Package 'ShedsHT'

August 26, 2019

Title The SHEDS-HT model for estimating human exposure to chemicals.

Version 0.1.8

Author Kristin Isaacs [aut, cre]

Maintainer Kristin Isaacs <isaacs.kristin@epa.gov>

Description The ShedsHT R package runs the Stochastic Human Exposure and Dose Simulation-High Throughput screening model which estmates human exposure to a wide range of chemicals.The people in SHEDS-HT are simulated individuals who collectively form a representative sample of the target population, as chosen by the user. The model is cross-sectional, with just one simulated day (24 hours) for each simulated person, although the selected day is not necessarily the same from one person to another. SHEDS-HT is stochastic, which means that many inputs are sampled randomly from user-specified distributions that are intended to capture variability. In the SHEDS series of models, variability and uncertainty are typically handled by a two-stage Monte Carlo process, but SHEDS-HT currently has a single stage and does not directly estimate uncertainty.

License MIT

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Imports data.table, ggplot2, stringr, plyr

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

R topics documented:

act.diary.pools *act.diary.pools*

Description

Assigns activity diaries from the Activity_diaries input on the Run file (read through the read.act.diaries function) to pools based on age, gender, and season

Usage

```
act.diary.pools(diaries, specs)
```

Arguments

diaries	A data set created internally in SHEDS.HT through the read.act.diaries function. The data are activity diaries, which indicate the amount of time and level of metabolic activity in various micro environments. Each line of data represents one person-day (24 hours).
specs	Output of the read.run.file function, which can be modified by the update.specs function before input into act.diary.pools.

Details

The act.diary.pools function assigns the activity diaries to pools. A given diary may belong to many pools, since every year of age has its own pool. Large pools may contain a list of several hundred diary numbers. The code contains four loops and on each step performs a sub-setting of the list of diary numbers.

Value

pool A vector of lists. The length of pool is the product of the genders, seasons, and ages inputs specified in the Run file. Each element is a list of acceptable activity diary numbers for each year of age, gender, weekend, and season combination. For example, pool[100] may have the name M0P99, which indicates that it is for males, on weekdays, in spring,for age=99. In addition, If the function runs successfully, the following message will be printed: "Activity Diary Pooling completed"

Note

The input to act.diary.pools, diaries, is created by reading in the Activity_diaries input file (specified on the Run file) with the read.act.diaries function within the run function.

Author(s)

Kristin Isaacs, Graham Glen

See Also

run

add.factors	*add.factors Adds specific exposure factors to the* pdm *data table, which is output from the* add.media *function. Here, "specific" means taking into account the age, gender, season, and exposure media for each person.*

Description

add.factors Adds specific exposure factors to the pdm data table, which is output from the add.media function. Here, "specific" means taking into account the age, gender, season, and exposure media for each person.

Usage

```
add.factors(n, gen.f, med.f, exp.f, surf, pdm)
```

Arguments

n	Number of persons
gen.f	Non-media specific exposure factors as a data table. Output from the gen.factor.tables function.
med.f	Media specific exposure factors presented as a data table. Output from the med.factor.tables function.
exp.f	Distributional parameters for the exposure factors. Output of the exp.factors function.
surf	A list of surface media. Modified output of the read.media.file function.
pdm	A data table containing the pd data frame of physiological and demographic parameters for each theoretical person, and the dur array, which specifies the duration of exposure to each potential exposure medium for each person in pd. Output of the add.media function.

Details

The process of adding specific exposure factors to pdm involves multiple steps. First, w is determined, which is the number of general factors plus the product of the number of media-specific factors and the number of surface media. Air media do not have media specific factors in this version of SHEDS. An array, q, of uniform random samples is generated, with one row per person and w columns. A zero matrix, r, of the same size is defined. Once these matrices are defined, the media-specific factors are determined. Two nested loops over variable and surface type generate the values, which are stored in r. Next, another FOR loop determines the general factors. The p data set contains the age, gender, and season for each person. These two data sets are then merged. The evaluation of these factors is handled by the eval.factors function. One of the exposure factors is handwash.freq. This was also part of SHEDS-Multimedia, where it represented the mean number of hours in the day with hand washing events. An important aspect of that model was that because each person was followed longitudinally, the actual number of hand washes on each day varied from one day to the next. Because of this, the distribution for handwash.freq did not need to be restricted to integer values, as (for example) a mean of 4.5 per day is acceptable and achievable, while choosing integer numbers of hand washes each day. One of the early goals with SHEDS.HT was to attempt to reproduce selected results from SHEDS-Multimedia.

Therefore, similar logic was built into the current model. The hand.washes variable is sampled from a distribution centered on handwash.freq, and then rounded to the nearest integer. The bath variable is another difficult concept. In theory, baths and showers are recorded on the activity diaries. In practice, the activity diaries were constructed from approximately 20 separate studies, some of which did not contain enough detail to identify separate bath or shower events. The result is that about half of all diaries record such events, but the true rate in the population is higher. The bath.p variable was created to address this. It represents the probability that a non bath/shower activity diary should actually have one. Therefore, if the diary has one, then SHEDS automatically has one. Otherwise, a binomial sample using bath.p as the probability is drawn. A bath/shower occurs unless both of these are zero. The effectiveness of hand washes or bath/shower at removing chemical from the skin is determined in the post.exposure function.

Value

pdmf A data set containing the pdm data table as well as media specific exposure factors, the number of baths taken, and the number of hand wash events occurring per day per person contained in pdm.

Author(s)

Kristin Isaacs, Graham Glen

See Also

eval.factors, post.exposure

---

add.fugs *add.fugs*

Description

Evaluates the variables in the fugacity input file and creates values for those variables corresponding to each simulated person.

Usage

```
add.fugs(n.per, x, pdmf)
```

Arguments

n.per	The total number of simulated persons in this model run specified in the Run file
x	the output of the read.fug.inputs function.
pdmf	he output of the add.factors function. A data set containing physiological and demographic parameters for each theoretical person, the duration of exposure to each potential exposure medium for each person, the media specific exposure factors, and the number of baths taken and hand wash events occurring per day for each person.

Details

This function evaluates the variables in the fugacity input file, creating one value for each simulated person and adding each variable as a new column in the pdmf data set (renamed as pdmff). All variables are left i n their original units except for those with units ug/cm2, which are converted to ug/m2. In the fugacity calculations, all masses are in ug and all lengths are in m.

Value

pdmff Output contains values sampled from the distributions of each relevant variable in the `Fugacity` input file for each theoretical person.

Author(s)

Kristin Isaacs, Graham Glen

See Also

`add.factors`, `add.media`, `select_people`, `Fugacity`, `run`

`add.media` *add.media*

Description

For each theoretical person parameterized in the `pd` data frame, which is output from the `select.people` function, this function generates the exposure duration for each potential exposure medium.

Usage

```
add.media(n, media, pd)
```

Arguments

`n`	Number of persons.
`media`	List of the potential contact media for the model; output of the `read.media.file` function. Each is found in a specific microenvironment (micro).
`pd`	Data frame containing internally assigned demographic and physiological parameters for each theoretical person modeled in SHEDS.HT. Output of the `select.people` function

Details

To generate the exposure duration for each theoretical person, an array q of uniform random samples is generated with rows = `n` (per set, where set size is specified by the user in the `Run` input file) and columns = `nrow(media)` (the number of potential exposure media). An array called `dur` is created for the number of minutes on the activity diary in the relevant micro multiplied by the relevant probability of contact (as specified in the `media` input). The array consists of a row for each person and a column for each of the exposure media. The output is a data table containing the `pd` data frame and the `dur` array.

Value

pdm A data table containing the `pd` data frame of physiological and demographic parameters for each theoretical person, and the `dur` array, which specifies the duration of exposure to each potential exposure medium for each person in `pd`.

Author(s)

Kristin Isaacs, Graham Glen

See Also

`select.people`, `read.media.file`

`check.foods` *check.foods This function creates a list of unique food types.*

Description

check.foods

This function creates a list of unique food types.

Usage

```
check.foods(s)
```

`check.src.scen.flags` *check.src.scen.flags*

Description

This function checks whether the settings on the source.scen object (input as "df") are set to numeric values of 1 or 0. If the column for a particular exposure scenario is missing from "df", it is created and assigned "0" for all sources.

Usage

```
check.src.scen.flags(df)
```

`check.src.scen.types` *Check.src.scen.types*

Description

This function checks whether the settings on the source.scen object (input as "df") are consistent with the source.type for each source.

Usage

```
check.src.scen.types(df)
```

chem.fug *chem.fug*

Description

Creates distributions of chemical specific parameters for each chemical of interest, in order to reflect real-world variability and uncertainty. These distributions are then sampled to create chemical specific parameters associated with the exposure of each simulated person.

Usage

```
chem.fug(n.per, cprops, x)
```

Arguments

n.per	The total number of simulated persons in this model run specified in the Run file.
cprops	The chemical properties required for SHEDS-HT (output of chem.props function). The default data was prepared from publicly available databases using a custom program (not part of SHEDS-HT). The default file contains about 17 numerical inputs per chemical, but most are not used. The required properties are molecular weight (MW), vapor pressure (VP.Pa), solubility (water.sol.mg.l), octanol-water partition coefficient (log.Kow), air decay rate (half.air.hr), decay rate on surfaces (half.sediment.hr), and permeability coefficient (Kp).
x	From the output of the read.fug.inputs function.

Details

This function obtains chemical-specific properties from the chem.data data set.That data set contains point value for each variable, which this function defines distributions around, with the exception of molecular weight. The constructed distributions reflect both uncertainty and variability (for example, vapor pressure will vary with temperature, humidity, and air pressure or altitude, and may depend on the product formulation). For each variable, a random sample is generated for each simulated person, and the set of variables becomes the output data set. Input surface loading variables are in units of ug/cm2 or ug/cm2/day and are converted by the function to meters, to avoid the need for conversion factors in later equations. These conversions are done after the random sampling, as otherwise the correct conversions depend on the distributional form (for example, par2 is changed for the normal, but not for the lognormal).

Value

samples A data set with the chemical specific parameters for each combination of chemical and simulated person. For each chemical, the chemical specific parameters assigned to a given person are randomly sampled from distributions on those parameters. These distributions are created from point estimates to reflect real-world uncertainty and variability.

Author(s)

Kristin Isaacs, Graham Glen

See Also

Fugacity, Run, run, Chem_props, get.fug.concs

---

chem.scenarios *chem.scenarios*

Description

This function summarizes the all.scenarios data set by condensing each chemical-scenario combination into one row. The number of rows of data for this combination on all.scenarios is recorded here.

Usage

```
chem.scenarios(all)
```

Arguments

all	This is the master list of all chemicals, and all exposure scenarios specific to each chemical, to be evaluated in the current model run. This data set is compiled internally according to the user's specifications on the Run_85687 file. The user may create multiple scenarios files for special purposes, for example, for selected chemical classes, or selected exposure pathways. A model run consists of two nested loops: the outer loop over chemicals and the inner loop over the scenarios specific to that chemical.

Value

chem.scen	A data set consisting of all chemicals of interest, and the categories and scenarios for that chemical with the potential for exposure. The relationship between chemicals and scenarios is one to many. It is possible for one chemical to have two or more "dermal" scenarios, provided each is in a different "category". However, each combination of chemical, category, and scenario must be unique. The last variable on chem.scen is "count", which is the number of rows of data on all.scenarios devoted to this combination.

Author(s)

Kristin Isaacs, Graham Glen

See Also

Run, Source_chem_foods, Source_chem_prods, Source_scen_food, Source_scen_prods

---

`combine_output` *combine_output*

Description

combine_output is a post-processing tool that extracts selected statistics (those specified in the 'metrics' argument) from the 'allstats' file for each chemical in a single model run, and combines them onto out.file.

Usage

```
combine_output(run.name = specs$run.name, out.file = "SHEDSOutFCS.csv",
  metrics = c("5%", "50%", "75%", "95%", "99%", "mean", "sd"))
```

Arguments

`run.name`	default = name of last run (in current R session)
`out.file`	default="SHEDSOutFCS.csv"
`metrics`	the summary statsistics to be pulled. default = c("5%", "50%", "75%", "95%", "99%","mean","sd")

Details

In a multichemical run, SHEDS creates separate output files for each chemical. If the analyst wants to compare exposure or dose metrics across chemicals, use combine_output to put the relevant information for all chemicals together on one file.

Value

finaldata A R object that has "exp.dermal", "exp.ingest", m"exp.inhal", "dose.inhal", "dose.intake", "abs.dermal.ug", "abs.ingest.ug", "abs.inhal.ug", "abs.tot.ug", "abs.tot.mgkg", "ddd.mass" for the chemcial of question. This is tablated for the "5%", "50%", "75%", "95%", "99%", quantile of exposure as well as the mean and sd "finaldata,paste0("output/",run.name,"/",out.file.csv") this is the CSV of finaldata

Author(s)

Kristin Isaacs, Graham Glen

`create.scen.factors` *create.scen.factors*

Description

This function takes the information on distributions from the `all.scenarios` data set (which comes from the `source_variables_12112015` input file) and converts it into the parameter set needed by SHEDS.HT.

Usage

```
create.scen.factors(f)
```

Arguments

`f`	An internally generated data set from the `Source_vars` input file (as specified in the `Run` file) containing data on the distribution of each source variable in SHEDS.HT.

Details

The steps involved in this function are 1) converting `prevalence` from a percentage to a binomial form, 2) converting `CV` to standard deviation for normals, and 3) converting `Mean` and `CV` to par1 and par2 for lognormals. Note that `prevalence` in SHEDS becomes a binomial distribution, which returns a value of either 0 or 1 when evaluated. Each simulated person either "does" or "does not" partake in this scenario. Similar logic applies to the `frequency` variable, except that the returned values may be larger than one (that is, 2 or more) for very frequent scenarios. All the exposure equations contain the `prevalence` variable. If `prevalence` is set to one for that person, the exposure is as expected, but if `prevalence`=0 then no exposure occurs.

Value

dt A data set with values and probabilities associated with each variable distribution presented in `f`. All variables in `f` are also retained.

Author(s)

Kristin Isaacs, Graham Glen

See Also

`Source_vars`, `Run`

`diet.diary.pools` *diet.diary.pools*

Description

Assigns activity diaries from the `Diet_diaries` input on the `Run` file (read through the `read.run.file` function) to pools based on age and gender.

Usage

```
diet.diary.pools(diaries, specs)
```

Arguments

`specs`	Output of the `read.run.file` function, which can be modified by the `update.specs` function before input into `diet.diary.pools`.
`diet.diaries`	A data set created internally in SHEDS.HT through the `read.diet.diaries` function. The data are daily diaries of dietary consumption by food group. Each line represents one person-day, with demographic variables followed by amounts (in grams/day) for a list of food types indicated by a short abbreviation on the header line.

Details

The diet.diary.pools function assigns the dietary diaries to pools. A given diary may belong to many pools, since every year of age has its own pool. Large pools may contain a list of several hundred diary numbers. The code contains four loops and on each step performs a sub-setting of the list of diary numbers.

Value

dpool A vector of lists. The length of dpool is the product of the gender and age inputs specified in the Run file. Each element is a list of acceptable diet diary numbers for each year of age and each gender combination. In addition, if the function runs successfully, the following message will be printed: "Dietary Diary Pooling completed"

Note

The input to diet.diary.pools, diet.diaries, is created by reading in the Diet_diaries input file (specified on the Run file) with the read.diet.diaries function within the run function.

Author(s)

Kristin Isaacs, Graham Glen

See Also

Diet_diaries, run, read.run.file, Run

---

dir.dermal *dir.dermal*

Description

Models the dermal exposure scenario for each theoretical person.

Usage

```
dir.dermal(sd, cd)
```

Arguments

sd	The chemical-scenario data specific to relevant combinations of chemical and scenario. Generated internally.
cd	The list of scenario-specific information for the chemicals being evaluated. Generated internally.

Details

The dermal exposure scenario is relatively straightforward. The function produces a prevalence value, which reflects the fraction of the population who use this scenario at all. It also produces a frequency value, which is the mean number of times per year this scenario occurs among that fraction of the population specified by prevalence. Since SHEDS operates on the basis of one random day, the frequency is divided by 365 and then passed to the `p.round` (probabilistic rounding) function, which rounds either up or down to the nearest integer. Very common events may happen more than once in a day. The function also produces a mass variable, which refers to the mass of the product in grams in a typical usage event. The composition is the percentage of that mass that is the chemical in question. The resid variable measures the fraction that is likely to remain on the skin when the usage event ends. The final output is the total dermal exposure for each chemical-individual combination in micrograms, which is the product of the above variables multiplied by a factor of 1E+06.

Value

dir.derm For each person, the calculated direct dermal exposure that occurs when a chemical-containing product is used. This does not include later contact with treated objects, which is indirect exposure.

Author(s)

Kristin Isaacs, Graham Glen

See Also

`run`, `p.round`

`dir.ingested` *dir.ingested*

Description

Models the ingestion exposure scenario for each theoretical person.

Usage

```
dir.ingested(sd, cd)
```

Arguments

`sd`	The chemical-scenario data specific to relevant combinations of chemical and scenario. Generated internally.
`cd`	The list of scenario-specific information for the chemicals being evaluated. Generated internally.

Details

This scenario is for accidental ingestion during product usage. Typical examples are toothpaste, mouthwash, lipstick or chap stick, and similar products used on the face or mouth. The function produces a `prevalence` value, which reflects the fraction of the population who use this scenario at all. It also produces a `frequency` value, which is the mean number of times per year this scenario occurs among that fraction of the population specified by prevalence. Since SHEDS operates on the basis of one random day, the frequency is divided by 365 and then passed to the `p.round` (probabilistic rounding) function, which rounds either up or down to the nearest integer. Very common events may happen more than once in a day. The function also produces a `mass` variable, which refers to the mass of the product in grams in a typical usage event. The `composition` is the percentage of that mass that is the chemical in question. The `ingested` variable represents the percentage of the mass applied that becomes ingested. Since these products are not intended to be swallowed, this should typically be quite small (under 5%). The final output is the total incidental ingested exposure for each chemical-individual combination in micrograms, which is the product of the above variables multiplied by a factor of 1E6.

Value

dir.ingest For each person, the calculated quantity of a given chemical incidentally ingested during or immediately after use of products such as toothpaste. Does not include exposure via food and drinking water.

Author(s)

Kristin Isaacs, Graham Glen

`dir.inhal.aer` *dir.inhal.aer*

Description

Models the inhalation exposure from the use of aerosol products for each theoretical person.

Usage

```
dir.inhal.aer(sd, cd, cb, io)
```

Arguments

`sd`	The chemical-scenario data specific to relevant combinations of chemical and scenario. Generated internally.
`cd`	The list of scenario-specific information for the chemicals being evaluated. Generated internally.
`cb`	A copy of the `base` data set output from the `make.cbase` function, with columns added for exposure variables.
`io`	A binary variable indicating whether the volume of the aerosol is used to approximate the affected volume.

Details

This scenario considers inhalation exposure from the use of aerosol products. Typical examples include hairspray and spray-on mosquito repellent. The function produces a `prevalence` value, which reflects the fraction of the population who use this scenario at all. It also produces a `frequency` value, which is the mean number of times per year this scenario occurs among that fraction of the population specified by prevalence. Since SHEDS operates on the basis of one random day, the `frequency` is divided by 365 and then passed to the `p.round` (probabilistic rounding) function, which rounds either up or down to the nearest integer. Very common events may happen more than once in a day. The function also produces a `mass` variable, which refers to the mass of the product in grams in a typical usage event. The `composition` is the percentage of that mass that is the chemical in question. `frac.aer` is the fraction of the product mass that becomes aerosolized, and the `volume` affected by the use is approximated to allow the calculation of a concentration or density. Defaults are set in the code if these variables are missing from the input file. Exposure for the inhalation pathway has units of micrograms per cubic meter, reflecting the average air concentration of the chemical. An `airconc` variable is defined using `mass`, `composition`, `frac.aer`, and `volume`. Since exposure depends on the time-averaged concentration, a duration is necessary. For example, if one spends five minutes in an aerosol cloud and the rest of the day in clean air, the daily exposure is the cloud concentration multiplied by 5/1440 (where 1440 is the number of minutes in a day). This function also calculates the inhaled dose, in units of micrograms per day. The dose equals the product of `exposure` (g/m3), basal ventilation rate, `bvr` (m3/day), the METS factor of 1.75 (typically people inhale air at an average of 1.75 times the basal rate to support common daily activities), and a conversion factor of 1E+06 from grams to micrograms.

Value

dir.inh.aer The calculated quantity of chemical inhalation from aerosols, like hairspray and similar products, that are directly injected into the air on or around each exposed person.

Author(s)

Kristin Isaacs, Graham Glen

keyword ~SHEDS

`dir.inhal.vap` *dir.inhal.vap*

Description

Models the inhalation exposure from the vapors of volatile chemicals for each theoretical person.

Usage

```
dir.inhal.vap(sd, cd, cprops, cb, io)
```

Arguments

sd	The chemical-scenario data specific to relevant combinations of chemical and scenario. Generated internally.
cd	The list of scenario-specific information for the chemicals being evaluated. Generated internally.
cprops	The chemical properties required for SHEDS-HT. The default file (the Chem_props file read in by the read.chem.props function and modified before input into the current function) was prepared from publicly available databases using a custom program (not part of SHEDS-HT). The default file contains 7 numerical inputs per chemical, and the required properties are molecular weight (MW), vapor pressure (VP.Pa), solubility (water.sol.mg.l), octanol-water partition coefficient (log.Kow), air decay rate (half.air.hr), decay rate on surfaces (half.sediment.hr), and permeability coefficient (Kp).
cb	A copy of the base data set output from the make.cbase function, with columns added for exposure variables.
io	A binary variable indicating whether the volume of the aerosol is used to approximate the affected volume.

Details

This scenario considers inhalation exposure from vapors (not aerosols). For example, painting will result in the inhalation of vapor, but it does not involve aerosols (unless it is spray paint). For this scenario, the vapor pressure and the molecular weight are relevant variables for determining exposure. These variables are included in the input to the cprops argument, which is drawn internally from the Chem_props file. The function produces a prevalence value, which reflects the fraction of the population who use this scenario at all. It also produces a frequency value, which is the mean number of times per year this scenario occurs among that fraction of the population specified by prevalence. Since SHEDS operates on the basis of one random day, the frequency is divided by 365 and then passed to the p.round (probabilistic rounding) function, which rounds either up or down to the nearest integer. Very common events may happen more than once in a day. The function also produces a mass variable, which refers to the mass of the product in grams in a typical usage event. The composition is the percentage of that mass that is the chemical in question. The evap variable is an effective evaporated mass, calculated using the mass, composition (converted from percent to a fraction), the vapor pressure as a surrogate for partial pressure, and duration of product use. The duration term is made unitless by dividing by 5 (minutes), which is an assumed time constant. The effective air concentation airconc is calculated as evap/volume. The value for airconce is capped by maxconc, which represents the point at which evaporation ceases. For chemicals used for a short duration, or with low vapor presssure, maxconc might not be reached before usage stops. Once airconc is established, the function also calculates the inhaled dose, in units of micrograms per day. The dose equals the product of exposure (g/m3), basal ventilation rate, bvr (m3/day), the METS factor of 1.75 (typically people inhale air at an average of 1.75 times the basal rate to support common daily activities), and a conversion factor of 1E6 from grams to micrograms.

Author(s)

Kristin Isaacs, Graham Glen

See Also

run, p.round, read.chem.props, Chem_props

`distrib` *distrib*

Description

produces random samples from distributuions

Usage

```
distrib(shape = "", par1 = NA, par2 = NA, par3 = NA, par4 = NA,
  lt = NA, ut = NA, resamp = "y", n = 1, q = NA, p = c(1),
  v = "")
```

Arguments

`shape`	Required with no default. The permited inputs are Bernoulli, binomial, beta, discrete, empirical, exponential, gamma, lognormal, normal, point, probability, triangle, uniform, and Weibull.
`par1`	optional value requared for some shapes. Defulat = none
`par2`	optional value requared for some shapes. Defulat = none
`par3`	optional value requared for some shapes. Defulat = none
`par4`	optional value requared for some shapes. Defulat = none
`resamp`	optional value requared for some shapes. Defulat = 'y'
`n`	Optional Default = 1
`q`	Optional Default = none
`p`	Optional Default = c(1)
`v`	Optional Default = none
`It`	optional value requared for some shapes. Defulat = none
`Ut`	optional value requared for some shapes. Defulat = none

Details

This process is central to SHEDS because it is a stochastic model. The "shape" is the essential argument, with others being required for certain shapes. For all shapes, one of "n" or "q" must be specified. If "n" is given, then Distrib returns a vector of "n" independent random samples from the specified distribution. If "q" is given, it must be a vector of numeric values, each between zero and one. These are interpreted as the quantiles of the distribution to be returned. When "q" is given, Distrib does not generate any random values, it just evaluates the requested quantiles. The empirical shape requires argument "v" as a list of possible values to be returned, each with equal probability. The other shapes require one or more of par1-par4 to be specified. See the SHEDS Technical Manual for more details on the meanings of par1-par4, which vary by shape. "Lower.trun" is the lower truncation point, meaning the smallest value that can be returned. Similarly, "upper.trun" is the largest value that may be returned. Not all distributions use lower.trun and/or upper.trun, but they should be specified for unbounded shapes like the Normal

distribution. "Resamp" is a flag to indicate the resolution for generated values outside the truncation limits. If resamp="yes" then effectively new values are generated until they are within the limits. If resamp="no", values outside the limits are moved to those limits. "P" is a list of probabilities that are used only with the "discrete" or "probability" shapes. The "p" values are essentially weights for a list of discrete values that may be returned. The "empirical" distribution also returns discrete values, but assigns them equal weights, so then "p" is not needed.

Value

A vector of "n" values from one distribution, where "n" is either the input argument (if given), or the length of the input vector "q".

Author(s)

Kristin Isaacs, Graham Glen

`down.the.drain.mass` *down.the.drain.mass*

Description

Models the quantity of chemical entering the waste water system on a per person-day basis.

Usage

```
down.the.drain.mass(sd, cd)
```

Arguments

`sd`	The chemical-scenario data specific to relevant combinations of chemical and scenario. Generated internally.
`cd`	The list of scenario-specific information for the chemicals being evaluated. Generated internally.

Details

This function models the simplest of all the current scenarios. It evaluates the amount of chemical entering the waste water system, on a per person-day basis. The "exposure" is to a system, not a person, but this method uses one person's actions to estimate their contribution to the total. The function produces a `prevalence` value, which reflects the fraction of the population who use this scenario at all. It also produces a `frequency` value, which is the mean number of times per year this scenario occurs among that fraction of the population specified by prevalence. Since SHEDS operates on the basis of one random day, the `frequency` is divided by 365 and then passed to the `p.round` (probabilistic rounding) function, which rounds either up or down to the nearest integer. Very common events may happen more than once in a day. The function also produces a `mass` variable, which refers to the mass of the product in grams in a typical usage event. The `composition` is the percentage of that mass that is the chemical in question. The final output, `exp.ddd.mass`, is the product of the `prevalence`, `frequency`, `mass`, `composition`, and the fraction going down the drain (`f.drain`, a variable in the `sd` input). The result is in grams per person-day.

Value

exp.ddd.mass The calculated quantity of chemical going down the drain (i.e., from laundry detergent) and entering the sewer system per person per day.

Author(s)

Kristin Isaacs, Graham Glen

See Also

`run`, `p.round`

`eval.factors` *eval.factors*

Description

Assigns distributions for each relevant exposure factor for each theoretical persons to be modeled in SHEDS.HT.

Usage

```
eval.factors(r, q, ef)
```

Arguments

`r`	Vector of row numbers, equivalent to the number of theoretical people in the model sample. This input is created internally by the `add.factors` function.
`q`	User-specified list of desired quantiles to be included in the output.
`ef`	Distributional parameters for the exposure factors; an output of the `exp.factors` function and an input argument to the `add.factors` function

Value

z A data frame specifying the form of the distribution and relevant parameters for each combination of exposure factor and individual person. The parameters include:

`form` The form of the distribution for each exposure factor (i.e., point, triangle).

`par1-par4` The parameters associated with the distributional form specified in form.

`lower.trun` the lower limit of values to be included in the distribution.

`upper.trun` The upper limit of values to be included in the distribution.

`resamp` Logical field where: yes=resample, no=stack at truncation bounds.

`q` The quantiles of the distribution corresponding to those specified by the user in the `q` input.

`p` The probabilities associated with the quantiles.

`v` The values associated with the quantiles.

Author(s)

Krisin Isaacs, Graham Glen

See Also

add.factors, exp.factors

`ExportDataTables` *A function used to export .rda files distributed with ShedsHT package*

Description

This is a utility function used to export consumer product data sets into CSV format.

Usage

```
ExportDataTables(data_set_name, output_pth, output_fname,
  quote_opt = TRUE, na_opt = "", row_names_opt = FALSE)
```

Arguments

`data_set_name`	A string type parameter, represents name of an ".rda" data set.
`output_pth`	A string type parameter, represents location to store exported CSV file.
`output_fname`	A string type parameter, represents name of the exported CSV file.
`quote_opt`	A string type parameter, represents a logical argument for if any character or factor columns should be surrounded by double quotes or not.
`na_opt`	A string type parameter, the string to use for missing values in the data.
`row_names_opt`	A string type parameter, either a logical value indicating whether the row names of x are to be written along with x, or a character vector of row names to be written.

Value

No variable will be returned. Instead, the function will save the data object into the file of choice.

`filter_sources` *filter_sources*

Description

This function is used to select a subset of sources from another "source" file.

Usage

```
filter_sources(in.file = "source_scen_prods.csv",
  out.file = "source_scen_subset.csv", ids = c("ALL"),
  types = c("ALL"))
```

Arguments

`In.file`	a csv file that contains expsoure info default = "source_scen_prods.csv"
`Out.file`	default = "source_scen_prods.csv"
`IDS`	default = "ALL"
`Types`	default = "ALL"

Details

This function is used to select a subset of sources from another "source" file. This is achieved by specifying either a list of the desired source ids, or one or more sources types. The allowed types are "A" for articles, "F" for foods, or "P" for products. Use the c() function to list more than one item (e.g. types=c("F","P") for foods and products). Any of the three types of source files (that is, source_scen, source_chem, or source_vars) may be used. Specifying specific sources is achieved using the "ids" argument. This may require examining the in.file beforehand, to obtain the correct source.id values for the desired sources.

Value

A .csv file of the same type as in.file, with the same variables and data, but fewer rows. The selected rows have source.type matching one of the elements in the "types" argument, and also have source.id matching one of the elements of the "ids" argument. If either argument is missing, then all sources automatically match it. Filter_sources also returns an R object containing the same data as the output .csv file.

`food.migration` *food.migration*

Description

Calculates chemical exposure from migration of chemicals from packaging or other contact materials into food (which is then consumed). This is added to the `dietary` output calculated by the `food.residue` function to establish a total exposure from the dietary pathway.

Usage

```
food.migration(cdata, sdata, cb, ftype)
```

Arguments

cdata	The list of scenario-specific information for the chemicals being evaluated. Generated internally.
sdata	The chemical-scenario data specific to relevant combinations of chemical and scenario. Generated internally.
cb	A copy of the base data set output from the make.cbase function, with columns added for exposure variables.
ftype	Food consumption database which stores data on consumption in grams per day of each food type for each person being modeled. Generated internally from the Diet_diaries data set.

Details

If migration data is stored in the cdata argument, this is added to the total exposure calculated in the food.residue function.

Value

dietary The calculated quantity of chemical exposure from food residue and migration of chemicals into food from packaging and other contact materials in grams per person per day.

Author(s)

Kristin Isaacs, Graham Glen

See Also

Diet_diaries, run, p.round, make.cbase, food.residue

---

food.residue *food.residue*

Description

Models exposure to chemicals from consumption of food containing a known chemical residue for each theoretical person.

Usage

```
food.residue(cdata, cb, ftype)
```

Arguments

cdata	The list of scenario-specific information for the chemicals being evaluated. Generated internally.
cb	Output of the make.cbase function.
ftype	Food consumption database which stores data on consumption in grams per day of each food type for each person being modeled. Generated internally from the Diet_diaries data set.

Details

In this function, the variable foods is defined as a list of the names of the food groups, which are stored in both the ftype and cb arguments. The FOR loop picks the name of each food group as a string, and converts it to a variable name. For example, the food group "FV" may have corresponding variables residue.FV, zeros.FV (number of nondetects), and nonzeros.FV (number of detects) on the cb data set. If these variables are not present, no exposure results. The variables may be present, but an individual person may still receive zero exposure because not all samples of that food are contaminated, as indicated by the zeros.FV value. The nonzeros.FV value is used to determine the likelihood of contamination, with the residue variable determining the amount found (when it is nonzero). The dietary exposure is the product of the consumption as reported in the ftypes input (in grams) and the residue.FV (in micrograms of chemical per gram of food), summed over all food groups. Three other vectors are returned (corresponding to dermal exposure, inhalation exposure, and inhalation dose), but these are currently set to zero for the dietary pathway. In principle, eating food could result in dermal exposure (for finger foods), or inhalation (as foods may have noticeable odors), but such exposures are not large enough to be of concern at present.

Value

dietary The calculated quantity of chemical exposure from food residue in grams per person per day.

Author(s)

Kristin Isaacs, Graham Glen

See Also

Diet_diaries, run, p.round, generate.person.vars, food.migration

---

gen.factor.tables *gen.factor.tables*

Description

Constructs tables of non-media specific exposure factors for each relevant combination of age, gender, and season. The input is from the Exp_actors file (specified on the Run file) after being read through the read.exp.factors function.

Usage

```
gen.factor.tables(ef = exp.factors)
```

Arguments

ef	A data set created internally using the run function and the read.exp.factors function to import the user specified Exp_factors input file. The data set contains the distributional parameters for the exposure factors. All of these variables may have age or gender-dependent distributions, although in the absence of data, many are assigned a single distribution from which all persons are sampled.

Value

exp.gen Output consists of non-media specific exposure factors as a data table. Depending on the user input, these generated exposure factors may be gender and/or season specific. Each gender-season combination is assigned a row number pointing to the appropriate distribution on the output dataset from the read.exp.factors function. Thus, exp.gen consists of 8 rows per variable (2 genders x 4 seasons), regardless of the number of different distributions used for that variable. In addition, if the function runs successfully, the following message is printed: "General Factor Tables completed"

Author(s)

Kristin Isaacs, Graham Glen

See Also

Exp_factors, Run, run, read.exp.factors

---

get.fug.concs *get.fug.concs*

Description

Performs fugacity calculations to evaluate time-dependent chemical flows.

Usage

```
get.fug.concs(sdata, chem.data, x, cfug)
```

Arguments

sdata	The chemical-scenario data specific to relevant combinations of chemical and scenario. Generated internally.
chem.data	The list of scenario-specific information for the chemicals being evaluated. Generated internally.
x	The output of the add.fugs function.
cfug	The output of the chem_fug function, a data set with the chemical specific parameters for each combination of chemical and simulated person. For each chemical, the chemical specific parameters assigned to a given person are randomly sampled from distributions on those parameters. These distributions are created from point estimates to reflect real-world uncertainty and variability.

Details

This is one of two functions that perform fugacity calculations. This one evaluates dynamic or time-dependent chemical First, a set of local variables are determined for use in later calculations. These are a mix of fixed and chemical-dependent variables (evaluated separately for each person). Some variables, like chemical mass and app.rates, vary with each source, so these calculations are repeated for each source-scenario. Second, the eigenvalues and eigenvectors of the jacobian matrix are calculated. Since the fugacity model has been reduced to just two compartments (air and surface), the solutions can be expressed analytically, and there is no explicit invocation of any linear algebra routines that would normally be required. Third, the variables composing the `concs` output are evaluated. The variables `m.c.air` and `m.c.sur` are the time-constant masses, while `m.t0.air` and `m.t0.sur` are the time-dependent masses at t=0. The time-constant parts are zero here because the permanent sources (i.e. `c.src.air` and `c.src.sur`) are assumed to be zero in these calculations. The time-dependent masses are multiplied exponentially as a function of time, and thus approach zero when enough time has passed.

Value

concs A data set containing calculated dynamic chemical flows for each unique combination of simulated person and chemical.

Author(s)

Kristin Isaacs, Graham Glen

See Also

`chem_fugs`, `Fugacity`, `Run`, `run`, `get.y0.concs`

`get.y0.concs` *get.y0.concs*

Description

Performs fugacity calculations to evaluate constant (time-independent) chemical flows, such as emissions from household articles.

Usage

```
get.y0.concs(sdata, chem.data, pdmff, cfug)
```

Arguments

`sdata`	The chemical-scenario data specific to relevant combinations of chemical and scenario. Generated internally.
`chem.data`	The list of scenario-specific information for the chemicals being evaluated. Generated internally.
`pdmff`	Output from the `add.fugs` function. A data set containing values sampled from the distributions of each relevant variable in the `Fugacity` input file for each theoretical person.

cfug — The output of the chem_fug function, a data set with the chemical specific parameters for each combination of chemical and simulated person. For each chemical, the chemical specific parameters assigned to a given person are randomly sampled from distributions on those parameters. These distributions are created from point estimates to reflect real-world uncertainty and variability.

Details

This function evaluates the chemical concentrations resulting from constant source emissions. Thus, the function employs the steady state solution to the fugacity equations. The basis for these calculations is that the chemical sources are from articles, and are thus permanent and unchanging. The chemical concentrations will therefore quickly adjust so that the flows are balanced, and the concentrations remain fixed thereafter.

Value

concs A data set containing calculated steady state chemical flows for each unique combination of simulated person and chemical.

Author(s)

Kristin Isaacs, Graham Glen

See Also

chem_fugs, Fugacity, Run, run, get.y0.concs

---

indir.exposure *indir.exposure*

Description

Models the indirect exposure to chemicals in the home for each theoretical person.

Usage

```
indir.exposure(sd, cb, concs, chem.data)
```

Arguments

sd — The chemical-scenario data specific to relevant combinations of chemical and scenario. Generated internally.

cb — A copy of the base data set output from the make.cbase function, with columns added for exposure variables.

concs — The concentration of the chemical (in air and/or on surfaces) being released into the environment. Outpu of the get.fug.concs function.

chem.data — The list of scenario-specific information for the chemicals being evaluated. Generated internally.

Details

Indirect exposure happens after a product is no longer being used or applied, due to chemical lingering on various surfaces or in the air. People who come along later may receive dermal or inhalation exposure from residual chemical in the environment. SHEDS.HT currently has two indirect exposure scenarios. One which applies to a one-time chemical treatment applied to a house, and another which applies to continual releases from articles. Both scenarios consist of two parts: the first determines the appropriate air and surface concentrations. That code is in the `Fugacity` module. The second part is the exposure calculation by the current function. Both types of indirect exposure scenarios call this function. The surface and air concentrations from the `Fugacity` module are premised on the product use actually occurring. Hence, `indir.exposure` starts by multiplying those concentrations by the `prevalence` (which is either 0 or 1, evaluated separately for each person). For air, the exposure is the average daily concentration, which is the event concentration multiplied by the fraction of the day spent in that event. The inhaled dose is the product of the exposure, the basal ventilation rate, and the PAI factor (multiplier for the basal rate). A factor of 1E+06 converts the result from grams per day to micrograms per day. Dermal exposure results from skin contact with surfaces. The surface concentration (ug/cm2) is multiplied by the fraction available for transfer (`avail.f`, unitless), the transfer coefficient (`dermal.tc`, in cm2/hr), and the contact `duration` (hr/day). The result is the amount of chemical transferred onto the skin (ug/day).

Value

indir Indirect exposure to chemicals in the home in ug per person per day.

Author(s)

Kristin Isaacs, Graham Glen

See Also

`Fugacity`, `get.fug.concs`, `make.cbase`, `run`

`make.cbase` *make.cbase*

Description

Extends the `base` data set, output from the `generate.person.vars` function, to include a set of chemical-specific exposure variables. Currently, all new variables are initialized to zero.

Usage

```
make.cbase(base, chem)
```

Arguments

`base`	Data set of all chemical-independent information needed for the exposure assessment. Each row corresponds to a simulated person. The variables consist of age, gender, weight, diet and activity diaries, food consumption, minutes in each micro, and the evaluation of the exposure factors for that person.

`chem` List of the chemical(s) of interest, determined via the `chemical` input in the Run file. The exposure variables for the specified chemicals will be appended to the `base` data set.

Value

The output is a data set cb consisting of the base data set, appended by the following columns:

chemrep The unique ID corresponding to each chemical specified in the `chem` argument.

inhal.abs.f For each person, the fraction of absorption of a given chemical when inhaled.

urine.f For each person, the fraction of intake of a given chemical that is excreted through urine.

exp.inhal.tot For each person, the total exposure of a given chemical via the direct inhalation scenario.

dose.inhal.tot For each person, the corresponding dose for total exposure of a given chemical via the direct inhalation scenario.

exp.dermal.tot For each person, the total exposure of a given chemical via the direct dermal application scenario.

exp.ingest.tot For each person, the total exposure of a given chemical via the direct ingestion scenario.

exp.dietary.tot For each person, the total exposure of a given chemical via the dietary scenario.

exp.migrat.tot For each person, the total exposure of a given chemical via the migration scenario.

exp.nondiet.tot For each person, the total exposure of a given chemical via the non-dietary food exposure scenario.

exp.ddd.tot For each person, the total exposure of a given chemical via the down the drain scenario.

exp.window.tot For each person, the total exposure of a given chemical via the out the window scenario.

Author(s)

Kristin Isaacs, Graham Glen

See Also

Run, run

`med.factor.tables` *med.factor.tables*

Description

Constructs tables of media specific exposure factors for each relevant combination of age, gender, and season, and each of three media specific variables in the `ef` argument. These are avail.f, dermal.tc, and om.ratio.

Usage

```
med.factor.tables(ef, media.sur)
```

Arguments

ef	A data set created internally using the run function and the read.exp.factors function to import the user specified Exp_factors input file. The data set contains the distributional parameters for the exposure factors. All of these variables may have age or gender-dependent distributions, although in the absence of data, many are assigned a single distribution from which all persons are sampled.
media.sur	A list of surface media. This data set is created internally by sub-setting the Media input file (read in with the read.media.file function within the run function) to extract only surface media.

Value

exp.med Media specific exposure factors presented as a data table. The present version of the model consists of only 3 such exposure factors, but the code will accept more. Depending on the user input, these generated exposure factors may be gender and/or season specific in addition to media specific. The output consists of 24 rows per variable (2 genders x 4 seasons x 3 media), when all ages share the same distribution. If a variable has N age categories (each with its own distribution) then there are (24 x N) rows for that variable. In addition, if the function runs successfully, the following message is printed: "Media-specific Factor Tables completed"

Note

The first input argument to med.factor.tables (ef) is created by reading in the Exp_factors input file (specified on the Run file) with the read.exp.factors function within the run function. The media.sur input is created by sub-setting the Media input file (read in with the read.media.file function within the run function).

Author(s)

Kristin Isaacs, Graham Glen

See Also

Media, Exp_factors, Run, run, read.exp.factors, read.media.file

---

p.round *p.round*

Description

p.round performs stochastic or probabilistic rounding of non-integer values.

Usage

```
p.round(x = NULL, q = NA)
```

Arguments

x	Required Default = none
q	Optional Default = none

Details

The input "x" is a vector of values to be rounded. Each value of x is rounded independently, either up or down. The fractional value of x is used to determine weights for rounding in each direction. For example, if x=4.3, then it is rounded up to 5 with probability 0.3, else rounded down to 4 (with probability 0.7). The p.round function always returns integer values, and does not introduce bias. In the above example, if the argument were 4.3 a large number of times, then the returned values (all either 4 or 5) would average 4.3 with a small residual error which approaches zero as "n" gets large.

Value

A vector of the same length as the input "x", containing all integer values.

Author(s)

Kristin Isaacs, Graham Glen

---

post.exposure *post.exposure*

Description

Resolves the fate of chemical after initial exposure has occurred. This involves parsing out the amount of chemical removed and amount ultimately contributing to the exposure dose for each person.

Usage

```
post.exposure(cb, cprops)
```

Arguments

cb	A copy of the base data set output from the make.cbase function, with columns added for exposure variables.
cprops	The chemical properties required for SHEDS-HT. The default file (the Chem_props file read in by the read.chem.props function and modified before input into the current function) was prepared from publicly available databases using a custom program (not part of SHEDS-HT). The default file contains 7 numerical inputs per chemical, and the required properties are molecular weight (MW), vapor pressure (VP.Pa), solubility (water.sol.mg.l), octanol-water partition coefficient (log.Kow), air decay rate (half.air.hr), decay rate on surfaces (half.sediment.hr), and permeability coefficient (Kp).

Details

This function resolves the fate of chemical after initial exposure has occurred. The same function applies to all exposure scenarios. The dermal exposure is the most complicated, as there are five removal methods. All five are randomly sampled. While the sum of the five means is close to one, the sum of five random samples might not be, so these samples are treated as fractions of their sum. The `rem.bath` variable is either 0 or 1, so the bath removal term is either zero or the sampled bath removal efficiency. The `rem.brush` term is also simple. The handwashing (`rem.wash`) and hand-to-mouth (`rem.hmouth`) transfer terms are non-linear, because higher frequencies have less chemical available for removal on each repetition. The algorithms in place were fitted to output from SHEDS-Multimedia, which were summed to daily totals. The final removal term is dermal absorption (`rem.absorb`). The base value is multiplied by the `Kp` factor from the `cprops` argument, and divided by the value for permethrin, as the values from SHEDS-Multimedia were based on a permethrin run. The five terms are evaluated separately for each person, as is their sum. Each is then converted to a fraction of the whole. The fractions may be quite different from one person to another. For one, perhaps 80% of the dermal loading is removed by a bath/shower, while for another person it is 0% because they did not take one. For the latter person, other four removal terms are (on average) five times larger than for the former person, because together they account for 100% of the removal, instead of just 20%. The rest of the `post.exposure` function is mostly a matter of bookkeeping. The hand-to-mouth dermal removal term becomes an ingestion exposure term. Summing exposures across routes is dubious, in part because inhalation exposures use different units from the others, and because much of the dermal exposure never enters the body. In addition, summing dermal and ingestion exposures may double-count the hand-to-mouth term. However, intake dose may be summed. In SHEDS, "intake dose" is the sum of the inhaled dose (which is the amount of chemical entering the lungs in ug/day), the ingestion exposure (which is the amount entering the GI tract in ug/day), and the dermal absorption (the amount penetrating into or through the skin, so it cannot otherwise be removed, in ug/day). The "absorbed dose" is also calculated: for dermal it is the same as the intake dose, but for ingestion and inhalation there is another absorption factor, which was set on the exposure factors input file. An estimate of the chemical in urine (in ug/day) is made. Both the intake dose and the absorbed dose are reported in both (ug/day) and in (mg/kg/day). Note that the latter requires the body weights of each individual. These cannot be obtained from the former just by knowing the average body weight in SHEDS. The above variables for each simulated person are written to the `fexp` object (the name stands for "final exposure"). Each chemical writes over the previous `fexp`, so the data must first be summarized and written to an output file.

Value

fexp Absorbed dose and intake dose of a given chemical for each theoretical person being modeled for all exposure scenarios.

Author(s)

Kristin Isaacs, Graham Glen

See Also

`make.cbase`, `Chemprops_small`

`quantiles` *quantiles*

Description

This function is similar to the built-in R function "quantile", but it returns a list of preselected quantiles of the set of values in the vector "x".

Usage

```
quantiles(x)
```

Arguments

`x`	Default = none

Details

This function is similar to the built-in R function "quantile", but it returns a list of preselected quantiles of the set of values in the vector "x". Specifically, it returns all the following quantiles: .005, .01, .025, .05, .1, .15, .2, .25, .3, .4, .5, .6, .7, .75, .8, .85, .9, .95, .975, .99, .995

Value

This function is used to construct the tables in the "Allstats" output files.

Author(s)

Kristin Isaacs, Graham Glen

`read.act.diaries` *read.act.diaries*

Description

Read.act.diaries reads human activity diaries from the .csv file indicated by "filename". "Specs" contains the run specifications from the run.file, and is used to subset the activity diaries by age, gender, or season, if requested.

Usage

```
read.act.diaries(filename, specs)
```

`read.chem.props` *read.chem.props*

Description

Read.chem.props reads the chemical properties from the .csv file indicated by "filename". "Specs" contains the run specifications from the run.file, and is used to subset the chemicals by the list provided on the run.file. If no such list was given, then all chemicals are kept.

Usage

```
read.chem.props(filename, specs)
```

`read.diet.diaries` *read.diet.diaries*

Description

Read.diet.diaries reads the food diaries from the .csv file indicated by "filename". "Specs" contains the run specifications from the run.file, and is used to subset the activity diaries by age and/or gender.

Usage

```
read.diet.diaries(filename, specs)
```

`read.exp.factors` *read.exp.factors*

Description

Read.exp.factors reads the exposure factors from the .csv file indicated by "filename".

Usage

```
read.exp.factors(filename)
```

`read.fug.inputs` *read.fug.inputs*

Description

Read.fug.inputs reads the non-chemical dependent inputs for fugacity modeling in SHEDS from the .csv file indicated by "filename".

Usage

```
read.fug.inputs(filename)
```

`read.media.file` *read.media.file*

Description

Read.media.file reads the names, properties, and associated microenvironments for each of the potential exposure media in SHEDS.

Usage

```
read.media.file(filename)
```

`read.phys.file` *read.phys.file*

Description

Read.phys.file reads the physiology data from the .csv file indicated by "filename".

Usage

```
read.phys.file(filename)
```

`read.pop.file` *read.pop.file*

Description

Read.pop.file reads the population data from the .csv file indicated by "filename".

Usage

```
read.pop.file(filename, specs)
```

`read.run.file` *read.run.file*

Description

Each SHEDS run has its own "run.file" that the user prepares before the run. This file contains all the settings and file references needed for the run. Read.run.file occurs at the start of each SHEDS run. The contents of the run.file are examined, and stored in the R object "specs".

Usage

```
read.run.file(run.file = "run_test.txt")
```

`read.source.chem.file` *read.source.chem.file*

Description

Read.source.chem.file reads the distributions that are specific to combinations of source and chemical from the indicated .csv file.

Usage

```
read.source.chem.file(filename, scenSrc, specs)
```

`read.source.scen.file` *read.source.scen.file*

Description

Read.source.scen.file reads the list of active exposure scenarios for each potential source of chemical from the .csv file indicated by "filename".

Usage

```
read.source.scen.file(filename)
```

`read.source.vars.file` *read.source.vars.file*

Description

Read.source.vars.file reads the distributions that are specific to combinations of source and chemical from the indicated .csv file.

Usage

```
read.source.vars.file(filename, src.scen)
```

run *run*

Description

Function to call the Run txt file, which consists of user-defined parameters and calls to input files required to initialize a SHEDS.HT run.

Usage

```
run(run.file = "", wd = "")
```

Arguments

run.file	The name of the run file to be used for a given run. Many different "Run"" files may be set up for special purposes.The one being invoked must be present in the inputs folder.
wd	The user's working directory. The working directory should contain an inputs folder, containing all necessary SHEDS.HT input files. The wd value should be set once for each SHEDS.HT installation by replacing the default value in the definition of run().

Details

This function is used to provide R with information necessary to produce a SHEDS-HT run and to initialize necessary parameters. No values are returned. In order to produce a successful run, all necessary input files should be stored in the working directory specified by the wd argument.

Value

No variable will be returned.

Author(s)

Kristin Isaacs, Graham Glen

See Also

Run.txt, read.run.file

scen.factor.indices *scen.factor.indices*

Description

Constructs scenario-specific exposure factors for each relevant combination of age and gender. The input is derived internally from the Source_vars file specified on the Run file.

Usage

```
scen.factor.indices(sdat, expgen)
```

Arguments

sdat	The chemical-scenario data specific to a given combination of chemical and scenario.
expgen	Non-media specific exposure factors as a data table. Output from gen.factor.tables

Details

The constructed factors are not media-specific, although they are scenario-specific, and most scenarios include just one surface medium. This function evaluats the scenario-specific exposure factors separately from the other factors because these scenario-specific factors may change with every chemical and scenario, whereas the other factors remain the same across chemicals and scenarios for each person.

Value

Returns indices of scenario-specific exposure factors for each relevant age and gender combination. The output is only generated internally.

Author(s)

Kristin Isaacs, Graham Glen

select.people *select.people*

Description

Assigns demographic and physiological variables to each theoretical person to be modeled.

Usage

```
select.people(n, pop, py, act.p, diet.p, act.d, diet.d, specs)
```

Arguments

n	Number of persons.
pop	The population input; output of read.pop.file function. Contains counts by gender and each year of age from the 2000 U.S. census. When a large age range is modeled, this ensures that SHEDS chooses age and gender with the correct overall probability. #'
py	Regression parameters on the three physiological variables of interest (weight, height, and body mass index) for various age and gender groups. Output of the read.phys.file function.
act.p	Activity diary pools; output of the act.diary.pools function. Each element in this input is a list of acceptable activity diary numbers for each year of age, for each gender, weekend, and season combination.
diet.p	Dietary diary pools; output of the diet.diary.pools function. Each element in this input is a list of acceptable activity diary numbers for each year of age, for each gender, weekend, and season combination.
act.d	Activity diaries, which indicate the amount of time and level of metabolic activity in various 'micros'. Each line of data represents one person-day (24 hours). Output of the read.act.diaries function.
diet.d	Daily diaries of dietary consumption by food group; output of the read.diet.diaries function. Each line represents one person-day, with demographic variables followed by amounts (in grams/day) for a list of food types indicated by a short abbreviation on the header line.
specs	Output of the read.run.file function, which can be modified by the update.specs function before input into select.people.

Details

This is the first real step in the modeling process. It first fills an array q with uniform random numbers, with ten columns because there are 10 random variables defined by this function. There number of rows correspond to the number of persons, capped at the set.size specified in the Run input file (typically 5000). Gender is selected from a discrete (binomial) distribution where the counts of males and females in the study age range determines the gender probabilities. Age is tabulated next, separately for each gender. The counts by year of age are chosen for the appropriate gender and used as selection weights. Season is assigned randomly (equal weights) using those specified in the Run input file. Weekend is set to one or zero, with a chance of 2/7 for the former. The next block of code assigns physiological variables. Weight is lognormal in SHEDS, so a normal is sampled first and then exp() is applied. This means that the weight parameters refer to the properties of log(weight), which were fit by linear regression. The basal metabolic rate (bmr), is calculated by regression. A minimum bmr is set to prevent extreme cases from becoming zero or negative. The alveolar breathing ventilation rate corresponding to bmr is also calculated. The SHEDS logic sets activities in each micro to be a multiple of these rates, with outdoor rates higher than indoor, and indoor rates higher than sleep rates. This calculated activities affect the inhaled dose. The skin surface area is calculated using regressions based on height and weight for 3 age ranges. The next step is to assign diaries. Here, a FOR loop over n persons (n rows) is used to assign appropriate diet and activity diary pools to each person. An empirical distribution is created, consisting of the list of diary numbers for each pool. The final step is to retrieve the actual data from the chosen activity and diet diaries, and the result becomes pd.

Value

pd A dataframe of "person-demographics": assigned demographic and physiological parameters for each theoretical person modeled in SHEDS.HT

Author(s)

Kristin Isaacs, Graham Glen

See Also

`run`, `read.pop.file`, `read.phys.file`, `act.diary.pools`, `diet.diary.pools`, `read.act.diaries`, `read.diet.diaries`, `update.specs`

`set.pars` *set.pars Adjusts certain SHEDS input distribution types to conform with the requirements of the* `distrib` *function.*

Description

set.pars Adjusts certain SHEDS input distribution types to conform with the requirements of the `distrib` function.

Usage

```
set.pars(vars)
```

Arguments

`vars`	Output of the `read.source.chem.file` or the `read.source.vars.file` functions.

Details

Set.pars adjusts certain SHEDS input distribution types to conform with the requirements of the Distrib function. The lognormal parameters are changed from arithmetic mean and standard deviation to geometric mean and geometric standard deviation. For the normal distribution, the standard deviation is computed from the mean and coefficient of variation (CV). For user prevalence, the input may be specified either as a point value (indicating probability) or as a Bernoulli distribution. If the former is used, it is converted to the latter.

Value

v The modified version of the input vars.

Author(s)

Kristin Isaacs, Graham Glen

See Also

`run`, `read.source.chem.file`, `read.source.vars.file`

setup *setup*

Description

Loads required R packages and sources the modules needed to perform a SHEDS.HT run. The user might need to download the packages if they are not already present (see Dependencies).

Usage

```
setup(wd = "")
```

Arguments

wd	The User's working directory (set in the inputs to the run function). Should contain all necessary SHEDS.HT inputs in an /Input folder.

Value

No values returned.

Note

Requires: data.table, plyr, stringr, ggplot2

Author(s)

Kristin Isaacs, Graham Glen

See Also

run

summarize.chemical *summarize.chemcial*

Description

Summarize.chemical writes a .csv file containing a summary of the exposure and dose results from the object "x".

Usage

```
summarize.chemical(x, c, chem, chemical, set, sets, specs)
```

Arguments

x	Default = none Exposure data set
c	Default = none Index # for chemical
chem	Default = none CAS for chemical
chemical	Default = none Full chemical name
set	Default = none Index # for set of simulated persons
sets	Default = none Total # of sets in this SHEDS run
specs	Default = none List of settings from the "run" input file

Details

Tables are produced for each of the following cohorts: males, females, females ages 16-49, age 0-5, age 6-11, age 12-19, age 20-65, age 66+, and a table for all persons. The variables that are summarized are: dermal exposure, ingestion exposure, inhalation exposure, inhaled dose, intake dose, dermal absorption, ingestion absorption, inhalation absorption, total absorption in micrograms per day, total absorption in milligrams per kilogram per day, and chemical mass down the drain. In each table, the following statistics are computed across the appropriate subpopulation: mean, standard deviation, and quantiles .005, .01, .025, .05, .1, .15, .2, .25, .3, .4, .5, .6, .7, .75, .8, .85, .9, .95, .975, .99, .995.

Value

If "x" is a single set of data (that is, if the argument "set" is between 1 and sets, inclusive), then the .csv file created by this function has the suffix "_set#stats.csv", where "#" is the set number. If the "set" argument is "allstats", then the .csv file has the suffix _allstats.csv". The output file contains the tables for all cohorts with a non-zero population.

Author(s)

Kristin Isaacs, Graham Glen

---

summary.stats *summary.stats*

Description

Summary.stats constructs the table of exposure and dose statistics for cohort entered into summarize.chemical.

Usage

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

Arguments

x.	Data set passed from summarize.chemical for a cohort

Details

Summary.stats is called by summarize.chemical. The input data set "y" is one population cohort from the exposure data set passed into summarize.chemical.

Value

y A data frame object with 23 rows and 11 columns, with each column being an exposure or dose variable, and each row containing a statistic for that variable. For each expsoure varianle the total expsoure, quantiles, mean, and SD.

Author(s)

Kristin Isaacs, Graham Glen

See Also

summarize.chemical

---

trimzero *Trimzero*

Description

This function removes initial zeroes from CAS numbers.

Usage

```
trimzero(x, y)
```

Arguments

x	aCAS number. Defult is none
y	A dummy argument This helps with the removal of the zeros inteh CAS number

Details

Each CAS number has three parts, separated by underscores. The first part is up to seven digits, but optionally, leading zeroes are omitted. For example, formaldehyde may be either "0000050_00_0" or "50_00_0". SHEDS needs to match CAS numbers across input files, and trimzero is used to ensure matching even when the input files follow different conventions.

Value

y a shorter CAS number. If the initial part is all zero (as in "0000000_12_3"), one zero is left in the first part (that is, "0_12_3" for this example).

Author(s)

Kristin Isaacs, Graham Glen

unpack *unpack*

Description

unpack

Usage

```
unpack(filelist = "")
```

Details

This function is used when ShedsHT is run as an R package. Each time a new working directory is chosen, use unpack() to convert the R data objects into CSV files. Note that both /inputs and /output folders are needed under the chosen directory.Unpack() may be used with a list of object names, in which case just those objects are converted to CSV. This is useful when re-loading one or more defaults into a folder where some of the CSV files have changes, and should not be overwritten. A blank argument or empty list means that all the csv and TXT files in the R package are converted.

Author(s)

Kristin Isaacs, Graham Glen

update.specs *update.specs*

Description

"Specs" is the list of run settings read from the run.file. "Dt" is a data table of source-chemical combinations for which distributions have been specified. "Specs" contains a list of chemicals to be processed in the SHEDS run, and if any of these chemicals are missing from the "dt" table, then update.specs removes them from the list. Otherwise, "specs" is not altered.

Usage

```
## S3 method for class 'specs'
update(specs, dt)
```

`vpos` *vpos*

Description

This function locates an item in a list.

Usage

```
vpos(v, list)
```

Details

this one was written because it does not require additional R packages and its behavior can be easily examined.

Author(s)

Kristin Isaacs, Graham Glen

`write.persons` *write.persons*

Description

This function writes demographic, exposure, and dose variables to a separate output file for each chemical.

Usage

```
write.persons(x, chem, set, specs)
```

Details

THis function rounds the variables to a reasonable precision so that the files are more readable, as unrounded output is cluttered with entries with (say) 14 digits, most of which are not significant.

Author(s)

Kristin Isaacs, Graham Glen

Index