

Supplementary materials 2. Examples in R

The full **biodosetools** R API reference detailing the available functions and parameters can be found on the project's website at <https://biodosetools-team.github.io/biodosetools/reference/>.

2.1. *Dicentrics dose-effect fitting in R*

2.1.1. *Input count data*

The first step is to input the count data. This step is accomplished in R by calling the `calculate_aberr_table()` function. This will calculate the total number of cells (N), total number of aberrations (X), as well as mean (\bar{y}), variance (σ^2), dispersion index (σ^2/\bar{y}), and u -value.

```
R> count_data <- system.file("extdata", "count-data-barquinero-1995.csv",
+   package = "biodosetools"
+ ) %>%
+   utils::read.csv() %>%
+   calculate_aberr_table(type = "count")

R> count_data

# A tibble: 11 x 13
      D      N      X    C0    C1    C2    C3    C4    C5    mean    var
  <dbl> <int> <dbl> <int> <int> <int> <int> <int> <int>  <dbl>  <dbl>
1  0     5000      8  4992     8     0     0     0     0  0.0016 0.00160
2  0.1    5002     14  4988    14     0     0     0     0  0.00280 0.00279
3  0.25   2008     22  1987    20     1     0     0     0  0.0110 0.0118
4  0.5    2002     55  1947    55     0     0     0     0  0.0275 0.0267
5  0.75   1832    100  1736    92     4     0     0     0  0.0546 0.0560
6  1     1168    109  1064    99     5     0     0     0  0.0933 0.0933
7  1.5     562    100   474    76    12     0     0     0  0.178  0.189
8  2       333    103   251    63    17     2     0     0  0.309  0.353
9  3       193    108   104    72    15     2     0     0  0.560  0.466
10 4       103    103    35    41    21     4     2     0  1      0.882
11 5        59    107    11    19    11     9     6     3  1.81   2.09
# ... with 2 more variables: DI <dbl>, u <dbl>
```

2.1.2. *Perform fitting*

To perform the fitting in R we call the `fit()` function, whilst selecting the appropriate `aberr_module` ("dicentrics" in this example), `model_formula` ("lin-quad" or "lin" for LQ or L models, respectively), and `model_family` ("automatic", "poisson", or "quasipoisson") parameters:

```
R> fit_results <- fit(
+   count_data = count_data,
+   model_formula = "lin-quad",
+   model_family = "automatic",
+   aberr_module = "dicentrics"
+ )
```

The `fit_results` object is a list that contains all necessary information about the count data as well as options selected when performing the fitting. This is a vital step to ensure traceability and reproducibility. Below we can see its elements:

```
R> names(fit_results)
```

```
[1] "fit_raw_data"          "fit_formula_raw"      "fit_formula_tex"
[4] "fit_coeffs"           "fit_cor_mat"         "fit_var_cov_mat"
[7] "fit_dispersion"       "fit_model_statistics" "fit_algorithm"
[10] "fit_model_summary"
```

In particular, we can see how `fit_coeffs` matches the results obtained in the UI (Figure 5):

```
R> fit_results$fit_coeffs
```

	estimate	std.error	statistic	p.value
coeff_C	0.001280319	0.0004714055	2.715961	6.608367e-03
coeff_alpha	0.021038724	0.0051576170	4.079156	4.519949e-05
coeff_beta	0.063032534	0.0040073856	15.729091	9.557291e-56

To visualise the dose-effect curve, we call the `plot_fit_dose_curve()` function:

```
R> plot_fit_dose_curve(
+   fit_results,
+   aberr_name = "Dicentrics"
+ )
```

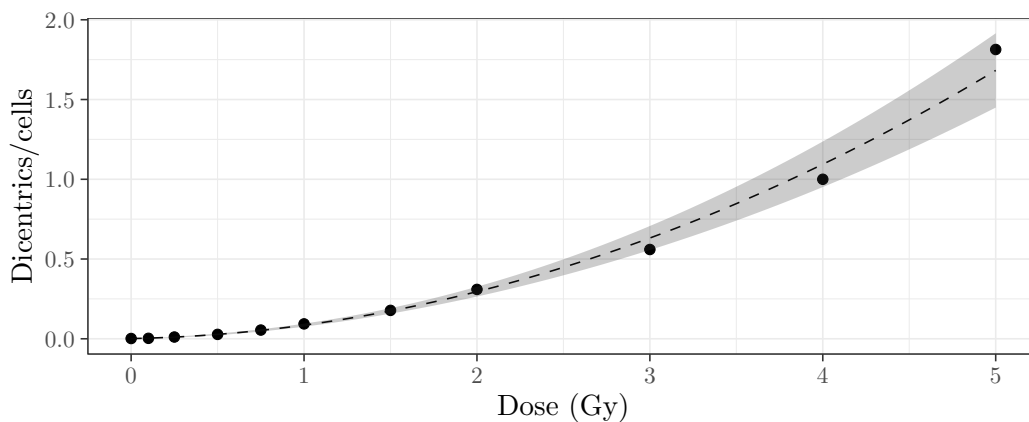


Figure S2.1. Plot of dose-effect curve generated by **biodosetools**. The grey shading indicates the uncertainties associated with the calibration curve.

2.2. *Dicentrics dose estimation in R*

2.2.1. *Input case data*

Next we can choose to either load the case data from a file or to input the data manually. The data is then parsed in R by calling the `calculate_aberr_table()` function. This will calculate the total number of cells (N), total number of aberrations (X), as well as mean (\bar{y}), standard error (σ), dispersion index (σ^2/\bar{y}), and u -value.

```

R> case_data <- system.file("extdata", "cases-data-partial.csv",
+   package = "biodosetools"
+ ) %>%
+   utils::read.csv(header = TRUE) %>%
+   calculate_aberr_table(
+     type = "case",
+     assessment_u = 1,
+     aberr_module = "dicentrics"
+   )

R> case_data

# A tibble: 1 x 12
      N      X    C0    C1    C2    C3    C4    C5      y y_err    DI
  <int> <int> <int> <int> <int> <int> <int> <int> <dbl> <dbl> <dbl>
1   361   100   302    28    22     8     1     0 0.277 0.0368 1.77
# ... with 1 more variable: u <dbl>

```

2.2.2. Perform dose estimation

Finally, to perform the dose estimation in R we can call the adequate `estimate_*()` functions and parameters depending on the characteristics of the accident. In this example, we will use `estimate_whole_body_merkle()` and `estimate_partial_body_dolphin()`. First of all, however, we will need to load the fit coefficients and variance-covariance matrix from the previously calculated `fit_results`:

```

R> fit_coeffs <- fit_results$fit_coeffs
R> fit_var_cov_mat <- fit_results$fit_var_cov_mat

```

After that is done, we can simply call `estimate_whole_body_merkle()` and `estimate_partial_body_dolphin()`:

```

R> results_whole_merkle <- estimate_whole_body_merkle(
+   case_data,
+   fit_coeffs,
+   fit_var_cov_mat,
+   conf_int_yield = 0.83,
+   conf_int_curve = 0.83,
+   protracted_g_value = 1,
+   aberr_module = "dicentrics"
+ )

R> results_partial <- estimate_partial_body_dolphin(
+   case_data,
+   fit_coeffs,
+   fit_var_cov_mat,
+   conf_int = 0.95,
+   gamma = 1 / 2.7,
+   aberr_module = "dicentrics"
+ )

```

To visualise the estimated doses, we call the `plot_estimated_dose_curve()` function:

```
R> plot_estimated_dose_curve(
+   est_doses = list(
+     whole = results_whole_merkle,
+     partial = results_partial
+   ),
+   fit_coeffs,
+   fit_var_cov_mat,
+   protracted_g_value = 1,
+   conf_int_curve = 0.95,
+   aberr_name = "Dicentrics"
+ )
```

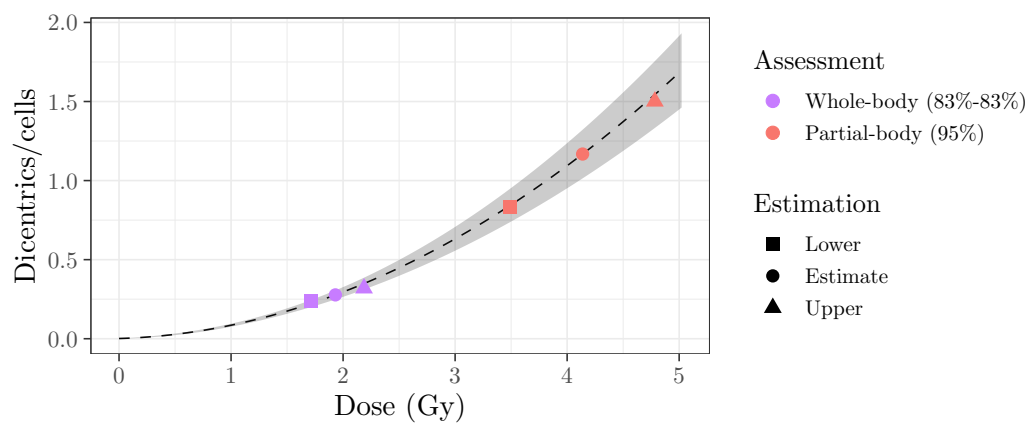


Figure S2.2. Plot of estimated doses generated by **biodosetools**. The grey shading indicates the uncertainties associated with the calibration curve.