Package 'biodosetools'

October 10, 2025

Title 'shiny' Application for Biological Dosimetry

Version 3.6.2

Description A tool to perform all different statistical tests and calculations needed by Biological dosimetry Laboratories. Detailed documentation is available in <https://biodosetools-team.github.io/documentation/>.

License GPL-3

URL https://biodosetools-team.github.io/biodosetools/, https://github.com/biodosetools-team/biodosetools/

BugReports https://github.com/biodosetools-team/biodosetools/issues/

Depends R (>= 3.5.0), shiny, golem

Imports bsplus, dplyr (>= 1.0.0), ggplot2, magrittr, maxLik, mixtools, msm, pdftools, rhandsontable, rlang (>= 0.4.0), readr, shinydashboard, shinyWidgets (>= 0.5.0), tidyr (>= 1.0.0), config, cli, openxlsx, MASS, gridExtra, rmarkdown

Encoding UTF-8

LazyData true

RoxygenNote 7.3.1

Suggests testthat (>= 3.0.0), covr, knitr, kableExtra, markdown, pander, tinytex, xtable

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Alfredo Hernández [aut] (ORCID: <https://orcid.org/0000-0002-2660-4545>), Anna Frances-Abellan [aut, cre] (ORCID: <https://orcid.org/0000-0003-4738-1712>), David Endesfelder [aut], Pere Puig [aut] (ORCID: <https://orcid.org/0000-0002-6607-9642>)

Maintainer Anna Frances-Abellan <anna.frances@uab.cat>

Repository CRAN

Date/Publication 2025-10-10 20:00:08 UTC

Contents

`AIC_from_data` *Calculate AIC (Akaike's 'An Information Criterion')*

Description

Calculate AIC (Akaike's 'An Information Criterion')

Usage

```
AIC_from_data(
  general_fit_coeffs,
  data,
  dose_var = "dose",
  yield_var = "yield",
  fit_link = "identity"
)
```

Arguments

`general_fit_coeffs`	Generalised fit coefficients matrix.
`data`	Data (dose, yield) to calculate AIC from.
`dose_var`	Name of the dose variable (enquoted).
`yield_var`	Name of the yield variable (enquoted).
`fit_link`	A specification for the model link function.

Value

Numeric value of AIC.

bar_plots *Bar plots dosimetry*

Description

Bar plots dosimetry

Usage

```
bar_plots(dat, curve, place)
```

Arguments

dat	data frame of data values.
curve	manual or auto.
place	UI or save.

Value

plot

Examples

```
dat <- data.frame(
  "Lab" = c("A1", "A2"),
  "Module" = c("dicentrics", "dicentrics"),
  "Type" = c("manual", "manual"),
  "radiation quality" = c("Cs-137", "Co-60"),
  "calibration" = c("air kerma", "air kerma"),
  "irradiation" = c("air", "air"),
  "temperature" = c(20, 37),
  "dose rate" = c(0.446, 0.27),
  "curve origin" = c("own", "own"),
  "C" = c(0.001189589, 0.0005),
  "alpha" = c(0.01903783, 0.0142),
  "beta" = c(0.0968831, 0.0759),
  "C std.error" = c(0.0001040828, 0.0005),
  "alpha std.error" = c(0.004119324, 0.0044),
  "beta std.error" = c(0.003505209, 0.0027),
  "max curve dose" = c(6.00, 5.05),
  stringsAsFactors = FALSE,
  check.names = FALSE
)


bar_plots(
  dat = dat,
  curve = "manual",
  place = "UI"
)
```

calc.zValue.new *Calculate ZScore*

Description

Calculate ZScore

Usage

```
calc.zValue.new(X, type, alg, c)
```

Arguments

X	vector of estimated doses.
type	dose or freq.
alg	algorithm. Choose between algA, algB or QHampel
c	value for reference dose.

Value

Numeric value of zscore.

Examples

```
calc.zValue.new(X = c(3.65, 2.5, 4.85),
                type = "dose",
                alg = "algA" ,
                c = c(3.5, 3, 4.25))
```

calculate_aberr *Aberration calculation functions*

Description

Aberration calculation functions

Usage

```
calculate_aberr_power(data, aberr_prefix = "C", power = 1)

calculate_aberr_mean(X, N)

calculate_aberr_var(X, X2, N)

calculate_aberr_disp_index(mean, var)
```

```
calculate_aberr_u_value(X, N, mean, var, assessment_u = 1)

init_aberr_table(
  data,
  type = c("count", "case"),
  aberr_module = c("dicentrics", "translocations", "micronuclei")
)
```

Arguments

data	Count or case data.
aberr_prefix	Prefix of the aberrations in the data.
power	Power of aberration.
X	Sum of detected aberrations.
N	Number of cells analysed.
X2	Quadratic sum of detected aberrations.
mean	Mean.
var	Variance.
assessment_u	Expected u-value of the assessment. For a Poisson distribution this should be unity.
type	Type of input data. Either "count" and "case".
aberr_module	Aberration module.

---

calculate_aberr_IRR *Calculate IRR (this is the same as the Odds-Ratio calculation in IAEA2011)*

Description

Calculate IRR (this is the same as the Odds-Ratio calculation in IAEA2011)

Usage

```
calculate_aberr_IRR(case_data, fit_coeffs, badge_dose)
```

Arguments

case_data	Case data in data frame form.
fit_coeffs	Fitting coefficients matrix.
badge_dose	Suspected (e.g. badge) dose

Value

character variable representing IRR in the for (P(X=k|D=0 Gy):P(X=k|D=badge_dose)).

`calculate_aberr_table` *Calculate aberrations table*

Description

Calculate aberrations table

Usage

```
calculate_aberr_table(
  data,
  type = c("count", "case"),
  aberr_module = c("dicentrics", "translocations", "micronuclei"),
  assessment_u = 1
)
```

Arguments

`data`	Count or case data.
`type`	Type of input data. Either "count" and "case".
`aberr_module`	Aberration module, required for `type = "case"`.
`assessment_u`	Expected u-value of the assessment. For a Poisson distribution this should be unity.

Value

Data frame containing cell count (N), aberrations (X), and other coefficients (dispersion index, u-value, ...), as well as raw count or case `data`.

Examples

```
data <- data.frame(
    ID = c("example1", "example2"),
    C0 = c(302, 160),
    C1 = c(28, 55),
    C2 = c(22, 19),
    C3 = c(8, 17),
    C4 = c(1, 9),
    C5 = c(0, 4)
)


calculate_aberr_table(data,
                      type = "case",
                      aberr_module = "dicentrics",
                      assessment_u = 1)
```

`calculate_characteristic_limits` *Characteristic limits function*

Description

Characteristic limits function

Usage

```
calculate_characteristic_limits(
  mu0 = NULL,
  n1 = NULL,
  y0 = NULL,
  n0 = NULL,
  alpha = 0.05,
  beta = 0.1,
  ymax = 100,
  type = "const"
)
```

Arguments

`mu0`	Background rate
`n1`	Number of cells that will be analysed.
`y0`	Background number of dicentrics.
`n0`	Background number of cells analysed.
`alpha`	Type I error rate, 0.05 by default.
`beta`	Type II error rate, 0.1 by default.
`ymax`	Max dicentrics to evaluate.
`type`	Type.

Value

List of characteristic limits (`decision_threshold`, `detection_limit`).

Examples

```
control_data <- c(aberr = 4,
                  cells = 1000)

calculate_characteristic_limits(
      y0 = control_data["aberr"],
      n0 = control_data["cells"],
      n1 = 20,
      alpha = 0.05,
```

```
      beta = 0.1,
      ymax = 100,
      type = "var"
)
```

`calculate_genome_factor` *Calculate genomic conversion factor*

Description

Method based on the paper by Lucas, J. N. et al. (1992). Rapid Translocation Frequency Analysis in Humans Decades after Exposure to Ionizing Radiation. International Journal of Radiation Biology, 62(1), 53-63. <doi:10.1080/09553009214551821>.

Usage

```
calculate_genome_factor(dna_table, chromosomes, colors, sex)
```

Arguments

`dna_table`	DNA content fractions table. Can be `dna_content_fractions_morton` or `dna_content_table_ihgsc`.
`chromosomes`	Vector of stained chromosomes.
`colors`	Vector of colors of the stains.
`sex`	Sex of the individual.

Value

Numeric value of genomic conversion factor.

Examples

```
#dna_table can be: dna_content_fractions_morton OR dna_content_fractions_ihgsc

calculate_genome_factor(
  dna_table = dna_content_fractions_morton,
  chromosome = c(1, 2, 3, 4, 5, 6),
  color = c("Red", "Red", "Green", "Red", "Green", "Green"),
  sex = "male"
)
```

`calculate_model_stats` *Calculate model statistics*

Description

Calculate model statistics

Usage

```
calculate_model_stats(
  model_data,
  fit_coeffs_vec,
  glm_results = NULL,
  fit_algorithm = NULL,
  response = "yield",
  link = c("identity", "log"),
  type = c("theory", "raw"),
  Y = NULL,
  mu = NULL,
  n = NULL,
  npar = NULL,
  genome_factor = NULL,
  calc_type = c("fitting", "estimation"),
  model_family
)
```

Arguments

`model_data`	Data of the model.
`fit_coeffs_vec`	Vector of fitting coefficients.
`glm_results`	Results of glm.
`fit_algorithm`	String of the algorithm used.
`response`	Type of response.
`link`	Fit link.
`type`	Theoretical or raw glm model statistics.
`Y`	Y response (required in constraint-maxlik-optimization).
`mu`	mu response required in constraint-maxlik-optimization).
`n`	number of parameters (required in constraint-maxlik-optimization).
`npar`	number of parameters (required in constraint-maxlik-optimization).
`genome_factor`	Genomic conversion factor used in translocations.
`calc_type`	Calculation type, either "fitting" or "estimation".
`model_family`	Model family.

Value

Data frame of model statistics.

`calculate_table` *Calculate the characteristic limits table*

Description

Calculate the characteristic limits table

Usage

```
calculate_table(input, project_yield, fit_coeffs)
```

Arguments

`input`	all input UI parameters
`project_yield`	function for calculating yield, in utils_estimation.R
`fit_coeffs`	curve coefficients

Value

list with all the table columns.

`calculate_trans_rate_manual`
Calculate manual translocation rate

Description

Calculate manual translocation rate

Usage

```
calculate_trans_rate_manual(cells, genome_factor, expected_aberr_value)
```

Arguments

`cells`	Number of cells `N`.
`genome_factor`	Genomic conversion factor.
`expected_aberr_value`	Expected aberrations.

Value

Numeric value of translocation rate.

Examples

```
calculate_trans_rate_manual(cells = 1000,
                            genome_factor = 0.58,
                            expected_aberr_value = 0.3)
```

`calculate_trans_rate_sigurdson`

Calculate Sigurdson's translocation rate

Description

Method based on the paper by Sigurdson, A. J. et al. (2008). International study of factors affecting human chromosome translocations. Mutation Research/Genetic Toxicology and Environmental Mutagenesis, 652(2), 112-121. <doi:10.1016/j.mrgentox.2008.01.005>.

Usage

```
calculate_trans_rate_sigurdson(
  cells,
  genome_factor,
  age_value,
  sex_bool = FALSE,
  sex_value = "none",
  smoker_bool = FALSE,
  ethnicity_value = "none",
  region_value = "none"
)
```

Arguments

`cells`	Number of cells `N`.
`genome_factor`	Genomic conversion factor.
`age_value`	Age of the individual.
`sex_bool`	If `TRUE`, `sex_value` will be used.
`sex_value`	Sex of the individual, either "male" of "female".
`smoker_bool`	Whether the individual smokes or not.
`ethnicity_value`	Ethnicity of the individual.
`region_value`	Region of the individual.

Value

Numeric value of translocation rate.

Examples

```
calculate_trans_rate_sigurdson(
  cells = 1000,
  genome_factor = 0.58,
  age_value = 30,
  sex_bool = FALSE,
  sex_value = "none",
  smoker_bool = FALSE,
  ethnicity_value = "none",
  region_value = "none"
)
```

---

calculate_yield *Calculate yield from dose*

Description

Calculate yield from dose

Usage

```
calculate_yield(
  dose,
  type = c("estimate", "lower", "upper"),
  general_fit_coeffs,
  general_fit_var_cov_mat = NULL,
  protracted_g_value = 1,
  conf_int = 0.95
)
```

Arguments

dose	Numeric value of dose.
type	Type of yield calculation. Can be "estimate", "lower", or "upper".
general_fit_coeffs	Generalised fit coefficients matrix.
general_fit_var_cov_mat	Generalised variance-covariance matrix.
protracted_g_value	Protracted $G(x)$ value.
conf_int	Curve confidence interval, 95% by default.

Value

Numeric value of yield.

calculate_yield_infimum *Calculate theoretical yield infimum*

Description

Calculate theoretical yield infimum

Usage

```
calculate_yield_infimum(
  type = c("estimate", "lower", "upper"),
  general_fit_coeffs,
  general_fit_var_cov_mat = NULL,
  conf_int = 0.95
)
```

Arguments

type	Type of yield calculation. Can be "estimate", "lower", or "upper".
general_fit_coeffs	Generalised fit coefficients matrix.
general_fit_var_cov_mat	Generalised variance-covariance matrix.
conf_int	Curve confidence interval, 95% by default.

Value

Numeric value of yield infimum.

correct_boundary *Correct boundary of irradiated fractions to be bounded by 0 and 1*

Description

Correct boundary of irradiated fractions to be bounded by 0 and 1

Usage

```
correct_boundary(x)
```

Arguments

x	Numeric value.

Value

Numeric value in [0, 1] range.

`correct_conf_int` *Correct yield confidence interval*

Description

Correct yield confidence interval if simple method is required.

Usage

```
correct_conf_int(
  conf_int,
  general_fit_var_cov_mat,
  protracted_g_value = 1,
  type,
  dose = seq(0, 10, 0.2)
)
```

Arguments

`conf_int`	Confidence interval.
`general_fit_var_cov_mat`	Generalised variance-covariance matrix.
`protracted_g_value`	Protracted $G(x)$ value.
`type`	Type of yield calculation. Can be "estimate", "lower", or "upper".
`dose`	Numeric value of dose.

Value

Numeric value of corrected confidence interval.

`correct_negative_vals` *Correct negative values*

Description

Correct negative values

Usage

```
correct_negative_vals(x)
```

Arguments

x	Numeric value.

Value

Numeric value corrected to zero if negative.

---

correct_yield *Correct yields if they are below the curve*

Description

Correct yields if they are below the curve

Usage

```
correct_yield(
  yield,
  type = "estimate",
  general_fit_coeffs,
  general_fit_var_cov_mat,
  conf_int
)
```

Arguments

yield	Numeric value of yield.
type	Type of yield calculation. Can be "estimate", "lower", or "upper".
general_fit_coeffs	Generalised fit coefficients matrix.
general_fit_var_cov_mat	Generalised variance-covariance matrix.
conf_int	Curve confidence interval.

Value

Numeric value of corrected yield.

curves_plot *Plot curves*

Description

Plot curves

Usage

```
curves_plot(dat, curve, curve_type = "lin_quad", place)
```

Arguments

dat	data frame of data values.
curve	manual or auto.
curve_type	lin or lin_quad.
place	UI or save.

Value

ggplot2 object.

Examples

```
dat <- data.frame(
  "Lab" = c("A1", "A2"),
  "Module" = c("dicentrics", "dicentrics"),
  "Type" = c("manual", "manual"),
  "radiation quality" = c("Cs-137", "Co-60"),
  "calibration" = c("air kerma", "air kerma"),
  "irradiation" = c("air", "air"),
  "temperature" = c(20, 37),
  "dose rate" = c(0.446, 0.27),
  "curve origin" = c("own", "own"),
  "C" = c(0.001189589, 0.0005),
  "alpha" = c(0.01903783, 0.0142),
  "beta" = c(0.0968831, 0.0759),
  "C std.error" = c(0.0001040828, 0.0005),
  "alpha std.error" = c(0.004119324, 0.0044),
  "beta std.error" = c(0.003505209, 0.0027),
  "max curve dose" = c(6.00, 5.05),
  stringsAsFactors = FALSE,
  check.names = FALSE
)


curves_plot(
  dat = dat,
```

```
  curve = "manual",
  curve_type = "lin_quad",
  place = "UI"
)
```

`DI_plot` *Dispersion index*

Description

Dispersion index

Usage

```
DI_plot(dat, place)
```

Arguments

`dat`	data frame of data values.
`place`	UI or save.

Value

plot

Examples

```
dat <- data.frame(
  Lab = c("A1", "A2", "A1", "A2", "A1", "A2"),
 Module = c("dicentrics", "dicentrics", "dicentrics", "dicentrics", "dicentrics", "dicentrics"),
  Type = c("manual", "manual", "manual", "manual", "manual", "manual"),
  Sample = c(1, 1, 2, 2, 3, 3),
  N = c(200, 200, 200, 200, 200, 200),
  X = c(140, 111, 204, 147, 368, 253),
  estimate = c(2.589230, 2.610970, 3.146055, 3.018682, 4.259400, 3.989222),
  lower = c(2.083403, 2.231150, 2.614931, 2.622748, 3.882349, 3.552963),
  upper = c(3.208857, 3.045616, 3.785879, 3.471269, 4.688910, 4.487336),
  y = c(0.700, 0.555, 1.020, 0.735, 1.840, 1.265),
  y.err = c(0.0610, 0.0495, 0.0733, 0.0556, 0.0923, 0.0785),
  DI = c(1.0625, 0.8818, 1.0539, 0.8406, 0.9255, 0.9731),
  u = c(0.6252, -1.1840, 0.5389, -1.5951, -0.7442, -0.2692),
  stringsAsFactors = FALSE
)

DI_plot(
  dat = dat,
  place = "UI"
)
```

dna_content_fractions_ihgsc *DNA Content Fractions of Human Chromosomes (IHGSC)*

Description

Normalised DNA Content of Human Chromosomes from the International Human Genome Sequencing Consortium.

Usage

```
dna_content_fractions_ihgsc
```

Format

A data frame with 24 rows and 3 variables:

chromosome Chromosome.

fraction_male Normalised content of megabases on male human DNA.

fraction_female Normalised content of megabases on female human DNA.

Details

Last accessed in July 2020.

Source

https://www.ncbi.nlm.nih.gov/grc/human/data

dna_content_fractions_morton *DNA Content Fractions of Human Chromosomes (Morton 1991)*

Description

Normalised DNA Content of Human Chromosomes from Morton, N. E. (1991). Parameters of the human genome. Proceedings of the National Academy of Sciences, 88(17), 7474-7476.

Usage

```
dna_content_fractions_morton
```

Format

A data frame with 24 rows and 3 variables:

chromosome Chromosome.

fraction_male Normalised content of megabases on male human DNA.

fraction_female Normalised content of megabases on female human DNA.

Source

doi:10.1073/pnas.88.17.7474

`dose_boxplot` *Boxplot dose estimates*

Description

Boxplot dose estimates

Usage

```
dose_boxplot(dat, place)
```

Arguments

`dat`	data frame of data values.
`place`	UI or save.

Value

plot

Examples

```
dat <- data.frame(
  Lab = c("A1", "A2", "A1", "A2", "A1", "A2"),
 Module = c("dicentrics", "dicentrics", "dicentrics", "dicentrics", "dicentrics", "dicentrics"),
  Type = c("manual", "manual", "manual", "manual", "manual", "manual"),
  Sample = c(1, 1, 2, 2, 3, 3),
  N = c(200, 200, 200, 200, 200, 200),
  X = c(140, 111, 204, 147, 368, 253),
  estimate = c(2.589230, 2.610970, 3.146055, 3.018682, 4.259400, 3.989222),
  lower = c(2.083403, 2.231150, 2.614931, 2.622748, 3.882349, 3.552963),
  upper = c(3.208857, 3.045616, 3.785879, 3.471269, 4.688910, 4.487336),
  y = c(0.700, 0.555, 1.020, 0.735, 1.840, 1.265),
  y.err = c(0.0610, 0.0495, 0.0733, 0.0556, 0.0923, 0.0785),
  DI = c(1.0625, 0.8818, 1.0539, 0.8406, 0.9255, 0.9731),
  u = c(0.6252, -1.1840, 0.5389, -1.5951, -0.7442, -0.2692),
  stringsAsFactors = FALSE
```

```
)

dose_boxplot(
  dat = dat,
  place = "UI"
)
```

`dose_estimation_mx` *Prepare data to dose estimate using mixed fields manual*

Description

Prepare data to dose estimate using mixed fields manual

Usage

```
dose_estimation_mx(input, reactive_data, data_type)
```

Arguments

`input`	what you upload in the UI
`reactive_data`	access to the reactive data
`data_type`	gamma or neutron

Value

data for dose estimation using manual input.

`estimate_hetero_mixed_poisson` *Heterogeneous dose estimation (Mixed Poisson model)*

Description

Method based on the paper by Pujol, M. et al. (2016). A New Model for Biological Dose Assessment in Cases of Heterogeneous Exposures to Ionizing Radiation. Radiation Research, 185(2), 151-162. <doi:10.1667/RR14145.1>.

Usage

```
estimate_hetero_mixed_poisson(
  case_data,
  fit_coeffs,
  fit_var_cov_mat,
  conf_int = 0.95,
  protracted_g_value = 1,
  gamma,
  gamma_error
)
```

Arguments

`case_data`	Case data in data frame form.
`fit_coeffs`	Fitting coefficients matrix.
`fit_var_cov_mat`	Fitting variance-covariance matrix.
`conf_int`	Confidence interval, 95% by default.
`protracted_g_value`	Protracted $G(x)$ value.
`gamma`	Survival coefficient of irradiated cells.
`gamma_error`	Error of the survival coefficient of irradiated cells.

Value

List containing estimated mixing proportions data frame, estimated yields data frame, estimated doses data frame, estimated fraction of irradiated blood data frame, AIC, and `conf_int_*` used.

Examples

```
#The fitting RDS result from the fitting module is needed. Alternatively, manual data
#frames that match the structure of the RDS can be used:


fit_coeffs <- data.frame(
  estimate   = c(0.001280319, 0.021038724, 0.063032534),
  std.error  = c(0.0004714055, 0.0051576170, 0.0040073856),
  statistic  = c(2.715961, 4.079156, 15.729091),
  p.value    = c(6.608367e-03, 4.519949e-05, 9.557291e-56),
  row.names =  c("coeff_C", "coeff_alpha", "coeff_beta")
)


fit_var_cov_mat <- data.frame(
  coeff_C      = c(2.222231e-07, -9.949044e-07,  4.379944e-07),
  coeff_alpha  = c(-9.949044e-07, 2.660101e-05, -1.510494e-05),
  coeff_beta   = c(4.379944e-07, -1.510494e-05, 1.605914e-05),
  row.names =  c("coeff_C", "coeff_alpha", "coeff_beta")
)


case_data <- data.frame(
  ID= "example1",
  N = 361,
  X = 100,
  C0 = 302,
  C1 = 28,
  C2 = 22,
  C3 = 8,
  C4 = 1,
  C5 = 0,
```

```
  y = 0.277,
  y_err = 0.0368,
  DI = 1.77,
  u = 10.4
)


#FUNCTION ESTIMATE_HETERO_MIXED_POISSON
estimate_hetero_mixed_poisson(
  case_data[1, ],
  fit_coeffs = as.matrix(fit_coeffs),
  fit_var_cov_mat = as.matrix(fit_var_cov_mat),
  conf_int = 0.95,
  protracted_g_value = 1,
  gamma = 1 / 2.7,
  gamma_error = 0
)
```

`estimate_partial_body_dolphin` *Partial-body dose estimation (Dolphin's method)*

Description

Method based on the paper by Dolphin, G. W. (1969). Biological Dosimetry with Particular Reference to Chromosome Aberration Analysis: A Review of Methods. International Atomic Energy Agency (IAEA) Retrieved from https://inis.iaea.org/search/search.aspx?orig_q=RN:45029080.

Usage

```
estimate_partial_body_dolphin(
  num_cases,
  case_data,
  fit_coeffs,
  fit_var_cov_mat,
  conf_int = 0.95,
  protracted_g_value = 1,
  genome_factor = 1,
  gamma,
  aberr_module = c("dicentrics", "translocations", "micronuclei")
)
```

Arguments

`num_cases`	number of cases.
`case_data`	Case data in data frame form.
`fit_coeffs`	Fitting coefficients matrix.

`fit_var_cov_mat`
: Fitting variance-covariance matrix.

`conf_int`
: Confidence interval, 95% by default.

`protracted_g_value`
: Protracted $G(x)$ value.

`genome_factor`
: Genomic conversion factor used in translocations, else 1.

`gamma`
: Survival coefficient of irradiated cells.

`aberr_module`
: Aberration module.

Value

List containing estimated doses data frame, observed fraction of cells scored which were irradiated, estimated fraction of irradiated blood data frame, AIC, and `conf_int_*` used.

Examples

```
#The fitting RDS result from the fitting module is needed. Alternatively, manual data
#frames that match the structure of the RDS can be used:
fit_coeffs <- data.frame(
    estimate   = c(0.001280319, 0.021038724, 0.063032534),
    std.error  = c(0.0004714055, 0.0051576170, 0.0040073856),
    statistic  = c(2.715961, 4.079156, 15.729091),
    p.value    = c(6.608367e-03, 4.519949e-05, 9.557291e-56),
    row.names =  c("coeff_C", "coeff_alpha", "coeff_beta")
  )


fit_var_cov_mat <- data.frame(
  coeff_C      = c(2.222231e-07, -9.949044e-07,  4.379944e-07),
  coeff_alpha  = c(-9.949044e-07, 2.660101e-05, -1.510494e-05),
  coeff_beta   = c(4.379944e-07, -1.510494e-05, 1.605914e-05),
  row.names =  c("coeff_C", "coeff_alpha", "coeff_beta")
)


case_data <- data.frame(
  ID= "example1",
  N = 361,
  X = 100,
  C0 = 302,
  C1 = 28,
  C2 = 22,
  C3 = 8,
  C4 = 1,
  C5 = 0,
  y = 0.277,
  y_err = 0.0368,
  DI = 1.77,
  u = 10.4
)
```

```
#FUNCTION ESTIMATE_PARTIAL_BODY_DOLPHIN
estimate_partial_body_dolphin(
  num_cases = 1,
  case_data = case_data,
  fit_coeffs = as.matrix(fit_coeffs),
  fit_var_cov_mat = as.matrix(fit_var_cov_mat),
  conf_int = 0.95,
  gamma = 1 / 2.7,
  aberr_module = "dicentrics"
)
```

`estimate_whole_body_delta` *Whole-body dose estimation (delta method)*

Description

Method based on 2001 manual by the International Atomic Energy Agency (IAEA). Cytogenetic Analysis for Radiation Dose Assessment, Technical Reports Series (2001). Retrieved from https://www.iaea.org/publications/6303/cytogenetic-analysis-for-radiation-dose-assessment.

Usage

```
estimate_whole_body_delta(
  num_cases,
  case_data,
  fit_coeffs,
  fit_var_cov_mat,
  conf_int = 0.95,
  protracted_g_value = 1,
  aberr_module = c("dicentrics", "translocations", "micronuclei")
)
```

Arguments

`num_cases`	number of cases.
`case_data`	Case data in data frame form.
`fit_coeffs`	Fitting coefficients matrix.
`fit_var_cov_mat`	Fitting variance-covariance matrix.
`conf_int`	Confidence interval, 95% by default.
`protracted_g_value`	Protracted $G(x)$ value.
`aberr_module`	Aberration module.

Value

List containing estimated doses data frame, AIC, and `conf_int` used.

Examples

```
#The fitting RDS result from the fitting module is needed. Alternatively, manual data
#frames that match the structure of the RDS can be used:
fit_coeffs <- data.frame(
  estimate   = c(0.001280319, 0.021038724, 0.063032534),
  std.error  = c(0.0004714055, 0.0051576170, 0.0040073856),
  statistic  = c(2.715961, 4.079156, 15.729091),
  p.value    = c(6.608367e-03, 4.519949e-05, 9.557291e-56),
  row.names =  c("coeff_C", "coeff_alpha", "coeff_beta")
)


fit_var_cov_mat <- data.frame(
  coeff_C      = c(2.222231e-07, -9.949044e-07,  4.379944e-07),
  coeff_alpha  = c(-9.949044e-07, 2.660101e-05, -1.510494e-05),
  coeff_beta   = c(4.379944e-07, -1.510494e-05, 1.605914e-05),
  row.names =  c("coeff_C", "coeff_alpha", "coeff_beta")
)


case_data <- data.frame(
  ID= "example1",
  N = 361,
  X = 100,
  C0 = 302,
  C1 = 28,
  C2 = 22,
  C3 = 8,
  C4 = 1,
  C5 = 0,
  y = 0.277,
  y_err = 0.0368,
  DI = 1.77,
  u = 10.4
)

#FUNCTION ESTIMATE_WHOLE_BODY_DELTA
estimate_whole_body_delta(
  num_cases = 1,
  case_data = case_data,
  fit_coeffs = as.matrix(fit_coeffs),
  fit_var_cov_mat = as.matrix(fit_var_cov_mat),
  conf_int = 0.95,
  protracted_g_value = 1,
  aberr_module = "dicentrics"
)
```

`estimate_whole_body_merkle` *Whole-body dose estimation (Merkle's method)*

Description

Method based on the paper by Merkle, W. (1983). Statistical methods in regression and calibration analysis of chromosome aberration data. Radiation and Environmental Biophysics, 21(3), 217-233. <doi:10.1007/BF01323412>.

Usage

```
estimate_whole_body_merkle(
  num_cases,
  case_data,
  fit_coeffs,
  fit_var_cov_mat,
  conf_int_yield = 0.83,
  conf_int_curve = 0.83,
  protracted_g_value = 1,
  genome_factor = 1,
  aberr_module = c("dicentrics", "translocations", "micronuclei")
)
```

Arguments

`num_cases`	number of cases.
`case_data`	Case data in data frame form.
`fit_coeffs`	Fitting coefficients matrix.
`fit_var_cov_mat`	Fitting variance-covariance matrix.
`conf_int_yield`	Confidence interval of the yield, 83% by default.
`conf_int_curve`	Confidence interval of the curve, 83% by default.
`protracted_g_value`	Protracted $G(x)$ value.
`genome_factor`	Genomic conversion factor used in translocations, else 1.
`aberr_module`	Aberration module.

Value

List containing estimated doses data frame, AIC, and `conf_int_*` used.

Examples

```
#The fitting RDS result from the fitting module is needed. Alternatively, manual data
#frames that match the structure of the RDS can be used:
fit_coeffs <- data.frame(
  estimate   = c(0.001280319, 0.021038724, 0.063032534),
  std.error  = c(0.0004714055, 0.0051576170, 0.0040073856),
  statistic  = c(2.715961, 4.079156, 15.729091),
  p.value    = c(6.608367e-03, 4.519949e-05, 9.557291e-56),
  row.names =  c("coeff_C", "coeff_alpha", "coeff_beta")
)


fit_var_cov_mat <- data.frame(
  coeff_C      = c(2.222231e-07, -9.949044e-07,  4.379944e-07),
  coeff_alpha  = c(-9.949044e-07, 2.660101e-05, -1.510494e-05),
  coeff_beta   = c(4.379944e-07, -1.510494e-05, 1.605914e-05),
  row.names =  c("coeff_C", "coeff_alpha", "coeff_beta")
)


case_data <- data.frame(
  ID= "example1",
  N = 361,
  X = 100,
  C0 = 302,
  C1 = 28,
  C2 = 22,
  C3 = 8,
  C4 = 1,
  C5 = 0,
  y = 0.277,
  y_err = 0.0368,
  DI = 1.77,
  u = 10.4
)

#FUNCTION ESTIMATE_WHOLE_BODY_MERKLE
estimate_whole_body_merkle(
  num_cases = 1,
  case_data = case_data,
  fit_coeffs = as.matrix(fit_coeffs),
  fit_var_cov_mat = as.matrix(fit_var_cov_mat),
  conf_int_yield = 0.83,
  conf_int_curve = 0.83,
  protracted_g_value = 1,
  aberr_module = "dicentrics"
)
```

`fit` *Perform dose-effect fitting algorithm*

Description

Perform dose-effect fitting. A generalized linear model (GLM) is used by default, with a maximum likelihood estimation (MLE) as a fallback method.

Usage

```
fit(
  count_data,
  model_formula,
  model_family,
  fit_link = "identity",
  aberr_module = c("dicentrics", "translocations", "micronuclei"),
  algorithm = c("glm", "maxlik")
)
```

Arguments

count_data	Count data in data frame form.
model_formula	Model formula.
model_family	Model family.
fit_link	Family link.
aberr_module	Aberration module.
algorithm	Optional selection of algorithm to be used, either "glm" (for GLM) or "maxlik" (for MLE). By default, "glm" is used, with "maxlik" as a fallback method.

Details

The GLM method is based on the paper by Edwards, A. A. et al. (1979). Radiation induced chromosome aberrations and the Poisson distribution. Radiation and Environmental Biophysics, 16(2), 89-100. <doi:10.1007/BF01323216>.

The MLE method is based on the paperby Oliveira, M. et al. (2016). Zero-inflated regression models for radiation-induced chromosome aberration data: A comparative study. Biometrical Journal, 58(2), 259-279. <doi:10.1002/bimj.201400233>.

Value

List object containing fit results either using GLM or maxLik optimization.

Examples

```
count_data <- data.frame(
  D = c(0, 0.1, 0.25, 0.5, 0.75, 1, 1.5, 2, 3, 4, 5),
  N = c(5000, 5002, 2008, 2002, 1832, 1168, 562, 333, 193, 103, 59),
  X = c(8, 14, 22, 55, 100, 109, 100, 103, 108, 103, 107),
  C0 = c(4992, 4988, 1987, 1947, 1736, 1064, 474, 251, 104, 35, 11),
  C1 = c(8, 14, 20, 55, 92, 99, 76, 63, 72, 41, 19),
  C2 = c(0, 0, 1, 0, 4, 5, 12, 17, 15, 21, 11),
  C3 = c(0, 0, 0, 0, 0, 0, 0, 2, 2, 4, 9),
```

```
  C4 = c(0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 6),
  C5 = c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3),
  mean = c(0.0016, 0.0028, 0.0110, 0.0275, 0.0546, 0.0933, 0.178, 0.309, 0.560, 1, 1.81),
 var = c(0.00160, 0.00279, 0.0118, 0.0267, 0.0560, 0.0933, 0.189, 0.353, 0.466, 0.882, 2.09),
  DI = c(0.999, 0.997, 1.08, 0.973, 1.03, 0.999, 1.06, 1.14, 0.834, 0.882, 1.15),
  u = c(-0.0748, -0.135, 2.61, -0.861, 0.790, -0.0176, 1.08, 1.82, -1.64, -0.844, 0.811)
)


fit(count_data = count_data,
    model_formula = "lin-quad",
    model_family = "automatic",
    fit_link = "identity",
    aberr_module = "dicentrics",
    algorithm = "maxlik")
```

`fit_glm_method` *Perform GLM (Generalised Linear Model) fitting*

Description

Method based on the paper by Edwards, A. A. et al. (1979). Radiation induced chromosome aberrations and the Poisson distribution. Radiation and Environmental Biophysics, 16(2), 89-100. <doi:10.1007/BF01323216>.

Usage

```
fit_glm_method(
  count_data,
  model_formula,
  model_family = c("automatic", "poisson", "quasipoisson", "nb2"),
  fit_link = "identity",
  aberr_module = c("dicentrics", "translocations", "micronuclei")
)
```

Arguments

`count_data`	Count data in data frame form.
`model_formula`	Model formula.
`model_family`	Model family.
`fit_link`	Family link.
`aberr_module`	Aberration module.

Value

List object containing GLM fit results.

fit_maxlik_method *Perform max-likelihood optimization fitting*

Description

Method based on the paper by Oliveira, M. et al. (2016). Zero-inflated regression models for radiation-induced chromosome aberration data: A comparative study. Biometrical Journal, 58(2), 259-279. <doi:10.1002/bimj.201400233>.

Usage

```
fit_maxlik_method(
  data,
  model_formula,
  model_family = c("automatic", "poisson", "quasipoisson", "nb2"),
  fit_link,
  aberr_module = c("dicentrics", "translocations", "micronuclei")
)
```

Arguments

data	Count data.
model_formula	Model formula.
model_family	Model family.
fit_link	Family link.
aberr_module	Aberration module.

Value

List object containing maxLik fit results.

fun.curve *Calculate curve*

Description

Calculate curve

Usage

```
fun.curve(d, coef, curve_type)
```

Arguments

d	dose
coef	curve coefficients
curve_type	type of curve: lin_quad, lin

Value

Numeric value.

---

fun.estimate.criticality

Calculate absorbed dose for mixed fields exposure

Description

Calculate absorbed dose for mixed fields exposure

Usage

```
fun.estimate.criticality(
  num_cases,
  dics,
  cells,
  coef_gamma,
  cov_gamma,
  coef_neutron,
  cov_neutron,
  ratio,
  p
)
```

Arguments

num_cases	number of cases to estimate.
dics	dicentrics.
cells	total cells.
coef_gamma	Coefficients for gamma curve.
cov_gamma	Covariance matrix for gamma curve.
coef_neutron	Coefficients for neutron curve.
cov_neutron	Covariance matrix for neutron curve.
ratio	gamma-neutron ratio.
p	photon contribution in mixed curve.

Value

dose estimation

Examples

```
#Results from the fitting module
fit_results_gamma <- system.file("extdata",
                                 "gamma_dicentrics-fitting-results.rds",
                                 package = "biodosetools") %>%
  readRDS()

fit_results_neutrons <- system.file("extdata",
                                    "neutrons-mixed-dicentrics-fitting-results.rds",
                                    package = "biodosetools") %>%
  readRDS()

#FUNCTION TO ESTIMATE DOSE IN CRITICALITY ACCIDENTS
fun.estimate.criticality(num_cases = 2,
                         dics = c(380,456),
                         cells = c(218,567),
                         coef_gamma = fit_results_gamma[["fit_coeffs"]][,1],
                         cov_gamma = fit_results_gamma[["fit_var_cov_mat"]],
                         coef_neutron = fit_results_neutrons[["fit_coeffs"]][,1],
                         cov_neutron = fit_results_neutrons[["fit_var_cov_mat"]],
                         ratio = 1.2,
                         p = 0)
```

---

generate_plot_and_download

Plot at the UI and save

Description

Plot at the UI and save

Usage

```
generate_plot_and_download(
  data_type,
  reactive_data,
  output,
  input,
  manual,
  dose_plot,
  name_num
)
```

Arguments

data_type	gamma or neutron
reactive_data	access to the reactive data
output	connection with UI
input	what you upload in the UI
manual	logical, indicates if using manual input or file data.
dose_plot	the number of the dose case to plot.
name_num	associates a number to the name.

Value

plot at UI and saving plot server logic

---

get_deltamethod_std_err

Get standard errors using delta method

Description

Delta method for approximating the standard error of a transformation $g(X)$ of a random variable $X = (x1, x2, ...)$, given estimates of the mean and covariance matrix of X.

Usage

```
get_deltamethod_std_err(
  fit_is_lq,
  variable = c("dose", "fraction_partial", "fraction_hetero"),
  mean_estimate,
  cov_estimate,
  protracted_g_value = NA,
  d0 = NA
)
```

Arguments

fit_is_lq	Whether the fit is linear quadratic (TRUE) or linear (FALSE).
variable	Variable resulting of the transformation $g(X)$.
mean_estimate	The estimated mean of X.
cov_estimate	The estimated covariance matrix of X.
protracted_g_value	Protracted $G(x)$ value.
d0	Survival coefficient of irradiated cells.

Value

Numeric value containing the standard error of the dose estimate.

include_help *Include Markdown help*

Description

Include Markdown help

Usage

```
include_help(...)
```

Arguments

...	Character vector specifying directory and or file to point to inside the current package.

load_manual_data *Load manual data*

Description

Load manual data

Usage

```
load_manual_data(data_type, input, reactive_data, output, session)
```

Arguments

data_type	gamma or neutron
input	what you upload in the UI
reactive_data	access to the reactive data
output	output for the UI
session	let interaction with the shiny UI

Value

manual curve information tables.

load_rds_data *Load and show rds files*

Description

Load and show rds files

Usage

```
load_rds_data(input_data, data_type, reactive_data, output, session)
```

Arguments

input_data	what you upload as rds file in the UI.
data_type	gamma or neutron.
reactive_data	access to the reactive data.
output	output for the UI.
session	let interaction with the shiny UI.

Value

uploaded rds file's curve information-tables.

load_rmd_report *Load RMarkdown report*

Description

Load RMarkdown report

Usage

```
load_rmd_report(...)
```

Arguments

...	Character vector specifying directory and or file to point to inside the current package.

M_estimate *Calculate algB*

Description

Calculate algB

Usage

```
M_estimate(x, iter_loc = 50, iter_scale = 1000)
```

Arguments

x	vector of n observations.
iter_loc	number of iteration steps for location estimate (default=50).
iter_scale	number of iteration steps for scale estimate (default=1000).

Value

Numeric value of zscore using algB.

Examples

```
X = c(3.65, 2.5, 4.85)

M_estimate(x = as.numeric(X),
           iter_loc=50,
           iter_scale=1000)
```

plot_deviation_all *Plot Deviaition from ref dose all blind samples*

Description

Plot Deviaition from ref dose all blind samples

Usage

```
plot_deviation_all(zscore, select_method, place)
```

Arguments

zscore	zscore vector to plot for each lab.
select_method	chosen algorithm for zscore.
place	UI or save.

Value

ggplot2 object.

Examples

```
data_frame_zscore <- data.frame(
  Lab = factor(c("A1", "A2", "A1", "A2", "A1", "A2")),
  Sample = factor(c(1, 1, 2, 2, 3, 3)),
  Type = rep("manual", 6),
  Reference = c(2.56, 3.41, 4.54, 2.56, 3.41, 4.54),
  Dose = c(2.589230, 2.610970, 3.146055, 3.018682, 4.259400, 3.989222),
 Deviation = c(0.02922998, -0.79902999, -1.39394510, 0.45868228, 0.84939953, -0.55077813),
  Zscore = c(1.677656, 2.925426, -2.585664, -3.833434, -1.295906, -2.543676),
  stringsAsFactors = FALSE
)


plot_deviation_all(
  zscore = data_frame_zscore,
  select_method = "algA",
  place = "UI"
)
```

---

plot_estimated_dose_curve

Plot dose estimation curve

Description

Plot dose estimation curve

Usage

```
plot_estimated_dose_curve(
  est_doses,
  fit_coeffs,
  fit_var_cov_mat,
  protracted_g_value = 1,
  conf_int_curve,
  aberr_name,
  place
)
```

Arguments

est_doses	List of dose estimations results from estimate_*() family of functions.
fit_coeffs	Fitting coefficients matrix.

`fit_var_cov_mat`
: Fitting variance-covariance matrix.

`protracted_g_value`
: Protracted $G(x)$ value.

`conf_int_curve`
: Confidence interval of the curve.

`aberr_name`
: Name of the aberration to use in the y-axis.

`place`
: UI or save.

Value

`ggplot2` object.

Examples

```
#The fitting RDS result from the fitting module is needed. Alternatively, manual data
#frames that match the structure of the RDS can be used:
fit_coeffs <- data.frame(
  estimate   = c(0.001280319, 0.021038724, 0.063032534),
  std.error  = c(0.0004714055, 0.0051576170, 0.0040073856),
  statistic  = c(2.715961, 4.079156, 15.729091),
  p.value    = c(6.608367e-03, 4.519949e-05, 9.557291e-56),
  row.names =  c("coeff_C", "coeff_alpha", "coeff_beta")
)

fit_var_cov_mat <- data.frame(
  coeff_C      = c(2.222231e-07, -9.949044e-07,  4.379944e-07),
  coeff_alpha  = c(-9.949044e-07, 2.660101e-05, -1.510494e-05),
  coeff_beta   = c(4.379944e-07, -1.510494e-05, 1.605914e-05),
  row.names =  c("coeff_C", "coeff_alpha", "coeff_beta")
)

results_whole_merkle <- list(
  list(
    est_doses = data.frame(
      lower = 1.541315,
      estimate = 1.931213,
      upper = 2.420912
    ),
    est_yield = data.frame(
      lower = 0.1980553,
      estimate = 0.277,
      upper = 0.3841655
    ),
    AIC = 7.057229,
    conf_int = c(yield_curve = 0.83)
  )
)


#FUNCTION PLOT_ESTIMATED_DOSE_CURVE
plot_estimated_dose_curve(
```

```
  est_doses = list(whole = results_whole_merkle),
  fit_coeffs = as.matrix(fit_coeffs),
  fit_var_cov_mat = as.matrix(fit_var_cov_mat),
  protracted_g_value = 1,
  conf_int_curve = 0.95,
  aberr_name = "Micronuclei",
  place = "UI"
)
```

`plot_estimated_dose_curve_mx` *Criticality Plot*

Description

Criticality Plot

Usage

```
plot_estimated_dose_curve_mx(
  name,
  est_doses,
  fit_coeffs,
  fit_var_cov_mat,
  curve_type,
  protracted_g_value = 1,
  conf_int_curve = 0.95,
  place
)
```

Arguments

`name`	the dose to plot.
`est_doses`	List of dose estimations results.
`fit_coeffs`	Fitting coefficients matrix.
`fit_var_cov_mat`	Fitting variance-covariance matrix.
`curve_type`	gamma or neutron.
`protracted_g_value`	Protracted $G(x)$ value.
`conf_int_curve`	Confidence interval of the curve.
`place`	UI, report or save. Where the plot will be displayed.

Value

ggcurve

Examples

```
#The fitting RDS result from the fitting module is needed. Alternatively, manual data
#frames that match the structure of the RDS can be used:
fit_coeffs <- data.frame(
  estimate   = c(0.001280319, 0.021038724, 0.063032534),
  std.error  = c(0.0004714055, 0.0051576170, 0.0040073856),
  statistic  = c(2.715961, 4.079156, 15.729091),
  p.value    = c(6.608367e-03, 4.519949e-05, 9.557291e-56),
  row.names =  c("coeff_C", "coeff_alpha", "coeff_beta")
)


fit_var_cov_mat <- data.frame(
  coeff_C      = c(2.222231e-07, -9.949044e-07,  4.379944e-07),
  coeff_alpha  = c(-9.949044e-07, 2.660101e-05, -1.510494e-05),
  coeff_beta   = c(4.379944e-07, -1.510494e-05, 1.605914e-05),
  row.names =  c("coeff_C", "coeff_alpha", "coeff_beta")
)


est_doses <- list(
  list(
    gamma = c(est = 2.173586, lwr = 1.784314, upr = 2.562857),
    neutron = c(est = 1.811322, lwr = 1.486929, upr = 2.135715),
    total = c(est = 3.984907, lwr = 3.271243, upr = 4.698572)
  )
  )

#FUNCTION PLOT_ESTIMATED_DOSE_CURVE_MX
plot_estimated_dose_curve_mx(
  name = "Sample1",
  est_doses = est_doses[[1]],
  fit_coeffs = as.matrix(fit_coeffs)[,1],
  fit_var_cov_mat = as.matrix(fit_var_cov_mat),
  curve_type = "gamma",
  protracted_g_value = 1,
  conf_int_curve = 0.95,
  place = "UI")
```

`plot_fit_dose_curve` *Plot fit dose curve*

Description

Plot fit dose curve

Usage

```
plot_fit_dose_curve(fit_results_list, aberr_name, place)
```

Arguments

fit_results_list	List of fit results.
aberr_name	Name of the aberration to use in the y-axis.
place	Where the plot will be displayed.

Value

ggplot2 object.

Examples

```
#In order to plot the fitted curve we need the output of the fit() function:
count_data <- data.frame(
  D = c(0, 0.1, 0.25, 0.5, 0.75, 1, 1.5, 2, 3, 4, 5),
  N = c(5000, 5002, 2008, 2002, 1832, 1168, 562, 333, 193, 103, 59),
  X = c(8, 14, 22, 55, 100, 109, 100, 103, 108, 103, 107),
  C0 = c(4992, 4988, 1987, 1947, 1736, 1064, 474, 251, 104, 35, 11),
  C1 = c(8, 14, 20, 55, 92, 99, 76, 63, 72, 41, 19),
  C2 = c(0, 0, 1, 0, 4, 5, 12, 17, 15, 21, 11),
  C3 = c(0, 0, 0, 0, 0, 0, 0, 2, 2, 4, 9),
  C4 = c(0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 6),
  C5 = c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3),
  mean = c(0.0016, 0.0028, 0.0110, 0.0275, 0.0546, 0.0933, 0.178, 0.309, 0.560, 1, 1.81),
 var = c(0.00160, 0.00279, 0.0118, 0.0267, 0.0560, 0.0933, 0.189, 0.353, 0.466, 0.882, 2.09),
  DI = c(0.999, 0.997, 1.08, 0.973, 1.03, 0.999, 1.06, 1.14, 0.834, 0.882, 1.15),
  u = c(-0.0748, -0.135, 2.61, -0.861, 0.790, -0.0176, 1.08, 1.82, -1.64, -0.844, 0.811)
)


fit_results <- fit(count_data = count_data,
                   model_formula = "lin-quad",
                   model_family = "automatic",
                   fit_link = "identity",
                   aberr_module = "dicentrics",
                   algorithm = "maxlik")

#FUNTION PLOT_FIT_DOSE_CURVE()
plot_fit_dose_curve(
  fit_results,
  aberr_name = "Dicentrics",
  place = "UI"
)
```

---

plot_interlab_deviation

Plot deviation

Description

Plot deviation

Usage

```
plot_interlab_deviation(zscore, sum_table, place)
```

Arguments

zscore	data frame with Lab, Sample, Type, Reference, Dose, Deviation and Zscore.
sum_table	summary table.
place	UI or save.

Value

ggplot2 object.

Examples

```
data_frame_zscore <- data.frame(
  Lab = factor(c("A1", "A2", "A1", "A2", "A1", "A2")),
  Sample = factor(c(1, 1, 2, 2, 3, 3)),
  Type = rep("manual", 6),
  Reference = c(2.56, 3.41, 4.54, 2.56, 3.41, 4.54),
  Dose = c(2.589230, 2.610970, 3.146055, 3.018682, 4.259400, 3.989222),
 Deviation = c(0.02922998, -0.79902999, -1.39394510, 0.45868228, 0.84939953, -0.55077813),
  Zscore = c(1.677656, 2.925426, -2.585664, -3.833434, -1.295906, -2.543676),
  stringsAsFactors = FALSE
)

sum_table <- data.frame(
  Lab = c("A1", "A2", "A1", "A2", "A1", "A2"),
 Module = c("dicentrics", "dicentrics", "dicentrics", "dicentrics", "dicentrics", "dicentrics"),
  Type = rep("manual", 6),
  Sample = c(1, 1, 2, 2, 3, 3),
  N = c(200, 200, 200, 200, 200, 200),
  X = c(140, 111, 204, 147, 368, 253),
  estimate = c(2.589230, 2.610970, 3.146055, 3.018682, 4.259400, 3.989222),
  lower = c(2.083403, 2.231150, 2.614931, 2.622748, 3.882349, 3.552963),
  upper = c(3.208857, 3.045616, 3.785879, 3.471269, 4.688910, 4.487336),
  y = c(0.700, 0.555, 1.020, 0.735, 1.840, 1.265),
  y.err = c(0.0610, 0.0495, 0.0733, 0.0556, 0.0923, 0.0785),
  DI = c(1.0625, 0.8818, 1.0539, 0.8406, 0.9255, 0.9731),
  u = c(0.6252, -1.1840, 0.5389, -1.5951, -0.7442, -0.2692),
  stringsAsFactors = FALSE
)


#FUNCTION PLOT_INTERLAB_DEVIATION:
plot_interlab_deviation(
  zscore = data_frame_zscore,
```

```
  sum_table = sum_table,
  place = "UI"
)
```

`plot_interlab_v2` *Plot zscore v2*

Description

Plot zscore v2

Usage

```
plot_interlab_v2(zscore, select_method, sum_table, place)
```

Arguments

`zscore`	data frame with Lab, Sample, Type, Reference, Dose, Deviation and Zscore.
`select_method`	Zscore algorithm.
`sum_table`	summary table.
`place`	UI or save.

Value

`ggplot2` object.

Examples

```
data_frame_zscore <- data.frame(
  Lab = factor(c("A1", "A2", "A1", "A2", "A1", "A2")),
  Sample = factor(c(1, 1, 2, 2, 3, 3)),
  Type = rep("manual", 6),
  Reference = c(2.56, 3.41, 4.54, 2.56, 3.41, 4.54),
  Dose = c(2.589230, 2.610970, 3.146055, 3.018682, 4.259400, 3.989222),
 Deviation = c(0.02922998, -0.79902999, -1.39394510, 0.45868228, 0.84939953, -0.55077813),
  Zscore = c(1.677656, 2.925426, -2.585664, -3.833434, -1.295906, -2.543676),
  stringsAsFactors = FALSE
)


sum_table <- data.frame(
  Lab = c("A1", "A2", "A1", "A2", "A1", "A2"),
 Module = c("dicentrics", "dicentrics", "dicentrics", "dicentrics", "dicentrics", "dicentrics"),
  Type = rep("manual", 6),
  Sample = c(1, 1, 2, 2, 3, 3),
  N = c(200, 200, 200, 200, 200, 200),
  X = c(140, 111, 204, 147, 368, 253),
  estimate = c(2.589230, 2.610970, 3.146055, 3.018682, 4.259400, 3.989222),
  lower = c(2.083403, 2.231150, 2.614931, 2.622748, 3.882349, 3.552963),
```

```
  upper = c(3.208857, 3.045616, 3.785879, 3.471269, 4.688910, 4.487336),
  y = c(0.700, 0.555, 1.020, 0.735, 1.840, 1.265),
  y.err = c(0.0610, 0.0495, 0.0733, 0.0556, 0.0923, 0.0785),
  DI = c(1.0625, 0.8818, 1.0539, 0.8406, 0.9255, 0.9731),
  u = c(0.6252, -1.1840, 0.5389, -1.5951, -0.7442, -0.2692),
  stringsAsFactors = FALSE
)


#FUNCTION PLOT_INTERLAB_V2
plot_interlab_v2(zscore = data_frame_zscore,
                 select_method = "algA",
                 sum_table = sum_table,
                 place = "UI"
                 )
```

`plot_triage` *Plot triage*

Description

Plot triage

Usage

```
plot_triage(num_cases, est_doses_whole, est_doses_partial, assessment, place)
```

Arguments

`num_cases`	number of cases.
`est_doses_whole`	dose estimation list of cases.
`est_doses_partial`	dose estimation list of cases.
`assessment`	partial or whole.
`place`	UI or save.

Value

`ggplot2` object.

Examples

```
est_doses_whole <- data.frame(
    ID = c("example1", "example2"),
    lower = c(1.407921, 2.604542),
    estimate = c(1.931245, 3.301014),
    upper = c(2.632199, 4.221749)
```

```
  )


plot_triage(
  num_cases = 2,
  est_doses_whole ,
  est_doses_partial = NULL,
  assessment = "whole",
  place = "UI"
)
```

`plot_triage_interlab` *Plot triage interlab*

Description

Plot triage interlab

Usage

```
plot_triage_interlab(line_triage, sum_table, place)
```

Arguments

`line_triage`	reference value for selected sample.
`sum_table`	summary table.
`place`	UI or save.

Value

`ggplot2` object.

Examples

```
line_triage <- list(`1` = 2.56, `2` = 3.41, `3` = 4.54)

sum_table <- data.frame(
  Lab = c("A1", "A2", "A1", "A2", "A1", "A2"),
 Module = c("dicentrics", "dicentrics", "dicentrics", "dicentrics", "dicentrics", "dicentrics"),
  Type = c("manual", "manual", "manual", "manual", "manual", "manual"),
  Sample = c(1, 1, 2, 2, 3, 3),
  N = c(200, 200, 200, 200, 200, 200),
  X = c(140, 111, 204, 147, 368, 253),
  estimate = c(2.589230, 2.610970, 3.146055, 3.018682, 4.259400, 3.989222),
  lower = c(2.083403, 2.231150, 2.614931, 2.622748, 3.882349, 3.552963),
  upper = c(3.208857, 3.045616, 3.785879, 3.471269, 4.688910, 4.487336),
  y = c(0.700, 0.555, 1.020, 0.735, 1.840, 1.265),
  y.err = c(0.0610, 0.0495, 0.0733, 0.0556, 0.0923, 0.0785),
```

```
  DI = c(1.0625, 0.8818, 1.0539, 0.8406, 0.9255, 0.9731),
  u = c(0.6252, -1.1840, 0.5389, -1.5951, -0.7442, -0.2692),
  stringsAsFactors = FALSE
)

plot_triage_interlab(
  line_triage = line_triage,
  sum_table = sum_table,
  place = "UI"
)
```

`plot_zscore_all` *Plot Z-scores all blind samples*

Description

Plot Z-scores all blind samples

Usage

```
plot_zscore_all(zscore, select_method, place)
```

Arguments

`zscore`	zscore vector to plot for each lab.
`select_method`	chosen algorithm for zscore.
`place`	UI or save.

Value

`ggplot2` object.

Examples

```
data_frame_zscore <- data.frame(
  Lab = factor(c("A1", "A2", "A1", "A2", "A1", "A2")),
  Sample = factor(c(1, 1, 2, 2, 3, 3)),
  Type = rep("manual", 6),
  Reference = c(2.56, 3.41, 4.54, 2.56, 3.41, 4.54),
  Dose = c(2.589230, 2.610970, 3.146055, 3.018682, 4.259400, 3.989222),
 Deviation = c(0.02922998, -0.79902999, -1.39394510, 0.45868228, 0.84939953, -0.55077813),
  Zscore = c(1.677656, 2.925426, -2.585664, -3.833434, -1.295906, -2.543676),
  stringsAsFactors = FALSE
)


plot_zscore_all(
  zscore = data_frame_zscore,
  select_method = "algA",
```

```
  place = "UI"
)
```

`prepare_maxlik_count_data` *Prepare count data for max-likelihood optimization fitting*

Description

Prepare count data for max-likelihood optimization fitting

Usage

```
prepare_maxlik_count_data(
  count_data,
  model_formula,
  aberr_module = c("dicentrics", "translocations", "micronuclei")
)
```

Arguments

`count_data`	Count data in data frame form.
`model_formula`	Model formula.
`aberr_module`	Aberration module.

Value

Data frame of parsed count data.

`project_yield` *Project yield into dose-effect fitting curve*

Description

Project yield into dose-effect fitting curve

Usage

```
project_yield(
  yield,
  type = "estimate",
  general_fit_coeffs,
  general_fit_var_cov_mat = NULL,
  protracted_g_value = 1,
  conf_int = 0.95
)
```

Arguments

`yield`	Yield to be projected.
`type`	Type of yield calculation. Can be "estimate", "lower", or "upper".
`general_fit_coeffs`	Generalised fit coefficients matrix.
`general_fit_var_cov_mat`	Generalised variance-covariance matrix.
`protracted_g_value`	Protracted $G(x)$ value.
`conf_int`	Curve confidence interval, 95% by default.

Value

Numeric value of projected dose.

Examples

```
fit_coeffs <- data.frame(
  estimate   = c(0.001280319, 0.021038724, 0.063032534),
  std.error  = c(0.0004714055, 0.0051576170, 0.0040073856),
  statistic  = c(2.715961, 4.079156, 15.729091),
  p.value    = c(6.608367e-03, 4.519949e-05, 9.557291e-56),
  row.names =  c("coeff_C", "coeff_alpha", "coeff_beta")
)



project_yield(yield = 0.67,
              type = "estimate",
              general_fit_coeffs = fit_coeffs[, "estimate"],
              general_fit_var_cov_mat = NULL,
              protracted_g_value = 1,
              conf_int = 0.95)
```

`protracted_g_function` *Calculate protracted function* $G(x)$

Description

Calculation based on the paper by Lea, D. E. & Catcheside, D. G. (1942). The mechanism of the induction by radiation of chromosome aberrations in*Tradescantia*. Journal of Genetics, 44(2-3), 216-245. <doi:10.1007/BF02982830>.

Usage

```
protracted_g_function(time, time_0 = 2)
```

Arguments

time	Time over which the irradiation occurred.
time_0	The mean lifetime of the breaks, which has been shown to be on the order of ~ 2 hours (default value).

Value

Numeric value of $G(x)$.

Examples

```
protracted_g_function(time = 3,
                      time_0 = 2)
```

QHampel *Calculate QHampel*

Description

Calculate QHampel

Usage

```
QHampel(y, lab, tol.G1 = 1e-06)
```

Arguments

y	vector of data values.
lab	corresponding lab numbers.
tol.G1	decimal place accuracy of the numerator of s.star.

Value

Numeric value of zscore using QHampel algorithm.

Examples

```
X = c(3.65, 2.5, 4.85)

QHampel(y = as.numeric(X),
        lab = 1:length(X),
        tol.G1=0.000001)
```

run_app *Run the Shiny Application*

Description

Run the Shiny Application

Usage

```
run_app(...)
```

Arguments

...	A series of options to be used inside the app.

Value

Used for side-effect.

R_factor *Calculate R regression confidence factor*

Description

Calculate R regression confidence factor depending on selected confidence interval and type of fit.

Usage

```
R_factor(general_fit_coeffs, conf_int = 0.95)
```

Arguments

general_fit_coeffs	Generalised fit coefficients matrix.
conf_int	Confidence interval, 95% by default.

Value

Numeric value of R regression confidence factor.

summary_curve_tables *Summary and curve tables*

Description

Summary and curve tables

Usage

```
summary_curve_tables(num_labs, list_lab_names, all_rds)
```

Arguments

num_labs	number of laboratories participating.
list_lab_names	list with the laboratory names.
all_rds	list with the rds files.

Value

list(sorted_table_ilc, sorted_table_curve)

Examples

```
#fit_results_X is the output from the Estimation module

fit_results_A1 <- system.file("extdata", "A1_Estimation_results.rds", package = "biodosetools") %>%
  readRDS()

fit_results_A2 <- system.file("extdata", "A2_Estimation_results.rds", package = "biodosetools") %>%
  readRDS()



summary_curve_tables(num_labs = 2,
                     list_lab_names = c("A", "B"),
                     all_rds = list(fit_results_A1, fit_results_A2)
                     )
```

update_outputs *Update dose estimation box with yield values*

Description

Update dose estimation box with yield values

Usage

```
update_outputs(num_cases, reactive_data, output, yield_fun, manual, table)
```

Arguments

num_cases	number of cases to estimate.
reactive_data	to access reactive data.
output	output for the UI.
yield_fun	function for calculating yield, in utils_estimation.R
manual	logical, indicates if using manual input or file data.
table	the data input table.

Value

yield per dose. Numeric

---

u_test_plot *U-test*

Description

U-test

Usage

```
u_test_plot(dat, place)
```

Arguments

dat	data frame of data values.
place	UI or save.

Value

plot

Examples

```
dat <- data.frame(
  Lab = c("A1", "A2", "A1", "A2", "A1", "A2"),
 Module = c("dicentrics", "dicentrics", "dicentrics", "dicentrics", "dicentrics", "dicentrics"),
  Type = c("manual", "manual", "manual", "manual", "manual", "manual"),
  Sample = c(1, 1, 2, 2, 3, 3),
  N = c(200, 200, 200, 200, 200, 200),
  X = c(140, 111, 204, 147, 368, 253),
  estimate = c(2.589230, 2.610970, 3.146055, 3.018682, 4.259400, 3.989222),
```

```
  lower = c(2.083403, 2.231150, 2.614931, 2.622748, 3.882349, 3.552963),
  upper = c(3.208857, 3.045616, 3.785879, 3.471269, 4.688910, 4.487336),
  y = c(0.700, 0.555, 1.020, 0.735, 1.840, 1.265),
  y.err = c(0.0610, 0.0495, 0.0733, 0.0556, 0.0923, 0.0785),
  DI = c(1.0625, 0.8818, 1.0539, 0.8406, 0.9255, 0.9731),
  u = c(0.6252, -1.1840, 0.5389, -1.5951, -0.7442, -0.2692),
  stringsAsFactors = FALSE
)

u_test_plot(
  dat = dat,
  place = "UI"
)
```

`yield_boxplot` *Boxplot yield*

Description

Boxplot yield

Usage

```
yield_boxplot(dat, place)
```

Arguments

`dat`	data frame of data values.
`place`	UI or save.

Value

plot

Examples

```
dat <- data.frame(
  Lab = c("A1", "A2", "A1", "A2", "A1", "A2"),
 Module = c("dicentrics", "dicentrics", "dicentrics", "dicentrics", "dicentrics", "dicentrics"),
  Type = c("manual", "manual", "manual", "manual", "manual", "manual"),
  Sample = c(1, 1, 2, 2, 3, 3),
  N = c(200, 200, 200, 200, 200, 200),
  X = c(140, 111, 204, 147, 368, 253),
  estimate = c(2.589230, 2.610970, 3.146055, 3.018682, 4.259400, 3.989222),
  lower = c(2.083403, 2.231150, 2.614931, 2.622748, 3.882349, 3.552963),
  upper = c(3.208857, 3.045616, 3.785879, 3.471269, 4.688910, 4.487336),
  y = c(0.700, 0.555, 1.020, 0.735, 1.840, 1.265),
  y.err = c(0.0610, 0.0495, 0.0733, 0.0556, 0.0923, 0.0785),
  DI = c(1.0625, 0.8818, 1.0539, 0.8406, 0.9255, 0.9731),
```

```
  u = c(0.6252, -1.1840, 0.5389, -1.5951, -0.7442, -0.2692),
  stringsAsFactors = FALSE
)


yield_boxplot(
  dat =  dat,
  place = "UI"
)
```

`yield_error_fun` *Calculate yield error*

Description

Calculate yield error using Merkle's method

Usage

```
yield_error_fun(dose, general_fit_var_cov_mat = NULL, protracted_g_value = 1)
```

Arguments

`dose`	Numeric value of dose.
`general_fit_var_cov_mat`	Generalised variance-covariance matrix.
`protracted_g_value`	Protracted $G(x)$ value.

Value

Numeric value of yield error.

`yield_fun` *Calculate yield*

Description

Calculate yield

Usage

```
yield_fun(dose, general_fit_coeffs, protracted_g_value = 1)
```

Arguments

`dose`	Numeric value of dose.
`general_fit_coeffs`	Generalised fit coefficients matrix.
`protracted_g_value`	Protracted $G(x)$ value.

Value

Numeric value of yield.

Index