Package 'catmodeling'

July 23, 2026

Type Package

Title Catastrophe Model Simulation and Adjustment

Version 0.0.1

Maintainer Stephen Jewson <stephen.jewson@gmail.com>

Description Manipulation of catastrophe model outputs, including
tasks such as simulating year loss tables (YLTs) from event loss tables (ELTs),
adjusting the frequencies of events in YLTs to create new YLTs,
applying catastrophe exceedance of loss contracts (catXL),
applying hours clauses,
and calculating diagnostics from ELTs and YLTs, such as average annual loss
and exceedance probability curves. Frequency adjustment routines are
based on the paper ``A new simulation algorithm for
more precise estimates of change in catastrophe risk models, with application to
hurricanes and climate change", Jewson, S. (2023); <doi:10.1007/s00477-023-02409-0>.
Uses the compiled language 'Rust' in places and so requires
a Rust compiler to be installed, or a binary installation.

Imports stats, dplyr, utils, pracma, geosphere

Depends R (>= 4.2)

License AGPL-3

BugReports https://github.com/stephenjewson/catmodeling/issues

URL https://www.catmodeling.info

Encoding UTF-8

RoxygenNote 7.3.3

Config/rextendr/version 0.5.0

SystemRequirements Cargo (Rust's package manager), rustc >= 1.65.0, xz

NeedsCompilation yes

Author Stephen Jewson [aut, cre] (ORCID:
<https://orcid.org/0000-0002-6011-6262>)

Repository CRAN

Date/Publication 2026-07-23 12:00:02 UTC

Contents

addvectors . . . 3
bootstrap_ep_uncertainty . . . 4
calc_beta_params . . . 5
catxl . . . 6
compare_distance_routines . . . 8
defineregion . . . 9
elt_add_nahu_cat . . . 10
elt_collapse_regions . . . 11
elt_diagnostics_aaeaal . . . 12
elt_diagnostics_aaeaal_by_cat . . . 13
elt_diagnostics_aaeaal_with_adj . . . 14
elt_diagnostics_epcurves . . . 15
elt_diagnostics_epcurves_with_adj . . . 16
elt_rate_adjustments_cat_2_event . . . 18
elt_rate_adjustments_regcat_2_event . . . 19
filetest . . . 20
hours_clause . . . 21
hours_clause_apply_part2 . . . 23
hours_clause_wrapper_ylt . . . 25
hours_clause_write_settings . . . 27
imports . . . 28
input_checks . . . 28
make_2_ep_by_sorting . . . 29
make_elt . . . 30
make_ep_by_sorting . . . 31
make_ylt . . . 32
nahu_define_gmst_scenarios . . . 33
nahu_define_k2020_zenodo_landfall_adj . . . 34
nahu_k2020_rates_interpolation . . . 35
nahu_write_settings . . . 36
nahu_ylt_surgery . . . 37
rust_dist_haversine . . . 43
rust_hello_world . . . 44
rust_hours_clause_apply_part2 . . . 45
rust_square . . . 46
stopi . . . 46
template . . . 47
whatreg . . . 48
writeNresults2csv . . . 49
ws2catf . . . 51
yltsim . . . 52
yltsim_inc . . . 54
yltsurgery . . . 56
yltsurgery_groups . . . 58
ylt_add_cat_2_longylt . . . 60
ylt_add_region_2_longylt . . . 61

ylt_check_for_2_events_on_same_day 62
ylt_check_for_same_events_in_one_year 63
ylt_diagnostics_aaeaal 64
ylt_diagnostics_aaeaal_by_catf 65
ylt_diagnostics_aaeaal_by_cat_change_printf 66
ylt_diagnostics_aaeaal_by_cat_printf 67
ylt_diagnostics_aaeaal_change_printf 69
ylt_diagnostics_aaeaal_printf 70
ylt_diagnostics_loss_thresholds 71
ylt_diagnostics_nchanges 72
ylt_long2short 73
ylt_rate_adjustments_catreg_2_row 74
ylt_rate_adjustments_cat_2_row 76
ylt_write2results2csv 77

Index **79**

`addvectors` *Adds two vectors element-wise up to length n.*

Description

Adds two vectors element-wise up to length n.

Usage

```
addvectors(x, y, n)
```

Arguments

`x`	A reference to the first input vector (slice of f64)
`y`	A reference to the second input vector (slice of f64)
`n`	The number of elements to process

Value

The sum of the two input vectors, as a vector.

bootstrap_ep_uncertainty *Bootstrap EP uncertainty*

Description

Reads in a vector of annual losses, bootstraps, and calculates the standard deviation of losses at a list of return levels

Usage

```
bootstrap_ep_uncertainty(losses, nbs, rps)
```

Arguments

losses	A vector of losses
nbs	The number of bootstrap resamples required
rps	The return periods required

Details

Can be used with make_ep_by_sorting, for the basic EP curve. Obviously using more bootstrap samples is better, but slower.

Value

A vector of standard deviations, one for each return level

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

Examples

```
#
# 1 create losses
#
message("1. make losses")
losses=seq(1,1000,1)
#
# 2 set the return periods to look at
#
rps=c(2,3,4,5,10,20,50,100)
#
#
message("2. calculate the losses at the given return levels")
ep=make_ep_by_sorting(losses,rps)
print(ep)
```

```
#
message("3. calculate the sd of the uncertainty around those losses")
epsd=bootstrap_ep_uncertainty(losses,nbs=100,rps)
#
print(epsd)
```

`calc_beta_params` *Calculate Beta Distribution Parameters for a List of Events*

Description

Calculates beta distribution parameters from by-event values of mean, standard deviation and exposure.

Usage

```
calc_beta_params(mloss, sloss, expo)
```

Arguments

`mloss`	mean loss by event
`sloss`	sd loss by event
`expo`	exposure by event

Details

Maybe I should change this one at some point so that it reads in an ELT, rather than just separate variables. But for now it reads 3 separate variables.

Value

A list containing beta distribution alpha and beta parameters, by event.

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

Examples

```
#
# create inputs
#
mloss=seq(100,300,100)
sloss=seq(100,300,100)
expo=rep(1000,3)
#
params=calc_beta_params(mloss,sloss,expo)
#
```

```
print(params$eventalpha)
print(params$eventbeta)
```

`catxl` *Applies CatXL Layers to a YLT*

Description

Reads a YLT generated by `yltsim` format and applies CatXL layers, to generate multiple outputs and diagnostics.

Usage

```
catxl(longylt, limit, deductible, nrst, premium, rst_premium_pc)
```

Arguments

`longylt`	A longylt in `yltsim` format
`limit`	A vector of limits for the CatXL layers.
`deductible`	A vector of deductibles for the CatXL layers.
`nrst`	A vector giving the numbers of reinstatements in each layer
`premium`	The overall premium (can be set to 0)
`rst_premium_pc`	The reinstatement premium, as a percentage

Details

Based on the description of catXL pricing given at:

Value

A list containing 3 data frames:

- `summary`AAL ann average annual premium for each layer
- `shortrecord`Diagnostics by year by layer
- `longrecord`Diagnostics by event (aka claim) by layer

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

See Also

`yltsim`, to generate the required YLT input

Examples

```
#
# 1 create an example ELT
#
nevents=1000
totalannualrate=3
evid=c(1:nevents)
mloss=(c(1:nevents)^2)/100 #max is 10k
mrate=rep(totalannualrate/nevents,nevents)
elt=data.frame(evid,mrate,mloss)
#
# 2 example 1
#
message("****************************************************************")
message("Example 1: 2 years of simulation, 2 layers, detailed outputs  ")
message("****************************************************************")
#
# 3 simulate a 2 year YLT
#
nyears=2
set.seed(3)
ylt=yltsim(nyears=nyears,elt)
#
# 4 set up 2 catXL layers
#
limit=c(2000,4000)
deductible=c(1000,3000)
nrst=c(2,2)
premium=1000
rst_premium_pc=c(80,90)
#
# 5 analyze the catXL
#
catxlresults=catxl(ylt,limit,deductible,nrst,premium,rst_premium_pc)
#
# 6 look at the results
#
message("\nsummary:")
print(catxlresults$summary)
message("\nshortrecord:")
print(catxlresults$shortrecord)
message("\nlongrecord:")
print(catxlresults$longrecord)
#
# 7 example 2
#
message("****************************************************************")
message("Example 2: 1k years of simulation, 1 layer, summary outputs")
message("****************************************************************")
nyears=100
#
# 8 simulate 2x 1k YLTs to investigate convergence
```

```
#
set.seed(1);ylt1=yltsim(nyears=nyears,elt)
set.seed(2);ylt2=yltsim(nyears=nyears,elt)
#
# 9 set up 1 catXL layer this time
#
limit=c(2000)
deductible=c(1000)
nrst=c(1)
premium=1000
rst_premium_pc=c(80)
#
# 10 analyze the catXL and look at the summary results
#
message("\nylt1 summary:")
print(catxl(ylt1,limit,deductible,nrst,premium,rst_premium_pc)$summary)
message("\nylt2 summary:")
print(catxl(ylt2,limit,deductible,nrst,premium,rst_premium_pc)$summary)
```

`compare_distance_routines` *Function for testing that R and Rust give the same distances between events*

Description

This isn't actually used for anything, except doing this test. But it's an important test. Getting R and Rust to agree was difficult.

Usage

```
compare_distance_routines(longylt)
```

Arguments

`longylt`	Input YLT

Value

Writes mismatches to the screen. Always stops.

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

Examples

```
#
# 1 make longylt
#
cat("1. make longylt:\n")
year=c(1,1,1)
lon=c(-111.8817,-111.1963,-119.891)
lat=c(35.1067,31.4196,39.3254)
longylt=data.frame(year,lon,lat)
#
# 2 run
#
compare_distance_routines(longylt)
```

`defineregion` *Defines the 19 Global Regions from My BAMS Paper*

Description

Defines the 19 Global Regions from My BAMS Paper

Usage

```
defineregion(iregion)
```

Arguments

`iregion`	Region index

Value

A list with two vectors, for longitudes and latitudes of corners of the region

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

Examples

```
reg=defineregion(2)
cat("regx=",reg$regx,"\n")
cat("regy=",reg$regy,"\n")
```

`elt_add_nahu_cat` *Adds a cat column to an elt, calculated from wind speed.*

Description

Takes an ELT, calculates NAHU cat from windspeed, and adds a column with cat. The ELT has to contain windspeed to start with.

Usage

```
elt_add_nahu_cat(elt, units)
```

Arguments

`elt`	A data frame containing the elt. Requires `wspd`.
`units`	must be `knots`, or `mph`.

Details

I've used the Wikipedia, and filled gaps with halves.

Value

A new elt with a cat column added, where cat runs from -1 to 5.

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

Examples

```
#
# create an elt with just wind speed
#
wspd=seq(10,150,10)
elt1=data.frame(wspd)
#
# call
#
elt2=elt_add_nahu_cat(elt1,units="knots")
#
print(elt2)
```

`elt_collapse_regions` *For an ELT with multiple occurrences of each event, collapses to 1 per event*

Description

Takes in an ELT that might have multiple occurrences of each event, representing different regions, and combines them together to give one occurrence of that event. Assumes that the occurrences of each event are consecutive. Processes other columns cleverly, depending on what they contain. For instance, adds up losses and exposures, counts regions.

Usage

```
elt_collapse_regions(elt1, columns2copy = NULL, combine = TRUE, verbose = TRUE)
```

Arguments

`elt1`	A data frame containing the ELT. The ELT must contain `evid`.
`columns2copy`	Copies these columns into the output
`combine`	If false, then just copies the first set of values
`verbose`	Logical for verbose or not

Details

Recognizes and processes the following columns if they exist: `mloss` (sums them up), `sloss` (assumes independent), `expo` (sums them up), `mrate` (copies the first one). `lfreg` (counts them). For other specified columns, copies the first value for each event. How to combine `sloss` though? Right now I'm using independence. But presumably I should use correlation of secondary uncertainty.

Value

A new ELT with just single occurrence of each event

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

Examples

```
#
# create an elt
#
evid=c(1,2,2,3,3,3,4,4,4,4) #evid is required
#mloss is not required, but is processed intelligently
mloss=rep(1,10)
#jim is not required, but is copied because we specify that
jim=c(1000:1009)
bob=c(2000:2009)
```

```
#
elt1=data.frame(evid,mloss,jim,bob)
elt2=elt_collapse_regions(elt1,columns2copy=c("jim"),verbose=TRUE)
#
message("elt1=\n")
print(elt1)
message("elt2=\n")
print(elt2)
```

`elt_diagnostics_aaeaal` *Calculates AAE and AAL from an ELT*

Description

Calculates AAE and AAL from an ELT

Usage

```
elt_diagnostics_aaeaal(elt)
```

Arguments

`elt`	A data frame containing the ELT. The ELT must contain `mrate` and `mloss` columns. See the example for how to make the ELT.

Value

AAE and AAL. EP curves are in a separate routine `epfromelt3`.

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

Examples

```
#
# create an elt
#
mrate=seq(0.1,0.3,0.1)
mloss=seq(100,300,100)
elt=data.frame(mrate,mloss)
#
op=elt_diagnostics_aaeaal(elt)
#
cat("AAE=",op$AAE,"\n")
cat("AAL=",op$AAL,"\n")
```

elt_diagnostics_aaeaal_by_cat *Calculates AAE and AAL from an ELT, and by cat*

Description

Calculates AAE and AAL from an ELT, and AAE and AAL by cat. There can be any number of cats, labelled using integers. So this routine is not only applicable to hurricane: the cat can indicate anything, for any peril.

Usage

```
elt_diagnostics_aaeaal_by_cat(elt)
```

Arguments

elt	A data frame containing the ELT. Must contain mrate, mloss and cat columns. See the example for how to make the ELT.

Value

A list containing AAE and AAL, and then AAE and AAL by cat.

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

Examples

```
#
# create an elt with cat
#
mrate=seq(0.1,0.5,0.1)
mloss=seq(100,500,100)
cat=c(1,1,1,2,2)
elt=data.frame(mrate,mloss,cat)
#
op1=elt_diagnostics_aaeaal(elt)
op2=elt_diagnostics_aaeaal_by_cat(elt)
#
cat("AAE=",op1$AAE,"\n")
cat("AAL=",op1$AAL,"\n")
cat("AAEbycat=",op2$AAEbycat,"\n")
cat("AALbycat=",op2$AALbycat,"\n")
cat("AALbycatpc=",op2$AALbycatpc,"\n")
```

`elt_diagnostics_aaeaal_with_adj` *Calculates AAE and AAL from an ELT*

Description

Calculates AAE and AAL from an ELT, taking into account an input matrix of stochastic parameter adjustments by cat and event. It doesn't produce an adjusted ELT, just gives the AAE and AAL you would get. It could, for instance, be used to test code that produces adjusted YLTs.

Usage

```
elt_diagnostics_aaeaal_with_adj(elt, adjustmentsbycatevent)
```

Arguments

`elt`	A data frame containing the ELT. The ELT must contain `mrate`, `mloss` and `cat` columns.
`adjustmentsbycatevent`	A matrix with rate adjustments (cat 1-ncat, event) An adjustment of 1 means no change. Copes with any number of cats, set by the first dimension of this matrix.

Details

For historical reasons the adjustments are stored in a matrix by cat-event, not just by event, which is a bit inefficient. I could change that at some point.

Value

AAE and AAL.

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

Examples

```
#
# create an elt
#
mrate=seq(0.1,0.5,0.1)
mloss=seq(100,500,100)
cat=c(1,1,1,2,2)
elt=data.frame(mrate,mloss,cat)
#
# make the rate adjustments (two sets, for comparison)
#
adjustmentsbycatevent1=matrix(1,7,5)
adjustmentsbycatevent2=matrix(2,7,5)
```

```
#
# calculate the AAE and AAL 3 ways
#
op0=elt_diagnostics_aaeaal(elt)
op1=elt_diagnostics_aaeaal_with_adj(elt,adjustmentsbycatevent1)
op2=elt_diagnostics_aaeaal_with_adj(elt,adjustmentsbycatevent2)
#
cat("no adjustments (from the basic elt_diagnostics routine, for comparison):\n")
cat(" AAE=",op0$AAE,"\n")
cat(" AAL=",op0$AAL,"\n")
cat("no adjustments (from this routine, but with adjustments all set to 1):\n")
cat(" AAE=",op1$AAE,"\n")
cat(" AAL=",op1$AAL,"\n")
cat("with adjustments (this routine, with some actual adjustments):\n")
cat(" AAE=",op2$AAE,"\n")
cat(" AAL=",op2$AAL,"\n")
```

`elt_diagnostics_epcurves`

Converts ELT rates into plotting positions for CEP, EEF and OEP

Description

Converts ELT rates into plotting positions for CEP, EEF and OEP

Usage

```
elt_diagnostics_epcurves(elt)
```

Arguments

`elt`	needs `mrate` only

Details

One could debate what plotting positions to use, as always. This one uses Hazen. Perhaps Weibull would be better.

Value

A list containing plotting positions for CEP, EEF and OEP

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

Examples

```
#
# 1 make elt
#
nx=100
mrate=runif(nx)
elt=data.frame(mrate)
#
# 2 make the curves
#
op=elt_diagnostics_epcurves(elt)
#
# 3 add some loss data
#
loss=sort(rnorm(nx))
#
# 4 plot
#
old_par <- par(no.readonly = TRUE)
par(old_par)
par(mfrow=c(2,2))
#
plot(loss,op$cep,main="CEP")
lines(loss,op$cep,col="red")
#
plot(loss,op$eef,main="EEF")
lines(loss,op$eef,col="red")
#
plot(loss,op$oep,main="OEP")
lines(loss,op$oep,col="red")
#
par(old_par)
```

`elt_diagnostics_epcurves_with_adj`

Converts ELT rates into plotting positions for CEP, EEF and OEP, with rate adjustments

---

Description

Takes an ELT, and takes adjustments by cat and *year*, and makes curves, without simulating, but using very clever analytical expressions. Supports any number of cats (so not just for hurricane). So generalizes `elt_diagnostics_epcurves`.

Usage

```
elt_diagnostics_epcurves_with_adj(elt, adjustmentsbycatyear)
```

Arguments

elt	needs mrate and cat
adjustmentsbycatyear	the adjustments

Details

One could debate what plotting positions to use, as always. This one uses Hazen. Perhaps Weibull would be better.

Value

A list containing plotting positions for CEP, EEF and OEP

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

Examples

```
#
# 1 make elt with mrate and cat columns
#
nevents=100
mrate=runif(nevents)
cat=sample(7,nevents,replace=TRUE)
elt=data.frame(mrate,cat)
#
# 2 make some adjustments by catyear
# -value of 1 does nothing
#
nyears=1000
ncats=7
adjustmentsbycatyear=matrix(2,ncats,nyears)
#
# 2 make the curves
#
op0=elt_diagnostics_epcurves(elt)
op1=elt_diagnostics_epcurves_with_adj(elt,adjustmentsbycatyear)
#
# 3 add some loss data
#
loss=sort(rnorm(nevents))
#
# 4 plot unadjusted
#
old_par <- par(no.readonly = TRUE)
par(mfrow=c(2,3))
#
plot(loss,op0$cep,main="CEP")
lines(loss,op0$cep,col="red")
```

```
#
plot(loss,op0$eef,main="EEF")
lines(loss,op0$eef,col="red")
#
plot(loss,op0$oep,main="OEP")
lines(loss,op0$oep,col="red")
#
#
# 4 plot adjusted (only the EEF changes because the adjustments are constant)
#
plot(loss,op1$cep,main="CEP")
lines(loss,op1$cep,col="red")
#
plot(loss,op1$eef,main="EEF")
lines(loss,op1$eef,col="red")
#
plot(loss,op1$oep,main="OEP")
lines(loss,op1$oep,col="red")
#
par(old_par)
```

`elt_rate_adjustments_cat_2_event` *Converts rate adjustments from cat to event, for events in an ELT*

Description

Converts rate adjustments by cat to rate adjustments by event. So that you can specify rate adjustments by cat (which is easy to do), and then run this routine to assign those adjustments to all the events in an ELT.

Usage

```
elt_rate_adjustments_cat_2_event(elt, rate_adjustments_by_cat)
```

Arguments

`elt`	A data frame containing the ELT. Requires `cat`.
`rate_adjustments_by_cat`	A list of two vectors containing the mean and sd adjustments by cat, given by `adjustments_by_cat$mn`, `adjustments_by_cat$sd`

Details

Doesn't do anything except just copy the adjustments. No calculations. Cat is a positive integer.

Value

A matrix with the adjustments by event column 1 is the mean of the adjustments. column 2 is the sd of the adjustments.

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

Examples

```
#
# create an elt
#
cat=c(1,2,2,3,3,3) #only cat is required
elt=data.frame(cat)
#
# make the adjustments by cat
#
rate_adjustments_by_cat=list(mn=c(2,4,6),sd=c(0,0,0))
#
# call
#
rate_adjustments_by_event=elt_rate_adjustments_cat_2_event(elt,rate_adjustments_by_cat)
#
print(elt)
print(rate_adjustments_by_event)
```

---

elt_rate_adjustments_regcat_2_event

Converts rate adjustments from region and cat to event, for events in an ELT

Description

Converts rate adjustments by region and cat to rate adjustments by event. So that you can specify rate adjustments by region and cat (which is easy to do), and then run this routine to assign those adjustments to all the events in an ELT.

Usage

```
elt_rate_adjustments_regcat_2_event(elt, rate_adjustments_by_regcat)
```

Arguments

`elt`	A data frame containing the ELT. Requires `cat` and `reg`
`rate_adjustments_by_regcat`	A list of two matrices containing the mean and sd adjustments by reg-cat, given by `adjustments_by_cat$mn`, `adjustments_by_cat$sd`

Details

Doesn't do anything except just copy the adjustments. No calculations. Cat is a positive integer.

Value

A matrix with the adjustments by event column 1 is the mean of the adjustments. column 2 is the sd of the adjustments.

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

Examples

```
#
# create an elt
#
cat_by_event=c(1,2,2,3,3,3)
reg_by_event=c(1,2,2,1,2,2)
elt=data.frame(cat=cat_by_event,region=reg_by_event)
#
# make the adjustments by reg-cat
# (2 regions, 3 cats)
#
mean=matrix(1,2,3)
sd=matrix(0,2,3)
rate_adjustments_by_regcat=list(mn=mean,sd=sd)
#
# call
#
rate_adjustments_by_event=elt_rate_adjustments_regcat_2_event(elt,rate_adjustments_by_regcat)
#
print(elt)
print(rate_adjustments_by_event)
```

---

filetest *Testing using files*

Description

This is just a test function to illustrate how to open files in functions inside a package.

Usage

```
filetest(filename)
```

Arguments

filename	the filename

Details

The answer to: how can I open a file in a function in a R package, and test that using an example?

Value

an integer read in from the file

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

Examples

```
#
filename=tempfile(fileext=".csv")
#
int=3
#
write.csv(int,file=filename,row.names=FALSE)
#
int2=filetest(filename)
#
message("temporary file name = ",filename)
message("before = ",int," and after = ",int2[1,1])
```

---

hours_clause *Applies an hours clause to a long YLT, and returns a YLT*

Description

I could simplify the code by just returning a long YLT, and then leaving the user to create a short YLT using long2short.

Usage

```
hours_clause(
  longylt1,
  hcdays,
  hckm,
  byregion = FALSE,
  rust,
  rrrr,
  verbose = FALSE
)
```

Arguments

longylt1	The input long YLT
hcdays	Hours clause parameter, number of days
hckm	Hours clause parameter, distance in km
byregion	Whether to use regions, or distances, to measure closeness
rust	Rust logical
rrrr	R logical
verbose	Logical, meaning obvious

Details

Can be called from hours_clause_wrapper, which manages the file i/o. Note that exactly what 'applying an hours clause' means some explanation. We don't have access to footprints, just days of the year (one day per event) and locations (one point per event). The algorithm looks at the days and the locations and merges events which are close in both time and space. It only applies a maximum of one hours clause per year, which is chosen to give the maximum possible event loss (i.e., the largest possible max loss recovery for the cedant).

Value

A long YLT with hours clause applied.

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

Examples

```
#
# 1 set the hours clause settings
#
hcdays=15
hckm=200
#
# 2 make an 8 year ylt with 10 events
#
cat("1. make longylt:\n")
nyearsinylt=8
year =c(1 ,1 ,1 ,2 ,4 ,5 ,6 ,7 ,8 ,8)
day =c(1 ,2 ,51 ,101,151,201,251,300,350,360)
evid =c(1 ,2 ,3 ,4 ,5 ,6 ,7 ,8 ,9 ,10)
lat =c(10 ,11 ,30 ,30 ,30 ,30 ,30 ,30 ,30 ,50)
lon =c(10 ,11 ,30 ,30 ,30 ,30 ,30 ,30 ,30 ,50)
loss =seq(10,55,5)
longylt1=data.frame(year,day,evid,lat,lon,loss)
#
# 3 apply the hours clause and generate a new ylt
#
longylt2=hours_clause(longylt1,hcdays,hckm,rust=FALSE,rrrr=TRUE,verbose=TRUE)
```

```
#
print(head(longylt1,n=10))
print(head(longylt2,n=10))
#
```

`hours_clause_apply_part2` *Function that loops through events in one year and applies an hours clause*

Description

Quite hard to understand. Having previously found out which value of k gives the hours clause with the largest loss, this routine applies the hours clause starting with that event. It also registers which events have been moved, which part 1 does not do. There are two separate, but very similar, routines, just to make the search go faster, since in the initial search, there is no need to register which events have been moved.

Usage

```
hours_clause_apply_part2(
  hcdays,
  hckm,
  byregion = FALSE,
  temploss,
  thisyearday,
  thisyearlon,
  thisyearlat,
  thisyearregion,
  nevents_in_year,
  k
)
```

Arguments

`hcdays`	Hours clause parameter, number of days
`hckm`	Hours clause parameter, distance in km
`byregion`	Whether to use regions, or distances, to measure closeness
`temploss`	Not sure. Part of the algorithm.
`thisyearday`	For events in this year, the days
`thisyearlon`	For events in this year, the lons
`thisyearlat`	For events in this year, the lats
`thisyearregion`	For events in this year, the regions
`nevents_in_year`	Number of events in this year
`k`	The index of the hours clause to apply, of all possible hours clauses

Details

To be called from hours_clause, which loops thru the whole YLT, and which itself is called from hours_clause_wrapper, which manages the file i/o.

Value

A loss and a counter

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

References

x

Examples

```
#
# 1 set the hours clause settings
#
hcdays=15
hckm=200
#
# 2 make a single year of a ylt
#
cat("1. make longylt:\n")
nyearsinylt=8
nevents_in_year1=10
thisyearloss =seq(10,55,5)
thisyearday =c(1 ,2 ,51 ,101,151,201,251,300,350,360)
thisyearlat =c(10 ,11 ,30 ,30 ,30 ,30 ,30 ,30 ,30 ,50)
thisyearlon =c(10 ,11 ,30 ,30 ,30 ,30 ,30 ,30 ,30 ,50)
thisyearregion=rep("A",nevents_in_year1)
#
# 3 apply the hours clause routine to position 1
#
cat("1. apply the hours clause calculator:\n")
output=hours_clause_apply_part2(hcdays,hckm,byregion=FALSE,thisyearloss,
thisyearday,thisyearlon,thisyearlat,thisyearregion,
nevents_in_year1,k=1)
#
# 4 look at the output
#
cat("losses=",output$temploss,"\n")
cat("hccounter=",output$hccounter,"\n")
```

`hours_clause_wrapper_ylt` *Applies an hours clause to a long YLT stored in csv file, and returns the adjusted YLT in memory and to a csv file*

Description

Can be called from `800_hours_clause_wrapper_example`.

Usage

```
hours_clause_wrapper_ylt(
  ipfilename,
  settingsfilename,
  opfilename,
  nyearsinylt,
  hcdays,
  hckm,
  byregion,
  rust,
  rrrr,
  test = FALSE,
  verbose = FALSE
)
```

Arguments

`ipfilename`	Input filename containing the input long YLT
`settingsfilename`	Output filename for the settings output
`opfilename`	Output filename containing the output YLT
`nyearsinylt`	Number of years in the input YLT
`hcdays`	Hours clause parameter, number of days
`hckm`	Hours clause parameter, distance in km
`byregion`	Whether to use regions, or distances, to measure closeness
`rust`	Rust flag
`rrrr`	R flag
`test`	Logical, not sure what it does
`verbose`	Logical, meaning obvious

Details

Just a pretty dumb wrapper, that writes out the settings, reads the input file, checks it, looks at some diagnostics, calls the hours clause routine to apply the hours clause, looks at some diagnostics, writes out the results and returns the results.

Value

A data frame containing the original long YLT, and the adjusted YLTs.

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

Examples

```
#
# 1 make filenames
#
ipfilename=tempfile(fileext=".csv")
opfilename=tempfile(fileext=".csv")
settingsfilename=tempfile(fileext=".csv")
#
# 2 set the hours clause settings
#
hcdays=15
hckm=200
#
# 3 make an 8 year ylt with 10 events
#
cat("1. make longylt:\n")
nyearsinylt=8
year =c(1 ,1 ,1 ,2 ,4 ,5 ,6 ,7 ,8 ,8)
day =c(1 ,2 ,51 ,101,151,201,251,300,350,360)
evid =c(1 ,2 ,3 ,4 ,5 ,6 ,7 ,8 ,9 ,10)
lat =c(10 ,11 ,30 ,30 ,30 ,30 ,30 ,30 ,30 ,50)
lon =c(10 ,11 ,30 ,30 ,30 ,30 ,30 ,30 ,30 ,50)
region=c("A" ,"A" ,"B" ,"B" ,"B" ,"B" ,"B" ,"B" ,"B" ,"B")
loss =seq(10,55,5)
longylt=data.frame(year,day,evid,lat,lon,loss,region)
write.csv(longylt,file=ipfilename,row.names=FALSE)
#
# 4 apply the hours clause and generate a new ylt
#
ylts=hours_clause_wrapper_ylt(ipfilename,settingsfilename,opfilename,nyearsinylt,
hcdays,hckm,
byregion=TRUE,
rust=TRUE,rrrr=FALSE,
test=FALSE,verbose=TRUE)
#
longylt1=ylts$longylt1
longylt2=ylts$longylt2
#
print(head(longylt1,n=6))
print(head(longylt2,n=6))
#
#
```

`hours_clause_write_settings` *When using* `hours_clause_wrapper`*, writes out the settings to a file.*

Description

Just for record keeping.

Usage

```
hours_clause_write_settings(
  ipfilename,
  settingsfilename,
  opfilename,
  time,
  distance
)
```

Arguments

`ipfilename`	Input filename containing the input long YLT
`settingsfilename`	Output filename for the settings output
`opfilename`	Output filename containing the input long YLT
`time`	Hours clause parameter, number of days
`distance`	Hours clause parameter, distance in km

Details

Just to provide a record of each analysis.

Value

Just writes to a file

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

References

x

Examples

```
#
# 1 make filenames
#
ipfilename=tempfile(fileext=".csv")
opfilename=tempfile(fileext=".csv")
settingsfilename=tempfile(fileext=".csv")
#
# 2 hc settings
#
time=1
distance=2
#
# 3 call the routine
#
hours_clause_write_settings(ipfilename,settingsfilename,opfilename,time,distance)
#
# 4 have a look at the file it produced
#
x=read.csv(settingsfilename)
print(x)
```

`imports` *Imports*

Description

Centralized imports

Usage

```
imports()
```

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

`input_checks` *Checks the columns in an input data frame.*

Description

A utility that checks the columns in the input data frame versus a list of required names

Usage

```
input_checks(df, names, location = FALSE)
```

Arguments

df	data frame to check
names	names to check
location	print where it's being called from

Value

x

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

Examples

```
#
# make an elt
#
set.seed(1)
evid=c(1,2,3)
mrate=seq(0.1,0.3,0.1)
mloss=seq(100,300,100)
elt=data.frame(evid,mrate,mloss)
print(elt)
#
# and check if it contains the two columns
#
input_checks(elt,c("evid","mrate","mloss"))
#
```

---

make_2_ep_by_sorting *Make 2 EP curves by sorting losses*

Description

Calculates 2 EPs, from two vectors of losses, just by calling make_ep_by_sorting twice. This somewhat trivial routine is included because it is a very common operation, to compare losses between two cases (two models, often an original model and an adjusted model).

Usage

```
make_2_ep_by_sorting(losses1, losses2, rps)
```

Arguments

losses1	A vector of losses.
losses2	A vector of losses.
rps	The return periods at which to calculate the EPs

Value

A matrix with 2 sets of return levels in it (with 2 rows)

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

See Also

- make_ep_by_sorting which is what this routine uses

Examples

```
#
# 1 create losses
#
cat("1. make losses:\n")
losses1=seq(1,1000,1)
losses2=seq(1001,2000,1)
#
# 2 set the return periods to look at
#
rps=c(1,2,3,4,5,10,20,50,100)
#
cat("2. calculate the EP:\n")
op=make_2_ep_by_sorting(losses1,losses2,rps)
#
print(op)
```

---

make_elt *Make an ELT in memory*

Description

This code does nothing at all. But the example shows how to make an ELT in memory.

Usage

```
make_elt()
```

Details

This code just exists as a way to present the example.

Value

Nothing.

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

Examples

```
#
# create an elt with 10 rows
#
set.seed(1)
#
# essentials
evid=seq(1000,1009,1)
mrate=rep(0.1,10)
mloss=seq(100,1000,100)
#
# add an optional column
wspd=seq(50,59,1)
#
# now make the elt
# the ordering of columns isn't important, but the following is standard
elt=data.frame(evid,mrate,mloss,wspd)
#
# and have a look at it, in different ways
print(elt)
head(elt,n=5)
```

---

make_ep_by_sorting *Make an EP curve by sorting losses*

Description

Reads in a vector of losses, sorts them, and picks out certain return level losses

Usage

```
make_ep_by_sorting(losses, rps)
```

Arguments

losses	A vector of annual losses
rps	A vector of return periods at which losses are required Must be 1 or greater.

Details

Assumes the input losses are one loss per year.

Value

Losses at the specified return periods, as integers.

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

Examples

```
#
# 1 create losses
#
cat("1. make losses:\n")
losses=seq(1,1000,1)
#
# 2 set the return periods to look at
#
rps=c(0.5,1,2,3,4,5,10,20,50,100,1000,10000)
#
cat("2. calculate the EP:\n")
op=make_ep_by_sorting(losses,rps)
#
print(op)
```

---

make_ylt *Make an YLT in memory*

Description

This code does nothing at all. But the example shows how to make a YLT in memory.

Usage

```
make_ylt()
```

Details

This code just exists as a way to present the example.

Value

Nothing.

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

Examples

```
#
# create an long ylt with 4 years and 10 events
#
set.seed(1)
#
# essentials, illustrating:
# -multiple events in one year
# -some events repeat, but could have different losses
yrid=c(1,1,1,2,2,2,3,3,4,4)
evid=c(1000,1001,1002,1003,1001,1004,1005,1000,1006,1007)
loss=seq(100,1000,100)
#
# now make the ylt
# column ordering is optional, but the following is standard
longylt=data.frame(yrid,evid,loss)
#
# and have a look at it, in different ways
print(longylt)
head(longylt,n=5)
```

---

nahu_define_gmst_scenarios

Defines 4 GMST Scenarios

Description

These are the 4 GMST scenarios for 2.6,...,8.5, by year, derived from CMIP (from the JAMC paper Jewson(2021)). From 1880 to 2100.

Usage

```
nahu_define_gmst_scenarios(verbose = FALSE)
```

Arguments

verbose	logical

Value

An array with 4 GMST scenarios.

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

Examples

```
gmst=nahu_define_gmst_scenarios()
print(head(gmst[,1:5]))
```

---

nahu_define_k2020_zenodo_landfall_adj
Define Jewson landfall rates Scenarios from the BAMS paper

Description

Multiplicative landfalling hurricane rate adjustments, mean and sd.

Usage

```
nahu_define_k2020_zenodo_landfall_adj()
```

Details

For the mean, a value of 1 is no change in the rate. For the sd, value of 0 is no uncertainty in the rate change. The distribution of rate changes is considered to be log-normal. Index 1, from 1 to 19, specifies the region, as in the BAMS paper. Index 2, specifies mean (1) or sd(2). Index 3, specifies what set of categories. The 7 categories are -1,0,1,2,3,4,5 on the SSHWS.

Value

An array with 19 landfall scenarios, described in the Jewson (2023) BAMS paper.

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

Examples

```
adjustments=nahu_define_k2020_zenodo_landfall_adj()
region=1
variance=1
cat=1
print(adjustments[1,1,1])
```

nahu_k2020_rates_interpolation *Interpolates rates in time*

Description

Longer description

Usage

```
nahu_k2020_rates_interpolation(
  rcp,
  baseyear1,
  baseyear2,
  targetyear,
  k2020settings,
  frequnc,
  quantile,
  randomseed,
  gmst,
  landfall,
  test = FALSE,
  verbose = FALSE,
  reflectinputs = FALSE
)
```

Arguments

rcp	rcp
baseyear1	baseyear1
baseyear2	baseyear2
targetyear	targetyear
k2020settings	k2020settings
frequnc	frequnc
quantile	quantile
randomseed	randomseed
gmst	gmst
landfall	landfall
test	test
verbose	verbose
reflectinputs	reflectinputs

Value

Mean rate, sd rate, and GMST change

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

`nahu_write_settings` *Writes out settings to file for* `nahu_ylt_surgery`

Description

Writes out settings to file for `nahu_ylt_surgery`

Usage

```
nahu_write_settings(
  ipfilename,
  settingsfilename,
  rcp,
  baseyear1,
  baseyear2,
  targetyear,
  k2020settings,
  frequnc,
  quantile,
  randomseed,
  manual,
  manualadjustments
)
```

Arguments

`ipfilename`	ipfilename
`settingsfilename`	settingsfilename
`rcp`	rcp
`baseyear1`	baseyear1
`baseyear2`	baseyear2
`targetyear`	targetyear
`k2020settings`	k2020settings
`frequnc`	frequnc
`quantile`	quantile
`randomseed`	randomseed
`manual`	manual
`manualadjustments`	manualadjustments

Value

Writes out a csv file

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

Examples

```
ipfilename="settings.csv"
settingsfilename=tempfile(fileext=".csv")
rcp=8
baseyear1=1900
baseyear2=2022
targetyear=2099
k2020settings="Linear and Landfall"
frequnc="Use distribution"
quantile=90
randomseed=0
manual=FALSE
manualadjustments=list(mn=1,sd=0)
#
nahu_write_settings(ipfilename,settingsfilename,rcp,baseyear1,baseyear2,targetyear,
k2020settings,frequnc,quantile,randomseed,manual,manualadjustments)
```

---

nahu_ylt_surgery *An example workflow for applying* ylt_surgery_groups *or* ylt_surgery_groups *to NAHU.*

Description

Shows how to take a YLT (that must have wspd, lat and lon columns) and apply climate change adjustments.

Usage

```
nahu_ylt_surgery(
  type_of_analysis,
  ipfilename,
  settingsfilename,
  ylt_opfilename,
  results_opfilename,
  nyearsinylt,
  rcp,
  baseyear1,
  baseyear2,
  targetyear,
  k2020settings,
```

```
  frequnc,
  quantile,
  units,
  randomseed,
  manual,
  manualadjustments,
  mincat,
  maxcat,
  test = FALSE,
  verbose = FALSE
)
```

Arguments

type_of_analysis	"groups" or "notgroups"
ipfilename	ipfilename
settingsfilename	settingsfilename
ylt_opfilename	ylt_opfilename
results_opfilename	results_opfilename
nyearsinylt	nyearsinylt
rcp	rcp
baseyear1	baseyear1
baseyear2	baseyear2
targetyear	targetyear
k2020settings	k2020settings
frequnc	frequnc
quantile	quantile
units	units
randomseed	randomseed
manual	manual
manualadjustments	manualadjustments
mincat	The minimum cat that is counted
maxcat	The maximum cat that is counted
test	test
verbose	verbose

Details

The sections in the code are as follows:

1-writes out the settings to a csv file, for future reference

2-reads an input longylt from the file ipfilename

3-makes some input checks, to make sure the right columns are available in the input file

4-uses the wspd column to add a cat column to the longylt (from 1 to 7, which is cat -1 to 5) (change this line if you want to make adjustments based on some other feature of the events)

5-uses the lon lat columns to add global region as a column to the longylt (regions from 1 to 18, from the Jewson BAMS paper) (change this line if you want to use different regions)

6-looks at some diagnostics on the input ylt

7-sets up the GMST scenarios (change this line if you want to use different GMST scenarios)

8-sets up the rates changes from the BAMS article supporting material on zenodo (change this line if you want to use different rate changes)

9-does the temporal interpolation based on the input options, GMST scenarios, and zenodo rates

10-converts the adjustments from "by reg-cat" to "by-event", so they are ready to apply to the ylt

11-defines groups in a certain way...currently just by cat) (change this line if you want to define groups differently)

12-actually does the adjustment, and makes a new ylt

13-makes joint ylts from the input and results

14-looks at some diagnostics for the new ylt

15-looks at AAE, AAL change diagnostics

16-looks at AAE, AAL change diagnostics, now by cat

17-writes out the new longylt to a csv file

18-returns

Value

Returns the two YLTs (each consisting of a long and short ylt), and writes the new longylt to a csv file.

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

Examples

```
#
# This long example gives a complete end-to-end example of how to adjust a
# hurricane ylt for climate change.
# It uses either surgery, or surgery with groups (according to the settings below).
#
# Here's roughly what it does:
# -creates a ylt
```

```
# -sets up some parameters that define climate change adjustments
#  (years, scenarios, etc)
# -sets up some parameters that define what outputs to look at
# -and then adjusts the ylt nylt times, so we can see the variability in the
#  effect of the adjustments
# -it also process the adjusted ylts, and processes and prints the results in
#  nice format to screen and files
#
# 1 make the input ylt csv file:
# -3 years, 4 events
# -wspd, lats and lons will be used to generate appropriate rate adjustments
#
library(pracma)
tic()
message("1. make the input ylt csv file")
year=c(1,1,2,3)
evid=c(12,34,56,78)
loss=c(10,20,30,40)
wspd=c(60,90,120,150)
lflat=c(34.779,29.855,28.891,28.83)
lflon=c(-76.596,-84.174,-95.425,-95.515)
df=data.frame(year,evid,loss,wspd,lflat,lflon)
ipfilename=tempfile(fileext=".csv")
write.csv(df,file=ipfilename,row.names=FALSE)
#
# 2 specify the filenames, and nyearsinylt
#
message("2. specify the filenames etc")
results_Nx_opfilename =tempfile(fileext=".csv")
settingsfilename =tempfile(fileext=".csv")
# base only filenames
ylt_opfilename_base ="ylt_op"
results_1x_opfilename_base="results"
nyearsinylt=4
# the nyearsinylt variable forces the code to understand that the ylt represents
# e.g. 50k, even if year 50k is missing, which it might be in some ylts,
# if there are no events in that year
#
# 3 set the other settings (scenarios, etc)
#
message("3. adjustment settings")
type_of_analysis="groups"
type_of_analysis="notgroups"
nylt=1
rcp=8
baseyear1=1900
baseyear2=2022
targetyear=2099
k2020settings="Linear and Landfall"
frequnc="Use distribution"
quantile=90
units="mph"
randomseed=0
```

```
manual=FALSE
mn=c(1,1,1.1,1.2,1.3,1.4,1.5)#note that it has 7 values for cat -1 to cat 5
sd=c(0,0,0,0,0,0,0)
manualadjustments=list(mn=mn,sd=sd)
#
# 4 output analyis settings
#
message("4. output analysis settings")
onemillion=1000000
lossthresholds=c(0,30)*onemillion
maxnumberevents=12
countequalto=TRUE
rps=c(2,5)
rpi=pmax(round(nyearsinylt/rps),1)
probs=round(100/rps,digits=1)
nrp=length(rps)
mincat=0
maxcat=5
ncat=1+maxcat-mincat
ncatp1=ncat+1 #since I've added an extra row at the end for the total
nbs=100
nlossthresholds=length(lossthresholds)
#
# 5 output analysis arrays
#
message("5. output analysis arrays")
aal =matrix(0,2,nylt)
pcaal=matrix(0,2,nylt)

aalbc =array(0,c(2,nylt,ncatp1))
pcaalbc=matrix(0,nylt,ncatp1)

aae =matrix(0,2,nylt)
pcaae=matrix(0,2,nylt)

aaebc =array(0,c(2,nylt,ncatp1))
pcaaebc=matrix(0,nylt,ncatp1)

nbc =array(0,c(2,nylt,ncatp1))

aep =array(0,c(2,nylt,nrp))
daep=matrix(0,nylt,nrp)

oep =array(0,c(2,nylt,nrp))
doep=matrix(0,nylt,nrp)

thresholdcounts =array(0,c(2,nylt,nlossthresholds,(maxnumberevents+1)))
#+1 to cope with 0
#
# 6 loop to generate the new ylts
#
message("6. loop to generate the new ylts")
for (iylt in 1:nylt){
```

```
cat("running version:",iylt,"\n")
randomseed=iylt

# the new two lines have two options
ylt_opfilename =tempfile(fileext=".csv")
results_1x_opfilename =tempfile(fileext=".csv")

cat("...calling nahu_ylt_surgery\n")
# this is where the actual adjustment takes place, given all the settings
bothylts=nahu_ylt_surgery(type_of_analysis=type_of_analysis,
ipfilename,settingsfilename,ylt_opfilename,results_opfilename="",nyearsinylt,
rcp,baseyear1,baseyear2,targetyear,k2020settings,frequnc,quantile,units,randomseed,
manual,manualadjustments,mincat,maxcat,
test=FALSE,verbose=TRUE)
cat("...back from nahu_ylt_surgery\n")

# for each iteration, extract and store what we need for the output file

# 2 extra ylts, make short ylts
longylt1=bothylts$longylt1
longylt2=bothylts$longylt2
shortylt1=ylt_long2short(longylt1)
shortylt2=ylt_long2short(longylt2)

# 1 bootstrap uncertainty on base
if(iylt==1){
bsaepsd=bootstrap_ep_uncertainty(shortylt1$loss_in_year,nbs,rps)
bsoepsd=bootstrap_ep_uncertainty(shortylt1$max_loss_in_year,nbs,rps)
}


# 3 aae and aal diagnostics
# -if there are 7 cats in the file, returns 7 cats
aae[1,iylt]=ylt_diagnostics_aaeaal(longylt1)$AAE
aae[2,iylt]=ylt_diagnostics_aaeaal(longylt2)$AAE
pcaae[iylt]=round(100*(aae[2,iylt]-aae[1,iylt])/aae[1,iylt],digits=2)
aal[1,iylt]=ylt_diagnostics_aaeaal(longylt1)$AAL
aal[2,iylt]=ylt_diagnostics_aaeaal(longylt2)$AAL
pcaal[iylt]=round(100*(aal[2,iylt]-aal[1,iylt])/aal[1,iylt],digits=2)

# 4 aae and aal diagnostics by cat
aaeaalbc1=ylt_diagnostics_aaeaal_by_catf(longylt1,mincat=mincat,maxcat=maxcat)
aaeaalbc2=ylt_diagnostics_aaeaal_by_catf(longylt2,mincat=mincat,maxcat=maxcat)
# aae by cat
aaebc[1,iylt,] =round(aaeaalbc1$AAEbycat,digits=2)
aaebc[2,iylt,] =round(aaeaalbc2$AAEbycat,digits=2)
nbc[1,iylt,] =aaeaalbc1$Nbycat
nbc[2,iylt,] =aaeaalbc2$Nbycat
pcaaebc[iylt,]=round(100*(aaeaalbc2$AAEbycat-aaeaalbc1$AAEbycat)/aaeaalbc1$AAEbycat,
digits=1)
# aal by cat
aalbc[1,iylt,]=round(aaeaalbc1$AALbycat,digits=0)
aalbc[2,iylt,]=round(aaeaalbc2$AALbycat,digits=0)
```

```
pcaalbc[iylt,]=round(100*(aaeaalbc2$AALbycat-aaeaalbc1$AALbycat)/aaeaalbc1$AALbycat,
digits=1)

# 5 calculate AEPs
aep[,iylt,]=make_2_ep_by_sorting(shortylt1$loss_in_year,shortylt2$loss_in_year,rps)
daep[iylt,]=round(100*(aep[2,iylt,]-aep[1,iylt,])/aep[1,iylt,],digits=1)

# 6 calculate OEPs
oep[,iylt,]=make_2_ep_by_sorting(shortylt1$max_loss_in_year,shortylt2$max_loss_in_year,
rps)
doep[iylt,]=round(100*(oep[2,iylt,]-oep[1,iylt,])/oep[1,iylt,],digits=1)

# 7 number of years with n events greater than x loss
m30=30000000
thresholdcounts[1,iylt,,]=ylt_diagnostics_loss_thresholds(longylt1,lossthresholds,
maxnumberevents,countequalto)
thresholdcounts[2,iylt,,]=ylt_diagnostics_loss_thresholds(longylt2,lossthresholds,
maxnumberevents,countequalto)

# write results for each ylt separately, just in case you need that
con=file(results_1x_opfilename)
sink(con,append=TRUE)
sink(con,append=TRUE,type="message")
writeNresults2csv(nyearsinylt,mincat,maxcat,nrp,nylt,maxnumberevents,bsaepsd,bsoepsd,
aal,pcaal,aalbc,pcaalbc,
aae,pcaae,aaebc,pcaaebc,
nbc,
probs,rps,aep,daep,oep,doep,lossthresholds,thresholdcounts,justone=TRUE,iylt)
closeAllConnections()

}

# write results for all ylts together into one file...this is the main output
con=file(results_Nx_opfilename)
sink(con,append=TRUE)
sink(con,append=TRUE,type="message")
writeNresults2csv(nyearsinylt,mincat,maxcat,nrp,nylt,maxnumberevents,bsaepsd,bsoepsd,
aal,pcaal,aalbc,pcaalbc,
aae,pcaae,aaebc,pcaaebc,
nbc,
probs,rps,aep,daep,oep,doep,lossthresholds,thresholdcounts,justone=FALSE,iylt=999)
closeAllConnections()

toc()
```

`rust_dist_haversine` *Function to calculate the Haversine distance between two points. This matches the functionality of geosphere::distHaversine in R.*

Description

Function to calculate the Haversine distance between two points. This matches the functionality of geosphere::distHaversine in R.

Usage

```
rust_dist_haversine(lon1, lat1, lon2, lat2)
```

Arguments

lon1	Lon of first point.
lat1	Lat of first point.
lon2	Lon of second point.
lat2	You guess.

Value

Returns distance in meters.

A message to the screen.

---

rust_hello_world *Return string "Hello world!" to R. Just a test.*

Description

Return string "Hello world!" to R. Just a test.

Usage

```
rust_hello_world()
```

Value

A message to the screen.

`rust_hours_clause_apply_part2` *Hours clause function For a given start event, calculates all possible hours clauses, returns new losses AND a counter*

Description

Hours clause function For a given start event, calculates all possible hours clauses, returns new losses AND a counter

Usage

```
rust_hours_clause_apply_part2(
  hcdays,
  hckm,
  byregion,
  temploss,
  thisyearday,
  thisyearlon,
  thisyearlat,
  thisyearregion,
  nevents_in_year,
  k
)
```

Arguments

`hcdays`	Definition of the hours clause in days
`hckm`	Definition of the hours clause in km
`byregion`	Whether to use regions, or distances, to measure closeness
`temploss`	The losses by event for this year
`thisyearday`	The days by event for this year
`thisyearlon`	The lons by event for this year
`thisyearlat`	The lats by event fo rthis year
`thisyearregion`	For events in this year, the regions
`nevents_in_year`	The number of events in the full year
`k`	The starting event

Value

Returns a vector of the new losses

`rust_square` *Square an integer. Just an example.*

Description

Square an integer. Just an example.

Usage

```
rust_square(x)
```

Arguments

`x`	An integer.

Value

The square of the input.

`stopi` *Stops code intentionally, with message that says that*

Description

Stops execution and returns a message saying the stop was intentional

Usage

```
stopi(location = FALSE)
```

Arguments

`location`	An optional string giving the routine being called from, which is then included in the message.

Details

Not much more to say about this really.

Value

A message to the screen.

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

`template` *A template for R code*

Description

Longer description here

Usage

```
template(x)
```

Arguments

`x`	Input parameter `x`

Details

Does something.

More details here.

Value

Something

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

See Also

- `something else` Does something else

Examples

```
#
# set up all the arguments
#
rcp=8
baseyear1=1900
baseyear2=2022
targetyear=2099
k2020settings="Linear and Landfall"
frequnc="Use distribution"
quantile=90
randomseed=0
gmst=nahu_define_gmst_scenarios()
landfall=nahu_define_k2020_zenodo_landfall_adj()
#
# and calculate the implied landfall rates changes
```

```
#
rates=nahu_k2020_rates_interpolation(rcp,baseyear1,baseyear2,targetyear,
k2020settings,frequnc,quantile,randomseed,gmst,landfall)
#
# have a look at region 1, mean change, all 7 cats
#
print(rates$mn1[1,])
```

`whatreg` *Converts US state and country region abbreviations into numbers from 1 to 7.*

Description

This is based on one particular method for assigning 7 regions to a NAHU model.

Usage

```
whatreg(rg)
```

Arguments

`rg`	Region abbreviations

Details

Uses the following regions:

region 1=c("TX","LA","MS","AL")

region 2=c("FL")

region 3=c("GA","SC","NC","VA")

region 4=c("MD","DE","RI","MA","ME","NY","NJ")

region 5=c("Mexico")

region 6=c("Panama","Costa Rica","Nicaragua","Honduras","El Salvador","Gautemala","Belize")

region 7=undefined

Value

The region from 1 to 7

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

Examples

```
reg="Mexico"
message("region input name = ",reg)
message("region output #   = ",whatreg(reg))
```

`writeNresults2csv` *Writes results from N YLTs to csv format.*

Description

Writes various results from N YLTs to csv. Can be used with `sink()` to make a nicely formatted csv file. The result are: AAE and AAL by cat, AEP, OEP and threshold counting.

Usage

```
writeNresults2csv(
  nyearsinylt,
  mincat,
  maxcat,
  nrp,
  nylt,
  maxnumberevents,
  bsaepsd,
  bsoepsd,
  aal,
  pcaal,
  aalbc,
  pcaalbc,
  aae,
  pcaae,
  aaebc,
  pcaaebc,
  nbc,
  probs,
  rps,
  aep,
  daep,
  oep,
  doep,
  lossthresholds,
  thresholdcounts,
  justone = FALSE,
  iylt
)
```

Arguments

nyearsinylt	number of years
mincat	minimum cat
maxcat	maximum cat
nrp	number of return periods
nylt	number of ylts
maxnumberevents	related to the threshold counting
bsaepsd	bootstraps sds for AEP
bsoepsd	bootstraps sds for OEP
aal	AAL results
pcaal	AAL as percentages
aalbc	AAL results
pcaalbc	AAL as percentages by cat
aae	AAE results
pcaae	AAE as percentages
aaebc	AAE results by cat
pcaaebc	AAE results as percentage
nbc	Number by cat results
probs	return probabilities being used
rps	return levels being used
aep	AEP results
daep	AEP changes
oep	OEP results
doep	OEP changes
lossthresholds	For the threshold counting
thresholdcounts	threshold counting results
justone	logical if there's just one run to be processed
iylt	if there's just one, what is the i

Value

CSV output, for screen or file.

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

Examples

```
#' Uses the example for routine 201, \code{nahu_ylt_surgery}.
```

ws2catf *Convert windspeed in knots into a generalized version of the SSHWS category.*

Description

Converts windspeed

Usage

```
ws2catf(ws, units)
```

Arguments

ws	input windspeed in units
units	The units: knots, mph or mph2

Details

I took the definitions from Wikipedia/Saffir-Simpson_scale. Note that the definitions on Wikipedia have gaps, such as between 63 and 64 knots. I close the gaps by using halves.

Value

The cat from -1,0,1,2,3,4,5, where -1 means TD and 0 means TS.

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

Examples

```
cat("\n")
ws=70
cat("ws in knots=",ws,"\n")
cat("cat=",ws2catf(ws,units="knots"),"\n")
#
cat("\n")
cat("ws in mph=",ws,"\n")
cat("cat=",ws2catf(ws,units="mph"),"\n")
```

yltsim *Simulates a long YLT from a Poisson ELT*

Description

Simulates a long YLT (long format year loss table) from an ELT (event loss table) using the Poisson distribution, with or without secondary uncertainty. Copies standard columns by default, and extra columns if asked to. Previous versions included the short YLT, but now the idea is that this routine just produces the longylt, and `ylt_long2short` can be used to generate the shortylt if it is required.

Usage

```
yltsim(nyearsinylt, elt, columns2copy = NULL, verbose = FALSE, secuncb = FALSE)
```

Arguments

`nyearsinylt`	The number of years of simulation required.
`elt`	A data frame containing the ELT. Must contain `evid`, `mrate` and `mloss` columns, at least.
`columns2copy`	Which extra columns to copy
`verbose`	A logical.
`secuncb`	A logical to indicate whether secondary uncertainty should be simulated. Requires `sloss` and `expo` columns.

Details

Uses Poisson simulation.

The ELT data frame must contain the following two columns

- `evid`: The event ids
- `mloss`: The mean losses by event
- `mrate`: The mean rates by event (can be constant if appropriate).

If secondary uncertainty is required, the ELT data frame must also contain the following two columns:

- `sloss`: The standard deviation of losses by event
- `expo`: The exposure by event

The following columns will be passed through into the output YLT.

- `evid`:
- `mloss`:

Other columns can be passed thru if they are specified (see the example).

The whole issue of event ids is complicated. Here's what I do:

- `evid:` The event id. This will have whatever numbers it had in the ELT. They don't have to start from one. This might end up repeated in the YLT, if the simulations end up repeating that event.
- `eltrow:` I number the events in the ELT from 1. This is that number. This might end up repeated in the YLT, if the simulations end up repeating that event.
- `yltrow:` I number the events in the YLT from 1. This is that number. This will not repeat in the YLT, even if two events are the same. So this is just the row in the YLT.

Value

A data frame containing a long YLT.

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

See Also

- `yltsim_inc:` YLT simulation, but incrementally.
- `yltreduce:` YLT length reduction
- `eltmerge:` A routine for merging historical and model ELTs
- `catxl:` A routine for evaluating catxl towers.

Examples

```
#
# create an elt with 3 rows
#
set.seed(1)
#
# essential parameters
evid=seq(1000,1002,1)
mrate=seq(0.1,0.3,0.1)
mloss=seq(100,300,100)
#
# optional parameters
wspd=c(50,51,52) #testing copying extra columns
region=c("A","B","C")
#
# make the input elt
elt=data.frame(evid,mrate,mloss,wspd,region)
print(elt)
#
# and simulate a ylt with 5 years
#
longylt=yltsim(5,elt,columns2copy=c("wspd","region"))
#
message("Here's the simulated long ylt:")
print(longylt)
#
```

```
shortylt=ylt_long2short(longylt,nyearsinylt=5)
#
message("And here's a short ylt derived from it:")
print(shortylt)
```

`yltsim_inc` *Simulates an Adjusted long YLT using Incremental Simulation, based on an input ELT and long YLT*

Description

You have an ELT. And you have a long YLT that you have previously simulated from the ELT using Poisson simulation (perhaps using the `longyltsim`). Now you want to simulate an adjusted version of the long YLT, based on multiplicative adjustments which are specified for each event in the ELT, as a mean and a sd for each event.

Usage

```
yltsim_inc(
  elt,
  longylt1,
  rate_adjustments_by_event,
  verbose = FALSE,
  secuncb = FALSE,
  randomday = TRUE
)
```

Arguments

`elt`	A data frame containing the ELT. Must contain `mrate` and `mloss` columns.
`longylt1`	A data frame containing a longylt in the format produced by the routine `yltsim()`.
`rate_adjustments_by_event`	A matrix of multiplicative adjustments for the Poisson rate. The first column gives the mean rate adjustment and the second column gives the standard deviation of the rate adjustment. Both columns must be of same length as the columns in `elt`.
`verbose`	A logical.
`secuncb`	A logical to indicate whether secondary uncertainty should be simulated (requires `sloss` column). Not implemented yet.
`randomday`	A logical. `TRUE` means days are randomized, `FALSE` means they are copied.

Details

The simulated long YLT is copied from the input long YLT, and events are then added and deleted to reflect the specified rates changes, according to the 'incremental simulation' algorithm. This algorithm is relevant for the workflow situation in which the base long YLT was simulated from the ELT. If you have a long YLT that wasn't simulated from an ELT, or you've lost the ELT, use `yltsurgery` instead. Incremental simulation attempts to minimize differences due to random noise between the two YLTs and gives more accurate estimates of change than just simulating the second YLT using independent simulation. The rates changes are specified by event. They can be scalar or log-normal distributed, or other distributed, in order to capture adjustment uncertainty. Adjustment uncertainty is simulated using the 'stochastic parameter' simulation algorithm. Changes within years are perfectly correlated.

The input mean and standard deviation rate adjustments are used to define a log-normal distribution for each event. The log-normal is used to simulate different rate adjustments for each year. The log-normal adjustments are perfectly correlated between different events.

The assumption is that YLT1 was sampled from the ELT, using the rates in the ELT. The rate adjustments are specified for all the events in the ELT (some of which may not even occur in YLT1). The eltrow in the longylt in YLT1 must correspond to the events in the ELT and in the specified rate adjustments (which it will if the longylt was created by `longyltsim`. The algorithm consists of two parts. In the first part, events may or may not be copied from YLT1 to YLT2, depending on a coin flip, based on a probability derived from the rate adjustment. If there is no adjustment, an event will be copied. If there is a large reduction in the rate, it may well not be copied. The longylt part of YLT1 provides the eltrow for each event, which is used to determine the rate adjustment. In the second part, events with increasing rates are selected randomly from the ELT (weighted by the increase), and added to the YLT.

The whole issue of event ids is complicated. Here's what I do:

- `evid`: An event id. This will have whatever numbers it had in the ELT. They don't have to start from one. This might end up repeated in the YLT, if the simulations end up repeating that event.
- `eltrow`: I number the events in the ELT from 1. This is that number. This might end up repeated in the YLT, if the simulations end up repeating that event.
- `oldyltrow`: For new events, there is no `oldyltrow`, because new events are selected from the ELT, not the old YLT, and so the value is not uniquely defined.
- `yltrow`: The row in the new YLT. Might well not be the same as the row in the old YLT, because of new events being added. This doesn't repeat in the new YLT.

Value

A data frame containing the new long YLTs.

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

References

For the stochastic parameter simulation algorithm see: https://doi.org/10.1007/s00477-023-02409-0

and: https://en.wikipedia.org/wiki/Year_loss_table#Incremental_Simulation

For the incremental simulation algorithm see: https://doi.org/10.1007/s00477-022-02198-y

and: https://en.wikipedia.org/wiki/Year_loss_table#Stochastic_Parameter_YLTs

Examples

```
#
# create an elt with 3 rows
#
set.seed(1)
evid=seq(1000,1002,1)
mrate=seq(0.1,0.3,0.1)
mloss=seq(100,300,100)
elt=data.frame(evid,mrate,mloss)
nevents=length(mrate)
message("************************elt************************")
print(elt)
#
# simulate longylt1 with 15 years
#
nyears=15
longylt1=yltsim(nyears,elt)
message("************************ylt1************************")
print(longylt1)
#
# define adjustments
#
rate_adjustments_by_event=matrix(0,nevents,2)
rate_adjustments_by_event[,1]=2 #double the rates
rate_adjustments_by_event[,2]=0
#
# simulate longylt2 with those adjustments
#
longylt2=yltsim_inc(elt,longylt1,rate_adjustments_by_event)
message("************************ylt2************************")
print(longylt2)
```

`yltsurgery` *Adjusts a long YLT to make a new long YLT using the surgery algorithm*

Description

The simulated YLT is copied from the input YLT, with events added and deleted to reflect the specified rates changes, according to the 'surgery' algorithm. This algorithm attempts to minimize differences due to random noise between the two YLTs and gives more accurate estimates of change than just simulating the second YLT using independent simulation. The rates changes are specified by event. They can be scalar or log-normal distributed, in order to capture adjustment uncertainty. Adjustment uncertainty is simulated using the 'stochastic parameter' simulation algorithm.

Usage

```
yltsurgery(
  longylt1,
  rate_adjustments_by_event,
  columns2copy = NULL,
  verbose = FALSE,
  randomday = TRUE
)
```

Arguments

`longylt1`	A data frame containing a YLT
`rate_adjustments_by_event`	A matrix of multiplicative adjustments for the Poisson rate. The first column gives the mean rate adjustment and the second column gives the standard deviation of the rate adjustment. Both columns must be of same length as the columns in the input ylt.
`columns2copy`	Specifies a list of columns to be copied from the original YLT, such as `cat`.
`verbose`	A logical.
`randomday`	Do you want the days to be randomized or copied? (requires `sloss` column). Not implemented yet.

Details

The input mean and standard deviation rate adjustments are used to define a log-normal distribution for each event. The log-normal is used to simulate different rate adjustments for each year. The log-normal adjustments are perfectly correlated between different events.

The rate adjustments are specified for all the events in the YLT, in order. The algorithm consists of two parts. In the first part, events may or may not be copied from YLT1 to YLT2, depending on a coin flip, based on a probability derived from the rate adjustment. If there is no adjustment, an event will be copied. If there is a large reduction in the rate, it may well not be copied. YLT1 provides the evid for each event, which is used to determine the rate adjustment. In the second part, events with increasing rates are selected randomly from YLT1 (weighted by the increase), and added to YLT2.

Value

A data frame containing the new long YLTs.

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

References

For the stochastic parameter simulation algorithm see: https://doi.org/10.1007/s00477-023-02409-0

and: https://en.wikipedia.org/wiki/Year_loss_table#Incremental_Simulation

For the incremental simulation algorithm see: https://doi.org/10.1007/s00477-022-02198-y

and: https://en.wikipedia.org/wiki/Year_loss_table#Stochastic_Parameter_YLTs

Examples

```
#
# create an elt with 3 rows
#
set.seed(2)
mrate=seq(0.1,0.3,0.1)
mloss=seq(100,300,100)
evid=seq(1000,1002,1)
region=c("A","B","C")
nevents=length(mrate)
elt=data.frame(evid,mrate,mloss,region)
#
# simulate ylt1 with 10 years
#
nyears=10
longylt1=yltsim(nyears,elt,columns2copy="region")
cat("*************************ylt1************************\n")
print(longylt1)
#
# define adjustments
#
nevents_in_ylt1=length(longylt1$year)
rate_adjustments_by_event=matrix(0,nevents_in_ylt1,2)
rate_adjustments_by_event[,1]=2 #double the rates
rate_adjustments_by_event[,2]=0 #no sd on the changes
#
# simulate ylt2 with those adjustments (but not referring back to the ELT)
#
cat("*************************ylt2************************\n")
longylt2=yltsurgery(longylt1,rate_adjustments_by_event,columns2copy="region")
print(longylt2)
```

`yltsurgery_groups` *Adjusts a long YLT to make a new long YLT using the grouped surgery algorithm*

Description

The simulated YLT is copied from the input YLT, with events added and deleted to reflect the specified rates changes, according to the 'grouped surgery' algorithm. This algorithm attempts to minimize differences due to random noise between the two YLTs and gives more accurate estimates of change than just simulating the second YLT using independent simulation. Within each group, it chooses the number of events to try and hit the target as closely as possible. The rates changes are specified by event. They can be scalar or log-normal distributed, in order to capture adjustment uncertainty. Adjustment uncertainty is simulated using the 'stochastic parameter' simulation algorithm.

Usage

```
yltsurgery_groups(
  longylt1,
  rate_adjustments_by_event,
  groups,
  columns2copy = NULL,
  secunc = FALSE,
  verbose = FALSE,
  randomday = TRUE
)
```

Arguments

longylt1	A data frame containing a YLT
rate_adjustments_by_event	A matrix of multiplicative adjustments for the Poisson rate. The first column gives the mean rate adjustment and the second column gives the standard deviation of the rate adjustment. Both columns must be of same length as the columns in elt.
groups	Specifies which group each event is in.
columns2copy	Specifies a list of columns to be copied from the original YLT, such as cat.
secunc	A logical to indicate whether secondary uncertainty should be simulated (requires sloss column). Not implemented yet.
verbose	A logical.
randomday	Do you want the days to be randomized or copied?

Details

The input mean and standard deviation rate adjustments are used to define a log-normal distribution for each event. The log-normal is used to simulate different rate adjustments for each year. The log-normal adjustments are perfectly correlated between different events.

The rate adjustments are specified for all the events in the YLT, in order. The algorithm consists of two parts. In the first part, events may or may not be copied from YLT1 to YLT2, depending on a coin flip, based on a probability derived from the rate adjustment. If there is no adjustment, an event will be copied. If there is a large reduction in the rate, it may well not be copied. YLT1 provides the evid for each event, which is used to determine the rate adjustment. In the second part, events with increasing rates are selected randomly from YLT1 (weighted by the increase), and added to YLT2.

Value

A data frame containing the new long YLTs.

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

References

For the stochastic parameter simulation algorithm see: https://doi.org/10.1007/s00477-023-02409-0

and: https://en.wikipedia.org/wiki/Year_loss_table#Incremental_Simulation

For the incremental simulation algorithm see: https://doi.org/10.1007/s00477-022-02198-y

and: https://en.wikipedia.org/wiki/Year_loss_table#Stochastic_Parameter_YLTs

Examples

```
#
# create an elt with 3 rows
#
set.seed(2)
evid=seq(1000,1002,1)
mrate=seq(0.1,0.3,0.1)
mloss=seq(100,300,100)
region=c("A","B","C")
nevents=length(mrate)
elt=data.frame(mrate,mloss,evid,region)
#
# simulate ylt1 with 10 years
#
nyears=10
longylt1=yltsim(nyears,elt,columns2copy=c("region"))
# add an evid flag to the ylt just to test that goes through into ylt2
cat("*************************ylt1************************\n")
print(head(longylt1))
#
# define adjustments
#
nevents_in_ylt1=length(longylt1$evid)
rate_adjustments_by_event=matrix(0,nevents_in_ylt1,2)
rate_adjustments_by_event[,1]=2 #double the rates
rate_adjustments_by_event[,2]=0
groups=matrix(1,nevents_in_ylt1)
#
# simulate ylt2 with those adjustments (but not referring back to the ELT)
#
ylt2=yltsurgery_groups(longylt1,rate_adjustments_by_event,groups,columns2copy="region")
cat("*************************ylt2************************\n")
print(head(ylt2))
```

`ylt_add_cat_2_longylt` *Adds a cat column to a long ylt, based on windspeed*

Description

Calculates cat from windspeed, and adds a column for that. Requires `wspd` in the input longylt.

Usage

```
ylt_add_cat_2_longylt(longylt, units)
```

Arguments

longylt	A data frame containing the long YLT
units	"knots","mph"

Details

Cat runs from -1 to 5. Different people use different cat boundaries. To see the boundaries being used, type ws2catf.

Value

A new longylt with a cat column added

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

Examples

```
#
# 1 create a long YLT
#
cat("1. longylt:\n")
year=c(1,1,1,2,2,3,3,4,4,4,5,5,5,5,5)
wspd=seq(10,150,10)
longylt1=data.frame(year,wspd)
print(longylt1)
#
longylt2=ylt_add_cat_2_longylt(longylt1,units="knots")
#
print(longylt2)
```

---

ylt_add_region_2_longylt

Adds a region column to a longylt, based on the lat-lons already in the longylt

---

Description

Calculates region from lat lon (where the regions relate to the Jewson (2023) BAMS paper).

Usage

```
ylt_add_region_2_longylt(longylt)
```

Arguments

longylt	A data frame containing the long YLT. Must contain year, lflat, lflon.

Value

A new ylt with a region column added

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

Examples

```
#
# 1 create a YLT
#
cat("1. longylt:\n")
year=c(1,2)
lflat=c(40,10)
lflon=c(-40,-40)
loss=c(10,20)
longylt=data.frame(year,loss,lflat,lflon)
print(longylt)
#
op=ylt_add_region_2_longylt(longylt)
print(op)
```

---

ylt_check_for_2_events_on_same_day

Checks a YLT for two events on the same day

Description

The assumption is that two events on the same day is bad, so if it finds two events on the same day, it stops running, and gives a message saying what day they are on.

Usage

```
ylt_check_for_2_events_on_same_day(longylt)
```

Arguments

longylt	The input longylt, which needs a day column.

Value

Screen output listing when there are two events on the same day

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

Examples

```
#
# 1 create a long YLT with cat
#
cat("1. make longylt:\n")
year=c(1,1,1,2,2)
day=c(1,2,3,4,5)
loss=rep(0,5) #needs an mloss column for long2short
longylt=data.frame(year,loss,day)
#
# 2 run the check
#
cat("4. run the diagnostics:\n")
op=ylt_check_for_2_events_on_same_day(longylt)
```

---

ylt_check_for_same_events_in_one_year

Checks a YLT for the same event occurring twice in one year

Description

The assumption is that this is undesirable, and so if it happens then the code stops and reports back when it happened.

Usage

```
ylt_check_for_same_events_in_one_year(longylt)
```

Arguments

longylt	The input long, which needs a evid column.

Value

Screen output listing when an event occurs twice in the same year

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

Examples

```
#
# 1 create a long YLT with evid
#
cat("1. make longylt:\n")
year=c(1,1,1,2,2)
loss=rep(0,5)
evid=c(1,2,3,4,5)
longylt=data.frame(year,loss,evid)
#
# 2 run the check
#
cat("4. run the diagnostics:\n")
op=ylt_check_for_same_events_in_one_year(longylt)
```

`ylt_diagnostics_aaeaal` *Long YLT Diagnostics*

Description

Long YLT diagnostics, Works by creating a shortylt using `ylt_long2short`, and then using `ylt_diagnostics_aaeaal`. A bit stupid because the output contains blank weighted values, because `ylt_long2short`, returns weighted values, but there's no way in this case to set the weights. Oh well, never mind. It's fine. You don't really need this one. You can always use `ylt_long2short` yourself, and then use `ylt_diagnostics_aaeaal`. So this is a good candidate for deletion tbh.

Usage

```
ylt_diagnostics_aaeaal(longylt, nyearsinylt = 0)
```

Arguments

`longylt`	A data frame containing the long YLT
`nyearsinylt`	Sometimes, by chance, the long YLT might not have any events in year 100, even though technically it's supposed to be 100 years long. So it's good to specify the number of actual years (in this example, 100). Then the short YLT will have 100 years in it. The last year just won't have any events in it. If you just leave this to the default value of 0, then the short YLT will just stop at the last year in the long YLT.

Value

AAE, AAL, SAE and SAL.

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

Examples

```
#
# 1 create an ELT
#
evid=c(1,2,3)
mrate=c(1,1,1)
mloss=c(10,20,30)
elt=data.frame(evid,mrate,mloss)
#
# 2 create a YLT
#
longylt=yltsim(1000,elt)
#
# 3 call diagnostics
#
op=ylt_diagnostics_aaeaal(longylt)
#
cat(" AAE=",op$AAE,"\n")
cat(" AAL=",op$AAL,"\n")
cat(" SAE=",op$SAE,"\n")
cat(" SAL=",op$SAL,"\n")
```

---

ylt_diagnostics_aaeaal_by_catf

Calculates YLT AAE and AAL by Cat

Description

Deals with the number-of-years issue because that's given by the shortylt.

Usage

```
ylt_diagnostics_aaeaal_by_catf(
  longylt,
  nyearsinylt = 0,
  mincat = 999,
  maxcat = 999
)
```

Arguments

longylt	A data frame containing the YLT
nyearsinylt	The number of years in the YLT.
mincat	The minimum cat that is counted
maxcat	The maximum cat that is counted

Details

Let me explain mincat and maxcat. For hurricane the cat column goes from -1 to 5 the code deals with that using dcat if you specify minc and maxc, then that's what it counts if you don't specify them, it uses the data but using the data has the problem that the returned data shape depends on the data which is really awkward for any further processing so it's best to specify

Value

AAE, AAL, SAE and SAL.

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

Examples

```
#
# 1 make YLT
#
cat("1. make ylt1:\n")
year=c(1,1,1,2,2,3,3,3)
loss=seq(10,80,10)
cat=c(1,1,1,2,2,3,4,5)
longylt=data.frame(year,loss,cat)
#
# 2 look
#
op=ylt_diagnostics_aaeaal_by_catf(longylt,mincat=1,maxcat=5)
print(op)
```

---

ylt_diagnostics_aaeaal_by_cat_change_printf

Prints changes in AAE and AAL from a two YLTs to the screen

Description

Only works for the case where there are *exactly* 5 cats in both YLTs. Which is a bit limiting. Could improve that I guess.

Usage

```
ylt_diagnostics_aaeaal_by_cat_change_printf(
  longylt1,
  longylt2,
  mincat,
  maxcat,
  nyearsinylt = 0,
  verbose = FALSE
)
```

Arguments

longylt1	A data frame containing YLT1
longylt2	A data frame containing YLT2
mincat	The minimum cat that is counted
maxcat	The maximum cat that is counted
nyearsinylt	Number of years in the YLT
verbose	Logical

Value

Prints changes in AAE, AAL to the screen

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

Examples

```
#
# 1 make YLT1
#
cat("1. make ylt1:\n")
year=c(1,1,1,2,2,3,3,3)
loss=seq(10,80,10)
cat=c(1,1,1,2,2,3,4,5)
longylt1=data.frame(year,loss,cat)
#
# 2 make YLT2
#
cat("1. make ylt2:\n")
year=c(1,1,2,2,2,2,3,3)
loss=seq(20,90,10)
cat=c(1,1,1,2,2,3,4,5)
longylt2=data.frame(year,loss,cat)
#
# look at changes
#
cat("calling ylt_change_diagnostics_aaeaal_by_cat_print:\n")
ylt_diagnostics_aaeaal_by_cat_change_printf(longylt1,longylt2,mincat=1,maxcat=5)
```

---

ylt_diagnostics_aaeaal_by_cat_printf

Prints AAE and AAL from a YLT to the screen

Description

Just a way to print the results from ylt_diagnostics_aaeaal_by_catf to the screen in a pretty way.

Usage

```
ylt_diagnostics_aaeaal_by_cat_printf(
  longylt,
  nyearsinylt = 0,
  mincat,
  maxcat,
  verbose = FALSE
)
```

Arguments

longylt	The YLT
nyearsinylt	The number of years in the YLT.
mincat	The minimum cat that is counted
maxcat	The maximum cat that is counted
verbose	Logical

Value

Prints changes in AAE, AAL to the screen

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

Examples

```
#
# 1 make YLT
#
cat("1. make ylt1:\n")
year=c(1,1,1,2,2,3,3,3)
loss=seq(10,80,10)
cat=c(1,1,1,2,2,3,4,5)
longylt=data.frame(year,loss,cat)
#
# 2 look
#
cat("2. print the diagnostics\n")
op=ylt_diagnostics_aaeaal_by_cat_printf(longylt,mincat=1,maxcat=5)
```

`ylt_diagnostics_aaeaal_change_printf` *Prints changes in AAE and AAL from a two longylts to the screen*

Description

Because they are longylts, requires the number of years.

Usage

```
ylt_diagnostics_aaeaal_change_printf(longylt1, longylt2, nyearsinylt = 0)
```

Arguments

`longylt1`	A data frame containing longylt1
`longylt2`	A data frame containing longylt2
`nyearsinylt`	nyearsinylt

Value

Prints changes in AAE, AAL to the screen

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

Examples

```
#
# 1 create an ELT
#
evid=c(1,2,3)
mrate=c(1,1,1)
mloss=c(10,20,30)
elt=data.frame(evid,mrate,mloss)
#
# 2 create YLTs
#
longylt1=yltsim(1000,elt)
longylt2=yltsim(1000,elt)
#
# 3 call diagnostics
#
ylt_diagnostics_aaeaal_change_printf(longylt1,longylt2)
#
```

`ylt_diagnostics_aaeaal_printf` *Prints AAE and AAL from a Long YLT to the screen*

Description

Because it's a longylt, you have to specify the `nyearsinylt`. If you have a full ylt, better to use `ylt_diagnostics_aaeaal_print`, because then you don't have to specify that.

Usage

```
ylt_diagnostics_aaeaal_printf(longylt, nyearsinylt = 0)
```

Arguments

`longylt`	A data frame containing the long YLT
`nyearsinylt`	Sometimes, by chance, the long YLT might not have any events in year 100, even though technically it's supposed to be 100 years long. So it's good to specify the number of actual years (in this example, 100). Then the short YLT will have 100 years in it. The last year just won't have any events in it. If you just leave this to the default value of 0, then the short YLT will just stop at the last year in the long YLT.

Value

Prints AAE, AAL to the screen

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

Examples

```
#
# 1 create an ELT
#
evid=c(1,2,3)
mrate=c(1,1,1)
mloss=c(10,20,30)
elt=data.frame(evid,mrate,mloss)
#
# 2 create a YLT
#
longylt=yltsim(1000,elt)
#
# 3 call diagnostics
#
ylt_diagnostics_aaeaal_printf(longylt)
#
```

`ylt_diagnostics_loss_thresholds` *Counts number of events per year above loss thresholds*

Description

Reads in a longylt, and some thresholds, and counts the number of years in which there are `n` or more events exceeding each loss threshold (think carefully: it's a complicated thing actually).

Usage

```
ylt_diagnostics_loss_thresholds(
  longylt,
  lossthresholds,
  maxnumber = 12,
  countequalto = TRUE
)
```

Arguments

`longylt`	A data frame containing the long YLT
`lossthresholds`	A vector of loss thresholds
`maxnumber`	Count up to this number of events
`countequalto`	Logical for how to count

Value

A matrix (lossthresholds by events) with the counts

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

Examples

```
#
# 1 create a YLT
#
cat("1. longylt:\n")
year=c(1,1,1,2,2,3,3,4,4,4,5,5,5,5,5)
loss=seq(10,150,10)
longylt=data.frame(year,loss)
print(head(longylt))
#
cat("\n2. calculate results by loss thresholds:\n")
lossthresholds=c(10,30)
maxnumber=3
op=ylt_diagnostics_loss_thresholds(longylt,lossthresholds,maxnumber)
print(op)
```

`ylt_diagnostics_nchanges` *Sees how many years have changed between a YLT and an adjusted YLT*

Description

Because they are longylts, requires the number of years.

Usage

```
ylt_diagnostics_nchanges(longylt1, longylt2, nyearsinylt = 0, maxchange = 5)
```

Arguments

`longylt1`	A data frame containing longylt1
`longylt2`	A data frame containing longylt2
`nyearsinylt`	nyearsinylt
`maxchange`	How high to count when looking at how many years have n more events. The default value of 5 is probably fine.

Value

Prints changes to the screen

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

Examples

```
#
# create an elt with 3 rows
#
set.seed(2)
mrate=seq(0.1,0.3,0.1)
mloss=seq(100,300,100)
evid=seq(1000,1002,1)
region=c("A","B","C")
nevents=length(mrate)
elt=data.frame(evid,mrate,mloss,region)
#
# simulate ylt1 with 10 years
#
nyears=10
longylt1=yltsim(nyears,elt,columns2copy="region")
cat("*************************ylt1************************\n")
print(longylt1)
```

```
#
# define adjustments
#
nevents_in_ylt1=length(longylt1$year)
rate_adjustments_by_event=matrix(0,nevents_in_ylt1,2)
rate_adjustments_by_event[,1]=2 #double the rates
rate_adjustments_by_event[,2]=0 #no sd on the changes
#
# simulate ylt2 with those adjustments (but not referring back to the ELT)
#
cat("************************ylt2************************\n")
longylt2=yltsurgery(longylt1,rate_adjustments_by_event,columns2copy="region")
print(longylt2)
#
# now look at changes
#
ylt_diagnostics_nchanges(longylt1,longylt2,maxchange=6)
```

`ylt_long2short` *Calculate a Short YLT from a Long YLT*

Description

Calculates a Short YLT from a Long YLT. Requires `year` and `loss`. Assumes that the `year` should start from 1 (if there is an event in the first year). Assumes that the years are in order. The short YLT output includes years that have no events in the long YLT input. The example shows how to combine them to make a full YLT.

Usage

```
ylt_long2short(
  longylt,
  nyearsinylt = 0,
  zeroeventyearsincluded = TRUE,
  verbose = FALSE
)
```

Arguments

`longylt`	A data frame containing the long YLT
`nyearsinylt`	Sometimes, by chance, the long YLT might not have any events in year 100, even though technically it's supposed to be 100 years long. So it's good to specify the number of actual years (in this example, 100). Then the short YLT will have 100 years in it. The last year just won't have any events in it. If you just leave this to the default value of 0, then the short YLT will just stop at the last year in the long YLT.

zeroeventyearsincluded
: What if year 17 doesn't have any events in it? Should it be included in the short YLT? Yes, it should. And that's the default. But if you set this to FALSE, then that year will not be included.

verbose
: If you set this to true, you get some nice information about the number of years, etc.

Value

A data frame containing the short YLT, with columns `year`, `longylt_start_row`, `nevents_in_year`, `loss_in_year`, `max_loss_in_year`. Note that if there are years that are missing from the long YLT because they don't contain any events, they will be listed as start_row=NA, nevents=0. You might think I could just put the start now, not NA. But if this is the last year, you need NA, because there is no row in the long YLT.

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

Examples

```
#
# 1 create a long YLT with cat
#
cat("1. make the longylt:\n")
year=c(1,1,1,2,4,5,6,7,8,8)
loss=seq(10,55,5)
cat=c(1,2,3,4,5,6,7,8,9,10)
longylt=data.frame(year,loss,cat)
print(longylt)
#
# 2 make the shortylt
#
cat("2. make the shortylt:\n")
shortylt=ylt_long2short(longylt,nyearsinylt=10)
print(shortylt)
```

---

ylt_rate_adjustments_catreg_2_row

Makes rates adjustments by event, for events in a YLT

Description

Maps rate adjustments by cat and region to rate adjustments by event, which is what `yltsurgery` needs as input.

Usage

```
ylt_rate_adjustments_catreg_2_row(longylt, ratesbycatreg)
```

Arguments

longylt	A data frame containing the long YLT
ratesbycatreg	A list containing the mean and sd adjustments, each of which is a matrix with dimensions nregions x ncat.

Value

A matrix with the adjustments by event column 1 is the mean of the adjustments. column 2 is the sd of the adjustments.

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

Examples

```
#
# 1 create a 10 year YLT
# -2 regions
# -7 cats
#
cat("1. longylt:\n")
year =c(1,1,1,2,2,3,3,4,4,4)
cat =c(-1,0,1,2,3,4,5,5,5,5)
region=c(1,1,1,1,1,2,2,2,2,2)
longylt=data.frame(year,cat,region)
print(head(longylt))
#
# 2 make adjustments
#
mn=matrix(0,2,7)
for (i in 1:2){
for (j in 1:7){
mn[i,j]=(i-1)*cat[j]+cat[j]
}
}
sd=matrix(0,2,7)
ratesbycatreg=list(mn=mn,sd=sd)
# 3 convert adjustments
#
adj=ylt_rate_adjustments_catreg_2_row(longylt,ratesbycatreg)
print(head(adj))
```

`ylt_rate_adjustments_cat_2_row` *Makes rates adjustments by event, for events in a YLT*

Description

Maps rate adjustments by cat to rate adjustments by event, which is what `yltsurgery` needs as input.

Usage

```
ylt_rate_adjustments_cat_2_row(longylt, ratesbycat)
```

Arguments

`longylt`	A data frame containing the long YLT
`ratesbycat`	A list containing the mean and sd adjustments, each of which is a vector with dimension ncat.

Value

A matrix with the adjustments by event column 1 is the mean of the adjustments. column 2 is the sd of the adjustments.

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

Examples

```
#
# 1 create a 10 year YLT
# -2 regions
# -7 cats
#
cat("1. longylt:\n")
year=c(1,1,1,2,2,3,3,4,4,4)
cat =c(-1,0,1,2,3,4,5,5,5,5)
longylt=data.frame(year,cat)
print(head(longylt))
#
# 2 make adjustments
#
mn=matrix(0,7)
for (j in 1:7){
mn[j]=cat[j]
}
sd=matrix(0,7)
ratesbycat=list(mn=mn,sd=sd)
```

```
# 3 convert adjustments
#
adj=ylt_rate_adjustments_cat_2_row(longylt,ratesbycat)
print(head(adj))
```

`ylt_write2results2csv` *Writes results from 2 YLTs to csv format.*

Description

Writes various results from 2 YLTs to a csv format. Not to a csv file...but to standard output that could be sent to a csv file using the `sink()` function. The result are: AAE and AAL by cat, and AEP.

Usage

```
ylt_write2results2csv(longylt1, longylt2, rps)
```

Arguments

`longylt1`	The first YLT
`longylt2`	The first YLT
`rps`	The return periods you want to calculate

Value

A csv file.

Author(s)

Stephen Jewson <stephen.jewson@gmail.com>

Examples

```
#
# 1 create YLT1
#
year=c(1,1,2,2,3)
cat=c(1,2,3,4,5)
loss=seq(10,50,10)
longylt1=data.frame(year,cat,loss)
#
# 2 create YLT2
#
year=c(1,1,2,2,3)
cat=c(1,2,3,4,5)
loss=seq(110,150,10)
longylt2=data.frame(year,cat,loss)
#
```

```
# 3 do it
#
rps=c(2,5,10,50,100,130,200,250,500,1000,5000)
ylt_write2results2csv(longylt1,longylt2,rps)
```

Index

addvectors, 3

bootstrap_ep_uncertainty, 4

calc_beta_params, 5
catxl, 6
compare_distance_routines, 8

defineregion, 9

elt_add_nahu_cat, 10
elt_collapse_regions, 11
elt_diagnostics_aaeaal, 12
elt_diagnostics_aaeaal_by_cat, 13
elt_diagnostics_aaeaal_with_adj, 14
elt_diagnostics_epcurves, 15
elt_diagnostics_epcurves_with_adj, 16
elt_rate_adjustments_cat_2_event, 18
elt_rate_adjustments_regcat_2_event, 19

filetest, 20

hours_clause, 21
hours_clause_apply_part2, 23
hours_clause_wrapper_ylt, 25
hours_clause_write_settings, 27

imports, 28
input_checks, 28

make_2_ep_by_sorting, 29
make_elt, 30
make_ep_by_sorting, 31
make_ylt, 32

nahu_define_gmst_scenarios, 33
nahu_define_k2020_zenodo_landfall_adj, 34
nahu_k2020_rates_interpolation, 35
nahu_write_settings, 36
nahu_ylt_surgery, 37

rust_dist_haversine, 43
rust_hello_world, 44
rust_hours_clause_apply_part2, 45
rust_square, 46

stopi, 46

template, 47

whatreg, 48
writeNresults2csv, 49
ws2catf, 51

ylt_add_cat_2_longylt, 60
ylt_add_region_2_longylt, 61
ylt_check_for_2_events_on_same_day, 62
ylt_check_for_same_events_in_one_year, 63
ylt_diagnostics_aaeaal, 64
ylt_diagnostics_aaeaal_by_cat_change_printf, 66
ylt_diagnostics_aaeaal_by_cat_printf, 67
ylt_diagnostics_aaeaal_by_catf, 65
ylt_diagnostics_aaeaal_change_printf, 69
ylt_diagnostics_aaeaal_printf, 70
ylt_diagnostics_loss_thresholds, 71
ylt_diagnostics_nchanges, 72
ylt_long2short, 73
ylt_rate_adjustments_cat_2_row, 76
ylt_rate_adjustments_catreg_2_row, 74
ylt_write2results2csv, 77
yltsim, 52
yltsim_inc, 54
yltsurgery, 56
yltsurgery_groups, 58