacierSMBM <- new("inputGlacierSMBM")

# Add the required data and information to the respective
# slots of the new object (for additional setting options read
# the help section of class "inputGlacierSMBM")
# Create a numeric vector containing date and time of
# the meteorological input data
InputGlacierSMBM@date <- seq.POSIXt(ISOdate(2011,8,15),
    ISOdate(2011,8,17), "days")
InputGlacierSMBM@decimalPlaces <- 4
InputGlacierSMBM@airT <- AirT
InputGlacierSMBM@airDensity <- stack(airDensity_30m_daily)
InputGlacierSMBM@netRad <- NetRad
InputGlacierSMBM@snowfall <- Snowfall
InputGlacierSMBM@glacierMask <- stack(glacierMask_30m)
InputGlacierSMBM@iceMask <- stack(iceMask_30m)
InputGlacierSMBM@firnMask <- stack(firnMask_30m)
InputGlacierSMBM@debrisMask <- stack(debrisMask_30m)
InputGlacierSMBM@debrisThickness <- stack(debrisThickness_30m)
InputGlacierSMBM@disTuningFacPrecip <- stack(precipTuningFactor_30m)

# Calculate glacier surface mass balance using standard settings,
# but suppress to write any output
InputGlacierSMBM@writeOutput <- rep(0, 17)

## Not run:
output <- glacierSMBM(inputGlacierSMBM = InputGlacierSMBM)

# Plot output
plot(output, main = "glacier surface mass balance",
    legend.args=list(text='Mass balance (m d-1)', side=3,
    line=1.5), col = colorRampPalette(c("darkred", "red",
    "blue"))(100))


## End(Not run)

# Calculate glacier surface mass balance using modified settings
# Change thermal conductivity and wind speed applied in the
# implemented function "debrisCoveredIceMelt"
InputGlacierSMBM@thermalConductivity <- 1.5
```

```
InputGlacierSMBM@windSpeed <- 5

## Not run:
output <- glacierSMBM(inputGlacierSMBM = InputGlacierSMBM)

# Plot output
plot(output, main = "glacier surface mass balance",
    legend.args=list(text='Mass balance (m d-1)', side=3,
    line=1.5), col = colorRampPalette(c("darkred", "red",
    "blue"))(100))

## End(Not run)
```

`iceMask_30m` *Data: Ice mask (30m)*

Description

Bare ice at the Liligo Glacier (Karakoram, Pakistan)

Usage

```
data(iceMask_30m)
```

Format

An object of class `'RasterLayer'`.

Details

Dataset: Landsat 5

Date: 2011-08-10

Pixel resolution: 30 m

1 = bare ice, 0 = ice covered by debris, firn or snow

Projection: UTM 43 N

Note: The bare ice distribution was derived from a Landsat 5 image (for more information see Groos et al., submitted).

Source

USGS EarthExplorer

References

Groos, A.R., Mayer, C., Smiraglia, C., Diolaiuti, G., and Lambrecht A. (submitted). A first attempt to model region-wide glacier surface mass balances in the Karakoram: findings and future challenges. Geografia Fisica e Dinamica Quaternaria.

Examples

```
data(iceMask_30m)
plot(iceMask_30m)
```

`iceMelt` *Function: Ice melt model*

Description

A simple model to calculate ice melt based on empirical melting factors.

Usage

```
iceMelt(airT, netRad, glacierMask, iceMask,  tUnit = "K",
    iceTMF = 67*10^-4, disIceTMF = stack(), iceRMF = 0.79*10^-4,
    disIceRMF = stack(), tuningFacAirT = 1,
    disTuningFacAirT = stack(), decimalPlaces = 4,
    outType = "mean", writeOutput = FALSE, outputName = "iceMelt",
    tmpCreate = FALSE, tmpDir = "", outDir = "", ... )
```

Arguments

`airT`	An object of class `'RasterStack'`. Distributed air temperature (Kelvin or degree Celsius). For every time step.
`netRad`	An object of class `'RasterStack'`. Distributed net radiation (W m-2). For every time step.
`glacierMask`	An object of class `'RasterStack'`. Glacier area (1 = glacier, 0 = no glacier). Stationary or for every time step.
`iceMask`	An object of class `RasterStack`. Area of bare glacier ice (1 = bare ice, 0 = no bare ice). Stationary or for every time step.
`tUnit`	An object of class `'character'`. Unit ("K" = Kelvin, "C" = degree Celsius) of air temperature (default = "K").
`iceTMF`	An object of class `numeric`. Temperature melting factor (m K-1 timestep-1) of ice (default = 67*10^-4).
`disIceTMF`	An object of class `RasterStack`. Distributed temperature melting factor (m K-1 timestep-1) of ice. Stationary or for every time step.
`iceRMF`	An object of class `numeric`. Radiative melting factor (m K-1 timestep-1) of ice (default = 0.79*10^-4).
`disIceRMF`	An object of class `RasterStack`. Distributed radiative temperature melting factor (m K-1 timestep-1) of ice. Stationary or for every time step.
`tuningFacAirT`	An object of class `'numeric'`. General air temperature tuning factor (<1 = temperature decrease, 1 = default, >1 = temperature increase).
`disTuningFacAirT`	An object of class `'RasterStack'`. Distributed air temperature tuning factor (`tuningFacAirT`). Stationary or for every time step.

decimalPlaces	An object of class 'numeric'. Number of decimal places (default = 4).
outType	An object of class 'character'. Type of output to be returned by the function: "mean" (default) ice melt or "sum".
writeOutput	An object of class 'logical'. Determines whether the ouput shall be exported as RasterLayer (TRUE) or not (FALSE, default).
outputName	An object of class 'character'. File name for the output RasterLayer(s) (default = "iceMelt").
tmpCreate	An object of class 'logical'. Determines whether a temporary directory should be used (TRUE) or not (FALSE, default). Recommendend if large datasets are processed.
tmpDir	An object of class 'character'. Directory where processing files can be temporarily stored if 'tmpCreate' = TRUE.
outDir	An object of class 'character'. Directory for the output files if 'writeOutput' = TRUE.
...	Further arguments.

Details

An enhanced degree-day model (e.g. Hock, 2003, 2005; Pellicciotti & alii, 2005) is applied to quantify glacier mass loss ascribed to melted ("bare") ice using empirical temperature (TMF) and radiative melting factors (RMF). Fore more information please refer to the examples below or the original publication (Groos et al., submitted, Equation 10)

Value

An object of class 'RasterLayer' returning the calculated spatial distribution of ice melt (e.g. in m d-1, depending on 'tmpRes').

Note

The following input variables are the requested minimum to run the model:

- 'airT' (for every time step)
- 'netRad' (for every time step)
- 'glacierMask' (stationary or for every time step)
- 'iceMask' (stationary or for every time step)

A default value (constant in space and time) is given for each additional argument like 'iceTMF' or 'iceRMF'. If desired, the default parameters can be modified. Furthermore, there is the option to pass distributed values (stationary or for every time step) instead of general values using the related 'dis*'-arguments like 'disIceTMF' or 'disIceRMF'. In this case, the general parameter is ignored.

File format of written ouput: GeoTIFF.

Author(s)

Alexander R. Groos (<alexander.groos@giub.unibe.ch>)

References

Groos, A.R., Mayer, C., Smiraglia, C., Diolaiuti, G., and Lambrecht A. (submitted). A first attempt to model region-wide glacier surface mass balances in the Karakoram: findings and future challenges. Geografia Fisica e Dinamica Quaternaria.

Hock, R. (2003). Temperature index melt modelling in mountain areas. Journal of Hydrology 282, 104-115.

Hock, R. (2005). Glacier melt: a review of processes and their modelling. Progress in Physical Geography 29, 362-391.

Pellicciotti F., Brock B., Strasser U., Burlando P., Funk M. and Corripio J. (2005). An enhanced temperature-index glacier melt model including the shortwave radiation balance: development and testing for Haut Glacier d'Arolla, Switzerland. Journal of Glaciology, 51, 573-587.

See Also

glacialMelt, snowMelt, debrisCoveredIceMelt

Examples

```
# Load the provided RasterLayer objects as exemplary
# input for the function
data(airTemperature_30m_daily, netRad_30m_hourly,
    glacierMask_30m, iceMask_30m, package = "glacierSMBM")
# Individual RasterLayer objects should be loaded or
# created using the function raster()

# Include RasterLayer in RasterStack
AirTemperature_30m_daily <- stack(airTemperature_30m_daily)
NetRad_30m_hourly <- stack(netRad_30m_hourly)
GlacierMask_30m <- stack(glacierMask_30m)
IceMask_30m <- stack(iceMask_30m)

# Calculate ice melt using standard settings
output <- iceMelt(airT = AirTemperature_30m_daily,
    netRad = NetRad_30m_hourly, glacierMask = GlacierMask_30m,
    iceMask = IceMask_30m)

# Plot output
plot(output, main = "ice melt",
    legend.args=list(text='Ice melt (m d-1)', side=3, line=1.5))

# Calculate ice melt using modified setting (e.g. air temperature
# in degree Celsius instead of Kelvin; changes melting factors)
# Therefore exemplarily convert temperature from kelvin to celsius
airTcelsius <- subset(AirTemperature_30m_daily, 1) - 273.15

# Include RasterLayer in RasterStack
airTcelsius <- stack(airTcelsius)

output <- iceMelt(airT = airTcelsius, netRad = NetRad_30m_hourly,
    glacierMask = GlacierMask_30m, tUnit = "C",
```

```
    iceMask = IceMask_30m, iceTMF = 75*10^-4, iceRMF = 1.2*10^-4)

# Plot output
plot(output, main = "ice melt",
    legend.args=list(text='Ice melt (m d-1)', side=3, line=1.5))
```

`iceMelt-method` *Method: Ice melt model*

Description

A simple model to calculate ice melt based on empirical melting factors.

Details

An enhanced degree-day model (e.g. Hock, 2003, 2005; Pellicciotti & alii, 2005) is applied to quantify glacier mass loss ascribed to melted ("bare") ice using empirical temperature (TMF) and radiative melting factors (RMF). Fore more information please refer to the examples below or the original publication (Groos et al., submitted, Equation 10)

Value

An object of class `'RasterLayer'` returning the calculated spatial distribution of ice melt (e.g. in m d-1, depending on `'tmpRes'`).

Note

The following input variables are the requested minimum to run the model:

- `'airT'` (for every time step)
- `'netRad'` (for every time step)
- `'glacierMask'` (stationary or for every time step)
- `'iceMask'` (stationary or for every time step)

A default value (constant in space and time) is given for each additional argument like `'iceTMF'` or `'iceRMF'`. If desired, the default parameters can be modified. Furthermore, there is the option to pass distributed values (stationary or for every time step) instead of general values using the related 'dis*'-arguments like `'disIceTMF'` or `'disIceRMF'`. In this case, the general parameter is ignored.

File format of written ouput: GeoTIFF.

Author(s)

Alexander R. Groos (<alexander.groos@giub.unibe.ch>)

References

Groos, A.R., Mayer, C., Smiraglia, C., Diolaiuti, G., and Lambrecht A. (submitted). A first attempt to model region-wide glacier surface mass balances in the Karakoram: findings and future challenges. Geografia Fisica e Dinamica Quaternaria.

Hock, R. (2003). Temperature index melt modelling in mountain areas. Journal of Hydrology 282, 104-115.

Hock, R. (2005). Glacier melt: a review of processes and their modelling. Progress in Physical Geography 29, 362-391.

Pellicciotti F., Brock B., Strasser U., Burlando P., Funk M. and Corripio J. (2005). An enhanced temperature-index glacier melt model including the shortwave radiation balance: development and testing for Haut Glacier d'Arolla, Switzerland. Journal of Glaciology, 51, 573-587.

See Also

glacialMelt, snowMelt, debrisCoveredIceMelt

Examples

```
# Load the provided RasterLayer objects as exemplary
# input for the function
data(airTemperature_30m_daily, netRad_30m_hourly,
    glacierMask_30m, iceMask_30m, package = "glacierSMBM")
# Individual RasterLayer objects should be loaded or
# created using the function raster()

# Include RasterLayer in RasterStack
AirTemperature_30m_daily <- stack(airTemperature_30m_daily)
NetRad_30m_hourly <- stack(netRad_30m_hourly)
GlacierMask_30m <- stack(glacierMask_30m)
IceMask_30m <- stack(iceMask_30m)

# Calculate ice melt using standard settings
output <- iceMelt(airT = AirTemperature_30m_daily,
    netRad = NetRad_30m_hourly, glacierMask = GlacierMask_30m,
    iceMask = IceMask_30m)

# Plot output
plot(output, main = "ice melt",
    legend.args=list(text='Ice melt (m d-1)', side=3, line=1.5))

# Calculate ice melt using modified setting (e.g. air temperature
# in degree Celsius instead of Kelvin; changes melting factors)
# Therefore exemplarily convert temperature from kelvin to celsius
airTcelsius <- subset(AirTemperature_30m_daily, 1) - 273.15

# Include RasterLayer in RasterStack
airTcelsius <- stack(airTcelsius)

output <- iceMelt(airT = airTcelsius, netRad = NetRad_30m_hourly,
    glacierMask = GlacierMask_30m, tUnit = "C",
```

```
    iceMask = IceMask_30m, iceTMF = 75*10^-4, iceRMF = 1.2*10^-4)

# Plot output
plot(output, main = "ice melt",
    legend.args=list(text='Ice melt (m d-1)', side=3, line=1.5))
```

`inputGlacierSMBM-class` *Class* `inputGlacierSMBM`

Description

A class consisting of parameters and variables required to run and modify `'glacierSMBM'`.

Objects from the Class

Objects can be created by calls of the form `new("inputGlacierSMBM", ...)`.

Slots

`date`: An object of class `'POSIXct'`. Date and time of each `RasterLayer` in the `RasterStack`. Required.

`timeStamp`: An object of class `'POSIXct'`. Date format used in the output table.

`outDir`: An object of class `'character'`. Directory for the output files if `'writeOutput'` = TRUE.

`tmpDir`: An object of class `'character'`. Directory where processing files can be temporarily stored if `'tmpCreate'` = TRUE.

`tmpCreate`: An object of class `'logical'`. Determines whether a temporary directory should be used (TRUE) or not (FALSE, default). Recommendend if large datasets are processed.

`decimalPlaces`: An object of class `'numeric'`. Number of decimal places (default = 4).

`writeOutput`: An object of class `'character'`. Determines which of the seventeen outputs shall be exported as `RasterLayer` (1) and which not (0). For more information see section values.

`outputSep`: An object of class `'character'`. Delimiter for the ouput table (tab is default).

`plotOutput`: An object of class `'logical'`. Determines whether the mass balance results shall be plotted (`TRUE`) or not (`FALSE`, default).

`outputName`: An object of class `'character'`. File name for the output `RasterLayer(s)` (default = "glacierSMBM_Output").

`glacierMask`: An object of class `'RasterStack'`. Glacier area (1 = glacier, 0 = no glacier). Stationary or for every time step. Required.

`iceMask`: An object of class `'RasterStack'`. Area of bare glacier ice (1 = bare ice, 0 = no bare ice). Stationary or for every time step. Required.

`firnMask`: An object of class `'RasterStack'`. Area of supraglacial snow or firn (1 = snow or firn, 0 = no snow or firn). Stationary or for every time step. Required.

`debrisMask`: An object of class `'RasterStack'`. Area of debris covered glacier ice (1 = debris, 0 = no debris). Stationary or for every time step. Required.

`debrisThickness`: An object of class `'RasterStack'`. Distributed supraglacial debris thickness (m). Stationary or for every time step. Required.

`airT`: An object of class `'RasterStack'`. Distributed air temperature (Kelvin or degree Celsius). For every time step. Required.

`airDensity`: An object of class `'RasterStack'`. Distributed air density (kg m-3). Stationary or for every time step. Required.

`netRad`: An object of class `'RasterStack'`. Distributed net radiation (W m-2). For every time step. Required.

`inRadSW`: An object of class `'RasterStack'`. Distributed incoming shortwave radiation (W m-2). For every time step. Optional in the function `'debrisCoveredIceMelt'` instead of `'netRad'`.

`inRadLW`: An object of class `'RasterStack'`. Distributed incoming longwave radiation (W m-2). For every time step. Optional in the function `'debrisCoveredIceMelt'` instead of `'netRad'`.

`snowHeight`: An object of class `'RasterStack'`. Initial snow and firn height (m).

`snowfall`: An object of class `'RasterStack'`. Distributed snowfall (m). Required.

`precip`: An object of class `'RasterStack'`. Distributed precipitation (m). Optional instead of `'snowfall'`.

`tUnit`: An object of class `'character'`. Unit ("K" = Kelvin, "C" = degree Celsius) of air temperature (default = "K").

`iceTMF`: An object of class `'numeric'`. Temperature melting factor (m K-1 timestep-1) of ice (default = 67*10^-4).

`disIceTMF`: An object of class `'RasterStack'`. Distributed temperature melting factor (m K-1 timestep-1) of ice. Stationary or for every time step.

`snowTMF`: An object of class `'numeric'`. Temperature melting factor (m K-1 timestep-1) of snow (default = 45*10^-4).

`disSnowTMF`: An object of class `'RasterStack'`. Distributed temperature melting factor (m K-1 timestep-1) of snow. Stationary or for every time step.

`iceRMF`: An object of class `'numeric'`. Radiative melting factor (m K-1 timestep-1) of ice (default = 0.79*10^-4).

`disIceRMF`: An object of class `'RasterStack'`. Distributed radiative temperature melting factor (m K-1 timestep-1) of ice. Stationary or for every time step.

`snowRMF`: An object of class `'numeric'`. Radiative melting factor (m K-1 timestep-1) of snow (default = 0.53*10^-4).

`disSnowRMF`: An object of class `'RasterStack'`. Distributed radiative temperature melting factor (m K-1 timestep-1) of snow. Stationary or for every time step.

`tuningFacAirT`: An object of class `'numeric'`. General air temperature tuning factor (<1 = temperature decrease, 1 = default, >1 = temperature increase).

`disTuningFacAirT`: An object of class `'RasterStack'`. Distributed air temperature tuning factor (`tuningFacAirT`). Stationary or for every time step.

`tuningFacPrecip`: An object of class `'numeric'`. General precipitation tuning factor (<1 = precipitation decrease, 1 = default, >1 = precipitation increase).

disTuningFacPrecip: An object of class 'RasterStack'. Distributed precipitation tuning factor (tuningFacPrecip). Stationary or for every time step.

snowTransTempThreshold: An object of class 'numeric'. Temperature threshold (same unit as airT) for the transition from rain- to snowfall (default = 274.15 Kelvin).

tmpRes: An object of class 'character'. Time aggregation (temporal resolution) of the input variables (default = "d"). "y" = year, "w" = week, "d" = day, "h" = hour, "s" = second.

measurementHeight: An object of class 'numeric'. Height (m) of meteorological measurements (default = 2).

relativeHumidity: An object of class 'numeric'. Relative humidity (0-1) at measurement height (default = 0.73).

disRelativeHumidity: An object of class 'RasterStack'. Distributed relative humidity (0-1) at measurement height. Stationary or for every time step.

windSpeed: An object of class 'numeric'. Wind speed (m s-1) at measurement height (default = 2).

disWindSpeed: An object of class 'RasterStack'. Distributed wind speed (m s-1) at measurement height. Stationary or for every time step.

debrisAlbedo: An object of class 'numeric'. Albedo (0-1) of the debris (default = 0.07).

disDebrisAlbedo: An object of class 'RasterStack'. Distributed albedo (0-1) of the debris. Stationary or for every time step.

thermalConductivity: An object of class 'numeric'. Thermal conductivity (W m-1 K-1) of the debris layer (default = 0.585).

disThermalConductivity: An object of class 'RasterStack'. Distributed thermal conductivity (W m-1 K-1) of the debris layer. Stationary or for every time step.

thermalEmissivity: An object of class 'numeric'. Thermal emissivity (0-1) of the debris layer (default = 0.95).

disThermalEmissivity: An object of class 'RasterStack'. Distributed thermal emissivity (0-1) of the debris layer. Stationary or for every time step.

surfaceRoughnessHeight: An object of class 'numeric'. Surface roughness height (m) of the debris layer (default = 0.01).

disSurfaceRoughnessHeight: An object of class 'RasterStack'. Distributed surface roughness height (m) of the debris layer. Stationary or for every time step.

frictionVelocity: An object of class 'numeric'. Friction velocity (m s-1) of the debris layer (default = 0.16).

disFrictionVelocity: An object of class 'RasterStack'. Distributed friction velocity (m s-1) of the debris layer. Stationary or for every time step.

volumeFractionDebrisInIce: An object of class 'numeric'. Volume fraction (0-1) of debris in the ice body (default = 0.01).

disVolumeFractionDebrisInIce: An object of class 'RasterStack'. Distributed volume fraction (0-1) of debris in the ice body. Stationary or for every time step.

debrisAirRatio: An object of class 'numeric'. Ratio of the debris surface area to the volume of air (m-1) in the debris layer (default = 188).

disDebrisAirRatio: An object of class 'RasterStack'. Distributed ratio of the debris surface area to the volume of air (m-1) in the debris layer. Stationary or for every time step.

`dragCoefficient`: An object of class `'numeric'`. Drag coefficient (m-1) of the debris layer (default = 5).

`disDragCoefficient`: An object of class `'RasterStack'`. Distributed drag coefficient (m-1) of the debris layer. Stationary or for every time step.

`iceDensity`: An object of class `'numeric'`. Density (kg m-3) of ice (default = 900).

`disIceDensity`: An object of class `'RasterStack'`. Distributed density (kg m-3) of ice. Stationary or for every time step.

Methods

glacierSMBM `signature(inputGlacierSMBM = "inputGlacierSMBM")`: ...

Note

The following input variables are the requested minimum to run the model:

- `date`
- `airT` (for every time step)
- `netRad` (for every time step)
- `airDensity` (stationary or for every time step)
- `snowfall` or `precip` (for every time step)
- `glacierMask` (stationary or for every time step)
- `iceMask` (stationary or for every time step)
- `firnMask` (stationary or for every time step)
- `debrisMask` (stationary or for every time step)
- `debrisThickness` (stationary or for every time step)

Default settings for the individual slots can be viewed and checked by calls in the form of `new("inputGlacierSMBM")`.

Author(s)

Alexander R. Groos (<alexander.groos@giub.unibe.ch>)

See Also

`glacierSMBM`

Examples

```
showClass("inputGlacierSMBM")
```

`interpolateAirP` *Function: Air pressure interpolation*

Description

A function to interpolate reanalysed air pressure data using the barometric height formula.

Usage

```
interpolateAirP(airP, airT, lapseRate, demDiff,
    decimalPlaces = 4, outType = "mean", writeOutput = FALSE,
    outputName = "interpolatedAirP", tmpCreate = TRUE,
    tmpDir = "", outDir = "" )
```

Arguments

`airP`	An object of class `'RasterStack'`. Distributed air pressure (Pa) to be interpolated. For every time step.
`airT`	An object of class `'RasterStack'`. Distributed air temperature (Kelvin or degree Celsius) with the same spatial resolution as `'demDiff'`. For every time step.
`lapseRate`	An object of class `numeric`. Temperature lapse rates (K m-1 or C m-1). For every time step.
`demDiff`	An object of class `RasterLayer`. Height difference (m) between a high-resolution DEM and a resampled DEM (of the same reanalysis dataset as `airP`).
`decimalPlaces`	An object of class `'numeric'`. Number of decimal places (default = 4).
`outType`	An object of class `'character'`. Type of output to be returned by the function: "mean" (default) sub-debris ice melt or "sum".
`writeOutput`	An object of class `'logical'`. Determines whether the ouput shall be exported as `'RasterLayer'` (TRUE) or not (FALSE, default).
`outputName`	An object of class `'character'`. File name for the output `'RasterLayer(s)'` (default = "interpolatedAirP").
`tmpCreate`	An object of class `'logical'`. Determines whether a temporary directory should be used (TRUE) or not (FALSE, default). Recommendend if large datasets are processed.
`tmpDir`	An object of class `'character'`. Directory where processing files can be temporarily stored if `'tmpCreate'` = TRUE.
`outDir`	An object of class `'character'`. Directory for the output files if `'writeOutput'` = TRUE.

Details

Reanalysis data from general circulation models are often the only comprehensive and consistent source of meteorlogical data in remote mountain environments. However, they mostly do not resolve the complex topography and the impacts on air temperature and pressure distribution. Air pressure in high elevations is therefore overestimated and in the valleys underestimated. This function uses the barometric height formula with the height difference between a resampled reanalysis DEM and a high-resolution DEM as input to apply a vertical correction to the reanalysed air pressure distribution. For more details please refer to the given examples or the original publication (Groos et al., submitted).

Value

An object of class `'RasterLayer'` returning the interpolated air pressure distribution (Pa).

Note

File format of written ouput: GeoTIFF.

Author(s)

Alexander R. Groos (<alexander.groos@giub.unibe.ch>)

References

Groos, A.R., Mayer, C., Smiraglia, C., Diolaiuti, G., and Lambrecht A. (submitted). A first attempt to model region-wide glacier surface mass balances in the Karakoram: findings and future challenges. Geografia Fisica e Dinamica Quaternaria.

See Also

`resampleStack`, `interpolateAirT`

Examples

```
# Load the provided RasterLayer objects as exemplary
# input for the function
data(airPressure_10km_daily, airTemperature_30m_daily,
    dem_30m, srtm_dem_30m, package = "glacierSMBM")
# Individual RasterLayer objects should be loaded or
# created using the function raster()

# Calculate difference between SRTM1 DEM and
# resampled reanalysis DEM
DEM_Diff <- dem_30m - srtm_dem_30m

# Interpolate the air pressure obtained from a reanalysis data
# set using a lapse rate and the altitude difference of the
# resampled DEM and high resolution DEM (e.g. SRTM1)
output <- interpolateAirP(airP = stack(airPressure_10km_daily),
    airT = stack(airTemperature_30m_daily), lapseRate = 0.007,
    demDiff = DEM_Diff)
```

```
# Plot output
plot(airPressure_10km_daily, main = "air pressure (10km)",
    legend.args=list(text='Pressure (Pa)', side=3, line=1.5))
plot(output, main = "interpolated air pressure (30m)",
    legend.args=list(text='Pressure (Pa)', side=3, line=1.5))
```

`interpolateAirP-method`

Method: Air pressure interpolation

Description

A function to interpolate reanalysed air pressure data using the barometric height formula.

Details

Reanalysis data from general circulation models are often the only comprehensive and consistent source of meteorlogical data in remote mountain environments. However, they mostly do not resolve the complex topography and the impacts on air temperature and pressure distribution. Air pressure in high elevations is therefore overestimated and in the valleys underestimated. This function uses the barometric height formula with the height difference between a resampled reanalysis DEM and a high-resolution DEM as input to apply a vertical correction to the reanalysed air pressure distribution. For more details please refer to the given examples or the original publication (Groos et al., submitted).

Value

An object of class `'RasterLayer'` returning the interpolated air pressure distribution (Pa).

Note

File format of written ouput: GeoTIFF.

Author(s)

Alexander R. Groos (<alexander.groos@giub.unibe.ch>)

References

Groos, A.R., Mayer, C., Smiraglia, C., Diolaiuti, G., and Lambrecht A. (submitted). A first attempt to model region-wide glacier surface mass balances in the Karakoram: findings and future challenges. Geografia Fisica e Dinamica Quaternaria.

See Also

`resampleStack`, `interpolateAirT`

Examples

```
# Load the provided RasterLayer objects as exemplary
# input for the function
data(airPressure_10km_daily, airTemperature_30m_daily,
    dem_30m, srtm_dem_30m, package = "glacierSMBM")
# Individual RasterLayer objects should be loaded or
# created using the function raster()

# Calculate difference between SRTM1 DEM and
# resampled reanalysis DEM
DEM_Diff <- dem_30m - srtm_dem_30m

# Interpolate the air pressure obtained from a reanalysis data
# set using a lapse rate and the altitude difference of the
# resampled DEM and high resolution DEM (e.g. SRTM1)
output <- interpolateAirP(airP = stack(airPressure_10km_daily),
    airT = stack(airTemperature_30m_daily), lapseRate = 0.007,
    demDiff = DEM_Diff)

# Plot output
plot(airPressure_10km_daily, main = "air pressure (10km)",
    legend.args=list(text='Pressure (Pa)', side=3, line=1.5))
plot(output, main = "interpolated air pressure (30m)",
    legend.args=list(text='Pressure (Pa)', side=3, line=1.5))
```

`interpolateAirT` *Function: Air temperature interpolation*

Description

A function to interpolate reanalysed air temperature data using (empirical) lapse rates and a high-resolution Digital Elevation Model.

Usage

```
interpolateAirT(airT, lapseRate, demDiff, decimalPlaces = 4,
    outType = "mean", writeOutput = FALSE,
    outputName = "interpolatedAirT", tmpCreate = TRUE,
    tmpDir = "", outDir = "" )
```

Arguments

`airT`	An object of class `'RasterStack'`. Distributed air temperature (Kelvin or degree Celsius). For every time step.
`lapseRate`	An object of class `numeric`. Temperature lapse rates (K m-1 or C m-1). For every time step.
`demDiff`	An object of class `RasterLayer`. Height difference (m) between a high-resolution DEM and a resampled DEM (of the same reanalysis dataset as `airT`).

decimalPlaces	An object of class 'numeric'. Number of decimal places (default = 4).
outType	An object of class 'character'. Type of output to be returned by the function: "mean" (default) sub-debris ice melt or "sum".
writeOutput	An object of class 'logical'. Determines whether the ouput shall be exported as 'RasterLayer' (TRUE) or not (FALSE, default).
outputName	An object of class 'character'. File name for the output 'RasterLayer(s)' (default = "interpolatedAirT").
tmpCreate	An object of class 'logical'. Determines whether a temporary directory should be used (TRUE) or not (FALSE, default). Recommendend if large datasets are processed.
tmpDir	An object of class 'character'. Directory where processing files can be temporarily stored if 'tmpCreate' = TRUE.
outDir	An object of class 'character'. Directory for the output files if 'writeOutput' = TRUE.

Details

Reanalysis data from general circulation models are often the only comprehensive and consistent source of meteorlogical data in remote mountain environments. However, they mostly do not resolve the complex topography and the impacts on air temperature and pressure distribution. Air temperatures in the accumulation areas are therefore generally too warm and those in the valleys too cold. This function uses (empirical) lapse rates and the height difference between a resampled reanalysis DEM and a high-resolution DEM to apply a vertical correction to the reanalysed air temperature distribution. For more details please refer to the given examples or the original publication (Groos et al., submitted, Equation 8).

Value

An object of class 'RasterLayer' returning the interpolated air temperature distribution (in Kelvin or degree Celsius, depending on the input).

Note

File format of written ouput: GeoTIFF.

Author(s)

Alexander R. Groos (<alexander.groos@giub.unibe.ch>)

References

Groos, A.R., Mayer, C., Smiraglia, C., Diolaiuti, G., and Lambrecht A. (submitted). A first attempt to model region-wide glacier surface mass balances in the Karakoram: findings and future challenges. Geografia Fisica e Dinamica Quaternaria.

See Also

resampleStack, interpolateAirP

Examples

```
# Load the provided RasterLayer objects as exemplary
# input for the function
data(airTemperature_10km_daily, dem_30m, srtm_dem_30m,
    package = "glacierSMBM")
# Individual RasterLayer objects should be loaded or
# created using the function raster()

# Calculate difference between SRTM1 DEM and resampled
# reanalysis DEM
DEM_Diff <- dem_30m - srtm_dem_30m

# Interpolate the temperature obtained from a reanalysis data set
# using a lapse rate and the altitude difference of the resampled
# reanalysis DEM and a high resolution DEM (e.g. SRTM1)
output <- interpolateAirT(airT = stack(airTemperature_10km_daily),
    lapseRate = 0.007, demDiff = DEM_Diff)

# Plot output
plot(airTemperature_10km_daily, main = "initial air
    temperature (10km)", legend.args=list(text='Temperature (K)',
    side=3, line=1.5))
plot(output, main = "interpolated air temperature (30m)",
    legend.args=list(text='Temperature (K)', side=3, line=1.5))
```

`interpolateAirT-method`

Method: Air temperature interpolation

Description

A function to interpolate reanalysed air temperature data using (empirical) lapse rates and a high-resolution Digital Elevation Model.

Details

Reanalysis data from general circulation models are often the only comprehensive and consistent source of meteorlogical data in remote mountain environments. However, they mostly do not resolve the complex topography and the impacts on air temperature and pressure distribution. Air temperatures in the accumulation areas are therefore generally too warm and those in the valleys too cold. This function uses (empirical) lapse rates and the height difference between a resampled reanalysis DEM and a high-resolution DEM to apply a vertical correction to the reanalysed air temperature distribution. For more details please refer to the given examples or the original publication (Groos et al., submitted, Equation 8).

Value

An object of class `'RasterLayer'` returning the interpolated air temperature distribution (in Kelvin or degree Celsius, depending on the input).

Note

File format of written ouput: GeoTIFF.

Author(s)

Alexander R. Groos (<alexander.groos@giub.unibe.ch>)

References

Groos, A.R., Mayer, C., Smiraglia, C., Diolaiuti, G., and Lambrecht A. (submitted). A first attempt to model region-wide glacier surface mass balances in the Karakoram: findings and future challenges. Geografia Fisica e Dinamica Quaternaria.

See Also

resampleStack, interpolateAirP

Examples

```
# Load the provided RasterLayer objects as exemplary
# input for the function
data(airTemperature_10km_daily, dem_30m, srtm_dem_30m,
    package = "glacierSMBM")
# Individual RasterLayer objects should be loaded or
# created using the function raster()

# Calculate difference between SRTM1 DEM and resampled
# reanalysis DEM
DEM_Diff <- dem_30m - srtm_dem_30m

# Interpolate the temperature obtained from a reanalysis data set
# using a lapse rate and the altitude difference of the resampled
# reanalysis DEM and a high resolution DEM (e.g. SRTM1)
output <- interpolateAirT(airT = stack(airTemperature_10km_daily),
    lapseRate = 0.007, demDiff = DEM_Diff)

# Plot output
plot(airTemperature_10km_daily, main = "initial air
    temperature (10km)", legend.args=list(text='Temperature (K)',
    side=3, line=1.5))
plot(output, main = "interpolated air temperature (30m)",
    legend.args=list(text='Temperature (K)', side=3, line=1.5))
```

---

lst_30m_hourly *Data: Land surface temperature (30m)*

Description

Distributed land surface temperature at the Liligo Glacier (Karakoram, Pakistan)

Usage

```
data(lst_30m_hourly)
```

Format

An object of class 'RasterLayer'.

Details

Dataset: Landsat 5

Date: 2011-08-10

Pixel resolution: 30 m

Unit: K

Projection: UTM 43 N

Note: The distributed land surface temperature was derived from thermal band information of a Landsat 5 image (for more information see Groos et al., submitted).

Source

USGS EarthExplorer

References

Groos, A.R., Mayer, C., Smiraglia, C., Diolaiuti, G., and Lambrecht A. (submitted). A first attempt to model region-wide glacier surface mass balances in the Karakoram: findings and future challenges. Geografia Fisica e Dinamica Quaternaria.

Examples

```
data(lst_30m_hourly)
plot(lst_30m_hourly)
```

---

lst_measured *Data: Land surface temperature (in-situ)*

Description

Land surface temperature "measurements" at the Liligo Glacier (Karakoram, Pakistan)

Usage

```
data(lst_measured)
```

Format

An object of class 'numeric'.

Details

Unit: K

Note: The values of the dataset do not represent real field measurements.

Examples

```
data(lst_measured)
print(lst_measured)
```

---

netRad_30m_daily *Data: Net radiation (30m, daily)*

Description

Distributed net radiation at the Liligo Glacier (Karakoram, Pakistan)

Usage

```
data(netRad_30m_daily)
```

Format

An object of class `'RasterLayer'`.

Details

Dataset: The High Asia Refined Analysis (TU Berlin, Chair of Climatology)

Date: 2011-08-15

Temporal resolution: daily

Pixel resolution: 10 km

Unit: W m-2

Projection: UTM 43 N

Note: The original dataset was resampled to a spatial resolution of 30 m using the function `resample`.

Source

High Asia Refined Analysis

References

Groos, A.R., Mayer, C., Smiraglia, C., Diolaiuti, G., and Lambrecht A. (submitted). A first attempt to model region-wide glacier surface mass balances in the Karakoram: findings and future challenges. Geografia Fisica e Dinamica Quaternaria.

Maussion, F., Scherer, D., Moelg, T., Collier, E., Curio, J., and Finkelnburg, R. (2014). Precipitation Seasonality and Variability over the Tibetan Plateau as Resolved by the High Asia Reanalysis. Journal of Climate 27, 1910-1927.

Examples

```
data(netRad_30m_daily)
plot(netRad_30m_daily)
```

`netRad_30m_hourly` *Data: Net radiation (30m, hourly)*

Description

Distributed net radiation at the Liligo Glacier (Karakoram, Pakistan)

Usage

```
data(netRad_30m_hourly)
```

Format

An object of class `'RasterLayer'`.

Details

Dataset: The High Asia Refined Analysis (TU Berlin, Chair of Climatology)

Date: 2011-08-15

Temporal resolution: hourly

Pixel resolution: 10 km

Unit: W m-2

Projection: UTM 43 N

Note: The original dataset was resampled to a spatial resolution of 30 m using the function `resample`.

Source

High Asia Refined Analysis

References

Groos, A.R., Mayer, C., Smiraglia, C., Diolaiuti, G., and Lambrecht A. (submitted). A first attempt to model region-wide glacier surface mass balances in the Karakoram: findings and future challenges. Geografia Fisica e Dinamica Quaternaria.

Maussion, F., Scherer, D., Moelg, T., Collier, E., Curio, J., and Finkelnburg, R. (2014). Precipitation Seasonality and Variability over the Tibetan Plateau as Resolved by the High Asia Reanalysis. Journal of Climate 27, 1910-1927.

Examples

```
data(netRad_30m_hourly)
plot(netRad_30m_hourly)
```

precipTuningFactor_30m *Data: Precipitation tuning (30m)*

Description

Distributed precipitation tuning data for the Liligo Glacier (Karakoram, Pakistan)

Usage

```
data(precipTuningFactor_30m)
```

Format

An object of class `'RasterLayer'`.

Details

Values: <1 = precipitation decrease, 1 = default, >1 = precipitation increase

References

Groos, A.R., Mayer, C., Smiraglia, C., Diolaiuti, G., and Lambrecht A. (submitted). A first attempt to model region-wide glacier surface mass balances in the Karakoram: findings and future challenges. Geografia Fisica e Dinamica Quaternaria.

Examples

```
data(precipTuningFactor_30m)
plot(precipTuningFactor_30m)
```

precip_10km_daily *Data: Precipitation (10km, daily)*

Description

Distributed precipitation at the Liligo Glacier (Karakoram, Pakistan)

Usage

```
data(precip_10km_daily)
```

Format

An object of class `'RasterLayer'`.

Details

Dataset: The High Asia Refined Analysis (TU Berlin, Chair of Climatology)

Date: 2011-08-15

Temporal resolution: daily

Pixel resolution: 10 km

Unit: m

Note: Projection: UTM 43 N

Source

High Asia Refined Analysis

References

Groos, A.R., Mayer, C., Smiraglia, C., Diolaiuti, G., and Lambrecht A. (submitted). A first attempt to model region-wide glacier surface mass balances in the Karakoram: findings and future challenges. Geografia Fisica e Dinamica Quaternaria.

Maussion, F., Scherer, D., Moelg, T., Collier, E., Curio, J., and Finkelnburg, R. (2014). Precipitation Seasonality and Variability over the Tibetan Plateau as Resolved by the High Asia Reanalysis. Journal of Climate 27, 1910-1927.

Examples

```
data(precip_10km_daily)
plot(precip_10km_daily)
```

---

precip_30m_daily *Data: Precipitation (30m, daily)*

Description

Distributed precipitation at the Liligo Glacier (Karakoram, Pakistan)

Usage

```
data(precip_30m_daily)
```

Format

An object of class `'RasterLayer'`.

Details

Dataset: The High Asia Refined Analysis (TU Berlin, Chair of Climatology)

Date: 2011-08-15

Temporal resolution: daily

Pixel resolution: 10 km

Unit: m

Projection: UTM 43 N

Note: The original dataset was resampled to a spatial resolution of 30 m using the function `resample`.

Source

High Asia Refined Analysis

References

Groos, A.R., Mayer, C., Smiraglia, C., Diolaiuti, G., and Lambrecht A. (submitted). A first attempt to model region-wide glacier surface mass balances in the Karakoram: findings and future challenges. Geografia Fisica e Dinamica Quaternaria.

Maussion, F., Scherer, D., Moelg, T., Collier, E., Curio, J., and Finkelnburg, R. (2014). Precipitation Seasonality and Variability over the Tibetan Plateau as Resolved by the High Asia Reanalysis. Journal of Climate 27, 1910-1927.

Examples

```
data(precip_30m_daily)
plot(precip_30m_daily)
```

---

resampleStack *Function: Resample a RasterStack object*

Description

`'ResampleStack'` transfers values between non matching `'RasterLayers'` in terms of origin and spatial resolution. To adjust the spatial resolution, a bilinear interpolation is applied.

Usage

```
resampleStack(rasterStack, resampleSettings, decimalPlaces = 4,
    outType = "mean", writeOutput = FALSE,
    outputName = "resample", tmpCreate = FALSE, tmpDir = "",
    outDir = "", ... )
```

Arguments

rasterStack	A 'RasterStack' object to be resampled.
resampleSettings	A 'RasterLayer' object with the settings that 'RasterStack' should be resampled to.
decimalPlaces	An object of class 'numeric'. Number of decimal places (default = 4).
outType	An object of class 'character'. Type of output to be returned by the function: "mean" (default) or "sum".
writeOutput	An object of class 'logical'. Determines whether the ouput shall be exported as RasterLayer (TRUE) or not (FALSE, default).
outputName	An object of class 'character'. File name for the output RasterLayer(s) (default = "dcIceMelt").
tmpCreate	An object of class 'logical'. Determines whether a temporary directory should be used (TRUE) or not (FALSE, default). Recommendend if large datasets are processed.
tmpDir	An object of class 'character'. Directory where processing files can be temporarily stored if 'tmpCreate' = TRUE.
outDir	An object of class 'character'. Directory for the output files if 'writeOutput' = TRUE.
...	Further arguments passed to the function unitConv, which is implemented in resampleStack. For more information see section details.

Details

In addition to the function resample, 'resampleStack' provides the export of each resampled 'RasterLayer' as GeoTIFF.

If it is desired to convert the unit ('u1') of the 'RasterStack' into a different unit ('u2'), the arguments ('u1') and ('u2') (see unitConv) can be additionally passed to 'rasterStack'.

Value

An object of class 'RasterLayer' returning the mean or sum (depending on outType) of the resampled 'RasterStack'.

Note

File format of written ouput: GeoTIFF.

Author(s)

Alexander R. Groos (<alexander.groos@giub.unibe.ch>)

See Also

resample, unitConv

Examples

```
# Load the provided RasterLayer objects as exemplary
# input for the function
data(airTemperature_30m_daily, precip_10km_daily,
    package = "glacierSMBM")
# Individual RasterLayer objects should be loaded or created
# using the function raster()

# Resample and interpolate a precipitation file with 10 km spatial
# resolution using the parameters of air temperature (30m)
# Additionally, convert from mm h-1 to m d-1
output <- resampleStack(stack(precip_10km_daily),
    airTemperature_30m_daily,1, writeOutput = FALSE,
    u1 = "mm/h", u2 = "mm/d", decimalPlaces = 5)

# Plot output
plot(precip_10km_daily, main = "precipitation (10km)",
    legend.args=list(text='precipitation (m)', side=3, line=1.5))
plot(output, main = "resampled precipitation (30m)",
    legend.args=list(text='precipitation (m)', side=3, line=1.5))
```

---

resampleStack-method *Method: Resample a RasterStack object*

Description

ResampleStack transfers values between non matching RasterLayers in terms of origin and spatial resolution. To adjust the spatial resolution, a bilinear interpolation is applied.

Details

In addition to the function resample, 'resampleStack' provides the export of each resampled 'RasterLayer' as GeoTIFF.

If it is desired to convert the unit ('u1') of the 'RasterStack' into a different unit ('u2'), the arguments ('u1') and ('u2') (see unitConv) can be additionally passed to 'rasterStack'.

Value

An object of class 'RasterLayer' returning the mean or sum (depending on outType) of the resampled 'RasterStack'.

Note

File format of written ouput: GeoTIFF.

Author(s)

Alexander R. Groos (<alexander.groos@giub.unibe.ch>)

See Also

resample, unitConv

Examples

```
# Load the provided RasterLayer objects as exemplary
# input for the function
data(airTemperature_30m_daily, precip_10km_daily,
    package = "glacierSMBM")
# Individual RasterLayer objects should be loaded or created
# using the function raster()

# Resample and interpolate a precipitation file with 10 km spatial
# resolution using the parameters of air temperature (30m)
# Additionally, convert from mm h-1 to m d-1
output <- resampleStack(stack(precip_10km_daily),
    airTemperature_30m_daily,1, writeOutput = FALSE,
    u1 = "mm/h", u2 = "mm/d", decimalPlaces = 5)

# Plot output
plot(precip_10km_daily, main = "precipitation (10km)",
    legend.args=list(text='precipitation (m)', side=3, line=1.5))
plot(output, main = "resampled precipitation (30m)",
    legend.args=list(text='precipitation (m)', side=3, line=1.5))
```

---

selectedCoordinates *Data: Locations of in-situ measurement*

Description

A selection of locations of in-situ "measurements" at the Liligo Glacier (Karakoram, Pakistan)

Usage

```
data(selectedCoordinates)
```

Format

An object of class 'numeric'.

Details

Coordinates in UTM 43 N x = longitude y = latitude

Note: The values of the dataset do not represent real field measurements.

Examples

```
data(selectedCoordinates)
print(selectedCoordinates)
```

snowFall *Function: Snowfall model*

Description

A simple model to derive snowfall from precipitation and air temperature.

Usage

```
snowFall(airT, precip, glacierMask,
    snowTransTempThreshold = 274.15, tuningFacPrecip = 1,
    disTuningFacPrecip = stack(), tuningFacAirT = 1,
    disTuningFacAirT = stack(), decimalPlaces = 4,
    outType = "mean", writeOutput = FALSE,
    outputName = "snowfall", tmpCreate = FALSE,
    tmpDir = "", outDir = "", ... )
```

Arguments

airT	An object of class 'RasterStack'. Distributed air temperature (Kelvin or Celsius). For every time step.
precip	An object of class 'RasterStack'. Distributed precipitation (m). For every time step.
glacierMask	An object of class 'RasterStack'. Glacier area (1 = glacier, 0 = no glacier). Stationary or for every time step.
snowTransTempThreshold	An object of class 'numeric'. Temperature threshold (same unit as 'airT') for the transition from rain- to snowfall (default = 274.15 Kelvin).
tuningFacPrecip	An object of class 'numeric'. General precipitation tuning factor (<1 = snowfall decrease, 1 = default, >1 = snowfall increase).
disTuningFacPrecip	An object of class 'numeric'. Distributed precipitation tuning factor (tuningFacPrecip). Stationary or for every time step.
tuningFacAirT	An object of class 'numeric'. General air temperature tuning factor (<1 = temperature decrease, 1 = default, >1 = temperature increase).
disTuningFacAirT	An object of class 'RasterStack'. Distributed air temperature tuning factor ('tuningFacAirT'). Stationary or for every time step.
decimalPlaces	An object of class 'numeric'. Number of decimal places (default = 4).
outType	An object of class 'character'. Type of output to be returned by the function: "mean" (default) snowfall or "sum".
writeOutput	An object of class 'logical'. Determines whether the ouput shall be exported as 'RasterLayer' (TRUE) or not (FALSE, default).

outputName	An object of class 'character'. File name for the output 'RasterLayer(s)' (default = "dcIceMelt").
tmpCreate	An object of class 'logical'. Determines whether a temporary directory should be used (TRUE) or not (FALSE, default). Recommendend if large datasets are processed.
tmpDir	An object of class 'character'. Directory where processing files can be temporarily stored if 'tmpCreate' = TRUE.
outDir	An object of class 'character'. Directory for the output files if 'writeOutput' = TRUE.
...	Further arguments.

Details

A simple empirical air temperature threshold (e.g. 1 degree Celsius) is applied to derive snowfall from precipitation (e.g. Rohrer and Braun, 1994). Snowfall is defined as precipitation below the temperature threshold. For temperatures above the threshold, no precipitation is considered since refreezing and percolation processes have yet not been implemented in 'glacierSMBM'. For more information see Groos et al. (submitted).

Value

An object of class 'RasterLayer' returning the calculated snowfall distribution (m timestep-1).

Note

The following input variables are the requested minimum to run the model:

- 'airT' (for every time step)
- 'precip' (for every time step)
- 'glacierMask' (stationary or for every time step)
- 'snowTransTempThreshold' (parameter)

A default value (constant in space and time) is given for the additional arguments like 'tuningFacPrecip' or 'tuningFacAirT'. If desired, the default parameters can be modified. Furthermore, there is the option to pass distributed values (stationary or for every time step) instead of general values using the related 'dis*'-arguments like 'disTuningFacPrecip' or disTuningFacAirT. In this case, the general parameter is ignored.

File format of written ouput: GeoTIFF.

Author(s)

Alexander R. Groos (<alexander.groos@giub.unibe.ch>)

References

Groos, A.R., Mayer, C., Smiraglia, C., Diolaiuti, G., and Lambrecht A. (submitted). A first attempt to model region-wide glacier surface mass balances in the Karakoram: findings and future challenges. Geografia Fisica e Dinamica Quaternaria.

Rohrer, M., and Braun, L. (1994). Long-Term Records of Snow Cover Water Equivalent in the Swiss Alps 2. Simulation. Nordic Hydrology 25, 65-78.

See Also

glacialMelt

Examples

```
# Load the provided RasterLayer objects as exemplary
# input for the function
# Individual RasterLayer objects should be loaded or
# created using the function raster()
data(airTemperature_30m_daily, precip_30m_daily,
    glacierMask_30m, package = "glacierSMBM")

# Include RasterLayer in RasterStack
AirTemperature_30m_daily <- stack(airTemperature_30m_daily)
Precip_30m_daily <- stack(precip_30m_daily)
GlacierMask_30m <- stack(glacierMask_30m)

# Calculate snow fall from air temperature and total
# precipitation using standard settings
output <- snowFall(airT = AirTemperature_30m_daily,
    precip = Precip_30m_daily, glacierMask = GlacierMask_30m)

# Plot output
plot(output, main = "snow fall",
    legend.args=list(text='Snowfall (m)', side=3, line=1.5))

# Calculate snow fall from air temperature and total
# precipitation using modified settings (e.g. snow transition
# is changed from 1 to 0 degree Celsius) and general
# precipitation tuning factor is set from 1 to 0.8)
output <- snowFall(airT = AirTemperature_30m_daily,
    precip = Precip_30m_daily, glacierMask = GlacierMask_30m,
    tuningFacPrecip = 0.8, snowTransTempThreshold = 273.15)

# Plot output
plot(output, main = "snow fall",
    legend.args=list(text='Snowfall (m)', side=3, line=1.5))
```

snowFall-method *Method: Snowfall model*

Description

A simple model to derive snowfall from precipitation and air temperature.

Details

A simple empirical air temperature threshold (e.g. 1 degree Celsius) is applied to derive snowfall from precipitation (e.g. Rohrer and Braun, 1994). Snowfall is defined as precipitation below the temperature threshold. For temperatures above the threshold, no precipitation is considered since refreezing and percolation processes have yet not been implemented in `'glacierSMBM'`. For more information see Groos et al. (submitted).

Value

An object of class `'RasterLayer'` returning the calculated snowfall distribution (m timestep-1).

Note

The following input variables are the requested minimum to run the model:

- `'airT'` (for every time step)
- `'precip'` (for every time step)
- `'glacierMask'` (stationary or for every time step)
- `'snowTransTempThreshold'` (parameter)

A default value (constant in space and time) is given for the additional arguments like `'tuningFacPrecip'` or `'tuningFacAirT'`. If desired, the default parameters can be modified. Furthermore, there is the option to pass distributed values (stationary or for every time step) instead of general values using the related 'dis*'-arguments like `'disTuningFacPrecip'` or `disTuningFacAirT`. In this case, the general parameter is ignored.

File format of written ouput: GeoTIFF.

Author(s)

Alexander R. Groos (<alexander.groos@giub.unibe.ch>)

References

Groos, A.R., Mayer, C., Smiraglia, C., Diolaiuti, G., and Lambrecht A. (submitted). A first attempt to model region-wide glacier surface mass balances in the Karakoram: findings and future challenges. Geografia Fisica e Dinamica Quaternaria.

Rohrer, M., and Braun, L. (1994). Long-Term Records of Snow Cover Water Equivalent in the Swiss Alps 2. Simulation. Nordic Hydrology 25, 65-78.

See Also

glacialMelt

Examples

```
# Load the provided RasterLayer objects as exemplary
# input for the function
# Individual RasterLayer objects should be loaded or
# created using the function raster()
data(airTemperature_30m_daily, precip_30m_daily,
    glacierMask_30m, package = "glacierSMBM")

# Include RasterLayer in RasterStack
AirTemperature_30m_daily <- stack(airTemperature_30m_daily)
Precip_30m_daily <- stack(precip_30m_daily)
GlacierMask_30m <- stack(glacierMask_30m)

# Calculate snow fall from air temperature and total
# precipitation using standard settings
output <- snowFall(airT = AirTemperature_30m_daily,
    precip = Precip_30m_daily, glacierMask = GlacierMask_30m)

# Plot output
plot(output, main = "snow fall",
    legend.args=list(text='Snowfall (m)', side=3, line=1.5))

# Calculate snow fall from air temperature and total
# precipitation using modified settings (e.g. snow transition
# is changed from 1 to 0 degree Celsius) and general
# precipitation tuning factor is set from 1 to 0.8)
output <- snowFall(airT = AirTemperature_30m_daily,
    precip = Precip_30m_daily, glacierMask = GlacierMask_30m,
    tuningFacPrecip = 0.8, snowTransTempThreshold = 273.15)

# Plot output
plot(output, main = "snow fall",
    legend.args=list(text='Snowfall (m)', side=3, line=1.5))
```

---

snowFall_30m_daily *Data: Snowfall (30m, daily)*

Description

Distributed snowfall at the Liligo Glacier (Karakoram, Pakistan)

Usage

```
data(snowFall_30m_daily)
```

Format

An object of class 'RasterLayer'.

Details

Dataset: The High Asia Refined Analysis (TU Berlin, Chair of Climatology)

Date: 2011-08-15

Temporal resolution: daily

Pixel resolution: 30 m

Unit: m

Projection: UTM 43 N

Note: Distributed snowfall was derived from precipitation using the function 'snowFall'.

Source

High Asia Refined Analysis

References

Groos, A.R., Mayer, C., Smiraglia, C., Diolaiuti, G., and Lambrecht A. (submitted). A first attempt to model region-wide glacier surface mass balances in the Karakoram: findings and future challenges. Geografia Fisica e Dinamica Quaternaria.

Maussion, F., Scherer, D., Moelg, T., Collier, E., Curio, J., and Finkelnburg, R. (2014). Precipitation Seasonality and Variability over the Tibetan Plateau as Resolved by the High Asia Reanalysis. Journal of Climate 27, 1910-1927.

Examples

```
data(snowFall_30m_daily)
plot(snowFall_30m_daily)
```

---

snowMelt *Function: Snow melt model*

Description

A simple model to calculate snow melt based on empirical melting factors.

Usage

```
snowMelt(airT, netRad, glacierMask, snowMask,  tUnit = "K",
    snowTMF = 45*10^-4, disSnowTMF = stack(),
    snowRMF = 0.53*10^-4, disSnowRMF = stack(),
    tuningFacAirT = 1, disTuningFacAirT = stack(),
    decimalPlaces = 4, outType = "mean", writeOutput = FALSE,
    outputName = "snowMelt", tmpCreate = FALSE,
    tmpDir = "", outDir = "", ... )
```

Arguments

airT	An object of class 'RasterStack'. Distributed air temperature (Kelvin or degree Celsius). For every time step.
netRad	An object of class 'RasterStack'. Distributed net radiation (W m-2). For every time step.
glacierMask	An object of class 'RasterStack'. Glacier area (1 = glacier, 0 = no glacier). Stationary or for every time step.
snowMask	An object of class 'RasterStack'. Area of supraglacial snow or firn (1 = snow or firn, 0 = no snow or firn). Stationary or for every time step.
tUnit	An object of class 'character'. Unit ("K" = Kelvin, "C" = degree Celsius) of air temperature (default = "K").
snowTMF	An object of class 'numeric'. Temperature melting factor (m K-1 timestep-1) of snow (default = 45*10^-4).
disSnowTMF	An object of class 'RasterStack'. Distributed temperature melting factor (m K-1 timestep-1) of snow. Stationary or for every time step.
snowRMF	An object of class 'numeric'. Radiative melting factor (m K-1 timestep-1) of snow (default = 0.53*10^-4).
disSnowRMF	An object of class 'RasterStack'. Distributed radiative temperature melting factor (m K-1 timestep-1) of snow. Stationary or for every time step.
tuningFacAirT	An object of class 'numeric'. General air temperature tuning factor (<1 = temperature decrease, 1 = default, >1 = temperature increase).
disTuningFacAirT	An object of class 'RasterStack'. Distributed air temperature tuning factor ('tuningFacAirT'). Stationary or for every time step.
decimalPlaces	An object of class 'numeric'. Number of decimal places (default = 4).
outType	An object of class 'character'. Type of output to be returned by the function: "mean" (default) snow melt or "sum".
writeOutput	An object of class 'logical'. Determines whether the ouput shall be exported as 'RasterLayer' (TRUE) or not (FALSE, default).
outputName	An object of class 'character'. File name for the output 'RasterLayer(s)' (default = "snowMelt").
tmpCreate	An object of class 'logical'. Determines whether a temporary directory should be used (TRUE) or not (FALSE, default). Recommendend if large datasets are processed.
tmpDir	An object of class 'character'. Directory where processing files can be temporarily stored if 'tmpCreate' = TRUE.
outDir	An object of class 'character'. Directory for the output files if 'writeOutput' = TRUE.
...	Further arguments.

Details

An enhanced degree-day model (e.g. Hock, 2003, 2005; Pellicciotti & alii, 2005) is applied to quantify glacier mass loss ascribed to melted snow and/or firn using empirical temperature (TMF) and radiative melting factors (RMF). Fore more information please refer to the examples below or the original publication (Groos et al., submitted, Equation 10)

Value

An object of class `'RasterLayer'` returning the calculated spatial distribution of snow melt (e.g. in m d-1, depending on `'tmpRes'`).

Note

The following input variables are the requested minimum to run the model:

- `'airT'` (for every time step)
- `'netRad'` (for every time step)
- `'glacierMask'` (stationary or for every time step)
- `'snowMask'` (stationary or for every time step)

A default value (constant in space and time) is given for each additional argument like `'snowTMF'` or `'snowRMF'`. If desired, the default parameters can be modified. Furthermore, there is the option to pass distributed values (stationary or for every time step) instead of general values using the related 'dis*'-arguments like `'disSnowTMF'` or `'disSnowRMF'`. In this case, the general parameter is ignored.

File format of written ouput: GeoTIFF.

Author(s)

Alexander R. Groos (<alexander.groos@giub.unibe.ch>)

References

Groos, A.R., Mayer, C., Smiraglia, C., Diolaiuti, G., and Lambrecht A. (submitted). A first attempt to model region-wide glacier surface mass balances in the Karakoram: findings and future challenges. Geografia Fisica e Dinamica Quaternaria.

Hock, R. (2003). Temperature index melt modelling in mountain areas. Journal of Hydrology 282, 104-115.

Hock, R. (2005). Glacier melt: a review of processes and their modelling. Progress in Physical Geography 29, 362-391.

Pellicciotti F., Brock B., Strasser U., Burlando P., Funk M. and Corripio J. (2005). An enhanced temperature-index glacier melt model including the shortwave radiation balance: development and testing for Haut Glacier d'Arolla, Switzerland. Journal of Glaciology, 51, 573-587.

See Also

`glacialMelt`, `iceMelt`, `debrisCoveredIceMelt`

Examples

```
# Load the provided RasterLayer objects as exemplary
# input for the function
data(airTemperature_30m_daily, netRad_30m_hourly,
    glacierMask_30m, firnMask_30m, package = "glacierSMBM")
# Individual RasterLayer objects should be loaded or
# created using the function raster()

# Include RasterLayer in RasterStack
AirTemperature_30m_daily <- stack(airTemperature_30m_daily * 1.01)
NetRad_30m_hourly <- stack(netRad_30m_hourly)
GlacierMask_30m <- stack(glacierMask_30m)
FirnMask_30m <- stack(firnMask_30m)

# Calculate snow melt using standard settings
output <- snowMelt(airT = AirTemperature_30m_daily,
    netRad = NetRad_30m_hourly, glacierMask = GlacierMask_30m,
    snowMask = FirnMask_30m)

# Plot output
plot(output, main = "snow melt",
    legend.args=list(text='Snow melt (m d-1)', side=3, line=1.5))

# Calculate snow melt using modified setting (e.g. air
# temperature in degree Celsius instead of Kelvin; changes
# melting factors)
# Therefore exemplarily convert temperature from kelvin to
# degree Celsius
airTcelsius <- subset(AirTemperature_30m_daily, 1) - 273.15

# Include RasterLayer in RasterStack
airTcelsius <- stack(airTcelsius)

output <- snowMelt(airT = airTcelsius,
    netRad = NetRad_30m_hourly, glacierMask = GlacierMask_30m,
    tUnit = "C", snowMask = FirnMask_30m, snowTMF = 75*10^-4,
    snowRMF = 1.2*10^-4)

# Plot output
plot(output, main = "snow melt",
    legend.args=list(text='Snow melt (m d-1)', side=3, line=1.5))
```

`snowMelt-method` *Method: Snow melt model*

Description

A simple model to calculate snow melt based on empirical melting factors.

Details

An enhanced degree-day model (e.g. Hock, 2003, 2005; Pellicciotti & alii, 2005) is applied to quantify glacier mass loss ascribed to melted snow and/or firn using empirical temperature (TMF) and radiative melting factors (RMF). Fore more information please refer to the examples below or the original publication (Groos et al., submitted, Equation 10)

Value

An object of class `'RasterLayer'` returning the calculated spatial distribution of snow melt (e.g. in m d-1, depending on `'tmpRes'`).

Note

The following input variables are the requested minimum to run the model:

- `'airT'` (for every time step)
- `'netRad'` (for every time step)
- `'glacierMask'` (stationary or for every time step)
- `'snowMask'` (stationary or for every time step)

A default value (constant in space and time) is given for each additional argument like `'snowTMF'` or `'snowRMF'`. If desired, the default parameters can be modified. Furthermore, there is the option to pass distributed values (stationary or for every time step) instead of general values using the related 'dis*'-arguments like `'disSnowTMF'` or `'disSnowRMF'`. In this case, the general parameter is ignored.

File format of written ouput: GeoTIFF.

Author(s)

Alexander R. Groos (<alexander.groos@giub.unibe.ch>)

References

Groos, A.R., Mayer, C., Smiraglia, C., Diolaiuti, G., and Lambrecht A. (submitted). A first attempt to model region-wide glacier surface mass balances in the Karakoram: findings and future challenges. Geografia Fisica e Dinamica Quaternaria.

Hock, R. (2003). Temperature index melt modelling in mountain areas. Journal of Hydrology 282, 104-115.

Hock, R. (2005). Glacier melt: a review of processes and their modelling. Progress in Physical Geography 29, 362-391.

Pellicciotti F., Brock B., Strasser U., Burlando P., Funk M. and Corripio J. (2005). An enhanced temperature-index glacier melt model including the shortwave radiation balance: development and testing for Haut Glacier d'Arolla, Switzerland. Journal of Glaciology, 51, 573-587.

See Also

`glacialMelt`, `iceMelt`, `debrisCoveredIceMelt`

Examples

```
# Load the provided RasterLayer objects as exemplary
# input for the function
data(airTemperature_30m_daily, netRad_30m_hourly,
    glacierMask_30m, firnMask_30m, package = "glacierSMBM")
# Individual RasterLayer objects should be loaded or
# created using the function raster()

# Include RasterLayer in RasterStack
AirTemperature_30m_daily <- stack(airTemperature_30m_daily * 1.01)
NetRad_30m_hourly <- stack(netRad_30m_hourly)
GlacierMask_30m <- stack(glacierMask_30m)
FirnMask_30m <- stack(firnMask_30m)

# Calculate snow melt using standard settings
output <- snowMelt(airT = AirTemperature_30m_daily,
    netRad = NetRad_30m_hourly, glacierMask = GlacierMask_30m,
    snowMask = FirnMask_30m)

# Plot output
plot(output, main = "snow melt",
    legend.args=list(text='Snow melt (m d-1)', side=3, line=1.5))

# Calculate snow melt using modified setting (e.g. air
# temperature in degree Celsius instead of Kelvin; changes
# melting factors)
# Therefore exemplarily convert temperature from kelvin to
# degree Celsius
airTcelsius <- subset(AirTemperature_30m_daily, 1) - 273.15

# Include RasterLayer in RasterStack
airTcelsius <- stack(airTcelsius)

output <- snowMelt(airT = airTcelsius,
    netRad = NetRad_30m_hourly, glacierMask = GlacierMask_30m,
    tUnit = "C", snowMask = FirnMask_30m, snowTMF = 75*10^-4,
    snowRMF = 1.2*10^-4)

# Plot output
plot(output, main = "snow melt",
    legend.args=list(text='Snow melt (m d-1)', side=3, line=1.5))
```

`srtm_dem_30m` *Data: SRTM DEM (30m)*

Description

Digital elevation model of the Liligo Glacier (Karakoram, Pakistan)

Usage

```
data(srtm_dem_30m)
```

Format

An object of class 'RasterLayer'.

Details

Dataset: SRTM DEM

Pixel resolution: 30m

Unit: m

Projection: UTM 43 N

Source

USGS EarthExplorer

References

Groos, A.R., Mayer, C., Smiraglia, C., Diolaiuti, G., and Lambrecht A. (submitted). A first attempt to model region-wide glacier surface mass balances in the Karakoram: findings and future challenges. Geografia Fisica e Dinamica Quaternaria.

Examples

```
data(srtm_dem_30m)
plot(srtm_dem_30m)
```

---

unitConv *Function: Unit conversion of a RasterLayer object*

Description

Converts the cell values of a 'RasterLayer' from one unit to another.

Usage

```
unitConv(rasterLayer, u1, u2, decimalPlaces = 4)
```

Arguments

rasterLayer	A 'RasterLayer' object to be converted.
u1	An object of class 'character'. Unit of the input 'RasterLayer'.
u2	An object of class 'character'. Requested unit of the output 'RasterLayer'.
decimalPlaces	An object of class 'numeric'. Number of decimal places (default = 4).

Details

This function makes use of ud.convert from the package 'udunits2' to calculate a conversion factor which is applied to convert unit 'u1' of a 'RasterLayer' into the requested unit 'u2'.

A list of common units: "kelvin", "celsius", "mm/d", "mm/h", "mm/s", "cm/d", "cm/h", "cm/s", "m/d", "m/h", "m/s".

Value

Returns the converted 'RasterLayer'.

Author(s)

Alexander R. Groos (<alexander.groos@giub.unibe.ch>)

See Also

ud.convert

Examples

```
# Load the provided RasterLayer objects as exemplary
# input for the function
data(airTemperature_30m_daily, precip_30m_daily,
    package = "glacierSMBM")
# Individual RasterLayer objects should be loaded or
# created using the function raster()

# Convert air temperature from Kelvin to degree Celsius
output <- unitConv(airTemperature_30m_daily, "kelvin", "celsius")

# Plot output
plot(airTemperature_30m_daily, main = "Air Temperature [Kelvin]",
    legend.args=list(text='Temperature (K)', side=3, line=1.5))
plot(output, main = "Air Temperature [Celsius]",
    legend.args=list(text='Temperature (Celsius)',
    side=3, line=1.5))

# Convert precipitation from m d-1 to mm h-1
output <- unitConv(precip_30m_daily, "m/d", "mm/h")

# Plot output
plot(precip_30m_daily, main = "Precipitation [m/d]",
    legend.args=list(text='Precipitation (m/d)',
    side=3, line=1.5))
plot(output, main = "Precipitation [mm/h]",
    legend.args=list(text='Precipitation (mm/h)',
    side=3, line=1.5))
```

unitConv-method *Method: Unit conversion of a RasterLayer object*

Description

Converts the cell values of a 'RasterLayer' from one unit to another.

Details

This function makes use of ud.convert from the package 'udunits2' to calculate a conversion factor which is applied to convert unit 'u1' of a 'RasterLayer' into the requested unit 'u2'.

A list of common units: "kelvin", "celsius", "mm/d", "mm/h", "mm/s", "cm/d", "cm/h", "cm/s", "m/d", "m/h", "m/s".

Value

Returns the converted 'RasterLayer'.

Author(s)

Alexander R. Groos (<alexander.groos@giub.unibe.ch>)

See Also

ud.convert

Examples

```
# Load the provided RasterLayer objects as exemplary
# input for the function
data(airTemperature_30m_daily, precip_30m_daily,
    package = "glacierSMBM")
# Individual RasterLayer objects should be loaded or
# created using the function raster()

# Convert air temperature from Kelvin to degree Celsius
output <- unitConv(airTemperature_30m_daily, "kelvin", "celsius")

# Plot output
plot(airTemperature_30m_daily, main = "Air Temperature [Kelvin]",
    legend.args=list(text='Temperature (K)', side=3, line=1.5))
plot(output, main = "Air Temperature [Celsius]",
    legend.args=list(text='Temperature (Celsius)',
    side=3, line=1.5))

# Convert precipitation from m d-1 to mm h-1
output <- unitConv(precip_30m_daily, "m/d", "mm/h")
```

```
# Plot output
plot(precip_30m_daily, main = "Precipitation [m/d]",
    legend.args=list(text='Precipitation (m/d)',
    side=3, line=1.5))
plot(output, main = "Precipitation [mm/h]",
    legend.args=list(text='Precipitation (mm/h)',
    side=3, line=1.5))
```

Index