Package ‘bootf2’

July 22, 2025

Type Package

Title Simulation and Comparison of Dissolution Profiles

Version 0.4.1

Date 2021-08-19

Description Compare dissolution profiles with confidence interval of similarity factor f2 using bootstrap methodology as described in the literature, such as Efron and Tibshirani (1993, ISBN:9780412042317), Davison and Hinkley (1997, ISBN:9780521573917), and Shah et al. (1998) <doi:10.1023/A:1011976615750>. The package can also be used to simulate dissolution profiles based on mathematical modelling and multivariate normal distribution.

License GPL (>= 3)

URL `https://github.com/zhengguoxu/bootf2`

BugReports `https://github.com/zhengguoxu/bootf2/issues`

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Depends R (>= 4.0.0)

Imports ggplot2, minpack.lm, MASS, readxl, stats

Suggests testthat (>= 3.0.0), knitr, rmarkdown

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Zhengguo Xu [aut, cre]

Maintainer Zhengguo Xu <`zhengguoxu@gmail.com`>

Repository CRAN

Date/Publication 2021-08-25 12:50:22 UTC

Contents

`bootf2` *Estimate 90% Confidence Intervals of f_2 with Bootstrap Methodology*

Description

Main function to estimate 90% confidence intervals of f_2 using bootstrap methodology.

Usage

```
bootf2(test, ref, path.in, file.in, path.out, file.out,
       n.boots = 10000L, seed = 306L, digits = 2L, alpha = 0.05,
       regulation = c("EMA", "FDA", "WHO","Canada", "ANVISA"),
       min.points = 1L, both.TR.85 = FALSE, print.report = TRUE,
       report.style = c("concise", "intermediate", "detailed"),
       f2.type = c("all", "est.f2", "exp.f2", "bc.f2",
                   "vc.exp.f2", "vc.bc.f2"),
       ci.type = c("all", "normal", "basic", "percentile",
                   "bca.jackknife", "bca.boot"),
       quantile.type = c("all", as.character(1:9), "boot"),
       jackknife.type = c("all", "nt+nr", "nt*nr", "nt=nr"),
       time.unit = c("min", "h"), output.to.screen = FALSE,
       sim.data.out = FALSE)
```

Arguments

`test, ref`	*Data frames* of dissolution profiles of test and reference product if `path.in` and `file.in` are not specified; otherwise, they should be *character* strings indicating the worksheet names of the Excel file where the dissolution data is saved. See Input/Output in Details.
`path.in, file.in, path.out, file.out`	*Character* strings of input and output directories and file names. See Input/Output in Details.
`n.boots`	An *integer* indicating the number of bootstrap samples.
`seed`	*Integer* seed value for reproducibility. If missing, a random seed will be generated for reproducibility purpose.

`digits`	An *integer* indicating the decimal points for the output.
`alpha`	A *numeric* value between 0 and 1 to estimate $(1 - 2 \times \alpha) \times 100$ confidence interval.
`regulation`	*Character* strings indicating regulatory guidelines. @seealso `calcf2()` for details on regulation rules.
`min.points`	An *integer* indicating the minimum time points to be used to calculate f_2. For conventional f_2 calculation, the default is 3, however, for bootstrap f_2, the value should be lower as there might be less time points available in certain bootstrap samples. The default is 1. @seealso `calcf2()`.
`both.TR.85`	*Logical*. If `TRUE`, and if `regulation = "FDA"`, all measurements up to the time points at which both test and reference products dissolve more than 85% will be used to calculate f_2. This is the conventional, but incorrect, interpretation of the US FDA rule. Therefore, the argument should only be set to `TRUE` for validation purpose such as comparing the results from old literature that use the wrong interpretation to calculate f_2. @seealso `calcf2()` for details on regulation rules.
`print.report`	*Logical*. If `TRUE`, a plain text report will be produced. See Input/Output in Details.
`report.style`	`"concise"` style produces the estimators and their confidence intervals; `"intermediate"` style adds a list of individual f_2s for all bootstrap samples in the end of `"concise"` report; `"detailed"` style further adds individual bootstrap samples along with their f_2s in the end of `"intermediate"` report. See Input/Output in Details.
`f2.type`	*Character* strings indicating which type of f_2 estimator should be calculated. See Types of estimators in Details.
`ci.type`	*Character* strings indicating which type of confidence interval should be estimated. See Types of confidence intervals in Details.
`quantile.type`	*Character* strings indicating the type of percentile.
`jackknife.type`	*Character* strings indicating the type of jackknife method. See Details.
`time.unit`	*Character* strings indicating the unit of time. It should be either `"min"` for minute or `"h"` for hour. It is mainly used for checking CV rules and making plot. @seealso `calcf2()`.
`output.to.screen`	*Logical*. If `TRUE`, a `"concise"` style summary report will be printed on screen. See Input/Output in Details.
`sim.data.out`	*Logical*. If `TRUE`, all individual bootstrap data sets will be included in the output.

Details

Minimum required arguments that must be provided by the user:

Arguments `test` and `ref` must be provided by the user. They should be `R data frames`, with *time as the first column*, and all individual profiles profiles as the rest columns. The actual names of the columns do not matter since they will be renamed internally.

Input/Output:

The dissolution data of test and reference product can either be provided as *data frames* for `test` and `ref`, as explained above, or be read from an *Excel file* with data of test and reference stored in *separate worksheets*. In the latter case, the argument `path.in`, the directory where the Excel file is, and `file.in`, the name of the Excel file *including the file extension* `.xls` *or* `.xlsx`, must be provided. In such case, the argument `test` and `ref` must be *the names of the worksheets in quotation marks*. The first column of each Excel worksheet must be time, and the rest columns are individual dissolution profiles. The first row should be column names, such as time, unit01, unit02, ... The actual names of the columns do not matter as they will be renamed internally.

Arguments `path.out` and `file.out` are the names of the output directory and file. If they are not provided, but argument `print.report` is `TRUE`, then a plain text report will be generated automatically in the current working directory with file name `test_vs_ref_TZ_YYYY-MM-DD_HHMMSS.txt`, where `test` and `ref` are data set names of test and reference, `TZ` is the time zone such as `CEST`, `YYYY-MM-DD` is the numeric date format and `HHMMSS` is the numeric time format for hour, minute, and second.

For a quick check, set argument `output.to.screen = TRUE`, a summary report very similar to `concise` style report will be printed on screen.

Types of Estimators:

According to Shah et al, the population f_2 for the inference is

$$f_2 = 100 - 25\log\left(1 + \frac{1}{P}\sum_{i=1}^{P}\left(\mu_{\mathrm{T},i} - \mu_{\mathrm{R},i}\right)^2\right),$$

where P is the number of time points; $\mu_{\mathrm{T},i}$ and $\mu_{\mathrm{R},i}$ are *population mean* of test and reference product at time point i, respectively; $\sum_{i=1}^{P}$ is the summation from $i = 1$ to P.

Five estimators for f_2 are included in the function:

1. The estimated f_2, denoted by $\hat{f}_2$, is the one written in various regulatory guidelines. It is expressed differently, but mathematically equivalently, as

$$\hat{f}_2 = 100 - 25\log\left(1 + \frac{1}{P}\sum_{i=1}^{P}\left(\bar{X}_{\mathrm{T},i} - \bar{X}_{\mathrm{R},i}\right)^2\right),$$

   where P is the number of time points; $\bar{X}_{\mathrm{T},i}$ and $\bar{X}_{\mathrm{R},i}$ are mean dissolution data at the ith time point of *random samples* chosen from the test and the reference population, respectively. Compared to the equation of population f_2 above, the only difference is that in the equation of $\hat{f}_2$ the *sample means* of dissolution profiles replace the *population means* for the approximation. *In other words, a point estimate is used for the statistical inference in practice.*

2. The Bias-corrected f_2, denoted by $\hat{f}_{2,\mathrm{bc}}$, was described in the article of Shah et al, as

$$\hat{f}_{2,\mathrm{bc}} = 100 - 25\log\left(1 + \frac{1}{P}\left(\sum_{i=1}^{P}\left(\bar{X}_{\mathrm{T},i} - \bar{X}_{\mathrm{R},i}\right)^2 - \frac{1}{n}\sum_{i=1}^{P}\left(S_{\mathrm{T},i}^2 + S_{\mathrm{R},i}^2\right)\right)\right),$$

   where $S_{\mathrm{T},i}^2$ and $S_{\mathrm{R},i}^2$ are unbiased estimates of variance at the ith time points of random samples chosen from test and reference population, respectively; and n is the sample size.

3. The variance- and bias-corrected f_2, denoted by $\hat{f}_{2,\mathrm{vcbc}}$, does not assume equal weight of variance as $\hat{f}_{2,\mathrm{bc}}$ does.

$$\hat{f}_{2,\mathrm{vcbc}} = 100 - 25\log\left(1 + \frac{1}{P}\left(\sum_{i=1}^{P}\left(\bar{X}_{\mathrm{T},i} - \bar{X}_{\mathrm{R},i}\right)^2 - \frac{1}{n}\sum_{i=1}^{P}\left(w_{\mathrm{T},i}\cdot S_{\mathrm{T},i}^2 + w_{\mathrm{R},i}\cdot S_{\mathrm{R},i}^2\right)\right)\right),$$

where $w_{\mathrm{T},i}$ and $w_{\mathrm{R},i}$ are weighting factors for variance of test and reference products, respectively, which can be calculated as follows:

$$w_{\mathrm{T},i} = 0.5 + \frac{S_{\mathrm{T},i}^2}{S_{\mathrm{T},i}^2 + S_{\mathrm{R},i}^2},$$

and

$$w_{\mathrm{R},i} = 0.5 + \frac{S_{\mathrm{R},i}^2}{S_{\mathrm{T},i}^2 + S_{\mathrm{R},i}^2}.$$

4. The expected f_2, denoted by $\hat{f}_{2,\mathrm{exp}}$, is calculated based on the mathematical expectation of estimated f_2,

$$\hat{f}_{2,\mathrm{exp}} = 100 - 25\log\left(1 + \frac{1}{P}\left(\sum_{i=1}^{P}\left(\bar{X}_{\mathrm{T},i} - \bar{X}_{\mathrm{R},i}\right)^2 + \frac{1}{n}\sum_{i=1}^{P}\left(S_{\mathrm{T},i}^2 + S_{\mathrm{R},i}^2\right)\right)\right),$$

using mean dissolution profiles and variance from samples for the approximation of population values.

5. The variance-corrected expected f_2, denoted by $\hat{f}_{2,\mathrm{vcexp}}$, is calculated as

$$\hat{f}_{2,\mathrm{vcexp}} = 100 - 25\log\left(1 + \frac{1}{P}\left(\sum_{i=1}^{P}\left(\bar{X}_{\mathrm{T},i} - \bar{X}_{\mathrm{R},i}\right)^2 + \frac{1}{n}\sum_{i=1}^{P}\left(w_{\mathrm{T},i}\cdot S_{\mathrm{T},i}^2 + w_{\mathrm{R},i}\cdot S_{\mathrm{R},i}^2\right)\right)\right).$$

Types of Confidence Interval:

The following confidence intervals are included:

1. The Normal interval with bias correction, denoted by `normal` in the function, is estimated according to Davison and Hinkley,

$$\hat{f}_{2,\mathrm{L,U}} = \hat{f}_{2,\mathrm{S}} - E_B \mp \sqrt{V_B}\cdot Z_{1-\alpha},$$

where $\hat{f}_{2,\mathrm{L,U}}$ are the lower and upper limit of the confidence interval estimated from bootstrap samples; $\hat{f}_{2,\mathrm{S}}$ denotes the estimators described above; $Z_{1-\alpha}$ represents the inverse of standard normal cumulative distribution function with type I error α; E_B and V_B are the *resampling estimates* of bias and variance calculated as

$$E_B = \frac{1}{B}\sum_{b=1}^{B}\hat{f}_{2,b}^{\star} - \hat{f}_{2,\mathrm{S}} = \bar{f}_2^{\star} - \hat{f}_{2,\mathrm{S}},$$

and

$$V_B = \frac{1}{B-1}\sum_{b=1}^{B}\left(\hat{f}_{2,b}^{\star} - \bar{f}_2^{\star}\right)^2,$$

where B is the number of bootstrap samples; $\hat{f}_{2,b}^{\star}$ is the f_2 estimate with bth bootstrap sample, and $\bar{f}_2^{\star}$ is the mean value.

2. The basic interval, denoted by `basic` in the function, is estimated according to Davison and Hinkley,

$$\hat{f}_{2,\mathrm{L}} = 2\hat{f}_{2,\mathrm{S}} - \hat{f}_{2,(B+1)(1-\alpha)}^{\star},$$

and

$$\hat{f}_{2,\mathrm{U}} = 2\hat{f}_{2,\mathrm{S}} - \hat{f}^{\star}_{2,(B+1)\alpha}\,,$$

where $\hat{f}^{\star}_{2,(B+1)\alpha}$ and $\hat{f}^{\star}_{2,(B+1)(1-\alpha)}$ are the $(B+1)\alpha$th and $(B+1)(1-\alpha)$th *ordered resampling estimates* of f_2, respectively. When $(B+1)\alpha$ is not an integer, the following equation is used for interpolation,

$$\hat{f}^{\star}_{2,(B+1)\alpha} = \hat{f}^{\star}_{2,k} + \frac{\Phi^{-1}(\alpha) - \Phi^{-1}\left(\frac{k}{B+1}\right)}{\Phi^{-1}\left(\frac{k+1}{B+1}\right) - \Phi^{-1}\left(\frac{k}{B+1}\right)}\left(\hat{f}^{\star}_{2,k+1} - \hat{f}^{\star}_{2,k}\right),$$

where k is the *integer part* of $(B+1)\alpha$, $\hat{f}^{\star}_{2,k+1}$ and $\hat{f}^{\star}_{2,k}$ are the $(k+1)$th and the kth ordered resampling estimates of f_2, respectively.

3. The percentile intervals, denoted by `percentile` in the function, are estimated using nine different types of quantiles, Type 1 to Type 9, as summarized in Hyndman and Fan's article and implemented in `R`'s `quantile` function. Using `R`'s `boot` package, program `bootf2BCA` outputs a percentile interval using the equation above for interpolation. To be able to compare the results among different programs, the same interval, denoted by `Percentile (boot)` in the function, is also included in the function.
4. The bias-corrected and accelerated (BCa) intervals are estimated according to Efron and Tibshirani,

$$\hat{f}_{2,\mathrm{L}} = \hat{f}^{\star}_{2,\alpha_1}\,,$$
$$\hat{f}_{2,\mathrm{U}} = \hat{f}^{\star}_{2,\alpha_2}\,,$$

where $\hat{f}^{\star}_{2,\alpha_1}$ and $\hat{f}^{\star}_{2,\alpha_2}$ are the $100\alpha_1$th and the $100\alpha_2$th percentile of the resampling estimates of f_2, respectively. Type I errors α_1 and α_2 are obtained as

$$\alpha_1 = \Phi\left(\hat{z}_0 + \frac{\hat{z}_0 + \hat{z}_\alpha}{1 - \hat{a}\left(\hat{z}_0 + \hat{z}_\alpha\right)}\right),$$

and

$$\alpha_2 = \Phi\left(\hat{z}_0 + \frac{\hat{z}_0 + \hat{z}_{1-\alpha}}{1 - \hat{a}\left(\hat{z}_0 + \hat{z}_{1-\alpha}\right)}\right),$$

where $\hat{z}_0$ and $\hat{a}$ are called *bias-correction* and *acceleration*, respectively.
There are different methods to estimate $\hat{z}_0$ and $\hat{a}$. Shah et al. used jackknife technique, denoted by `bca.jackknife` in the function,

$$\hat{z}_0 = \Phi^{-1}\left(\frac{N\left\{\hat{f}^{\star}_{2,b} < \hat{f}_{2,\mathrm{S}}\right\}}{B}\right),$$

and

$$\hat{a} = \frac{\sum_{i=1}^{n}\left(\hat{f}_{2,\mathrm{m}} - \hat{f}_{2,i}\right)^3}{6\left(\sum_{i=1}^{n}\left(\hat{f}_{2,\mathrm{m}} - \hat{f}_{2,i}\right)^2\right)^{3/2}}\,,$$

where $N\{\cdot\}$ denotes the number of element in the set, $\hat{f}_{2,i}$ is the ith jackknife statistic, $\hat{f}_{2,\mathrm{m}}$ is the mean of the jackknife statistics, and $\sum$ is the summation from 1 to sample size n.
Program `bootf2BCA` gives a slightly different BCa interval with `R`'s `boot` package. This approach, denoted by `bca.boot` in the function, is also implemented in the function for estimating the interval.

Notes on the argument `jackknife.type`:

For any sample with size n, the jackknife estimator is obtained by estimating the parameter for each subsample omitting the ith observation. However, when two samples (e.g., test and reference) are involved, there are several possible ways to do it. Assuming sample size of test and reference are n_T and n_R, the following three possibility are considered:

- Estimated by removing one observation from both test and reference samples. In this case, the prerequisite is $n_\mathrm{T} = n_\mathrm{R}$, denoted by `nt=nr` in the function. So if there are 12 units in test and reference data sets, there will be 12 jackknife estimators.
- Estimate the jackknife for test sample while keeping the reference data unchanged; and then estimate jackknife for reference sample while keeping the test sample unchanged. This is denoted by `nt+nr` in the function. This is the default method. So if there are 12 units in test and reference data sets, there will be $12 + 12 = 24$ jackknife estimators.
- For each observation deleted from test sample, estimate jackknife for reference sample. This is denoted by `nt*nr` in the function. So if there are 12 units in test and reference data sets, there will be $12 \times 12 = 144$ jackknife estimators.

Value

A list of 3 or 5 components.

- `boot.ci`: A *data frame* of bootstrap f_2 confidence intervals.
- `boot.f2`: A *data frame* of all individual f_2 values for all bootstrap data set.
- `boot.info`: A *data frame* with detailed information of bootstrap for reproducibility purpose, such as all arguments used in the function, time points used for calculation of f_2, and the number of `NA`s.
- `boot.summary`: A *data frame* with descriptive statistics of the bootstrap f_2.
- `boot.t` and `boot.r`: *Lists* of individual bootstrap samples for test and reference product if `sim.data.out = TRUE`.

References

Shah, V. P.; Tsong, Y.; Sathe, P.; Liu, J.-P. In Vitro Dissolution Profile Comparison—Statistics and Analysis of the Similarity Factor, f_2. *Pharmaceutical Research* 1998, **15** (6), 889–896. DOI: 10.1023/A:1011976615750.

Davison, A. C.; Hinkley, D. V. Bootstrap Methods and Their Application. Cambridge University Press, 1997.

Hyndman, R. J.; Fan, Y. Sample Quantiles in Statistical Packages. *The American Statistician* 1996, **50** (4), 361–365. DOI: /10.1080/00031305.1996.10473566.

Efron, B.; Tibshirani, R. An Introduction to the Bootstrap. Chapman & Hall, 1993.

Examples

```
# time points
tp <- c(5, 10, 15, 20, 30, 45, 60)
# model.par for reference with low variability
par.r <- list(fmax = 100, fmax.cv = 3, mdt = 15, mdt.cv = 14,
              tlag = 0, tlag.cv = 0, beta = 1.5, beta.cv = 8)
```

```
# simulate reference data
dr <- sim.dp(tp, model.par = par.r, seed = 100, plot = FALSE)
# model.par for test
par.t <- list(fmax = 100, fmax.cv = 3, mdt = 12.29, mdt.cv = 12,
              tlag = 0, tlag.cv = 0, beta = 1.727, beta.cv = 9)
# simulate test data with low variability
dt <- sim.dp(tp, model.par = par.t, seed = 100, plot = FALSE)

# bootstrap. to reduce test run time, n.boots of 100 was used in the example.
# In practice, it is recommended to use n.boots of 5000--10000.
# Set `output.to.screen = TRUE` to view the result on screen
d <- bootf2(dt$sim.disso, dr$sim.disso, n.boots = 100, print.report = FALSE)
```

calcf2 *Calculate Similarity Factor f_2*

Description

Main function to calculate f_2 according to different regulatory guidelines.

Usage

```
calcf2(test, ref, path.in, file.in, path.out, file.out,
       regulation = c("EMA", "FDA", "WHO", "Canada", "ANVISA"),
       cv.rule = TRUE, message = FALSE, min.points = 3L,
       f2.type = c("est.f2", "exp.f2", "bc.f2", "vc.exp.f2",
                   "vc.bc.f2", "all"), both.TR.85 = FALSE,
       digits = 2L, time.unit = c("min", "h"),  plot = TRUE,
       plot.start.time = 0, plot.max.unit = 24L)
```

Arguments

`test`, `ref`	*Data frames* of dissolution profiles of test and reference product if `path.in` and `file.in` are not specified; otherwise, they should be *character* strings indicating the worksheet names of the Excel file where the dissolution data is saved. See Input/Output in Details.
`path.in`, `file.in`, `path.out`, `file.out`	*Character* strings of input and output directories and file names. See Input/Output in Details.
`regulation`	*Character* strings indicating regulatory guidelines. See Regulation in Details.
`cv.rule`	*Logical.* If `TRUE`, CV rule will be checked according to regulatory guidelines. See Regulation in Details.
`message`	*Logical.* If `TRUE`, the results and messages will be printed on screen. Users are recommended to set it to `TRUE`.

`min.points`	An *integer* indicating the minimum time points to be used to calculate f_2. The default value 3 should be used for conventional f_2 calculation. This parameter is mainly used for bootstrap f_2 method. See Regulation in Details. @seealso `bootf2()`.
`f2.type`	*Character* strings indicating which f_2 estimators should be calculated. For conventional f_2 calculation, the default `"est.f2"` should be used. Other estimators are mainly for the bootstrap method. @seealso `bootf2()`.
`both.TR.85`	*Logical*. If `TRUE`, and if `regulation = "FDA"`, all measurements up to the time points at which both test and reference products dissolve more than 85% will be used to calculate f_2. This is the conventional, but incorrect, interpretation of the US FDA rule. Therefore, the argument should only be set to `TRUE` for validation purpose such as comparing the results from old literature that use the wrong interpretation to calculate f_2. See Regulation in Details.
`digits`	An *integer* indicating the decimal points for the output.
`time.unit`	*Character* strings indicating the unit of time. It should be either `"min"` for minute or `"h"` for hour. It is mainly used for checking CV rules and making plot. See Regulation in Details.
`plot`	*Logical*. If `TRUE`, a dissolution versus time plot will be printed.
`plot.start.time`	*Numeric* value indicating the starting time for the plot.
`plot.max.unit`	*Integer*. If the number of individual units is no more than this value, the mean and all individual profiles will be plotted; otherwise, individual profiles will be represented by boxplots at each time point. Therefore, to avoid overplotting, this value should not be too large. @seealso `calcf2()`.

Details

Minimum required arguments that must be provided by the user:

Arguments `test` and `ref` must be provided by the user. They should be `R data frames`, with *time as the first column*, and all individual profiles profiles as the rest columns, or mean profile as the second column if only mean profile is available. The actual names of the columns do not matter since they will be renamed internally.

Input/Output:

The dissolution data of test and reference product can either be provided as *data frames* for `test` and `ref`, as explained above, or be read from an *Excel file* with data of test and reference stored in *separate worksheets*. In the latter case, the argument `path.in`, the directory where the Excel file is, and `file.in`, the name of the Excel file *including the file extension* `.xls` *or* `.xlsx`, must be provided. In such case, the argument `test` and `ref` must be *the names of the worksheets in quotation marks*. The first column of each Excel worksheet must be time, and the rest columns are individual dissolution profiles, or the second column must be mean profile if only mean data is available. The first row should be column names, such as time, unit01, unit02, ... The actual names of the columns do not matter as they will be renamed internally.

Arguments `path.out` and `file.out` are the names of the output directory and file. It is an overkill to output such simple calculations; therefore, unless these two arguments are specified by the user, results are printed on screen by default.

Regulation:

To apply f_2 method, different regulatory guidelines have slightly different requirements. Some requirements are almost universal, such as same time points for the test and reference product, minimum 3 time points (excluding time zero), and twelve individual profiles for each formulation. Other requirements are slightly different among different regulatory guidelines, or at least interpreted differently. Two main issues are the rules for the variability (CV Rule) and time points where dissolution is more than 85% (85% Rule).

CV rule:

- `EMA`, `Canada`, and `ANVISA`: The CV of the *first time point* should not be greater than 20%, and the CV of the rest time points should not be greater than 10%.
- `WHO`: The CV should not be greater than 20% for *time points up to 10 min*, and not greater than 10% for the rest time points.
- `FDA`: US FDA is more flexible. The CV for the *early time points* should not be greater than 20%, and for the rest time points, not greater than 10%.

The phrase *the first time point* in `EMA` rule was later interpreted as all time points up to 10 min, according to an unofficial communication with an European regulator. This makes the `EMA` *rule the same as* `WHO` *rule*. For example, if there are 5 min and 10 min time points in the dissolution profiles, the CV for both 5 min and 10 min should not be greater than 20%.
The *first time point* in `ANVISA` rule corresponds to *40% of the total collected points*. For example, for a dissolution profile with five collection times, the first two collection times are considered first points.
The phrase *early time points* in `FDA` rule is typically interpreted as those points up to 15 min, sometimes even up to 20 min according to an unofficial communication with FDA staff. In the function `calcf2()`, the cutting point for FDA rule is 15 min.

85% Rule:
This rule is implemented as follows:

- `EMA`, `FDA`, `Canada`, and `ANVISA`: Only one measurement is considered after 85% of dissolution for any product.
- `WHO`: Dissolution profiles should be 'cut' at the time point where the reference release more than 85%. Therefore, `WHO` rule only differs from rule of `EMA`, `FDA`, `Canada`, and `ANVISA` when test product dissolve faster than reference. If reference product dissolve faster, then rules of all five regulatory bodies are same in this regard.

Notes on conventional FDA rule:
The exact phrase in the guidance of US FDA regarding this rule is that "*Only one measurement should be considered after 85% dissolution of both the products*." Due to the ambiguous word "both" used in the sentence, the conventional interpretation was that all measurements up to the time point at which both test and reference dissolved more than 85% should be included in the calculation of f_2. However, this is only true when both test and reference dissolve more than 85% at the same time points.
Consider the following example:

time	test	reference
5	7	10
10	15	20
15	50	55
20	69	86

30	82	90
45	84	95
60	86	97

According to conventional interpretation, all measurements up to 60 min should be included to calculate f_2 because both test and reference dissolved more than 85% only at 60 min, not at any earlier time point. However, in such case, there would be 4 measurement of reference (20, 30, 45, and 60 min) included in the calculation, which would be a direct contradictory to the phrase "Only *one measurement* should be considered after 85% ..." in the same statement in the guidance!
In an unofficial communication using this example, an FDA staff confirmed that only the first 4 time points (up to 20 min) would be used. In other words, *FDA rule in this regard is the same as EMA rule*.
The statement in `ANVISA` guideline also uses the word "ambos" (means both), which could also lead to the similar confusion. Follow the same logic as demonstrated above, it should also be interpreted as the same rule in EMA guideline.
Read vignette *Introduction to bootf2* for more details.

Value

A *data frame* of f_2 type and f_2 value, the number of time points used for the calculation (`f2.tp`), indication if both test and reference dissolve more than 85% at 15 min (`d85at15`), and other information used for the calculation.

Examples

```
tp <- c(5, 10, 15, 20, 30, 45, 60)

mod.par.t <- list(fmax = 100, fmax.cv = 2, tlag = 0, tlag.cv = 0,
                  mdt = 20, mdt.cv = 5, beta = 2.2, beta.cv = 5)

d.t <- sim.dp(tp, model.par = mod.par.t, seed = 100, n.units = 120L,
              plot = FALSE)$sim.disso

mod.par.r <- list(fmax = 100, fmax.cv = 2, tlag = 0, tlag.cv = 0,
                  mdt = 25, mdt.cv = 4, beta = 2.1, beta.cv = 3)

d.r <- sim.dp(tp, model.par = mod.par.r, seed = 100, n.units = 120L,
              plot = FALSE)$sim.disso

# set `message = TRUE` to view the compliance of the regulatory guidelines.
calcf2(d.t, d.r, plot = FALSE)
```

`helper` *Helper Functions*

Description

Helper Functions

Usage

```
bpwhisker.l(x)

bpwhisker.u(x)

ci.header(boot.info)

mod.ref(tp, ref.dp, digits = 4, model, max.disso, time.unit)

rpt.ci(f2type, btsum, boot.info)

rpt.f2(f2type, f2o, boot.info, a.jack, btsum)

rpt.concise(boot.f2.ci, boot.info, f2o, a.jack, btsum)

rpt.detailed(data.t, data.r, boot.t, boot.r, boot.f2, boot.info, f2o)

rpt.info(boot.info)

rpt.intermediate(boot.info, boot.f2)

rpt.screen(boot.f2.ci, boot.info, f2o, a.jack, btsum)
```

Arguments

`x`	Numeric vector
`boot.info`	A data frame of bootstrap information from `bootf2` function.
`tp`, `ref.dp`	Numeric vector of time points `tp` and their corresponding mean dissolution profiles `ref.dp`.
`digits`	An integer indicating the decimal points for the output.
`model`	Strings of model names. Currently only 'Weibull' and 'first-order' models are supported.
`max.disso`	Numeric value indicating the maximum dissolution.
`time.unit`	Character strings indicating the unit of time. It should be either `"min"` for minute or `"h"` for hour. It is mainly used for checking CV rules and making plot. @seealso `calcf2()`.
`f2type`	Character strings indicating the f2 type.
`btsum`	A data frame of descriptive statistics of the bootstrap data set.
`f2o`	Vector of f2 values calculated with the original data set
`a.jack`	Data frame of acceleration from `jackf2` function
`boot.f2.ci`	Matrix of f2 values from bootstrap data sets

`data.t`, `data.r`	Input data sets for test and reference.
`boot.t`, `boot.r`	List of bootstrap data sets for test and reference.
`boot.f2`	Matrix of f2 calculated from bootstrap data sets.

`shah1998` *Dissolution data from the article of Shah et al 1998*

Description

Dissolution data of reference and 5 batches of test products published in the article of Shah et al, Pharm. Res.1998, 15(6), 889–896.

Usage

```
shah1998
```

Format

A list of 6 data frames. Each data frame is a set of dissolution profiles with the format: the first column is time, the rest columns are individual profiles.

ref dissolution of reference batch

test1 dissolution of test batch 1

test2 dissolution of test batch 2

test3 dissolution of test batch 3

test4 dissolution of test batch 4

test5 dissolution of test batch 5

Source

Shah VP, Tsong Y, Sathe P, Liu JP. In vitro dissolution profile comparison–statistics and analysis of the similarity factor, f2. Pharm Res. 1998 Jun;15(6):889-96. doi: 10.1023/a:1011976615750.

sim.dp *Simulate Dissolution Profiles*

Description

Function to simulate dissolution profiles based on mathematical models or multivariate normal distribution.

Usage

```
sim.dp(tp, dp, dp.cv, model = c("Weibull", "first-order"),
       model.par = NULL, seed = NULL, n.units = 12L, product,
       max.disso = 100, ascending = FALSE, message = FALSE,
       time.unit = c("min", "h"), plot = TRUE,
       plot.max.unit = 36L)
```

Arguments

`tp`	*Numeric* vector of time points for the dissolution profiles. See Details.
`dp, dp.cv`	*Numeric* vectors of the *target mean dissolution profile* (`dp`) and its respective CV at time points `tp` (`dp.cv`). See Details.
`model`	*Character* strings of model names. Currently only `"Weibull"` and `"first-order"` models are supported.
`model.par`	A *list* with model parameters. If missing, a list of random `model.par` will be generated. See Details.
`seed`	*Integer* seed value for reproducibility. If missing, a random seed will be generated for reproducibility purpose.
`n.units`	An *integer* indicating the number of individual profiles to be simulated.
`product`	*Character* strings indicating the product name of the simulated profiles. If missing, a random name with 3 letters and 4 digits will be generated.
`max.disso`	*Numeric* value for the maximum possible dissolution. In theory, the maximum dissolution is 100%, but in practice, it is not uncommon to see higher values, such as 105%, or much lower values, especially for products with low solubility.
`ascending`	*Logical*. If `TRUE`, simulated profiles will always increase with time. Only applicable when the approach based on multivariate normal distribution is used. See Details.
`message`	*Logical*. If `TRUE`, basic information of the simulation will be printed on screen.
`time.unit`	*Character* strings indicating the unit of time. It should be either `"min"` for minute or `"h"` for hour. It is mainly used for and making plot and generating `model.par` and `dp.cv` when they are not provided by the user. @seealso `calcf2()`.
`plot`	*Logical*. If `TRUE`, a a dissolution versus time plot will be included in the output.
`plot.max.unit`	*Integer*. If the number of individual units is no more than this value, the mean and all individual profiles will be plotted; otherwise, individual profiles will be represented by boxplots at each time point. Therefore, to avoid overplotting, this value should not be too large. @seealso `calcf2()`.

Details

Simulation approaches:

The approach used to simulate individual dissolution profiles depends on if the *target mean dissolution profile* `dp` is provided by the user or not.

- If `dp` is not provided, then it will be calculated using `tp`, `model`, and `model.par`. The parameters defined by `model.par` are considered as the *population parameter*; consequently, the calculated `dp` is considered as the *targeted population profile*. In addition, `n.units` sets of *individual model parameters* will be simulated based on exponential error model, and *individual dissolution profiles* will be generated using those individual parameters. The mean of all simulated individual profiles, `sim.mean`, included in one of the outputs data frames, `sim.summary`, will be *similar, but not identical, to* `dp`. The difference between `sim.mean` and `dp` is determined by the number of units and the variability of the model parameters. In general, the larger the number of units and the lower of the variability, the smaller the difference. Additional details on the mathematical models are given below.
- If `dp` is provided, then `n.units` of individual dissolution profiles will be simulated using multivariate normal distribution. The mean of all simulated individual profiles, `sim.mean`, will be *identical to* `dp`. In such case, it is recommended that `dp.cv`, the CV at time points `tp`, should also be provided by the user. If `dp.cv` is not provided, it will be generated automatically such that the CV is between 10% and 20% for time points up to 10 min, between 1% and 3% for time points where the dissolution is more than 95%, between 3% and 5% for time points where the dissolution is between 90% and 95%, and between 5% and 10% for the rest of time points. Whether the `dp.cv` is provided or generated automatically, the CV calculated using all individual profiles will be equal to `dp.cv`. Additional details on this approach are given below.

Minimum required arguments that must be provided by the user:

If `dp` is provided by the user, logically, `tp` must also be provided, and the approach based on multivariate normal distribution is used, as explained above. If `dp` is not provided, `tp` could be missing, i.e., a simple function call such as `sim.dp()` will simulate dissolution profiles. In such case, a default `tp` will be generated depending on the `time.unit`: if the `time.unit` is `"min"`, then `tp` would be `c(5, 10, 15, 20, 30, 45, 60)`; otherwise the `tp` would be `c(1, 2, 3, 4, 6, 8, 10, 12)`. The rest arguments are either optional, or required by the function but default values will be used.

Notes on mathematical models:

The first-order model is expressed as

$$f_t = f_{\max}\left(1 - e^{-k(t - t_{\text{lag}})}\right),$$

and the Weibull model was expressed either as

$$f_t = f_{\max}\left(1 - e^{-\left(\frac{t - t_{\text{lag}}}{\text{MDT}}\right)^{\beta}}\right)$$

or

$$f_t = f_{\max}\left(1 - e^{-\frac{(t - t_{\text{lag}})^{\beta}}{\alpha}}\right)$$

where $f_{\max}$ is the maximum dissolution, MDT is the mean dissolution time, t_{lag} is the lag time, α and β are the scale and shape factor in Weibull function, and k is the rate constant in the first-order model. Obviously, The two Weibull models are mathematically equivalent by letting $\alpha = \mathrm{MDT}^{\beta}$.

Individual model parameter were simulated according to the exponential error model

$$P_i = P_\mu e^{N\left(0,\sigma^2\right)}$$

where P_i and P_μ denote the individual and population model parameters; $N\left(0,\sigma^2\right)$ represents the normal distribution with mean zero and variance σ^2 ($\sigma = \mathrm{CV}/100$).

How to supply `model.par`:

The argument `model.par` should be supplied as a *named list* of model parameters as explained above, and their respective CV for simulation of individual parameters. Therefore, for the first-order model, three pairs of parameters should be specified: `fmax/fmax.cv`, `k/k.cv`, and `tlag/tlag.cv`; and for Weibull model, four pairs: `fmax/fmax.cv`, `tlag/tlag.cv`, `beta/beta.cv`, and either `alpha/alpha.cv` or `mdt/mdt.cv`, depending on the mathematical formula used. CV should be given in percentage, e.g., if CV for `beta` is 30%, it should be specified as `beta.cv = 30`, *not* `beta.cv = 0.3`. The order of the parameters does not matter, but the name of the parameters must be given *exactly same* as described above. For example:

- `model.par = list(fmax = 100, fmax.cv = 5, k = 0.6, k.cv = 25, tlag = 0, tlag.cv = 0)` for the first-order model.
- `model.par = list(fmax = 100, fmax.cv = 5, tlag = 5, tlag.cv = 10, mdt = 15, mdt.cv = 20, beta = 1.5, beta.cv = 5)`, or
- `model.par = list(fmax = 100, fmax.cv = 5, tlag = 5, tlag.cv = 10, alpha = 60, alpha.cv = 20, beta = 1.5, beta.cv = 5)` for Weibull models.

Notes on multivariate normal distribution approach:

When this approach is used, depending on `dp/dp.cv`, there is no guarantee that all individual profiles increase with time; near the end of the time points, some individual profiles may decrease, especially when the dissolution is close to the plateau and the variability is high. This can happen to real life data, especially for those products with drug substances that are unstable in dissolution medium. To force increase for all profiles, set `ascending = TRUE`. Depending on the data, the program may take long time to run, or may even fail.

Value

A list of 3 to 5 components:

- `sim.summary`: A *data frame* with summary statistics of all individual dissolution profiles.
- `sim.disso`: A *data frame* with all individual dissolution profiles.
- `sim.info`: A *data frame* with information of the simulation such as the seed number and number of individual profiles. If modelling approach is used, the data frame will contain model parameters as well.
- `model.par.ind`: A *data frame* of all individual model parameters that were used for the simulation of individual dissolution profiles. Available only if the modelling approach is used, i.e., when `dp` is missing.
- `sim.dp`: A plot. Available only if the argument `plot` is `TRUE`.

Examples

```
# time points
tp <- c(5, 10, 15, 20, 30, 45, 60)

# using all default values to simulate profiles
d1 <- sim.dp(tp, plot = FALSE)

# summary statistics
d1$sim.summary

# individual profiles
d1$sim.disso

# simulation information
d1$sim.info

#individual model parameters
d1$mod.par.ind

# using Weibull model to simulate 100 profiles without lag time
mod.par <- list(fmax = 100, fmax.cv = 2, tlag = 0, tlag.cv = 0,
                mdt = 20, mdt.cv = 5, beta = 2.2, beta.cv = 5)
d2 <- sim.dp(tp, model.par = mod.par, seed = 100, n.units = 100L,
             plot = FALSE)

# using multivariate normal distribution approach
# target mean profile with same length as tp
dp <- c(39, 56, 67, 74, 83, 90, 94)

# CV% at each time point
dp.cv <- c(19, 15, 10, 8, 8, 5, 3)

# simulation
d3 <- sim.dp(tp, dp = dp, dp.cv = dp.cv, seed = 1234, plot = FALSE)
```

`sim.dp.byf2` *Simulate Dissolution Profiles by f_2 Values*

Description

Given any mean dissolution profile dp, this function will simulate a mean dissolution profile such that when the newly simulated profile is compared to dp, the calculated f_2 will be equal to the predefined target f_2 value.

Usage

```
sim.dp.byf2(tp, dp, target.f2, seed = NULL, min.points = 3L,
            regulation = c("EMA", "FDA", "WHO", "Canada", "ANVISA"),
            model = c("Weibull", "first-order"), digits = 2L,
            max.disso = 100, message = FALSE, both.TR.85 = FALSE,
            time.unit = c("min", "h"), plot = TRUE, sim.dp.out,
            sim.target = c("ref.pop", "ref.median", "ref.mean"),
            model.par.cv = 50, fix.fmax.cv = 0, random.factor = 3)
```

Arguments

`tp`, `dp`	*Numeric* vector of time points `tp` and the mean dissolution profiles `dp`.
`target.f2`	*Numeric* value of target f_2. It can be a *single value*, or a *vector of two values* that represent the lower and upper limit of target f_2 value. See Details.
`seed`	*Integer* seed value for reproducibility. If missing, a random seed will be generated for reproducibility purpose.
`min.points`	An *integer* indicating the minimum time points to be used to calculate f_2. The default value 3 should be used for conventional f_2 calculation. See Details. @seealso `calcf2()`.
`regulation`	*Character* strings indicating regulatory guidelines. @seealso `calcf2()` for the discussion of those guidelines.
`model`	*Character* strings of model names. Currently only `"Weibull"` and `"first-order"` models are supported. @seealso `sim.dp()` for the description of models.
`digits`	An *integer* indicating the decimal points for the output.
`max.disso`	*Numeric* value for the maximum possible dissolution. In theory, the maximum dissolution is 100%, but in practice, it is not uncommon to see higher values, such as 105%, or much lower values, especially for products with low solubility.
`message`	*Logical*. If `TRUE`, basic information of the simulation will be printed on screen.
`both.TR.85`	*Logical*. If `TRUE`, and if `regulation = "FDA"`, all measurements up to the time points at which both test and reference products dissolve more than 85% will be used to calculate f_2. This is the conventional, but incorrect, interpretation of the US FDA rule. Therefore, the argument should only be set to `TRUE` for validation purpose such as comparing the results from old literature that use the wrong interpretation to calculate f_2. @seealso `calcf2()` for detailed discussion.
`time.unit`	*Character* strings indicating the unit of time. It should be either `"min"` for minute or `"h"` for hour. It is mainly used for checking CV rules and making plot. See Regulation in Details. @seealso `calcf2()`.
`plot`	*Logical*. If `TRUE`, a a dissolution versus time plot will be included in the output.
`sim.dp.out`	Output of function `sim.dp()`. If this argument is supplied by the user, then `tp/dp`, `regulation`, `model`, `max.disso`, and `time.unit` will be ignored, if they are provided by the user, since all those arguments are included in `sim.dp.out`.
`sim.target`	*Character* strings indicating to which target profile should the newly simulated profile be compared for the calculation of f_2. This argument is only applicable when `sim.dp.out` is provided. See Details.
`model.par.cv`, `fix.fmax.cv`	*Numeric* values expressed as percentages used for random generation of model parameters and fmax when optimization algorithm is not used, i.e., when `target.f2` is a vector of two numbers. See Details.

`random.factor`	*Numeric* value used for random generation of model parameters when optimization algorithm is used, i.e., when `target.f2` is a single number. See Details.

Details

The main principle of the function is as follows:

1. For any given mean dissolution profile `dp`, fit a suitable mathematical model and obtain model parameters.
    - No precise fitting is required since those parameters will be served as *initial value* for further model fitting.
    - If `sim.dp.out`, the output of the function `sim.dp()`, is available, no initial fitting is necessary as model parameters can be read directly from the output, unless multivariate normal distribution approach was used in `sim.dp()`. In such case, initial model fitting will be done.
2. Find a suitable model parameters and simulate a new dissolution profile, comparing the new profile to the provided reference profile `dp` by calculating f_2. If the the obtained f_2 is *equal to*, or *within the lower and upper limit of*, the `target.f2`, then the function will output the obtained model parameters and the new profile.

There are two approaches used to find the suitable model parameters:

- If `target.f2` is a single value, optimization algorithm will be used and the newly simulated dissolution profile will have f_2 equal to `target.f2` when compared to `dp` (within numeric precision defined by the tolerance).
- If `target.f2` is a vector of two numbers representing the lower and upper limit of target f_2 value, such as `target.f2 = c(lower, upper)`, then dissolution will be obtained by random searching and the calculated f_2 will be within the range of lower and upper.

For example, you can set `target.f2 = c(54.95, 55.04)` to have target f_2 of 55. Since f_2 should be normally reported without decimal, in practice, this precision is enough. You might be able to do with `c(54.99, 55.01)` if you really need more precision. However, the narrower the range, the longer it takes the function to run. With narrow range such as `c(54.999, 55.001)`, it is likely the program will fail. In such case, provide single value to use optimization algorithm.

Arguments `model.par.cv`, `fix.fmax.cv`, and `random.factor` are certain numeric values used to better control the random generation of model parameters. The default values should work in most scenarios. Those values should be changed only when the function failed to return any value. Read vignette of the function (`vignette("sim.dp.byf2", package = "bootf2")`) for more details.

The data frame `sim.summary` in `sim.dp.out`, the output of function `sim.dp()`, contains `dp`, the population profile, and `sim.mean` and `sim.median`, the mean and median profiles calculated with `n.units` of simulated individual profiles. All these three profiles could be used as the target profile that the newly simulated profile will be compare to. Argument `sim.target` defines which of the three will be used: `ref.pop`, `ref.mean`, and `ref.median` correspond to `dp`, `sim.mean` and `sim.median`, respectively.

Value

A *list* of 2 components: a *data frame of model parameters* and a *data frame of mean dissolution profile* generated using the said model parameters. The output can be passed to function `sim.dp()` to further simulate multiple individual profiles.

Examples

```
tp <- c(5, 10, 15, 20, 30, 45, 60)

mod.par.r <- list(fmax = 100, fmax.cv = 2, tlag = 0, tlag.cv = 0,
                  mdt = 25, mdt.cv = 4, beta = 2.1, beta.cv = 3)

d.r <- sim.dp(tp, model.par = mod.par.r, seed = 100, n.units = 120L,
              plot = FALSE)

model.par1 <- sim.dp.byf2(sim.dp.out = d.r, target.f2 = 60, seed = 123)
model.par2 <- sim.dp.byf2(sim.dp.out = d.r, target.f2 = c(59.95, 60.04),
                          seed = 123)
```

Index