Package: kwb.demeau (via r-universe)

August 30, 2024

Title Heat tracer study SVH

Version 0.0.0.9000

Description Heat tracer study for SVH site using USGS VS2DH model.

License MIT + file LICENSE

URL https://github.com/KWB-R/kwb.demeau

BugReports https://github.com/KWB-R/kwb.demeau/issues

Depends R (>= 3.0.0), shapefiles, hydroGOF, lattice, latticeExtra, kwb.vs2dh, kwb.utils, kwb.db

Imports RCurl, plyr, sp

Suggests testthat

Remotes github::kwb-r/kwb.utils, github::kwb-r/kwb.db, github::kwb-r/kwb.vs2dh

Encoding UTF-8

LazyLoad yes

RoxygenNote 6.1.1

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

RemoteUrl https://github.com/KWB-R/kwb.demeau

RemoteRef HEAD

RemoteSha 3c82a60433956879b9812472fba7075dd9318bee

Contents

addHorizontalDistances *Helper function: add horizontal distances from left boundary coordinates*

Description

Helper function: add horizontal distances from left boundary coordinates

Usage

```
addHorizontalDistances(gisData, leftBoundaryRow = 1)
```

Arguments

gisData	data.frame as retrieved by importShapefiles()
leftBoundaryRow	row number index of "gisData" which contains the left model boundary (Default: 1)

Value

Model coordinates instead of "real" world coordinates

Examples

```
shp.dir <- system.file("extdata", "qgis", package="kwb.demeau")
shp.files <- dir(path = shp.dir, pattern = ".shp", full.names = TRUE)
### Store GIS data in R data.frame "gisData"
gisData <- importShapefiles(shp.files)
### Add horizontal distances from left boundary and ignore all features that
### have larger distances than "right" boundary
gisData <- addHorizontalDistances(gisData)
### Plot horizontal model coordinates
maxVertical <- -abs(max(gisData$fcBottom, na.rm=TRUE))
```

calcWaterLevelChange *Helper function: calculate water level change*

Description

Helper function: calculate water level change

Usage

```
calcWaterLevelChange(refDate = "2009-03-02", df)
```

Arguments

refDate	reference date for start (Default: "2009-03-02")
df	data frame with structure like moniDataggdailyMedian

---

calibrateModel *Calibration: single parameter*

Description

Calibration: single parameter

Usage

```
calibrateModel(preparedHeatModel, moniDat, obsPoints = "*",
  parameterName = "ratioKzKh", parameterRange = c(0.01, 0.1),
  objState = "waterLevelChange", objCrit = "RMSE", ...)
```

Arguments

preparedHeatModel	as retrieved by prepareHeatModel()
moniDat	moniDat
obsPoints	regular expression of observation points/wells to be included for goodness of fit calculation (Default: *, i.e. all); if only BSV-6_3, then: "BSV-6-3"
parameterName	parameterName (Default: "ratioKzKh")
parameterRange	parameterRange (Default: c(0.01, 0.1))
objState	model state variable to be optimised either "waterLevelChange" or "temp" (for temperature) (Default: "waterLevelChange")
objCrit	vector with performance parameters produced by function hydroGOF:gof(), Default: "RMSE" (valid parameters: "ME", MAE", "MSE", "RMSE", "NRMSE", "PBIAS", "RSR", "rSD", "NSE", "mNSE", "rNSE", "d", "md", "rd", "cp", "r", "R2", "bR2", "KGE", "VE"), ATTENTION: currently optimising is implemented to MINIMISE the value of only ONE selected objCrit parameter. Thus please make sure that the best model fit results of the MINIMUM of the selected parameter
...	further arguments passed to hydroGOF::gof()

Value

calibration results

`compareModelledMeasured` *Compare measured & modelled resutls*

Description

Compare measured & modelled resutls

Usage

```
compareModelledMeasured(heatModel, moniDat, toPlot = TRUE)
```

Arguments

`heatModel`	object as retrieved by runHeatModel()
`moniDat`	as retrieved by processingData()
`toPlot`	If TRUE results are plotted (Default: TRUE)

Value

Plot/List of water level, water level change & temperature of measured vs. modelled data

`convertToListAndAddMoniColumns` *Monitoring: convert to list & add monitoring columns*

Description

Monitoring: convert to list & add monitoring columns

Usage

```
convertToListAndAddMoniColumns(df, keyFields = "myDateTime")
```

Arguments

`df`	dataframe containing the data to be transformed
`keyFields`	keyFields to be used as keyFields in kwb.hsMatrixToListForm (Default: "myDateTime")

`convHeadBoundary` *Helper function: convert head boundary*

Description

Helper function: convert head boundary

Usage

```
convHeadBoundary(leftHead = TRUE, modelStructure, depthToWaterTable,
  hydraulicGradient)
```

Arguments

`leftHead`	if TRUE left head, if FALSE right head (Default: TRUE)
`modelStructure`	modelStructure as retrieved by
`depthToWaterTable`	depthToWaterTable
`hydraulicGradient`	hydraulicGradient

Value

head boundary meta info

`convModelCoordinatesToNodes`
Conversion: model coordinates to nodes

Description

Conversion: model coordinates to nodes

Usage

```
convModelCoordinatesToNodes(modelCoords, dx = 1, dy = 1)
```

Arguments

`modelCoords`	as retrieved by convRealToModelCoordinates()
`dx`	horizontal model grid spacing (Default: 1)
`dy`	vertical model grid spacing (Default: 1)

Value

Model nodes corresponding to model coordinates

Examples

```
shp.dir <- system.file("extdata", "qgis", package="kwb.demeau")
shp.files <- dir(path = shp.dir, pattern = ".shp", full.names = TRUE)
gisData <- importShapefiles(shp.files)
### Optionally remove some features
# gisData <- removeFeatures(gisData = gisData, ignoreFeatureIDs = c(3,20))
modelCoords <- convRealToModelCoordinates(gisData, dx=1, dy=1)
modelNodes <- convModelCoordinatesToNodes(modelCoords=modelCoords)
```

---

convRealCoordinatesToNodes

Conversion: "real" coordinates to model nodes

Description

By calling functions convRealToModelCoordinates() and convModelCoordinatesToNodes()

Usage

```
convRealCoordinatesToNodes(gisData, dx = 1, dy = 1, y = NULL)
```

Arguments

gisData	data.frame() as retrieved by importShapefiles()
dx	horizontal model grid spacing (Default: 1)
dy	vertical model grid spacing (Default: 1)
y	vertical model extent (Default: NULL, i.e. maximum filter screen depth below ground level)

Value

model nodes

convRealToModelCoordinates *Conversion: "real" coordinates to model coordinates*

Description

Conversion: "real" coordinates to model coordinates

Usage

```
convRealToModelCoordinates(gisData, dx = 1, dy = 1, y = NULL)
```

Arguments

gisData	data.frame() as retrieved by importShapefiles()
dx	horizontal model grid spacing (Default: 1)
dy	vertical model grid spacing (Default: 1)
y	vertical model extent (Default: NULL, i.e. maximum filter screen depth below ground level)

Examples

```
shp.dir <- system.file("extdata", "qgis", package="kwb.demeau")
shp.files <- dir(path = shp.dir, pattern = ".shp", full.names = TRUE)
gisData <- importShapefiles(shp.files)
### Optionally remove some features
# gisData <- removeFeatures(gisData = gisData, ignoreFeatureIDs = c(3,20))
modelCoords <- convRealToModelCoordinates(gisData, dx=1, dy=1)
xyplot(y ~ x, groups = Name, data=modelCoords, pch=16, auto.key=TRUE)
```

defineHeadBoundary *Helper function: define head boundary*

Description

Helper function: define head boundary

Usage

```
defineHeadBoundary(head, nly, temp, ntx = 4, ntc = 1)
```

Arguments

head	as retrieved by convHeadBoundary()
nly	number of vertical nodes
temp	temperature at boundary
ntx	vector with node type identifier for boundary conditions. 0 (for no specified boundary (needed for resetting some nodes after initial recharge period); 1 (for specified pressure head); 2 (for specified flux per unit horizontal surface area in units of L/T); 3 (for possible seepage face); 4 (for specified total head); 5 (for evaporation, Note: is not implemented yet!); 6 (for specified volumetric flow in units of L3/T). 7 (for gravity drain). (The gravity drain boundary condition allows gravity driven vertical flow out of the domain assuming a unit vertical hydraulic gradient. Flow into the domain cannot occur.)"
ntc	vector with node type identifier for transport boundary conditions. 0 (for no specified boundary); 1 (for specified temperatures), (Default: 1)

Value

boundary head parameterisation

---

dominantTravelTimes *Dominant travel time: data preprocessing*

Description

Dominant travel time: data preprocessing

Usage

```
dominantTravelTimes(concModelled, offset = 0.01)
```

Arguments

concModelled	as retrieved by kwb.demeau::soluteModelled()
offset	offset (Default: 0.01) used for filtering soluteModel results (i.e. maxConc/2 +- offset)

Value

list with dominant travel times with sublists "raw" (multiple values for each TIME_day possible) and "agg" (median "TIME_day" and "modelled" concentration)

dominantTravelTimesAgg *Helper function for dominantTravelTimes*

Description

Helper function for dominantTravelTimes

Usage

```
dominantTravelTimesAgg(domTimes)
```

Arguments

domTimes	intermediate result of function kwb.demeau::dominantTravelTimes

Value

list with aggregated dominant travel times (median!)

downloadMeteoData *Monitoring: download meteo data*

Description

Monitoring: download meteo data

Usage

```
downloadMeteoData(startYear = 2008, endYear = 2014)
```

Arguments

startYear	1.1.startYear (Default: 1.1.2008)
endYear	31.12.endYear (Default: 31.12.2014)

`filterMoniData` *Helper function: filter monitoring data*

Description

Helper function: filter monitoring data

Usage

```
filterMoniData(locations = NULL, locationsCol = "moniLocation",
  paras = NULL, paraCol = "moniParName", minDate = "2009-03-02",
  maxDate = "2009-04-06", df)
```

Arguments

`locations`	(Default: NULL)
`locationsCol`	(Default: "moniLocation")
`paras`	(Default: NULL)
`paraCol`	(Default: "moniParName")
`minDate`	(Default: "2009-03-02")
`maxDate`	(Default: "2009-04-06")
`df`	data.frame structure like: moniDataggdailyMedian

`fitnessAdaptedModelConfiguration` *Calibration: helper function "fitnessAdaptedModelConfiguration" (called by function calibrateModel()*

Description

Calibration: helper function "fitnessAdaptedModelConfiguration" (called by function calibrateModel()

Usage

```
fitnessAdaptedModelConfiguration(parameterValue, parameterName,
  preparedHeatModel, objState = "waterLevelChange", objCrit = "RMSE",
  moniDat, obsPoints, ...)
```

Arguments

parameterValue	parameterValue
parameterName	parameterName
preparedHeatModel	as retrieved by prepareHeatModel()
objState	model state variable to be optimised either "waterLevelChange" or "temp" (for temperature) (Default: "waterLevelChange")
objCrit	vector with performance parameters produced by function hydroGOF:gof(), Default: "RMSE" (valid parameters: "ME", MAE", "MSE", "RMSE", "NRMSE", "PBIAS", "RSR", "rSD", "NSE", "mNSE", "rNSE", "d", "md", "rd", "cp", "r", "R2", "bR2", "KGE", "VE")
moniDat	moniDat
obsPoints	regular expression of observation points/wells to be included for goodness of fit calculation (Default: *, i.e. all); if only BSV-6_3, then: "BSV-6-3"
...	further arguments passed to hydroGOF::gof()

Value

fitness of model configuration

---

fitnessWithLabel *Compare measured & modelled results*

Description

Compare measured & modelled results

Usage

```
fitnessWithLabel(heatModel, moniDat, objState = "waterLevelChange",
  objCrit = c("RMSE", "R2"), plotIt = TRUE, plot.type = "b",
  cex.label = 1, main = NULL, performance.in.label = TRUE, ...)
```

Arguments

heatModel	object as retrieved by runHeatModel()
moniDat	as retrieved by processingData()
objState	model state variable to be optimised either "waterLevelChange" or "temp" (for temperature) (Default: "waterLevelChange")
objCrit	vector with performance parameters produced by function hydroGOF:gof(), Default: "RMSE" (valid parameters: "ME", MAE", "MSE", "RMSE", "NRMSE", "PBIAS", "RSR", "rSD", "NSE", "mNSE", "rNSE", "d", "md", "rd", "cp", "r", "R2", "bR2", "KGE", "VE")

plotIt	if TRUE lattice plot will be produced (Default: TRUE)
plot.type	plot type
cex.label	character expansion factor for labels
main	plot title
performance.in.label	if TRUE, the performance indicator appears as label
...	additional parameters passe to plot function lattice:xyplot()

Value

Plot of water level, water level change & temperature of measured vs. modelled data

---

genuchten *vanGenuchten : helper for multiple models*

Description

vanGenuchten : helper for multiple models

Usage

```
genuchten(pressureHeads = -rev(seq(0, 1000, 5)), alpha = 1, beta = 2)
```

Arguments

pressureHeads	pressureHeads
alpha	alpha to be used for van Genuchten model
beta	betaa to be used for van Genuchten model

Value

data.frame with columns "pressureHead, alpha, beta, effSaturation, Kr"

References

http://wwwbrr.cr.usgs.gov/projects/GW_Unsat/vs2di/hlp/solute/vanGenuchten.html

genuchtenModel *vanGenuchten : single model*

Description

vanGenuchten : single model

Usage

```
genuchtenModel(pressureHead, alpha, beta)
```

Arguments

pressureHead	pressureHead
alpha	alpha
beta	beta

Value

data.frame with columns "pressureHead, alpha, beta, effSaturation, Kr"

References

http://wwwbrr.cr.usgs.gov/projects/GW_Unsat/vs2di/hlp/solute/vanGenuchten.html

genuchtenModels *vanGenuchten : multiple models*

Description

vanGenuchten : multiple models

Usage

```
genuchtenModels(pressureHeads = -rev(seq(0, 6, 0.5)), alphas = seq(1,
  2, 0.5), betas = 1:5)
```

Arguments

pressureHeads	pressureHeads
alphas	vector of alphas to be used for multiple model construction
betas	vector of betas to be used for multiple model construction

Value

data.frame with columns "pressureHead, alpha, beta, effSaturation, Kr"

References

http://wwwbrr.cr.usgs.gov/projects/GW_Unsat/vs2di/hlp/solute/vanGenuchten.html

---

`getFeatures` *Helper function: Gets feature information*

Description

Aggregates gis features by shape.name & id add adds new column fID and returns additional metainformation (parameter)

Usage

```
getFeatures(gisData, addColNames = NULL)
```

Arguments

`gisData`	data.frame with gis features as retrieved by importShapefile(s)()
`addColNames`	vector with additional colnames for output data.frame (Default: NULL), for valid inputs check: colNames(gisData)

Value

Return the features with attributes feature id (fID), shapefile name (shape.name) and feature name (Name)

`importData` *Monitoring: excel data import*

Description

Monitoring: excel data import

Usage

```
importData(xlsPath = NULL, metaTables = c("metadataTable",
  "commentTable"), moniTables = c("bsv4", "bsvAll", "tuberia", "inflow"),
  ignoredParNames = c("Temp_graphics_C", "DiverBaro_graphFictive_cm",
  "DiverBaro_cm"))
```

Arguments

xlsPath	full path to excel file http://192.168.22.12/svn/kwb/DEMEAU/Work Areas/WA1 MAR/TracerSVH/datosTOT_StVicen+_jun2008-abr2009OK.xls", if FALSE already imported data object moniDat is loaded (Default: NULL)
metaTables	vector with names of tables with meta information (Default: "metadataTable", "commentTable")
moniTables	vector with names of tables with monitoring data to be imported (Default: "bsv4", "bsvAll", "tuberia", "inflow")
ignoredParNames	vector of parNames that are ignored for storing (Default:Temp_graphics_C, DiverBaro_graphFictive_cm, DiverBaro_cm: syntetical or calculated parameters!)

Examples

```
### xlsDir needs to be set correctly !!!!!
xlsDir <- "C:/Users/mrustl/Documents/WC_Server/DEMEAU/Work Areas/WA1 MAR/TracerSVH"
xlsFile <- "datosTOT_StVicen_jun2008-abr2009OK.xls"
xlsPath <- file.path(xlsDir, xlsFile)
##importData(xlsPath=xlsPath)

#### Loading with stored moniDat.RData object
#importData()
```

---

importShapefile *GIS: imports shapefiles & dbf*

Description

GIS: imports shapefiles & dbf

Usage

```
importShapefile(shp.path)
```

Arguments

shp.path	full path to file (with or without file extension ".shp")

Value

Imported GIS shapefile as R data.frame

importShapefiles *GIS: imports shapefiles & dbf*

Description

GIS: imports shapefiles & dbf

Usage

```
importShapefiles(shp.files = dir(path = system.file("extdata", "qgis",
  package = "kwb.demeau"), pattern = ".shp", full.names = TRUE))
```

Arguments

shp.files	vector with full paths to shapefiles "Boundary", "Ponds", "Observation wells" to be imported (with or without file extension ".shp")

Value

Imported GIS shapefiles in an R data.frame

Examples

```
shp.dir <- system.file("extdata", "qgis", package="kwb.demeau")
shp.files <- dir(path = shp.dir, pattern = ".shp", full.names = TRUE)
### Store GIS data in R data.frame "gisData"
gisData <- importShapefiles(shp.files)
### Plot imported GIS data:
```

leftBoundaryInPolygon *Helper function: check whether left model boundary lies within polygon*

Description

Helper function: check whether left model boundary lies within polygon

Usage

```
leftBoundaryInPolygon(gisData, leftBoundaryRow = 1)
```

Arguments

gisData	data.frame as retrieved by importShapefiles()
leftBoundaryRow	row number index of "gisData" which contains the left

listToMatrixForm *Monitoring: convert data to matrix forma*

Description

Monitoring: convert data to matrix forma

Usage

```
listToMatrixForm(df)
```

Arguments

df	data frame with structure like: moniDataggdailyMedian

mainLabel *Helper function for main label of comparision plot*

Description

Helper function for main label of comparision plot

Usage

```
mainLabel(model)
```

Arguments

model	object as retrieved by runHeatModel()

Value

main label

`modelConfiguration` *Model: prepare model configuration*

Description

Model: prepare model configuration

Usage

```
modelConfiguration(modelStructure, pondTemp = 20, gwTempIni = 12,
  infRate = 0.03, depthToWaterTable = 6, hydraulicGradient = 0.001,
  bnd = list(tmp = gwTempIni, ntx = 4, ntc = 1),
  hk = kwb.vs2dh::vs2dh.ConfigureGenuchten(ratioKzKh = 1, ss = 0, satKh =
  750, porosity = 0.2, alpha = 2.3, rmc = 0, beta = 5.8),
  ht = kwb.vs2dh::vs2dh.ConfigureTrans(), iniOutputTime = 1/(3600 *
  24), minSimTime = 0.5, maxSimTime = 31, outputTimeStep = 1,
  solver = kwb.vs2dh::vs2dh.ConfigureBasicSolver(),
  rSolver = kwb.vs2dh::vs2dh.ConfigureRechargePeriodSolver())
```

Arguments

`modelStructure`	as retrieved by convRealCoordinatesToNodes()
`pondTemp`	constant pond temperature (default: 20)
`gwTempIni`	initial groundwater temperature (default: 12)
`infRate`	infiltration rate per unit area (Default: 0.9514151 m/d)
`depthToWaterTable`	water table below ground level (default: 6 m)
`hydraulicGradient`	hydraulic gradient between left & right model boundary (Default: 0), if positive flow is from left to right, if negative from right to left
`bnd`	list of structure list(temp=VALUE, ntx=VALUE, ntx=VALUE) passed to function defineHeadBoundary (i.e. boundary with seepage face), if bnd=NULL left/right boundaries are no-flow boundaries
`hk`	hydraulic properties of soil as retrieved by kwb.vs2dh::vs2dh.ConfigureGenuchten()
`ht`	transport properties of soil as retrieved by kwb.vs2dh::vs2dh.ConfigureTrans()
`iniOutputTime`	automatically output results after 1 second of simulation
`minSimTime`	minimum simulation time in days (default: 0.5)
`maxSimTime`	maximum simulation time in days (default: 31)
`outputTimeStep`	at which timestep are the results printed (default: 1), i.e. each day
`solver`	general solver (Default: kwb.vs2dh::vs2dh.ConfigureBasicSolver())
`rSolver`	recharge period solver (Default: kwb.vs2dh::vs2dh.ConfigureRechargePeriodSolver())

Value

SVH model configuration

See Also

`defineHeadBoundary` for valid additional arguments

Examples

```
## Not run:
### Importing GIS features
shp.dir <- system.file("extdata", "qgis", package="kwb.demeau")
shp.files <- dir(path = shp.dir, pattern = ".shp", full.names = TRUE)
gisData <- importShapefiles(shp.files)

### Optionally remove some features
gisData <- removeFeatures(gisData = gisData, ignoreFeatureIDs = c(3,20))
modelStructure <- convRealCoordinatesToNodes(gisData = gisData)

### Model config
conf <- modelConfiguration(modelStructure = modelStructure)

### Running the configuration in VS2DH
res <- kwb.vs2dh::vs2di.runConfig(conf = conf, openTargetDir = TRUE)

### Plotting results
kwb.vs2dh::vs2dh.plotObservationPoints(
  paras = "TEMP", paraLabel = "Temperature", data = res$obsPoints
)
kwb.vs2dh::vs2dh.plotVariables(para = "Temp", data = res$variables)

## End(Not run)
```

---

modelCoordinatesToFeature

Conversion: feature parameterisation

Description

Conversion: feature parameterisation

Usage

```
modelCoordinatesToFeature(x = c(100, 220), y = 0, dx = 0.5,
  dy = 0.5, pondDepth = verticalCmToMeters(cm = 1.2), steepness = 45)
```

Arguments

x	absolute horizontal min/max coordinates of model feature. L, Default: c(100,220)
y	absolute vertical min/max coordinates of model feature. L, Default: c(0,0)
dx	dx
dy	dy
pondDepth	Depth of pond below ground level (Default: verticalCmToMeters(cm = 1.2), i.e. 3.4m, derived from Figure 8, p.11 ENSAT report)
steepness	of pond bank (in degree) only used for ponds if "pondDepth" is defined (Default: 45 degree)

Value

Parameterisation of model features

---

modelFitness *Calibration: helper function "modelFitness" (called by function modelFitnessAggregated())*

Description

Calibration: helper function "modelFitness" (called by function modelFitnessAggregated())

Usage

```
modelFitness(modelledMeasured, obsPoints = "*", ...)
```

Arguments

modelledMeasured	as retrieved by compareModelledMeasured() either $temp or $waterLevelChange
obsPoints	regular expression of observation points/wells to be included for goodness of fit calculation (Default: *, i.e. all); if only BSV-6_3, then: "BSV-6-3"
...	further arguments passed to hydroGOF::gof()

Value

matrix with goodness of fit criteria

`modelFitnessAggregated` *Calibration: helper function "modelFitnessAggregated" (called by function fitnessAdaptedModelConfiguration()*

Description

Calibration: helper function "modelFitnessAggregated" (called by function fitnessAdaptedModelConfiguration()

Usage

```
modelFitnessAggregated(modelledMeasured, obsPoints = "*",
  objCrit = "RMSE", ...)
```

Arguments

`modelledMeasured`	as retrieved by compareModelledMeasured() either $temp or $waterLevelChange
`obsPoints`	regular expression of observation points/wells to be included for goodness of fit calculation (Default: *, i.e. all); if only BSV-6_3, then: "BSV-6-3"
`objCrit`	vector with performance parameters produced by function hydroGOF:gof(), Default: "RMSE" (valid parameters: "ME", MAE", "MSE", "RMSE", "NRMSE", "PBIAS", "RSR", "rSD", "NSE", "mNSE", "rNSE", "d", "md", "rd", "cp", "r", "R2", "bR2", "KGE", "VE")
`...`	further arguments passed to hydroGOF::gof()

Value

matrix with goodness of fit criteria

`nodeIdToRowColumn` *Helper function: converting node ID to row/column*

Description

Helper function: converting node ID to row/column

Usage

```
nodeIdToRowColumn(nodeID, numberOfRows)
```

Arguments

`nodeID`	vector of node ids
`numberOfRows`	number of rows (here: nly)

Value

row (yNode) and column of node ID in grid

`orderedDataFrame` *Helper function to order data frame by two columns (col1, col2)*

Description

Helper function to order data frame by two columns (col1, col2)

Usage

```
orderedDataFrame(df, col1 = "Name", col2 = "TIME_day")
```

Arguments

`df`	data.frame
`col1`	first column to be used for ordering (Default: Name)
`col2`	second column to be used for ordering (Default: TIME_day)
`...`	further arguments passed to function order(), e.g. decreasing = TRUE

Value

ordered (default: accending) data.frame according to col1 & col2

`plotGenuchtenModels` *vanGenuchten : plot models*

Description

vanGenuchten : plot models

Usage

```
plotGenuchtenModels(models = genuchtenModels(), ...)
```

Arguments

`models`	as retrieved by genuchtenModels()
`...`	further arguments passed to lattice::xyplot()

Value

plot of all genuchten models

plotModelStructure *Plot model Structure*

Description

Plot model Structure

Usage

```
plotModelStructure(df, xColName = "x", yColName = "y")
```

Arguments

df	data.frame as retrieved by convRealToModelCoordinates() or convModelCoordinatesToNodes()
xColName	name of x column to plot
yColName	name of y column to plot

plotMonitoringData *Monitoring: plot data*

Description

Monitoring: plot data

Usage

```
plotMonitoringData(moniParName = "WaterLevel_cm",
  moniLocationPattern = "BSV|Tuberia", minDate = "2000-01-01",
  maxDate = "2015-01-01", groups = FALSE, df)
```

Arguments

moniParName	(Default: "WaterLevel_cm")
moniLocationPattern	(Default: "BSV|Tuberia")
minDate	(Default: "2000-01-01")
maxDate	(Default: "2015-01-01")
groups	(Default: FALSE)
df	data frame

plotMonitoringWithTwoYAxes *Monitoring: plot data with two y axex*

Description

Monitoring: plot data with two y axex

Usage

```
plotMonitoringWithTwoYAxes(parY1 = "WaterLevelChange_cm",
  parY2 = "Temp_C", labelParY1 = "Water level change (cm)",
  labelParY2 = "Temperature (C)", df, ...)
```

Arguments

parY1	name of parameter for Y1 axis (see: colnames(listToMatrixForm(df)), (Default: "WaterLevelChange_cm")
parY2	name of parameter for Y2 axis (see: colnames(listToMatrixForm(df)), (Default: "Temp_C")
labelParY1	user defined label for Y1 for legend (Default: "Water level change (cm)")
labelParY2	user defined label for Y2 for legend (Default: "Temperature (degree C)")
df	data.frame with structure like moniDataggdailyMedian
...	further parameters passed to xyplot()

prepareModel *Prepare model: wrapper for modelConfiguration()*

Description

Prepare model: wrapper for modelConfiguration()

Usage

```
prepareModel(gisData, type = "heat", rech_pondInfRate = 1,
  rech_pondTemp = 13.7, rech_pondConc = 1, init_gwTemp = 19,
  init_gwConc = 0, init_depthToWaterTable = 6, init_hydrGrad = 0.001,
  grid_dx = 1, grid_dy = 1, grid_yMax = NULL,
  flow_ratioKzKh = 0.01, flow_satKh = 450, flow_ss = 0,
  flow_neff = 0.2, flow_rmc = 0.05, flow_alpha = 2, flow_beta = 5,
  heat_alphaL = 1, heat_alphaT = 0.1, heat_cs = 2180000,
  heat_ktRmc = 129600, heat_ktSat = 155520, heat_cw = 4180000,
  solu_alphaL = 1, solu_alphaT = 0.1, solu_molDiffCoeff = 4.5e-05 *
  3600 * 24/100, solu_decayConst = 0, solu_bulkDensity = 0,
```

```
  solu_adsorp = 0, time_firstOutput = 1/(3600 * 24),
  time_minSimTime = 0.5, time_maxSimTime = 30.5,
  time_outputTimeStep = 1, solv_cis = TRUE, solv_cit = TRUE,
  solv_numt = 5000, solv_minit = 2, solv_itmax = 300,
  solv_eps = 0.001, solv_eps1 = 0.001, solv_eps2 = 0.001,
  solv_hmax = 0.7, solv_itstop = FALSE, solr_delt = 1e-06,
  solr_tmlt = 1.2, solr_dltmx = 1, solr_dltmin = 1e-06,
  solr_tred = 0.5, solr_dsmax = 10, solr_sterr = 0, solr_pond = 0)
```

Arguments

gisData	as retrieved by kwb.demeau::importShapefiles()
type	either "heat" (for heat model VS2DH) or "solute" (for solute transport model VSDT), Default: "heat"
rech_pondInfRate	recharge rate of infiltration pond during model simulation in meter / day (Default: 1 m/d)
rech_pondTemp	median pond temperature during recharge period (Default: 13.7 C during whole simulation time, i.e. 30 days)
rech_pondConc	median sunbstance concentration in pond during recharge period (Default: 1 during whole simulation time, i.e. 30 days)
init_gwTemp	initial groundwater temperature before infiltration (Default: 19 C)
init_gwConc	initial substance concentration in GW before infiltration (Default: 0)
init_depthToWaterTable	initial depth to groundwater table (Default: 6 m, assumption, no data)
init_hydrGrad	hydraulic gradient between left & right model boundary, if positive flow is from left to right, if negative from right to left (Default: 0.001)
grid_dx	model grid spacing in horizontal direction (Default: 1 m),
grid_dy	model grid spacing in vertical direction (Default: 1 m),
grid_yMax	maximum vertical model extent in meters (Default: NULL, i.e. maximum filter screen depth below ground level is set as maximum vertical model extent)
flow_ratioKzKh	Ratio of hydraulic conductivity in the z-coordinate direction to that in the x-coordinate direction (Default: 0.01)
flow_satKh	saturated hydraulic conductivity (Default: 450 m / day),
flow_ss	Specific storage (Ss), L^-1. (Default: 0)
flow_neff	effective porosity (Default: 0.2), sum of effective porosity and residual moisture content (rmc) are equal to parameter "porosity" used for van Genuchten model
flow_rmc	residual moisture content (Default: 0.05), sum of effective porosity and residual moisture content (rmc) are equal to parameter "porosity" used for van Genuchten model
flow_alpha	van Genuchten alpha. NOTE: alpha is as defined by van Genuchten (1980) and is the negative reciprocal of alpha' used in earlier versions (prior to version 3.0) of VS2DT, L. (Default: 2)

flow_beta — van Genuchten parameter, beta' in Healy (1990) and Lappala and others (1987), (Default: 5)

heat_alphaL — Longitudinal dispersion (Default: 1 m),

heat_alphaT — Transversal dispersion (Default: 0.1 m)

heat_cs — Heat capacity of dry solids (Cs), Q/L3 C. (Default: 2180000.0 J/m3C)

heat_ktRmc — Thermal conductivity of water sediment at residual moisture content, Q/LTC. (Default: 129600)

heat_ktSat — Thermal conductivity of water sediments at full saturation, Q/LC. (Default: 155520)

heat_cw — Heat capacity of water (Cw), which is the product of density times specific heat of water, Q/L3 C. (default: 4180000.0)

solu_alphaL — Longitudinal dispersion (Default: 1 m),

solu_alphaT — Transversal dispersion (Default: 0.1 m)

solu_molDiffCoeff — Molecular diffusion coefficient, Dm, L2/T. (Default: (4.5e-05*3600*24/100 m2/day, ### http://en.wikipedia.org/wiki/Mass_diffusivity#Liquids for hydrogen - water at 25 C)

solu_decayConst — Decay constant, l, T-1. (Default: 0)

solu_bulkDensity — Bulk density, Dm (set to 0 for no adsorption or ion exchange) M/L3. (Default: 0)

solu_adsorp — 0 for no adsorption or ion exchange; Kd for linear adsorption isotherm; K1 for Langmuir isotherm; Kf for Freundlich isotherm; or Km for ion exchange. (Default: 0)

time_firstOutput — first output after x time (default: after 1 second), (-> proxy for initial setting)

time_minSimTime — start of regular output (Default: after 0.5 days)

time_maxSimTime — end of regular output & model simulation (Default: 30.5 days)

time_outputTimeStep — time interval in days for which model results are written to output file (Default: 1 day)

solv_cis — If TRUE spatial discretisation is realised by centered-in-space differencing; if FALSE backward-in-space differencing is to be used for transport equation. (default: TRUE)

solv_cit — If TRUE temporal discretisation is realised by centered-in-time differencing; if FALSE backward-in-time or fully implicit differencing is to be used. (default: TRUE)

solv_numt — Maximum number of time steps.(default: 5000). (NOTE: if enhanced precision in print out to file "balance.out" and file 11 "obsPoints.out", is desired set NUMT equal to a negative number. That is, multiply actual maximum number of time steps by -1)1

solv_minit	Minimum number of iterations per time step. (default: 2)
solv_itmax	Maximum number of iterations per time step. (default: 300)
solv_eps	Head closure criterion for iterative solution of flow equation, L. (default: 0.001)
solv_eps1	Temperature closure criterion for iterative solution of transport equation, C. (default: 0.001)
solv_eps2	Velocity closure criterion for outer iteration loop at each time step, L/T. (default: 0.001)
solv_hmax	Relaxation parameter for iterative solution. See discussion in Lappala and others (1987) for more detail. Value is generally in the range of 0.4 to 1.2. (default: 0.7)
solv_itstop	If TRUE simulation is terminated after ITMAX iterations in one time step; otherwise = F. (default: FALSE)
solr_delt	Length of initial time step for this period, T. (default: 1.0E-6)
solr_tmlt	Multiplier for time step length. (default: 1.2)
solr_dltmx	Maximum allowed length of time step, T. (default: 1)
solr_dltmin	Minimum allowed length of time step, T. (default: 1.0E-6)
solr_tred	Factor by which time-step length is reduced if convergence is not obtained in ITMAX iterations. Values usually should be in the range 0.1 to 0.5. If no reduction of time-step length is desired, input a value of 0.0. (default: 0.1)
solr_dsmax	Maximum allowed change in head per time step for this period, L. (default: 10)
solr_sterr	Steady-state head criterion; when the maximum change in head between successive time steps is less than STERR, the program assumes that steady state has been reached for this period and advances to next recharge period, L. (default: 0)
solr_pond	Maximum allowed height of ponded water for constant flux nodes. See Lappala and other (1987) for detailed discussion of POND, L. (default: 0)

Value

Prepared SVH model configuration

Examples

```
### Importing GIS features
shp.dir <- system.file("extdata", "qgis", package="kwb.demeau")
shp.files <- dir(path = shp.dir, pattern = ".shp", full.names = TRUE)
gisData <- importShapefiles(shp.files)
### Optionally remove some features
gisData <- removeFeatures(gisData = gisData, ignoreFeatureIDs = 20)
#### 1) Prepare heat
preparedHeatModel  <- prepareModel(gisData = gisData, type = "heat")
#' #### 1) Prepare solute transport model
preparedSoluteModel  <- prepareModel(gisData = gisData, type = "solu")
```

processingData *Processing: data preprocessing*

Description

Processing: data preprocessing

Usage

```
processingData(rawData)
```

Arguments

rawData	raw data imported with importData()

removeFeatures *Helper function: remove features which should be ignored*

Description

Helper function: remove features which should be ignored

Usage

```
removeFeatures(gisData, ignoreFeatureIDs = NULL)
```

Arguments

gisData	data.frame with gis features as retrieved by importShapefile(s)()
ignoreFeatureIDs	should be a valid feature id column gisData$fID, for possible values check: unique(gisData$fID)

Value

input gisData (possibly removed by ignoredFeatureIDs)

`removeFileExtension` *Helper function: removes file extensions (copied from "kwb.quantum" package)*

Description

Helper function: removes file extensions (copied from "kwb.quantum" package)

Usage

```
removeFileExtension(x)
```

Arguments

`x`	file.path

Value

file path without extension (i.e. without ".*")

`renameValues` *Helper function: rename values*

Description

Helper function: rename values

Usage

```
renameValues(df, colName, oldVal, newVal)
```

Arguments

`df`	data.frame to be modified
`colName`	olName containing the values to be changed
`oldVal`	old value
`newVal`	new value

runHeatModel *Run heat model: wrapper for vs2di.runConfig()*

Description

Run heat model: wrapper for vs2di.runConfig()

Usage

```
runHeatModel(preparedHeatModel, tDir = tempdir(),
  openTargetDir = FALSE, ...)
```

Arguments

preparedHeatModel	prepared heat model as retrieved by prepareModel(type="heat")
tDir	target directory where vs2dh model input/output files should be stored. (Default: tempdir())
openTargetDir	If TRUE: model directory with heat model results will be opened in Windows explorer (Default: FALSE)
...	additional arguements passed to function kwb.vs2dh::vs2di.runConfig()

Value

List with heat model results ($res) and configuration

runSoluteModel *Run solute transport model: wrapper for vs2di.runConfig()*

Description

Run solute transport model: wrapper for vs2di.runConfig()

Usage

```
runSoluteModel(preparedSoluteModel, tDir = tempdir(),
  returnOutput = TRUE, openTargetDir = FALSE, ...)
```

Arguments

preparedSoluteModel	prepared solute transport model as retrieved by prepareModel(type="solu")
tDir	target directory where vs2dh model input/output files should be stored. (Default: tempdir())
returnOutput	Default: TRUE
openTargetDir	If TRUE: model directory with heat model results will be opened in Windows explorer (Default: FALSE)
...	additional arguements passed to function kwb.vs2dh::vs2di.runConfig()

Value

List with heat model results ($res) and configuration

---

selectModelled *Compare measured & modelled results*

Description

Compare measured & modelled results

Usage

```
selectModelled(modDf, parName = "TEMP", func = stats::median)
```

Arguments

modDf	modDf
parName	(Default: "TEMP")
func	(Default: median)

Value

Plot of water level, water level change & temperature of measured vs. modelled data

setFeatureIDs *Helper function: Defines feature ids for gis objectes*

Description

Aggregates gis features by shape.name & id add adds new column fID

Usage

```
setFeatureIDs(x)
```

Arguments

x	data.frame with gis features as retrieved by importShapefile()

Value

input data.frame and new column feature id (fID)

setSoilZero *Helper function: set soil ITEX to zero*

Description

Helper function: set soil ITEX to zero

Usage

```
setSoilZero(nodes, soilGrid, shape.name = "Ponds")
```

Arguments

nodes	as retrieved by convModelCoordinatesToNodes() or convRealCoordinatesToNodes()
soilGrid	soil grid (as retrieved by kwb.vs2dh::vs2dh.ConfigureSoilGrid())
shape.name	name of polygon shape files with ponds (Default: "Ponds)

Value

soil grid (with possible ITEX values set to zero)

`soluteModelled` *Compare measured & modelled resutls*

Description

Compare measured & modelled resutls

Usage

```
soluteModelled(soluteModel, offset = 0.01, toPlot = TRUE, ...)
```

Arguments

`soluteModel`	object as retrieved by runSoluteModel()
`offset`	offset (Default: 0.01) used for filtering soluteModel results (i.e. maxConc/2 +- offset)
`toPlot`	If TRUE results are plotted (Default: TRUE)
`...`	additional parameters passe to plot function lattice:xyplot()

Value

Plot/List of water level, water level change & temperature of measured vs. modelled data

`verticalCmToMeters` *Helper function: converting vertical scale into meters (Figure. 08, page 11, ENSAT report)*

Description

Helper function: converting vertical scale into meters (Figure. 08, page 11, ENSAT report)

Usage

```
verticalCmToMeters(cm)
```

Arguments

`cm`	measured centimeters in Figure 8

Value

vertical length in meters

Index