Package 'newmark'

November 11, 2026

Type Package

Title Uncertainty Analysis in Dynamic Site and Slope Response

Version 1.14.0

Description Estimates the seismic response of sites and slopes from the results of a seismic hazard analysis. For a site, it computes the fundamental period of the soil profile and the amplification of spectral acceleration. For a slope, it computes the fundamental period, the earthquake-induced displacement, and the pseudostatic coefficient that keeps a selected quantile of the displacement within a given limit. Uncertainty in the soil profile and in the ground motion is represented by Monte Carlo samples. Every result is reported with its representative value, its applicable quantiles or dispersion, and the population or conditioning to which it refers.

License AGPL (>= 3)

Encoding UTF-8

Language en-US

LazyData true

LazyDataCompression xz

Depends R (>= 4.2)

Imports data.table, jsonlite, mvtnorm (>= 1.2-3), parallel, stats, utils

Suggests testthat (>= 3.0.0), knitr, readxl, rmarkdown

Config/testthat/edition 3

Config/roxygen2/version 8.1.1

URL https://kanameishi.github.io/newmark/

NeedsCompilation no

Author Alejandro Verri Kozlowski [aut, cre, cph]

Maintainer Alejandro Verri Kozlowski <averri@fi.uba.ar>

Repository CRAN

Date/Publication 2026-11-11 22:10:02 UTC

Contents

assembleDnPOETables . . . 3
assembleDnTables . . . 4
assembleKmaxTables . . . 5
buildDnPOETable . . . 6
buildDnTables . . . 7
buildGeometry . . . 10
buildKmaxTable . . . 11
buildKyGrid . . . 12
buildKyLimits . . . 13
buildPGATable . . . 14
buildQuantileSpline . . . 15
buildRootModel . . . 16
buildShearDraws . . . 17
buildShearTable . . . 19
characterizeProfile . . . 20
classifySite . . . 21
convertAEP . . . 22
correlateResiduals . . . 23
CylinderRoots . . . 24
describeModule . . . 24
estimateAI . . . 25
estimatePGV . . . 26
estimateRayleighPeriod . . . 27
estimateVS30 . . . 28
estimateWeightedRayleighPeriod . . . 29
evaluateBeamPeriod . . . 30
evaluateDisplacements . . . 31
evaluateFlexibleDisplacement . . . 32
evaluateRigidDisplacement . . . 35
evaluateRoot . . . 36
evaluateSiteFactor . . . 37
evaluateSiteSa . . . 39
evaluateSoilLayers . . . 40
evaluateUniformHazard . . . 41
integrateDnHazard . . . 42
integrateDnHazardProfiles . . . 44
integrateSiteMixture . . . 46
integrateSiteSpectrum . . . 48
interpolateSpectrum . . . 49
listDisplacementFamilies . . . 50
listProjectTables . . . 51
processProject . . . 52
propagateBeamPeriod . . . 56
propagateSitePeriod . . . 58
queryProjectTable . . . 59
readHazardWorkbook . . . 60

refineSiteMixture 61
refineSiteSpectrum 62
sampleIntensities 63
sampleSoilLayers 64
sampleSpectra 66
selectDisplacementModels 67
ShearModelParameters 68
simulateSiteSpectrum 69
simulateSoilProfiles 70
SiteClass 71
solveColumnPeriod 72
solveKmax 74
solveProfilePeriods 76
solveWedgePeriod 77
summarizeDisplacements 79
summarizeEnsemble 80
summarizeKmax 81
summarizeMixture 83
summarizeProfiles 84
VancouverSpectra 84

Index **86**

`assembleDnPOETables` *Assemble the displacement exceedance tables of sites*

Description

Validates and stacks the site products of `buildDnPOETable()`. Each site must have the declared demands, geometry-material pairs and thresholds. The weighted counts are already in the site tables and are not recomputed.

Usage

```
assembleDnPOETables(tables, shear, modelID, TR, weights, DaH)
```

Arguments

`tables`	Named list of site DnPOETable products, one per site identifier.
`shear`, `modelID`, `TR`, `weights`, `DaH`	As for `buildDnPOETable()`. Weights define the active models of the contract; stored tables do not retain individual model counts or weights, so their values cannot be recovered or checked from the aggregate POE.

Value

A data.table with the columns, key and meanings of `buildDnPOETable()`, holding every supplied site. Inputs and the random stream are unchanged; no files are written.

`assembleDnTables` *Global displacement products of release 1.7.1*

Description

Compatibility route that reproduces, value for value, the global `DnTable` and `DnPlotTable` that the dn step of `newmark` 1.7.1 assembles from the site `DnTable` products of `buildDnTables()`, for comparison with the installed library.

Usage

```
assembleDnTables(tables, shear, weights, modelID, TR, Vref)
```

Arguments

`tables`	Named list of site `DnTable` products, one per site identifier.
`shear`, `weights`, `modelID`, `TR`, `Vref`	As for `buildDnTables()`.

Value

A list with the keyed data.tables `DnTable` (all sites) and `DnPlotTable` (`siteID`, `ID`, `TR`, `IDg`, `IDn`, `p`, `ky` in g and `Dn` in cm, keyed by the first seven). The site tables are not modified.

Semantics reproduced from `newmark` 1.7.1

Every site table is checked against the product contract and the shear table, the tables are stacked and keyed by `siteID`, `ID`, `TR`, `IDg`, `IDm`, `IDn`, `p` and `ky`. The plot table averages the material curves of each site, demand, geometry, model and level on a common grid: rows with $D_n \leq 10^{-4}$ cm are dropped; the grid has $N = \max(2, \mathrm{round}(\mathrm{median}_m\, n_m))$ points log-spaced between the smallest and largest remaining k_y, where n_m counts the distinct k_y of material m; each material curve is interpolated linearly in $(\ln k_y, \ln D_n)$ without extrapolation (`stats::approx`, `rule = 1`, ties averaged), the finite values are averaged arithmetically over materials, and the points at or below 10^{-4} cm are dropped again. It is a display product: the average over materials has no probabilistic meaning of its own.

Examples

```
# The inputs of the product test of newmark 1.7.1: one rock lognormal per
# period, its MCE quantiles and two geometry-material pairs.
Rock <- data.table::data.table(ID = "TEST", siteID = "TST", TR = 100, Tn = c(0, 0.2, 1),
                               Sa = c(0.3, 0.45, 0.24), Vs30 = 760, Vref = 760)
Rock[, `:=`(SaF = Sa, AF = NA_real_, k = "oq", muLn = log(Sa) - 0.4^2 / 2, sigmaLn = 0.4,
            w = 1, fitErr = 0)]
Levels <- c(0.05, 0.10, 0.16, 0.50, 0.84, 0.90, 0.95)
MCE <- Rock[, .(p = c(sprintf("%.2f", Levels), "mean"),
                Sa = c(exp(muLn + 0.4 * qnorm(Levels)), Sa)), by = Tn]
MCE[, `:=`(ID = "MCE", siteID = "TST", Vs30 = 760, Vref = 760)]
```

```
Shear <- data.table::data.table(IDg = c("G1", "G2"), IDm = c("U1", "U2"), VS30 = c(360, 560),
                                Ts = 0.3, Vref = 760, level = "mean")
Draws <- Shear[, .(IDg, IDm, realization = 1L, Ts, VS30, Vref)]
Site <- buildDnTables(Rock, MCE, Shear, modelID = "TEST", siteID = "TST", TR = 100, Vref = 760,
                      Mw = 7, NS = 2, weights = c(AM88 = 1, BM19 = 1), zTrunc = 3,
                      workers = 1, draws = Draws)
Global <- assembleDnTables(list(TST = Site$DnTable), shear = Shear, weights = c(AM88 = 1, BM19 = 1),
                           modelID = "TEST", TR = 100, Vref = 760)
Global$DnPlotTable
```

`assembleKmaxTables` *Global yield-coefficient table of release 1.7.1*

Description

Compatibility route that reproduces, byte for byte once saved with `saveRDS()`, the global `kmaxTable` product that the kmax step of `newmark` 1.7.1 assembles with its `getKmaxTable()` from the site `kmaxTable` products of `buildKmaxTable()`, so that the stored project tables can serve as comparison oracle. With one site, as in the current projects, the global table equals the table of that site.

Usage

```
assembleKmaxTables(tables, shear, modelID, TR, weights, DaH)
```

Arguments

`tables`	Named list of site `kmaxTable` products, one per site identifier.
`shear`, `modelID`, `weights`, `DaH`	As for `buildKmaxTable()`.
`TR`	Distinct positive return periods of the model demands, in years. A repeated period stops the call at the row count of the site tables, whereas `buildKmaxTable()` drops the repeat.

Value

A data.table of the rows of every site, keyed by `siteID`, `ID`, `TR`, `IDg`, `IDm`, `IDn`, `p` and `Da`, with the columns of `buildKmaxTable()`. The caller's tables are not modified.

Semantics reproduced from `newmark` 1.7.1

Every site table is checked as `buildKmaxTable()` checks its own, except against its PGA table: the global assembly does not read the per-site PGA tables, which `buildKmaxTable()` validates, with each site table against its own, when it builds that table. A site table is checked against the mean rows of the shear table, the demands of `modelID` at `TR` and of the MCE, the models of `weights` and every family with an active member, the eight levels of `p` and the levels $D_a = D_{aH} H_s$ cm rounded to six significant digits over every height of the shear table. The tables are stacked in the order of `tables` and keyed by `siteID`, `ID`, `TR`, `IDg`, `IDm`, `IDn`, `p` and `Da`, and a warning labelled `global`

reports the rows that rely on roots beyond the stored grid and the largest such kmax. The number of realizations is not an argument, and of weights only the names are used: the table records neither the number of realizations nor the numerical weights that generated it, so a stored table cannot be checked against them.

Examples

```
# The inputs of the product test of newmark 1.7.1: one rock lognormal per
# period, its MCE quantiles and two geometry-material pairs,
# here with the heights Hs from which the levels of DaH are derived.
Rock <- data.table::data.table(ID = "TEST", siteID = "TST", TR = 100, Tn = c(0, 0.2, 1),
                               Sa = c(0.3, 0.45, 0.24), Vs30 = 760, Vref = 760)
Rock[, `:=`(SaF = Sa, AF = NA_real_, k = "oq", muLn = log(Sa) - 0.4^2 / 2, sigmaLn = 0.4,
            w = 1, fitErr = 0)]
Levels <- c(0.05, 0.10, 0.16, 0.50, 0.84, 0.90, 0.95)
MCE <- Rock[, .(p = c(sprintf("%.2f", Levels), "mean"),
                Sa = c(exp(muLn + 0.4 * qnorm(Levels)), Sa)), by = Tn]
MCE[, `:=`(ID = "MCE", siteID = "TST", Vs30 = 760, Vref = 760)]
Shear <- data.table::data.table(IDg = c("G1", "G2"), IDm = c("U1", "U2"), Hs = c(10, 120),
                                VS30 = c(360, 560), Ts = 0.3, Vref = 760, level = "mean")
Weights <- c(AM88 = 1, BM19 = 1)
Draws <- Shear[, .(IDg, IDm, realization = 1L, Ts, VS30, Vref)]
Site <- buildDnTables(Rock, MCE, Shear, modelID = "TEST", siteID = "TST", TR = 100, Vref = 760,
                  Mw = 7, NS = 2, weights = Weights, zTrunc = 3, workers = 1, draws = Draws)
PGATable <- buildPGATable(Rock, MCE, Shear, modelID = "TEST", siteID = "TST", TR = 100)
Kmax <- buildKmaxTable(Site$DnDraws, PGATable = PGATable, shear = Shear, modelID = "TEST",
                   siteID = "TST", TR = 100, NS = 2, weights = Weights, DaH = c(0.05, 0.1))
Global <- assembleKmaxTables(list(TST = Kmax), shear = Shear, modelID = "TEST", TR = 100,
                             weights = Weights, DaH = c(0.05, 0.1))
# With one site, the global table equals the table of that site.
identical(Global, Kmax)
```

---

buildDnPOETable *Empirical displacement exceedance table of a site*

Description

Counts the stored, pointwise DnDraws realizations above each admissible displacement. For each demand, geometry, material and ky, each draw receives its model's weight and POE is sum(w[Dn > Da]) / sum(w). The threshold is DaH * Hs in cm, with DaH in percent and the mean shear-table height Hs in metres. No displacement or threshold is rounded.

Usage

```
buildDnPOETable(draws, shear, modelID, siteID, TR, NS, weights, DaH)
```

Arguments

draws	Site DnDraws, with siteID, ID, TR, IDg, IDm, IDn, ky, realization, Dn and units.Dn. Every cell and ky must contain the NS realizations of every active model, with no duplicate identities.
shear	ShearTable with IDg, IDm, Hs and level. Its mean rows define the geometry-material pairs and one height per geometry.
modelID	Identifier of the return-period demand model.
siteID	Site identifier.
TR	Distinct positive return periods in years; numeric() selects MCE alone. MCE is always included, with ID = "MCE" and TR = NA.
NS	Positive number of realizations per active model and cell.
weights	Positive finite weights named by the active displacement models, as returned by selectDisplacementModels().
DaH	Distinct positive displacement thresholds in percent of Hs.

Details

This fraction is conditional on the demand (return period or MCE) and inherits the population represented by the draws: the positive lognormal branch for the flexible models. Stored zeros, including SR08 underflow, count as non-exceedance. It is not an integrated displacement hazard.

Value

A data.table with columns siteID, ID, TR, IDg, IDm, IDn ("ensemble"), ky (g), DaH (%), Da (cm), POE, n (integer count of all draws in the cell and ky) and units.Da ("cm"), keyed by siteID, ID, TR, IDg, IDm, ky, DaH. The inputs and random stream are unchanged; no files are written.

---

buildDnTables *Site displacement tables with soil-column uncertainty*

Description

Builds DnDraws and DnTable from the paired Ts and VS30 values of ShearDraws. PGATable is built separately by buildPGATable() on the mean rows of ShearTable. For every return period of the probabilistic demand and for the MCE, and for every geometry and material pair, it amplifies the rock demand to the VS30 of each column rounded to the nearest 1 m/s for spectrum lookup, samples its scenario intensities at that column's Ts with sampleIntensities(), evaluates the active models with evaluateDisplacements() and summarizes them with summarizeDisplacements() and, for the families ensemble, rigid and flexible, the weighted summary of summarizeEnsemble() with the quantile of newmark 1.7.1 (see below).

Usage

```
buildDnTables(
  rock,
  MCE,
  shear,
  modelID,
  siteID,
  TR,
  Vref,
  Mw,
  NS,
  weights,
  zTrunc,
  workers,
  draws
)
```

Arguments

rock	The UHSRock table of the site: one lognormal distribution at each oscillator period per return period at Vs30 = Vref = 760 m/s (ID, siteID, TR, Tn, Sa, Vs30, Vref, SaF, AF, k, muLn, sigmaLn, w, fitErr).
MCE	The MCETable: rock quantiles and mean of the MCE (ID, siteID, Tn, p, Sa, Vs30, Vref), without TR.
shear	The ShearTable, whose mean rows define the cell reference properties and common ky grid.
modelID	Identifier of the probabilistic demand model.
siteID	Identifier of the site.
TR	Return periods of the probabilistic demand, in years.
Vref	Reference rock velocity in m/s; the product contract admits only 760.
Mw	Moment magnitude of the scenarios: one number for every demand, or a numeric vector named by the return periods of TR, written as they print, and MCE.
NS	Realizations per geometry, material and demand.
weights	Named positive weights of the active models, for example from selectDisplacementModels().
zTrunc	Truncation level of the probabilistic rock demand, in standard deviations.
workers	Number of forked processes that evaluate the pairs of a demand after their draws; 1 evaluates them in this process. The products do not depend on it.
draws	The ShearDraws table with IDg, IDm, realization, Ts, VS30 and Vref. Each pair of shear must be represented, with realization identifiers from 1 without gaps or duplicates. Its columns are sampled in order and reused cyclically until all NS movements are assigned; NS remains the total per pair and demand.

Value

A list with the keyed data.tables `DnDraws` (one row per demand, pair, model, realization and yield acceleration, with the integer draw identifier `realization` and the sampled PGA `PGA.s`) and `DnTable` (quantiles and mean of every model and family). Stored draws whose identifier is named `s` require that column to be renamed to `realization` before passing them to the draw consumers. The rows of the flexible models (`BT07`, `BM17`, `BM19`) are conditional on a nonzero displacement, as `evaluateFlexibleDisplacement()` documents; the probability of zero displacement is not represented. The inputs are not modified.

Population and summaries

Pairs are ordered by `IDg` and `IDm`. Within each pair, the `NS` movements cycle through the columns in increasing `realization` order; 200 movements and 100 columns give two movements per column. Each movement uses the same column for site response, intensity sampling and displacement evaluation, and keeps its identity over models and ky. Each measure (PGA, Sa13 and Sa15) is centered once over the complete `NS` sample on the column means weighted by their assigned movements. The column subsets are not centered separately. Innovations are drawn in movement order, so identical columns reproduce a single-column sample. The `mean` rows of `ShearTable` supply the reference `Ts` and `Vs30` metadata in `DnTable` and the PGA from which its common ky grid is built. Site response for a mean row uses that row's stored VS30. Column spectrum lookup uses `round(VS30)`, with a 1 m/s step; `ShearDraws` and the table reference metadata retain their stored values. The probabilistic demands sample the analytic mixture of the amplified spectrum, truncated at `zTrunc` like the hazard calculation; the MCE samples its amplified quantile spectrum untruncated. The yield grid is `buildKyGrid()` with 30 points on the mean PGA of the amplified spectrum. Each demand reseeds the generator with a fixed seed of the site, the demand and the return period (a base-31 hash of `"<siteID>/<demand>/<TR>"` modulo $2^{31} - 1$), after the deterministic site response and before its draws, so a site's products do not depend on the sites or demands run with it (the session's random stream is replaced, as the dn step of 1.7.1 replaced it); within a demand the pairs draw in order, each its `NS` intensity realizations and then its `NS` model deviates. Quantile levels are 0.05, 0.10, 0.16, 0.50, 0.84, 0.90 and 0.95 plus the mean, labelled `"%.2f"`; `BM19` takes ordinary motions (`NFC = "D100"`, `pulse = FALSE`). The MCE rows carry `ID = "MCE"` and `TR = NA`; PGA and ky are in g, Dn in cm. The family quantiles are those of `Hmisc::wtd.quantile()` as 1.7.1 called it: once the pool of one yield acceleration holds a tie, every pooled value is read back from its 15-significant-digit text before the quantile, while the mean uses the exact values; `summarizeEnsemble()`, the general summary, merges ties at full precision.

Examples

```
# The inputs of the product test of newmark 1.7.1: one rock lognormal per
# period, its MCE quantiles and two geometry-material pairs.
Rock <- data.table::data.table(ID = "TEST", siteID = "TST", TR = 100, Tn = c(0, 0.2, 1),
                               Sa = c(0.3, 0.45, 0.24), Vs30 = 760, Vref = 760)
Rock[, `:=`(SaF = Sa, AF = NA_real_, k = "oq", muLn = log(Sa) - 0.4^2 / 2, sigmaLn = 0.4,
            w = 1, fitErr = 0)]
Levels <- c(0.05, 0.10, 0.16, 0.50, 0.84, 0.90, 0.95)
MCE <- Rock[, .(p = c(sprintf("%.2f", Levels), "mean"),
                Sa = c(exp(muLn + 0.4 * qnorm(Levels)), Sa)), by = Tn]
MCE[, `:=`(ID = "MCE", siteID = "TST", Vs30 = 760, Vref = 760)]
Shear <- data.table::data.table(IDg = c("G1", "G2"), IDm = c("U1", "U2"), VS30 = c(360, 560),
```

```
                                Ts = 0.3, Vref = 760, level = "mean")
Draws <- Shear[, .(IDg, IDm, realization = 1L, Ts, VS30, Vref)]
Site <- buildDnTables(Rock, MCE, Shear, modelID = "TEST", siteID = "TST", TR = 100, Vref = 760,
                      Mw = 7, NS = 2, weights = c(AM88 = 1, BM19 = 1), zTrunc = 3,
                      workers = 1, draws = Draws)
Site$DnTable[IDn == "ensemble" & p == "0.50" & ky < 0.02]
```

`buildGeometry` *Paired embankment geometries from height, truncation ratio and slope*

Description

Paired embankment geometries from height, truncation ratio and slope

Usage

```
buildGeometry(Hs, lo, s)
```

Arguments

`Hs`	Positive heights in metres.
`lo`	Truncation ratios strictly between 0 and 1, aligned with `Hs`.
`s`	Positive horizontal-to-vertical slopes, aligned with `Hs`.

Value

A data.table with `IDg`, `b`, `bmax`, `s`, `Hs` and `beta`.

Mathematics

The truncation ratio of the shear beam is $\lambda = b/(2H_s s + b)$ for a berm width b, height H_s and horizontal-to-vertical slope s; inverting it gives $b = 2H_s s/(1/\lambda - 1)$. The base width is $b + 2H_s s$ and the slope angle $\beta = \arctan(1/s)$ in degrees. Widths and heights are rounded to metres and slopes to two decimals, as in `newmark` 1.7.1; the identifier `IDg` is `S` followed by the rounded height.

Examples

```
buildGeometry(Hs = c(10, 120), lo = c(0.35, 0.075), s = c(2.5, 3.5))
```

buildKmaxTable *Yield-coefficient table of release 1.7.1 from displacement realizations*

Description

Compatibility route that reproduces, byte for byte once saved with saveRDS(), the site kmaxTable product of the fitKmaxProducts() function of newmark 1.7.1, so that the stored project tables can serve as comparison oracle. For every demand (each return period of TR and the MCE), geometry and material of the site, the realizations of every active model are inverted with solveKmax() and summarized, model by model and for each model family, with the pooled summary of summarizeKmax(). With one site, as in the current projects, this table is also the global kmaxTable of 1.7.1.

Usage

```
buildKmaxTable(draws, PGATable, shear, modelID, siteID, TR, NS, weights, DaH)
```

Arguments

draws	The DnDraws product of the site: IDg, IDm, ID, TR, ky, realization, PGA.s, IDn, Dn, siteID and units.Dn, in this order, the realizations of one cell sharing one ky grid.
PGATable	The PGATable product of the site as read back from its CSV: IDg, IDm, ID, TR, PGA, Vs30, siteID and units.PGA, in this order, with TR numeric and NA on the MCE rows. It is named after the product because PGA names the numeric acceleration elsewhere.
shear	The shear table of buildShearTable(); its heights Hs fix the levels and its mean rows supply Ts and VS30.
modelID	Identifier of the hazard model, the ID of the return-period demands.
siteID	Identifier of the site.
TR	Positive return periods of the model demands, in years.
NS	Number of realizations of every curve.
weights	Positive weights named by the active displacement models.
DaH	Distinct positive admissible displacements in percent of the height.

Value

A data.table keyed by siteID, ID, TR, IDg, IDm, IDn, p and Da, with columns IDg, IDm, p ("mean" or the probability as "%.2f"), ID (modelID or "MCE"), TR (years, NA for the MCE), Da (cm), kmax (g), the mean or quantile of the yield coefficient that limits the displacement to Da over the pooled realizations of the model or family IDn under that demand, geometry and material, PGA (g), Kh (ratio), beyond (weighted share of the pool beyond the stored grid), Ts (s), Vs30 (m/s), IDn, siteID and the unit columns units.Da ("cm"), units.kmax and units.PGA ("g") and units.Kh ("ratio"). The caller's tables are not modified.

Semantics reproduced from `newmark` 1.7.1

Every geometry is inverted at the same levels, the union over all the heights of the shear table of $D_a = D_{aH} H_s$ cm rounded to six significant digits, so a geometry's own levels are the rows whose `Da` equals `signif(DaH * Hs, 6)` for its height. Each model is summarized alone with weight 1; the families `ensemble`, `rigid` and `flexible` of the displacement models pool their active members with their weights normalized, and normalized once more by the summary. The levels are the mean and the probabilities 0.05, 0.10, 0.16, 0.50, 0.84, 0.90 and 0.95. The quantiles are those of `Hmisc::wtd.quantile()` as 1.7.1 called it: once the pool of one `Da` holds a tie (a saturated root of 0 or a root at the grid resolution), every pooled value is read back from its 15-significant-digit text before the quantile, while the mean uses the exact roots; `summarizeKmax()` has no such text step. `PGA` is the amplified peak ground acceleration of the cell as read back from the CSV of the PGA table, `Kh = kmax / PGA` is a dimensionless ratio (the exported `fitKmaxScenario()` of 1.7.1 returned `100 kmax / PGA`, which its product replaced by the ratio), and `Ts` and `Vs30` are the fundamental period of the shear beam and `VS30` of the mean row of the geometry and material in the shear table. The inputs are checked as 1.7.1 checked them (schema, 30 distinct `ky` per curve spanning 0.01 g to `max(PGA, 0.8)` g, one sampled PGA per realization, `NS` realizations of every active model, coverage of the PGA table); the result is checked and keyed by `siteID`, `ID`, `TR`, `IDg`, `IDm`, `IDn`, `p` and `Da`, and a warning reports the rows that rely on roots beyond the stored grid and the largest such `kmax`.

Examples

```
# The inputs of the product test of newmark 1.7.1: one rock lognormal per
# period, its MCE quantiles and two geometry-material pairs,
# here with the heights Hs from which the levels of DaH are derived.
Rock <- data.table::data.table(ID = "TEST", siteID = "TST", TR = 100, Tn = c(0, 0.2, 1),
                               Sa = c(0.3, 0.45, 0.24), Vs30 = 760, Vref = 760)
Rock[, `:=`(SaF = Sa, AF = NA_real_, k = "oq", muLn = log(Sa) - 0.4^2 / 2, sigmaLn = 0.4,
            w = 1, fitErr = 0)]
Levels <- c(0.05, 0.10, 0.16, 0.50, 0.84, 0.90, 0.95)
MCE <- Rock[, .(p = c(sprintf("%.2f", Levels), "mean"),
                Sa = c(exp(muLn + 0.4 * qnorm(Levels)), Sa)), by = Tn]
MCE[, `:=`(ID = "MCE", siteID = "TST", Vs30 = 760, Vref = 760)]
Shear <- data.table::data.table(IDg = c("G1", "G2"), IDm = c("U1", "U2"), Hs = c(10, 120),
                               VS30 = c(360, 560), Ts = 0.3, Vref = 760, level = "mean")
Weights <- c(AM88 = 1, BM19 = 1)
Draws <- Shear[, .(IDg, IDm, realization = 1L, Ts, VS30, Vref)]
Site <- buildDnTables(Rock, MCE, Shear, modelID = "TEST", siteID = "TST", TR = 100, Vref = 760,
                  Mw = 7, NS = 2, weights = Weights, zTrunc = 3, workers = 1, draws = Draws)
PGATable <- buildPGATable(Rock, MCE, Shear, modelID = "TEST", siteID = "TST", TR = 100)
Kmax <- buildKmaxTable(Site$DnDraws, PGATable = PGATable, shear = Shear, modelID = "TEST",
                   siteID = "TST", TR = 100, NS = 2, weights = Weights, DaH = c(0.05, 0.1))
Kmax[IDn == "ensemble" & p == "0.50"]
```

`buildKyGrid` *Yield-acceleration grid of a displacement scenario*

Description

Builds the log-spaced yield accelerations at which the displacement curves of one scenario are evaluated, from the calibration ranges of the displacement models and the mean peak ground acceleration of the scenario.

Usage

```
buildKyGrid(PGA, n)
```

Arguments

PGA	One finite positive mean peak ground acceleration of the scenario in g.
n	Number of grid points, one positive integer (the displacement products use 30).

Value

A numeric vector of n increasing yield accelerations in g.

Mathematics

For $n > 1$,

$$k_{y,i} = \exp\left(\ln 0.01 + \frac{i-1}{n-1}\left[\ln\max(\overline{PGA}, 0.80) - \ln 0.01\right]\right), \quad i = 1, \ldots, n$$

in g. For $n = 1$ the grid is the lower bound alone, $k_{y,1} = \exp(\ln 0.01)$, whatever the PGA. The lower bound 0.01 g is the lowest calibration point of BM17; the upper bound is the mean peak ground acceleration of the scenario, beyond which a rigid block does not slide, extended to the 0.80 g calibration ceiling of BM17 for low-intensity scenarios. The displacement products take $\overline{PGA}$ as the mixture mean $\sum_k w_k e^{\mu_k + \sigma_k^2/2}$ at zero period of an analytic spectrum, or the "mean" row at the smallest period of a quantile spectrum.

Examples

```
buildKyGrid(PGA = 0.5, n = 30)
```

---

buildKyLimits *Calibration range of yield acceleration of each displacement model*

Description

Tabulates the yield accelerations within which the regression of each displacement model was calibrated. The rigid model AM88 is bounded by the peak ground acceleration of the scenario; the others have absolute bounds. The displacement products evaluate every model on the common grid of buildKyGrid() and do not restrict it to these ranges.

Usage

```
buildKyLimits(PGA)
```

Arguments

PGA	One finite positive peak ground acceleration in g, the upper bound of AM88.

Value

A data.table with IDn and the bounds kyMin and kyMax in g.

Examples

```
buildKyLimits(PGA = 0.5)
```

---

buildPGATable *PGA table of release 1.7.1 from rock demands and a shear table*

Description

Compatibility route that reproduces, value for value, the PGATable product that the dn step of newmark 1.7.1 writes for one site, so that the stored products serve as comparison oracle while the calculations migrate. For every return period of TR and for the MCE, and for every geometry-material pair of the shear table, it evaluates the mean site PGA at the pair's VS30 with the ST20 factor and the numerical controls of that product.

Usage

```
buildPGATable(rock, MCE, shear, modelID, siteID, TR)
```

Arguments

rock	The UHSRock table of the site: columns ID, siteID, TR (years), Tn (s), k, muLn, sigmaLn, w, Vs30 and Vref (m/s).
MCE	The MCE table: columns ID, siteID, Tn (s), p, Sa (g), Vs30 and Vref (m/s), without TR.
shear	The shear table, see buildShearTable(): columns IDg, IDm, VS30 (m/s), Vref (760 m/s) and level.
modelID	One identifier of the probabilistic demand model (ID of rock).
siteID	One site identifier.
TR	Positive return periods in years; the return periods of rock for the model and site must be exactly these.

Value

A data.table with IDg, IDm, ID (modelID or "MCE"), TR (years, NA for the MCE), PGA (g, the mean site PGA, unrounded), Vs30 (m/s), siteID and units.PGA ("g"), one row per demand and pair, keyed as the product.

Semantics reproduced from newmark 1.7.1

The pairs are the level == "mean" rows of shear, whose VS30 is the site velocity; the rock condition is 760 m/s. A return period's rock demand is the analytic mixture of rock for modelID and siteID at that period, and its PGA is the mean $\sum we^{\mu+\sigma^2/2}$ of the rows at $|T_n| \leq 10^{-10}$ of the site mixture of refineSiteMixture() with the levels 0.05, 0.10, 0.16, 0.50, 0.84, 0.90 and 0.95, the orders 16, 32, 64, 128 and 256 and the tolerance 2×10^{-3}. The MCE demand is the quantile spectrum of MCE (ID == "MCE") for the site, and its PGA is the "mean" site ordinate at $|T_n| \leq 10^{-10}$ of refineSiteSpectrum() with the orders 32, 64, 128 and 256 and the same tolerance. Pairs with equal VS30 share one evaluation, which is deterministic. The table is keyed by siteID, ID, TR, IDg and IDm, as the product was written; data.table::fwrite(x, na = "NA") gives its CSV file.

Examples

```
# The inputs of the product test of newmark 1.7.1: one rock lognormal per
# period, its MCE quantiles and two geometry-material pairs.
Rock <- data.table::data.table(ID = "TEST", siteID = "TST", TR = 100, Tn = c(0, 0.2, 1),
                               Sa = c(0.3, 0.45, 0.24), Vs30 = 760, Vref = 760)
Rock[, `:=`(SaF = Sa, AF = NA_real_, k = "oq", muLn = log(Sa) - 0.4^2 / 2, sigmaLn = 0.4,
            w = 1, fitErr = 0)]
Levels <- c(0.05, 0.10, 0.16, 0.50, 0.84, 0.90, 0.95)
MCE <- Rock[, .(p = c(sprintf("%.2f", Levels), "mean"),
                Sa = c(exp(muLn + 0.4 * qnorm(Levels)), Sa)), by = Tn]
MCE[, `:=`(ID = "MCE", siteID = "TST", Vs30 = 760, Vref = 760)]
Shear <- data.table::data.table(IDg = c("G1", "G2"), IDm = c("U1", "U2"), VS30 = c(360, 560),
                                Ts = 0.3, Vref = 760, level = "mean")
buildPGATable(Rock, MCE, Shear, modelID = "TEST", siteID = "TST", TR = 100)
```

---

buildQuantileSpline *Quantile function of log spectral acceleration, monotone between its knots*

Description

Builds, from the tabulated quantiles of one spectral ordinate, the function $Q(u) = \ln S_a(u)$ that sampleSpectra() uses to map the uniforms of its Gaussian copula to accelerations when a spectrum is given by quantiles.

Usage

```
buildQuantileSpline(quantiles)
```

Arguments

`quantiles`	A table with the probability levels `p` (numbers strictly between 0 and 1, or text that converts to them) and the finite positive spectral accelerations `Sa` of one ordinate. Rows with `p` equal to `"mean"` are ignored; a missing `p` is rejected; at least three quantile rows are required. Any other column must hold a single value, so that the rows belong to one spectrum slice. Duplicate levels signal a warning and are collapsed by the spline.

Value

A function of probabilities u returning $\ln S_a$ (natural logarithm of the acceleration, in the units of `Sa`).

Mathematics

The quantile rows, sorted by probability, give the nodes $(p_i, \ln S_{a,i})$, $0 < p_i < 1$, with finite logarithms because every $S_{a,i}$ is finite and positive; a row outside these conditions stops the function. The ordinates are replaced by their running maximum, so that no node descends, and when two consecutive ordinates are then equal every ordinate is lifted by $i \times 10^{-8}$. The function is the cubic spline through these nodes with the monotonicity filter of Hyman (1983) (`stats::splinefun(method = "hyman")`). Its argument is clamped to $[0, 1]$; between 0 and the first tabulated probability, and between the last one and 1, the end pieces of the spline extrapolate. The monotonicity filter constrains the spline between the tabulated knots only: an end piece can descend (observed on the tail fixture of the site-response tests above $u = 0.9916$, on no slice the products sample). A change of the end pieces is a separate scientific candidate.

Examples

```
Q <- buildQuantileSpline(data.frame(p = c(0.16, 0.50, 0.84), Sa = c(0.3, 0.5, 0.8)))
exp(Q(c(0.16, 0.5, 0.84)))
```

`buildRootModel` *Build the characteristic-root model of one mode*

Description

The roots a_n of the truncated inhomogeneous shear beam are tabulated in CylinderRoots over the inhomogeneity exponent `m` and the truncation ratio `l`. Method `"exact"` solves the characteristic relation of the beam for every query; methods `"lm"` and `"nlm"` predict off-grid roots with a polynomial surrogate fitted once to the whole table of the mode. The root model is reused by `evaluateRoot()` for every query.

Usage

```
buildRootModel(mode, method)
```

Arguments

mode	One mode number, 1 to 8.
method	The solution method of the root model: "exact", or the surrogate "lm" or "nlm".

Value

The root model, a list with mode, method, table (the unique (l, m, an) rows of the mode), fit (the surrogate, NULL for "exact"), and the support loMin, loMax, moMin and moMax: for a surrogate the minimum and maximum l and m of the table, outside which predictions are extrapolations; for "exact" the domain of the relation, $0 \le \lambda < 1$ and $0 \le m < 2$.

Mathematics

For the shear beam of height H truncated at $h = \lambda H$ with modulus $G \propto z^m$, Dakoulas and Gazetas (1985) reduce the free vibration to a Bessel equation of order $q = m/(2 - m)$ and obtain the characteristic relation (hydrated copy, eq-free-characteristic)

$$J_{q+1}(a\lambda^{1-m/2})\, Y_q(a) - Y_{q+1}(a\lambda^{1-m/2})\, J_q(a) = 0,$$

whose ordered positive roots are a_n, $n = 1, 2, \ldots$; for $\lambda = 0$ the relation reduces to $J_q(a) = 0$. Method "exact" brackets the n-th sign change of the left-hand side on a grid of step 0.02 between $a = 0.05$ and $a = 100/(1 - \lambda)$ and refines it with stats::uniroot() to a tolerance of 10^{-10}, using R's fractional-order Bessel functions. The search is finite: a root above the limit (for example mode 6 at $m = 1.8$, $\lambda = 0.5$, near 258.8) is not found and evaluateRoot() stops with an error that names the limit.

With $l_2 = l^2$ and $m_2 = m^2$, the "lm" surrogate fits the Gaussian linear model $a_n \sim l + m + l_2 + m_2$ and the "nlm" surrogate adds the interactions lm and $l_2 m_2$. These fits are numerical interpolants of the table CylinderRoots, whose roots the "exact" method solves in R at every node of its grid; the fits themselves do not solve the characteristic relation. The decision-tree and random-forest surrogates of newmark 1.7.1 are not carried into this version.

Examples

```
RootModel <- buildRootModel(mode = 1, method = "nlm")
evaluateRoot(RootModel, mo = 0.45, lo = 0.20)
```

---

buildShearDraws *Shear-table realizations of release 1.7.1 with their beam periods*

Description

Draws the population of soil columns that buildShearTable() summarizes and keeps it: for every geometry crossed with every selected material (or only the supplied pairs), the realizations that buildShearTable() simulates, each with the first-mode period of the truncated shear beam evaluated on that realization, together with the shear table summarized from the same draws.

Usage

```
buildShearDraws(geometry, USCS, IDm, Vref, levels, pairs = NULL, NS = 100)
```

Arguments

geometry	A data.frame or data.table with IDg, Hs, b and s, such as buildGeometry() returns.
USCS	Named list of USCS code vectors, indexed by material IDm (the uscs argument of getTs()).
IDm	Character vector of selected material identifiers.
Vref	One positive reference velocity in m/s, recorded in the table.
levels	Quantile levels and/or the literal "mean".
pairs	Optional table of unique IDg/IDm pairs to evaluate, in the supplied order; NULL evaluates the full geometry by material product.
NS	Realizations per pair, a whole number of at least 100.

Details

Realizations 1 to 100 of every pair are drawn pair by pair, in the order of the table, with the calls, the rounding and the random numbers of buildShearTable(), and the shear table summarizes exactly those 100: under the same random stream it is the table that buildShearTable() returns. When NS exceeds 100, realizations 101 to NS of every pair are drawn after all the pairs have their first 100, in a second pass over the pairs in the same order, so the first 100 and the shear table do not depend on NS.

The period Ts of a realization is the beam expression of evaluateBeamPeriod() evaluated on its own mo and VSo with the truncation ratio lo of its geometry, rounded as on a summary row of the shear table: lo to 3 decimals, the first-mode root an (the "nlm" surrogate of buildRootModel() within its tabulated support, the exact root outside it) to 4 and Ts to 3. It is neither the Rayleigh period of the realization nor a statistic of the summary rows. The other observables of a realization are the ones buildShearTable() summarizes, with the rounding of the newmark 1.7.1 generator: Go to 2 decimals, mo to 3, and VSo, VS30, Z500 and Z1000 to 1. As a rule of this product, a realization whose fitted mo is negative, a column whose stiffness fit decreases with depth, is returned with mo forced to 0, the homogeneous beam, and with an and Ts evaluated at that 0, and a message reports how many realizations were forced, while the shear table summarizes the fitted mo. The root and the period are defined for mo below 2: a realization at or above it stops the function naming it before any period is evaluated.

Value

A list of two data.tables. ShearDraws has one row per pair and realization, ordered by pair as the shear table and then by realization, with the columns Hs, b, s, Vref, IDm, IDg, POP, Hw, USCS, Go, mo, Ts (fundamental period of the shear beam of the realization, s), VSo, VS30, Z500, Z1000, realization (an integer from 1), lo and an. ShearTable is the table that buildShearTable() describes, summarized from realizations 1 to 100 of every pair.

Examples

```
Geometry <- buildGeometry(Hs = 10, lo = 0.35, s = 2.5)
set.seed(20260915L)
Shear <- buildShearDraws(Geometry, USCS = list(U1 = c("ML", "MH")), IDm = "U1",
                         Vref = 760, levels = c(0.16, "mean", 0.84))
Shear$ShearDraws[, .(realization, mo, VSo, VS30, Ts)]
Shear$ShearTable
```

`buildShearTable` *Shear table of release 1.7.1 from geometries and material pools*

Description

Compatibility route that reproduces, value for value, the `ShearTable` product of the `getTs()` function of `newmark` 1.7.1 with its slope-period height correction, so that the corrected library can serve as comparison oracle while the calculations migrate. It crosses every geometry with every selected material (or only the supplied `pairs`), simulates 100 synthetic columns of 1 m layers per pair with a dry column, 100 kPa of preconsolidation and a 760 m/s completion velocity, summarizes their properties marginally and then evaluates the first-mode truncated-beam period on each summary row with the apex-to-base height $H_s/(1-\lambda_o)$ of `evaluateBeamPeriod()`. Shear tables built before that correction took H_s itself, so their unrounded periods are $1-\lambda_o$ times these. The first-mode root is the `"nlm"` surrogate of `buildRootModel()` where `mo` and `lo` lie within its tabulated support and the exact root of the characteristic relation outside it, where `newmark` 1.7.1 extrapolated the surrogate.

Usage

```
buildShearTable(geometry, USCS, IDm, Vref, levels, pairs = NULL)
```

Arguments

`geometry`	A data.frame or data.table with `IDg`, `Hs`, `b` and `s`, such as `buildGeometry()` returns.
`USCS`	Named list of USCS code vectors, indexed by material `IDm` (the `uscs` argument of `getTs()`).
`IDm`	Character vector of selected material identifiers.
`Vref`	One positive reference velocity in m/s, recorded in the table.
`levels`	Quantile levels and/or the literal `"mean"`.
`pairs`	Optional table of unique `IDg`/`IDm` pairs to evaluate, in the supplied order; `NULL` evaluates the full geometry by material product.

Value

A data.table with `Hs`, `b`, `s`, `Vref`, `IDm`, `IDg`, `POP`, `Hw`, `USCS`, `Go`, `mo`, `Ts` (fundamental period of the shear beam, s), `VSo`, `VS30`, `Z500`, `Z1000`, `level`, `lo` and `an`, one row per pair and level.

Semantics reproduced from `newmark` 1.7.1

The period of a summary row is the beam expression evaluated on the summarized `mo` and `VSo` of that row (`summarize-then-evaluate`), not a quantile of the period population; `propagateBeamPeriod()` gives the latter. The completion velocity of the upper 30 m is 760 m/s whatever `Vref` is, because `getTs()` did not forward it; `Vref` is only recorded. The layers take the plasticity exponent m_1 of 1.7.1, interpolated through `IP` 0, 20, 40, 60, 80 and 100 with 0, 0.18, 0.30, 0.41, 0.48 and 0.48 and `NA` above 100, instead of the published table that `evaluateSoilLayers()` uses by default. Before the summary, every realization takes the observables of the 1.7.1 generator: `Go` rounded to 2 decimals, `mo` and the Rayleigh period (`TrRayleigh` of `characterizeProfile()`, which 1.7.1 stored as `Ts`) to 3, `VS30` to 1, `VSo` derived from the rounded `Go` and rounded to 1, and the threshold depth ratios `Z500` and `Z1000`, the fitted normalized depth z/H_s at which the inverse regression $\log(z_m/H_s) = a + c \log V_S$ reaches 500 and 1000 m/s, rounded to 1 (a dimensionless ratio that 1.7.1 labelled as metres). The summaries then round `Go` to 1 decimal, `mo` to 3, `Ts` to 2, `VSo`, `VS30` and `Z500` to 1, and the quantile rows of `Z1000` to 1 while its mean row is unrounded; `lo` is rounded to 3 decimals before the root and the period use it, the root to 4 decimals and the beam period to 3. The realizations are summarized newest first, as the original accumulation did, so the means reproduce exactly. The general route (`characterizeProfile()`, `summarizeProfiles()`) rounds nothing and does not compute the threshold ratios. `buildShearDraws()` draws the same realizations, keeps them with the beam period of each one and returns this table beside them.

Examples

```
Geometry <- buildGeometry(Hs = 10, lo = 0.35, s = 2.5)
set.seed(20260915L)
buildShearTable(Geometry, USCS = list(U1 = c("ML", "MH")), IDm = "U1",
                Vref = 760, levels = c(0.16, "mean", 0.84))
```

`characterizeProfile` *Characterize one evaluated soil column*

Description

Reduces the layers of one column to the profile quantities used by the shear-beam vibration and site-response calculations: the travel-time velocity of the upper 30 m, its site class, the fitted power law of the stiffness profile, the basal velocity, the constant-density Rayleigh estimate of the site period and the thickness composition.

Usage

```
characterizeProfile(layers, Hs, Vref)
```

Arguments

`layers`	The result of `evaluateSoilLayers()`, or a table with columns `USCS`, `hs`, `zm`, `gsat`, `Gm` and `VSm`.
`Hs`	One positive column height in metres.

`Vref` One positive reference velocity in m/s completing the upper 30 m when the column is shallower; see `estimateVS30()`.

Value

A one-row data.table: `NL` (layers), `Go` (MPa), `mo`, `VSo` and `VS30` (m/s), `SID`, `TrRayleigh` (s, the Rayleigh estimate of the site period with constant density of `estimateRayleighPeriod()`, as `newmark` 1.7.1 computed it per realization), `Gravels`, `Sands`, `Fines`, `Clays`, `Silts`, `Organic` (percent of thickness), `USCS` (sorted classes present, joined by `.`), `fitSigma` and `fitLayers`. Every value is a conditional evaluation of the supplied layers.

Mathematics

The stiffness profile is fitted by least squares in logarithmic coordinates over the layers with positive modulus,

$$\log G_m = \log G_o + m_o \log(z_m/H_s),$$

so that G_o is the modulus at the base $z = H_s$ and m_o the exponent of a profile whose origin is the crest, z being the depth below it. `evaluateBeamPeriod()` uses m_o with the roots of the truncated shear beam, whose profile has its origin at the apex of the extended slopes, as an approximation. The basal velocity uses the saturated unit weight of the deepest layer, $V_{So} = \sqrt{9.81 \cdot 1000 G_o/\gamma_{sat,N_L}}$. The composition columns are thickness percentages.

No value is rounded: every quantity is returned at the precision of its calculation. The rounding of `newmark` 1.7.1, its basal velocity derived from a rounded modulus and its threshold depth ratios `Z500` and `Z1000` belong to the compatibility route of `buildShearTable()`. `fitSigma` is the residual standard error of the stiffness fit in log units, a numerical diagnostic separate from any physical dispersion.

Examples

```
set.seed(1)
Layers <- evaluateSoilLayers(sampleSoilLayers(Hs = 10, USCS = c("ML", "MH"), h = 1),
                             Hs = 10, water = 0, POP = 100)
characterizeProfile(Layers, Hs = 10, Vref = 760)
```

---

classifySite *ASCE 7-22 site class of a shear-wave velocity*

Description

Classifies V_{S30} by the intervals of ASCE/SEI 7-22 Table 20.2-1: A at or above 1500 m/s, B from 910, BC from 640, C from 440, CD from 300, D from 210, DE from 150 and E below 150 m/s. The lower bound of each interval belongs to it. Class F requires site-specific criteria and is not assigned from a velocity.

Usage

```
classifySite(VS30)
```

Arguments

VS30	Finite non-negative velocities in m/s.

Value

A character vector of class identifiers aligned with VS30.

Examples

```
classifySite(c(120, 150, 790, 910, 3000))
```

---

convertAEP *Probability of exceedance and return period of an annual rate*

Description

Probability of exceedance and return period of an annual rate

Usage

```
convertAEP(AEP, IT)
```

Arguments

AEP	Annual rates of exceedance in 1/year.
IT	Investigation time in years, finite and positive: one value or one per AEP.

Value

A data.table with POE (probability within IT) and TR (years), one row per AEP.

Mathematics

Under Poisson occurrence an annual rate AEP gives, over an investigation time IT, the probability of at least one exceedance and the return period

$$POE = 1 - e^{-IT \cdot AEP}, \qquad TR = 1/AEP.$$

The conversion is exact; it carries no dispersion.

Examples

```
convertAEP(AEP = 1 / unique(VancouverSpectra$TR), IT = 50)
```

`correlateResiduals` *Correlation of spectral residuals between two oscillator periods*

Description

Evaluates the Baker and Jayaram (2008) correlation between the logarithmic residuals of spectral acceleration at two oscillator periods, the coefficient that couples the ordinates of a correlated spectral sample (`sampleSpectra()`). It depends on the pair of periods only, not on magnitude or distance.

Usage

```
correlateResiduals(Tn, T0)
```

Arguments

`Tn`, `T0`	Oscillator periods in seconds: finite numeric vectors of equal length, or one of them of length one.

Value

Dimensionless correlation coefficients, one per period pair.

Mathematics

With $T_{min} = \min(T_n, T_0)$ and $T_{max} = \max(T_n, T_0)$, equations 5 and 6 of Baker and Jayaram (2008) are

$$C_1 = 1 - \cos\left(\frac{\pi}{2} - 0.366 \ln \frac{T_{max}}{\max(T_{min}, 0.109)}\right)$$

$$C_2 = 1 - 0.105 \left(1 - \frac{1}{1 + e^{100 T_{max} - 5}}\right) \frac{T_{max} - T_{min}}{T_{max} - 0.0099}$$

when $T_{max} < 0.2$ and $C_2 = 0$ otherwise, $C_3 = C_2$ when $T_{max} < 0.109$ and $C_3 = C_1$ otherwise,

$$C_4 = C_1 + \frac{1}{2}\left(\sqrt{C_3} - C_3\right)\left(1 + \cos\frac{\pi T_{min}}{0.109}\right)$$

and the correlation is C_2 when $T_{max} < 0.109$, C_1 when $T_{min} > 0.109$, $\min(C_2, C_4)$ when $T_{max} < 0.2$ and C_4 otherwise, tried in that order. The coefficient is symmetric in the two periods. The fit is calibrated for periods from 0.01 to 10 s; a pair outside that range is still evaluated and a warning is signalled. As in `newmark` 1.7.1, a period of zero or less stands for peak ground acceleration and is evaluated at 0.01 s, a convention of the package and not of the paper.

References

Baker, J. W. and Jayaram, N. (2008). Correlation of spectral acceleration values from NGA ground motion models. *Earthquake Spectra*, 24(1), 299-317. doi:10.1193/1.2857544

Examples

```
correlateResiduals(Tn = c(0, 0.3, 1.5), T0 = 0.01)
```

`CylinderRoots` *Characteristic roots of the truncated inhomogeneous shear beam*

Description

Roots a_n of the characteristic equation of Dakoulas and Gazetas (1985), a shear beam whose modulus grows with depth as a power law of exponent m and whose crest truncates the full wedge at the ratio λ. Every root is solved in R at its node of the grid by the exact method of `buildRootModel()` (`data-raw/CylinderRoots.R`, recorded in `data-raw/README.md`). The grid, modes 1 to 8 with m from 0 to 0.95 in steps of 0.005 and λ from 0 to 0.5 in steps of 0.0025, is that of the tables of `newmark` 1.7.1, computed with Mathematica. The surrogates of `buildRootModel()` are fitted to this table.

Usage

```
CylinderRoots
```

Format

A data.table with 307128 rows and 4 variables, ordered by `n`, `m` and `l`:

n Integer: mode number, 1 to 8.

m Double: inhomogeneity exponent, 0 to 0.95.

l Double: truncation ratio, 0 to 0.5.

an Double: characteristic root of mode `n`.

Source

Dakoulas, P. and Gazetas, G. (1985). A class of inhomogeneous shear models for seismic response of dams and embankments. Soil Dynamics and Earthquake Engineering 4(4), 166-182.

`describeModule` *Describe the installed module*

Description

Diagnoses the installation of this module without repairs; the installed command prints this report for `doctor`.

Usage

```
describeModule(resources = NULL)
```

Arguments

resources	The resource directory of the installed command, or NULL outside an installed product.

Value

A list with schema, package, version, summary and checks. Each check has name, required, status (ok or fail) and detail. The checks library, exports, packages, tools and receipt are required: the library loads with the version of lib/DESCRIPTION, the declared exports exist, the declared packages load, the declared tools are on PATH and the files of the installation receipt keep their recorded MD5.

Examples

```
describeModule()
```

---

estimateAI *Arias intensity from peak ground acceleration*

Description

Evaluates the in-house rock regression of newmark 1.7.1 that gives each sampled peak ground acceleration its Arias intensity, the intensity the rigid models JB07 and SR08 of evaluateRigidDisplacement() read.

Usage

```
estimateAI(PGA)
```

Arguments

PGA	Peak ground accelerations in g, finite and positive.

Value

Arias intensities in m/s, aligned with PGA.

Mathematics

$$I_a = PGA^{1.9228}\, e^{2.6109}$$

with PGA in g and I_a in m/s, a median relation without dispersion, fitted on the rock records of Verri (2023), as recorded in newmark 1.7.1. It is evaluated realization by realization, so a sampled intensity stays paired with its PGA.

References

Verri, A. D. (2023). Modelos Parametricos para la Seleccion de Sismos de Diseno en Infraestructura Minera. Seminario de Vinculacion y Transferencia (SeVyT) 2023.

Examples

```
estimateAI(PGA = c(0.1, 0.5))
```

`estimatePGV` *Peak ground velocity from peak ground acceleration*

Description

Evaluates the in-house rock regression of `newmark` 1.7.1 that gives each sampled peak ground acceleration its peak ground velocity, the velocity the pulse equations of `BM19` in `evaluateFlexibleDisplacement()` read.

Usage

```
estimatePGV(PGA)
```

Arguments

`PGA`	Peak ground accelerations in g, finite and positive.

Value

Peak ground velocities in cm/s, aligned with `PGA`.

Mathematics

$$PGV = 100\, PGA^{1.0529}\, e^{0.1241}$$

with PGA in g and PGV in cm/s, a median relation without dispersion, fitted on the rock records of Verri (2023), as recorded in `newmark` 1.7.1. It is evaluated realization by realization, so a sampled PGV stays paired with its PGA.

References

Verri, A. D. (2023). Modelos Parametricos para la Seleccion de Sismos de Diseno en Infraestructura Minera. Seminario de Vinculacion y Transferencia (SeVyT) 2023.

Examples

```
estimatePGV(PGA = c(0.1, 0.5))
```

`estimateRayleighPeriod` *Fundamental period of a layered soil profile by the Rayleigh quotient*

Description

Fundamental period of a layered soil profile by the Rayleigh quotient

Usage

```
estimateRayleighPeriod(VSm, hs, zm)
```

Arguments

`VSm`	Positive layer shear-wave velocities in m/s.
`hs`	Positive layer thicknesses in metres, aligned with `VSm`.
`zm`	Layer midpoint depths from the surface in metres, aligned.

Value

One estimate of the fundamental period in seconds, unrounded.

Mathematics

For a fixed-base, free-surface column of uniform density under a uniform horizontal body force, the shear strain at depth z is proportional to $z/V_S(z)^2$. Integrating it from the base gives the static deflection, used as trial mode shape X_k at each layer midpoint (Dobry, Oweis and Urzua 1976), and the Rayleigh quotient yields

$$T_r = 2\pi\sqrt{\frac{\sum_k h_k X_k^2}{\sum_k h_k X_k}}.$$

Layers are ordered from the base internally, so the input order does not change the result. The midpoint approximation depends on the subdivision of the profile: for a uniform column of height H the estimate tends to $3.974H/V_S$ as the layers become thin, against the exact $4H/V_S$, while one single layer gives $\pi H/V_S$. It is a conditional evaluation of the supplied profile and carries no dispersion.

Examples

```
estimateRayleighPeriod(VSm = c(200, 200), hs = c(10, 10), zm = c(5, 15))
```

`estimateVS30` *Time-averaged shear-wave velocity of the upper 30 m*

Description

Time-averaged shear-wave velocity of the upper 30 m

Usage

```
estimateVS30(hs, VSm, Vref)
```

Arguments

`hs`	Positive layer thicknesses in metres, from the surface down.
`VSm`	Positive layer shear-wave velocities in m/s, aligned with `hs`.
`Vref`	One positive velocity in m/s of the material below the column, used only when the column is shallower than 30 m.

Value

One velocity in m/s, unrounded.

Mathematics

With layer thicknesses h_k from the surface and velocities V_k, the travel time through the part of each layer inside the upper 30 m is accumulated and

$$V_{S30} = \frac{30}{\sum_k h_k^{(30)}/V_k + \max(30 - \sum_k h_k, 0)/V_{ref}},$$

where the second term completes a column shallower than 30 m with the reference velocity `Vref` and vanishes otherwise. The result is the harmonic average of the velocities weighted by thickness (ASCE/SEI 7-22 Eq. 20.4-1), evaluated at the supplied profile; it carries no dispersion.

Examples

```
estimateVS30(hs = c(10, 20), VSm = c(200, 400), Vref = 760)   # 300
estimateVS30(hs = 10, VSm = 200, Vref = 760)                  # padded
```

`estimateWeightedRayleighPeriod` *Fundamental period of a layered soil profile by the density-weighted Rayleigh quotient*

Description

Fundamental period of a layered soil profile by the density-weighted Rayleigh quotient

Usage

```
estimateWeightedRayleighPeriod(VSm, hs, zm, gsat)
```

Arguments

`VSm`	Positive layer shear-wave velocities in m/s.
`hs`	Positive layer thicknesses in metres, aligned with `VSm`.
`zm`	Layer midpoint depths from the surface in metres, aligned; they order the layers, the overburden is accumulated from `hs`.
`gsat`	Positive layer unit weights in kN/m3, aligned; any unit cancels, only the ratios between layers matter.

Value

One estimate of the fundamental period in seconds, unrounded.

Mathematics

For a shear beam whose density $\rho(z)$ and velocity $V(z)$ vary with depth, Dobry, Oweis and Urzua (1976) write the Rayleigh quotient

$$\omega^2 = \frac{\int_0^H \rho V^2 (dX/dz)^2 \, dz}{\int_0^H \rho X^2 \, dz}$$

and obtain the first-iteration mode shape from a unit initial shape,

$$X(z) = \int_0^z \frac{1}{\rho(z_a) V^2(z_a)} \int_{z_a}^H \rho(z_b) \, dz_b \, dz_a,$$

with z measured from the base (hydrated copy, equations `eq-rayleigh-frequency` and `eq-rayleigh-mode-iteration`). The slope of that shape is the overburden mass above the point divided by ρV^2, so the numerator integrates by parts into $\int_0^H \rho X \, dz$. Evaluated at layer midpoints with layer-wise constant properties, and with g cancelling between ρ and the unit weight γ,

$$T = 2\pi \sqrt{\frac{\sum_k \gamma_k h_k X_k^2}{\sum_k \gamma_k h_k X_k}}, \qquad X_k' = \frac{\sigma_k}{\gamma_k V_k^2},$$

where σ_k is the total overburden at the midpoint of layer k, accumulated from the thicknesses and unit weights of the layers above it, and X_k the deflection accumulated from the base. With a uniform unit weight the expression reduces to `estimateRayleighPeriod()`, the constant-density form the paper calls Method 7. The result is a conditional evaluation of the supplied profile and carries no dispersion.

Examples

```
estimateWeightedRayleighPeriod(VSm = c(150, 300), hs = c(10, 10), zm = c(5, 15),
                               gsat = c(16, 20))
```

`evaluateBeamPeriod` *Modal period of the inhomogeneous truncated shear beam*

Description

Modal period of the inhomogeneous truncated shear beam

Usage

```
evaluateBeamPeriod(Hs, lo, mo, VSo, an)
```

Arguments

`Hs`	Slope heights from crest to base, in metres.
`lo`	Truncation ratios λ_o in $[0, 1)$, aligned with `Hs`; see `buildGeometry()`.
`mo`	Inhomogeneity exponents, below 2.
`VSo`	Basal shear-wave velocities in m/s.
`an`	Characteristic roots of the requested mode at `mo` and `lo`.

Value

Periods in seconds, aligned with the inputs, unrounded.

Mathematics

Dakoulas and Gazetas (1985) measure the depth z from the apex where the extended slopes meet: the crest lies at $z = h$, the base at $z = H$, the modulus is $G = G_b(z/H)^m$ and the truncation ratio is $\lambda = h/H$ (hydrated copy, Fig. 1 on printed page 167 and equations `eq-modulus-depth-power` and `eq-free-truncation-ratio`). The slope of `buildGeometry()` has height $H_s = H - h$ from crest to base, and its truncation ratio $\lambda_o = b/b_{max}$ is that h/H, because the extended slopes meet $h = b/(2s)$ above the crest; hence $H_s = H(1 - \lambda_o)$. The period of the mode whose characteristic root is $a_n(m_o, \lambda_o)$, with basal velocity V_{So}, is (`eq-natural-frequency-period`)

$$T_n = \frac{4\pi H}{a_n(2 - m_o)V_{So}}, \qquad H = \frac{H_s}{1 - \lambda_o}.$$

With exact roots it reproduces the shear-beam periods of the five dams of Table 8 of the paper (printed page 175) within 0.1 percent from their printed height, truncation ratio, exponent and average velocity.

The exponent m_o and velocity V_{So} that this package estimates come from the fit of `characterizeProfile()`, whose origin is the crest ($G = G_o(z/H_s)^{m_o}$, z below the crest), while the roots belong to the apex-origin profile of the paper. Evaluating the period of a fitted profile this way is an approximation. The theory base measures it on a continuous profile at 1.4, 3.7 and 6.0 percent for $m = 0.5$ and λ = 0.075, 0.20 and 0.35; for a layered realization the measure is the difference that `propagateBeamPeriod()` reports against the numerical period, with its divergence flag.

The result is a conditional evaluation of its five inputs; evaluating it on summary statistics of m_o and V_{So} does not give the statistics of the period, which `propagateBeamPeriod()` obtains realization by realization.

Examples

```
# Dam A of Table 8 of Dakoulas and Gazetas (1985): 0.534 s, with the basal
# velocity derived from its average velocity of 200 m/s and the exact root.
evaluateBeamPeriod(Hs = 40, lo = 0.043, mo = 0.57, VSo = 228.25, an = 3.0123)
```

---

evaluateDisplacements *Displacement realizations of the models of a scenario*

Description

Evaluates, for every requested displacement model, yield acceleration and realization of a scenario, the Newmark displacement of that realization. One standard normal deviate per realization is shared by every model and every yield acceleration, so each realization traces one coherent displacement curve $D_{n,s}(k_y)$ per model and the models stay paired realization by realization.

Usage

```
evaluateDisplacements(intensities, eps, ky, Ts, Mw, models, NFC, pulse)
```

Arguments

`intensities`	Paired realizations of one scenario with columns `realization`, `PGA`, `Sa13`, `Sa15`, `PGV` and `AI`, as `sampleIntensities()` returns; `realization` identifies each realization and must be non-missing and unique.
`eps`	Standard normal deviates of the displacement models, one per row of `intensities`, finite.
`ky`	Yield accelerations in g, finite and positive.
`Ts`	One initial fundamental period of the sliding mass in seconds, the one the intensities were sampled for.
`Mw`	One moment magnitude of the scenario, read by the flexible models.
`models`	Distinct model identifiers among `AM88`, `JB07`, `SR08`, `BT07`, `BM17` and `BM19`.
`NFC`, `pulse`	Near-fault component and pulse declaration of the `BM19` model, see `evaluateFlexibleDisplacement()`; read only when `models` includes `BM19`.

Value

A data.table with IDn, ky (g), realization and Dn (cm), one row per model, yield acceleration and realization, ordered by the order of models, then ky, then the rows of intensities. Each Dn is one realization, not a statistic: the population is the realizations of the scenario, identified by the caller's realization and paired across models and yield accelerations, conditional on a nonzero displacement for the flexible models; summarizeDisplacements() and summarizeEnsemble() reduce it to quantiles and means.

Mathematics

$$D_{n,s}(k_y) = \exp\left(\mu_{\ln D}(k_y, IM_s) + \sigma_{\ln D}(k_y, IM_s)\,\varepsilon_s\right)$$

in cm, where IM_s are the intensity measures of realization s (sampleIntensities()), ε_s its deviate, and $\mu_{\ln D}$, $\sigma_{\ln D}$ the conditional lognormal parameters of each model: evaluateRigidDisplacement() for AM88 (PGA), JB07 and SR08 (PGA, I_a), and evaluateFlexibleDisplacement() for BT07 and BM17 ($S_a(1.5T_s)$) and BM19 ($S_a(1.3T_s)$, PGV). The calculation is deterministic; the caller supplies the deviates. For BT07, BM17 and BM19 these are the parameters of the nonzero branch of the published models (evaluateFlexibleDisplacement()): their probability of zero or negligible displacement is omitted, so every realization of those models is a positive displacement drawn conditional on a nonzero displacement.

Examples

```
Sigma <- c(0.2, 0.45, 0.75)
Spectrum <- data.frame(Tn = c(0, 0.4, 1), k = "oq", muLn = log(c(0.8, 0.55, 0.42)) - Sigma^2 / 2,
                       sigmaLn = Sigma, w = 1)
set.seed(46)
Intensities <- sampleIntensities(Spectrum, Ts = 0.4, NS = 5, zTrunc = NA)
evaluateDisplacements(Intensities, eps = stats::rnorm(5), ky = c(0.05, 0.1), Ts = 0.4, Mw = 7,
                      models = c("AM88", "BM19"), NFC = "D100", pulse = FALSE)
```

---

evaluateFlexibleDisplacement

Deformable sliding-mass displacement

Description

Evaluates the lognormal distribution of the nonzero seismic displacement of a deformable sliding mass with yield coefficient k_y and initial fundamental period T_s under an earthquake of moment magnitude M_w, with one of the three flexible models of the displacement ensemble, named by its author-year identifier model. The spectral acceleration is read at the degraded period of the model: $1.5T_s$ for BT07 and BM17, $1.3T_s$ for BM19.

Usage

```
evaluateFlexibleDisplacement(model, ky, Ts, Sa, Mw, PGV, NFC, pulse)
```

Arguments

model	One flexible model identifier: "BT07", "BM17" or "BM19".
ky	Yield coefficients in g, finite and positive.
Ts	The initial fundamental period of the sliding mass in seconds, one finite non-negative value.
Sa	Spectral accelerations in g at the degraded period of the model, finite and positive.
Mw	One finite positive moment magnitude.
PGV	Peak ground velocities in cm/s, finite and positive, aligned with Sa, for example estimatePGV() of each PGA; only the pulse equations of BM19 read them.
NFC	Near-fault component of the BM19 pulse equations, "D100" or "D50".
pulse	One logical: TRUE for a near-fault pulse motion of BM19, FALSE for ordinary motions.

Details

Each published model treats the displacement as a mixed random variable: a probability of negligible ("zero") displacement, with a threshold of 1 cm in BT07 and 0.5 cm in BM17 and BM19, and a lognormal distribution of the nonzero displacement. This function evaluates only the nonzero branch, equation 14.5 of Bray (2007, p. 344), equation 4 of Bray, Macedo and Travasarou (2018, p. 5) and equations 6, 8 and 10 of Bray and Macedo (2023, pp. 6-7), with the D50 correction described below, and omits the probability of zero displacement of equation 14.4 (p. 343), equations 2 and 3 (p. 5) and equations 5, 7 and 9 (pp. 6-7) of the same sources. Its result is the distribution of D_n conditional on the inputs and on a nonzero displacement.

ky and Sa are evaluated element by element: one of length one is repeated for every element of the other, otherwise both must have the same length, and PGV has the length of Sa; the function stops and names the lengths otherwise.

Value

A data.table with IDn (the model), muLnD and sigmaLnD, the mean and standard deviation of $\ln D_n$ (D_n in cm) given the inputs and a nonzero displacement: $e^{\mu_{\ln D}}$ is the median nonzero displacement and $\sigma_{\ln D}$ its dispersion in natural logarithms. One row per element of the longer of ky and Sa.

Mathematics

With D_n in cm, S_a in g, T_s in s and $x = \ln k_y$, $y = \ln S_a$:

BT07, Bray and Travasarou (2007), equation 5, with $a_0 = -1.10$, replaced by -0.22 for nearly rigid masses ($T_s < 0.05$ s):

$$\ln D_n = a_0 - 2.83x - 0.333x^2 + 0.566xy + 3.04y - 0.244y^2 + 1.50T_s + 0.278(M_w - 7), \qquad \sigma = 0.66$$

BM17, Bray, Macedo and Travasarou (2018), equation 4, subduction interface earthquakes, with $(c_1, c_2, c_3) = (-5.864, -9.421, 0)$ for $T_s < 0.1$ s and $(-6.896, 3.081, -0.803)$ otherwise:

$$\ln D_n = c_1 + 0.550M_w + c_2T_s + c_3T_s^2 - 3.353x - 0.390x^2 + (3.060 + 0.538x)y - 0.225y^2, \qquad \sigma = 0.73$$

`BM19`, Bray and Macedo (2019), using the 2023 equations with the D50 intercept from the 2021 erratum, for shallow crustal earthquakes. Ordinary motions (`pulse = FALSE`, equation 10 of the 2023 paper):

$$\ln D_n = a_0 + (2.649 + 0.344x)y - 0.090y^2, \qquad \sigma = 0.72$$

with $a_0 = -4.684 + 0.603M_w - 9.471T_s - 2.482x - 0.244x^2$ for $T_s < 0.1$ s and $a_0 = -5.981 + 0.603M_w + 3.223T_s - 0.945T_s^2 - 2.482x - 0.244x^2$ otherwise. Near-fault forward-directivity pulse motions (`pulse = TRUE`), a property of the scenario declared by the analyst:

$$\ln D_n = c_1 - b_1x - b_2x^2 + b_3xy + b_4y - b_5y^2 + c_2T_s + c_3T_s^2 + b_mM_w + c_4 \ln PGV$$

with PGV in cm/s and the coefficients of equation 6 for the maximum component (`NFC = "D100"`: $b = (2.632, 0.278, 0.527, 1.978, 0.233)$, $b_m = 0.01$, $\sigma = 0.56$) or equation 8 for the median component (`"D50"`: $b = (2.931, 0.319, 0.584, 2.261, 0.241)$, $b_m = 0.05$, $\sigma = 0.54$). c_1 and c_4 change at PGV = 150 cm/s, c_1, c_2, c_3 at $T_s = 0.1$ s: D100 $c_1 = -6.724/1.991$ and $(c_2, c_3) = (-2.744, 0)$ below 0.1 s, $c_1 = -6.951/1.764$ and $(1.069, -0.498)$ above, $c_4 = 1.547/-0.097$; D50 $c_1 = -7.497/-0.148$ and $(-2.731, 0)$, $c_1 = -7.718/-0.369$ and $(1.031, -0.480)$, $c_4 = 1.458/0.025$ (PGV up to / above 150 cm/s). For D50 above 150 cm/s and $T_s < 0.1$ s, $c_1 = -0.148$ follows the 2021 erratum to equation 7 of the 2019 paper. Equation 8 of the 2023 paper instead prints $c_1 = 2.480$; that value is not used here.

References

Bray, J. D. and Travasarou, T. (2007). Simplified procedure for estimating earthquake-induced deviatoric slope displacements. *Journal of Geotechnical and Geoenvironmental Engineering*, 133(4), 381-392. doi:10.1061/(ASCE)10900241(2007)133:4(381)

Bray, J. D. (2007). Simplified seismic slope displacement procedures. In K. D. Pitilakis (ed.), *Earthquake Geotechnical Engineering*, 327-353. Springer.

Bray, J. D., Macedo, J. and Travasarou, T. (2018). Simplified procedure for estimating seismic slope displacements for subduction zone earthquakes. *Journal of Geotechnical and Geoenvironmental Engineering*, 144(3), 04017124. doi:10.1061/(ASCE)GT.19435606.0001833

Bray, J. D. and Macedo, J. (2019). Procedure for estimating shear-induced seismic slope displacement for shallow crustal earthquakes. *Journal of Geotechnical and Geoenvironmental Engineering*, 145(12), 04019106.

Bray, J. D. and Macedo, J. (2021). Erratum for "Procedure for estimating shear-induced seismic slope displacement for shallow crustal earthquakes". *Journal of Geotechnical and Geoenvironmental Engineering*, 147(2). doi:10.1061/(ASCE)GT.19435606.0002476

Bray, J. D. and Macedo, J. (2023). Performance-based seismic assessment of slope systems. *Soil Dynamics and Earthquake Engineering*, 168, 107835. doi:10.1016/j.soildyn.2023.107835 The `BM19` coefficients implemented here follow its equations 6, 8 and 10, except for the D50 intercept described above.

Examples

```
evaluateFlexibleDisplacement(model = "BM19", ky = 0.1, Ts = 0.3, Sa = 0.8, Mw = 7,
                             PGV = estimatePGV(0.5), NFC = "D100", pulse = FALSE)
```

`evaluateRigidDisplacement` *Rigid sliding-block displacement*

Description

Evaluates the conditional lognormal distribution of the Newmark displacement of a rigid block with yield acceleration k_y under a ground motion of peak acceleration PGA and Arias intensity I_a, with one of the three rigid models of the displacement ensemble, named by its author-year identifier `model`.

Usage

```
evaluateRigidDisplacement(model, PGA, ky, AI)
```

Arguments

`model`	One rigid model identifier: `"AM88"`, `"JB07"` or `"SR08"`.
`PGA`	Peak ground accelerations in g, finite and positive.
`ky`	Yield accelerations in g, finite and positive.
`AI`	Arias intensities in m/s, finite and positive, for example `estimateAI()` of each PGA; `AM88` does not read them.

Value

A data.table with `IDn` (the model), `muLnD` and `sigmaLnD`, the mean and standard deviation of $\ln D_n$ (D_n in cm), one row per element of the longest vector the model reads.

Mathematics

With $r = k_y/PGA$, D_n in cm, PGA in g and I_a in m/s:

`AM88`, Ambraseys and Menu (1988), with the ratio capped at 0.9999 so that $1 - r > 0$ when the yield acceleration reaches the peak:

$$\log_{10} D_n = 0.90 + \log_{10}\left[(1-r)^{2.53} r^{-1.09}\right], \qquad \sigma_{\log_{10} D_n} = 0.30$$

`JB07`, Jibson (2007), equation 10:

$$\log_{10} D_n = 0.561 \log_{10} I_a - 3.833 \log_{10} r - 1.474, \qquad \sigma_{\log_{10} D_n} = 0.616$$

`SR08`, Saygili and Rathje (2008), equation 6 with the (PGA, I_a) coefficients of tables 1 and 2 (the authors recommend the (PGA, PGV) vector; this published variant keeps the rigid models on Arias intensity):

$$\ln D_n = 2.39 - 5.24r - 18.78r^2 + 42.01r^3 - 29.15r^4 - 1.56 \ln PGA + 1.38 \ln I_a, \qquad \sigma_{\ln D_n} = 0.46 + 0.56r$$

The base-10 results are returned in natural logarithms ($\times \ln 10$).

The vectors a model reads, `PGA` and `ky` and, for `JB07` and `SR08`, `AI`, are evaluated element by element: a vector of length one is repeated for every element of the others, and the vectors longer than one must have one common length; otherwise the function stops and names the lengths.

References

Ambraseys, N. N. and Menu, J. M. (1988). Earthquake-induced ground displacements. *Earthquake Engineering & Structural Dynamics*, 16(7), 985-1006. doi:10.1002/eqe.4290160704

Jibson, R. W. (2007). Regression models for estimating coseismic landslide displacement. *Engineering Geology*, 91(2-4), 209-218. doi:10.1016/j.enggeo.2007.01.013

Saygili, G. and Rathje, E. M. (2008). Empirical predictive models for earthquake-induced sliding displacements of slopes. *Journal of Geotechnical and Geoenvironmental Engineering*, 134(6), 790-803. doi:10.1061/(ASCE)10900241(2008)134:6(790)

Examples

```
evaluateRigidDisplacement(model = "SR08", PGA = 0.5, ky = c(0.05, 0.1), AI = estimateAI(0.5))
```

`evaluateRoot` *Characteristic root for given inhomogeneity and truncation*

Description

With a surrogate root model, returns the tabulated root when the query falls exactly on the grid of CylinderRoots and the surrogate prediction otherwise; with the `"exact"` method, solves the characteristic relation of Dakoulas and Gazetas (1985) for every query (see `buildRootModel()`). The diagnostics tell a consumer where each value comes from.

Usage

```
evaluateRoot(rootModel, mo, lo)
```

Arguments

`rootModel`	The root model that `buildRootModel()` returns.
`mo`	Inhomogeneity exponents, aligned with `lo`; below 2 for `"exact"`.
`lo`	Truncation ratios $\lambda = b/(2H_s s + b)$, aligned; in $[0, 1)$ for `"exact"`.

Value

A data.table with `mo`, `lo`, `an` (the root), `exact` (`TRUE` when read from the table or solved from the relation) and `withinSupport` (`FALSE` when `lo` or `mo` lies outside the tabulated range of a surrogate, below its minimum or above its maximum, so the value is an extrapolation; both minima are 0 in CylinderRoots). A surrogate root is a numerical interpolation whose error is not characterized; an `"exact"` root is converged to 10^{-10}.

Examples

```
RootModel <- buildRootModel(mode = 1, method = "nlm")
# A node of CylinderRoots, a query between its nodes and one beyond its largest m.
Query <- data.frame(mo = c(0, 0.148, 1.5), lo = c(0, 0.351, 0.351))
evaluateRoot(RootModel, mo = Query$mo, lo = Query$lo)
# The exact method solves the characteristic relation at the same queries.
evaluateRoot(buildRootModel(mode = 1, method = "exact"), mo = Query$mo, lo = Query$lo)
```

---

evaluateSiteFactor *Conditional site amplification factor of the NGA-East models*

Description

Natural-log mean and standard deviation of the site amplification factor F that converts a spectral ordinate at the reference condition `Vref` into the ordinate at a site of time-averaged velocity `VS30`, given the peak ground acceleration at the reference condition and the oscillator period. Model `"ST20"` is the final NGA-East ergodic model (linear part of Stewart et al. 2020, nonlinear part of Hashash et al. 2020, coefficients of the USGS nshmp-lib table carried in `SiteAmp2020`); model `"ST17"` is the 2017 generation (PEER reports 2017/04 and 2017/05). The pair defines the lognormal $\ln F \sim N(\mu, \sigma)$ conditional on PGA, period, `VS30` and `Vref`; it is the factor, not an amplified spectrum (see `evaluateSiteSa()`, `integrateSiteMixture()`, `integrateSiteSpectrum()` and `simulateSiteSpectrum()`, which integrate it over the rock demand).

Usage

```
evaluateSiteFactor(PGA, Tn, VS30, Vref, model)
```

Arguments

`PGA`	Peak ground accelerations in g at the reference condition, finite and non-negative.
`Tn`	One oscillator period in seconds, finite and non-negative; 0 is PGA.
`VS30`	One target time-averaged shear-wave velocity in m/s.
`Vref`	One reference velocity in m/s, 760 or 3000.
`model`	`"ST20"` or `"ST17"`.

Value

A data.table with `muLnF` and `sigmaLnF` (natural-log mean and standard deviation of F, one row per `PGA`) and `ID` (the model), unrounded.

Mathematics

Model "ST20". The coefficients $c, V_1, V_2, V_f, V_c, f_3, f_4, f_5, \sigma_{vc}, \sigma_l, \sigma_u, \sigma_c, f_{760i}, f_{760g}, f_{760is}, f_{760gs}$ of the period are read from SiteAmp2020: the PGA row for $T_n \le 0.01$ s, the tabulated row at a tabulated period, and otherwise the linear interpolation in $\ln T_n$ between the bracketing rows; a period above 10 s is clamped to 10 s with a warning. With $w(v) = \min(\max(0.667\ln(v/400)/\ln 1.5 + 0.1, 0.1), 0.767)$ and $F_{760}(v) = wf_{760i} + (1-w)f_{760g}$, the linear term is $f_{lin}(v) = f_v(v) + F_{760}(v)$, where $f_v(v)$ is $c\ln(V_1/760)$ for $v \le V_1$, $c\ln(v/760)$ up to V_2, $c\ln(V_2/760)$ up to 2000 m/s, $c\ln(V_2/760) - (c\ln(V_2/760) + F_{760}(v))\ln(v/2000)/\ln 1.5$ below 3000 m/s and $-F_{760}(v)$ from 3000 m/s (Stewart et al. 2020, Eqs. 3 and 5-7). The nonlinear slope is $f_2(v) = f_4[e^{f_5(\min(v,V_{NL})-360)} - e^{f_5(V_{NL}-360)}]$ for $v < V_c$ and 0 otherwise, with $V_{NL} = 760$ m/s for $T_n < 0.4$ s and 3000 m/s above (Hashash et al. 2020, Eq. 3), and it multiplies $F_{Rk} = \ln((PGA_{3000} + f_3)/f_3)$, where $PGA_{3000} = PGA/2.275$ when Vref is 760 m/s (PEER 2017/05, Eq. 2.4) and PGA when it is 3000 m/s. The linear dispersion is $\sigma_l - 2Dx + Dx^2$ with $D = \sigma_l - \sigma_{vc}$ and $x = (v-200)/(V_f - 200)$ below V_f, σ_{vc} up to V_2, $\sigma_{vc} + (\sigma_u - \sigma_{vc})((v-V_2)/(2000-V_2))^2$ below 2000 m/s, $\sigma_u(1 - \ln(v/2000)/\ln 1.5)$ up to 3000 m/s and 0 above (Stewart et al. 2020, Eq. 4); the nonlinear dispersion is $\sigma_{f2}F_{Rk}$ with $\sigma_{f2} = \sigma_c$ below 300 m/s, $\sigma_c(1 - \ln(v/300)/\ln(1000/300))$ below 1000 m/s and 0 above. For Vref = 760 m/s

$$\mu = f_{lin}(V_{S30}) - f_{lin}(760) + (f_2(V_{S30}) - f_2(760))F_{Rk}, \quad \sigma = \sqrt{\sigma_{lin}^2(V_{S30}) + (\sigma_{f2}(V_{S30})F_{Rk})^2},$$

the reference side adding no dispersion; for Vref = 3000 m/s $\mu = f_{lin}(V_{S30}) + f_2(V_{S30})F_{Rk}$ and the weighted F_{760} dispersion $wf_{760is} + (1-w)f_{760gs}$ enters the root sum of squares.

Model "ST17". The coefficients are interpolated linearly in T_n from the tables of PEER 2017/04 (linear part, F_{760}) and 2017/05 (nonlinear part, Table 2.1). The linear mean is $c\ln(V_1/760)$, $c\ln(V_{S30}/760)$ or $c\ln(V_2/760) + (c/2)\ln(V_{S30}/V_2)$ below V_1, up to V_2 and above; the nonlinear mean is $f_2\ln(PGA/b + 1)$ for $V_{S30} < V_c$ with $f_2 = f_4[e^{f_5(\min(V_{S30},3000)-360)} - e^{f_5(3000-360)}]$ and $b = 2.275f_3$ (Vref 760) or f_3 (Vref 3000), and $F_{760}(T_n)$ is added for Vref = 3000. The dispersion combines the linear term (quadratic below V_f between 200 m/s and V_f, σ_{vc} up to V_2, quadratic towards 2000 m/s above), the tabulated F_{760} dispersion for Vref = 3000 and the nonlinear term $\sigma_c\ln(PGA/b + 1)$ tapered from 300 to 1000 m/s.

When VS30 equals Vref the factor is the deterministic unit factor, $\mu = 0$ and $\sigma = 0$. Every result is a conditional evaluation at the supplied PGA; the dispersion combines the epistemic uncertainties of the published median model, not the uncertainty of the rock demand.

References

Stewart, J.P. et al. (2020). Ergodic site amplification model for central and eastern North America. Earthquake Spectra 36(1). Hashash, Y.M.A. et al. (2020). Nonlinear site amplification model for ergodic seismic hazard analysis in central and eastern North America. Earthquake Spectra 36(1). Stewart, J.P. et al. (2017). PEER Report 2017/04. Hashash, Y.M.A. et al. (2017). PEER Report 2017/05.

Examples

```
evaluateSiteFactor(PGA = c(0.05, 0.2, 0.5), Tn = 0.2, VS30 = 300, Vref = 760,
                   model = "ST20")
evaluateSiteFactor(PGA = 0.1, Tn = 1.0, VS30 = 760, Vref = 3000, model = "ST20")
```

`evaluateSiteSa` *Site spectral acceleration given the rock ordinate and PGA*

Description

Natural-log mean and standard deviation of the spectral acceleration at a site of velocity `VS30`, conditional on the rock ordinate `Sa` and on the rock PGA of the same realization at the reference condition, under a site factor of `evaluateSiteFactor()`. Applied to the mean ordinates of a spectrum with the mean PGA, it is the site spectrum of a deterministic rock spectrum.

Usage

```
evaluateSiteSa(Sa, PGA, Tn, VS30, Vref, model)
```

Arguments

`Sa`	Rock spectral accelerations in g at period `Tn`, finite and positive.
`PGA`	Rock peak ground accelerations in g, aligned with `Sa`.
`Tn`	One oscillator period in seconds.
`VS30`	One target velocity in m/s.
`Vref`	One reference velocity in m/s, 760 or 3000.
`model`	Site factor model of `evaluateSiteFactor()`, `"ST20"` or `"ST17"`.

Value

A data.table with `muLnSaF` and `sigmaLnSaF` (natural-log mean and standard deviation of the site ordinate, one row per `Sa`) and `ID` (the model), unrounded.

Mathematics

With $\ln F \mid PGA \sim N(\mu_F, \sigma_F)$ from `evaluateSiteFactor()`,

$$\ln S_{aF} \mid S_a, PGA \sim N(\ln S_a + \mu_F, \sigma_F),$$

row by row; its quantile at probability p is $\exp(\ln S_a + \mu_F + \sigma_F \Phi^{-1}(p))$ and its mean $\exp(\ln S_a + \mu_F + \sigma_F^2/2)$. The dispersion is that of the site factor alone: `Sa` and `PGA` are conditioning values, not distributions, so the result does not carry the uncertainty of the rock demand. When `VS30` equals `Vref` the result is $\ln S_a$ with zero dispersion.

Examples

```
evaluateSiteSa(Sa = VancouverSpectra[Tn == 0.5][order(TR), Sa], Tn = 0.5, VS30 = 300,
               PGA = VancouverSpectra[Tn == 0][order(TR), Sa], Vref = 760, model = "ST17")
```

`evaluateSoilLayers` *Evaluate stress, stiffness and shear-wave velocity of ordered layers*

Description

Deterministic physics of a layered column whose index properties are supplied: effective unit weight under the water table, octahedral mean effective stress, overconsolidation ratio, small-strain shear modulus from the correlations of the soil group and the shear-wave velocity of each layer. The layers may come from `sampleSoilLayers()` or from measured or specified data with the same columns.

Usage

```
evaluateSoilLayers(layers, Hs, water, POP, m1 = HardinExponent)
```

Arguments

`layers`	A data.frame or data.table with one row per layer from the surface down and columns `USCS`, `hs` (m), `zm` (m), `eo`, `gsat` (kN/m3) and `IP` (percent, `NA` for coarse soils); `emin` and `emax` are used for the relative density when present.
`Hs`	One positive column height in metres.
`water`	Water-table height as a fraction of `Hs` (0 for a dry column).
`POP`	Preconsolidation pressure in kPa, added to the mean stress in the overconsolidation ratio.
`m1`	Nodes of the interpolation of m_1 in `IP`: a data.frame or data.table with numeric columns `IP` (percent) and `m1`. The default is Table 1 of Hardin and Drnevich (1972): `IP` 0, 20, 40, 60, 80, 100 and `Inf` with 0, 0.18, 0.30, 0.41, 0.48, 0.50 and 0.50.

Value

A data.table with the input columns and `group` (`Gravels`, `Sands` or `Fines`), `Dr` (relative density, when the void-ratio limits are supplied), `gs` (effective unit weight, kN/m3), `pm` (kPa), `OCR`, `Gm` (MPa) and `VSm` (m/s). Each row is a conditional evaluation of its inputs; it carries no dispersion of its own.

Mathematics

With $K_o = 0.5$ and the effective unit weight γ' of each layer, assigned by its midpoint depth z_m: $\max(\gamma_{sat} - 10, 0)$ kN/m3 when z_m lies below the water table $z_w = H_s(1 - w)$, else γ_{sat}; the octahedral stress increases by $\Delta p = (1 + 2K_o)\gamma' h/3$ across each layer; the layer mean p_m is the average of its top and bottom values, and $OCR = \max((p_m + POP)/p_m, 1)$ with the preconsolidation `POP` in kPa. For every correlation `j` of the layer's soil group in ShearModelParameters the modulus at the mean stress is

$$G_j = A_j F_j(e_o)\, OCR^{m_1}\, 100^{n_j} (p_m/100)^{n_j},$$

with $F_j(e) = (C_e - e)^2/(1 + e)$, except the linear $F(e) = C_e - e/(1 + e)$ of SS75, and m_1 the exponent K of the overconsolidation ratio of Hardin and Drnevich (1972) for fine soils and zero otherwise (hydrated copy, equation eq-design-gmax, printed page 671). m_1 is interpolated linearly in IP through the nodes of m1 and is NA outside them, so a fine layer whose IP lies below the first node or above the last has an NA modulus whenever $OCR > 1$. The default nodes are Table 1 of that paper (printed page 672): IP 0, 20, 40, 60 and 80 with 0, 0.18, 0.30, 0.41 and 0.48, and the row "PI $\geq$ 100: 0.50", encoded as the nodes (100, 0.50) and (Inf, 0.50) so that m_1 is 0.50 at and above 100. The Inf node is that encoding of the published row, not a further choice. The layer modulus is the arithmetic mean $G_m = \overline{G_j}$ over the group's correlations, in MPa, and

$$V_S = \sqrt{9.81 \cdot 1000\, G_m/\gamma_{sat}}$$

in m/s uses the saturated unit weight. These are the relations of newmark 1.7.1 evaluated layer by layer, except the default nodes: 1.7.1 ended them at (100, 0.48), with NA above, and buildShearTable() passes those nodes to reproduce it. The spread of G_j across correlations is not retained by this version.

Examples

```
Layers <- data.frame(USCS = c("SM", "ML"), hs = c(5, 5), zm = c(2.5, 7.5),
                     eo = c(0.7, 0.9), gsat = c(20, 18), IP = c(NA, 10))
evaluateSoilLayers(Layers, Hs = 10, water = 0, POP = 100)
```

---

evaluateUniformHazard *Uniform-hazard ordinates of hazard curves at given return periods*

Description

Evaluates every hazard curve of a table at the requested return periods, which gives the ordinates of the uniform-hazard spectra of those curves.

Usage

```
evaluateUniformHazard(curves, TR, by)
```

Arguments

curves	A table with columns TR (years), Sa (g) and the columns named in by.
TR	Finite positive return periods in years.
by	Names of the columns of curves that identify one curve.

Value

A data.table with the by columns, TR and Sa (g), one row per curve and requested period, curves in order of first appearance and periods in the order given. A curve with fewer than two usable points stops the call.

Mathematics

For one curve with points $(TR_i, S_{a,i})$, only points with finite positive TR and S_a are used, and points sharing a TR keep the largest S_a (a message reports it when their ordinates differ beyond 12 significant digits). The ordinate at each requested TR_o is the log-log linear interpolation

$$S_a(TR_o) = \exp L\left(\ln TR_o;\ \ln TR_i, \ln S_{a,i}\right),$$

with L piecewise linear. A requested period may lie outside the curve's range by at most $\max(0.5, 10^{-4}\max(TR_i, TR_o))$ years, where the end ordinate is taken; beyond that margin the call stops rather than extrapolate. The result evaluates the given curve (for OpenQuake, a statistic curve of the logic tree) at the return period; the ordinate of a mean curve is not, in general, the mean of the ordinates.

Examples

```
# The rows of one period Tn form one hazard curve.
evaluateUniformHazard(VancouverSpectra, TR = c(100, 2475), by = "Tn")
```

---

integrateDnHazard *Displacement exceedance integrated over rock hazard curves*

Description

Integrates the existing displacement regressions and ST20/ST17 site uncertainty over annual rock exceedance rates, at a fixed yield coefficient. Rigid models use the PGA curve. Flexible models use Sa at the degraded period of the model, 1.5 Ts for BT07 and BM17 and 1.3 Ts for BM19, and, for the site factor only, the PGA at the same return period. This last pairing is an equal-hazard approximation, not an event joint distribution.

Usage

```
integrateDnHazard(
  curves,
  ky,
  Ts,
  VS30,
  Vref,
  Mw,
  weights,
  Da,
  IT,
  TR,
  siteModel
)
```

Arguments

curves	One site's mean rock hazard: a data.frame or data.table with finite numeric columns Tn (s, non-negative, with a PGA curve at Tn = 0), Sa (g, positive) and AEP (1/year, positive), each curve decreasing in Sa with at least two distinct rates.
ky	Fixed yield coefficient in g, one finite positive value.
Ts	Initial sliding-mass period in seconds, one finite positive value.
VS30	One target time-averaged shear-wave velocity in m/s.
Vref	One reference velocity in m/s, 760 or 3000.
Mw	Fixed moment magnitude, one finite positive value.
weights	Positive named dimensionless weights of AM88, JB07, SR08, BT07, BM17 or BM19.
Da	Displacement threshold in cm, one finite positive value.
IT	Exposure in years, one finite positive value.
TR	Two increasing finite positive return periods in years; the band contribution covers their intersection with each model's observed support and is zero for an empty intersection.
siteModel	Site-factor model, ST20 or ST17.

Details

For BT07, BM17 and BM19 (evaluateFlexibleDisplacement(); BM19 with its ordinary-motion equation, as the DnTable products take it), the integrand uses the lognormal distribution conditional on nonzero displacement; it is not multiplied by the probability of nonzero displacement, so these rates do not represent the unconditional exceedance hazard of the published mixed models. For JB07 and SR08 (evaluateRigidDisplacement()), the Arias intensity is estimateAI() of each site-amplified PGA, without additional dispersion of the Arias intensity.

Repeated AEP nodes retain the largest Sa. Curves are interpolated linearly in log rate and log acceleration; an intermediate oscillator period uses linear interpolation of log acceleration in log(Tn + 1e-8) between its two neighbours at equal annual rate. Integration is restricted to their common observed support and the PGA support. No tail is extrapolated. The integral is split at the native hazard knots before adaptive quadrature. The rare-tail bound assumes exceedance probability one beyond that support. Model rates are combined with normalized weights before the Poisson transform. The calibration range of ky of each model comes from buildKyLimits() with the largest rock PGA of the model's common support; extrapolated is 1 when ky lies outside it, and the rate is integrated all the same.

The standard-normal residual of the site factor is integrated between -8 and 8 with its density, in pieces split at each rate where the site-amplified PGA reaches the AM88 cap ky / PGA = 0.9999 and at the crossings of the median displacement with Da on each monotone side of the regression's maximum, by Gauss-Legendre rules of 32, 64 and 128 points per piece. An order is accepted when, against the previous one, the rate and the band rate each agree within 1e-4 of the value plus 1e-10 and the probabilities at the nodes within 1e-4; the function refuses the site with site quadrature did not converge when the last order fails. Each interval between hazard knots is integrated adaptively with a relative tolerance of 1e-6, an absolute tolerance of 1e-12

shared by the intervals and at most 1000 subdivisions. quadratureError, bandQuadratureError and probabilityError report the disagreements of the accepted order, and integrationError and bandIntegrationError the adaptive quadrature's own estimates; none of them is an error bound.

Value

A list of three data.tables. summary has one row: ky (g), Da (cm), IT (years), the weighted annual exceedance rate AEP (1/year) and its Poisson probability of exceedance in IT years POE, the band contribution bandAEP (1/year) and bandPOE, the Poisson probability of exceedance in IT years of the band rate, not a fraction of POE, and the rare-tail upper bounds rareTailAEPBound (1/year) and rareTailPOEBound, which bound only the omitted events rarer than the common support; the omitted frequent side is not bounded. models has one row per displacement model, IDn: the normalized weight, the model period Tn (s), ky, Da, the bounds kyMin and kyMax (g) of the calibration range of ky of the model, extrapolated (1 when ky lies outside that range, else 0; the diagnostic at the largest rock PGA of the common support, not a certification of calibration along the integral), AEP, POE, bandAEP, rareTailBound (1/year), the common support TR.min and TR.max (years), quadratureError (1/year), bandQuadratureError (1/year) and probabilityError, the disagreement between the last two Gauss-Legendre orders of the rate, of the band rate and of the node probabilities (the largest over the nodes), each not an error bound, integrationError (1/year), the sum of the adaptive quadrature's own estimates over the intervals, and bandIntegrationError (1/year), the same sum over the intervals of the band. conditional has one row per model and rate node, IDn and AEP (1/year): TR (years), the rock Sa at the model period and the rock PGA (g), and the probability that the displacement exceeds Da at that node. Probabilities are dimensionless.

Examples

```
Curves <- data.table::as.data.table(VancouverSpectra)[, .(Tn, Sa, AEP = 1 / TR)]
OUT <- integrateDnHazard(Curves, ky = 0.1, Ts = 0.3, VS30 = 300, Vref = 760, Mw = 7,
                         weights = c(BT07 = 1, BM17 = 1), Da = 15, IT = 50,
                         TR = c(475, 2475), siteModel = "ST20")
OUT$summary
OUT$models[, .(IDn, Tn, AEP, POE, extrapolated)]
```

---

integrateDnHazardProfiles

Displacement exceedance mixed over site profiles and yield coefficients

---

Description

Mixes the annual displacement exceedance rates of integrateDnHazard() over a discrete distribution of the site profile, given by pairs of the period Ts and velocity VS30 of each profile, and over yield coefficients ky with weights, under the declared hypothesis that the profile and the yield coefficient are independent.

Usage

```
integrateDnHazardProfiles(
  curves,
  profiles,
  ky,
  Mw,
  weights,
  Da,
  IT,
  TR,
  siteModel,
  Vref
)
```

Arguments

curves	One site's mean rock hazard: a data.frame or data.table with finite numeric columns Tn (s, non-negative, with a PGA curve at Tn = 0), Sa (g, positive) and AEP (1/year, positive), each curve decreasing in Sa with at least two distinct rates.
profiles	A data.frame or data.table with one row per profile and the columns Ts (s) and VS30 (m/s), each value as integrateDnHazard() takes it, and optionally w, positive finite weights.
ky	One yield coefficient in g, or a data.frame or data.table with the columns ky (g) and w, positive finite weights.
Mw	Fixed moment magnitude, one finite positive value.
weights	Positive named dimensionless weights of AM88, JB07, SR08, BT07, BM17 or BM19.
Da	Displacement threshold in cm, one finite positive value.
IT	Exposure in years, one finite positive value.
TR	Two increasing finite positive return periods in years; the band contribution covers their intersection with each model's observed support and is zero for an empty intersection.
siteModel	Site-factor model, ST20 or ST17.
Vref	One reference velocity in m/s, 760 or 3000.

Value

A list of two data.tables. summary has one row: Da (cm), IT (years), the mixed annual rate AEP (1/year) and its Poisson probability of exceedance in IT years POE, the mixed band rate bandAEP (1/year) and its probability bandPOE, the mixed rare-tail bound rareTailAEPBound (1/year) and its probability rareTailPOEBound, the counts nProfiles and nKy, extrapolated, the weighted fraction of the components in which some model has its ky outside its calibration range, and mixtureError and bandMixtureError (1/year). components has one row per profile and yield coefficient, profiles first and in their order: Ts, VS30, ky, w (the product of the normalized weights of the profile and of ky), the columns of the summary of integrateDnHazard() after ky (Da,

IT, AEP, POE, bandAEP, bandPOE, rareTailAEPBound, rareTailPOEBound) and extrapolated (1 when some model of integrateDnHazard() reports its ky outside its calibration range, else 0).

Mathematics

By the total probability theorem, with the normalized weights w_i of the profiles and v_k of the yield coefficients, the mixed annual rate is

$$\lambda_D(D_a) = \sum_i \sum_k w_i v_k \lambda_D(D_a \mid T_{s,i}, V_{S30,i}, k_{y,k}),$$

with each conditional rate the AEP of integrateDnHazard(). The band rate and the rare-tail bound are mixed with the same weights, and the Poisson probabilities in IT years are formed after the mixture, on the mixed rates. Each weight set is normalized to sum one; without a column w the profiles weigh equally. For the rigid models the conditional rate does not depend on Ts, so the mixture acts on them through VS30 and ky only.

When the profiles are realizations of a Monte Carlo population, such as the rows of one pair of the ShearDraws product of processProject(), the mixed rate is an estimate. mixtureError is the standard error of the weighted mean over the profiles of $r_i = \sum_k v_k \lambda_D(D_a \mid T_{s,i}, V_{S30,i}, k_{y,k})$, the weighted standard deviation of the r_i (with the divisor $1 - \sum_i w_i^2$ of reliability weights) over the square root of the effective number of profiles $1/\sum_i w_i^2$; with equal weights it is the sample standard deviation over $\sqrt{n}$. bandMixtureError is the same quantity for the band rate. Neither is an error bound, and both are NA for one profile.

Examples

```
Curves <- data.table::as.data.table(VancouverSpectra)[, .(Tn, Sa, AEP = 1 / TR)]
Profiles <- data.frame(Ts = c(0.25, 0.3, 0.4), VS30 = c(320, 300, 270))
Ky <- data.frame(ky = c(0.08, 0.1), w = c(1, 3))
OUT <- integrateDnHazardProfiles(Curves, profiles = Profiles, ky = Ky, Mw = 7,
                                 weights = c(BT07 = 1, BM17 = 1), Da = 15, IT = 50,
                                 TR = c(475, 2475), siteModel = "ST20", Vref = 760)
OUT$summary
OUT$components[, .(Ts, VS30, ky, w, AEP, extrapolated)]
```

---

integrateSiteMixture *Site mixture of an analytic rock mixture by Gauss-Hermite quadrature*

Description

Transforms a rock spectrum represented at each oscillator period as a finite lognormal mixture into the corresponding site spectrum, integrating the site factor of evaluateSiteFactor() over the rock PGA of each component with Gauss-Hermite quadrature. Every quadrature node becomes one component of the returned mixture, so the result is again an analytic mixture: summarizeMixture() reduces it to quantiles and mean, and refineSiteMixture() chooses the order.

Usage

```
integrateSiteMixture(mixture, VS30, Vref, model, order)
```

Arguments

`mixture`	A table with one row per `(Tn, k)` and columns `Tn` (oscillator period, s), `k` (component label), `muLn`, `sigmaLn` (natural-log location and scale, `sigmaLn` non-negative) and `w` (positive weights summing to one at every period). It must hold the PGA rows (`Tn == 0`), and every component used at another period must have a PGA row of the same label.
`VS30`	One target velocity in m/s.
`Vref`	One reference velocity in m/s; 760 or 3000 unless it equals `VS30`.
`model`	Site factor model of `evaluateSiteFactor()`, `"ST20"` or `"ST17"`.
`order`	One integer, at least 2: Gauss-Hermite nodes per component.

Value

A data.table with `Tn` (s), `k`, `muLn`, `sigmaLn` and `w`, ordered by `Tn` and `k`: the site spectrum as a lognormal mixture at every period, conditional on the target velocity and the model, its population being that of the rock mixture. `mixture` is not modified.

Mathematics

At period T_n the rock component k gives $\ln S_a \sim N(\mu_k, \sigma_k)$ with weight w_k, and the component of the same label at $T_n = 0$ gives $\ln PGA \sim N(\mu_{0k}, \sigma_{0k})$. With the Gauss-Hermite rule of `order` nodes x_m and weights ω_m normalized to one (Golub-Welsch, weight e^{-x^2}), the PGA nodes are $\ln PGA_{km} = \mu_{0k} + \sqrt{2}\sigma_{0k}x_m$. With $\rho = \rho_{BJ}(\max(T_n, 0.01), 0.01)$ of `correlateResiduals()`, $\rho_k = \rho$ and $r_k = \sigma_k/\sigma_{0k}$ when $\sigma_{0k} > 0$ and both zero otherwise, the ordinate conditional on the PGA node is

$$\ln S_a \mid PGA_{km} \sim N(\mu_k + \rho_k r_k(\ln PGA_{km} - \mu_{0k}), \sqrt{1-\rho_k^2}\,\sigma_k),$$

and with (μ_F, σ_F) of the factor at PGA_{km} the site component (k, m), labelled `"k:m"`, is

$$\mu_{km} = \mu_k + \rho_k r_k(\ln PGA_{km} - \mu_{0k}) + \mu_F, \quad \sigma_{km} = \sqrt{(1-\rho_k^2)\sigma_k^2 + \sigma_F^2}, \quad w_{km} \propto w_k\omega_m,$$

the weights renormalized to one at every period. The site residual is taken independent of the rock residual given PGA. When `VS30` equals `Vref` the site mixture is the rock mixture. No random numbers are drawn; the error of the order is not measured here (see `refineSiteMixture()`).

Examples

```
Mixture <- data.table::CJ(Tn = c(0, 0.2, 1), k = c("a", "b"))
Mixture[, `:=`(muLn = log(0.4) + 0.1 * (k == "b") - 0.5 * Tn, sigmaLn = 0.5, w = 0.5)]
integrateSiteMixture(Mixture, VS30 = 360, Vref = 760, model = "ST20", order = 16)
```

`integrateSiteSpectrum` *Site spectrum of a rock quantile spectrum by quadrature*

Description

Quantiles and mean of the spectral acceleration at a site of velocity `VS30`, from a rock spectrum given at each oscillator period by its mean and its quantiles, integrating the joint rock demand of PGA and the ordinate and the site factor of `evaluateSiteFactor()` without random numbers. The site quantiles are reported at the probability levels of the rock spectrum. `refineSiteSpectrum()` chooses the order; `simulateSiteSpectrum()` samples the same model; a rock spectrum given only by its means is the conditional case of `evaluateSiteSa()`.

Usage

```
integrateSiteSpectrum(spectrum, VS30, Vref, model, order)
```

Arguments

`spectrum`	A table with columns `Tn` (s), `p` (`"mean"` or a probability label) and `Sa` (g): exactly one row at `Tn == 0` and `p == "mean"`, and at PGA and every period at least three unique, positive, non-decreasing quantiles. Periods without a single mean row are skipped.
`VS30`	One target velocity in m/s.
`Vref`	One reference velocity in m/s; 760 or 3000 unless it equals `VS30`.
`model`	Site factor model of `evaluateSiteFactor()`, `"ST20"` or `"ST17"`.
`order`	One integer, at least 4: Gauss-Legendre nodes per latent dimension, n^2 joint nodes per period.

Value

A data.table ordered by `Tn` and `p` with `Tn`, `p` (the labels of the rock spectrum), `Sa` (the rock ordinate at that level, g) and `SaF` (the site ordinate at the same level, g), unrounded. Quantile and mean rows are marginal statistics at each period over the population of the rock spectrum and the site-factor residual; the integration error of `order` is not reported (see `refineSiteSpectrum()`). The random number stream is not used.

Mathematics

At PGA and at each period the rock quantile function $Q(u) = \ln S_a$ is the Hyman monotone spline through the tabulated knots $(p_k, \ln S_{a,k})$, continued below the first and above the last knot by the adjacent secant slopes (floored at zero) and clamped to $u \in [0, 1]$. The Gauss-Legendre rule of `order` nodes on $(0, 1)$ gives u_i and w_i (Golub-Welsch); with $z_i = \Phi^{-1}(u_i)$ and $\rho = \rho_{BJ}(T_n, 0)$ of `correlateResiduals()`, the joint node (i, j) carries

$$\ln PGA_i = Q_0(u_i) + c_0, \quad \ln S_{a,ij} = Q_n(\Phi(\rho z_i + \sqrt{1-\rho^2} z_j)) + c_n, \quad W_{ij} = w_i w_j,$$

a Gaussian copula between PGA and the ordinate, where c_0 and c_n re-centre each rock marginal on its tabulated mean ($\sum_i w_i PGA_i = \overline{PGA}$ and $\sum_{ij} W_{ij} S_{a,ij} = \bar{S}_a$). Each joint node is one lognormal component of the site ordinate, $\mu_{ij} = \ln S_{a,ij} + \mu_F(PGA_i)$ and $\sigma_{ij} = \max(\sigma_F(PGA_i), 10^{-9})$, the site residual being integrated analytically and taken independent of the rock residual given PGA. The site quantile at a tabulated probability p solves $\sum W_{ij}\Phi((x-\mu_{ij})/\sigma_{ij}) = p$ (root to 10^{-10} on $[\min(\mu - 9\sigma), \max(\mu + 9\sigma)]$) and the site mean is $\sum W_{ij} e^{\mu_{ij} + \sigma_{ij}^2/2}$. When `VS30` equals `Vref` the site spectrum is the rock spectrum. Each period is integrated with its own copula with PGA; no dependence between periods is implied.

Examples

```
P <- c("0.05", "0.10", "0.16", "0.50", "0.84", "0.90", "0.95", "mean")
Factor <- c(0.45, 0.60, 0.72, 0.95, 1.30, 1.48, 1.78, 1.00)
Spectrum <- data.table::data.table(Tn = rep(c(0, 0.2, 1), each = 8), p = rep(P, 3),
                                   Sa = rep(c(0.30, 0.45, 0.24), each = 8) * rep(Factor, 3))
integrateSiteSpectrum(Spectrum, VS30 = 270, Vref = 760, model = "ST20", order = 32)
```

`interpolateSpectrum` *Spectral quantiles at one oscillator period by log-log interpolation*

Description

Interpolates every quantile curve of one spectrum slice (one site, model, Vs30 and return period) at a target oscillator period, as the displacement sampling does when the spectrum lacks the periods $1.3T_s$ or $1.5T_s$. Each curve is interpolated independently; the slice is not grouped.

Usage

```
interpolateSpectrum(spectrum, Tn)
```

Arguments

`spectrum`	A table of one spectrum slice with `Tn` (s), `p` (probability labels and optionally `"mean"`) and `Sa` (g). Every quantile label needs two or more rows; other columns are ignored.
`Tn`	One finite non-negative target oscillator period in seconds.

Value

A data.table with `Tn`, `p` and `Sa`, one row per quantile label plus the mean row when the slice has one, ordered by `Tn` and `p`.

Mathematics

For the rows $(T_i, S_{a,i})$ of one probability label,

$$S_a(T) = \exp\, L\left(\ln T;\ \ln(T_i + 10^{-8}),\ \ln S_{a,i}\right)$$

where L is linear interpolation (`stats::approx`) with constant extension beyond the tabulated periods (`rule = 2`). The offset 10^{-8} admits a zero-period node; a target of zero falls below every node and takes the lowest-period ordinate. Duplicate (T, p) rows signal a warning and are averaged in $\ln S_a$, a geometric mean. The `"mean"` rows follow the same rule when there are two or more; a single mean row is copied to the target period and a slice without one returns no mean row.

Examples

```
Spectrum <- data.frame(Tn = rep(c(0.2, 0.8), each = 3L),
                       p = rep(c("0.16", "0.84", "mean"), 2L),
                       Sa = c(1, 2, 1.5, 4, 8, 6))
interpolateSpectrum(Spectrum, Tn = 0.4)
```

---

listDisplacementFamilies

Families of the displacement models

Description

Lists the families of displacement models that the displacement and kmax tables summarize as trees: `ensemble` holds the six models, `rigid` the models of `evaluateRigidDisplacement()` and `flexible` those of `evaluateFlexibleDisplacement()`. `buildDnTables()` and `buildKmaxTable()` pool the active members of every family that has one, with their weights normalized, and write the pool with the family name as `IDn`; `queryProjectTable()` marks a tree as extrapolated from the calibration ranges of its members.

Usage

```
listDisplacementFamilies()
```

Value

A named list of three character vectors, `ensemble`, `rigid` and `flexible`, in this order: the identifiers of the models of each family, in the order in which `buildDnTables()` and `buildKmaxTable()` pool them.

Examples

```
listDisplacementFamilies()
```

listProjectTables *List the tables of a project*

Description

Reads the project contract and lists the table files of processProject() that exist under path.tables, reading each one as its step wrote it to report its rows, columns and key. It writes nothing.

Usage

```
listProjectTables(file, root)
```

Arguments

file	Path of the JSON contract, resolved against root when it is relative.
root	Existing project root against which the contract and its paths are resolved.

Details

The tables are the files at the destinations of processProject(): ShearTable.Rds, ShearDraws.Rds, DnPlotTable.Rds, DnTable.Rds, kmaxTable.Rds and DnPOETable.Rds at the root of path.tables, and DnDraws.Rds, PGATable.csv, DnTable.Rds, kmaxTable.Rds and DnPOETable.Rds under <ID.gmdp>/<siteID>/ for every directory there whose name is a site identifier (a letter followed by letters, digits, _ or -). A destination without its file is not listed, and neither is any other file. A contract without ID.gmdp, as one for the ShearTable step alone may be, has global tables only. The contract is read and checked, and its paths resolved against root, as processProject() does. The RDS files are read with readRDS() and PGATable.csv with data.table::fread() and TR numeric, as the kmaxTable step reads it back; a table file that cannot be read, or that holds no table, stops the listing naming it.

Value

A data.table with one row per table file, in the order in which processProject() publishes them, and the columns table, step (the step that publishes it), siteID (NA for a global table), path (the file spelled from path.tables as the contract declares it, as in the effects record of processProject()), bytes (the size of the file), rows (the rows of the table it holds), columns and key (list columns: the column names of that table, and its key columns, empty when it has none) and date (the modification time of the file). Without tables, the table has these columns and no rows.

Examples

```
Root <- file.path(tempdir(), "newmarkProducts")
dir.create(Root)
writeLines('{"path": {"tables": "data/newmark"}, "Vref.gmdp": 760,
  "Hs": [10, 12], "lo.min": 0.35, "lo.max": 0.075, "s.min": 2.5, "s.max": 3.5,
  "uscs": {"U1": ["ML", "MH"]}, "IDm": ["U1"], "seed": 20260915}',
  file.path(Root, "newmark.json"))
listProjectTables("newmark.json", root = Root)
```

```
processProject(root = Root, steps = "ShearTable")
listProjectTables("newmark.json", root = Root)

unlink(Root, recursive = TRUE)
```

`processProject` *Run the steps of a project*

Description

Reads the project contract, plans the selected steps and publishes their tables under `path.tables`, file by file, returning the record of what it published. The steps are named after the table each one publishes and always run in the order `ShearTable`, `ShearDraws`, `DnTable`, `kmaxTable`, `DnPOETable`; `ShearDraws` runs only when `steps` names it.

Usage

```
processProject(
  file = "newmark.json",
  root,
  steps = c("ShearTable", "DnTable", "kmaxTable", "DnPOETable"),
  sites = character(),
  dryRun = FALSE
)
```

Arguments

`file`	Path of the JSON contract, resolved against `root` when it is relative.
`root`	Existing project root against which the contract and its relative paths are resolved; required, with no default. The CLI passes the working directory when `--root` is absent.
`steps`	Any of `"ShearTable"`, `"ShearDraws"`, `"DnTable"`, `"kmaxTable"` and `"DnPOETable"`; they run in that order. The default selects every step except `"ShearDraws"`.
`sites`	Identifiers of the sites to process; empty selects every site found under `path.uhs`. `DnTable`, `kmaxTable` and `DnPOETable` process sites, and an unknown site stops them before anything is written. The global tables are rebuilt from the tables of every site found, selected or not.
`dryRun`	`TRUE` makes the checks that a run makes before its first calculation (see Details), reports with `message()` the contract, the steps, the sites and every destination it would write or replace, and returns without calculating, seeding, creating a directory or writing anything.

Details

Everything the run decides is resolved before its first calculation, and a dry run makes the same checks: the contract keys of the selected steps, the paths, the sites and the destinations. For `ShearTable` and `ShearDraws`, the plan also applies to the geometries, the materials, the pairs and `Vref.gmdp` the rules that `buildShearTable()` applies to its arguments, and to the USCS codes of every selected material the rule of `sampleSoilLayers()`. For `ShearDraws`, it also requires `seed` and, unless `ShearTable` runs before it, the published `ShearTable.Rds`. For `DnTable` and `kmaxTable`, the inputs that no selected step produces must exist, and what the shear table and the MCE table hold is checked before those steps calculate anything: for `DnTable`, the reference rows of the shear table, the paired realizations in `ShearDraws.Rds` and the MCE demand over `1.5 * max(Ts)` of the columns used, as `buildDnTables()` requires them. A dry run cannot check columns that a selected `ShearDraws` step has yet to produce; it checks the reference rows then. For `kmaxTable`, it checks the heights `Hs` from which `buildKmaxTable()` derives its displacements. A dry run makes the checks of `DnTable` first and a run the check of the heights first, as each phase of `newmark` 1.7.1 does, so that when both fail the two report different failures. With `ShearTable` selected as well, the shear table is the one that step publishes: a run makes these checks after publishing it, and a dry run, which publishes nothing, does not make them. A contract or an input that these checks reject stops the run before the step it belongs to writes anything, in a dry run as in a run.

Value

Invisibly, the effects record: a data.table with one row per published file in publication order (with `dryRun`, one per destination) and the columns `step`, `siteID` (`NA` for a global table), `product`, `path` (the destination spelled from `path.tables` as the contract declares it, so relative to `root` when that path is relative), `action` (`"writes"` when the destination did not exist, `"replaces"` when it did) and `rows` (the rows of the published table; `NA` in a dry run).

Project contract

The contract is one JSON object with the keys of `newmark` 1.7.1 (schema 1), validated strictly. Keys are looked up exactly, and a key outside the list below stops the run naming it; that includes `Hs.min`, `Hs.max`, `NTS`, `KMAX_PARALLEL` and `KMAX_WORKERS`, which 1.7.1 read. Each key is required only by the steps named with it. `file` and the paths of the `path` block are resolved against `root` when they are relative; the working directory is never changed and reading evaluates no code. Numbers are read as doubles whatever their JSON spelling, so `760` and `760.0` are the same value.

`path.tables` Directory that `newmark` writes, and the only one: the global tables at its root and the tables of each site under `<ID.gmdp>/<siteID>/`. Every step; `ShearTable` creates it when it is absent.

`path.data` Directory of the hazard table `MCETable.Rds`, read only. `DnTable`.

`path.uhs` Directory holding `<ID.gmdp>/<siteID>/UHSRock.Rds`, read only; the sites are its directories that hold one. `DnTable`, `kmaxTable` and `DnPOETable`.

`path.calc` JSON of the classical hazard calculation, whose finite positive `job.calculation.truncation_level` truncates the sampling of the rock demand. `DnTable`.

`Vref.gmdp` Reference rock velocity in m/s, one number. Every step; the site steps require 760.

`ID.gmdp`, `TR.gmdp`, `NS`, `subduction`, `Dn_weights` Identifier of the hazard model (a letter followed by letters, digits, _ or -), return periods in years (sorted, duplicates dropped), realizations per cell (a positive whole number), the regime (`true` for subduction) and the 0/1 switches of the displacement models, which `selectDisplacementModels()` turns into the weights of the active models. `DnTable`, `kmaxTable` and `DnPOETable`.

`Mw.gmdp` Moment magnitude of the scenarios: one number for every demand, or an object that names each return period of `TR.gmdp` and `MCE` with its magnitude. `DnTable`.

`DaH.gmdp` Admissible displacements, distinct positive percentages of the height `Hs`. `kmaxTable` and `DnPOETable`.

`DN_PARALLEL`, `DN_WORKERS` Optional parallel controls of `DnTable`: `true` (the default) or `false`, and a positive whole number of workers, by default the cores that R detects minus one, at least one. A string, a logical or a fraction is rejected. The workers evaluate the geometry and material pairs of a demand, so the step never uses more of them than the mean rows of the shear table, and `false` keeps it in one process; the tables do not depend on them.

`uscs`, `IDm` Object of USCS code arrays indexed by material, and the distinct selected materials, each one a key of `uscs`. `ShearTable` and `ShearDraws`.

`Hs`, `lo.min`, `lo.max`, `s.min`, `s.max` Range form of the geometries: the heights in metres, and the first and last truncation ratio and slope, spread evenly over the heights in their declared order (they are not a minimum and a maximum) and passed to `buildGeometry()`. `ShearTable` and `ShearDraws`, unless `geometryTable` is declared; the heights have no default.

`geometryTable` Optional array of `IDg`, `Hs`, `b` and `s` row objects, used instead of the range form.

`pairs` Optional array of `IDg` and `IDm` row objects restricting the product of geometries by materials.

`seed` Optional whole number between 0 and 2147483647 that seeds the `ShearTable` step. A contract without it, such as one written for `newmark` 1.7.1, runs that step on the caller's random stream, as 1.7.1 does without a seed, and the contract alone does not make its table reproducible. `ShearDraws` requires it.

`NS.shear` Optional realizations per pair of `ShearDraws`, a whole number of at least 100; without it, 100. Realizations beyond 100 extend the population of the shear table without changing it. `ShearDraws`, the only step that reads it.

Steps

`ShearTable` builds the geometries, calls `set.seed()` with `seed` when the contract defines it, which replaces the caller's random stream, calls `buildShearTable()` with the levels 0.02, 0.05, 0.10, 0.16, mean, 0.84, 0.90, 0.95 and 0.98, and publishes `<path.tables>/ShearTable.Rds` with `saveRDS()`. Its message reports the seed, or `seed none` without one.

`ShearDraws` reads `ShearTable.Rds` under `path.tables` (the one the `ShearTable` step has just published, when both run), calls `set.seed()` with `seed` and `buildShearDraws()` with the inputs and levels of `ShearTable` and `NS.shear` realizations per pair, and compares the shear table it returns with the one it read, value for value: the same column names in the same order and, in each column, the same values of the same type (`identical()`), whatever the keys and indices of the two tables. When they differ it stops naming the columns that differ and publishes nothing; otherwise it publishes the realizations as `<path.tables>/ShearDraws.Rds` with `saveRDS()`. Realizations whose fitted `mo` is negative are published with `mo` forced to 0 and with the `an` and `Ts` of that homogeneous beam, as `buildShearDraws()` returns them, and the message of the step reports the seed, `NS.shear` and how many realizations were forced. `ShearTable.Rds` is never written by this step.

DnTable reads ShearTable.Rds under path.tables (the one the ShearTable step has just published, when both run), ShearDraws.Rds, MCETable.Rds and the UHSRock.Rds of each selected site, which it processes in alphabetical order. For a site it calls buildDnTables() with ID.gmdp, TR.gmdp, Vref.gmdp, Mw.gmdp, NS, the weights that selectDisplacementModels() makes of Dn_weights and subduction, the truncation level of path.calc and the workers, and buildPGATable(), and publishes under <path.tables>/<ID.gmdp>/<siteID>/, in this order, DnDraws.Rds with saveRDS(), PGATable.csv with data.table::fwrite() and na = "NA", and DnTable.Rds with saveRDS(). It then rebuilds the global DnPlotTable.Rds and DnTable.Rds with assembleDnTables() from the DnTable.Rds of every site found under path.uhs that has one, selected or not, and publishes them, in that order, at the root of path.tables; a site found without that file stays out of them. buildDnTables() seeds each site and demand with its own fixed seed, so the step replaces the caller's random stream and the tables of a site do not depend on the sites run with it. Its unchanged NS movements per pair cycle through the ShearDraws columns in realization order. The same column supplies Ts and VS30; the common ky grid and PGATable use the mean-row reference.

kmaxTable reads ShearTable.Rds under path.tables and, for each selected site in alphabetical order, its DnDraws.Rds and its PGATable.csv, read back with data.table::fread() and TR numeric. For a site it calls buildKmaxTable() with ID.gmdp, TR.gmdp, NS, the weights that selectDisplacementModels() makes of Dn_weights and subduction, and DaH.gmdp, and publishes <path.tables>/<ID.gmdp>/<siteID>/kmaxTable.Rds with saveRDS(). It then assembles the global kmaxTable.Rds with assembleKmaxTables() from the kmaxTable.Rds of every site found under path.uhs that has one, selected or not, without reading their PGATable.csv, and publishes it at the root of path.tables; a site found without kmaxTable.Rds stays out of it. The warnings of both functions about roots beyond the stored grid reach the caller, and the step draws no random number.

The inputs of the site steps that no selected step produces must exist when they are planned: ShearTable.Rds under path.tables, ShearDraws.Rds and MCETable.Rds for DnTable, and the DnDraws.Rds and PGATable.csv of each selected site for kmaxTable without DnTable. DnPOETable requires the same shear table and site draws, without PGATable.csv.

Publication

The tables are published in groups: the shear table; the shear realizations; the three files of a site and the two global files of DnTable; the site and the global kmaxTable; the site and global DnPOETable. Every file of a group is first written to a temporary file .<name>.<suffix> beside its destination, in the format its name declares, and then each destination is replaced by renaming its temporary file, in the order of the group. Each rename is atomic; the group is not, and nothing is rolled back. A failure while writing the temporary files changes no file of the group; a failure at the rename of a file leaves the files before it replaced and the others with their previous content, or still absent; a failure in a later calculation or check leaves everything published before it: the checks of the site steps that follow a published shear table, the calculation of a later site, or a table of another site that the global tables cannot take (a DnTable.Rds that assembleDnTables() rejects, or a kmaxTable.Rds that assembleKmaxTables() rejects), which stops the step with the selected sites published and the global files unchanged. The temporary files are removed when the function exits, by return or by error, but not when the R process ends between two renames. Running a step again with the same inputs publishes the same tables, except ShearTable without seed.

An error raised before the first file is published reaches the caller unchanged, as the same condition object; that includes the error of the writer of a temporary file. An error raised after it, while pub-

lishing, calculating or validating, is signalled as a condition of class `newmarkPartialPublication`. Its message names the step and site that failed, the destination when the failure was a publication, the original message, the files already published and the files of the failing group that keep their previous content; its field `cause` holds the original condition, `effects` the effects record of the files already published, and `path` the destination whose temporary file or rename failed (`NULL` when the failure was not a publication). A rename that fails returns `FALSE`, with at most a warning, so it is reported as an error naming its destination and the reason.

Displacement exceedance product

`DnPOETable` reads the accepted shear table and each selected site's `DnDraws.Rds`, counts pointwise exceedances with `buildDnPOETable()` and publishes the site `DnPOETable.Rds`. It assembles the global file from available site tables with `assembleDnPOETables()`. It uses `ID.gmdp`, `TR.gmdp`, `NS`, `Dn_weights`, `subduction` and `DaH.gmdp`; site discovery uses `path.uhs`, as for the other site steps. It needs neither PGATable nor the kmaxTable step, consumes no random numbers and uses the same publication and partial-failure rules. The fractions are conditional on each demand and the population of the stored draws, not integrated hazard.

Examples

```
Root <- file.path(tempdir(), "newmarkProject")
dir.create(Root)
writeLines('{"path": {"tables": "data/newmark"}, "Vref.gmdp": 760,
  "Hs": [10, 12], "lo.min": 0.35, "lo.max": 0.075, "s.min": 2.5, "s.max": 3.5,
  "uscs": {"U1": ["ML", "MH"]}, "IDm": ["U1"], "seed": 20260915}',
  file.path(Root, "newmark.json"))
print(processProject(root = Root, steps = "ShearTable", dryRun = TRUE))

Effects <- processProject(root = Root, steps = "ShearTable")
Effects
readRDS(file.path(Root, Effects$path))[level == "mean"]

unlink(Root, recursive = TRUE)
```

---

propagateBeamPeriod *Fundamental period of the truncated shear beam for every profile realization*

Description

Solves, for each realization of `simulateSoilProfiles()`, the fundamental period of the truncated shear beam directly from the realization's own layers with `solveWedgePeriod()`, and keeps beside it the analytic period of the truncated shear beam evaluated on the realization's fitted `mo` and `VSo`, with the controls of both. The resulting population of periods stays paired with the other properties of each realization: it can be summarized with `summarizeProfiles()`, naming `"Ts"` or `"TsAnalytic"` among its `properties`, or propagated further. This is the calculation-before-summary route; the compatibility route of `buildShearTable()` evaluates the analytic period on summary rows instead.

Usage

```
propagateBeamPeriod(profiles, lo, method)
```

Arguments

profiles	The result of simulateSoilProfiles() or propagateSitePeriod(): one row per realization with columns Hs, mo, VSo and layers, the list column of layer tables whose thicknesses add up to Hs or exceed it by less than the deepest layer (section Height; see solveWedgePeriod()).
lo	One truncation ratio of the geometry strictly between 0 and 1, see buildGeometry().
method	The solution method of the root model of the analytic route: "exact", or the surrogate "lm" or "nlm"; see buildRootModel().

Value

A copy of profiles with these columns appended, one value per realization paired with the other properties of its row: lo; Ts (s), the numerical fundamental period of the realization's layers trimmed to Hs; TsMeshChange, its relative change under mesh doubling; TsAnalytic (s), the period of the analytic route; TsAnalyticDifference, TsAnalytic / Ts - 1; TsDivergent, TRUE when abs(TsAnalyticDifference) exceeds 0.10; and an, anExact and anWithinSupport, the root of the analytic route and its diagnostics. The population is that of profiles; TsMeshChange, TsAnalyticDifference and TsDivergent are numerical diagnostics, not physical dispersion. An invalid realization stops the call with its row.

Settings

Every layer of a realization is divided into two linear elements, and the mesh-doubling control of solveWedgePeriod() repeats the solve with four, so Ts is the period on four elements per layer. A realization is flagged divergent when the analytic period differs from the numerical one by more than 10 percent, $|T_{s,analytic}/T_s - 1| > 0.10$; the flag rejects nothing. Both values are design settings.

Height

The $\lceil H_s/h \rceil$ layers of sampleSoilLayers() reach below Hs when Hs is not a multiple of h. Before the numerical period, the deepest layer of such a realization (largest zm) is trimmed so that the thicknesses add up to Hs: the trimmed geometry is the one Ts uses. Only that thickness changes; the properties of every layer stay those sampled, and TsAnalytic keeps the Hs, mo and VSo of the row. The layers column is returned untrimmed, so the site period of propagateSitePeriod() is that of the sampled column. A sum that already meets the equality of solveWedgePeriod() is solved as it is; an excess not smaller than the deepest layer stops the call with the row. This is a design decision.

Analytic route

TsAnalytic takes the apex-to-base height $H_s/(1 - \lambda_o)$ of evaluateBeamPeriod() and the first characteristic root at mo and lo from evaluateRoot() with method. Each mo is the exponent of the profile that characterizeProfile() fits from the crest, used with the roots of the apex-origin profile of Dakoulas and Gazetas (1985): an approximation that the theory base measures

on a continuous profile at 1.4 to 6.0 percent for $m = 0.5$ and λ from 0.075 to 0.35, and whose size for each realization is `TsAnalyticDifference`, flagged by `TsDivergent`. `Ts` makes no such approximation: it is the period of the realization's layered shear beam.

Examples

```
set.seed(20260915)
Profiles <- simulateSoilProfiles(Hs = 10, USCS = c("ML", "MH"), NS = 3, h = 1,
                                 water = 0, POP = 100, Vref = 760)
Periods <- propagateBeamPeriod(Profiles, lo = 0.351, method = "nlm")
Periods[, .(mo, VSo, Ts, TsAnalytic, TsDivergent)]
```

`propagateSitePeriod` *Fundamental period of the soil profile for every realization*

Description

Solves, for each realization of `simulateSoilProfiles()`, the fundamental period T_r of its layered soil column on rigid base with `solveColumnPeriod()`, and keeps beside it the density-weighted Rayleigh estimate and its relative difference. The resulting population of site periods stays paired with the other properties of each realization: it can be summarized with `summarizeProfiles()`, naming `"Tr"` among its `properties`, or propagated further.

Usage

```
propagateSitePeriod(profiles)
```

Arguments

`profiles`	The result of `simulateSoilProfiles()` or `propagateBeamPeriod()`: one row per realization with the list column `layers` of layer tables (see `solveColumnPeriod()`).

Value

A copy of `profiles` with these columns appended, one value per realization paired with the other properties of its row: `Tr` (s), the numerical site period of the realization's column; `TrMeshChange`, its relative change under mesh doubling; `TrRayleighWeighted` (s), the density-weighted Rayleigh period of `estimateWeightedRayleighPeriod()`; and `TrRayleighWeightedDifference`, `TrRayleighWeighted / Tr - 1`. The population is that of `profiles`; `TrMeshChange` and `TrRayleighWeightedDifference` are numerical diagnostics, not physical dispersion. An invalid realization stops the call with its row.

Settings

Every layer of a realization is divided into two linear elements, and the mesh-doubling control of `solveColumnPeriod()` repeats the solve with four, so `Tr` is the period on four elements per layer: a design setting shared with `propagateBeamPeriod()`.

Rayleigh estimates

The column is the realization's layers stacked from the surface, so its height is the sum of their thicknesses, which exceeds `Hs` when `Hs` is not a multiple of the layer thickness of `sampleSoilLayers()`. `TrRayleighWeighted` never exceeds the exact period of that column (see `solveColumnPeriod()`). The `TrRayleigh` of `characterizeProfile()`, already in the realization table, is the constant-density estimate of `newmark` 1.7.1: it ignores the variation of the unit weight between layers, so it is not a bound of `Tr` when the unit weight varies.

Examples

```
set.seed(20260915)
Profiles <- simulateSoilProfiles(Hs = 10, USCS = c("ML", "MH"), NS = 3, h = 1,
                                 water = 0, POP = 100, Vref = 760)
propagateSitePeriod(Profiles)[, .(TrRayleigh, Tr, TrRayleighWeighted, TrMeshChange)]
```

`queryProjectTable` *Query a table of a project*

Description

Reads one table that `processProject()` published under `path.tables` and returns its rows whose `where` columns take any of the given values, with the selected `columns`. It writes nothing.

Usage

```
queryProjectTable(file, root, table, siteID, where, columns)
```

Arguments

`file`, `root`	As for `listProjectTables()`.
`table`	Name of the table.
`siteID`	`NULL` for a global table, or the identifier of the site whose table is read.
`where`	Named list of non-empty character vectors, one per filtered column; `list()` keeps every row.
`columns`	Names of the columns returned, in that order; `NULL` returns every column, `extrapolated` last in a `DnTable` or `kmaxTable`.

Details

The table is named as in the effects record of `processProject()` and in `listProjectTables()`: with `siteID` = `NULL`, one of the global tables `ShearTable`, `ShearDraws`, `DnPlotTable`, `DnTable`, `kmaxTable` and `DnPOETable` at the root of `path.tables`; with a site identifier, one of the tables `DnDraws`, `PGATable`, `DnTable`, `kmaxTable` and `DnPOETable` of that site under `<ID.gmdp>/<siteID>/`, which locates them with the `ID.gmdp` of the contract. The contract is read and checked, and its paths resolved against `root`, as `processProject()` does, and the table is read as `listProjectTables()` reads it.

Each element of where names a column and gives, as text, the values it may take; a row is returned when every named column takes one of its values. The text is converted to the type of the column, as the values arrive from the command line: a number as JSON spells it ("100", "1.35", "1e-3") for a double column and a whole number for an integer column, "NA" being a missing value in either, such as the TR of the MCE rows; a text column compares the text itself, so its levels are written as stored, "mean" or "0.50" for p. Numbers match exactly: a decimal names its nearest double, so the 17 significant digits of a stored value, as the JSON of a query writes it, select that value. An unknown table or site, a table whose file is absent, a column that the table does not have, a text that does not convert and a value that its column never takes stop the query naming them.

A DnTable or kmaxTable gains the column extrapolated, derived as it is read and never written: 1 when the ky of the row, or its kmax other than 0, lies outside the calibration range that buildKyLimits() gives its model at the mean PGA of the scenario (for DnTable, that of the PGATable of its site, whose absence stops the query naming the file) or, for a tree (ensemble, rigid, flexible), the range of any of its models stored for the same site, scenario, geometry and material, and 0 otherwise. The upper bound of AM88, the mean PGA, is compared with a relative tolerance of 1e-9: when that PGA exceeds 0.80 g the grid of buildKyGrid() ends at it, and its last point can exceed the PGA, as computed or as rounded in the CSV of PGATable, in the last significant digits. where filters the column and columns selects it as a stored column.

Value

A data.table with the columns of the rows of the table that pass where, in their stored order and with their stored types.

Examples

```
Root <- file.path(tempdir(), "newmarkQuery")
dir.create(Root)
writeLines('{"path": {"tables": "data/newmark"}, "Vref.gmdp": 760,
  "Hs": [10, 12], "lo.min": 0.35, "lo.max": 0.075, "s.min": 2.5, "s.max": 3.5,
  "uscs": {"U1": ["ML", "MH"]}, "IDm": ["U1"], "seed": 20260915}',
  file.path(Root, "newmark.json"))

processProject(root = Root, steps = "ShearTable")
queryProjectTable("newmark.json", root = Root, table = "ShearTable", siteID = NULL,
                  where = list(level = c("0.16", "mean", "0.84")),
                  columns = c("IDg", "level", "Ts", "VS30"))

unlink(Root, recursive = TRUE)
```

---

readHazardWorkbook *Hazard curves of a user workbook*

Description

Reads hazard curves tabulated by hand in an Excel workbook: one sheet per curve kind named p=<kind> (for example p=mean or p=0.84), a column Tn with oscillator periods in seconds and one column per intensity level in g, named by the level, holding annual rates of exceedance. Reading needs the suggested package readxl.

Usage

```
readHazardWorkbook(file, IT)
```

Arguments

file	Path of the .xlsx workbook.
IT	Investigation time in years over which the probabilities of exceedance are stated, finite and positive.

Value

A data.table with one row per positive cell: Tn (s), p (the sheet's kind; a numeric kind written as a number, text otherwise), IT (years), Sa (g), POE, AEP (1/year) and TR (years). Sheets without intensity columns are skipped with a warning.

Mathematics

Each cell is an annual rate AEP at the intensity level of its column; convertAEP() gives the probability of exceedance over the investigation time IT and the return period $1/AEP$. Cells with a rate that is not positive are dropped. The curves are taken as supplied: the workbook states which statistic of the hazard each sheet holds.

Examples

```
# The mean hazard curves of the package tests: five periods by 39 levels, sheet p=mean.
Workbook <- system.file("extdata", "hazardWorkbook.xlsx", package = "newmark")
if (requireNamespace("readxl", quietly = TRUE)) readHazardWorkbook(Workbook, IT = 50)
```

---

refineSiteMixture — *Site mixture at the first Gauss-Hermite order that agrees with the previous*

Description

Evaluates integrateSiteMixture() at the increasing quadrature orders and returns the site mixture of the first order whose quantiles and mean agree with those of the previous order, together with the order and the agreement reached, so that the numerical error of the result is visible.

Usage

```
refineSiteMixture(mixture, VS30, Vref, model, levels, orders, tolerance)
```

Arguments

`mixture`	The rock mixture, see `integrateSiteMixture()`.
`VS30`	One target velocity in m/s.
`Vref`	One reference velocity in m/s.
`model`	Site factor model of `evaluateSiteFactor()`, `"ST20"` or `"ST17"`.
`levels`	Probabilities in $(0, 1)$ whose quantiles enter the comparison with the mean.
`orders`	At least three increasing integers, each at least 2.
`tolerance`	One positive relative change.

Value

A list with `mixture` (the site mixture of the accepted order, as `integrateSiteMixture()` returns it), `order` (the accepted order) and `change` (the pair Δ of that order, named `all` and `mean`); `order` and `change` are `NA` when `VS30` equals `Vref`.

Mathematics

At order n_i the report R_i is `summarizeMixture()` of the site mixture at `levels` and the mean, at every period. With the relative change $\delta(a, b) = \max|a - b| / \max(|b|, \epsilon)$ over corresponding entries (ϵ the smallest positive double), the change between R_{i-1} and R_i is the pair $\Delta_i = (\delta$ over all entries, δ over the means$)$. The first change Δ_2 is the reference; order n_i, $i \geq 3$, is accepted when $\max \Delta_i \leq$ `tolerance` and each component of Δ_i does not exceed that of Δ_2 by more than 10^{-12}. No order accepted is an error. When `VS30` equals `Vref` the rock mixture is returned without quadrature.

Examples

```
Mixture <- data.table::CJ(Tn = c(0, 0.2, 1), k = c("a", "b"))
Mixture[, `:=`(muLn = log(0.4) + 0.1 * (k == "b") - 0.5 * Tn, sigmaLn = 0.5, w = 0.5)]
refineSiteMixture(Mixture, VS30 = 360, Vref = 760, model = "ST20",
                  levels = c(0.16, 0.50, 0.84), orders = c(8, 16, 32, 64),
                  tolerance = 2e-3)[c("order", "change")]
```

`refineSiteSpectrum` *Site spectrum at the first quadrature order that agrees with the previous*

Description

Evaluates `integrateSiteSpectrum()` at the increasing Gauss-Legendre `orders` and returns the site spectrum of the first order whose ordinates agree with those of the previous order, together with the order and the agreement reached, so that the numerical error of the result is visible.

Usage

```
refineSiteSpectrum(spectrum, VS30, Vref, model, orders, tolerance)
```

Arguments

spectrum	The rock quantile spectrum, see integrateSiteSpectrum().
VS30	One target velocity in m/s.
Vref	One reference velocity in m/s.
model	Site factor model of evaluateSiteFactor(), "ST20" or "ST17".
orders	At least two increasing integers, each at least 4.
tolerance	One positive relative change.

Value

A list with spectrum (the site spectrum of the accepted order, as integrateSiteSpectrum() returns it), order (the accepted order) and change (δ of that order).

Mathematics

With S_i the site ordinates (SaF, every period and level) at order n_i and the relative change $\delta_i = \max|S_{i-1} - S_i| / \max(|S_i|, \epsilon)$ (ϵ the smallest positive double), order n_i, $i \geq 2$, is accepted when δ_i is finite and at most tolerance. No order accepted is an error.

Examples

```
P <- c("0.05", "0.10", "0.16", "0.50", "0.84", "0.90", "0.95", "mean")
Factor <- c(0.45, 0.60, 0.72, 0.95, 1.30, 1.48, 1.78, 1.00)
Spectrum <- data.table::data.table(Tn = rep(c(0, 0.2, 1), each = 8), p = rep(P, 3),
                                   Sa = rep(c(0.30, 0.45, 0.24), each = 8) * rep(Factor, 3))
refineSiteSpectrum(Spectrum, VS30 = 270, Vref = 760, model = "ST20",
                   orders = c(32, 64, 128), tolerance = 2e-3)[c("order", "change")]
```

---

sampleIntensities *Paired intensity measures of a displacement scenario*

Description

Draws NS joint realizations of the intensity measures that the six displacement models read for a sliding mass of initial fundamental period Ts: peak ground acceleration, the spectral accelerations at the degraded periods $1.3T_s$ and $1.5T_s$, and the peak ground velocity and Arias intensity derived from each sampled PGA. One row is one realization; every column of a row belongs to the same draw.

Usage

```
sampleIntensities(spectrum, Ts, NS, zTrunc)
```

Arguments

spectrum	The spectrum of the scenario as sampleSpectra() accepts it: a quantile table (Tn, p, Sa in g) or an analytic mixture of $\ln S_a$ (Tn, k, muLn, sigmaLn, w).
Ts	One finite positive initial fundamental period of the sliding mass in seconds.
NS	Number of realizations, one positive integer.
zTrunc	Truncation of the copula marginals in standard deviations for an analytic mixture, or NA, see sampleSpectra().

Value

A data.table with one row per realization: realization, PGA (g), Sa13 and Sa15 (spectral accelerations at $1.3T_s$ and $1.5T_s$, g), PGV (cm/s) and AI (m/s).

Mathematics

The periods $(0,\ 1.3T_s,\ 1.5T_s)$ are sampled jointly with sampleSpectra() under the correlation matrix $C_{jk} = \rho(T_j, T_k)$ of correlateResiduals(), with $C_{jj} = 1$ and the zero period evaluated at 0.01 s. A quantile spectrum that lacks $1.3T_s$ or $1.5T_s$ is first interpolated there with interpolateSpectrum(); an analytic mixture is interpolated by sampleSpectra() itself. Then PGV = estimatePGV() and I_a = estimateAI() of each sampled PGA. The draws consume NS trivariate normal deviates of the caller's random stream.

Examples

```
Sigma <- c(0.2, 0.45, 0.75)
Spectrum <- data.frame(Tn = c(0, 0.4, 1), k = "oq", muLn = log(c(0.8, 0.55, 0.42)) - Sigma^2 / 2,
                       sigmaLn = Sigma, w = 1)
set.seed(46)
sampleIntensities(Spectrum, Ts = 0.4, NS = 5, zTrunc = NA)
```

---

sampleSoilLayers *Sample one synthetic soil column from a USCS material pool*

Description

Discretizes a column of height Hs into layers of thickness h and draws the index properties of each layer from the ranges tabulated by USCS class. This is the stochastic step of the synthetic route; the physics of the sampled layers is evaluated afterwards by evaluateSoilLayers().

Usage

```
sampleSoilLayers(Hs, USCS, h)
```

Arguments

Hs	One positive height of the column in metres.
USCS	Character vector of USCS codes forming the material pool; each layer draws its class from this vector.
h	One positive layer thickness in metres.

Value

A data.table with one row per layer, from the surface down: layer, USCS, hs (thickness, m), zo, zi, zm (top, bottom and midpoint depth, m), emin, eo, emax (void ratios), gsat (saturated unit weight, kN/m3), LL and IP (percent; NA for coarse classes, LL also NA for peat, IP 0 where the A line is null). Every value is one realization, not a statistic.

Mathematics

The column has $N_L = \lceil H_s/h \rceil$ layers of thickness h, so its depth $N_L h$ can exceed Hs when Hs is not a multiple of h. A quotient that exceeds an integer by at most 4ϵ times its value, with $\epsilon = 2^{-52}$ the double-precision epsilon, counts as that integer, so decimal lengths whose exact quotient is a whole number, as with 2.1 m divided by 0.3 m, do not add a layer by floating-point rounding. Layer k spans $[(k-1)h, kh]$ with midpoint z_m. The class of every layer is drawn uniformly, with replacement, from USCS. For a layer of class c, with the intervals of VoidRatiosUSCS and UnitWeightRanges,

$$e_{min} \sim U(e^{lo}_{min}, e^{hi}_{min}), \quad e_{max} \sim U(e^{lo}_{max}, e^{hi}_{max}), \quad e_o \sim U(e_{min}, e_{max}), \quad \gamma_{sat} \sim U(\gamma^{lo}_{sat}, \gamma^{hi}_{sat}).$$

Fine soils also draw their Atterberg limits: the liquid limit $LL \sim U(50, 100)$ for CH, MH and OH and $LL \sim U(8, 50)$ for CL, ML and OL; peat (PT) draws no liquid limit. With the Casagrande lines $U = 0.9(LL - 8)$ and $A = 0.73(LL - 20)$ for $LL \geq 20$ (else $A = 0$), the plasticity index is $IP \sim U(A, U)$ for clays and $IP \sim U(0, A)$ for silts and organic soils; when the A line is null (peat, or a liquid limit below 20) that draw is the degenerate $U(0, 0)$, so IP is 0 and the random stream is not consumed, as in newmark 1.7.1.

Random stream

The draws consume the caller's random stream in the order class vector, then for each layer in depth order e_{min}, e_{max}, e_o, γ_{sat}, and for fine soils LL then IP. This is the order of the newmark 1.7.1 generator, so a seeded stream reproduces its columns.

Examples

```
set.seed(1)
sampleSoilLayers(Hs = 5, USCS = c("ML", "MH"), h = 1)
```

`sampleSpectra` *Correlated sample of spectral accelerations*

Description

Draws `NS` joint realizations of the spectral accelerations at the oscillator periods `Tn` through a Gaussian copula: the dependence comes from the correlation matrix of the log residuals and each marginal from the spectrum, given either as tabulated quantiles or as an analytic lognormal mixture. The rows that share `realization` form one paired realization of every ordinate.

Usage

```
sampleSpectra(spectrum, Tn, correlation, NS, zTrunc)
```

Arguments

`spectrum`	A quantile spectrum (`Tn`, `p`, `Sa` in g, with at least three quantile rows at every requested period, each with a probability strictly between 0 and 1 and a finite positive `Sa`, and optionally a `"mean"` row with a finite positive `Sa`), or an analytic mixture of $\ln S_a$ (`Tn`, `k`, `muLn`, `sigmaLn`, `w`, with rows at `Tn == 0` and weights summing to one at every period).
`Tn`	Requested oscillator periods in seconds, finite and non-negative, in any order; the result lists them in this order.
`correlation`	The symmetric correlation matrix of the log residuals at `Tn`, with unit diagonal, for example from `correlateResiduals()`. Correlations given only between a reference period `Tn[1]` and the others, a vector `rho`, form the matrix `C <- diag(length(Tn)); C[1, -1] <- C[-1, 1] <- rho`.
`NS`	Number of realizations, one positive integer.
`zTrunc`	Symmetric truncation of the copula marginals in standard deviations, one finite positive number for an analytic mixture, or `NA` for untruncated sampling.

Value

A data.table of paired realizations with `realization` (realization, 1 to `NS`), `Tn` (s) and `Sa` (g), one row per realization and requested period, ordered by the position of the period in `Tn` and then by `realization`. The rows that share `realization` form one joint draw.

Mathematics

Standard normal vectors $Z_s \sim N(0, C)$, $s = 1, \ldots, N_S$, are drawn with `mvtnorm::rmvnorm` (eigen factorization), after adding $|\lambda_{min}| + 10^{-8}$ to the diagonal of C when its smallest eigenvalue is not positive, and mapped to uniforms $U = \Phi(Z)$. With a truncation z_t, the uniforms are remapped to $\Phi(-z_t) + U\,(\Phi(z_t) - \Phi(-z_t))$, which keeps their ranks and restricts them to $[\Phi(-z_t), \Phi(z_t)]$. Each column is then inverted through the quantile function Q of $\ln S_a$ of its marginal and rescaled to its mean,

$$S_{a,s} = e^{Q(U_s)}\,\frac{\bar{S}_a}{\frac{1}{N_S}\sum_{r=1}^{N_S} e^{Q(U_r)}};$$

this finite-sample recentering multiplies every draw of one period by one positive constant: it keeps their ranks, but no fixed bound on S_a.

For a quantile spectrum, Q is `buildQuantileSpline()` of the rows at that exact period and $\bar{S}_a$ is its `"mean"` row; a period without a mean row takes the sample mean, which leaves the draws at $e^{Q(U_s)}$. A quantile spectrum is sampled untruncated, so U spans (0, 1) and reaches the end pieces that the spline extrapolates beyond its first and last tabulated probabilities. For a mixture $F(x) = \sum_k w_k \Phi((x - \mu_k)/\sigma_k)$ of $\ln S_a$, Q is the bisection inverse of the untruncated F, so with a truncation the draws of a tabulated period lie, before the recentering, between the mixture quantiles of probability $\Phi(-z_t)$ and $\Phi(z_t)$. The mean is $\sum_k w_k e^{\mu_k + \sigma_k^2/2}$, or with truncation that of the components truncated individually at $\pm z_t$,

$$\sum_k w_k e^{\mu_k + \sigma_k^2/2} \frac{\Phi(z_t - \sigma_k) - \Phi(-z_t - \sigma_k)}{\Phi(z_t) - \Phi(-z_t)},$$

while the draws invert the untruncated mixture over the restricted probabilities. A mixture period that is not tabulated interpolates Q linearly and the mean geometrically in the coordinate $\ln(T + 10^{-8})$ of the tabulated periods, the requested period entering as $\ln T$ ($T > 0$) or $-\infty$ ($T = 0$), between the two neighbouring tabulated periods, and takes the end period outside them.

Validation, including the quantile and mean rows of every requested period, happens before any draw, so an invalid request leaves the random stream untouched; a valid one consumes `NS` multivariate normal draws of the caller's stream.

Examples

```
Spectrum <- data.frame(Tn = rep(c(0, 1), each = 4),
                       p = rep(c("0.16", "0.50", "0.84", "mean"), 2),
                       Sa = c(0.30, 0.48, 0.72, 0.50, 0.10, 0.19, 0.31, 0.20))
C <- diag(2)
C[1, 2] <- C[2, 1] <- correlateResiduals(Tn = 1, T0 = 0.01)
set.seed(1)
Sample <- sampleSpectra(Spectrum, Tn = c(0, 1), correlation = C, NS = 20, zTrunc = NA)
Sample[, .(mean = mean(Sa)), by = Tn]
```

---

selectDisplacementModels

Displacement models active in a tectonic regime

Description

Turns the on/off switches of the six displacement models of a project into the named weights of the active models. The flexible model comes in two fits for two populations: `BM17`, Bray, Macedo and Travasarou (2018), for subduction interface earthquakes, and `BM19`, Bray and Macedo (2019), for shallow crustal ones; both are declared with the same switch and the regime keeps the one fitted for it.

Usage

```
selectDisplacementModels(switches, subduction)
```

Arguments

switches	Named switches, each 0 or 1, of distinct models among AM88, JB07, BT07, SR08, BM17 and BM19, with BM17 and BM19 declared together and alike. A non-binary value would change the ensemble mean and quantiles without any product recording it, so it is rejected.
subduction	One logical: TRUE keeps BM17, FALSE keeps BM19.

Value

The named weights (all 1) of the active models, in the order of switches. A message reports which half of the flexible pair the regime switched off.

Examples

```
selectDisplacementModels(switches = c(AM88 = 1, JB07 = 1, BT07 = 1, SR08 = 1, BM17 = 1, BM19 = 1),
                         subduction = FALSE)
```

ShearModelParameters *Small-strain shear-modulus correlations by soil group*

Description

Coefficients of the empirical correlations $G_0 = AF(e)(\sigma'_0)^n$ tabulated by Ishihara (1996), grouped by soil family. Ishihara's Table 6.1 states σ'_0 and G_0 in kPa; this table keeps the stress in kPa and expresses A for the modulus in MPa (7.00 for the round-grained Ottawa sand of Hardin and Richart (1963), printed 7000 for the modulus in kPa). evaluateSoilLayers() applies each row at the octahedral mean effective stress p_m (kPa) of a layer as

$$G = AF(e)\, OCR^{m_1}\, 100^n (p_m/100)^n = AF(e)\, OCR^{m_1}\, p_m^n$$

in MPa, with the overconsolidation factor OCR^{m_1} of Hardin and Drnevich (1972) for fine soils ($m_1 = 0$ otherwise). The factor $100^n(p_m/100)^n$ equals p_m^n; it does not normalize A, which is not the modulus at 100 kPa. Carried byte-identical from newmark 1.7.1 (data-raw/MaterialProperties.xlsx).

Usage

```
ShearModelParameters
```

Format

A data.table with 20 rows and 10 variables:

ModelID Character: correlation identifier; SS75 uses the linear void-ratio function, every other row the quadratic one.

GroupID Character: Gravels, Sands or Fines.

NameID, AuthorID Character: material and authors.

emin, emax Double: void-ratio interval of the correlation.

A Double: coefficient of $F(e)\,p^n$ for the modulus in MPa with the stress p in kPa.

Ce Double: void-ratio coefficient of $F(e)$.

n Double: stress exponent.

UN Character: units of the modulus.

Source

Ishihara, K. (1996). Soil Behaviour in Earthquake Geotechnics. Oxford University Press, Table 6.1.

`simulateSiteSpectrum` *Paired realizations of a site spectrum from a rock quantile spectrum*

Description

Draws, at every oscillator period, `NS` paired realizations of the rock PGA, the rock ordinate and the site ordinate from a rock spectrum given by its mean and quantiles and from the site factor of `evaluateSiteFactor()`. The realizations are returned before any summary; reduce them per period with `summarizeProfiles()` (see the example). `integrateSiteSpectrum()` evaluates the same model without random numbers.

Usage

```
simulateSiteSpectrum(spectrum, VS30, Vref, model, NS)
```

Arguments

`spectrum`	The rock quantile spectrum, see `integrateSiteSpectrum()`.
`VS30`	One target velocity in m/s.
`Vref`	One reference velocity in m/s; 760 or 3000 unless it equals `VS30`.
`model`	Site factor model of `evaluateSiteFactor()`, `"ST20"` or `"ST17"`.
`NS`	One positive integer: realizations per period.

Value

A data.table with one row per period and realization: `Tn` (s), `realization` (1 to `NS`), `PGA`, `Sa` and `SaF` (g; rock PGA, rock ordinate and site ordinate of the realization), in the order drawn. Advances the random number stream.

Mathematics

At PGA and at each period the rock quantile function $Q(u) = \ln S_a$ is that of `integrateSiteSpectrum()`: the Hyman monotone spline through the tabulated knots $(p_k, \ln S_{a,k})$, continued below the first and above the last knot by the adjacent secant slopes (floored at zero) and clamped to $u \in [0,1]$. It is non-decreasing, so every copula uniform maps to the rock quantile of its rank. At period T_n, (Z_{0j}, Z_{nj}), $j = 1, \ldots, n_s$, are drawn from the bivariate normal with unit variances and correlation $\rho_{BJ}(T_n, 0)$ of `correlateResiduals()` (`mvtnorm::rmvnorm()`), a Gaussian copula between PGA and the ordinate, and each marginal sample is re-centred on its tabulated mean:

$$PGA_j = e^{Q_0(\Phi(Z_{0j}))} \frac{\overline{PGA}}{\langle e^{Q_0(\Phi(Z_0))} \rangle}, \quad S_{a,j} = e^{Q_n(\Phi(Z_{nj}))} \frac{\bar{S}_a}{\langle e^{Q_n(\Phi(Z_n))} \rangle},$$

with $\langle \cdot \rangle$ the sample mean. Then $\ln F_j \sim N(\mu_F(PGA_j), \sigma_F(PGA_j))$ independently and $S_{aF,j} = S_{a,j} F_j$. When `VS30` equals `Vref`, for any reference velocity, F is the unit factor ($\mu_F = 0$, $\sigma_F = 0$) and $S_{aF,j} = S_{a,j}$.

The triple $(PGA_j, S_{a,j}, S_{aF,j})$ is paired within one period only: every period draws its own copula sample, so rows with the same `realization` at different periods do not form one spectrum. Periods are drawn in increasing order, each consuming from the caller's random stream the `NS` copula draws and then, where the factor dispersion is positive, the `NS` factor residuals (`stats::rnorm` draws nothing where the dispersion is zero, as ST20 gives from 3000 m/s against a 760 m/s reference); when `VS30` equals `Vref` no factor residual is drawn. Sample statistics carry Monte Carlo error of order $n_s^{-1/2}$.

Examples

```
P <- c("0.05", "0.10", "0.16", "0.50", "0.84", "0.90", "0.95", "mean")
Factor <- c(0.45, 0.60, 0.72, 0.95, 1.30, 1.48, 1.78, 1.00)
Spectrum <- data.table::data.table(Tn = rep(c(0, 0.2, 1), each = 8), p = rep(P, 3),
                                   Sa = rep(c(0.30, 0.45, 0.24), each = 8) * rep(Factor, 3))
set.seed(919)
Draws <- simulateSiteSpectrum(Spectrum, VS30 = 270, Vref = 760, model = "ST20", NS = 300)
Draws[, summarizeProfiles(.SD, levels = c(0.16, 0.5, 0.84, "mean"), properties = "SaF"), by = Tn]
```

`simulateSoilProfiles` *Simulate paired soil-profile realizations for a material pool*

Description

Draws `NS` synthetic columns with `sampleSoilLayers()`, evaluates each with `evaluateSoilLayers()` and characterizes it with `characterizeProfile()`. Every realization is kept with its layers, so the profile quantities stay paired: the `Go`, `mo`, `VSo`, `VS30` and `TrRayleigh` of one row belong to the same column. `summarizeProfiles()` reduces this population to marginal statistics; `propagateSitePeriod()` and `propagateBeamPeriod()` solve the fundamental periods of the soil profile and truncated shear beam on each row before any reduction.

Usage

```
simulateSoilProfiles(Hs, USCS, NS, h, water, POP, Vref, m1 = HardinExponent)
```

Arguments

Hs	One positive height in metres.
USCS	Character vector of USCS codes forming the material pool.
NS	Number of realizations, a positive integer.
h	One positive layer thickness in metres.
water	Water-table height as a fraction of Hs.
POP	Preconsolidation pressure in kPa.
Vref	One positive reference velocity in m/s, see estimateVS30().
m1	Nodes of the plasticity exponent, see evaluateSoilLayers(), whose published table is also the default here.

Value

A data.table with one row per realization: realization, Hs, Hw (water-table height, m), POP, Vref, the columns of characterizeProfile() and layers, a list column holding the evaluated layer table of the row. The rows are the population generated by the pool under the caller's random stream.

Examples

```
set.seed(20260915)
simulateSoilProfiles(Hs = 10, USCS = c("ML", "MH"), NS = 3, h = 1,
                     water = 0, POP = 100, Vref = 760)
```

SiteClass *ASCE 7-22 site classes*

Description

Site classes of ASCE/SEI 7-22 Table 20.2-1 with their shear-wave velocity intervals in m/s and ft/s. Carried byte-identical from newmark 1.7.1.

Usage

```
SiteClass
```

Format

A data.table with 9 rows and 4 variables: SC, Description, Vs30 (m/s) and Vs30 (ft/s).

`solveColumnPeriod` *Fundamental period of a layered soil profile as a Galerkin eigenvalue problem*

Description

Solves the fundamental period of the soil column on rigid base, the site period T_r, directly from the layers of one realization, and returns it with its controls: the change under mesh doubling and the density-weighted Rayleigh period beside it. `solveWedgePeriod()` solves the fundamental period of the truncated shear beam with the same numerical method, and `solveProfilePeriods()` applies both to a list of realizations.

Usage

```
solveColumnPeriod(layers, elements)
```

Arguments

`layers`	One layer table, such as an element of the `layers` column of `simulateSoilProfiles()`: a data.frame or data.table with one row per layer and numeric columns `hs` (thickness, m), `zm` (midpoint depth, m; distinct, it orders the layers), `gsat` (unit weight, kN/m3; any unit, only the ratios between layers matter) and `VSm` (m/s) or, when `VSm` is absent, `Gm` (MPa), converted with $V_S = \sqrt{9.81 \cdot 1000\, G_m/\gamma_{sat}}$ as in `evaluateSoilLayers()`. Every value must be finite and positive; other columns are ignored.
`elements`	One positive whole number of linear elements per layer of the declared mesh; the control doubles it.

Value

A one-row data.table: `Tr` (s), the numerical site period on `2 * elements` elements per layer; `TrMeshChange`, the relative change $1 - T_e/T_{2e}$; `TrRayleighWeighted` (s), the density-weighted Rayleigh period; and `TrRayleighWeightedDifference`, `TrRayleighWeighted / Tr - 1`. Each value is a conditional evaluation of the supplied layers; the two controls are numerical diagnostics, not physical dispersion.

Mathematics

Dobry, Oweis and Urzua (1976) model the profile as a shear beam on rigid rock (hydrated copy, `eq-beam-wave-equation`, `eq-beam-rock-condition` and `eq-beam-surface-condition`):

$$\rho \frac{\partial^2 u}{\partial t^2} = \frac{\partial}{\partial z}\left(G \frac{\partial u}{\partial z}\right), \qquad u(H,t) = 0, \qquad \left(G \frac{\partial u}{\partial z}\right)_{z=0} = 0,$$

with z the depth below the surface, H the depth of the rigid base and $\rho = \gamma/g$ (section Basic Equation). For harmonic motion $u = X(z)e^{i\omega t}$ (`eq-beam-harmonic-motion`) it is the regular Sturm-Liouville problem $-(GX')' = \omega^2 \rho X$ on $[0, H]$, whose smallest eigenvalue gives $T_r =$

$2\pi/\omega_1$. Multiplying by a test function v with $v(H) = 0$ and integrating by parts, the surface term vanishes by the free condition and

$$\int_0^H GX'v'\,dz = \omega^2 \int_0^H \rho X v\,dz.$$

On layer k, of thickness h_k, $G_k = \rho_k V_k^2$ and $\rho_k = \gamma_k/g$ are constant; the layers are stacked from the surface in the order of their midpoint depths `zm`, and H is the sum of their thicknesses.

Numerical method

The weak form is discretised with continuous piecewise-linear functions on a mesh that follows the layers: every layer is divided into `elements` equal elements, so every interface is a node, and the node at the bottom of the last layer is fixed. With a width $w(z)$, 1 for the column and the distance to the apex for the wedge of `solveWedgePeriod()`, the integrals of an element of length L between an upper node a and a lower node b are exact, because G and ρ are constant within a layer and w is linear: the element stiffness is

$$k_e = G\frac{w_a + w_b}{2L},$$

assembled as $k_e(\delta_a - \delta_b)(\delta_a - \delta_b)^T$, and the consistent mass has diagonal entries $\rho L(3w_a + w_b)/12$ and $\rho L(w_a + 3w_b)/12$ and coupling $\rho L(w_a + w_b)/12$. The generalized eigenvalue problem $K\phi = \omega^2 M\phi$ has K and M symmetric positive definite for positive G and ρ with the base fixed. The stiffness factors as $K = U^T U$ with $U = D^{1/2}C$, where D is the diagonal of the k_e and C the bidiagonal difference matrix (1 on the diagonal, -1 above it: element e joins nodes e and $e+1$, the base node removed); U is the Cholesky factor of K, and its inverse is explicit because C^{-1} is the upper triangle of ones. The Cholesky reduction gives the symmetric matrix

$$B = U^{-T}MU^{-1} = D^{-1/2}SD^{-1/2}, \qquad S_{ij} = c_{\min(i,j)}\ (i \neq j), \qquad S_{ii} = c_i - M_{i,i+1},$$

with c_i the consistent mass from the top to node i (the cumulative row sums of M). The eigenvalues of B are $1/\omega^2$, so the fundamental mode is its largest eigenvalue μ_1, computed by the direct symmetric solver `eigen(symmetric = TRUE)`, and $T = 2\pi\sqrt{\mu_1}$; no iterative solver or convergence loop is involved. Every entry of S is a sum of positive masses, and the largest eigenvalue of a symmetric matrix is resolved to relative machine precision.

The discretisation is a Rayleigh-Ritz method: the numerical period never exceeds the exact period of the layered beam, and it does not decrease when every element is halved, since the trial spaces are nested. The period returned is the one with `2 * elements` elements per layer, and the mesh change $1 - T_e/T_{2e} \geq 0$ between the declared and the doubled mesh is its convergence certificate. Within a layer the solution is smooth and the interfaces are nodes, so the error decreases with the square of the element length and is asymptotically a third of the mesh change. A profile whose stiffness tends to zero at the surface converges more slowly. No threshold is applied to the mesh change: it is returned for the consumer.

Rayleigh bound

`TrRayleighWeighted` is `estimateWeightedRayleighPeriod()` on the same layers. It evaluates the Rayleigh quotient of Dobry, Oweis and Urzua (`eq-rayleigh-frequency`), $\omega^2 = \int \rho V^2 X'^2 dz / \int \rho X^2 dz$, on the first-iteration deflection of `eq-rayleigh-mode-iteration`. Let u be the piecewise-linear shape whose values U_k at the interfaces are that deflection accumulated from the base with the slopes $s_k = \sigma_k/(\gamma_k V_k^2)$, σ_k the overburden at the midpoint of layer k, and $X_k = (U_{k-1} +$

$U_k)/2$. Summation by parts gives exactly $\int \rho V^2 u'^2 dz = g^{-1} \sum_k \gamma_k h_k X_k$, while $\int \rho u^2 dz = g^{-1} \sum_k \gamma_k h_k (U_{k-1}^2 + U_{k-1}U_k + U_k^2)/3 \geq g^{-1} \sum_k \gamma_k h_k X_k^2$. The estimate's ω^2 is therefore at least the Rayleigh quotient of u, which is at least ω_1^2 by Rayleigh's principle: `TrRayleighWeighted` never exceeds the exact period of the layered column, and `TrRayleighWeightedDifference` cannot exceed the discretisation error of `Tr`, asymptotically `TrMeshChange / 3`. With a uniform unit weight it is the constant-density Method 7 of `estimateRayleighPeriod()`.

The package tests reproduce the uniform layer $T = 4H/V$ (`eq-uniform-period`), the power-law stratum of Dobry, Oweis and Urzua (`eq-power-period`, with the height exponent $(2 - p)/2$), their two-layer equation (`eq-twolayer-period-equation`) and the n-layer frequency equation of Charara (2019, `eq-frequency-equation-condensed`).

Examples

```
Layers <- data.frame(hs = c(24, 8), zm = c(12, 28), gsat = c(17.66, 19.62),
                     VSm = sqrt(c(20e6 / 1800, 90e6 / 2000)))
solveColumnPeriod(Layers, elements = 8)
```

`solveKmax` *Yield coefficient that limits each realized displacement curve*

Description

For every realization of one displacement model, finds the yield coefficient k_{max} at which the realized Newmark displacement equals each admissible displacement D_a, or the censored value that the rules below assign when no such equality exists on the grid (null demand, grid floor, rigid-block cap). The realizations stay separate and paired with their row of `Dn`; `summarizeKmax()` reduces them, alone or pooled with other models, to a mean and quantiles.

Usage

```
solveKmax(ky, Dn, PGA, model, Da)
```

Arguments

`ky`	Strictly increasing positive yield coefficients in g, at least two; a continuation beyond the grid needs three positive displacements at the top of the curve.
`Dn`	Matrix of finite non-negative displacements in cm, one row per realization and one column per `ky`.
`PGA`	Sampled peak ground acceleration in g of each realization (one per row of `Dn`), or `NULL` when the realizations carry none; the continuation of AM88 and SR08 and the inverse of a constant AM88 curve require it.
`model`	The displacement model of the realizations: `"AM88"`, `"JB07"`, `"SR08"`, `"BT07"`, `"BM17"` or `"BM19"`.
`Da`	Unique positive admissible displacements in cm.

Value

A data.table with one row per realization and `Da`: `IDn`, `realization` (the row of `Dn`), `Da` (cm), `kmax` (g), the yield coefficient at which that realization of the model reaches `Da`, and `beyond` (`TRUE` when the root comes from the model's continuation beyond the grid or from the constant-AM88 inverse; `FALSE` for null demand, interpolated roots and the grid floor).

Mathematics

A realization gives the displacements D_j (cm) at the yield coefficients $k_{y,1} < \cdots < k_{y,J}$ (g). Displacements below 10^{-4} cm (one micron) are set to zero and the curve is replaced by its upper monotone envelope $\bar{D}_j = \max_{i \geq j} D_i$, non-increasing in k_y. The coefficient k_{max} solves $\bar{D}(k_{max}) = D_a$:

- when $D_a > \bar{D}_1$, the ceiling of the realization, every coefficient meets the target and $k_{max} = 0$ (null demand);
- between two positive envelope values, $\log k_{max}$ is linear in $\log \bar{D}$;
- below the smallest positive envelope value of a curve that reaches the zero floor, k_{max} is the first k_y of that floor (the resolution of the grid);
- below a curve that stays positive on the whole grid, the root is continued beyond the grid with the closed form of the model, whose free constants are recovered from the top of the drawn curve. BT07, BM17 and BM19 are quadratic in $L = \log k_y$: the parabola $\log D = c_1 L^2 + c_2 L + c_3$ through the three largest positive points is solved for its larger root, and a curvature $-10^{-8} < c_1 \leq 10^{-8}$, the linear limit of the same closed form, takes the line through the top two points (a curvature above 10^{-8} stops with an error). JB07 is a power law in k_y, inverted along the line through the top two points. AM88 and SR08 are written in $r = k_y / PGA$, with PGA the sampled peak ground acceleration of the realization: $\log D = C + 2.53 \log(1 - r) - 1.09 \log r$ for AM88 and $\log D = C - 5.24r - 18.78r^2 + 42.01r^3 - 29.15r^4 + 0.56\varepsilon r$ for SR08, with ε recovered from the top two points and C from the top point. The root is sought for $r \leq 0.9999$, the rigid-block limit; a target below the curve at that limit gives $k_{max} = 0.9999\, PGA$.

An AM88 realization whose displacements are one positive constant lies wholly at the rigid-block limit of the model; its exact inverse solves $\mu(r) = \log D_a - \log D + \mu(0.9999)$, with $\mu(r) = 0.90 \log 10 + 2.53 \log(1 - r) - 1.09 \log r$, and $k_{max} = PGA\, r$. Roots found by continuation or by this inverse lie outside the stored grid and are flagged with `beyond`; null demand, interpolated roots and the grid floor are not.

No random number is drawn and nothing is rounded.

Sources of the closed forms

BT07: Bray and Travasarou (2007), Eq. 5 (p. 387). BM17: Bray, Macedo and Travasarou (2018), Eq. 4 (p. 04017124-5). BM19: Bray and Macedo (2019), Eqs. 3a, 5 and 7, with the corrections of 2023. JB07: Jibson (2007), Eq. 10 (p. 215). SR08: Saygili and Rathje (2008), Eq. 6 and Tables 1-2 (pp. 797-800). AM88: Ambraseys and Menu (1988) as `newmark` 1.7.1 implemented it; the original paper was not verified locally. The full entries are in `evaluateRigidDisplacement()` and `evaluateFlexibleDisplacement()`. The displacement floor of 10^{-4} cm, the monotone envelope, the rigid-block cap $r \leq 0.9999$ and the first-zero-grid rule are policies of this implementation, not published model terms.

Examples

```
ky <- c(0.05, 0.10, 0.20, 0.40)
Dn <- matrix(0.02 / ky, nrow = 4, ncol = 4, byrow = TRUE)
solveKmax(ky = ky, Dn = Dn, PGA = NULL, model = "BT07", Da = c(0.1, 1))
```

---

solveProfilePeriods *Fundamental periods of the soil profile and truncated shear beam for every realization*

Description

Applies solveWedgePeriod() and solveColumnPeriod() to each layer table of a list, with the paired height, truncation ratio, exponent and basal velocity of the same realization, so that the numerical periods and their controls stay paired with the realization before any summary.

Usage

```
solveProfilePeriods(layers, Hs, lo, mo, VSo, method, elements)
```

Arguments

layers	A non-empty list of layer tables, such as the layers column of simulateSoilProfiles(); see solveColumnPeriod().
Hs	Slope heights in metres: one for every table, or one per table.
lo	Truncation ratios in $[0, 1)$: one for every table, or one per table.
mo	Inhomogeneity exponents of the analytic route, one per table.
VSo	Basal shear-wave velocities of the analytic route in m/s, one per table.
method	The solution method of the root model: "exact", or the surrogate "lm" or "nlm"; see buildRootModel().
elements	One positive whole number of linear elements per layer of the declared mesh; the control doubles it.

Value

A data.table with one row per layer table, in the order of the list, and the columns of solveWedgePeriod() followed by those of solveColumnPeriod(). An invalid table stops the call with its position in the list.

Examples

```
set.seed(20260915)
Profiles <- simulateSoilProfiles(Hs = 10, USCS = c("ML", "MH"), NS = 2, h = 1,
                                 water = 0, POP = 100, Vref = 760)
solveProfilePeriods(Profiles$layers, Hs = 10, lo = 0.351, mo = Profiles$mo,
                    VSo = Profiles$VSo, method = "exact", elements = 2)
```

`solveWedgePeriod` *Fundamental period of the truncated shear beam as a Galerkin eigenvalue problem*

Description

Solves the fundamental period T_s of the truncated inhomogeneous shear beam directly from the layers of one realization, and returns it with its controls: the change under mesh doubling and the period of the analytic route beside it, with their relative difference. `solveColumnPeriod()` documents the shared numerical method, and `solveProfilePeriods()` applies both to a list of realizations.

Usage

```
solveWedgePeriod(layers, Hs, lo, mo, VSo, method, elements)
```

Arguments

`layers`	One layer table, as for `solveColumnPeriod()`, whose thicknesses add up to `Hs`.
`Hs`	One positive slope height from crest to base, in metres.
`lo`	One truncation ratio λ_o in $[0, 1)$; see `buildGeometry()`.
`mo`	One inhomogeneity exponent of the analytic route, the `mo` that `characterizeProfile()` fitted on the same layers.
`VSo`	One basal shear-wave velocity of the analytic route in m/s, the `VSo` of the same fit.
`method`	The solution method of the root model of the analytic route: `"exact"`, or the surrogate `"lm"` or `"nlm"`; see `buildRootModel()`.
`elements`	One positive whole number of linear elements per layer of the declared mesh; the control doubles it.

Value

A one-row data.table: `Ts` (s), the numerical fundamental period on `2 * elements` elements per layer; `TsMeshChange`, the relative change $1 - T_e/T_{2e}$; `TsAnalytic` (s), the period of the analytic route; `TsAnalyticDifference`, `TsAnalytic / Ts - 1`; and `an`, `anExact` and `anWithinSupport`, the root of the analytic route and its diagnostics from `evaluateRoot()`. Each value is a conditional evaluation of the supplied layers and paired inputs; the controls are numerical diagnostics, not physical dispersion.

Mathematics

Dakoulas and Gazetas (1985) measure the depth z from the apex where the extended slopes meet; the crest lies at $z = h$, the base at $z = H$ and the truncation ratio is $\lambda = h/H$ (hydrated copy, Fig.

1 on printed page 167 and `eq-free-truncation-ratio`). The width of the dam is proportional to z, and the free lateral vibration obeys (`eq-free-motion`)

$$\rho \frac{\partial^2 u}{\partial t^2} = \frac{1}{z} \frac{\partial}{\partial z} \left[G(z)\, z\, \frac{\partial u}{\partial z} \right],$$

with zero displacement at the base and zero shear at the crest (`eq-free-base-condition` and `eq-free-crest-condition`): $u(H) = 0$ and $Gz\,\partial u/\partial z = 0$ at $z = h$. For the slope of `buildGeometry()`, of height H_s from crest to base and truncation ratio λ_o, the extended slopes meet $h = \lambda_o H_s/(1-\lambda_o)$ above the crest and $H = H_s/(1-\lambda_o)$ (see `evaluateBeamPeriod()`). For harmonic motion $u = U(z)e^{i\omega t}$ (`eq-free-harmonic`), multiplying by zv with $v(H) = 0$ and integrating by parts, the crest term vanishes by the free condition and

$$\int_h^H G\, z\, U' v'\, dz = \omega^2 \int_h^H \rho\, z\, U v\, dz,$$

a regular Sturm-Liouville problem with weight z whose smallest eigenvalue gives $T_s = 2\pi/\omega_1$. The layers of the realization are stacked from the crest, $z = h$ plus the depth below it, in the order of their midpoint depths, with $G_k = \rho_k V_k^2$ and $\rho_k = \gamma_k/g$ constant in each; their thicknesses must add up to H_s.

With $\lambda_o = 0$ the crest is the apex, where the width and the crest term vanish with z itself. The problem stays well posed in the energy space: the stiffness and mass of the first element remain positive, and the branch Y_q of the general solution (`eq-free-solution`) has infinite energy $\int GzU'^2 dz$ at $z = 0$, so the Galerkin solution converges to the regular branch, the paper's untruncated case. The mesh starts at the apex, with no offset; the package tests reproduce the untruncated periods of Table 7 of the paper (printed page 172).

Analytic route

`TsAnalytic` is the closed form of the paper (`eq-natural-frequency-period`) evaluated by `evaluateBeamPeriod()` with the apex-to-base height $H_s/(1-\lambda_o)$, the basal velocity `VSo` and the first root of the characteristic relation at `mo` and `lo` from `evaluateRoot()` with the solution method `method`. It is exact for the power law $G = G_b(z/H)^m$ measured from the apex; with the exponent that `characterizeProfile()` fits from the crest it is an approximation. `TsAnalyticDifference` (`TsAnalytic / Ts - 1`) is returned so that a consumer can flag a divergence under a threshold of its own; no threshold is applied here. The package tests reduce the numerical period to the closed form on layered power-law profiles with m = 0, 0.5, 0.95 and 1.5 and λ = 0.075, 0.2, 0.351 and 0.5, and reproduce the shear-beam periods of the five dams of Table 8 of the paper (printed page 175) within 0.2 percent.

Numerical method

The weak form is discretised with continuous piecewise-linear functions on a mesh that follows the layers: every layer is divided into `elements` equal elements, so every interface is a node, and the node at the bottom of the last layer is fixed. With a width $w(z)$, 1 for the column and the distance to the apex for the wedge of `solveWedgePeriod()`, the integrals of an element of length L between an upper node a and a lower node b are exact, because G and ρ are constant within a layer and w is linear: the element stiffness is

$$k_e = G \frac{w_a + w_b}{2L},$$

assembled as $k_e(\delta_a-\delta_b)(\delta_a-\delta_b)^T$, and the consistent mass has diagonal entries $\rho L(3w_a+w_b)/12$ and $\rho L(w_a+3w_b)/12$ and coupling $\rho L(w_a+w_b)/12$. The generalized eigenvalue problem $K\phi = \omega^2 M\phi$ has K and M symmetric positive definite for positive G and ρ with the base fixed. The stiffness factors as $K = U^TU$ with $U = D^{1/2}C$, where D is the diagonal of the k_e and C the bidiagonal difference matrix (1 on the diagonal, -1 above it: element e joins nodes e and $e+1$, the base node removed); U is the Cholesky factor of K, and its inverse is explicit because C^{-1} is the upper triangle of ones. The Cholesky reduction gives the symmetric matrix

$$B = U^{-T}MU^{-1} = D^{-1/2}SD^{-1/2}, \qquad S_{ij} = c_{\min(i,j)}\ (i \neq j), \qquad S_{ii} = c_i - M_{i,i+1},$$

with c_i the consistent mass from the top to node i (the cumulative row sums of M). The eigenvalues of B are $1/\omega^2$, so the fundamental mode is its largest eigenvalue μ_1, computed by the direct symmetric solver `eigen(symmetric = TRUE)`, and $T = 2\pi\sqrt{\mu_1}$; no iterative solver or convergence loop is involved. Every entry of S is a sum of positive masses, and the largest eigenvalue of a symmetric matrix is resolved to relative machine precision.

The discretisation is a Rayleigh-Ritz method: the numerical period never exceeds the exact period of the layered beam, and it does not decrease when every element is halved, since the trial spaces are nested. The period returned is the one with `2 * elements` elements per layer, and the mesh change $1 - T_e/T_{2e} \geq 0$ between the declared and the doubled mesh is its convergence certificate. Within a layer the solution is smooth and the interfaces are nodes, so the error decreases with the square of the element length and is asymptotically a third of the mesh change. A profile whose stiffness tends to zero at the surface converges more slowly. No threshold is applied to the mesh change: it is returned for the consumer.

Examples

```
Layers <- data.frame(hs = rep(1, 10), zm = seq(0.5, 9.5, by = 1), gsat = 19, VSm = 200)
solveWedgePeriod(Layers, Hs = 10, lo = 0.35, mo = 0, VSo = 200, method = "exact", elements = 4)
```

`summarizeDisplacements`

Marginal statistics of displacement realizations by model

Description

Reduces the realizations of `evaluateDisplacements()` to the requested quantiles and mean of the displacement of each model at each yield acceleration. The statistics are marginal: a quantile row of one yield acceleration and the row of the next do not belong to the same realization. Keep the realizations when a later calculation needs paired values.

Usage

```
summarizeDisplacements(draws, levels)
```

Arguments

`draws`	A table with `IDn`, `ky` and finite non-negative `Dn` (cm), one row per realization, as `evaluateDisplacements()` returns.
`levels`	Probabilities in (0, 1) and/or the literal `"mean"`, in the order the rows are wanted.

Details

The population of each row is the n realizations given for that model and yield acceleration. Realizations of the flexible models of evaluateDisplacements() are conditional on a nonzero displacement (evaluateFlexibleDisplacement()), so their quantiles and mean describe the nonzero displacement and do not include the probability of zero or negligible displacement of the published models.

Value

A data.table with IDn, ky (g), level (the probability as text, or "mean"), n (realizations summarized) and Dn (cm), one row per model, yield acceleration and level, the models and yield accelerations in their order of appearance in draws.

Mathematics

For the N realizations $D_{n,s}$ of one model at one k_y, the quantile of probability q is type 7 of stats::quantile() (linear interpolation between the order statistics $\lfloor h \rfloor$ and $\lceil h \rceil$, $h = 1 + (N - 1)q$) and the mean is $\frac{1}{N}\sum_s D_{n,s}$.

Examples

```
Draws <- data.frame(IDn = "AM88", ky = 0.1, Dn = c(0.5, 1, 2, 4))
summarizeDisplacements(Draws, levels = c(0.16, 0.5, 0.84, "mean"))
```

---

summarizeEnsemble *Weighted statistics of a displacement model ensemble*

Description

Pools the realizations of the models named in weights at each yield acceleration and reduces them to weighted quantiles and a weighted mean, the ensemble displacement of those models. Every realization of a model carries the normalized weight of its model. The statistics are marginal over yield acceleration, as in summarizeDisplacements().

Usage

```
summarizeEnsemble(draws, weights, levels)
```

Arguments

draws	A table with IDn, ky and finite non-negative Dn (cm), as evaluateDisplacements() returns; rows of models not named in weights are ignored.
weights	Named positive finite weights of the ensemble models; only their ratios matter.
levels	Probabilities in (0, 1) and/or the literal "mean".

Details

The population of each row is the n pooled realizations at that yield acceleration. The pool inherits the conditioning of its members: the realizations of the flexible models of evaluateDisplacements() are conditional on a nonzero displacement (evaluateFlexibleDisplacement()), so the probability of zero or negligible displacement of those published models is not represented in the ensemble quantiles and mean.

Value

A data.table with ky (g), level, n (pooled realizations) and Dn (cm), one row per yield acceleration and level, the yield accelerations in their order of appearance.

Mathematics

With $\omega_m = w_m / \sum_j w_j$ over the named models, the pooled sample at one k_y holds $D_{n,s}^{(m)}$ with weight ω_m. The weighted mean is $\sum \omega_m D_{n,s}^{(m)} / \sum \omega_m$ and the weighted quantile of probability q is the seventh sample quantile definition of Hyndman and Fan (1996), the type 7 of quantile(), extended to cumulative weights: with the distinct sorted values x_i, their summed weights and the total weight W, the order statistics of rank $\lfloor h \rfloor$ and $\lfloor h \rfloor + 1$, $h = 1 + (W - 1)q$, are the smallest x_i whose cumulative weight reaches each rank, interpolated linearly by the fractional part of h. With equal model weights and N realizations per model, $W = N$. Both statistics are computed here without Hmisc and at full precision: equal values are merged exactly, with the estimator of summarizeKmax(). When the pooled sample holds equal values, Hmisc merges values by their 15-significant-digit text, so every pooled value is then read back from that text; the compatibility route of buildDnTables() keeps that text merge, as newmark 1.7.1 did, and this general summary does not (design decision).

References

Hyndman, R. J. and Fan, Y. (1996). Sample quantiles in statistical packages. The American Statistician, 50(4), 361-365. doi:10.1080/00031305.1996.10473566

Examples

```
Draws <- data.frame(IDn = rep(c("AM88", "BT07"), each = 4), ky = 0.1,
                    Dn = c(0.5, 1, 2, 4, 1, 2, 3, 5))
summarizeEnsemble(Draws, weights = c(AM88 = 1, BT07 = 1), levels = c(0.5, "mean"))
```

---

summarizeKmax *Weighted mean and quantiles of pooled yield coefficients*

Description

Pools the yield-coefficient realizations of one or more displacement models with their epistemic weights and reduces the pool, at every admissible displacement, to its weighted mean, its weighted quantiles and the weighted share of roots that lie beyond the yield-coefficient grid.

Usage

```
summarizeKmax(roots, weights, levels)
```

Arguments

`roots`	The roots of `solveKmax()` for one or more models, bound by rows: `IDn`, `realization`, `Da`, `kmax` and `beyond`, one row per model, realization and `Da`; every pooled model must hold the same realizations at every `Da`.
`weights`	Named non-negative weights by model; the models with positive weight form the pool and must all be present in `roots` with the same realizations.
`levels`	Unique probabilities in (0, 1), distinct at two decimals.

Value

A data.table with one row per `Da` and level: `Da` (cm), `level` (`"mean"`, then each probability formatted `"%.2f"`), `kmax` (g), the weighted mean or quantile of the pooled realizations at that `Da`, `beyond`, the weighted share of the pool beyond the grid, and `n`, the number of pooled roots (models times realizations), repeated on every level of one `Da`.

Mathematics

The positive weights W_m of the pooled models are normalized, $w_m = W_m / \sum_l W_l$. Every model contributes the same realizations $s = 1, \ldots, N_s$, so at one D_a the pool holds the $N_m N_s$ roots $k_{m,s}$, each weighted by its model's w_m. The mean is

$$\bar{k}_{max} = \frac{\sum_{m,s} w_m k_{m,s}}{\sum_{m,s} w_m}.$$

For the quantile of level p, the pooled values are sorted and equal values merged with their summed weights into $x_1 < \cdots < x_K$ with cumulative weights $C_1 < \cdots < C_K = W$; with $h = 1 + (W - 1)p$, $a = \max(\lfloor h \rfloor, 1)$ and $b = \min(a + 1, W)$, the quantile is $(1 - f)\, x(a) + f\, x(b)$, $f = h - \lfloor h \rfloor$, where $x(c)$ is the smallest x_k with $C_k \geq c$ (or x_K when none). This is the seventh sample quantile definition of Hyndman and Fan (1996) extended to cumulative weights, the estimator `newmark` 1.7.1 obtained from `Hmisc::wtd.quantile()`; for one model of unit weight it is the type-7 sample quantile of its N_s roots. The share beyond the grid is $\sum_m w_m \bar{b}_m$, with $\bar{b}_m$ the fraction of the realizations of model m whose root lies beyond the grid.

A root of 0 (a target above the ceiling of its realization, see `solveKmax()`) stays in the pool as a realization of null demand.

References

Hyndman, R. J. and Fan, Y. (1996). Sample quantiles in statistical packages. The American Statistician, 50(4), 361-365. doi:10.1080/00031305.1996.10473566

Examples

```
ky <- c(0.05, 0.10, 0.20, 0.30, 0.40)
Roots <- rbind(
  solveKmax(ky, matrix(0.10 / ky, 20, 5, byrow = TRUE), PGA = NULL, model = "AM88", Da = 1),
```

```
  solveKmax(ky, matrix(0.30 / ky, 20, 5, byrow = TRUE), PGA = NULL, model = "BT07", Da = 1))
summarizeKmax(Roots, weights = c(AM88 = 0.5, BT07 = 0.5), levels = c(0.05, 0.50, 0.95))
```

`summarizeMixture` *Quantiles and mean of a spectral mixture*

Description

Reduces a spectrum represented as a finite lognormal mixture at every oscillator period, such as the rock spectra of the hazard products or the result of `integrateSiteMixture()`, to the spectral accelerations at the requested probabilities and to the mean, period by period.

Usage

```
summarizeMixture(mixture, levels)
```

Arguments

`mixture`	A mixture as `integrateSiteMixture()` takes and returns it, validated by the same rules: one row per `(Tn, k)` with columns `Tn` (s), `k` (component label), `muLn`, `sigmaLn` (finite, `sigmaLn` non-negative) and `w` (positive weights summing to one at every period), including the PGA rows (`Tn == 0`).
`levels`	Probabilities in $(0, 1)$.

Value

A data.table ordered by `Tn` with `Tn`, `level` (the probability as text, then `"mean"`) and `Sa` (g), one row per period and level, unrounded.

Mathematics

At each period the weights are normalized, $\tilde{w}_k = w_k / \sum w$, and the mixture distribution function is $F(x) = \sum_k \tilde{w}_k \Phi((x - \mu_k)/\sigma_k)$ for $x = \ln S_a$, a component with $\sigma_k = 0$ contributing the step $1\{x \geq \mu_k\}$. The quantile at probability p is $\exp(x_p)$ with $x_p = \inf\{x : F(x) \geq p\}$, the smallest value whose distribution function reaches p (generalized inverse). When every dispersion is positive, F is continuous and strictly increasing and x_p is the root of $F(x) = p$, found by `stats::uniroot()` to 10^{-12} on $[\min(\mu - 12s), \max(\mu + 12s)]$, $s = \max(\sigma, 1)$; otherwise F jumps at the atoms, $F(x) = p$ may have no root, and x_p is the upper end of 100 bisections of that interval, each keeping $F \geq p$ at the upper end, so that a probability inside a jump returns the atom. The mean is $\sum_k \tilde{w}_k e^{\mu_k + \sigma_k^2/2}$. These are marginal statistics of the ordinate at each period; they do not describe one spectrum across periods.

Examples

```
Mixture <- data.table::CJ(Tn = c(0, 0.2, 1), k = c("a", "b"))
Mixture[, `:=`(muLn = log(0.4) + 0.1 * (k == "b") - 0.5 * Tn, sigmaLn = 0.5, w = 0.5)]
summarizeMixture(Mixture, levels = c(0.16, 0.50, 0.84))
```

summarizeProfiles *Marginal statistics of a profile population*

Description

Reduces the realizations of `simulateSoilProfiles()`, of `propagateSitePeriod()`, of `propagateBeamPeriod()` or of any table with one row per realization to the requested quantiles and mean of the named numeric properties. The statistics are marginal: a quantile row does not describe one column, and the pairing between properties is lost in this reduction. Keep the realization table when a later calculation needs paired values.

Usage

```
summarizeProfiles(profiles, levels, properties)
```

Arguments

`profiles`	A table with one row per realization.
`levels`	Probabilities in (0, 1) and/or the literal `"mean"`, in the order the rows are wanted.
`properties`	Names of the numeric columns of `profiles` to summarize, for example `c("Go", "mo", "VSo")`, `"Tr"` or `"Ts"`.

Value

A data.table with one row per requested level: `level` (the probability as text, or `"mean"`), `n` (realizations summarized) and the requested properties. Quantiles use type 7 of `stats::quantile()` on the stored values; the mean is arithmetic.

Examples

```
set.seed(20260915)
Profiles <- simulateSoilProfiles(Hs = 10, USCS = c("ML", "MH"), NS = 5, h = 1,
                                 water = 0, POP = 100, Vref = 760)
summarizeProfiles(Profiles, levels = c(0.16, "mean", 0.84), properties = c("Go", "mo", "VSo"))
```

VancouverSpectra *Uniform hazard spectra of Vancouver on rock from the 2020 National Building Code of Canada*

Description

Spectral accelerations of the 2020 National Building Code of Canada seismic hazard model for Vancouver (latitude 49.2827, longitude -123.1207) on rock, Vs30 760 m/s, as published by the seismic hazard tool of Natural Resources Canada. Each row gives the acceleration `Sa` at the oscillator period `Tn` whose annual frequency of exceedance is `1 / TR`; `Tn = 0` holds the peak ground acceleration. The ten return periods correspond to probabilities of exceedance in 50 years between 2 and 40 percent. Values retrieved on 2026-10-24 through the tool's public interface; they are reproduced under the Open Government Licence - Canada.

Usage

```
VancouverSpectra
```

Format

A data.table with 100 rows and 3 variables: `TR` (return period, years), `Tn` (oscillator period, s), `Sa` (spectral acceleration, g).

Source

Natural Resources Canada, 2020 National Building Code of Canada Seismic Hazard Tool, https://www.earthquakescanada.nrcan.gc.ca/hazard-alea/interpolat/nbc2020-cnb2020-en.php

Index

∗ **datasets**
CylinderRoots, 24
ShearModelParameters, 68
SiteClass, 71
VancouverSpectra, 84

assembleDnPOETables, 3
assembleDnPOETables(), *56*
assembleDnTables, 4
assembleDnTables(), *55*
assembleKmaxTables, 5
assembleKmaxTables(), *55*

buildDnPOETable, 6
buildDnPOETable(), *3*, *56*
buildDnTables, 7
buildDnTables(), *4*, *50*, *53*, *55*, *81*
buildGeometry, 10
buildGeometry(), *18, 19*, *30*, *54*, *57*, *77, 78*
buildKmaxTable, 11
buildKmaxTable(), *5*, *50*, *53*, *55*
buildKyGrid, 12
buildKyGrid(), *9*, *13*, *60*
buildKyLimits, 13
buildKyLimits(), *43*, *60*
buildPGATable, 14
buildPGATable(), *7*, *55*
buildQuantileSpline, 15
buildQuantileSpline(), *67*
buildRootModel, 16
buildRootModel(), *18, 19*, *24*, *36*, *57*, *76, 77*
buildShearDraws, 17
buildShearDraws(), *20*, *54*
buildShearTable, 19
buildShearTable(), *11*, *14*, *17, 18*, *21*, *41*, *53, 54*, *56*

characterizeProfile, 20
characterizeProfile(), *20*, *31*, *57*, *59*, *70, 71*, *77, 78*
classifySite, 21
convertAEP, 22
convertAEP(), *61*
correlateResiduals, 23
correlateResiduals(), *47, 48*, *64*, *66*, *70*
CylinderRoots, *16, 17*, 24, *36*

data.table::fread(), *51*, *55*
data.table::fwrite(), *55*
describeModule, 24

estimateAI, 25
estimateAI(), *35*, *43*, *64*
estimatePGV, 26
estimatePGV(), *33*, *64*
estimateRayleighPeriod, 27
estimateRayleighPeriod(), *21*, *30*, *74*
estimateVS30, 28
estimateVS30(), *21*, *71*
estimateWeightedRayleighPeriod, 29
estimateWeightedRayleighPeriod(), *58*, *73*
evaluateBeamPeriod, 30
evaluateBeamPeriod(), *18, 19*, *21*, *57*, *78*
evaluateDisplacements, 31
evaluateDisplacements(), *7*, *79–81*
evaluateFlexibleDisplacement, 32
evaluateFlexibleDisplacement(), *9*, *26*, *31, 32*, *43*, *50*, *75*, *80, 81*
evaluateRigidDisplacement, 35
evaluateRigidDisplacement(), *25*, *32*, *43*, *50*, *75*
evaluateRoot, 36
evaluateRoot(), *16, 17*, *57*, *77, 78*
evaluateSiteFactor, 37
evaluateSiteFactor(), *39*, *46–48*, *62, 63*, *69*
evaluateSiteSa, 39
evaluateSiteSa(), *37*, *48*
evaluateSoilLayers, 40

evaluateSoilLayers(), *20*, *64*, *68*, *70–72*
evaluateUniformHazard, 41

identical(), *54*
integrateDnHazard, 42
integrateDnHazard(), *44–46*
integrateDnHazardProfiles, 44
integrateSiteMixture, 46
integrateSiteMixture(), *37*, *61, 62*, *83*
integrateSiteSpectrum, 48
integrateSiteSpectrum(), *37*, *62, 63*, *69, 70*
interpolateSpectrum, 49
interpolateSpectrum(), *64*

listDisplacementFamilies, 50
listProjectTables, 51
listProjectTables(), *59*

message(), *52*

processProject, 52
processProject(), *46*, *51*, *59*
propagateBeamPeriod, 56
propagateBeamPeriod(), *20*, *31*, *58*, *70*, *84*
propagateSitePeriod, 58
propagateSitePeriod(), *57*, *70*, *84*

queryProjectTable, 59
queryProjectTable(), *50*

readHazardWorkbook, 60
readRDS(), *51*
refineSiteMixture, 61
refineSiteMixture(), *15*, *46, 47*
refineSiteSpectrum, 62
refineSiteSpectrum(), *15*, *48*

sampleIntensities, 63
sampleIntensities(), *7*, *31, 32*
sampleSoilLayers, 64
sampleSoilLayers(), *40*, *53*, *57*, *59*, *70*
sampleSpectra, 66
sampleSpectra(), *15*, *23*, *64*
saveRDS(), *5*, *11*, *54, 55*
selectDisplacementModels, 67
selectDisplacementModels(), *7, 8*, *54, 55*
set.seed(), *54*
ShearModelParameters, *40*, 68
simulateSiteSpectrum, 69
simulateSiteSpectrum(), *37*, *48*
simulateSoilProfiles, 70
simulateSoilProfiles(), *56–58*, *72*, *76*, *84*
SiteClass, 71
solveColumnPeriod, 72
solveColumnPeriod(), *58, 59*, *76, 77*
solveKmax, 74
solveKmax(), *11*, *82*
solveProfilePeriods, 76
solveProfilePeriods(), *72*, *77*
solveWedgePeriod, 77
solveWedgePeriod(), *56, 57*, *72, 73*, *76*, *78*
stats::quantile(), *80*, *84*
stats::uniroot(), *17*, *83*
summarizeDisplacements, 79
summarizeDisplacements(), *7*, *32*, *80*
summarizeEnsemble, 80
summarizeEnsemble(), *7*, *9*, *32*
summarizeKmax, 81
summarizeKmax(), *11, 12*, *74*, *81*
summarizeMixture, 83
summarizeMixture(), *46*, *62*
summarizeProfiles, 84
summarizeProfiles(), *20*, *56*, *58*, *69, 70*

VancouverSpectra, 84