Package: kwbGompitz (via r-universe)

September 11, 2024

Title Interface to GompitZ Tool for Modelling the Degradation of Sewer Pipelines

Version 0.8.0

Description Functions enabling the writing of GompitZ input files, running of GompitZ Tools (gompcal.exe, gompred.exe) and reading of GompitZ output files.

License MIT + file LICENSE

URL https://github.com/kwb-r/kwbGompitz

BugReports https://github.com/kwb-r/kwbGompitz/issues>

Imports data.table, ggplot2, kwb.plot, kwb.utils, manipulate, Rcpp

Suggests gridExtra, knitr, microbenchmark, rmarkdown, testthat

LinkingTo Rcpp

VignetteBuilder knitr

Remotes github::kwb-r/kwb.plot, github::kwb-r/kwb.utils

Encoding UTF-8

RoxygenNote 7.2.1

Repository https://kwb-r.r-universe.dev

RemoteUrl https://github.com/KWB-R/kwbGompitz

RemoteRef HEAD

RemoteSha dac7a197fc4ce5cee7a6feb2b2b5db3009261f30

Contents

`.calibrationAvailable` *Check for Convergence in Calibration*

Description

Check for Convergence in Calibration

Usage

```
.calibrationAvailable(calibration, strata = NULL, param = NULL)
```

Arguments

`calibration` result of Gompitz calibration as provided by `runGompitzCalibration`

`strata` vector of strata for which convergence is checked

`param` if not NULL, this structure, representing the content of the "param.txt" file, is used instead of `calibration` to check for convergence

Value

named logical vector with as many elements as there are in *strata* each of which indicates if convergence was achieved for the corresponding stratum

`.fileContentStrategy0` *Content for Strategy File 0*

Description

Content for Strategy File 0

Usage

```
.fileContentStrategy0(range.years, condition.labels)
```

Arguments

`range.years` vector of two integer numbers: first and last year of simulation

`condition.labels`
vector of condition labels

`.fileContentStrategy1` *Content for Strategy File 1*

Description

Content for Strategy File 1

Usage

```
.fileContentStrategy1(rehabilitation.costs, annual.total.length)
```

Arguments

`rehabilitation.costs`
: list of rehabilitation costs per condition. The names of the list elements are the condition labels and the values of the list elements are the corresponding rehabilitation costs. Example: list(C4 = 200, C3 = 300, C2 = 400, C1 = 500)

`annual.total.length`
: list of annual total lengths to be rehabilitated. The names of the list elements are the year numbers and the values of the list elements are the lenghts to be rehabilitated in the corresponding year. Example: list("2005" = 100, "2006" = 110, "2007" = 120)

`.fileContentStrategy1or2` *File Content for Strategy 1 or 2*

Description

File Content for Strategy 1 or 2

Usage

```
.fileContentStrategy1or2(rehabilitation.costs, annual.total.length.or.budget)
```

Arguments

`rehabilitation.costs`
: list of rehabilitation costs per condition. The names of the list elements are the condition labels and the values of the list elements are the corresponding rehabilitation costs. Example: list(C4=200, C3=300, C2=400, C1=500)

`annual.total.length.or.budget`
: list of annual total lengths to be rehabilitated or annual total budgets.

`.fileContentStrategy2` *Content for Strategy File 2*

Description

Content for Strategy File 2

Usage

```
.fileContentStrategy2(rehabilitation.costs, annual.total.budget)
```

Arguments

`rehabilitation.costs`
: list of rehabilitation costs per condition. The names of the list elements are the condition labels and the values of the list elements are the corresponding rehabilitation costs. Example: list(C4=200, C3=300, C2=400, C1=500)

`annual.total.budget`
: list of annual total budget of rehabilitation operations. The names of the list elements are the year numbers and the values of the list elements are the annual total budgets for the corresponding year. Example: list("2005"=100000, "2006"=110000, "2007"=120000)

`.fileContentStrategy3` *Content for Strategy File 3*

Description

Content for Strategy File 3

Usage

```
.fileContentStrategy3(
  condition.labels,
  rehabilitation.costs,
  max.tol.prop.of.length,
  range.years,
  target.year
)
```

Arguments

`condition.labels`
: vector of condition labels Example: c("C4", "C3", "C2", "C1")

`rehabilitation.costs`
: vector of rehabilitation costs per condition in the order of the corresponding condition labels in condition.labels Example: c(200, 300, 400, 500)

`max.tol.prop.of.length`
: vector of maximum tolerated proportion of network length in each condition in the order of the corresponding condition labels in condition.labels. Example: c(1.0, 0.7, 0.4, 0.05)

`range.years`
: two element vector containing the first and last year of simulation

`target.year`
: year at which the proportions of the network length in each condition must have been brought just below their maximum tolerated value

`.getObservationByCondition` *Number of observations*

Description

Extract number of observations from calibration result

Usage

```
.getObservationByCondition(calibration)
```

Arguments

`calibration`	calibration result as retrieved by `runGompitzCalibration`

`.orderByWeightedProbabilities` *probability order 2*

Description

order by weighted probabilities

Usage

```
.orderByWeightedProbabilities(
  probabilities,
  weight = seq_len(ncol(probabilities))
)
```

Arguments

`probabilities`	matrix of probabilities
`weight`	vector of weights with as many elements as there are columns in probabilities

.runModuleInDirectory *.runModuleInDirectory*

Description

.runModuleInDirectory

Usage

```
.runModuleInDirectory(
  module,
  target.dir,
  input.file,
  sep,
  ...,
  verbose = 1,
  show.error = TRUE
)
```

Arguments

`module`	module
`target.dir`	target.dir
`input.file`	input.file
`sep`	sep
`...`	additional arguments passed to kwb.utils::runInDirectory
`verbose`	verbose (default: 1)
`show.error`	show.error (default: TRUE)

Value

???

addAssignment *Append assignment "a=b"*

Description

Append assignment "a=b"

Usage

```
addAssignment(main = "", a, b)
```

Arguments

`main`	current assignment string
`a`	key of the assignment
`b`	value of the assignment

Value

`main` with `a` = `b` appended with a comma as separator if `main` is not empty

`catHeader` *Print an underlined Header*

Description

Print an underlined Header

Usage

```
catHeader(x, char)
```

Arguments

`x`	caption
`char`	character used for the underline

`catStructure` *Print the Structure of an Object*

Description

Print the Structure of an Object

Usage

```
catStructure(x, max.level = NA)
```

Arguments

`x`	any R object
`max.level`	passed to `capture.output`

Value

print the structure of an object

cbindDataFrames *Bind Columns from a list of Data Frames*

Description

Bind Columns from a list of Data Frames

Usage

```
cbindDataFrames(dataFrames, keyindex = 1:2)
```

Arguments

`dataFrames`	list of data frames
`keyindex`	indices of (key) columns to be excluded

checkConvergence *Warn if Model did not converge for all Strata*

Description

Warn if the model did not converge for all strata

Usage

```
checkConvergence(calibration, do.warn = TRUE)
```

Arguments

`calibration`	calibration object as returned by `kwbGompitz:::readCalibration`
`do.warn`	if `TRUE` warnings are shown for non-calibrated strata

Value

named logical vector with element names corresponding to the names of the elements of `calibration` that represent the different strata.

checkInputData *Check Input Data*

Description

Checks if input data is defined properly

Usage

```
checkInputData(input.data)
```

Arguments

`input.data`	input.data

Value

error in case input data was not defined properly

columnwise *Apply a Function for each Column*

Description

Apply a Function for each Column

Usage

```
columnwise(x, FUN, ...)
```

Arguments

`x`	two dimensional object
`FUN`	function to be called for each column
`...`	arguments passed to `FUN`

compareEstimates *Compare Estimates in calib.txt and param.txt*

Description

Compare Estimates in calib.txt and param.txt

Usage

```
compareEstimates(calibration, parameters, digits, warn = FALSE)
```

Arguments

`calibration`	list structure as returned by `kwbGompitz:::readCalibration`
`parameters`	list structure as returned by `kwbGompitz:::readParameters`
`digits`	round the estimates to this number of significant (not decimal!) digits before comparing
`warn`	if `TRUE` (the default is `FALSE`) a warning is given if the strata read from `calibr.txt` do not correspond to the strata read from `param.txt` (containing only the successfully calibrated strata!?).

Value

number of warnings that occurred

composeGompitzInputData *Compose Input Data for Gompitz Functions*

Description

Compose input data for Gompitz functions `runGompitzCalibration`, `runGompitzPrediction`

Usage

```
composeGompitzInputData(
  masterdata,
  covariates,
  weight,
  covariates.status,
  condition.labels = NULL,
  warn = FALSE
)
```

Arguments

`masterdata` data.frame containing master data as retrieved by `composeMasterData`

`covariates` data.frame containing covariates in columns. Must have as many rows as masterdata

`weight` weight to be given for each inspection (each row) in *masterdata*. Should be a vector of same length as there are rows in *masterdata*. Default: 1

`covariates.status`
matrix of covariate status as retrieved by `createStatusMatrix`

`condition.labels`
All possible condition labels, e.g. c("1", "2", "3", "4"). Default: unique values in column "condition" of *masterdata*

`warn` if TRUE (default), a warning is given if *weight* does not have the expected length

Value

list with elements *masterdata* (data.frame), *covariates* (data.frame), *weight* (numeric vector), *covariates.status* (numeric matrix) and *condition.labels* (character vector)

`composeMasterData` *Prepare "Master Data" for Gompitz Functions*

Description

Prepare "master data" (stratum, pipe-ID, year of installation, year of inspection, condition class) for Gompitz functions

Usage

```
composeMasterData(stratum, pipeid, instyear, inspyear, condition)
```

Arguments

`stratum` Stratum alphanumerical label

`pipeid` Pipeline Identifier alphanumerical label

`instyear` Installation year (integer)

`inspyear` Inspection Year (integer - void field if not inspected)

`condition` Condition Class alphanumerical label

Value

data.frame with columns *stratum*, *pipeid*, *instyear*, *inspyear*, *condition*,

See Also

`composeGompitzInputData`

copyParameters *Copy Model Parameters from one Stratum to another*

Description

Copy Model Parameters from one Stratum to another

Usage

```
copyParameters(calibration, from = NULL, to = NULL, dbg = TRUE)
```

Arguments

calibration	calibration object as returned by runGompitzCalibration
from	name of stratum to copy parameters from
to	vector of names of strata to which parameters are to be copied to. If not given, the parameters of from are copied to all strata for which model parameters did not converge.
dbg	if TRUE (default) debug messages are shown

Examples

```
## Not run:
# Get an example calibration
calibration <- kwbGompitz::exampleCalibration()

# Check for which strata the model parameters converged
checkConvergence(calibration, do.warn = FALSE)

# Copy parameters from a stratum for which the model parameters converged
# to a stratum for which the model parameters did not converge
calibration <- copyParameters(calibration, from = "Cast Iron", to = "Concrete")

# Check again
checkConvergence(calibration, do.warn = FALSE)

# The following gives a warning (no convergence for source stratum) and returns
# the calibration unchanged
calibration <- copyParameters(calibration, "Clay", "Brick")

# The following gives a warning (differing distinct condition classes) and
# returns the calibration unchanged
calibration <- copyParameters(calibration, "Cast Iron", "Brick")

## End(Not run)
```

createExampleFiles *Create Example Files*

Description

Create Example Files Using Different Windows Executables. Run different executables of gompcal and gompred (as provided in subfolders "bin_<version>" with version in ["win32", "win32_kwb", "unix_home"]) on the example input file (obs.txt) and store the results in folders "example_<version>".

Usage

```
createExampleFiles()
```

createExampleFilesSmall *Create smaller versions of the example files*

Description

Create smaller versions of the example files

Usage

```
createExampleFilesSmall(
  parts = c(2, 4, 8, 16),
 version = c("unix", "unix_home", "win32", "win32_kwb")[ifelse(kwb.utils::.OStype() ==
    "unix", 1, 3)],
  types = c("obs", "predict0", "predict1", "predict2", "iff"),
  targetdir = NULL
)
```

Arguments

parts	vector of integer determining the parts of the files to be created: 2 = first half, 3 = first third, ..., 10 = first 10 percent, etc.
version	one of c("unix", "unix_home", "win32", "win32_kwb")
types	vector of file types to be created. See the default assignment for possible items
targetdir	full path to the target directory in which to put the files. By default the files go into the package's example directory related to the selected version.

Examples

```
## Not run: kwbGompitz:::createExampleFilesSmall(c(2, 4, 10))
```

`createStatusMatrix` *Default Covariate Status Matrix*

Description

Default covariate status matrix

Usage

```
createStatusMatrix(
  strata = NULL,
  covariateNames = NULL,
  default.other = 3,
  default.length = 0,
  covariates = "Length_num",
  Data = NULL,
  column.strata = "Material_cat",
  column.length = NULL,
  note = TRUE
)
```

Arguments

`strata` names of strata (e.g. unique values in column *stratum* of masterdata)

`covariateNames` names of covariates

`default.other` default status value to be used in any cell of the matrix, except the cells in column *LENGTH*. Default: 3

`default.length` status value to be used in column *LENGTH*. Default: 0

`covariates` names of columns in `Data` that shall be used as covariates

`Data` data frame with one row per inspection

`column.strata` name of column in `Data` containing the stratifying variable. Default: "Material_cat", i.e. strata are built by material

`column.length` name of column in `Data` containing the pipe lengths

`note` if `TRUE` (default) a note about the length column is given

`default_condition_colours` *Get default Vector of Condition Colours*

Description

Get default Vector of Condition Colours

Usage

```
default_condition_colours(condition_labels)
```

Arguments

`condition_labels`	vector of condition class names ordered from the best to the worst condition

Value

vector of colour codes with the conditions as element names

`demo_survival` *Interactive Plot of Survival Curves*

Description

Interactive Plot of Survival Curves

Usage

```
demo_survival(
  t = 1:100,
  alpha.0 = -1.15,
  bz0.0 = -0.88,
  bz1.0 = -2.77,
  theme = ggplot2::theme_bw()
)
```

Arguments

`t`	vector of times at which to calculate the survival curves
`alpha.0`	initial value of the `alpha` parameter
`bz0.0`	initial value of the `bz0` parameter
`bz1.0`	initial value of the `bz1` parameter
`theme`	ggplot2-theme applied to the plots

exampleCalibration *Calibration Object of the Gompitz Example*

Description

This function returns a calibration object (as returned by `runGompitzCalibration`), corresponding to the example provided with the Gompitz software package

Usage

```
exampleCalibration(VERSION = getOperatingSystemType())
```

Arguments

`VERSION`	one of "unix", "unix_home", "win32", "win32_kwb"

exampleFile *Get Path to Example File*

Description

Get the full path to one of the example files provided with the GompitZ software package of which copies are available in this R package.

Usage

```
exampleFile(filename = "none", ..., dbg = TRUE)
```

Arguments

`filename`	name of the example file
`...`	passed to `getDefaultPaths`
`dbg`	if `TRUE`, debug messages are shown, else not.

Value

full path to the example file

`getCalibration` *Get a full Calibration Object from File*

Description

Get a full calibration object as returned by `runGompitzCalibration`, but not by calling this function but by just reading the file `calibr.txt`

Usage

```
getCalibration(file_calib, file_param = NULL, digits = 3)
```

Arguments

`file_calib`	full path to calibration file `calibr.txt`
`file_param`	full path to file `param.txt`
`digits`	passed to `kwbGompitz:::compareEstimates`

`getCalibrationFile` *Source File of Calibration Result*

Description

Source File of Calibration Result

Usage

```
getCalibrationFile(calib.result)
```

Arguments

`calib.result`	calibration result of Gompitz calibration as provided by `runGompitzCalibration`

getCalibrationParameters *Get parameters from Calibration Object*

Description

Get the calibration parameters as they are required to generate the param.txt file for the gompred program, from the calibration object

Usage

```
getCalibrationParameters(
  calibration = exampleCalibration(),
  remove_non_calibrated = TRUE
)
```

Arguments

`calibration`	calibration object as returned by `runGompitzCalibration`
`remove_non_calibrated`	if `TRUE` (default) parameters for non-calibrated strata are removed

getConstants *Gompitz Covariate Status Constants*

Description

Gompitz constants regarding the status of covariates (not used, influencing the initial state, influencint the deterioration speed, influencing both)

Usage

```
getConstants()
```

Details

@export

`getDefaultPaths` *Default Paths to GompitZ Files*

Description

Default paths to GompitZ executable and example files

Usage

```
getDefaultPaths(VERSION = getOperatingSystemType(), ...)
```

Arguments

`VERSION`	one of "unix", "win32", "win32_kwb"
`...`	arguments passed to resolve

`getExampleStatusMatrix` *Example Covariate Status Matrix*

Description

Example Covariate Status Matrix

Usage

```
getExampleStatusMatrix()
```

See Also

`createStatusMatrix`

getFileContentForInputFile *Get file Content for Input File*

Description

Get file Content for Input File

Usage

```
getFileContentForInputFile(
  masterdata,
  covariates,
  covariates.status,
  condition.labels,
  weight,
  sep,
  file = NULL
)
```

Arguments

masterdata	masterdata as returned by composeMasterData
covariates	data.frame with as many rows as there are in masterdata and as many columns as there are covariates, containing the covariate values
covariates.status	matrix of covariate status as retrieved by createStatusMatrix
condition.labels	condition labels
weight	Weight of the pipeline for the calibration
sep	column separator
file	if a path to a file is given here, the content will be written to the file instead of returned by this function

getSampleObservations *Random Observation Data*

Description

Random Observation Data

Usage

```
getSampleObservations(n = 100)
```

Arguments

`n`	number of rows to be generated

Value

data frame with columns MATERIAL, Baujahr, DN, Pipe, Note, Inspection, PipeLength, before1970

`getStataLabels` *Read Strata Labels from Observation File*

Description

Read Strata Labels from Observation File

Usage

```
getStataLabels(input.file, sep)
```

Arguments

`input.file`	path to input fule
`sep`	column separator

`get_label` *Provide Label for Plot in selected Language*

Description

Provide Label for Plot in selected Language

Usage

```
get_label(element = NULL, lng = "en")
```

Arguments

`element`	one of "title", "ylab", "ylab.rel", "ylab.dev", "title.legend"
`lng`	language acronym, "de" for German or "en" for English

Value

???

get_or_create_target_dir *Create target directory if required and return the path to it*

Description

Create target directory if required and return the path to it

Usage

```
get_or_create_target_dir(version)
```

Arguments

`version`	one of `c("unix", "unix_home", "win32", "win32_kwb")`

get_survivals *Get Survival Curves from a Calibration Object*

Description

Get Survival Curves from a Calibration Object

Usage

```
get_survivals(calibration, stratum, t = 1:100, matrix = TRUE, ...)
```

Arguments

`calibration`	calibration object as returned by `runGompitzCalibration`
`stratum`	name of stratum for which to return the survival curves
`t`	vector of times for which to calculate the survival probabilities
`matrix`	if `TRUE` (default) the result is returned as a matrix, otherwise as a list.
`...`	further arguments passed to `survivals`

See Also

`survivals`

`gg_stacked_bars` *Generate gg-plot of stacked Bars*

Description

Generate gg-plot of stacked Bars

Usage

```
gg_stacked_bars(
  data,
  relative = TRUE,
  labels = FALSE,
  legend = c("bottom", "left", "top", "right", "none")[5],
  reverse = TRUE,
  lng = "en",
  colours = NULL
)
```

Arguments

`data` data frame with columns `length.abs` (absolute pipe length in m), `length.rel` (relative pipe length in percent), `condition`

`relative` logical indicating whether to show absolute or relative lenghts

`labels` logical indicating whether to show labels (axes, title) are shown.

`legend` legend position: one of `"bottom"`, `"left"`, `"top"`, `"right"`,`"none"`

`reverse` logical indicating whether to stack the bars in reversed order

`lng` language acronym, "de" for German or "en" for English, that is passed to `kwbGompitz:::get_label`

`colours` named vector that maps the values in `data$condition` to colour names.

`inputFileBody` *Calibration Input File Body*

Description

Calibration Input File Body

Usage

```
inputFileBody(masterdata, covariates, weight, sep)
```

Arguments

`masterdata`	masterdata as returned by `composeMasterData`
`covariates`	data.frame with as many rows as there are in masterdata and as many columns as there are covariates, containing the covariate values
`weight`	Weight of the pipeline for the calibration
`sep`	column separator

See Also

`kwbGompitz:::inputFileHeader`

`inputFileHeader` *Calibration Input File Header*

Description

Calibration Input File Header

Usage

```
inputFileHeader(condition.labels, covariates, covariates.status, sep = ";")
```

Arguments

`condition.labels`	condition labels
`covariates`	data.frame containing covariates. Needed to determine if covariates are numeric or categorical
`covariates.status`	matrix of covariate status as retrieved by `createStatusMatrix`
`sep`	column separator

See Also

`kwbGompitz:::inputFileBody`

interpolate_between *Interpolate equidistantly between End Points*

Description

Interpolate equidistantly between End Points

Usage

```
interpolate_between(y1, y2, n = 2L, version = 3)
```

Arguments

y1	numeric vector of y-values at the beginning
y2	numeric vector of y-values at the end. Must be as long as y1
n	number of interpolation points including first and last value
version	version of implementation

Value

matrix M with n columns and as many rows as there are values in y1 (and also in y1). The first column contains the values from y1, the last column contains the values from y2 and the n - 2 columns in between contain the interpolated values.

Examples

```
y1 <- c(1, 1, 1)
y2 <- c(0.1, 0.5, 0.8)
n <- 10

y <- interpolate_between(y1, y2, n)
barplot(y, beside = TRUE)

# Compare the performance of slightly different implementations
microbenchmark::microbenchmark(
  v1 = interpolate_between(y1, y2, n, version = 1),
  v2 = interpolate_between(y1, y2, n, version = 2),
  v3 = interpolate_between(y1, y2, n, version = 3),
  times = 1000,
  check = function(x) kwb.utils::allAreIdentical(x[2:3]) &&
    all(kwb.utils::almostEqual(x[[1]], x[[2]]))
)
```

`marginal_survival` *Condition state marginal survival function surv(j,t,theta,Z)*

Description

alpha = theta[cond] Integr PY(t) <= j | theta, Z, IFF . phi(IFF) . dIFF Gauss-Legendre Integration on [a,b]

Usage

```
marginal_survival(t, alpha, bz0, bz1, s, version = 1)
```

Arguments

`t`	time
`alpha`	parameter "alpha"
`bz0`	parameter "bz0"
`bz1`	parameter "bz1"
`s`	standard deviation?
`version`	1, 2, or 3

`matrixToLongDataFrame` *Convert Data in wide View to long View*

Description

Convert Data in wide View to long View

Usage

```
matrixToLongDataFrame(x, columnName = deparse(substitute(x)))
```

Arguments

`x`	data frame or list of data frames
`columnName`	column name in result data frame

multicall *Call a Function with all Combinations of Argument Ranges*

Description

Call a function with all possible combinations of argument ranges

Usage

```
multicall(FUN, ..., fix. = names(formals(FUN))[1], max.combinations = 1000)
```

Arguments

`FUN`	function to be called
`...`	scalars or vectors named as the arguments that are accepted by `FUN`. From all objects with a name that is not in `fix.` combinations of their values are created and `FUN` is called with each of these combinations.
`fix.`	names of arguments to be kept constant
`max.combinations`	maximum number of argument combinations to be created at maximum

Examples

```
bmi <- function(mass, height) round(mass / (height * height), 1)

kwbGompitz:::multicall(
  bmi, mass = 60:70, height = seq(1.7, 1.8, 0.1), fix. = NULL
)
```

multiplot_survival *Plot several Survival Curves*

Description

Plot several Survival Curves

Usage

```
multiplot_survival(
  t,
  alpha = 0,
  bz0 = 0,
  bz1 = 0,
  span = rep(TRUE, 3),
  theme = ggplot2::theme_bw()
)
```

Arguments

`t`	vector of times for which to calculate the survival curves
`alpha`	parameter alpha of the survival function
`bz0`	parameter bz0 of the survival function
`bz1`	parameter bz1 of the survival function
`span`	vector of logical of length three indicating for each of the parameters `alpha`, `bz0`, `bz1` if they are to be varied or to be kept fix
`theme`	ggplot2-theme applied to the plots

`parameterLines` *Generate text lines for param.txt*

Description

Convert the list structure containing calibration parameters as retrieved by `kwbGompitz:::readParameters` to a vector of character representing the text lines to be written to file

Usage

```
parameterLines(parameters, sep = ";", digits_exp = 2L)
```

Arguments

`parameters`	list structure containing calibration parameters as provided by `kwbGompitz:::readParameters`
`sep`	column separator. Default: ";"
`digits_exp`	number of exponent digits in scientific notation, such as "1.23e+01" (digits_exp = 2L, the default) or "1.23e+001" (digits_exp = 3L)

Value

vector of character representing the lines of param.txt, containing the calibration parameters as generated by /codegompcal

parameterplot *Plot multiple Survival Curves*

Description

Plot multiple Survival Curves

Usage

```
parameterplot(t, alpha = 0, bz0 = 0, bz1 = 0, xlim = c(0, 1.2 * max(t)))
```

Arguments

`t`	vector of times for which to calculate the survival curves
`alpha`	vector of parameters alpha to be passed to the survival function
`bz0`	vector of parameters bz0 to be passed to the survival function
`bz1`	vector of parameters bz1 to be passed to the survival function
`xlim`	limits of the x axis

plotCalibration1 *Plot Calibration Result*

Description

Plot Calibration Result

Usage

```
plotCalibration1(calib, to.pdf = FALSE)
```

Arguments

`calib`	calibration result as retrieved by `runGompitzCalibration`
`to.pdf`	if `TRUE` the plot is directed into a temporary PDF file

plotCalibration2 *Plot Calibration Result (2)*

Description

Plot Calibration Result (2)

Usage

```
plotCalibration2(calib, to.pdf = FALSE)
```

Arguments

calib	calibration result as retrieved by runGompitzCalibration
to.pdf	if TRUE the plot is directed into a temporary PDF file

plotPrediction *Plot the Result of a Prediction*

Description

Plot the distribution of condition classes by predicted year

Usage

```
plotPrediction(
  prediction,
  legend_pos = c("bottom", "left", "top", "right", "none")[5],
  do_print = TRUE
)
```

Arguments

prediction	data frame with columns prob1, prob2, ..., pipeLength, InspectionYear
legend_pos	legend position: one of "bottom", "left", "top","right", "none"
do_print	logical indicating whether to actually print the plot to the current graphical device.

Examples

```
prediction <- data.frame(
  prob1 = c(0, 1, 0, 0.0, 0),
  prob2 = c(1, 0, 0, 0.5, 0),
  prob3 = c(0, 0, 0, 0.5, 1),
  prob4 = c(0, 0, 1, 0.0, 0),
  pipeLength = 1:5,
  InspectionYear = seq(2001, 2005)
)

kwbGompitz:::plotPrediction(prediction, legend_pos = "right")
```

plotPredictionByYear *Plot Prediction by Year to temp. PDF File*

Description

Plot prediction by year to temporary PDF file

Usage

```
plotPredictionByYear(
  prediction,
  to.pdf = TRUE,
  FUN.name = "orderByWeightedProbabilities",
  column.year = "year"
)
```

Arguments

prediction	prediction result as provided by e.g. runGompitzPrediction
to.pdf	if TRUE (default) the plot goes into a PDF file
FUN.name	name of function to be used to order the matrix of probabilities. Available functions: "orderByWeightedProbabilities", "orderByMostProbClassAndDecreasingProb"; default: "orderByWeightedProbabilities"
column.year	name of column containing the year

See Also

runGompitzPrediction

plotPredictionForYear *Plot Prediction for the given Year*

Description

Plot Prediction for the given Year

Usage

```
plotPredictionForYear(
  res,
  year,
  FUN.name = "orderByWeightedProbabilities",
  main = NULL,
  column.year = "year"
)
```

Arguments

res	prediction result as provided by e.g. runGompitzPrediction
year	year number for which results are to be filtered an plot is to be generated
FUN.name	name of function to be used to order the matrix of probabilities. Available functions: "orderByWeightedProbabilities", "orderByMostProbClassAndDecreasingProb"; default: "orderByWeightedProbabilities"
main	main title of plot
column.year	name of column containing the year

See Also

runGompitzPrediction

plot_pipe_conditions *Plot the Pipe Conditions over Time*

Description

Plot the Pipe Conditions over Time

Usage

```
plot_pipe_conditions(
  x,
  column_instyear = "instyear",
  column_ident = "ident",
  column_inspyear = "inspyear",
  column_condition = "condition",
  colour_values = c(C4 = "green", C3 = "yellow", C2 = "orange", C1 = "red", "grey"),
  y_labels = FALSE,
  facet_by = sprintf(kwb.utils::underscoreToPercent("~(10*as.integer(_s/10))"),
    column_instyear),
  by_pipe = FALSE
)
```

Arguments

`x` data frame

`column_instyear` name of column containing the installation year

`column_ident` name of column containing the pipe identifier

`column_inspyear` name of column containing the inspection year

`column_condition` name of column containing the pipe condition

`colour_values` named vector of colour names. The names must refer to the values in `column_condition`.

`y_labels` logical. If `TRUE` tick marks and labels are plotted on the y axis, else not.

`facet_by` character string representing a formula to be used to create facets with `facet_wrap`. Set to `NULL` if no facets are desired.

`by_pipe` if `TRUE` one plot is created for each pipe and a list of these plots is returned

Examples

```
## Not run:
x <- kwbGompitz:::readObservations(kwbGompitz::exampleFile("obs.txt"))
plot_pipe_conditions(x[1:100, ])

## End(Not run)
```

`plot_prediction_by_pipe` *Plot Condition Probabilities by Pipe*

Description

List of ggplots with each plot representing the evolution of condition probabilities over time for one pipe

Usage

```
plot_prediction_by_pipe(
  prediction,
  prefix = "prob",
  pipe_ids = unique(kwb.utils::selectColumns(prediction, "pipe_id")),
  width = 1
)
```

Arguments

`prediction`	data frame as returned by `runGompitzPrediction`
`prefix`	prefix of column names containing the probabilities. Default: "prob"
`pipe_ids`	vector of pipe IDs for which a plot is to be generated. By default all available pipes are considered!
`width`	passed to `geom_col` (default: 1)

Value

???

`plot_stacked_bars` *Generate gg-plot of Stacked Bars*

Description

Generate gg-plot of Stacked Bars

Usage

```
plot_stacked_bars(x, x.sd = NULL, reverse = TRUE, ...)
```

Arguments

`x`	matrix containing the bar heights
`x.sd`	optional. Standard deviations.
`reverse`	logical indicating whether to reverse the stack order
`...`	additional arguments passed to `gg_stacked_bars`

Examples

```
values <- c(
  0, 2, 0, 0, 0,
  1, 0, 0, 2, 0,
  0, 0, 0, 2, 5,
  0, 0, 3, 0, 0
)

conditionStat <- matrix(values, nrow = 4, byrow = TRUE, dimnames = list(
  condition = paste0("prob", 1:4),
  year = 2001:2005
))

# Get the base plot
baseplot <- kwbGompitz::plot_stacked_bars(conditionStat, legend = "right")

# Show the base plot
baseplot

# Modify the base plot (titles, axis titles, legend title, legend content)
baseplot + ggplot2::labs(
  x = "Jahr", y = "Anteil Kanaele in %", title = "Zustandsverteilung",
  subtitle = "Netzwerk: Berlin"
) +
ggplot2::guides(fill = ggplot2::guide_legend("Zustandsklasse")) +
ggplot2::scale_fill_manual(
  values = c("darkgreen", "yellow", "darkorange3", "red3"),
  labels = c("good", "ok", "not so good", "bad")
)
```

plot_survival_curves *Plot Survival Curves*

Description

Plot Survival Curves

Usage

```
plot_survival_curves(
  calibration = exampleCalibration(),
  stratum = NULL,
  marginal = TRUE,
  t = 1:100,
  col = NULL,
  ...,
  args = NULL
)
```

Arguments

`calibration`	calibration object
`stratum`	name of stratum
`marginal`	logical. If `TRUE` (default) the marginal survival curves are plotted, otherwise the non-marginal survival curves
`t`	vector of times at which to calculate the survival curves
`col`	vector of colour names, named according to the conditions as stored in `attr(calibration, "parameters")$conditions`
`...`	arguments passed to `plot_curve_areas_gg`, such as `legend`, `line_colour`
`args`	arguments passed to get_survivals

Examples

```
## Not run:
# Get the example calibration provided with the Gompitz software
calibration <- kwbGompitz::exampleCalibration()

# Generate one ggplot2-object for each calibrated stratum
(plots_1 <- plot_survival_curves(calibration))

# By default the marginal survival curves are shown. You may set marginal to
# FALSE to get the non-marginal survival curves
(plots_2 <- plot_survival_curves(calibration, marginal = FALSE))

# Compare the plots
gridExtra::grid.arrange(plots_1[[1]], plots_2[[1]], plots_1[[2]], plots_2[[2]])

## End(Not run)
```

prepare_for_gg_stacked_bars

Prepare data for plot_stacked_bars()

Description

Prepare data for plot_stacked_bars()

Usage

```
prepare_for_gg_stacked_bars(x, x.sd = NULL)
```

Arguments

`x`	numeric vector
`x.sd`	standard deviation

print.gompitz.calibration

Print a GompitZ Calibration Structure

Description

Print a GompitZ Calibration Structure

Usage

```
## S3 method for class 'gompitz.calibration'
print(x, ...)
```

Arguments

`x`	list of class "gompitz.calibration" as returned by `runGompitzCalibration`
`...`	further arguments (not used)

Value

print GompitZ Calibration Structure

print.gompitz_stratum_calib

Print the Calibration for one Stratum

Description

Print the Calibration for one Stratum

Usage

```
## S3 method for class 'gompitz_stratum_calib'
print(x, ...)
```

Arguments

`x`	list of class "gompitz_stratum_calib" as contained in a calibration object returned by `runGompitzCalibration`
`...`	further arguments (not used)

Value

print the Calibration for one Stratum

`printConversgence` *Print the Convergence Information*

Description

Print the Convergence Information

Usage

```
printConvergence(x)
```

Arguments

`x`	`NULL` or a list with elements `num.iterations`, `log.likelihood`, `covariances`, `estimates`,

Value

Print the convergence information

`readCalibration` *Read Calibration Result*

Description

Read calibration result from given file

Usage

```
readCalibration(file, verbose = 1)
```

Arguments

`file`	path to gompcal result file "calibr.txt"
`verbose`	integer number specifying the level of verbosity

readObservations *Read Observation Data from File in GompitZ-Format*

Description

Read Observation Data from File in GompitZ-Format

Usage

```
readObservations(file = exampleFile("obs.txt"), sep = ";", dbg = TRUE)
```

Arguments

`file`	full path to text file in the format that is required by GompitZ (see "The GompitZ Tool User's Guide")
`sep`	column separator
`dbg`	if `TRUE` debug messages are shown, else not.

readParameters *Read Parameters from "param.txt"*

Description

Read parameters from param.txt into list structure

Usage

```
readParameters(file, sep = ";", dbg = FALSE)
```

Arguments

`file`	full path to parameter file "param.txt", generated by gompcal.exe
`sep`	column separator, default: ";"
`dbg`	if `TRUE` (default) debug messages are shown

Value

list with sections `conditions`, /codestrata, /codecovariates, /codecategoryLevels, /codebyStratum,

readPredictionFile *Read Prediction Result File*

Description

Read Prediction Result File

Usage

```
readPredictionFile(file, sep = ";", stringsAsFactors = FALSE)
```

Arguments

`file`	full path to prediction result file (predict<n>.txt) with <n> being the strategy number
`sep`	column separator, default: ";"
`stringsAsFactors`	if `TRUE` character columns will appear as factors in the result, passed to `data.frame`. The default is `FALSE`.

runGompcalExample *Run Calibration Example*

Description

Run calibration example provided with GompitZ

Usage

```
runGompcalExample()
```

runGompcalInDirectory *Run gompcal.exe in given Directory*

Description

Run gompcal.exe in given Directory

Usage

```
runGompcalInDirectory(
  target.dir = tempGompitzDir(),
  input.file = exampleFile("obs.txt"),
  sep = ";",
  ...
)
```

Arguments

`target.dir`	path to target directory
`input.file`	path to input file
`sep`	column separator
`...`	arguments that are passed to .runModuleInDirectory, such as `verbose` or `show.error`

`runGompitzCalibration` *Perform GompitZ Calibration*

Description

Perform GompitZ Calibration

Usage

```
runGompitzCalibration(
  input.data,
  subset = NULL,
  verbose = 1,
  sep = ";",
  digits = 3,
  VERSION = getOperatingSystemType()
)
```

Arguments

`input.data`	prepared input data as retrieved by `composeGompitzInputData`
`subset`	indexes of rows in `input.data` to be used for calibration. If `NULL` (default), all rows are used.
`verbose`	verbosity level
`sep`	column separator
`digits`	round estimates read from calibr.txt and param.txt, respectively, to this number of (significant) digits before comparing them. Default: 3
`VERSION`	name of subdirectory in package containing the binary files to be executed. Possible values: "unix", "win32", "win32_kwb"

Examples

```
# For an example, see the Tutorial vignette "How to Use the Package"
```

`runGompitzPrediction` *Run Gompitz Predicion*

Description

Run Gompitz Predicion

Usage

```
runGompitzPrediction(
  input.data,
  subset = NULL,
  calibration,
  strategy = 0,
  ...,
  verbose = 1,
  do.stop = TRUE,
  clear.observations = TRUE,
  VERSION = getOperatingSystemType(),
  use.data.table = getOption("kwbGompitz.use.data.table", FALSE)
)
```

Arguments

`input.data` prepared input data as retrieved by `composeGompitzInputData`

`subset` indexes of rows in `input.data` to be used for prediction. if `NULL`, all rows in `input.data` will be used.

`calibration` result of Gompitz calibration as retrieved by `runGompitzCalibration`

`strategy` strategy identifier. Must be one of 0 (do nothing), 1 (length-driven strategy), 2 (budget-driven strategy), or 3 (optimised strategy).")

`...` arguments passed to the corresponding `.fileContentStrategy` functions, such as `range.years` (two-element vector with first and last year of prediction), rehabilitation.costs (needed for strategy = 1 or 2 or 3, see `kwbGompitz:::.fileContentStrategy1` or `kwb.gomiptz:::.fileContentStrategy2` or `kwb.gomiptz:::.fileContentStrategy3`), annual.total.length (needed for strategy = 1, see `kwbGompitz:::.fileContentStrategy1`), annual.total.budget (needed for strategy = 2, see `kwb.gomiptz:::.fileContentStrategy2`), max.tol.prop.of.length (needed for strategy = 3, see `kwb.gomiptz:::.fileContentStrategy3`), target.year (needed for strategy = 3, see `kwb.gomiptz:::.fileContentStrategy3`)

`verbose` verbosity level, default: 1

`do.stop` if `TRUE` the program stops in case of inconsistencies

`clear.observations`
if `TRUE` (default) the columns containing the inspection year and the observed condition class, respectively, are cleared in the input file given to `gompred`. Otherwise the observed condition classes are kept in the input file and thus considered by `gompred`.

`VERSION`	name of subdirectory in package containing the binary files to be executed. Possible values: "unix", "win32", "win32_kwb"
`use.data.table`	if `TRUE`, `fread` is used to read the result file

Examples

```
# For an example, see the Tutorial vignette "How to Use the Package"
```

`runGompredInDirectory` *Run gompred.exe in given Directory*

Description

Run gompred.exe in given Directory

Usage

```
runGompredInDirectory(
  target.dir = tempdir(),
  input.file = exampleFile("obs.txt"),
  sep = ";",
  strategy = 0,
  ...
)
```

Arguments

`target.dir`	path to target directory
`input.file`	path to input file
`sep`	column separator
`strategy`	integer number specifying the strategy to be applied
`...`	arguments that are passed to .runModuleInDirectory, such as `verbose` or `show.error`

`spanIf` *Span a Vector around x if Condition is met*

Description

Span a Vector around x if Condition is met

Usage

```
spanIf(condition, x, digits = 4)
```

Arguments

condition	logical. If TRUE a vector around x is spanned
x	value around which to span a vector (or not if condition is FALSE)
digits	number of digits to which the values are rounded

SQRT_2_PI *Mathematical Constant* sqrt(2 * pi)

Description

Mathematical Constant sqrt(2 * pi)

Usage

```
SQRT_2_PI
```

Format

An object of class numeric of length 1.

---

standard_survival *Weibull condition state survival function*

Description

Weibull condition state survival function

Usage

```
standard_survival(
  alpha,
  t,
  bz1,
  bz0 = 0,
  offset = t * exp(bz1) + bz0,
  limits = NULL
)
```

Arguments

alpha parameter "alpha"

t time

bz1 parameter "bz1"

bz0 parameter "bz0"

offset in order not to recalculate the following expression each time again, its result can be given here: = bz0 + t * exp(bz1). If given, the arguments t, bz0, and bz1 can be omitted, otherwise they are required and the offset is calculated according to the above expression.

limits numeric vector of two elements giving the minimum and maximum value to which the result shall be restricted. If not given or NULL the result will not be restricted to a value range.

---

strategyFileContent *Content for Strategy File*

Description

Create the text content for a gompred strategy file

Usage

```
strategyFileContent(strategy, ...)
```

Arguments

strategy strategy identifier. Must be one of 0 (do nothing), 1 (length-driven strategy), 2 (budget-driven strategy), or 3 (optimised strategy).")

... arguments passed to the corresponding .fileContentStrategy functions, see hidden functions kwbGompitz:::.fileContentStrategy0, kwbGompitz:::.fileContentStrategy1, kwbGompitz:::.fileContentStrategy2, kwbGompitz:::.fileContentStrategy3

Value

character vector of length one representing the file content of the strategy file

summary_generic_sim *Aggregate simulated Condition Classes*

Description

Aggregate simulated Condition Classes

Usage

```
summary_generic_sim(x, column.group.by, column.length)
```

Arguments

x	data frame
column.group.by	name of column in x by which values to group by
column.length	name of column in x containing the pipe lengths

survivals *Get Values of (Standard or Marginal) Survival Curves*

Description

Get Values of (Standard or Marginal) Survival Curves

Usage

```
survivals(
  t = 0:99,
  alpha,
  bz1,
  bz0,
  s = NULL,
  marginal = !is.null(s),
  matrix = TRUE,
  set_attributes = FALSE,
  interpol_info = NULL,
  ...
)
```

Arguments

t — numeric vector of times (ages)

alpha — numeric vector of alpha-parameter(s)

bz1 — bz1 parameter

bz0 — bz0 parameter

s — passed to marginal_survival

marginal — if TRUE the marginal survival curve with s as standard deviation is calculated instead of the standard survival curve. By default marginal is TRUE if s is not NULL.

matrix — if TRUE and the length of alpha is greater than one the result is a matrix with each row representing one alpha value and each column representing a time. Otherwise the result is a list with each list element representing one alpha value

set_attributes — if TRUE (the default is FALSE) an attribute args containing the arguments given to this function is set in the result

interpol_info — if not NULL it is expected to be a list containing the elements start (start year of interpolation) and length (duration of interpolation in years)

... — further arguments passed to marginal_survival or standard_survival

survivals_original — *Get Values of (Standard or Marginal) Survival Curves*

Description

Get Values of (Standard or Marginal) Survival Curves

Usage

```
survivals_original(
  t = 0:99,
  alpha,
  bz1,
  bz0,
  s = NULL,
  marginal = !is.null(s),
  matrix = TRUE,
  set_attributes = FALSE,
  ...
)
```

Arguments

`t` numeric vector of times (ages)

`alpha` numeric vector of alpha-parameter(s)

`bz1` bz1 parameter

`bz0` bz0 parameter

`s` passed to `marginal_survival`

`marginal` if `TRUE` the marginal survival curve with `s` as standard deviation is calculated instead of the standard survival curve. By default `marginal` is `TRUE` if `s` is not `NULL`.

`matrix` if `TRUE` and the length of `alpha` is greater than one the result is a matrix with each row representing one `alpha` value and each column representing a time. Otherwise the result is a list with each list element representing one `alpha` value

`set_attributes` if `TRUE` (the default is `FALSE`) an attribute `args` containing the arguments given to this function is set in the result

`...` further arguments passed to `marginal_survival` or `standard_survival`

`tempGompitzDir` *Create temporary gompitz directory*

Description

Create temporary gompitz directory

Usage

```
tempGompitzDir(verbose = 1)
```

Arguments

`verbose` integer value determining the level of verbosity

`test_marginal_survival` *Test Marginal Survival*

Description

Test Marginal Survival

Usage

```
test_marginal_survival(t = 0:99)
```

Arguments

`t`	t (default: 0:99)

Value

???

`toStatusMatrix` *Convert Text Lines to Status Matrix*

Description

Convert text lines to status matrix, as e.g. required by `kwb.rsproto::configure`

Usage

```
toStatusMatrix(textlines, sep = ";", order_rows = FALSE)
```

Arguments

`textlines`	vector of character linees as they appear in the GompitZ input file header (only the lines defining the status matrix)
`sep`	column separator. Default: ";"
`order_rows`	if `TRUE` (default) the rows are ordered by their name

Examples

```
## Not run:
file <- kwbGompitz::exampleFile("obs.txt")
kwbGompitz:::toStatusMatrix(textlines = readLines(file, 8)[-(1:2)])

## End(Not run)
```

`to_theme_legend` *Generate ggplot Theme for Legend*

Description

Generate ggplot Theme for Legend

Usage

```
to_theme_legend(legend)
```

Arguments

`legend`	logical indicating whether to put a legend or not or a string giving the legend position ("left", "right", "top", "bottom")

underline *Create a String to be used as an Underline*

Description

Create a String to be used as an Underline

Usage

```
underline(n = 10, char = "-")
```

Arguments

`n`	number of characters
`char`	character used for the underline

writeInputFile *Write the Input File*

Description

Write the given character vector into the input file at given path

Usage

```
writeInputFile(textlines, file, verbose = 1)
```

Arguments

`textlines`	vector of text lines to be written to `file`
`file`	path to the file to be written
`verbose`	integer number specifying the verbosity level. If this is a positive value, debug messages are shown.

`writeParameters` *Write Calibration Parameters to File*

Description

Write Calibration Parameters to File

Usage

```
writeParameters(
  parameters,
  file,
  sep = ";",
  dbg = TRUE,
  warn = FALSE,
  digits_exp = 2L
)
```

Arguments

`parameters` list structure containing calibration parameters as provided by `kwbGompitz:::readParameters`

`file` full path to file to which parameters are to be written

`sep` column separator. Default: ";"

`dbg` if `TRUE` (default) debug messages are shown

`warn` if `TRUE` a message is shown that the existing parameter file was overwritten

`digits_exp` passed to `parameterLines`

Index