Package 'sarp.snowprofile.pyface'

August 26, 2025

Version 0.4.3

Date 2025-08-26

Title 'python' Modules from Snowpack and Avalanche Research

Description The development of post-processing functionality for simulated snow profiles by the snow and avalanche community is often done in 'python'. This package aims to make some of these tools accessible to 'R' users. Currently integrated modules contain functions to calculate dry snow layer instabilities in support of avalache hazard assessments following the publications of Richter, Schweizer, Rotach, and Van Herwijnen (2019) <doi:10.5194/tc-13-3353-2019>, and Mayer, Van Herwijnen, Techel, and Schweizer (2022) <doi:10.5194/tc-2022-34>.

URL http://www.avalancheresearch.ca

License CC BY-SA 4.0

Encoding UTF-8

RoxygenNote 7.3.2

Language en-CA

Depends sarp.snowprofile (>= 1.3.0)

Imports utils, reticulate, data.table

Config/reticulate list(packages = list(list(package = ``scikit-learn''), list(package = ``joblib''), list(package = ``numpy''), list(package = ``pandas'')))

NeedsCompilation no

Author Florian Herla [aut, cre],
Stephanie Mayer [aut],
SFU Avalanche Research Program [fnd]

Maintainer Florian Herla <fherla@sfu.ca>

Repository CRAN

Date/Publication 2025-08-26 07:10:02 UTC

Contents

`computeCritCutLength` *Compute critical crack length*

Description

This function implements Bettina Richter's (2019) parametrization for the critical crack length for flat simulations based on density, grain size, and shear strength. The parametrization also needs the mean density of the slab, which can be computed automatically if a snowprofile object is provided. In case the functions gets a snowprofileLayers object it expects `slab_rho` being precomputed. This acts as a safety mechanism to ensure that `slab_rho` is computed over one profile and not over a stacked layers data.frame containing multiple profiles. Note that the critical crack length can be computed alongside the layer probabilities for instability `p_unstable` in computePunstable.

Usage

```
computeCritCutLength(x)

## S3 method for class 'snowprofileSet'
computeCritCutLength(x)

## S3 method for class 'snowprofile'
computeCritCutLength(x)

## S3 method for class 'snowprofileLayers'
computeCritCutLength(x)
```

Arguments

x	sarp.snowprofile::snowprofileSet, sarp.snowprofile::snowprofile, or sarp.snowprofile::snowprofileLayers object

Value

Input object is returned with `$crit_cut_length` (and potentially `$slab_rho`) appended to the layers object.

Methods (by class)

- `computeCritCutLength(snowprofileSet)`: for sarp.snowprofile::snowprofileSets
- `computeCritCutLength(snowprofile)`: for sarp.snowprofile::snowprofiles
- `computeCritCutLength(snowprofileLayers)`: for sarp.snowprofile::snowprofileLayers

Author(s)

fherla based on the python function by smayer and brichter

References

Richter, B., Schweizer, J., Rotach, M. W., & Van Herwijnen, A. (2019). Validating modeled critical crack length for crack propagation in the snow cover model SNOWPACK. The Cryosphere, 13(12), 3353–3366. https://doi.org/10.5194/tc-13-3353-2019

See Also

computePunstable

`computePunstable` *Compute probability of layer instability based on random forest model*

Description

This function enables comfortable and fast R access to Stephanie Mayer's python implementation of her random forest model to estimate the probability of dry snow layer instability. The routine can be run very efficiently on large sarp.snowprofile::snowprofileSets. Layer properties required are sphericity, viscous deformation rate (10e-6 s-1), density (kg m-3), grain size (mm), and the critical crack length (m) (which can be computed very efficiently automatically if shear strength (kPA) is available.) Additionally, skier penetration depth in (m) is required.

Usage

```
computePunstable(x, ...)

## S3 method for class 'snowprofileSet'
computePunstable(
  x,
  ski_pen = NA,
  recompute_crit_cut_length = TRUE,
  buffer = TRUE,
  ...
)

## S3 method for class 'snowprofile'
computePunstable(x, ski_pen = NA, recompute_crit_cut_length = TRUE, ...)
```

```
## S3 method for class 'snowprofileLayers'
computePunstable(x, ski_pen = NA, ...)
```

Arguments

x	sarp.snowprofile::snowprofile, sarp.snowprofile::snowprofileSet, or sarp.snowprofile::snowprofileLayers
...	passed on to subsequent methods
ski_pen	skier penetration depth (m), one scalar for each profile in x
recompute_crit_cut_length	This routine can very efficiently compute the critical crack length with computeCritCutLength. SNOWPACK often provides NA values of the critical crack length even for layers that have a real solution to it. With this flag you can conveniently recompute all critical crack lengths (TRUE). If set to FALSE, it will only be computed if not all profiles already contain it. Note that shear strength must be available to compute the critical crack length!
buffer	internal switch to ensure fast computation at low memory cost. Leave at TRUE!

Value

x is returned with $p_unstable (and potentially $crit_cut_length, $slab_rho, and slab_rhogs) appended to each profile's layers object.

Methods (by class)

- computePunstable(snowprofileSet): for sarp.snowprofile::snowprofileSets
- computePunstable(snowprofile): for sarp.snowprofile::snowprofiles
- computePunstable(snowprofileLayers): for sarp.snowprofile::snowprofileLayers

Author(s)

fherla and smayer

References

Mayer, S., Herwijnen, A. Van, Techel, F., & Schweizer, J. (accepted, 2022). A random forest model to assess snow instability from simulated snow stratigraphy. The Cryosphere Discussions. https://doi.org/10.5194/tc-2022-34

Examples

```
# The following example cannot be run on CRAN:
on_cran <- TRUE
if (!on_cran && have_dependencies()) {

## --- Example ---
## load a handful of example profiles from a PRO file
profiles <- snowprofilePro(system.file("extdata/snowprofile.pro",
                                       package = "sarp.snowprofile.pyface"),
```

```
                              remove_soil = TRUE, suppressWarnings = TRUE)
summary(profiles)
names(profiles[[1]]$layers)
## compute p_unstable alongside critical crack length, slab_rho, slab_rhogs:
profiles <- computePunstable(profiles)
names(profiles[[1]]$layers)

}  # END dependency and cran clause
```

`have_dependencies` *Check whether python and dependencies are available on system*

Description

Check whether python and dependencies are available on system

Usage

```
have_dependencies()
```

Value

boolean TRUE/ FALSE

`import_pyunstable` *Import python module pyunstable*

Description

Convenience wrapper to make python module pyunstable accessible in R session

Usage

```
import_pyunstable()
```

Value

attach python module `pyunstable` to use python functions therein

See Also

computePunstable

`import_RFmodel` *Make RFmodel available for direct python calls*

Description

Convenience wrapper to make the python random forest model for snow layer instability 'p_unstable' accessible in R session

Usage

```
import_RFmodel()
```

Value

attach python RandomForestClassifier to variable `RFmodel`

See Also

computePunstable

Index