Package 'tugHall.3'

December 15, 2022

Title R-based script to simulate the cancer cell evolution

Version 3.0

Description tugHall (tumor gene-Hallmark) is a cancer-cell evolution
model simulator, wherein gene mutations are linked to the hallmarks of
cancer, which influence tumor cell behaviors.

License GPL (>= 3)

Depends R (>= 3.6.0)

Imports actuar,
graphics,
grDevices,
methods,
randomcoloR,
stats,
stringr,
withr,
utils,
dplyr,
ggplot2

Suggests rmarkdown,
knitr,
testthat,
DiagrammeR

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.1

LazyData true

VignetteBuilder knitr

R topics documented:

`add_deletion` *Function to add deletion to gene map (chromosomal location data frame)*

Description

Function to add deletion to gene map (chromosomal location data frame)

Usage

```
add_deletion(gm, Ref_start, Ref_end, Chr)
```

Arguments

`gm`	Chromosomal location data frame
`Ref_start`	Starting position of deletion
`Ref_end`	Final position of deletion
`Chr`	Chromosome name

Value

Chromosomal location data frame with additional deletion info

Examples

```
gene_map  =  tugHall_dataset$gene_map
gene_map$pnts = ''
gm_new = add_deletion( gm = gene_map, Ref_start = 112775658, Ref_end = 112775716, Chr = '5')
# to check add_del ... add_dupl, please, use diff library and:
# diff_data( gm_ref, gm, show_unchanged_columns = TRUE, always_show_order = TRUE )
```

---

add_duplication — *Function to add duplication to gene map (chromosomal location data frame)*

Description

Function to add duplication to gene map (chromosomal location data frame)

Usage

```
add_duplication(gm, Ref_start, Ref_end, Chr)
```

Arguments

`gm`	Chromosomal location data frame
`Ref_start`	Starting position of duplication
`Ref_end`	Final position of duplication
`Chr`	Chromosome name

Value

Chromosomal location data frame with additional duplication info

Examples

```
gene_map  =  tugHall_dataset$gene_map
gene_map$pnts = ''
gm_new = add_duplication( gm = gene_map, Ref_start = 112775658, Ref_end = 112775716, Chr = '5')
# to check add_del ... add_dupl, please, use diff library and:
# diff_data( gm_ref, gm, show_unchanged_columns = TRUE, always_show_order = TRUE )
```

---

add_pnt_mutation — *Function to add point mutation to data.frame gene_map (chromosomal location data frame)*

Description

Function to add point mutation to data.frame gene_map (chromosomal location data frame)

Usage

```
add_pnt_mutation(gm = gm, pos_pnt, Chr = Chr)
```

Arguments

`gm`	Chromosomal location data frame
`pos_pnt`	Position of point mutation
`Chr`	Chromosome name

Value

Chromosomal location data frame with additional point mutation info

Examples

```
gene_map = tugHall_dataset$gene_map
gene_map$pnts  =  ''
gm2 = add_pnt_mutation( gm = gene_map, pos_pnt = 112775637 , Chr = '5' )
```

`calc_binom` *Function to calculate binomial distribution including BIG NUMBERS like 10^12 and more using approximation with normal distribution*

Description

Function to calculate binomial distribution including BIG NUMBERS like 10^12 and more using approximation with normal distribution

Usage

```
calc_binom(tr, n, p)
```

Arguments

`tr`	Length of vector with successes trials
`n`	Number of independent Bernoulli trials
`p`	Probability to get successes in trials

Value

Vector of integer numbers of successes trials

Examples

```
calc_binom(tr = 3, n = 40, p = 0.9)
calc_binom(tr = 3, n = 4E20, p = 9E-9)
```

`change_allele_A_by_cna` *Function to change copy number of the allele A of the point mutation at the allele B due to CNA*

Description

Function to change copy number of the allele A of the point mutation at the allele B due to CNA

Usage

```
change_allele_A_by_cna(pnt1, start_end, t)
```

Arguments

`pnt1`	Object of class 'Point_Mutations'
`start_end`	Vector with initial and final positions of CNA
`t`	'dup' or 'del' for duplication or deletion respectively

Value

NULL, but data of pnt1 is updated due to CNA

Examples

```
pnt1 = tugHall_dataset$pnt_clones[[ 2 ]]  # pnt of allele A
start_end = c( pnt1$Phys_pos - 50 , pnt1$Phys_pos + 50  )
message( pnt1$Copy_number )
change_allele_A_by_cna( pnt1, start_end, t = 'dup' )  # View( pnt1 )
message( pnt1$Copy_number )
change_allele_A_by_cna( pnt1, start_end, t = 'del' )  # View( pnt1 )
message( pnt1$Copy_number )
```

`change_pnt_by_cna` *Function to change the point mutation due to CNA*

Description

Function to change the point mutation due to CNA

Usage

```
change_pnt_by_cna(pnt1, start_end, t)
```

Arguments

`pnt1`	Object of class 'Point_Mutations'
`start_end`	Vector with initial and final positions of CNA
`t`	'dup' or 'del' for duplication or deletion respectively

Value

NULL, but pnt1 data is updated due to CNA

Examples

```
pnt1 = tugHall_dataset$pnt_clones[[ 1 ]]
start_end = c( pnt1$Phys_pos - 50 , pnt1$Phys_pos + 50  )
message( pnt1$Copy_number )
change_pnt_by_cna( pnt1, start_end, t = 'dup' ) # View( pnt1 )
message( pnt1$Copy_number )
change_pnt_by_cna( pnt1, start_end, t = 'del' ) # View( pnt1 )
message( pnt1$Copy_number )
```

---

check_packages — *Check the installation of packages and attach them with corresponding functions*

Description

Check the installation of packages and attach them with corresponding functions

Usage

```
check_packages(pkgs = NULL)
```

Arguments

pkgs	List of package names with related function names, by default (or when pkgs = NULL) the list of packages are described in Namespace file of the package or 'R/MaxWiK-package.R' file

Value

if the packages are installed then it returns NULL else it returns error message

Examples

```
check_packages(  )
```

---

check_pkg — *Check the installation of a package for some functions*

Description

Check the installation of a package for some functions

Usage

```
check_pkg(pkg)
```

Arguments

`pkg`	Package name

Value

if the package is installed then it returns NULL else it returns error message

Examples

```
check_pkg( pkg = 'grDevices' )
```

`check_pnts` *Function to check what pnts do fall into the range?*

Description

Function to check what pnts do fall into the range?

Usage

```
check_pnts(gm_w1)
```

Arguments

`gm_w1`	A row from data.frame gene_map

Value

Return the point mutations which fall into the range

Examples

```
gene_map = tugHall_dataset$gene_map
gene_map$pnts = ''
gene_map$pnts[6] = '112792451, 112792442'
gm_w1 = gene_map[6,]
check_pnts( gm_w1 )
```

`check_previous_data` *Function to check the files from the previous simulation*

Description

Function to check the files from the previous simulation are exist and if so to move all of them to the folder with name `/Output[Time.stamp]/` , the `[Time.stamp]/` in the format `2022_10_22_15_51_09` or `year_month_day_hour_min_sec`

Usage

```
check_previous_data()
```

Value

`check_previous_data` returns NULL and renames Output folder as well as monitoring file to the folder and file with time stamp

Examples

```
NULL
```

`chk_pnt_mut` *Function to check point mutations match or don't match into duplication or deletion*

Description

Function to check point mutations match or don't match into duplication or deletion

Usage

```
chk_pnt_mut(pnt1, Ref_start, Ref_end, Chr, prntl)
```

Arguments

`pnt1`	Object of class 'Point_Mutations'
`Ref_start`	Initial position of CNA
`Ref_end`	Final position of CNA
`Chr`	Chromosome name
`prntl`	Parental chromosome 1 or 2

Value

Logical: TRUE if point mutation matches CNA, FALSE if it doesn't match

Examples

```
pnt1 = tugHall_dataset$pnt_clones[[ 5 ]]
pstn = pnt1$Phys_pos[1]
message( pstn )
prntl = pnt1$Parental_1or2
Chr = pnt1$Chr
chk_pnt_mut( pnt1 , Ref_start = pstn - 200, Ref_end = pstn + 200, Chr, prntl )
chk_pnt_mut( pnt1 , Ref_start = pstn - 200, Ref_end = pstn - 100, Chr, prntl )
```

`Clone-class` *Class 'Clone' for clones*

Description

Class 'Clone' for clones

Fields

`id` numeric. ID of a clone

`parent` numeric. Parent ID (for first - 0)

`N_cells` numeric. Number of cells in clone

`c` numeric. Split counter as average value for all cells in clone

`d` numeric. Probability of division

`i` numeric. Probability of Hayflick limit

`m` numeric. Probability that gene normal function is destroyed due to epigenome abnormality / mutation rate

`a` numeric. Probability of apoptosis for a cell in the clone

`s` numeric. Coefficient in the sigmoid function of the mutation density

`k` numeric. Probability of cell death by environment

`E` numeric. Coefficient of friction term against to the split probability.

`Nmax` numeric. Coefficient for determination the max number of cells that can exist in the primary tumor (Nmax = 1/E)

`im` numeric. Probability of the invasion/ metastatic transformation

`Ha` numeric. Apoptosis hallmark value

`Him` numeric. Invasion/ metastasis hallmark

`Hi` numeric. Mitotic restriction hallmark (immortalization hallmark)

`Hd` numeric. Growth/antigrowth hallmark (division rate hallmark)

`Hb` numeric. Angiogenesis hallmark

`gene` numeric. Vector of flags for each genes if they have driver mutation

`pasgene` numeric. Vector of flags for each genes if they have passenger mutation

`PointMut_ID` numeric. ID of point mutation in list of objects of class 'Point_Mutations'

`CNA_ID` numeric. ID of CNA mutation in list of objects of class 'CNA_Mutations'

`mutden` numeric. Gene mutation density

`invasion` logical. Indicator that clone is metastatic (invasion/metastatic transformation occured or not)

`primary` logical. Logical variable is clone primary tumor or not (normal)

`birthday` numeric. Time step of birth of clone

Examples

```
clone = tugHall_dataset$clones[[ 1 ]]
print(clone$Ha)
print(clone$N_cells)
clone$calcApoptosis()   # to calculate apoptosis death probability based on mutation density
```

---

clone_copy — *Function to make one copy for clone1 in clone_init function*

Description

Function to make one copy for clone1 in clone_init function

Usage

```
clone_copy(clone1)
```

Arguments

`clone1`	Object of class 'Clone'

Value

New object of class 'Clone' with the same info and new ID

Examples

```
clone1 = tugHall_dataset$clones[[ 1 ]]
load_tugHall.Environment( tugHall_dataset )
withr::with_environment( pck.env, code = clone_copy(clone1) )
```

CNA_Mutations-class — *Class 'CNA_Mutations'*

Description

Class 'CNA_Mutations'

Fields

`CNA_ID` numeric. ID of CNA mutation

`Parental_1or2` numeric. Parental chromosome, could be 1 or 2

`dupOrdel` character. dup for duplication or del for deletion

`Chr` character. Chromosome name

`Ref_start` numeric. Reference start position

`Ref_end` numeric. Reference final position

`Gene_names` character. Names of genes involved in CNA

`MalfunctionedByCNA` logical. True for driver mutation and False for passenger mutation

`mut_order` numeric. Order of mutations in the lists of point mutations and CNA mutations

Examples

```
cna = tugHall_dataset$cna_clones[[ 1 ]]
cna$safe()   # to save as row of data.frame
cna$copy()
cna$initialize()
cna$show()   # After initialization
```

---

copy_CNA *Function to copy CNA info*

Description

Function to copy CNA info

Usage

```
copy_CNA(CNA1)
```

Arguments

CNA1	Object of class 'CNA_Mutations'

Value

The same object of class 'CNA_Mutations'

Examples

```
cna = tugHall_dataset$cna_clones[[ 1 ]]
cna2 = copy_CNA( cna )
cna$safe()
cna2$safe()
```

---

copy_files_to_Input *Function to copy the files by default from extdata folder in the library to Input folder in the working directory*

Description

Function to copy the files by default from extdata folder in the library to Input folder in the working directory

Usage

```
copy_files_to_Input(
  files = c("CCDS.current.txt", "CF.txt", "cloneinit.txt", "gene_hallmarks.txt",
    "gene_map.txt", "parameters.txt"),
  dir = "Input"
)
```

Arguments

`files`	Files to copy, vector of names of files by default: files = c('CCDS.current.txt', 'CF.txt', 'cloneinit.txt', 'gene_hallmarks.txt','gene_map.txt','parameter)
`dir`	Folder to where files should be save, by default dir = 'Input'

Value

List of logic numbers for each copied file, TRUE - success, FALSE - not success

Examples

```
files = c('CF.txt', 'cloneinit.txt','gene_hallmarks.txt','gene_map.txt','parameters.txt' )
copy_files_to_Input( files, dir = 'Input' )
```

`copy_files_to_Output` *Function to copy the files of an example of simulation or from '/extdata/Output/' folder in the library to '/Output/' folder in the working directory*

Description

Function to copy the files of an example of simulation or from '/extdata/Output/' folder in the library to '/Output/' folder in the working directory

Usage

```
copy_files_to_Output(
  files = c("cloneout.txt", "CNA_mutations.txt", "point_mutations.txt", "gene_MAP.txt",
    "geneout.txt", "log.txt", "order_genes_dysfunction.txt", "VAF_data.txt", "VAF.txt",
      "weights.txt"),
   dir = "Output"
)
```

Arguments

`files`	Files to copy, vector of names of files by default: files = c('cloneout.txt', 'CNA_mutations.txt', 'point_mutations.txt', 'gene_MAP.txt','geneout.txt','lc 'order_genes_dysfunction.txt', 'VAF_data.txt', 'VAF.txt', 'weights.txt')
`dir`	Folder to where files should be save, by default dir = 'Output'

Value

List of logic numbers for each copied file, TRUE - success, FALSE - not success

Examples

```
files = c( 'cloneout.txt', 'CNA_mutations.txt', 'point_mutations.txt', 'gene_MAP.txt')
copy_files_to_Output( files )
files = c('geneout.txt','log.txt', 'VAF_data.txt', 'VAF.txt', 'weights.txt' )
copy_files_to_Output( files )
```

`copy_pipelines`	*Function to copy the pipelines from extdata folder in the library to /Pipelines/ folder in the working directory*

Description

Function to copy the pipelines from extdata folder in the library to /Pipelines/ folder in the working directory

Usage

```
copy_pipelines(dir = "./")
```

Arguments

`dir`	Folder to where files should be save, by default dir = './'

Value

List of logic numbers for each copied file, TRUE - success, FALSE - not success

Examples

```
copy_pipelines( dir = 'Input' )
```

`copy_pnt`	*Function to copy of point mutation info*

Description

Function to copy of point mutation info

Usage

```
copy_pnt(pnt1)
```

Arguments

`pnt1`	Object of class 'Point_Mutations'

Value

The same object of class 'Point_Mutations' with the same ID

Examples

```
pnt = tugHall_dataset$pnt_clones[[ 1 ]]
copy_pnt( pnt ) # View( pnt )
```

copy_pnt_no_mutation *Function to copy of pnt1 without mutation info for allele A*

Description

Function to copy of pnt1 without mutation info for allele A

Usage

```
copy_pnt_no_mutation(pnt1)
```

Arguments

pnt1	Object of class 'Point_Mutations'

Value

Object of class 'Point_Mutations' for another chromosome

Examples

```
pnt = tugHall_dataset$pnt_clones[[ 1 ]]
copy_pnt_no_mutation( pnt )  # View( pnt )
```

define_compaction_factor *Define compaction factor*

Description

Define compaction factor

Usage

```
define_compaction_factor(
  cf = data.frame(Ha = 1, Hb = 1, Hd = 1, Hi = 1, Him = 1),
  read_fl = TRUE,
  file_name = "./Input/CF.txt"
)
```

Arguments

cf	Data frame with compaction factors for all the hallmarks, for example, data.frame(Ha = 1, Hb = 1, Hd = 1, Hi = 1, Him = 1)
read_fl	Indicator to read file or not, logical type only
file_name	File name to rad all the parameters, it is used only if read_fl == TRUE

Value

Data frame with with compaction factors for all the hallmarks

Examples

```
copy_files_to_Input()
define_compaction_factor( read_fl = TRUE , file_name = './Input/CF.txt' )
CF1 = pck.env$CF
cf = data.frame( Ha = 0.1, Hb = 0.2, Hd = 0.7, Hi = 1, Him = 0.5 )
define_compaction_factor( cf = cf, read_fl = FALSE )  # View( c( CF, CF1 ) ) to compare
```

---

define_files_names *Function to define all the files names*

Description

Function to define all the files names

Usage

```
define_files_names(
  mainDir = getwd(),
  sbdr_Input = "Input",
  sbdr_Output = "Output"
)
```

Arguments

mainDir	Working directory for simulation, can be different from working directory of user
sbdr_Input	Sub directory for input files, by default sbdr_Input = 'Input'
sbdr_Output	Sub directory for output files, by default sbdr_Output = 'Output'

Value

NULL, but all file names are defined in GLOBAL environment

Examples

```
define_files_names()
```

---

define_gene_location *Define genes' location in chromosome*

Description

Define genes' location in chromosome

Usage

```
define_gene_location(
  file_input = "Input/CCDS.current.txt",
  genes_list = c("CCDS4107.1", "CCDS8702.1", "CCDS43171.1", "CCDS11118.1")
)
```

Arguments

`file_input`	is a name of file to input where the information about genes location is defined. That is loaded from CCDS database https://ftp.ncbi.nlm.nih.gov/pub/CCDS/current_human/
`genes_list`	is a list of genes' names like CCDS4107.1 in the CCDS database.

Value

Function returns the table of genes' locations in DNA

Examples

```
copy_files_to_Input()
define_gene_location()
file_input  =  'Input/CCDS.current.txt'
genes_list  =  c( 'CCDS4107.1', 'CCDS8702.1', 'CCDS43171.1', 'CCDS11118.1' )
define_gene_location( file_input = file_input,  genes_list = genes_list )
```

`define_parameters` *Define all the parameters for a simulation*

Description

Define all the parameters for a simulation

Usage

```
define_parameters(
  E0 = 1e-04,
  F0 = 10,
  m0 = 1e-07,
  uo = 0.9,
  us = 0.9,
  s0 = 10,
  k0 = 0.12,
  d0 = 0.4,
  ctmax = 50,
  censor_cells_number = 1e+05,
  censor_time_step = 80,
  m_dup = 1e-08,
  m_del = 1e-08,
  lambda_dup = 5000,
  lambda_del = 7000,
  uo_dup = 0.8,
  us_dup = 0.5,
  uo_del = 0,
  us_del = 0.8,
  Compaction_factor = TRUE,
 model = c("proportional_metastatic", "threshold_metastatic", "simplified")[1],
  real_time_stop = 120,
  read_fl = FALSE,
  file_name = "./Input/parameters.txt",
  n_repeat = 1000,
  monitor = TRUE,
  tumbler_for_metastasis_trial = TRUE,
  tumbler_for_apoptosis_trial = TRUE,
  tumbler_for_immortalization_trial = TRUE,
  tumbler_for_angiogenesis_trial = TRUE,
  tumbler_for_drug_intervention_trial = TRUE
)
```

Arguments

E0	Parameter in the division probability, numeric type only
F0	Parameter in the division probability, numeric type only
m0	Mutation probability for point mutation, numeric type only
uo	Oncogene mutation probability, numeric type only
us	Suppressor mutation probability, numeric type only
s0	Parameter in the sigmoid function, numeric type only
k0	Environmental death probability, numeric type only
d0	Initial probability to divide cells, numeric type only
ctmax	Hayflick limitation for cell division, integer type
censor_cells_number	Max cell number where the program forcibly stops, integer type only
censor_time_step	Max time where the program forcibly stops, integer type only
m_dup	Mutation probability for duplication, numeric type only
m_del	Mutation probability for deletion, numeric type only
lambda_dup	CNA duplication average length (of the geometrical distribution for the length), integer type only
lambda_del	CNA deletion average length (of the geometrical distribution for the length), integer type only
uo_dup	Gene malfunction probability by CNA duplication for oncogene, numeric type only
us_dup	Gene malfunction probability by CNA duplication for suppressor, numeric type only
uo_del	Gene malfunction probability by CNA deletion for oncogene, numeric type only
us_del	Gene malfunction probability by CNA deletion for suppressor, numeric type only
Compaction_factor	Logical indicator for Compaction factor CF. True means 'to use', False means 'do not use' Compaction factor for hallmarks variables
model	Name of the model to use. Can be 'proportional_metastatic' or 'threshold_metastatic' or 'simplified'
real_time_stop	Max time in seconds of running after that the program forcibly stops, integer type only
read_fl	Indicator to read file or not, logical type only
file_name	File name to rad all the parameters, it is used only if read_fl == TRUE

`n_repeat`	Max number of repetition of the program until the NON-ZERO output will be getting, integer type only
`monitor`	The indicator to make monitor file during a simulation or do not make, logical type only
`tumbler_for_metastasis_trial`	Logical parameter to turn on/off invasion/metastasis transformation trial
`tumbler_for_apoptosis_trial`	Logical parameter to turn on/off the apoptosis trial
`tumbler_for_immortalization_trial`	Logical parameter to turn on/off the immortalization trial
`tumbler_for_angiogenesis_trial`	Logical parameter to turn on/off angiogenesis trial
`tumbler_for_drug_intervention_trial`	Logical parameter to turn on/off drug intervention trial

Value

Values of all the parameters

Examples

```
copy_files_to_Input()
define_parameters( read_fl = TRUE , file_name = './Input/parameters.txt' )
define_parameters( read_fl = FALSE )
```

`drug_intervention` *Function to emulate drug intervention to pool of tumor cells by killing a part of these cells and blocking a part of clones corresponding to malfunctioned gene*

Description

Function to emulate drug intervention to pool of tumor cells by killing a part of these cells and blocking a part of clones corresponding to malfunctioned gene

Usage

```
drug_intervention(
  kill_prob = 0,
  block_prob = 1,
  gene,
  generate_mutations = TRUE
)
```

Arguments

`kill_prob`	Probability of killing cancer cells corresponding to the malfunctioned gene
`block_prob`	Probability of blocking cancer cells corresponding to the malfunctioned gene
`gene`	Name of target gene to kill and block tumor cells by a drug
`generate_mutations`	Logical to generate or not new mutations states with the same positions but for passenger genes instead drivers

Value

NULL changing clones and onco_clones objects in tugHall environment pck.env

Examples

```
NULL
```

`Environ-class` *Class 'Environ'*

Description

Class 'Environ'

Fields

`T` numeric. Time counter

`N` numeric. Number of normal cells

`P` numeric. Number of primary tumor cells

`M` numeric. Number of metastatic cells

`F` numeric. Coefficient that determines the maximal number of cells in pool of primary tumor cells

`c` numeric. Average number of divisions in pool of clones

`d` numeric. Mean value of splitting probability

`i` numeric. Average value of immortalization probability

`a` numeric. Average value of apoptosis probability

`k` numeric. Average probability of cell death via environment death

`E` numeric. Average value of coefficients of friction term

`Nmax` numeric. Maximal number of primary tumor cells that can exist in pool of clones

`im` numeric. Average value of invasion/metastasis probability

`Ha` numeric. Average value of apoptosis hallmark Ha

`Him` numeric. Average value of invasion/metastasis hallmark Him

`Hi` numeric. Average value of immortalization hallmark Hi

`Hd` numeric. Average value of growth/antigrowth hallmark Hd

`Hb` numeric. Average value of angiogenesis hallmark Hb

`type` numeric. Invasion / metastatic ratio

`gene` numeric. Cancer gene damage rate

`mutden` numeric. Average density of gene malfunction

`last_id` numeric. Maximal ID in the pool of clones.

Examples

```
env = tugHall_dataset$env
print( env )
env$initFields()
```

`foolproof` *Foolproof function allows to checking the consistency of all the input parameters*

Description

Foolproof function allows to checking the consistency of all the input parameters. It should be used just before a loop of simulation. So, all the parameters should be defined, objects onco and hall should be initialized as well. The function checks the list of parameters' names, absence of NA and NULL in the input data, self-consistency of genes names, correctness of hallmarks values, and finally, that all the necessary information is defined.

Usage

```
foolproof()
```

Examples

```
NULL
```

`generate_cna` *Function to generate object of CNA mutation*

Description

Function to generate object of CNA mutation

Usage

```
generate_cna(prntl, genes, start_end, onco1, dupOrdel)
```

Arguments

`prntl`	The 1st or 2nd parental chromosome
`genes`	Genes names
`start_end`	vector with start and final positions of CNA
`onco1`	Object of class 'OncoGene'
`dupOrdel`	It could be 'dup' or 'del' to denote duplication or deletion

Value

Object of class 'CNA_Mutations'

Examples

```
copy_files_to_Input()
load_tugHall.Environment( tugHall_dataset )
onco = tugHall_dataset$onco
gene_map = tugHall_dataset$gene_map
pnt_clones = tugHall_dataset$pnt_clones
cna_clones = tugHall_dataset$cna_clones
mut_order = 234
start_end = c(112775658, 112775716 )
withr::with_environment( env = pck.env, code = generate_cna( prntl = 1, genes = 'APC', start_end = start_end, on
```

---

generate_pnt *Function to generate an object of class 'Point_Mutations'*

Description

Function to generate an object of class 'Point_Mutations'

Usage

```
generate_pnt(prntl, gene, pos, onco1, Chr, mutation = NA)
```

Arguments

`prntl`	Parental chromosome, could be 1 or 2
`gene`	Gene name
`pos`	Position of point mutation
`onco1`	Object of class 'OncoGene'
`Chr`	Chromosome name
`mutation`	If mutation is NOT NA then MalfunctionedByPointMut = TRUE, else it is defined by corresponding probabilities

Value

Object of class 'Point_Mutations'

Examples

```
gm = tugHall_dataset$gene_map
gm_1_2 = list( gm, gm )
onco = tugHall_dataset$onco
pnt_clones = tugHall_dataset$pnt_clones
copy_files_to_Input()
load_tugHall.Environment( tugHall_dataset )
mut_order = 234  # As an example
withr::with_environment( env = pck.env, code = generate_pnt( prntl = 1, gene = 'APC', pos = 112767192, onco, Chr
```

`generate_to_copy_pnt` *Function to generate the same object of class 'Point_Mutations' with coping all information from input object*

Description

Function to generate the same object of class 'Point_Mutations' with coping all information from input object

Usage

```
generate_to_copy_pnt(pnt)
```

Arguments

`pnt`	Object of class 'Point_Mutations'

Value

The same object of class 'Point_Mutations' with different ID

Examples

```
pnt = tugHall_dataset$pnt_clones[[ 1 ]]
pnt_clones = tugHall_dataset$pnt_clones
pnt2 = generate_to_copy_pnt( pnt )
```

`gen_colors` *Function to make a large number of colors*

Description

Function to make a large number of colors

Usage

```
gen_colors(nm = 12)
```

Arguments

`nm`	Number of colors

Value

Vector of colors with length more than nm

Examples

```
clrs = gen_colors( nm = 120 )
```

get_cds_rna *Function to get length of CDS and of genes from data.frame gene_map and related probabilities*

Description

Function to get length of CDS and of genes from data.frame gene_map and related probabilities

Usage

```
get_cds_rna(gm)
```

Arguments

gm	data.frame gene_map with info about genes' location

Value

list(names, CDS, RNA, PROB, SUM, P0)

Examples

```
gene_map  =  tugHall_dataset$gene_map
load_tugHall.Environment( tugHall_dataset )
withr::with_environment( env = pck.env, code = get_cds_rna( gm = gene_map ) )
```

get_cna_mutation *Generation CNA mutation info*

Description

Generation CNA mutation info

Usage

```
get_cna_mutation(onco1, dupOrdel, gm_1_2)
```

Arguments

onco1	Object of class 'OncoGene'
dupOrdel	It could be 'dup' or 'del' to denote duplication or deletion
gm_1_2	List of two data frames (for 1st and 2nd parental chromosomes) with genes' location information

Value

List of (prntl - 1 or 2 parental chromosome, Chr - name of chromosome, genes - genes names, start_end - vector with start and end positions of CNA, w_cna - rows of CNA in gene_map data frame)

Examples

```
copy_files_to_Input()
load_tugHall.Environment( tugHall_dataset )
onco = tugHall_dataset$onco
gm = tugHall_dataset$gene_map
withr::with_environment( env = pck.env, code = get_cna_mutation( onco1 = onco, dupOrdel = 'dup', gm_1_2 = list(g
withr::with_environment( env = pck.env, code = get_cna_mutation( onco1 = onco, dupOrdel = 'del', gm_1_2 = list(g
```

`get_flow_data` *Function to get data about last simulation from cloneoutfile*

Description

Function to get data about last simulation from cloneoutfile

Usage

```
get_flow_data(
  cloneoutfile,
  genefile,
  mainDir = getwd(),
  sbdr_Output = "/Output"
)
```

Arguments

`cloneoutfile`	Name of file to read data about clone evolution
`genefile`	Name of file with hallmarks values
`mainDir`	Working directory, by default mainDir = getwd()
`sbdr_Output`	Directory for output data getting from mainDir

Value

list of data.frames like onco, hall, data_last (data of last time step),
data_avg (average data for all time steps), data_flow (data without average rows), time_max (max time step),
pnt_mut and pnt_mut_B (data.frame of point mutations for both alleles and for allele B only)
and cna_mut (data.frame of CNA mutations)

Examples

```
copy_files_to_Input()
load_tugHall.Environment( results = tugHall_dataset )
copy_files_to_Output()
withr::with_environment( env = pck.env, code = { dataset = get_flow_data(cloneoutfile, genefile, mainDir = getw
# View(dataset)
```

get_len_cds_rna — *Function to get length of CDS and whole gene from gene_map data.frame*

Description

Function to get length of CDS and whole gene from gene_map data.frame

Usage

```
get_len_cds_rna(gene_map)
```

Arguments

gene_map	data.frame with info about genes' locations

Value

list of (Name, CDS, LEN_Genes) where Name is a vector of genes' names, CDS is a vector of CDS lengths, LEN_Genes is a vector of length of whole genes including introns and exons

Examples

```
gene_map = tugHall_dataset$gene_map
onco = tugHall_dataset$onco
get_len_cds_rna( gene_map)
```

get_order_of_genes_dysfunction — *Function to get order of genes' dysfunction*

Description

Function to get order of genes' dysfunction

Usage

```
get_order_of_genes_dysfunction(
  pnt_mut,
  data_last,
  cna_mut,
  file_name = "./Output/order_genes_dysfunction.txt"
)
```

Arguments

pnt_mut	data.frame with info about all the point mutations
data_last	data.frame with data of simulation at the last time step
cna_mut	data.frame with info about all the CNA mutations
file_name	Name of file to save data

Value

data.frame of genes' dysfunction and save it in a file

Examples

```
load_tugHall.Environment( results = tugHall_dataset )
copy_files_to_Output()
dtst = get_flow_data( pck.env$cloneoutfile, pck.env$genefile )
pnt_mut   =  dtst$pnt_mut
data_last  =  dtst$data_last
cna_mut = dtst$cna_mut
file_name = './Output/order_genes_dysfunction.txt'
rdr = get_order_of_genes_dysfunction( pnt_mut, data_last, cna_mut, file_name = file_name )
```

`get_point_mutation` *Generation point mutation info*

Description

Generation point mutation info

Usage

```
get_point_mutation(onco1, gm_1_2)
```

Arguments

`onco1`	Object of class 'OncoGene'
`gm_1_2`	List of two data frames (for 1st and 2nd parental chromosomes) with genes' location information

Value

list of (prntl - 1 or 2 parental chromosome, gene - gene name, pos - position of point mutation, Chr - name of chromosome)

Examples

```
gm = tugHall_dataset$gene_map
gm_1_2 = list( gm, gm )
onco = tugHall_dataset$onco
get_point_mutation( onco, gm_1_2 )
```

get_point_mutation_for_gene

Generation point mutation info for the particular gene

Description

Generation point mutation info for the particular gene

Usage

```
get_point_mutation_for_gene(onco1, gm_1_2, gene)
```

Arguments

onco1	Object of class 'OncoGene'
gm_1_2	List of two data frames (for 1st and 2nd parental chromosomes) with genes' location information
gene	Gene's name where point mutation should be occured

Value

list of (prntl - 1 or 2 parental chromosome, gene - gene name, pos - position of point mutation, Chr - name of chromosome)

Examples

```
gm = tugHall_dataset$gene_map
gm_1_2 = list( gm, gm )
onco = tugHall_dataset$onco
get_point_mutation_for_gene( onco, gm_1_2, gene = 'APC')
get_point_mutation_for_gene( onco, gm_1_2, gene = 'KRAS')
```

get_rho_VAF

Function to get Variant allele frequencies (VAF) based on rho input parameters

Description

Function to get Variant allele frequencies (VAF) based on rho input parameters

Usage

```
get_rho_VAF(
  vf = NULL,
  rho = c(0, 0.1, 0.5),
  file_name = "./Output/VAF.txt",
  save_to_file = TRUE
)
```

Arguments

`vf`	data.frame getting from get_VAF() function
`rho`	Vector of rho parameter in the range (0,1)
`file_name`	Name of file to save VAF
`save_to_file`	Logical parameter to save or do not save data to the file. By default save_to_file = TRUE

Value

VAF for different rho with separation for metastatic cells and (primary tumor + speckled normal) cells

Examples

```
pnt_mut  =  tugHall_dataset$pnt_mut
data_last  =  tugHall_dataset$data_last
if ( !dir.exists('./Output') ) dir.create('./Output')
vf = get_VAF( pnt_mut, data_last, file_name = 'Output/VAF_data.txt')
VAF = get_rho_VAF( vf = vf, rho = c( 0.0, 0.1, 0.5 ) , file_name = './Output/VAF.txt' )
```

---

get_type — *Function to get type of the clone: normal, primary or metastatic*

Description

Function to get type of the clone: normal, primary or metastatic

Usage

```
get_type(clone1)
```

Arguments

`clone1`	Object of class ’Clone’

Value

One of characters ’normal’, ’primary’ or ’metastatic’

Examples

```
clone1 = tugHall_dataset$clones[[1]]
get_type( clone1 )
clone1 = tugHall_dataset$clones[[56]]
get_type( clone1 )
```

get_u_cna *Function to choose probability of CNA mutation for several genes*

Description

Function to choose probability of CNA mutation for several genes

Usage

```
get_u_cna(genes, dupOrdel)
```

Arguments

genes	Names of genes, vector of names
dupOrdel	It could be 'dup' or 'del' to denote duplication or deletion

Value

Single value of maximal probability from probabilities for several genes

Examples

```
copy_files_to_Input()
load_tugHall.Environment( tugHall_dataset )
onco = tugHall_dataset$onco
withr::with_environment( env = pck.env, code = get_u_cna( genes = 'APC', dupOrdel = 'dup' ) )
withr::with_environment( env = pck.env, code = get_u_cna( genes = c('KRAS','APC'), dupOrdel = c('dup', 'del') )
```

get_VAF *Function to get data about Variant allele frequencies (VAF)*

Description

Function to get data about Variant allele frequencies (VAF)

Usage

```
get_VAF(pnt_mut, data_last, file_name = "Output/VAF_data.txt")
```

Arguments

pnt_mut	data.frame with point mutation info
data_last	data.frame with data of simulation at the last time step
file_name	Name of file to save data

Value

data.frame with info about Variant allele frequencies

Examples

```
pnt_mut  =  tugHall_dataset$pnt_mut
data_last  =  tugHall_dataset$data_last
vf = get_VAF( pnt_mut, data_last, file_name = 'Output/VAF_data.txt')
```

`HallMark-class` *Class 'HallMark'*

Description

Class 'HallMark'

Fields

`Ha` numeric. Apoptosis hallmark indexes of genes in onco$name,
where onco is object of class OncoGene

`Hi` numeric. Immortalization hallmark indexes of genes in onco$name,
where onco is object of class OncoGene

`Hd` numeric. Growth/antigrowth hallmark indexes of genes in onco$name,
where onco is object of class OncoGene

`Hb` numeric. Angiogenesis hallmark indexes of genes in onco$name,
where onco is object of class OncoGene

`Him` numeric. Invasion/metastatic transformation hallmark indexes of genes in onco$name,
where onco is object of class OncoGene

`Ha_w` numeric. Apoptosis hallmark weights of genes

`Hi_w` numeric. Immortalization hallmark weights of genes

`Hd_w` numeric. Growth/antigrowth hallmark weights of genes

`Hb_w` numeric. Angiogenesis hallmark weights of genes

`Him_w` numeric. Invasion/metastatic transformation hallmark weights of genes

`notHa` numeric. Indexes of genes which are not in apoptosis hallmark

Examples

```
hall = tugHall_dataset$hall
print( hall )
hall$copy()
hall$show()
```

`init_clones` *Function to read file with initial clones*

Description

Function to read file with initial clones

Usage

```
init_clones(clonefile, clone1)
```

Arguments

`clonefile`	File to read
`clone1`	Object of class 'Clone'

Value

List of objects of class 'Clone

Examples

```
copy_files_to_Input()
load_tugHall.Environment( tugHall_dataset )
clone1 = tugHall_dataset$clones[[ 1 ]]
withr::with_environment( env = pck.env, code = init_clones(clonefile, clone1) )
```

`init_onco_clones` *Function to make list of objects of class 'OncoGene' and generate initial onco settings for all clones (onco_clones)*

Description

Function to make list of objects of class 'OncoGene' and generate initial onco settings for all clones (onco_clones)

Usage

```
init_onco_clones(onco1, clones)
```

Arguments

`onco1`	Object of class 'OncoGene'
`clones`	List of objects of class 'Clone'

Value

List of objects of class 'OncoGene'

Examples

```
copy_files_to_Input()
load_tugHall.Environment( tugHall_dataset )
clone1 = tugHall_dataset$clones[[ 1 ]]
withr::with_environment( env = pck.env, code = { clones = init_clones(clonefile, clone1)  } )
withr::with_environment( env = pck.env, code = { onco_clones = init_onco_clones( onco1 = onco, clones ) } )
```

`init_pnt_clones` *Function to generate point mutations for initial clones*

Description

Function to generate point mutations for initial clones

Usage

```
init_pnt_clones(clones, onco_clones)
```

Arguments

`clones`	List of objects of class 'Clone'
`onco_clones`	List of objects of class 'OncoGene'

Examples

```
clones = tugHall_dataset$clones
load_tugHall.Environment( tugHall_dataset )
onco = tugHall_dataset$onco
onco_clones = tugHall_dataset$onco_clones
copy_files_to_Input()
copy_files_to_Output()
define_gene_location()
pnt_clones = tugHall_dataset$pnt_clones
cna_clones = tugHall_dataset$cna_clones
mut_order = 234
## Not run:
init_pnt_clones( clones, onco_clones )  # change pnt_clones for initialization

## End(Not run)
```

`make_input_format` *Function to prepare dataset of input parameters for parallel calculations*

Description

`make_input_format()` function allows to prepare a format of dataset of input parameters from results of a trial simulation.

Usage

```
make_input_format(
  par_exclude = c("censor_cells_number", "censor_time_step", "clonefile", "cloneoutfile",
    "ctmax", "genefile", "geneoutfile", "lambda_del", "lambda_dup", "logoutfile",
     "model_name", "monitor", "n_repeat", "real_time_stop",
     "tumbler_for_metastasis_trial", "tumbler_for_apoptosis_trial",
     "tumbler_for_immortalization_trial", "tumbler_for_angiogenesis_trial",
     "tumbler_for_drug_intervention_trial")
)

make_input_range(frmt)

make_input_dataset(
  frmt,
  rng,
  n_simulations = 10,
  discrete = TRUE,
  n_graduations = 11
)
```

Arguments

`par_exclude`	List of parameters to exclude from data frame of input parameters because they will be constant for all the simulations
`frmt`	List of results of function `make_input_format()` as input format for the range of each parameter
`rng`	Data frame was gotten as a result of function `make_input_range()`
`n_simulations`	Number of rows for output data frame corresponding to a number of simulations.
`discrete`	Logical parameter, if TRUE then random values will be generated from discrete set of values, if FALSE then random values will be generated from continuous range.
`n_graduations`	Number of discrete values for parameter generation. Applicable only if discrete is TRUE.

Value

`make_input_format()` returns data frame with a single row corresponding to a set of current input parameters

`make_input_range()` returns a data frame with two rows, the first row is minimal values, and the second row is maximal values of parameters.

`make_input_dataset()` returns data frame with different sets of input parameters

Functions

- `make_input_format()`: Function to prepare a format of dataset of input parameters for parallel calculations
- `make_input_range()`: Function to make the range for each input parameter in the data frame

Examples

```
NULL
NULL
NULL
```

---

make_map *Function to make a gene_map data.frame with information of genes' locations*

Description

Function to make a gene_map data.frame with information of genes' locations

Usage

```
make_map(
  f_out = "Input/map.txt",
  ls = c("CCDS4107.1", "CCDS8702.1", "CCDS43171.1", "CCDS11118.1"),
  f_in = "Input/CCDS.current.txt"
)
```

Arguments

`f_out`	Name of file to save gene_map data.frame
`ls`	List of IDs of genes corresponding CCDS database https://ftp.ncbi.nlm.nih.gov/pub/CCDS/current_hu
`f_in`	Name of file to input downloaded from CCDS database

Value

gene_map data.frame with information of genes' locations for genes of interest

Examples

```
url = 'https://ftp.ncbi.nlm.nih.gov/pub/CCDS/current_human/CCDS.current.txt'
download.file( url = url, destfile = 'CCDS.current.txt')
ls   =  c( 'CCDS4107.1', 'CCDS8702.1', 'CCDS43171.1', 'CCDS11118.1' )
gene_map = make_map(f_out    =  'map.txt', ls = ls, f_in =  'CCDS.current.txt' )
```

---

mixed_mut_order *Function to get order of mutation for all possible types*

Description

Function to get order of mutation for all possible types

Usage

```
mixed_mut_order(clone1)
```

Arguments

`clone1`	Object of class ’Clone’

Value

data.frame with fields order, type, ID

Examples

```
clone = tugHall_dataset$clones[[ 46 ]]
clone$PointMut_ID
clone$CNA_ID
pnt_clones = tugHall_dataset$pnt_clones
cna_clones = tugHall_dataset$cna_clones
mixed_mut_order( clone )
```

`model` *Main function ’model’ to simulate clones’ evolution*

Description

Main function ’model’ to simulate clones’ evolution

Usage

```
model()

model_keep_run()
```

Value

`model()` returns the list of (clones, onco_clones), where clones - list of objects of class ’Clone’, and onco_clones - list of objects of class ’OncoGene’. During a simulation it saves data to geneoutfile.

`model_keep_run()` returns the list of (clones, onco_clones), where clones - list of objects of class ’Clone’, and onco_clones - list of objects of class ’OncoGene’. During a simulation it saves data to geneoutfile.

Functions

- `model_keep_run()`: `model_keep_run` is needed for `restart_simulation()` function

Examples

```
copy_files_to_Input()
define_files_names()
define_gene_location()
define_parameters( read_fl = TRUE , file_name = './Input/parameters.txt' )
define_compaction_factor( read_fl = TRUE , file_name = './Input/CF.txt' )
real_time_stop = 3  #  Duration of simulation time is 3 sec
## Not run:
res = model( )

## End(Not run)
NULL
```

modify_gene_map *Function to add the mutations to the data.frame gene_map*

Description

Function to add the mutations to the data.frame gene_map

Usage

```
modify_gene_map(clone1, onco1)
```

Arguments

`clone1`	Object of class 'Clone'
`onco1`	Object of class 'OncoGene'

Value

list(gm1, gm2), where gm1 and gm2 are data.frames gene_maps with mutation information

Examples

```
clone = tugHall_dataset$clones[[ 46 ]]
onco = tugHall_dataset$onco_clones[[ 46 ]]
gene_map = tugHall_dataset$gene_map
pnt_clones = tugHall_dataset$pnt_clones
cna_clones = tugHall_dataset$cna_clones
gene_map$pnts = ''
## Not run:
gm_1_2 = modify_gene_map( clone , onco )  # View(gm_1_2)

## End(Not run)
```

number_N_P_M *Function to get number of cells of a clone with indicator (normal, primary tumor or metastatic)*

Description

Function to get number of cells of a clone with indicator (normal, primary tumor or metastatic)

Usage

```
number_N_P_M(clone1)
```

Arguments

`clone1`	Object of class 'Clone'

Value

Vector c(N_normal, N_primary, N_metastatic)

Examples

```
clone1 = tugHall_dataset$clones[[ 1 ]]
number_N_P_M(clone1)
message( paste('Format is as follow: ', 'N_normal', 'N_primary', 'N_metastatic' ) )
```

OncoGene-class *Class 'OncoGene'*

Description

Class 'OncoGene'

Fields

`id` numeric. ID is same as in clone (key for clones)

`name` character. Onco genes' names list

`onsp` character. Oncogene/suppressor indicator for each gene in list of names

`len` numeric. Lengths of onco genes

`cds_1` numeric. Onco genes' CDS base lengths for parental chr 1

`cds_2` numeric. Onco genes' CDS base lengths for parental chr 2

`rna_1` numeric. Onco genes RNA base number length for parental chr 1 (exons+introns)

`rna_2` numeric. Onco genes RNA base number length for parental chr 2 (exons+introns)

`p0_1` numeric. Probability of absent of mutations for parental chr 1

`p0_2` numeric. Probability of absent of mutations for parental chr 2

`prob_1` numeric. Vector of relative probabilities for point mutation, deletion and duplication: prob = c(m0 x sumCDS, m_del x sumRNA, m_dup x sumRNA) / sum(m0 x sumCDS, m_del x sumRNA, m_dup x sumRNA)

`prob_2` numeric.

`sum_prob_1` numeric.

`sum_prob_2` numeric.

Examples

```
onco = tugHall_dataset$onco
onco$copy()
```

onco_copy *Function to make one copy for onco1 in init_onco_clones function*

Description

Function to make one copy for onco1 in init_onco_clones function

Usage

```
onco_copy(onco1)
```

Arguments

onco1	Object of class 'OncoGene'

Value

New object of class 'OncoGene' with the same info

Examples

```
onco1  =  tugHall_dataset$onco_clones[[ 1 ]]
onco2 = onco_copy( onco1 )  # ID + 1
```

onco_update *Function to update onco1 after mutation (for usage in trial_mutagenesis() function)*

Description

Function to update onco1 after mutation (for usage in trial_mutagenesis() function)

Usage

```
onco_update(onco1, gm)
```

Arguments

onco1	Object of class 'OncoGene'
gm	data.frame gene_map

Value

onco1 with updated info

Examples

```
onco1 = tugHall_dataset$onco_clones[[ 1 ]]
copy_files_to_Input()
load_tugHall.Environment( tugHall_dataset )
withr::with_environment( env = pck.env, code = onco_update( onco1, gm = list(gene_map, gene_map[1:42, ] ) ) )
# Check CDS length for TP53 gene
```

`order_gene_map` *Function to order info in gene_map data.frame with information of genes' locations*

Description

Function to order info in gene_map data.frame with information of genes' locations

Usage

```
order_gene_map(gene_map)
```

Arguments

`gene_map`	data.frame with information of genes' locations

Value

The same data.frame gene_map with ordered positions for each gene and each chromosome

Examples

```
gene_map = tugHall_dataset$gene_map
gene_map = order_gene_map( gene_map )
```

`pck.env` *Environment of the package 'tugHall.3' to store all the objects of a simulation*

Description

`pck.env` is environment of the package 'tugHall.3' where all the objects of a simulation are stored and used

`get_tugHall.Environment` function returns all the objects in the pck.env environment of the package tugHall.3

`load_tugHall.Environment` loads list 'results' that is results of simulation to the environment pck.env or tugHall.Environment

`clear_tugHall.Environment` clears the environment pck.env or tugHall.Environment

Usage

```
pck.env

get_tugHall.Environment()

load_tugHall.Environment(results)

clear_tugHall.Environment()
```

Arguments

`results`	List of results of a simulation to load to the environment pck.env or tugHall.Environment

Format

An object of class `environment` of length 0.

Value

`get_tugHall.Environment` returns all the objects in the pck.env or tugHall.Environment environment

`load_tugHall.Environment` returns NULL and loads results of simulation to the environment pck.env or tugHall.Environment

`clear_tugHall.Environment` returns NULL and clears the environment pck.env or tugHall.Environment

Functions

- `get_tugHall.Environment()`: Get results of simulation stored in pck.env or tugHall.Environment environment
- `load_tugHall.Environment()`: Load previous results of simulation to the environment pck.env or tugHall.Environment
- `clear_tugHall.Environment()`: Remove all the objects from the environment pck.env or tugHall.Environment

Examples

```
NULL
NULL
NULL
```

`plot_2D` *Function to plot 2D figure of lines*

Description

`plot_2D()` function used to plot 2D figure of points y = y(x)

`plot_2D_lines()` function returns NULL and plot 2D figure of lines from data.frame DF like `y_i` = `DF[, nl[ i ] ] )` , nl - indexes of columns

`plot_order_dysfunction()` function draw the order of genes dysfunction as a step function with number of cells related to each order

`plot_clone_evolution()` function draw the clones' evolution as cells numbers for each clone

Usage

```
plot_2D(
  x,
  y,
  names = c("X", "Y"),
  pch = 18,
  col = "blue",
  cex = 1.2,
  xr = c(-10, 10),
  yr = c(-10, 10),
  safe_pdf = FALSE,
  filename = "./plot.pdf",
  par_list = list(xpd = TRUE, cex.lab = 2, lwd = 2, mar = c(5, 5, 5, 5), tcl = 0.5,
    cex.axis = 1.75, mgp = c(2.2, 0.5, 0), font.axis = 2, font.lab = 2)
)

plot_2D_lines(
  x,
  DF,
  nl = 1:2,
  names = c("X", "Y"),
  legend_names = "",
  col = c("blue3", "darkmagenta", "red", "green4", "darkorange", "steelblue1"),
  cex = 1.2,
  lwd = 2,
  lt = c(1:6),
  xr = c(-10, 10),
  yr = c(-10, 10),
  safe_pdf = FALSE,
  filename = "./plot.pdf",
  type = "l",
  logscale = "",
  draw_key = TRUE,
  par_list = list(xpd = TRUE, cex.lab = 2, lwd = 2, mar = c(5, 5, 5, 5), tcl = 0.5,
    cex.axis = 1.75, mgp = c(2.2, 0.5, 0), font.axis = 2, font.lab = 2),
  cex.legend = 1.3
)

plot_order_dysfunction(
  rdr_dysf,
  pos = c(0, 100),
  logscale = "y",
  cex = 1,
  par_list = list(xpd = TRUE, cex.lab = 2, lwd = 2, mar = c(5, 5, 5, 5), tcl = 0.5,
    cex.axis = 1.75, mgp = c(2.2, 0.5, 0), font.axis = 2, font.lab = 2),
  cex.legend = 1.3
)

plot_clone_evolution(
  data_flow,
  threshold = c(0.05, 1),
  lwd = 2,
  hue = c(" ", "random", "red", "orange", "yellow", "green", "blue", "purple", "pink",
    "monochrome")[1],
  luminosity = c(" ", "random", "light", "bright", "dark")[5],
  yr = NA,
  add_initial = TRUE,
  log_scale = FALSE,
  par_list = list(xpd = TRUE, cex.lab = 2, lwd = 2, mar = c(5, 5, 5, 5), tcl = 0.5,
    cex.axis = 1.75, mgp = c(2.2, 0.5, 0), font.axis = 2, font.lab = 2)
)
```

Arguments

`x`	Input data for axes X
`y`	Input data for axes Y
`names`	Vector of two characters with names for X and Y axes
`pch`	Parameter pch for plot function corresponding types of dots
`col`	Vector of colors for lines or dots
`cex`	Parameter cex for plot function
`xr`	Range for X
`yr`	Range for Y
`safe_pdf`	Indicator to save plot to a file or not
`filename`	Name of file to save plot if safe_pdf == TRUE
`par_list`	List of parameters to set locally for `par()` function. By default `par_list = list( xpd=TRUE, cex.lab=2, lwd = 2, mar = c(5, 5, 5, 5), tcl = 0.5, cex.axis = 1.75, mgp = c(2.2, 0.5, 0), font.axis = 2, font.lab = 2 )`
`DF`	data.frame with data to plot
`nl`	indexes of columns in DF to plot
`legend_names`	Name of legend
`lwd`	Vector of width of lines
`lt`	Vector of types of lines
`type`	Parameter type in plot function
`logscale`	Parameter logscale in plot function, can be " or 'y' or 'x'
`draw_key`	Indicator to draw key or not
`cex.legend`	Character expansion factor for text of legend on the plot
`rdr_dysf`	Order of genes dysfunction as a data.frame
`pos`	Coordinates of list of order of genes dysfunction
`data_flow`	data.frame with results of simulation at each time step
`threshold`	Vector two numbers from 0 to 1 to show clones with relative final numbers of cells in the range of threshold
`hue`	Parameter hue in the function randomColor from library randomcoloR. `hue = c(" ", "random", "red", "orange", "yellow", "green", "blue", "purple", "pink", "monochrome")[1]`, so by default ' ' (blank space)
`luminosity`	Parameter luminosity in the function randomColor from library randomcoloR. It can be `luminosity = c(" ", "random", "light", "bright", "dark")[5]`, so by default 'dark'
`add_initial`	Logical indicator to add or do not add initial clones to plot
`log_scale`	Logical indicator to use logarithmic scale or not for Y axes

Value

`plot_2D()` function returns NULL, making 2D plot using points

NULL, making 2D plot using lines

`plot_order_dysfunction()` returns NULL making plot with step function of order of genes' dysfunction

`plot_clone_evolution()` function returns NULL making plot with clones evolution

Functions

- `plot_2D()`: Function to plot 2D figure of points y = y(x)
- `plot_order_dysfunction()`: Function to plot order of genes dysfunction as a step function with number of cells related to each order
- `plot_clone_evolution()`: Function to plot clone evolution

Examples

```
plot_2D( x=-5:5, y=-3:7 )
DF = tugHall_dataset$data_avg
plot_2D_lines( x = DF[, 1 ], DF, nl = 8:12 , xr = c(1,max(DF$Time) ), yr = c(0,1) )
xr = c(1,max(DF$Time) )
yr = c(0,max(DF[,14],DF[,16],DF[,17] ))
plot_2D_lines( x = DF[, 1 ], DF, nl = c(14,16,17) , xr =xr, yr = yr )
plot_2D_lines( x = DF[, 1 ], DF, nl = 18:22 , xr = c(1,max(DF$Time) ), yr = c(0,1) )
rdr_dysf = tugHall_dataset$rdr_dysf
plot_order_dysfunction( rdr_dysf , logscale = '', pos = c(3, 4000), cex = 1.3)
plot_order_dysfunction( rdr_dysf , logscale = 'y', pos = c(4, 400), cex = 1.2)
data_flow = tugHall_dataset$data_flow
plot_clone_evolution( data_flow, threshold = c(0.01, 1 ), add_initial = TRUE, log_scale = FALSE )
plot_clone_evolution( data_flow, threshold = c(0, 0.01 ), add_initial = FALSE, log_scale = TRUE )
```

---

plot_average_simulation_data

Function to plot main data from data.frame with average data

Description

Function to plot main data from data.frame with average data

Usage

```
plot_average_simulation_data(data_avg, time_max)
```

Arguments

`data_avg`	data.frame with average values from cloneout.txt file
`time_max`	Maximal time step in a simulation

Value

NULL, draw many plot with average data

Examples

```
data_avg = tugHall_dataset$data_avg
time_max = tugHall_dataset$time_max
plot_average_simulation_data( data_avg , time_max = time_max )
```

`plot_VAF` *Function to plot the distributions of VAF for each gene after simulation*

Description

`plot_VAF()` function draw the distributions of VAF for each gene after simulation

Usage

```
plot_VAF(
  VAF,
  rho = 0,
  violin = FALSE,
  save_to_file = FALSE,
  file_name = "./plot_VAF.pdf",
  wait_for_user = FALSE,
  y_lim = range(0, 1)
)
```

Arguments

`VAF`	is Variant allele frequencies of genes in the output format of the function `get_rho_VAF`
`rho`	is `rho` value of VAF.
`violin`	Logical parameter to draw the distribution in the form of violin or box plot. By default `violin = FALSE`, i.e. it draws in the form of box plot.
`save_to_file`	Logical parameter to save or do not save plot to the file. by default `save_to_file = FALSE`
`file_name`	Name of file to save plot. By default `file_name = './plot_VAF.pdf'`
`wait_for_user`	Logical parameter to stop at each plot or do not stop. By default `wait_for_user = FALSE`

Value

`plot_VAF()` function returns NULL making plot with VAF distributions for each gene

Examples

```
NULL
```

pnts_add_dlt *Function to subtract delta from position of point mutations*

Description

Function to subtract delta from position of point mutations

Usage

```
pnts_add_dlt(gm_w1, dlt)
```

Arguments

gm_w1	A row from data.frame gene_map
dlt	Delta to subtract from positions of point mutations

Value

Return the pnts - dlt for one row of data.frame gene_map

Examples

```
gene_map = tugHall_dataset$gene_map
gene_map$pnts = ''
gene_map$pnts[6] = '112792451'
gm_w1 = gene_map[6,]
pnts_add_dlt( gm_w1 , dlt = 1000 )
pnts_add_dlt( gm_w1 , dlt = -1001 )
```

Point_Mutations-class *Class 'Point_Mutations'*

Description

Class 'Point_Mutations'

Fields

`PointMut_ID` numeric. ID of point mutation

`Allele` character. A or B allele

`Parental_1or2` numeric. Parental chromosome, could be 1 or 2

`Chr` character. Chromosome name

`Ref_pos` numeric. Reference position

`Phys_pos` vector. Physical positions

`Delta` vector. Delta of positions

`Copy_number` numeric. Copy number of allele

`Gene_name` character. Gene's name

`MalfunctionedByPointMut` logical. True for driver mutation and False for passenger mutation

`mut_order` numeric. Number in order of mutation to reproduce the gene_map data.frame

Examples

```
pnt = tugHall_dataset$pnt_clones[[ 1 ]]
print( pnt )
pnt$copy()
pnt$show()
pnt$initialize()
pnt$show()
pnt = tugHall_dataset$pnt_clones[[ 3 ]]
pnt$safe()   # save as row of data.frame
```

`print_parameters` *Function to print GLOBAL parameters*

Description

Function to print GLOBAL parameters

Usage

```
print_parameters()
```

Value

Message with values of all the GLOBAL parameters

Examples

```
copy_files_to_Input()
define_parameters( read_fl = FALSE )
define_compaction_factor()
print_parameters()
```

`read_file` *Function to read file*

Description

Function to read file

Usage

```
read_file(file_name = "", stringsAsFactors = FALSE, header = TRUE)
```

Arguments

`file_name`	Name of file to read
`stringsAsFactors`	Parameter for read.table function, by default stringsAsFactors = FALSE
`header`	Logical type to read or do not read head of a file

Value

data.frame of data from a file

Examples

```
fl = system.file('extdata/Input', 'gene_map.txt',package = 'tugHall.3', mustWork = TRUE )
read_file(file_name = fl, stringsAsFactors = FALSE )
fl = system.file('extdata/Input', 'CF.txt',package = 'tugHall.3', mustWork = TRUE )
read_file(file_name = fl, stringsAsFactors = FALSE, header = FALSE )
```

`safe_pnt_mut` *Function to save 1 point mutation in a data frame*

Description

Function to save 1 point mutation in a data frame

Usage

```
safe_pnt_mut(pnt)
```

Arguments

`pnt`	Object of class 'Point_Mutations'

Value

data frame with 1 row of point mutation info

Examples

```
pnt = tugHall_dataset$pnt_clones[[ 1 ]]
df = safe_pnt_mut( pnt ) # View( pnt )
```

`simulation` *Simulation for lazy start with parameters from Input folder*

Description

`simulation()` makes a simulation with parameters from Input folder and results save in pck.env as well in './Results_of_simulation.RDS' file in `work_dir` folder

`restart_simulation()` is needed to start simulation from previous results with new parameter set. Parameter set can be defined as usually from Input folder or keep all the parameters excluding input list of parameters.

Usage

```
simulation(
  verbose = TRUE,
  to_plot = TRUE,
  seed = 123456,
  work_dir = getwd(),
  copy_input = TRUE
)

restart_simulation(
  loadRDS = TRUE,
  fileRDS = "./Results_of_simulation.RDS",
  loadInput = FALSE,
  change_parameters = list(censor_cells_number = 1e+06, censor_time_step = 60),
  seed = NA,
  work_dir = getwd(),
  digits = 6,
  to_plot = TRUE,
  verbose = FALSE
)
```

Arguments

`verbose`	Logical type to show or do not show messages during execution
`to_plot`	Logical type to plot or do not plot graphical results of a simulation
`seed`	Numeric type to set seed for a simulation, if seed = NA then it will be skipped
`work_dir`	Working directory for a simulation, by default `work_dir = getwd()`
`copy_input`	Logical parameter to copy or do not copy default Input folder to the simulation folder
`loadRDS`	logical to load data of previous simulation from file fileRDS. If `loadRDS = FALSE` then it loads data from pck.env that should contain the data of a simulation.
`fileRDS`	file name to load data of previous simulation, only if `loadRDS = TRUE`
`loadInput`	Logical to load parameters from Input folder or not.
`change_parameters`	List of parameters to change from the previous simulation, each parameter should be corresponding to variable name. For example, `change_parameters = list(censor_cells_numb` `= 1E06, censor_time_step = 60 )`

Value

List of results of simulation with default values for all the parameters

List of (clones, onco_clones), where clones - list of objects of class 'Clone', and onco_clones - list of objects of class 'OncoGene'. During a simulation it saves data to geneoutfile.

Functions

- `restart_simulation()`: `restart_simulation()` is needed to start simulation from previous results with new parameter set

Examples

```
# it takes a time for a simulation and then it will demonstrates results, \cr
# so, please, wait for a while
simulation( verbose = FALSE , to_plot = FALSE )
NULL
## Not run:



## End(Not run)
```

sum_cell *Aggregate data of a clone for environment object*

Description

Aggregate data of a clone for environment object

Usage

```
sum_cell(env, clones)
```

Arguments

env	Object of class 'Environ'
clones	List of all the objects of class 'Clone'

Value

NULL, but global variable env is updated

Examples

```
clones = tugHall_dataset$clones
env = tugHall_dataset$env
sum_cell(env, clones)
message( paste0('Number of primary tumor cells in the pool of clones is ', env$P ) )
message( paste0('Number of normal cells in the pool of clones is ', env$N ) )
message( paste0('Number of metastatic cells in the pool of clones is ', env$M ) )
```

sum_mutation *Serve function for sum_cell() function*

Description

Serve function for sum_cell() function

Usage

```
sum_mutation(clone1)
```

Arguments

clone1	Object of class 'Clone'

Value

vector of clone1 variables to aggregate in sum_cell() function

Examples

```
clone1 = tugHall_dataset$clones[[ 1 ]]
sum_mutation(clone1)
```

sum_N_P_M *Function to calculate N and M numbers - normal and metastatic cells*

Description

Function to calculate N and M numbers - normal and metastatic cells

Usage

```
sum_N_P_M(env, clones)
```

Arguments

env	Object of class 'Environ'
clones	List of all the objects of class 'Clone'

Value

Number of all the cells in a simulation (normal + primary tumor + metastatic)

Examples

```
clones = tugHall_dataset$clones
env = tugHall_dataset$env
env$M = 0
env$P = 0
env$N = 0  # View( env )
sum_N_P_M(env, clones)  # View( env )
message( paste(env$N, env$P, env$M ) )
```

`trial_complex` *Function trial for complex case of models*

Description

Function trial for complex case of models

Usage

```
trial_complex(clone1, onco1)
```

Arguments

`clone1`	Object of class 'Clone'
`onco1`	Object of class 'OncoGene'

Value

Number of new clones originated by clone1

Examples

```
clone1 = tugHall_dataset$clones[[ 1 ]]
onco1 = tugHall_dataset$onco
trial_complex( clone1, onco1 )
unlist( lapply( X = 1:20, FUN = function( x ) trial_complex( clone1, onco1 ) ) )
```

`trial_mutagenesis` *Function for mutagenesis trial*

Description

Function for mutagenesis trial

Usage

```
trial_mutagenesis(clone1, num_mut, onco1)
```

Arguments

`clone1`	Object of class 'Clone'
`num_mut`	Number of mutations in this NEW clone1
`onco1`	Object of class 'OncoGene' corresponding to clone1 (with the same ID)

Value

Changed object clone1, add related mutations to the lists of point mutations and/or CNA mutations

Examples

```
copy_files_to_Input()
copy_files_to_Output()
load_tugHall.Environment( tugHall_dataset )
clone1 = tugHall_dataset$clones[[ 1 ]]
onco1 = tugHall_dataset$onco_clones[[ 1 ]]
onco = tugHall_dataset$onco
define_gene_location()
pnt_clones = tugHall_dataset$pnt_clones
cna_clones = tugHall_dataset$cna_clones
mut_order = 234  # Just an example number
message( c('CNA mutation IDs ', paste(clone1$CNA_ID, collapse = ' ') ) )
message( c('Point mutation IDs ', paste(clone1$PointMut_ID, collapse = ' ') ) )
## Not run:
trial_mutagenesis( clone1, num_mut = 1, onco1 )  # it adds info to clone1
message( c('CNA mutation IDs ', paste(clone1$CNA_ID, collapse = ' ') ) )
message( c('Point mutation IDs ', paste(clone1$PointMut_ID, collapse = ' ') ) )

trial_mutagenesis( clone1, num_mut = 10, onco1 )  # it adds info to clone1
message( c('CNA mutation IDs ', paste(clone1$CNA_ID, collapse = ' ') ) )
message( c('Point mutation IDs ', paste(clone1$PointMut_ID, collapse = ' ') ) )

## End(Not run)
```

---

trial_simple *Function trial for simplified case of model*

Description

Function trial for simplified case of model

Usage

```
trial_simple(clone1, onco1)
```

Arguments

clone1	Object of class 'Clone'
onco1	Object of class 'OncoGene'

Value

Number of new clones originated by clone1

Examples

```
clone1 = tugHall_dataset$clones[[ 1 ]]
onco1 = tugHall_dataset$onco
trial_simple( clone1, onco1 )
unlist( lapply( X = 1:20, FUN = function( x ) trial_simple( clone1, onco1 ) ) )
```

`tugHall_dataset` *tugHall dataset named 'tugHall_dataset'*

Description

Dataset contains all the necessary data.frames and objects to check functions of tugHall. Description of each data.frame and object could be found in documentation to tugHall package.

Usage

```
tugHall_dataset
```

Format

A data frame with 15 data.frames/lists and 33 objects:

Input parameters 'Compaction_factor', 'E0', 'F0', 'censor_cells_number', 'censor_time_step', 'clonefile', 'cloneoutfile', 'd0', 'ctmax', 'gene_map', 'genefile', 'geneoutfile', 'k0', 'lambda_del', 'lambda_dup', 'logoutfile', 'm0', 'm_del', 'm_dup', 'model_name', 'monitor', 'n_repeat', 's0', 'real_time_stop', 'uo', 'uo_del', 'uo_dup', 'us', 'us_del', 'us_dup', 'tumbler_for_metastasis_trial', 'tumbler_for_apoptosis_trial', 'tumbler_for_immortalization_trial', 'tumbler_for_angiogenesis_trial', 'tumbler_for_drug_intervention_trial'

CF Data frame of compaction factor

Names of files and folder Names of files to input and output data: clonefile, cloneoutfile, file_monitor, genefile, geneoutfile, logoutfile, mainDir

data_flow simulation data for all time steps, data from file cloneout.txt

data_last simulation data for the last time step, data from file cloneout.txt

data_avg simulation data averaged for the each time step, data from file cloneout.txt

pnt_clones list of all the point mutations

cna_clones list of all the CNA mutations

clones list of all the clones

env list of average data for the last timestep (environment of clones)

gene_map data.frame with genes' locations information

hall Object of class 'HallMark'

onco Object of class 'OncoGene'

time_max Value of maximal time step in an example simulation

mut_order Value of integer indicator of current mutation order in the simulation

vf data.frame of preliminary data for VAF calculations

VAF data.frame with VAF values for different rho

rdr_dysf data.frame of order of genes dysfunction for each clone

`update_Hallmarks` *Function to update Hallmark and variable after division or under initialization*

Description

Function to update Hallmark and variable after division or under initialization

Usage

```
update_Hallmarks(clone1)
```

Arguments

`clone1`	Object of class 'Clone'

Value

The same object of class 'Clone' with updated fields

Examples

```
clone = tugHall_dataset$clones[[ 1 ]]
load_tugHall.Environment( tugHall_dataset )
withr::with_environment( env = pck.env, code = update_Hallmarks( clone ) )
```

`write_cloneout` *Function to write data to cloneout file at a time step*

Description

Function to write data to cloneout file at a time step

Usage

```
write_cloneout(outfile, env, clones, isFirst, onco_clones)
```

Arguments

`outfile`	File name for output info
`env`	Object of class 'Environ'
`clones`	List of objects of class 'Clone'
`isFirst`	logical type = TRUE as default
`onco_clones`	List of objects of class 'OncoGene'

Value

NULL, but add rows to output file with clone evolution data

Examples

```
env = tugHall_dataset$env
onco = tugHall_dataset$onco
if ( !dir.exists('./Output') ) dir.create('./Output')
clones = tugHall_dataset$clones
onco_clones = tugHall_dataset$onco_clones
write_header(outfile='./Output/exmpl.txt', env, onco)
write_cloneout( outfile = './Output/exmpl.txt', env, clones, isFirst = TRUE, onco_clones )
```

---

write_geneout *Function to write info about HallMark data*

Description

Function to write info about HallMark data

Usage

```
write_geneout(outfile, hall, Compaction_factor, CF)
```

Arguments

`outfile`	File name for output info
`hall`	Object of class "HallMark"
`Compaction_factor`	Compaction factor, logical type only. True means 'to use', False means 'do not use' Compaction factor for hallmarks variables
`CF`	Vector with values of compaction factor for each hallmark

Value

NULL, but data will save to a file

Examples

```
copy_files_to_Input()
load_tugHall.Environment( tugHall_dataset )
if ( !dir.exists('./Output') ) dir.create('./Output')
withr::with_environment( env = pck.env, code = write_geneout(outfile = geneoutfile, hall, Compaction_factor, C
```

`write_header` *Function to write the header to a file*

Description

Function to write the header to a file

Usage

```
write_header(outfile, env, onco)
```

Arguments

`outfile`	File name for output info
`env`	Object of class 'Environ'
`onco`	Object of class "OncoGene"

Value

NULL, but the header will save to a file and delete old info

Examples

```
env = tugHall_dataset$env
onco = tugHall_dataset$onco
if ( !dir.exists('./Output') ) dir.create('./Output')
write_header(outfile='./Output/exmpl.txt', env, onco)
```

`write_log` *Function to write log file*

Description

Function to write log file

Usage

```
write_log(
  genefile,
  clonefile,
  geneoutfile,
  cloneoutfile,
  logoutfile,
  E0,
  F0,
  m0,
  uo,
  us,
  s0,
  k0,
  ctmax,
  m_dup,
  m_del,
  lambda_dup,
  lambda_del,
  uo_dup,
  us_dup,
  uo_del,
  us_del,
  censor_cells_number,
  censor_time_step,
  d0,
  Compaction_factor,
  model_name,
  real_time_stop,
  n_repeat,
  monitor,
  tumbler_for_metastasis_trial,
  tumbler_for_apoptosis_trial,
  tumbler_for_immortalization_trial,
  tumbler_for_angiogenesis_trial,
  tumbler_for_drug_intervention_trial
)
```

Arguments

genefile	File name of initial OncoGene information
clonefile	File name of info about initial clones
geneoutfile	File name for output info about OncoGene information
cloneoutfile	File name for output info with clone evolution data
logoutfile	Name of log file with all the parameters
E0	Parameter in the division probability, numeric type only
F0	Parameter in the division probability, numeric type only
m0	Mutation probability for point mutation, numeric type only
uo	Oncogene mutation probability, numeric type only
us	Suppressor mutation probability, numeric type only
s0	Parameter in the sigmoid function, numeric type only
k0	Environmental death probability, numeric type only
ctmax	Hayflick limitation for cell division, integer type
m_dup	Mutation probability for duplication, numeric type only
m_del	Mutation probability for deletion, numeric type only
lambda_dup	CNA duplication average length (of the geometrical distribution for the length), integer type only
lambda_del	CNA deletion average length (of the geometrical distribution for the length), integer type only
uo_dup	Gene malfunction probability by CNA duplication for oncogene, numeric type only

`us_dup`	Gene malfunction probability by CNA duplication for suppressor, numeric type only
`uo_del`	Gene malfunction probability by CNA deletion for oncogene, numeric type only
`us_del`	Gene malfunction probability by CNA deletion for suppressor, numeric type only
`censor_cells_number`	Max cell number where the program forcibly stops, integer type only
`censor_time_step`	Max time where the program forcibly stops, integer type only
`d0`	Initial probability to divide cells, numeric type only
`Compaction_factor`	Compaction factor, logical type only. True means 'to use', False means 'do not use' Compaction factor for hallmarks variables
`model_name`	Name of the model to use. Can be 'proportional_metastatic' or 'threshold_metastatic' or 'simplified'
`real_time_stop`	Max time in seconds of running after that the program forcibly stops, integer type only
`n_repeat`	Max number of repetition of the program until the NON-ZERO output will be getting, integer type only
`monitor`	The indicator to make monitor file during a simulation or do not make, logical type only
`tumbler_for_metastasis_trial`	Logical parameter to turn on/off invasion/metastasis transformation trial
`tumbler_for_apoptosis_trial`	Logical parameter to turn on/off the apoptosis trial
`tumbler_for_immortalization_trial`	Logical parameter to turn on/off the immortalization trial
`tumbler_for_angiogenesis_trial`	Logical parameter to turn on/off angiogenesis trial
`tumbler_for_drug_intervention_trial`	Logical parameter to turn on/off drug intervention trial

Value

NULL, write log file to Output folder

Examples

```
copy_files_to_Input()
define_files_names()
load_tugHall.Environment( tugHall_dataset )
if ( !dir.exists('./Output') ) dir.create('./Output')
## Not run:
write_log(genefile, clonefile, geneoutfile, cloneoutfile, logoutfile,
E0, F0, m0, uo, us, s0, k0, ctmax, m_dup, m_del, lambda_dup, lambda_del,
uo_dup, us_dup, uo_del, us_del, censor_cells_number, censor_time_step, d0,
Compaction_factor, model_name, real_time_stop, n_repeat, monitor )

## End(Not run)
```

write_monitor *Function to write a simulation monitoring data into the file_monitor*

Description

Function to write a simulation monitoring data into the file_monitor

Usage

```
write_monitor(outfile, start = FALSE, env, clones)

get_VAF_clones(env, clones, pnt_clones)
```

Arguments

`outfile`	File name for output info
`start`	Indicator to start from beginning (TRUE) or not (FALSE)
`env`	Object of class 'Environ'
`clones`	List of objects of class 'Clone'
`pnt_clones`	list of point mutations usually saved in tugHall environment pck.env

Value

NULL, but info about current state of simulation will write to a file

get_VAF_clones() returns data frame same as output of get_VAF() function

Functions

- `get_VAF_clones()`: Function to get VAF info for each site during a simulation in order to get TMB - number of point mutations per 10^6 bps (per M bps)

Examples

```
env = tugHall_dataset$env
if ( !dir.exists('./Output') ) dir.create('./Output')
clones = tugHall_dataset$clones
onco_clones = tugHall_dataset$onco_clones
cna_clones = tugHall_dataset$cna_clones
pnt_clones = tugHall_dataset$pnt_clones
write_monitor( outfile = './Sim_monitoring.txt', start = TRUE , env, clones )
write_monitor( outfile = './Sim_monitoring.txt', start = FALSE , env, clones )
NULL
```

write_pnt_clones — *Function to write the point mutation info for all clones for all time steps, used at the last time step or after simulation*

Description

Function to write the point mutation info for all clones for all time steps, used at the last time step or after simulation

Usage

```
write_pnt_clones(pnt_clones, file_out = "Output/point_mutations.txt")
```

Arguments

`pnt_clones`	List of objects of class 'Point_Mutations'
`file_out`	File name to write

Value

NULL, but info will write to a file

Examples

```
pnt_clones = tugHall_dataset$pnt_clones
if ( !dir.exists('./Output') ) dir.create('./Output')
write_pnt_clones(pnt_clones, file_out = 'Output/point_mutations.txt')
```

write_weights — *Function to write info about relationship between genes and hallmarks*

Description

Function to write info about relationship between genes and hallmarks

Usage

```
write_weights(outfile, hall)

write_break_points(outfile, hall)
```

Arguments

`outfile`	File name for output info
`hall`	Object of class 'HallMark'

Value

NULL, but info about relationship between genes and hallmarks will write to a file

`write_break_points` returns NULL, but break points of weights between genes and hallmarks will write to a file

Functions

- `write_break_points()`: Function to write info about relationship between genes and hallmarks in the framework of break points

Examples

```
if ( !dir.exists('./Output') ) dir.create('./Output')
hall = tugHall_dataset$hall
onco = tugHall_dataset$onco
write_weights(outfile = './Output/weights.txt', hall)
if ( !dir.exists('./Output') ) dir.create('./Output')
hall = tugHall_dataset$hall
onco = tugHall_dataset$onco
write_break_points(outfile = './Output/break_points.txt', hall)
```

Index