Package 'iglu'

July 1, 2020

Type Package

Title Interpreting Glucose Data from Continuous Glucose Monitors

Version 1.0.2

Description Implements a wide range of metrics for measuring glucose control and glucose variability based on continuous glucose monitoring data. The list of implemented metrics is summarized in Rodbard (2009) <doi:10.1089/dia.2009.0015>. Additional visualization tools include time-series and lasagna plots.

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Depends R (>= 3.1.0)

Imports caTools, scales, stats, ggplot2, lubridate, shiny, dplyr, magrittr, tibble, tidyr

Suggests knitr, rmarkdown, testthat (>= 2.1.0)

VignetteBuilder knitr

NeedsCompilation no

Author Steve Broll [aut], Jacek Urbanek [aut], David Buchanan [aut], John Muschelli [aut], Irina Gaynanova [aut, cre]

Maintainer Irina Gaynanova <irinag@stat.tamu.edu>

Repository CRAN

Date/Publication 2020-07-01 12:30:06 UTC

R topics documented:

above_percent 2
adrr 3

below_percent . . . 4
CGMS2DayByDay . . . 5
conga . . . 6
cv_glu . . . 7
example_data_1_subject . . . 8
example_data_5_subject . . . 9
grade . . . 9
grade_eugly . . . 10
grade_hyper . . . 11
grade_hypo . . . 12
hbgi . . . 13
hyper_index . . . 14
hypo_index . . . 15
igc . . . 17
iglu_shiny . . . 18
in_range_percent . . . 18
iqr_glu . . . 19
j_index . . . 20
lbgi . . . 21
mage . . . 22
mean_glu . . . 23
median_glu . . . 24
modd . . . 25
m_value . . . 26
plot_glu . . . 27
plot_lasagna . . . 28
plot_lasagna_1subject . . . 30
quantile_glu . . . 31
range_glu . . . 32
sd_glu . . . 33
sd_measures . . . 33
summary_glu . . . 35

Index **37**

`above_percent` *Calculate percentage of values above target thresholds*

Description

The function above_percent produces a tibble object with values equal to the percentage of glucose measurements above target values. The output columns correspond to the subject id followed by the target values and the output rows correspond to the subjects. The values will be between 0 (no measurements) and 100 (all measurements).

Usage

```
above_percent(data, targets_above = c(140, 180, 200, 250))
```

Arguments

`data`	DataFrame object with column names "id", "time", and "gl", or numeric vector of glucose values.
`targets_above`	Numeric vector of glucose thresholds. Glucose values from data argument will be compared to each value in the targets_above vector. Default list is (140, 180, 200, 250).

Details

A tibble object with 1 row for each subject, a column for subject id and column for each target value is returned. NA's will be omitted from the glucose values in calculation of percent.

Value

If a data.frame object is passed, then a tibble object with a column for subject id and then a column for each target value is returned. If a vector of glucose values is passed, then a tibble object without the subject id is returned. as.numeric() can be wrapped around the latter to output a numeric vector.

References

Rodbard (2009) Interpretation of continuous glucose monitoring data: glycemic variability and quality of glycemic control, *Diabetes Technology and Therapeutics* **11** .55-67, doi: 10.1089/dia.2008.0132.

Examples

```
data(example_data_1_subject)

above_percent(example_data_1_subject)
above_percent(example_data_1_subject, targets_above = c(100, 150, 180))

data(example_data_5_subject)

above_percent(example_data_5_subject)
above_percent(example_data_5_subject, targets_above = c(70, 170))
```

`adrr` *Calculate average daily risk range (ADRR)*

Description

The function adrr produces ADRR values in a tibble object.

Usage

```
adrr(data)
```

Arguments

`data`	DataFrame object with column names "id", "time", and "gl".

Details

A tibble object with 1 row for each subject, a column for subject id and a column for ADRR values is returned. NA glucose values are omitted from the calculation of the ADRR values.

ADRR is the average sum of HBGI corresponding to the highest glucose value and LBGI corresponding to the lowest glucose value for each day, with the average taken over the daily sums. If there are no high glucose or no low glucose values, then 0 will be substituted for the HBGI value or the LBGI value, respectively, for that day.

Value

A tibble object with two columns: subject id and corresponding ADRR value.

References

Kovatchev et al. (2006) Evaluation of a New Measure of Blood Glucose Variability in, Diabetes *Diabetes care* **29** .2433-2438, doi: 10.2337/dc061085.

Examples

```
data(example_data_1_subject)
adrr(example_data_1_subject)

data(example_data_5_subject)
adrr(example_data_5_subject)
```

`below_percent` *Calculate percentage below targeted values*

Description

#' @description The function below_percent produces a tibble object with values equal to the percentage of glucose measurements below target values. The output columns correspond to the subject id followed by the target values and the output rows correspond to the subjects. The values will be between 0 (no measurements) and 100 (all measurements).

Usage

```
below_percent(data, targets_below = c(50, 80))
```

Arguments

data	DataFrame with column names ("id", "time", and "gl"), or numeric vector of glucose values.
targets_below	Numeric vector of glucose thresholds. Glucose values from data argument will be compared to each value in the targets_below vector. Default list is (50, 80).

Details

A tibble object with 1 row for each subject, a column for subject id and column for each target value is returned. NA's will be omitted from the glucose values in calculation of percent.

Value

If a data.frame object is passed, then a tibble object with a column for subject id and then a column for each target value is returned. If a vector of glucose values is passed, then a tibble object without the subject id is returned. as.numeric() can be wrapped around the latter to output a numeric vector.

References

Rodbard (2009) Interpretation of continuous glucose monitoring data: glycemic variability and quality of glycemic control, *Diabetes Technology and Therapeutics* **11** .55-67, doi: 10.1089/dia.2008.0132.

Examples

```
data(example_data_1_subject)

below_percent(example_data_1_subject)
below_percent(example_data_1_subject, targets_below = c(50, 100, 180))

data(example_data_5_subject)

below_percent(example_data_5_subject)
below_percent(example_data_5_subject, targets_below = c(80, 180))
```

CGMS2DayByDay *Interpolate glucose value on an equally spaced grid from day to day*

Description

Interpolate glucose value on an equally spaced grid from day to day

Usage

```
CGMS2DayByDay(data, dt0 = NULL, inter_gap = 45, tz = "")
```

Arguments

data	DataFrame object with column names "id", "time", and "gl". Should only be data for 1 subject. In case multiple subject ids are detected, the warning is produced and only 1st subject is used.
dt0	The time frequency for interpolation in minutes, the default will match the CGM meter's frequency (e.g. 5 min for Dexcom).
inter_gap	The maximum allowable gap (in minutes) for interpolation. The values will not be interpolated between the glucose measurements that are more than inter_gap minutes apart. The default value is 45 min.
tz	A character string specifying the time zone to be used. System-specific (see as.POSIXct), but " " is the current time zone, and "GMT" is UTC (Universal Time, Coordinated). Invalid values are most commonly treated as UTC, on some platforms with a warning.

Value

A list with

gd2d	A matrix of glucose values with each row corresponding to a new day, and each column corresponding to time
actual_dates	A vector of dates corresponding to the rows of gd2d
dt0	Time frequency of the resulting grid, in minutes

Examples

```
CGMS2DayByDay(example_data_1_subject)
```

---

conga *Calculate continuous overall net glycemic action (CONGA)*

Description

The function conga produces a CONGA values a tibble object. conga currently only supports calculation of CONGA24.

Usage

```
conga(data, tz = "")
```

Arguments

data	DataFrame object with column names "id", "time", and "gl".
tz	A character string specifying the time zone to be used. System-specific (see as.POSIXct), but " " is the current time zone, and "GMT" is UTC (Universal Time, Coordinated). Invalid values are most commonly treated as UTC, on some platforms with a warning.

Details

A tibble object with 1 row for each subject, a column for subject id and a column for the CONGA values is returned.

Missing values will be linearly interpolated when close enough to non-missing values.

CONGA_n is the standard deviation of the difference between glucose values that are exactly n hours apart. CONGA_24 is currently the only supported CONGA type (n = 24), and is computed by taking the standard deviation of differences in measurements separated by 24 hours.

Value

A tibble object with two columns: subject id and corresponding CONGA value.

References

McDonnell et al. (2005) : A novel approach to continuous glucose analysis utilizing glycemic variation *Diabetes Technology and Therapeutics* **7** .253-263, doi: 10.1089/dia.2005.7.253.

Examples

```
data(example_data_1_subject)
conga(example_data_1_subject)

data(example_data_5_subject)
conga(example_data_5_subject)
```

`cv_glu` *Calculate Coefficient of Variation (CV) of glucose levels*

Description

The function cv_glu produces CV values in a tibble object.

Usage

```
cv_glu(data)
```

Arguments

`data`	DataFrame object with column names "id", "time", and "gl", or numeric vector of glucose values.

Details

A tibble object with 1 row for each subject, a column for subject id and a column for CV values is returned. NA glucose values are omitted from the calculation of the CV.

CV (Coefficient of Variation) is calculated by $100 * sd(BG)/mean(BG)$ Where BG is the list of all Blood Glucose measurements for a subject.

Value

If a data.frame object is passed, then a tibble object with two columns: subject id and corresponding CV value is returned. If a vector of glucose values is passed, then a tibble object with just the CV value is returned. as.numeric() can be wrapped around the latter to output just a numeric value.

References

Rodbard (2009) Interpretation of continuous glucose monitoring data: glycemic variability and quality of glycemic control, *Diabetes Technology and Therapeutics* **11** .55-67, doi: 10.1089/dia.2008.0132.

Examples

```
data(example_data_1_subject)
cv_glu(example_data_1_subject)

data(example_data_5_subject)
cv_glu(example_data_5_subject)
```

---

example_data_1_subject

Example CGM data for one subject with Type II diabetes

Description

Dexcom G4 CGM measurements from 1 subject with Type II diabetes, this is a subset of example_data_5_subject.

Usage

```
example_data_1_subject
```

Format

A data.frame with 2915 rows and 3 columns, which are:

id identifier of subject

time 5-10 minute time value

gl glucose level

example_data_5_subject *Example CGM data for 5 subjects with Type II diabetes*

Description

Dexcom G4 CGM measurements for 5 subjects with Type II diabetes. These data are part of a larger study sample that consisted of patients with Type 2 diabetes recruited from the general community. To be eligible, patients with Type 2 diabetes, not using insulin therapy and with a glycosylated hemoglobin (HbA$_1c$) value at least 6.5

Usage

```
example_data_5_subject
```

Format

A data.frame with 13866 rows and 3 columns, which are:

id identifier of subject

time date and time stamp

gl glucose level as measured by CGM (mg/dL)

grade *Calculate mean GRADE score*

Description

The function grade produces GRADE score values in a tibble object.

Usage

```
grade(data)
```

Arguments

data	DataFrame object with column names "id", "time", and "gl", or numeric vector of glucose values.

Details

A tibble object with 1 row for each subject, a column for subject id and a column for GRADE values is returned. NA glucose values are omitted from the calculation of the GRADE.

GRADE score is calculated by $1/n * \sum[425 * (log(log(BG_i/18)) + .16)^2]$ Where BG_i is the ith Blood Glucose measurement and n is the total number of measurements.

Value

If a data.frame object is passed, then a tibble object with two columns: subject id and corresponding GRADE value is returned. If a vector of glucose values is passed, then a tibble object with just the GRADE value is returned. as.numeric() can be wrapped around the latter to output just a numeric value.

References

Hill et al. (2007): A method for assessing quality of control from glucose profiles *Diabetic Medicine* **24** .753-758, doi: 10.1111/j.14645491.2007.02119.x.

Examples

```
data(example_data_1_subject)
grade(example_data_1_subject)

data(example_data_5_subject)
grade(example_data_5_subject)
```

---

grade_eugly *Percentage of GRADE score attributable to target range*

Description

The function grade_eugly produces %GRADE euglycemia values in a tibble object.

Usage

```
grade_eugly(data, lower = 70, upper = 140)
```

Arguments

data	DataFrame object with column names "id", "time", and "gl", or numeric vector of glucose values.
lower	Lower bound used for hypoglycemia cutoff, in mg/dL. Default is 70
upper	Upper bound used for hyperglycemia cutoff, in mg/dL. Default is 140.

Details

A tibble object with 1 row for each subject, a column for subject id and a column for %GRADE euglycemia values is returned. NA glucose values are omitted from the calculation of the %GRADE euglycemia values.

%GRADE euglycemia is determined by calculating the percentage of GRADE score (see grade function) attributed to values in the target range, i.e. values not below hypoglycemic or above hyperglycemic cutoffs.

Value

If a data.frame object is passed, then a tibble object with two columns: subject id and corresponding %GRADE euglycemia value is returned. If a vector of glucose values is passed, then a tibble object with just the %GRADE euglycemia value is returned. as.numeric() can be wrapped around the latter to output just a numeric value.

References

Hill et al. (2007): A method for assessing quality of control from glucose profiles *Diabetic Medicine* **24** .753-758, doi: 10.1111/j.14645491.2007.02119.x.

Examples

```
data(example_data_1_subject)
grade_eugly(example_data_1_subject)
grade_eugly(example_data_1_subject, lower = 80, upper = 180)

data(example_data_5_subject)
grade_eugly(example_data_5_subject)
grade_eugly(example_data_5_subject, lower = 80, upper = 160)
```

---

grade_hyper *Percentage of GRADE score attributable to hyperglycemia*

Description

The function grade_hyper produces %GRADE hyperglycemia values in a tibble object.

Usage

```
grade_hyper(data, upper = 140)
```

Arguments

data	DataFrame object with column names "id", "time", and "gl", or numeric vector of glucose values.
upper	Upper bound used for hyperglycemia cutoff, in mg/dL. Default is 140.

Details

A tibble object with 1 row for each subject, a column for subject id and a column for %GRADE hyperglycemia values is returned. NA glucose values are omitted from the calculation of the %GRADE hyperglycemia values.

%GRADE hyperglycemia is determined by calculating the percentage of GRADE score (see grade function) attributed to hyperglycemic glucose values.

Value

If a data.frame object is passed, then a tibble object with two columns: subject id and corresponding %GRADE hyperglycemia value is returned. If a vector of glucose values is passed, then a tibble object with just the %GRADE hyperglycemia value is returned. as.numeric() can be wrapped around the latter to output just a numeric value.

References

Hill et al. (2007): A method for assessing quality of control from glucose profiles *Diabetic Medicine* **24** .753-758, doi: 10.1111/j.14645491.2007.02119.x.

Examples

```
data(example_data_1_subject)
grade_hyper(example_data_1_subject)
grade_hyper(example_data_1_subject, upper = 180)

data(example_data_5_subject)
grade_hyper(example_data_5_subject)
grade_hyper(example_data_5_subject, upper = 160)
```

---

grade_hypo *Percentage of GRADE score attributable to hypoglycemia*

Description

The function grade_hypo produces %GRADE hypoglycemia values in a tibble object.

Usage

```
grade_hypo(data, lower = 80)
```

Arguments

data	DataFrame object with column names "id", "time", and "gl", or numeric vector of glucose values.
lower	Lower bound used for hypoglycemia cutoff, in mg/dL. Default is 80

Details

A tibble object with 1 row for each subject, a column for subject id and a column for %GRADE hypoglycemia values is returned. NA glucose values are omitted from the calculation of the %GRADE hypoglycemia values.

%GRADE hypoglycemia is determined by calculating the percentage of GRADE score (see grade function) attributed to hypoglycemic glucose values.

Value

If a data.frame object is passed, then a tibble object with two columns: subject id and corresponding %GRADE hypoglycemia value is returned. If a vector of glucose values is passed, then a tibble object with just the %GRADE hypoglycemia value is returned. as.numeric() can be wrapped around the latter to output just a numeric value.

References

Hill et al. (2007): A method for assessing quality of control from glucose profiles *Diabetic Medicine* **24** .753-758, doi: 10.1111/j.14645491.2007.02119.x.

Examples

```
data(example_data_1_subject)
grade_hypo(example_data_1_subject)
grade_hypo(example_data_1_subject, lower = 70)

data(example_data_5_subject)
grade_hypo(example_data_5_subject)
grade_hypo(example_data_5_subject, lower = 65)
```

---

hbgi *Calculate High Blood Glucose Index (HBGI)*

Description

The function hbgi produces HBGI values in a tibble object.

Usage

```
hbgi(data)
```

Arguments

`data`	DataFrame object with column names "id", "time", and "gl", or numeric vector of glucose values.

Details

A tibble object with 1 row for each subject, a column for subject id and a column for HBGI values is returned. NA glucose values are omitted from the calculation of the HBGI.

HBGI is calculated by $1/n * \sum(10 * fbg_i^2)$, where $fbg_i = max(0, 1.509 * (log(BG_i)^{1.084} - 5.381)$, BG_i is the ith Blood Glucose measurement for a subject, and n is the total number of measurements for that subject.

Value

If a data.frame object is passed, then a tibble object with two columns: subject id and corresponding HBGI value is returned. If a vector of glucose values is passed, then a tibble object with just the HBGI value is returned. as.numeric() can be wrapped around the latter to output just a numeric value.

References

Kovatchev et al. (2006) Evaluation of a New Measure of Blood Glucose Variability in, Diabetes *Diabetes care* **29** .2433-2438, doi: 10.2337/dc061085.

Examples

```
data(example_data_1_subject)
hbgi(example_data_1_subject)

data(example_data_5_subject)
hbgi(example_data_5_subject)
```

---

hyper_index *Calculate Hyperglycemia Index*

Description

The function hyper_index produces Hyperglycemia Index values in a tibble object.

Usage

```
hyper_index(data, ULTR = 140, a = 1.1, c = 30)
```

Arguments

data	DataFrame object with column names "id", "time", and "gl", or numeric vector of glucose values.
ULTR	Upper Limit of Target Range, default value is 140 mg/dL.
a	Exponent, generally in the range from 1.0 to 2.0, default value is 1.1.
c	Scaling factor, to display Hyperglycemia Index, Hypoglycemia Index, and IGC on approximately the same numerical range as measurements of HBGI, LBGI and GRADE, default value is 30.

Details

A tibble object with 1 row for each subject, a column for subject id and a column for the Hyperglycemia Index values is returned. NA glucose values are omitted from the calculation of the Hyperglycemia Index values.

Hyperglycemia Index is calculated by $n/c * \sum[(hyperBG_j - ULTR)^a]$ Here n is the total number of Blood Glucose measurements (excluding NA values), $hyperBG_j$ is the jth Blood Glucose measurement above the ULTR cutoff, a is an exponent, and c is a scaling factor.

Value

If a data.frame object is passed, then a tibble object with two columns: subject id and corresponding Hyperglycemia Index value is returned. If a vector of glucose values is passed, then a tibble object with just the Hyperglycemia Index value is returned. as.numeric() can be wrapped around the latter to output just a numeric value.

References

Rodbard (2009) Interpretation of continuous glucose monitoring data: glycemic variability and quality of glycemic control, *Diabetes Technology and Therapeutics* **11** .55-67, doi: 10.1089/dia.2008.0132.

Examples

```
data(example_data_1_subject)
hyper_index(example_data_1_subject)
hyper_index(example_data_1_subject, ULTR = 160)

data(example_data_5_subject)
hyper_index(example_data_5_subject)
hyper_index(example_data_5_subject, ULTR = 150)
```

---

hypo_index *Calculate Hypoglycemia Index*

Description

The function hypo_index produces Hypoglycemia index values in a tibble object.

Usage

```
hypo_index(data, LLTR = 80, b = 2, d = 30)
```

Arguments

data	DataFrame object with column names "id", "time", and "gl", or numeric vector of glucose values.
LLTR	Lower Limit of Target Range, default value is 80 mg/dL.
b	Exponent, generally in the range from 1.0 to 2.0, default value is 2.
d	Scaling factor,to display Hyperglycemia Index, Hypoglycemia Index, and IGC on approximately the same numerical range as measurements of HBGI, LBGI and GRADE, default value is 30.

Details

A tibble object with 1 row for each subject, a column for subject id and a column for the Hypoglycemia Index values is returned. NA glucose values are omitted from the calculation of the Hypoglycemia Index values.

Hypoglycemia Index is calculated by $n/d * \sum[(LLTR - hypoBG_j)^b]$ Here n is the total number of Blood Glucose measurements (excluding NA values), and $hypoBG_j$ is the jth Blood Glucose measurement below the LLTR cutoff, b is an exponent, and d is a scaling factor.

Value

If a data.frame object is passed, then a tibble object with two columns: subject id and corresponding Hypoglycemia Index value is returned. If a vector of glucose values is passed, then a tibble object with just the Hypoglycemia Index value is returned. as.numeric() can be wrapped around the latter to output just a numeric value.

References

Rodbard (2009) Interpretation of continuous glucose monitoring data: glycemic variability and quality of glycemic control, *Diabetes Technology and Therapeutics* **11** .55-67, doi: 10.1089/dia.2008.0132.

Examples

```
data(example_data_1_subject)
hypo_index(example_data_1_subject, LLTR = 60)

data(example_data_5_subject)
hypo_index(example_data_5_subject)
hypo_index(example_data_5_subject, LLTR = 70)
```

igc *Calculate Index of Glycemic Control*

Description

The function igc produces IGC values in a tibble object.

Usage

```
igc(data, LLTR = 80, ULTR = 140, a = 1.1, b = 2, c = 30, d = 30)
```

Arguments

data	DataFrame object with column names "id", "time", and "gl", or numeric vector of glucose values.
LLTR	Lower Limit of Target Range, default value is 80 mg/dL.
ULTR	Upper Limit of Target Range, default value is 140 mg/dL.
a	Exponent, generally in the range from 1.0 to 2.0, default value is 1.1.
b	Exponent, generally in the range from 1.0 to 2.0, default value is 2.
c	Scaling factor, to display Hyperglycemia Index, Hypoglycemia Index, and IGC on approximately the same numerical range as measurements of HBGI, LBGI and GRADE, default value is 30.
d	Scaling factor,to display Hyperglycemia Index, Hypoglycemia Index, and IGC on approximately the same numerical range as measurements of HBGI, LBGI and GRADE, default value is 30.

Details

A tibble object with 1 row for each subject, a column for subject id and a column for the IGC values is returned.

IGC is calculated by taking the sum of the Hyperglycemia Index and the Hypoglycemia index. See hypo_index and hyper_index.

Value

A tibble object with two columns: subject id and corresponding IGC value.

References

Rodbard (2009) Interpretation of continuous glucose monitoring data: glycemic variability and quality of glycemic control, *Diabetes Technology and Therapeutics* **11** .55-67, doi: 10.1089/dia.2008.0132.

Examples

```
data(example_data_1_subject)
igc(example_data_1_subject)
igc(example_data_1_subject, ULTR = 160)

data(example_data_5_subject)
igc(example_data_5_subject)
igc(example_data_5_subject, LLTR = 75, ULTR = 150)
```

---

iglu_shiny *Run IGLU Shiny App*

Description

Run IGLU Shiny App

Usage

```
iglu_shiny()
```

---

in_range_percent *Calculate percentage in targeted value ranges*

Description

The function in_range_percent produces a tibble object with values equal to the percentage of glucose measurements in ranges of target values. The output columns correspond to subject id followed by the target value ranges, and the rows correspond to the subjects. The values will be between 0 (no measurements) and 100 (all measurements).

Usage

```
in_range_percent(data, target_ranges = list(c(80, 200), c(70, 180), c(70, 140)))
```

Arguments

data	DataFrame object with column names "id", "time", and "gl", or numeric vector of glucose values.
target_ranges	List of target value ranges wrapped in an r 'list' structure. Default list of ranges is ((80, 200), (70, 180), (70, 140)).

Details

A tibble object with 1 row for each subject, a column for subject id and column for each range of target values is returned. NA's will be omitted from the glucose values in calculation of percent.

in_range_percent will only work properly if the target_ranges argument is a list of paired values in the format list(c(a1,b1), c(a2,b2), ...). The paired values can be ordered (min, max) or (max, min). See the Examples section for proper usage.

Value

If a data.frame object is passed, then a tibble object with a column for subject id and then a column for each target value is returned. If a vector of glucose values is passed, then a tibble object without the subject id is returned. as.numeric() can be wrapped around the latter to output a numeric vector.

References

Rodbard (2009) Interpretation of continuous glucose monitoring data: glycemic variability and quality of glycemic control, *Diabetes Technology and Therapeutics* **11** .55-67, doi: 10.1089/dia.2008.0132.

Examples

```
data(example_data_1_subject)

in_range_percent(example_data_1_subject)
in_range_percent(example_data_1_subject, target_ranges = list(c(50, 100), c(200,
300), c(80, 140)))

data(example_data_5_subject)

in_range_percent(example_data_5_subject)
in_range_percent(example_data_1_subject, target_ranges = list(c(60, 120), c(140,
250)))
```

---

iqr_glu *Calculate glucose level iqr*

Description

The function iqr_glu outputs the distance between the 25th percentile and the 25th percentile of the glucose values in a tibble object.

Usage

```
iqr_glu(data)
```

Arguments

data	DataFrame object with column names "id", "time", and "gl", or numeric vector of glucose values.

Details

A tibble object with 1 row for each subject, a column for subject id and a column for the IQR values is returned. NA glucose values are omitted from the calculation of the IQR.

Value

If a data.frame object is passed, then a tibble object with two columns: subject id and corresponding IQR value is returned. If a vector of glucose values is passed, then a tibble object with just the IQR value is returned. as.numeric() can be wrapped around the latter to output just a numeric value.

Examples

```
data(example_data_1_subject)
iqr_glu(example_data_1_subject)

data(example_data_5_subject)
iqr_glu(example_data_5_subject)
```

---

j_index *Calculate J-index*

Description

The function j_index produces J-Index values a tibble object.

Usage

```
j_index(data)
```

Arguments

data	DataFrame object with column names "id", "time", and "gl", or numeric vector of glucose values.

Details

A tibble object with 1 row for each subject, a column for subject id and a column for J-Index values is returned. NA glucose values are omitted from the calculation of the J-Index.

J-Index score is calculated by $.001 * [mean(BG) + sd(BG)]^2$ where BG is the list of Blood Glucose Measurements.

Value

If a data.frame object is passed, then a tibble object with two columns: subject id and corresponding J-Index value is returned. If a vector of glucose values is passed, then a tibble object with just the J-Index value is returned. as.numeric() can be wrapped around the latter to output just a numeric value.

References

Wojcicki (1995) "J"-index. A new proposition of the assessment of current glucose control in diabetic patients *Hormone and Metabolic Research* **27** .41-42, doi: 10.1055/s2007979906.

Examples

```
data(example_data_1_subject)
j_index(example_data_1_subject)

data(example_data_5_subject)
j_index(example_data_5_subject)
```

`lbgi` *Calculate Low Blood Glucose Index (LBGI)*

Description

The function lbgi produces LBGI values in a tibble object.

Usage

```
lbgi(data)
```

Arguments

`data`	DataFrame object with column names "id", "time", and "gl", or numeric vector of glucose values.

Details

A tibble object with 1 row for each subject, a column for subject id and a column for LBGI values is returned. NA glucose values are omitted from the calculation of the LBGI.

LBGI is calculated by $1/n * \sum(10 * fbg_i^2)$, where $fbg_i = min(0, 1.509 * (log(BG_i)^{1.084} - 5.381)$, BG_i is the ith Blood Glucose measurement for a subject, and n is the total number of measurements for that subject.

Value

If a data.frame object is passed, then a tibble object with two columns: subject id and corresponding LBGI value is returned. If a vector of glucose values is passed, then a tibble object with just the LBGI value is returned. as.numeric() can be wrapped around the latter to output just a numeric value.

References

Kovatchev et al. (2006) Evaluation of a New Measure of Blood Glucose Variability in, Diabetes *Diabetes care* **29** .2433-2438, doi: 10.2337/dc061085.

Examples

```
data(example_data_1_subject)
lbgi(example_data_1_subject)

data(example_data_5_subject)
lbgi(example_data_5_subject)
```

---

mage *Calculate Mean Amplitude of Glycemic Excursions*

Description

The function mage produces MAGE values in a tibble object.

Usage

```
mage(data, sd_multiplier = 1)
```

Arguments

`data`	DataFrame object with column names "id", "time", and "gl", or numeric vector of glucose values.
`sd_multiplier`	A numeric value that can change the sd value used to determine size of glycemic excursions used in the calculation.

Details

A tibble object with 1 row for each subject, a column for subject id and a column for the MAGE values is returned. NA glucose values are omitted from the calculation of MAGE.

MAGE is calculated by taking the mean of absolute differences (between each value and the mean) that are greater than the standard deviation. A multiplier can be added to the standard deviation by the sd_multiplier argument.

Value

If a data.frame object is passed, then a tibble object with two columns: subject id and corresponding MAGE value is returned. If a vector of glucose values is passed, then a tibble object with just the MAGE value is returned. as.numeric() can be wrapped around the latter to output just a numeric value.

References

Service et al. (1970) Mean amplitude of glycemic excursions, a measure of diabetic instability *Diabetes* **19** .644-655, doi: 10.2337/diab.19.9.644.

Examples

```
data(example_data_1_subject)
mage(example_data_1_subject)
mage(example_data_1_subject, sd_multiplier = 2)

data(example_data_5_subject)
mage(example_data_5_subject, sd_multiplier = .9)
```

`mean_glu` *Calculate mean glucose level*

Description

The function mean_glu is a wrapper for the base function mean(). Output is a tibble object with subject id and mean values.

Usage

```
mean_glu(data)
```

Arguments

`data`	DataFrame object with column names "id", "time", and "gl", or numeric vector of glucose values.

Details

A tibble object with 1 row for each subject, a column for subject id and a column for the mean values is returned. NA glucose values are omitted from the calculation of the mean.

Value

If a data.frame object is passed, then a tibble object with two columns: subject id and corresponding mean value is returned. If a vector of glucose values is passed, then a tibble object with just the mean value is returned. as.numeric() can be wrapped around the latter to output just a numeric value.

Examples

```
data(example_data_1_subject)
mean_glu(example_data_1_subject)

data(example_data_5_subject)
mean_glu(example_data_5_subject)
```

`median_glu` *Calculate median glucose level*

Description

The function median_glu is a wrapper for the base function median(). Output is a tibble object with subject id and median values.

Usage

```
median_glu(data)
```

Arguments

`data`	DataFrame object with column names "id", "time", and "gl", or numeric vector of glucose values.

Details

A tibble object with 1 row for each subject, a column for subject id and a column for the median values is returned. NA glucose values are omitted from the calculation of the median.

Value

If a data.frame object is passed, then a tibble object with two columns: subject id and corresponding median value is returned. If a vector of glucose values is passed, then a tibble object with just the median value is returned. as.numeric() can be wrapped around the latter to output just a numeric value.

Examples

```
data(example_data_1_subject)
median_glu(example_data_1_subject)

data(example_data_5_subject)
median_glu(example_data_5_subject)
```

modd *Calculate mean difference between glucose values obtained at the same time of day (MODD)*

Description

The function modd produces MODD values in a tibble object.

Usage

```
modd(data, lag = 1, tz = "")
```

Arguments

data	DataFrame object with column names "id", "time", and "gl".
lag	Integer indicating which lag (# days) to use. Default is 1.
tz	A character string specifying the time zone to be used. System-specific (see as.POSIXct), but " " is the current time zone, and "GMT" is UTC (Universal Time, Coordinated). Invalid values are most commonly treated as UTC, on some platforms with a warning.

Details

A tibble object with 1 row for each subject, a column for subject id and a column for the MODD values is returned.

Missing values will be linearly interpolated when close enough to non-missing values.

MODD is calculated by taking the mean of absolute differences between measurements at the same time 1 day away, or more if lag parameter is set to an integer > 1.

Value

A tibble object with two columns: subject id and corresponding MODD value.

References

Service, Nelson (1980) Characteristics of glycemic stability. *Diabetes care* **3** .58-62, doi: 10.2337/diacare.3.1.58.

Examples

```
data(example_data_1_subject)
modd(example_data_1_subject)
modd(example_data_1_subject, lag = 2)

data(example_data_5_subject)
modd(example_data_5_subject, lag = 2)
```

`m_value` *Calculate the M-value*

Description

Calculates the M-value of Schlichtkrull et al. (1965) for each subject in the data, where the M-value is the mean of the logarithmic transformation of the deviation from a reference value. Produces a tibble object with subject id and M-values.

Usage

```
m_value(data, r = 90)
```

Arguments

`data`	DataFrame object with column names "id", "time", and "gl", or numeric vector of glucose values.
`r`	A reference value corresponding to basal glycemia in normal subjects; default is 90 mg/dL.

Details

A tibble object with 1 row for each subject, a column for subject id and a column for the M-values is returned. NA glucose values are omitted from the calculation of the M-value.

M-value is computed by averaging the transformed glucose values, where each transformed value is equal to $|1000 * log_10(glucose/100)|^3$

Value

If a data.frame object is passed, then a tibble object with two columns: subject id and corresponding M-value is returned. If a vector of glucose values is passed, then a tibble object with just the M-value is returned. as.numeric() can be wrapped around the latter to output just a numeric value.

References

Schlichtkrull J, Munck O, Jersild M. (1965) The M-value, an index of blood-sugar control in diabetics. *Acta Medica Scandinavica* **177** .95-102. doi: 10.1111/j.09546820.1965.tb01810.x.

Examples

```
data(example_data_5_subject)

m_value(example_data_5_subject)
m_value(example_data_5_subject, r = 100)
```

plot_glu *Plot time series and lasagna plots of glucose measurements*

Description

The function plot_glu supports several plotting methods for both single and multiple subject data.

Usage

```
plot_glu(
  data,
  plottype = c("tsplot", "lasagna"),
  datatype = c("all", "average", "single"),
  lasagnatype = c("unsorted", "timesorted"),
  LLTR = 80,
  ULTR = 140,
  subjects = NULL,
  tz = ""
)
```

Arguments

data	DataFrame with column names ("id", "time", and "gl").
plottype	String corresponding to the desired plot type. Options are 'tsplot' for a time series plot and 'lasagna' for a lasagna plot. See the 'lasagnatype' parameter for further options corresponding to the 'lasagna' 'plottype'. Default is 'tsplot'.
datatype	String corresponding to data aggregation used for plotting, currently supported options are 'all' which plots all glucose measurements within the first maxd days for each subject, and 'average' which plots average 24 hour glucose values across days for each subject
lasagnatype	String corresponding to plot type when usingdatatype = "average", currently supported options are 'unsorted' for an unsorted lasagna plot, 'timesorted' for a lasagna plot with glucose values sorted within each time point across subjects, and ''subjectsorted'' for a lasagna plot with glucose values sorted within each subject across time points.
LLTR	Lower Limit of Target Range, default value is 80 mg/dL.
ULTR	Upper Limit of Target Range, default value is 140 mg/dL.
subjects	String or list of strings corresponding to subject names in 'id' column of data. Default is all subjects.
tz	A character string specifying the time zone to be used. System-specific (see as.POSIXct), but " " is the current time zone, and "GMT" is UTC (Universal Time, Coordinated). Invalid values are most commonly treated as UTC, on some platforms with a warning.

Details

For the default option 'tsplot', a time series graph for each subject is produced with hypo- and hyperglycemia cutoffs shown as horizontal red lines. The time series plots for all subjects chosen (all by default) are displayed on a grid.

The 'lasagna' plot type works best when the datatype argument is set to average.

Value

Any output from the plot object

Examples

```
data(example_data_1_subject)
plot_glu(example_data_1_subject)

data(example_data_5_subject)
plot_glu(example_data_5_subject, subjects = 'Subject 2')
plot_glu(example_data_5_subject, plottype = 'tsplot', tz = 'EST', LLTR = 70, ULTR= 150)
plot_glu(example_data_5_subject, plottype = 'lasagna', lasagnatype = 'timesorted')
```

---

plot_lasagna *Lasagna plot of glucose values for multiple subjects*

Description

Lasagna plot of glucose values for multiple subjects

Usage

```
plot_lasagna(
  data,
  datatype = c("all", "average"),
  lasagnatype = c("unsorted", "timesorted", "subjectsorted"),
  maxd = 14,
  limits = c(50, 500),
  midpoint = 105,
  LLTR = 80,
  ULTR = 140,
  dt0 = NULL,
  inter_gap = 60,
  tz = ""
)
```

Arguments

data	DataFrame object with column names "id", "time", and "gl".
datatype	String corresponding to data aggregation used for plotting, currently supported options are 'all' which plots all glucose measurements within the first maxd days for each subject, and 'average' which plots average 24 hour glucose values across days for each subject
lasagnatype	String corresponding to plot type when usingdatatype = "average", currently supported options are 'unsorted' for an unsorted lasagna plot, 'timesorted' for a lasagna plot with glucose values sorted within each time point across subjects, and ''subjectsorted'' for a lasagna plot with glucose values sorted within each subject across time points.
maxd	For datatype "all", maximal number of days to be plotted from the study. The default value is 14 days (2 weeks).
limits	The minimal and maximal glucose values for coloring grid which is gradient from blue (minimal) to red (maximal), see scale_fill_gradient2)
midpoint	The glucose value serving as midpoint (white) of the diverging gradient scale (see scale_fill_gradient2). The default value is 125 mg/dL. The values above are colored in red, and below in blue.
LLTR	Lower Limit of Target Range, default value is 80 mg/dL.
ULTR	Upper Limit of Target Range, default value is 140 mg/dL.
dt0	The time frequency for interpolated aligned grid in minutes, the default will match the CGM meter's frequency (e.g. 5 min for Dexcom).
inter_gap	The maximum allowable gap (in minutes) for interpolation of NA glucose values. The values will not be interpolated between the glucose measurements that are more than inter_gap minutes apart. The default value is 60 min.
tz	A character string specifying the time zone to be used. System-specific (see as.POSIXct), but " " is the current time zone, and "GMT" is UTC (Universal Time, Coordinated). Invalid values are most commonly treated as UTC, on some platforms with a warning.

Value

A ggplot object corresponding to lasagna plot

References

Swihart et al. (2010) Lasagna Plots: A Saucy Alternative to Spaghetti Plots, *Epidemiology* **21**(5), 621-625, doi: 10.1097/EDE.0b013e3181e5b06a

Examples

```
plot_lasagna(example_data_5_subject, datatype = "average", lasagnatype = 'timesorted', tz = "EST")
plot_lasagna(example_data_5_subject, lasagnatype = "subjectsorted", LLTR = 100, tz = "EST")
```

`plot_lasagna_1subject` *Lasagna plot of glucose values for 1 subject aligned across times of day*

Description

Lasagna plot of glucose values for 1 subject aligned across times of day

Usage

```
plot_lasagna_1subject(
  data,
  lasagnatype = c("unsorted", "timesorted", "daysorted"),
  limits = c(50, 500),
  midpoint = 105,
  LLTR = 80,
  ULTR = 140,
  dt0 = NULL,
  inter_gap = 60,
  tz = ""
)
```

Arguments

`data`	DataFrame object with column names "id", "time", and "gl".
`lasagnatype`	String corresponding to plot type, currently supported options are 'unsorted' for an unsorted single-subject lasagna plot, 'timesorted' for a lasagna plot with glucose values sorted within each time point across days, and 'daysorted' for a lasagna plot with glucose values sorted within each day across time points.
`limits`	The minimal and maximal glucose values for coloring grid which is gradient from blue (minimal) to red (maximal), see `scale_fill_gradient2`)
`midpoint`	The glucose value serving as midpoint (white) of the diverging gradient scale (see `scale_fill_gradient2`). The default value is 125 mg/dL. The values above are colored in red, and below in blue.
`LLTR`	Lower Limit of Target Range, default value is 80 mg/dL.
`ULTR`	Upper Limit of Target Range, default value is 140 mg/dL.
`dt0`	The time frequency for interpolated aligned grid in minutes, the default will match the CGM meter's frequency (e.g. 5 min for Dexcom).
`inter_gap`	The maximum allowable gap (in minutes) for interpolation of NA glucose values. The values will not be interpolated between the glucose measurements that are more than inter_gap minutes apart. The default value is 60 min.
`tz`	A character string specifying the time zone to be used. System-specific (see `as.POSIXct`), but " " is the current time zone, and "GMT" is UTC (Universal Time, Coordinated). Invalid values are most commonly treated as UTC, on some platforms with a warning.

Value

A ggplot object corresponding to lasagna plot

References

Swihart et al. (2010) Lasagna Plots: A Saucy Alternative to Spaghetti Plots, *Epidemiology* **21**(5), 621-625, doi: 10.1097/EDE.0b013e3181e5b06a

Examples

```
plot_lasagna_1subject(example_data_1_subject)
plot_lasagna_1subject(example_data_1_subject, lasagnatype = 'timesorted')
plot_lasagna_1subject(example_data_1_subject, lasagnatype = 'daysorted')
```

---

quantile_glu *Calculate glucose level quantiles*

Description

The function quantile_glu is a wrapper for the base function quantile(). Output is a tibble object with columns for subject id and each of the quantiles.

Usage

```
quantile_glu(data, quantiles = c(0, 25, 50, 75, 100))
```

Arguments

`data`	DataFrame object with column names "id", "time", and "gl", or numeric vector of glucose values.
`quantiles`	List of quantile values between 0 and 100.

Details

A tibble object with 1 row for each subject, a column for subject id and a column for each quantile is returned. NA glucose values are omitted from the calculation of the quantiles.

The values are scaled from 0-1 to 0-100 to be consistent in output with above_percent, below_percent, and in_range_percent.

The command quantile_glu(...) / 100 will scale each element down from 0-100 to 0-1.

Value

If a data.frame object is passed, then a tibble object with a column for subject id and then a column for each quantile value is returned. If a vector of glucose values is passed, then a tibble object without the subject id is returned. as.numeric() can be wrapped around the latter to output a numeric vector.

Examples

```
data(example_data_1_subject)

quantile_glu(example_data_1_subject)
quantile_glu(example_data_1_subject, quantiles = c(0, 33, 66, 100))

data(example_data_5_subject)

quantile_glu(example_data_5_subject)
quantile_glu(example_data_5_subject, quantiles = c(0, 10, 90, 100))
```

`range_glu` *Calculate glucose level range*

Description

The function range_glu outputs the distance between minimum and maximum glucose values per subject in a tibble object.

Usage

```
range_glu(data)
```

Arguments

`data`	DataFrame object with column names "id", "time", and "gl", or numeric vector of glucose values.

Details

A tibble object with 1 row for each subject, a column for subject id and a column for the range values is returned. NA glucose values are omitted from the calculation of the range.

Value

If a data.frame object is passed, then a tibble object with two columns: subject id and corresponding range value is returned. If a vector of glucose values is passed, then a tibble object with just the range value is returned. as.numeric() can be wrapped around the latter to output just a numeric value.

Examples

```
data(example_data_1_subject)
range_glu(example_data_1_subject)

data(example_data_5_subject)
range_glu(example_data_5_subject)
```

sd_glu *Calculate sd glucose level*

Description

The function sd_glu is a wrapper for the base function sd(). Output is a tibble object with subject id and sd values.

Usage

```
sd_glu(data)
```

Arguments

data	DataFrame object with column names "id", "time", and "gl", or numeric vector of glucose values.

Details

A tibble object with 1 row for each subject, a column for subject id and a column for the sd values is returned. NA glucose values are omitted from the calculation of the sd.

Value

If a data.frame object is passed, then a tibble object with two columns: subject id and corresponding sd value is returned. If a vector of glucose values is passed, then a tibble object with just the sd value is returned. as.numeric() can be wrapped around the latter to output just a numeric value.

Examples

```
data(example_data_1_subject)
sd_glu(example_data_1_subject)

data(example_data_5_subject)
sd_glu(example_data_5_subject)
```

sd_measures *Calculate SD subtypes*

Description

The function sd_measures produces SD subtype values in a tibble object with a row for each subject and columns corresponding to id followed by each SD subtype.

Usage

```
sd_measures(data,dt0 = NULL, inter_gap = 45, tz = "")
```

Arguments

data	DataFrame object with column names "id", "time", and "gl".
dt0	The time frequency for interpolation in minutes, the default will match the CGM meter's frequency (e.g. 5 min for Dexcom).
inter_gap	The maximum allowable gap (in minutes) for interpolation. The values will not be interpolated between the glucose measurements that are more than inter_gap minutes apart. The default value is 45 min.
tz	A character string specifying the time zone to be used. System-specific (see as.POSIXct), but " " is the current time zone, and "GMT" is UTC (Universal Time, Coordinated). Invalid values are most commonly treated as UTC, on some platforms with a warning.

Details

A tibble object with 1 row for each subject, a column for subject id and a column for each SD subtype values is returned.

Missing values will be linearly interpolated when close enough to non-missing values.

1. SdW- vertical within days:

   Calculated by first taking the standard deviation of each day's glucose measurements, then taking the mean of all the standard deviations. That is, for d days we compute SD_1 ... SD_d daily standard deviations and calculate $1/d * \sum[(SD_i)]$

2. SdHHMM - between time points:

   Calculated by taking the mean glucose values at each time point in the grid across days, and taking the standard deviation of those mans. That is, for t time points we compute X_t means for each time point and then compute SD([X_1, X_2, ... X_t]).

3. SdWSH - within series:

   Calculated by taking the hour-long intervals starting at every point in the interpolated grid, computing the standard deviation of the points in each hour-long interval, and then finding the mean of those standard deviations. That is, for n time points compute SD_1 ... SD_n, where SD_i is the standard deviation of the set [SD_i, SD_i+2, ... SD_k-1] where SD_k is the first measurement more than an hour later than SD_1. Then, take $1/n * \sum[(SD_i)]$.

4. SdDM - horizontal sd:

   Calculated by taking the daily mean glucose values, and then taking the standard deviation of those daily means. That is, for d days we take X_1 ... X_d daily means, and then compute SD([X_1, X_2, ... X_d]).

5. SdB - between days, within timepoints:

   Calculated by taking the standard deviation of the glucose values across days for each time point, and then taking the mean of those standard deviations. That is, for t time points take SD_1 ... SD_t standard deviations, and then compute $1/t * \sum[(SD_i)]$

6. SdBDM - between days, within timepoints, corrected for changes in daily means: Calculated by subtracting the daily mean from each glucose value, then taking the standard deviation of the corrected glucose values across days for each time point, and then taking the mean of those standard deviations. That is, for t time points take SD_1 ... SD_t standard deviations, and then compute $1/t * \sum[(SD_i)]$. where SD_i is the standard deviation of d daily values at the 1st time point, where each value is the dth measurement for the ith time point subtracted by the mean of all glucose values for day d.

Value

A tibble object with a column for id and a column for each of the six SD subtypes.

References

Rodbard (2009) New and Improved Methods to Characterize Glycemic Variability Using Continuous Glucose Monitoring *Diabetes Technology and Therapeutics* **11** .551-565, doi: 10.1089/dia.2009.0015.

Examples

```
data(example_data_1_subject)
sd_measures(example_data_1_subject)
```

---

summary_glu *Calculate summary glucose level*

Description

The function summary_glu is a wrapper for the base function summary(). Output is a tibble object with subject id and the summary value: Minimum, 1st Quantile, Median, Mean, 3rd Quantile and Max.

Usage

```
summary_glu(data)
```

Arguments

`data`	DataFrame object with column names "id", "time", and "gl", or numeric vector of glucose values.

Details

A tibble object with 1 row for each subject, a column for subject id and a column for each of summary values is returned. NA glucose values are omitted from the calculation of the summary values.

Value

If a data.frame object is passed, then a tibble object with a column for subject id and then a column for each summary value is returned. If a vector of glucose values is passed, then a tibble object without the subject id is returned. as.numeric() can be wrapped around the latter to output a numeric vector with values in order of Min, 1st Quantile, Median, Mean, 3rd Quantile and Max.

Examples

```
data(example_data_1_subject)
summary_glu(example_data_1_subject)

data(example_data_5_subject)
summary_glu(example_data_5_subject)
```

Index

∗ **datasets**
    example_data_1_subject, 8
    example_data_5_subject, 9

above_percent, 2
adrr, 3
as.POSIXct, *6*, *25*, *27*, *29, 30*, *34*

below_percent, 4

CGMS2DayByDay, 5
conga, 6
cv_glu, 7

example_data_1_subject, 8
example_data_5_subject, *8*, 9

grade, 9
grade_eugly, 10
grade_hyper, 11
grade_hypo, 12

hbgi, 13
hyper_index, 14, *17*
hypo_index, 15, *17*

igc, 17
iglu_shiny, 18
in_range_percent, 18
iqr_glu, 19

j_index, 20

lbgi, 21

m_value, 26
mage, 22
mean_glu, 23
median_glu, 24
modd, 25

plot_glu, 27
plot_lasagna, 28
plot_lasagna_1subject, 30

quantile_glu, 31

range_glu, 32

scale_fill_gradient2, *29, 30*
sd_glu, 33
sd_measures, 33
summary_glu, 35