

# Package: gen5helper (via r-universe)

September 7, 2024

**Type** Package

**Title** Processing 'Gen5' 2.06 Exported Data

**Version** 1.0.1

**Author** Yanxian Lin [aut, cre]

**Maintainer** Yanxian Lin <yanxian00@gmail.com>

**Description** A collection of functions for processing 'Gen5' 2.06 exported data. 'Gen5' is an essential data analysis software for BioTek plate readers  
<<https://www.biotek.com/products/software-robotics-software/gen5-microplate-reader-and-imager-software/>>.  
This package contains functions for data cleaning, modeling and plotting using exported data from 'Gen5' version 2.06. It exports technically correct data defined in (Edwin de Jonge and Mark van der Loo (2013)  
<[https://cran.r-project.org/doc/contrib/de\\_Jonge+van\\_der\\_Loo-Introduction\\_to\\_data\\_cleaning\\_with\\_R.pdf](https://cran.r-project.org/doc/contrib/de_Jonge+van_der_Loo-Introduction_to_data_cleaning_with_R.pdf)>)  
for customized analysis. It contains Boltzmann fitting for general kinetic analysis. See  
<<https://www.github.com/yanxianUCSB/gen5helper>> for more information, documentation and examples.

**Depends** dplyr, utils

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 6.1.1

**Imports** ggplot2, minpack.lm, plyr, pracma, tidyr, stats, natural sort, rlang

**Repository** <https://ylin00.r-universe.dev>

**RemoteUrl** <https://github.com/ylin00/gen5helper>

**RemoteRef** HEAD

**RemoteSha** f24b9970b90cbef15acf7083877b91f8976944d2

## Contents

as.is . . . . .	2
Boltzmann . . . . .	3
factor2num . . . . .	3
fit.boltzmann . . . . .	4
g5h.annotate . . . . .	4
g5h.clean2 . . . . .	5
g5h.clean_ . . . . .	6
g5h.set_time2 . . . . .	6
gen5helper . . . . .	7
get.halftime . . . . .	7
mapvalues_ . . . . .	8
most.freq . . . . .	8
normalize . . . . .	9
range_ . . . . .	9
saveRDS_ . . . . .	10
smooth.mean . . . . .	11
ungroup_ . . . . .	11
write.csv_ . . . . .	12

<b>Index</b>	<b>13</b>
--------------	-----------

---

as.is	<i>Cast an object to match class of another object</i>
-------	--

---

### Description

Cast an object to match class of another object

### Usage

```
as.is(x, vec)
```

### Arguments

x	object to transform
vec	object to extract class

### Examples

```
as.is(c("1", "2", "3"), 1:3)
```

---

Boltzmann

*Boltzmann model for fitting time series data*

---

### Description

Boltzmann model for fitting time series data

### Usage

```
Boltzmann(time_, val_, A0 = 1, k0 = 1, t20 = 1)
```

### Arguments

time_	time series
val_	normalized value
A0	amplitude
k0	rate constant
t20	halt time

### Value

a model

### Examples

```
Boltzmann(1:10, c(0,0,1,3,5,7,9,10,10,10), A0 = 10, k0 = 10, t20 = 5)
```

---

factor2num

*this is a file for functions that are universally useful at common data manipulations Convert factor to numeric*

---

### Description

this is a file for functions that are universally useful at common data manipulations Convert factor to numeric

### Usage

```
factor2num(x)
```

### Arguments

x	factor
---	--------

**Examples**

```
factor2num(factor(c('1', '10', '100')))
```

---

fit.boltzmann	<i>Fit readings with Boltzmann model</i>
---------------	--

---

**Description**

fit.boltzmann() using Boltzmann model to fit readings and time intervals with unit of hours, using start as initial guesses. It appends A, y0, k, t2 and val.predict, while preserving existing variables.

**Usage**

```
fit.boltzmann(.data, A0 = 1, k0 = 1, t20 = 1)
```

**Arguments**

.data	data.frame with x as time, y as value
A0	initial guess of amplititue, default 1
k0	initial guess, default 1
t20	initial guess, default 1

**Value**

data.frame with fitted parameter and predicted value

**Examples**

```
fit.boltzmann(data.frame(x=1:10,y=c(0,0,1,3,5,7,9,10,10,10)), A0 = 10, k0 = 10, t20 = 5)
```

---

g5h.annotate	<i>Annotate a cleaned dataset</i>
--------------	-----------------------------------

---

**Description**

Add time interval in hour from the oldest timestamp

**Usage**

```
g5h.annotate(.data, by = "col")
```

**Arguments**

.data            data.frame cleaned by g5h.clean()  
 by                'col' or 'row', default is 'col'. See ?g5h.gather\_col for more info.

**Value**

data.frame

**Examples**

```
# suppose "gen5_export.txt" is the export from Gen5 2.06

g5h.clean2("gen5_export.txt") %>%
  g5h.annotate()
```

---

g5h.clean2	<i>Clean Gen5 exported data</i>
------------	---------------------------------

---

**Description**

g5h.clean2() returns technically correct data.frame from Gen5 2.06 exported tab-delim data. The exported data can be generated using default export protocol in Gen5 2.06. See Gen5 User Guide for more information.

**Usage**

```
g5h.clean2(files)
```

**Arguments**

files            a vector of names of the file which the data are to be read from. If it does not contain an absolute path, the file name is relative to the current working directory, getwd().

**Value**

technically correct data.frame.

**Examples**

```
# suppose "exported_data_1.txt" and "exported_data_2.txt" are the exports from Gen5 2.06

# this line will clean one exported data
data <- g5h.clean2("exported_data_1.txt")

# this line will clean two exported data and return one appended dataset
data <- g5h.clean2(c("exported_data_1.txt", "exported_data_2.txt"))
```

---

g5h.clean_	<i>Clean Gen5 exported data</i>
------------	---------------------------------

---

**Description**

g5h.clean\_() returns technically correct data.frame from Gen5 2.06 exported tab-delim data. The exported data can be generated using default export protocol in Gen5 2.06. See Gen5 User Guide for more information.

**Usage**

```
g5h.clean_(file)
```

**Arguments**

file	the name of the file which the data are to be read from. If it does not contain an absolute path, the file name is relative to the current working directory, getwd().
------	--

**Value**

technically correct data.frame.

---

g5h.set_time2	<i>Add time intervals to cleaned dataset</i>
---------------	--

---

**Description**

g5h.set\_time() preserves existing variables and add new variable, time, which are time intervals in hours.

**Usage**

```
g5h.set_time2(.data, units = "hours")
```

**Arguments**

.data	data.frame cleaned by g5h.clean()
units	"hours" or "minutes"

**Value**

input data.frame appended with time

---

gen5helper	<i>gen5helper: A Collection of Functions for Processing Gen5 2.06 Exported Data</i>
------------	---

---

### Description

A collection of functions for processing Gen5 2.06 exported data. Gen 5 is a popular data analysis software for BioTek plate readers. This packages contains functions for data cleaning, modeling and plotting using exported data from Gen5 version 2.06. It exports technically correct data defined in (Edwin de Jonge and Mark van der Loo, 2013) for customized analysis. It contains Boltzmann fitting for general kinetic analysis. It also implement line plot and bar plot for generating publishable figures. See <https://www.github.com/yanxianUCSB/gen5helper> for more information, documentation and examples.

---

get.halftime	<i>Get half time by linear fitting</i>
--------------	--

---

### Description

get half time according to <http://www.amylofit.ch.cam.ac.uk> The algorithm for the extraction of the half times proceeds as follows: first the middle part of the curve is selected, by determining when the average over several points is first above 0.3 and when the average is last below 0.7. The number of points to be averaged over depends on the number of points in the curve. A straight line is then fitted to this middle part of the curve, the point at which it crosses the value of 0.5 is recorded as the half time. (source: DOI: nprot.2016.010)

### Usage

```
get.halftime(time, val)
```

### Arguments

time	vector of time
val	vector of values

### Value

half time

### Examples

```
get.halftime(c(1:10), c(0,1,2,3,4,5,6,7,8,9))  
get.halftime(c(1:10), c(0,0,1,3,5,7,9,10,10,10))
```

---

mapvalues\_ *Map the unique values of a vector*

---

**Description**

Map the unique values of a vector

**Usage**

```
mapvalues_(x, facts, bNaturalSort = FALSE)
```

**Arguments**

x                    factor or character  
facts                character. It maps unique(x) to facts  
bNaturalSort        binary. Whether to convert factor in natural order.

**Value**

factor

**Examples**

```
mapvalues_(c("A", "A", "B", "C"), c("one", "two", "three"))  
mapvalues_(c("apple", "apple", "banana", "pineable", "pineable"), c(3, 2, 1), bNaturalSort = TRUE)
```

---

most.freq *Return most frequent numbers*

---

**Description**

get the n most frequent elements in an array

**Usage**

```
most.freq(x, n = 1)
```

**Arguments**

x                    an array of elements  
n                    integer, default is 1

**Value**

the most n elements



**Examples**

```
most.freq(c('a', 'a', 'b', 'b', 'b', 'c'), n = 2)
most.freq(c(1, 1, 2, 3, 3, 3, 4, 4), n = 2)
```

---

normalize	<i>Scale a vector to 0-1 by min and max</i>
-----------	---

---

**Description**

Scale a vector to 0-1 by min and max

**Usage**

```
normalize(x, na.rm = TRUE)
```

**Arguments**

x	numeric
na.rm	bool whether to remove NA values.

**Value**

a normalized vector

**Examples**

```
normalize(0:10)
normalize(c(1, 100, NA, 10), na.rm = TRUE)
```

---

range_	<i>Return range of a vector</i>
--------	---------------------------------

---

**Description**

This computes the range of a vector as a value.

**Usage**

```
range_(x, na.rm = TRUE)
```

**Arguments**

x	numeric
na.rm	bool whether to remove NA values.

**Value**

numeric value

**Examples**

```
range_(c(1, 5, 10))  
range_(c(1, 5, 10, NA), na.rm = TRUE)
```

---

saveRDS_	<i>saveRDS and return .data</i>
----------	---------------------------------

---

**Description**

This function returns the object passed in, can be used for dplyr pipeline.

**Usage**

```
saveRDS_(.data, file, ...)
```

**Arguments**

.data	object to be saved
file	filename to save
...	for saveRDS

**Value**

whatever object passed in.

**Examples**

```
data <- data.frame(a=1, b=2, c=3)  
data %>%  
  saveRDS_(file.path(tempdir(), "data.rds")) %>%  
  print()
```

---

smooth.mean	<i>Smooth a vector using moving average</i>
-------------	---

---

**Description**

Smooth a vector using moving average

**Usage**

```
smooth.mean(vec, naverage)
```

**Arguments**

vec	numeric vector
naverage	width of moving average

**Value**

smoothed

**Examples**

```
smooth.mean(1:10, 2)  
smooth.mean(1:10, 3)  
smooth.mean(1:10, 5)
```

---

ungroup_	<i>Run ungroup() and as.data.frame()</i>
----------	--

---

**Description**

Run ungroup() and as.data.frame()

**Usage**

```
ungroup_(.data)
```

**Arguments**

.data	grouped data.frame
-------	--------------------

**Value**

an ungrouped data.frame()

**Examples**

```
data <- data.frame(m=c(1,2), n=c(2,3), group=c('a','b'))
data %>%
  group_by(group) %>%
  ungroup_()
```

---

`write.csv_`*write.csv without row.names*

---

**Description**

This function returns the object passed in, can be used for dplyr pipeline.

**Usage**

```
write.csv_(x, file)
```

**Arguments**

<code>x</code>	object
<code>file</code>	filename for write.csv

**Value**

whatever object passed in

**Examples**

```
write.csv_(data.frame(a=1, b=2, c=3), file.path(tempdir(), "data.csv"))

data <- data.frame(a=1, b=2, c=3)
data %>%
  file.path(tempdir(), "data.csv") %>%
  print()
```

# Index

`as.is`, 2

Boltzmann, 3

`factor2num`, 3

`fit.boltzmann`, 4

`g5h.annotate`, 4

`g5h.clean2`, 5

`g5h.clean_`, 6

`g5h.set_time2`, 6

`gen5helper`, 7

`gen5helper-package (gen5helper)`, 7

`get.halftime`, 7

`mapvalues_`, 8

`most.freq`, 8

`normalize`, 9

`range_`, 9

`saveRDS_`, 10

`smooth.mean`, 11

`ungroup_`, 11

`write.csv_`, 12