Introduction to gdscIC50

```
## Warning: replacing previous import 'dplyr::collapse' by 'nlme::collapse' when
## loading 'gdscIC50'
```

Analysing raw data from the Genomics of Drug Sensitivity in Cancer resource

The Genomics of Drugs Sensitivity in Cancer (GDSC) is one of the largest public resources of information on drug sensitivity in cancer cells and molecular markers of drug response. High-throughput drug screens in human cancer cell cultures are used to identify genetic features of cancer cells that are predictive of drug sensitivity. GDSC data is available from the website http://www.cancerrxgene.org.

gdscIC50 is a package to process raw data from the GDSC project and fit dose response curves to individual experiments. From the fitted model sensitivity metrics are calculated including the half maximal inhibitory concentration (IC50) and the area under the curve (AUC). This is the model used to calculate IC50 and AUC values as presented on www.cancerrxgene.org and in Iorio, F. et al. Cell 2016 167(1):260-274. Furthermore the gdscIC50 package wrangles the nlme model outputs into dataframes, plots dose response curves, and prepares IC50 matrices to be used in an ANOVA analysis of the dose response data that links them to genetic biomarkers of drug sensitivity. The ANOVA analysis uses the Python package GDSCtools as detailed in Cokelaer, T et al. 2017.

Model fitting uses a non-linear mixed effects model (nlme) to model the sigmoidal dose response. This model is published - (Vis, D.J. et al. Pharmacogenomics 2016, 17(7):691-700) - and the original R code is available in the package djvMixedIC50 package.

From GDSC raw data to the dose response curve fit

In this vignette we will demonstrate how to wrangle a GDSC raw dataset to the format needed for the nlme model fit, before fitting and taking a look at the results.

GDSC raw data

A GDSC raw data file can be read in as csv format:

```
gdsc_raw_data <- read.csv("path_to_my_data_file/my_gdsc_raw_data.csv")
```

Here we will use the example data set included within the package. The example data set is real data from the GDSC screen. The experiments use 384 well or 1536 well plates, and in rare cases 96 well plates. In the example data some drugged wells are missing. This is often the case with GDSC data because some compounds are provided by collaborators and are not publicly released.

```
data("gdsc_example")
head(gdsc_example, 2)
#>   RESEARCH_PROJECT BARCODE SCAN_ID        DATE_CREATED           SCAN_DATE
#> 1          GDSC_SA    3230    2945 2015-02-12 23:00:00 2015-02-16 23:00:00
#> 2          GDSC_SA    3230    2945 2015-02-12 23:00:00 2015-02-16 23:00:00
#>   CELL_ID MASTER_CELL_ID COSMIC_ID CELL_LINE_NAME SEEDING_DENSITY DRUGSET_ID
#> 1    4712            198    753608          PC-14             250        159
#> 2    4712            198    753608          PC-14             250        159
#>   ASSAY DURATION POSITION     TAG DRUG_ID CONC INTENSITY
```

```
#> 1   Glo        4        1 UN-USED      NA   NA         0
#> 2   Glo        4        2 UN-USED      NA   NA         0
```

- Individual experimental plates are identified by `BARCODE`.
- For each well `POSITION` on a plate there is an `INTENSITY` measurement that indicates the number of viable cells in the well.
- Each row in the raw data presents data for a single treatment of one well in a screening assay plate.
- The treatment is represented by the `TAG` column.
- The experimental design for a plate is gievn by a `DRUGSET_ID`. Drug treatments are most commonly referred to as library drugs within the drugset, e.g., L1, L2 etc.
- In some cases, although not in the example data, there might be more than one treatment for a well and therefore there will be as many rows for that well as there are treatments (or tags).

For more details on the `TAG` and the other columns in the raw data see **GDSC raw data description** - `vignette("gdsc_raw_data_format")`.

Data filters

With the data loaded we first remove any drug treatments that were failed during the internal QC process. These are represented by a tag with the value `FAIL`.

```
gdsc_example <- removeFailedDrugs(gdsc_example)
```

We might also have rows with drug treatments such as a library drug, e.g., `TAG = L12-D1-S`, but there is no associated `DRUG_ID`. This is rare, but usually occurs because the tag was part of the experimental design but in practice was not used for an actual treatment. We should remove these rows from the data as well.

```
gdsc_example <- removeMissingDrugs(gdsc_example)
```

Data normalization

Now the data can be normalized. By normalization we mean converting the raw fluourescence intensities for the treated wells (the read-out from the assay) to a cell viability value between 0 and 1. We assume that the dynamic range for the cell viability is bounded by the mean of the positive controls (μ_{pos}) and the mean of the negative controls (μ_{neg}), equivalent to a viablilty of 0 and 1 respectively. The normalization is always done on a per plate basis.

$$viability = \frac{intensity - \mu_{pos}}{\mu_{neg} - \mu_{pos}}$$

```
normalized_gdsc_example <- normalizeData(gdsc_example,
                                         trim = T,
                                         neg_control = "NC-1",
                                         pos_control = "B")
```

The negative and positive controls are selected using tags that are present in the data. With the trim parameter set to T (the default) we can ensure that all the viablilities are between 0 and 1 - due to experimental variablity some treated wells may have intensity readings greater than μ_{neg} and thus viabilities greater than 1. For the curve fit, it is necessary to have trimmed values (the default).

```
head(normalized_gdsc_example, 2)
#>   SCAN_ID BARCODE RESEARCH_PROJECT        DATE_CREATED DRUGSET_ID
#> 1    2945    3230          GDSC_SA 2015-02-12 23:00:00        159
#> 2    2945    3230          GDSC_SA 2015-02-12 23:00:00        159
```

```
#>   CELL_LINE_NAME CELL_ID MASTER_CELL_ID COSMIC_ID POSITION DRUG_ID_lib CONC
#> 1          PC-14    4712            198    753608      100        1003 0.10
#> 2          PC-14    4712            198    753608      101        1060 0.25
#>   INTENSITY lib_drug dose treatment       NC       PC normalized_intensity
#> 1     23093       L1   D1         S 44497.82 958.3214            0.5083816
#> 2     38426       L2   D1         S 44497.82 958.3214            0.8605446
#>   norm_neg_pos          time_stamp      sw_version
#> 1       NC-1+B 2024-01-18 14:32:39 gdscIC50_0.99.4
#> 2       NC-1+B 2024-01-18 14:32:39 gdscIC50_0.99.4
```

- The resulting data frame will now have a single row for each treated well on each plate in the data.
- Each well has a `normalized_intensity` value - a cell viability normalized to the control wells on the plate.
- Treatment tags for drug treatments (library drugs) are split into `lib_drug` and `dose`.
- The plate average for the chosen negative control (`NC`) and the positive control (`PC`) are shown along with the tags used to choose each control (`norm_neg_pos`).

Drug treatment concentration scale normalization

The nlme model used to fit the GDSC data needs every drug treatment to be compared on a single scale no matter what concentrations were used for a particular drug. This scale is set with the maximum concentration at 9 because the original GDSC data used a 9 point 2 fold dilution for each drug treatment, hence dilution points from 9 down to 1. Newer data has adopted different dilution ranges but the maximum of 9 has been kept and so the same function can be used to normalize the concentration scale.

To normalize the concentration scale use the function `setConcsForNlme()`.

```
scaled_gdsc_example <- setConcsForNlme(normalized_gdsc_example, group_conc_ranges = F)
# Check the scale normalization for a given drug.
unique(subset(scaled_gdsc_example, DRUGSET_ID == 159 & DRUG_ID_lib == 1003,
              select =  c("DRUGSET_ID", "lib_drug", "dose", "CONC", "maxc", "x")))
#> # A tibble: 7 x 6
#>   DRUGSET_ID lib_drug dose      CONC  maxc      x
#>        <dbl> <chr>    <chr>    <dbl> <dbl>  <dbl>
#> 1        159 L1       D1     0.1       0.1  9
#> 2        159 L1       D2     0.0316    0.1  7.34
#> 3        159 L1       D3     0.0100    0.1  5.68
#> 4        159 L1       D4     0.00316   0.1  4.02
#> 5        159 L1       D5     0.00100   0.1  2.36
#> 6        159 L1       D6     0.000316  0.1  0.696
#> 7        159 L1       D7     0.000100  0.1 -0.965
```

Two additional columns are added to the data frame:

- `maxc` is the maximum concentration for that drug treatment in micromolar.
- `x` is the concentration on a scale up to 9, where:

$$x = \frac{log\frac{CONC}{maxc}}{log(2)} + 9$$

Drugs with more than one concentration range The `group_conc_ranges` argument to `setConcsForNlme()` has a default of `FALSE`. It controls the granularity for setting the concentration range for a drug. If a given drug is titrated as two separate library drugs and those library drugs are

screened at different concentration ranges, then this will result in different `maxc` for the two libraries and the x values will correspond to different micromolar concentrations. In the data `DRUG_ID` 1510 has been screened at two different maximum concentrations represented as library drugs L9 and L32 in the drugsets 158 and 217.

```
unique(subset(normalized_gdsc_example, DRUG_ID_lib == 1510 & dose == "D1",
              select =  c("DRUGSET_ID", "lib_drug", "dose", "CONC")))
#>       DRUGSET_ID lib_drug dose CONC
#> 115          158       L9   D1  2.5
#> 117          158      L32   D1 10.0
#> 21115        217       L9   D1  2.5
#> 21117        217      L32   D1 10.0
```

By setting `group_conc_ranges = TRUE` a single concentration range will be used for the drug in question, spanning the ranges used for the separate libraries.

```
set_concs_test <- setConcsForNlme(normalized_gdsc_example, group_conc_ranges = T)

unique(subset(set_concs_test, DRUG_ID_lib == 1510,
              select =  c("lib_drug", "dose", "CONC", "maxc", "x")))
#> # A tibble: 14 x 5
#>    lib_drug dose      CONC  maxc      x
#>    <chr>    <chr>    <dbl> <dbl>  <dbl>
#>  1 L9       D1     2.5        10  7
#>  2 L32      D1    10          10  9
#>  3 L9       D2     0.791      10  5.34
#>  4 L32      D2     3.16       10  7.34
#>  5 L9       D3     0.250      10  3.68
#>  6 L32      D3     1.00       10  5.68
#>  7 L9       D4     0.0791     10  2.02
#>  8 L32      D4     0.316      10  4.02
#>  9 L9       D5     0.0250     10  0.357
#> 10 L32      D5     0.100      10  2.36
#> 11 L9       D6     0.00791    10 -1.30
#> 12 L32      D6     0.0316     10  0.696
#> 13 L9       D7     0.00250    10 -2.97
#> 14 L32      D7     0.0100     10 -0.965
rm(set_concs_test)
```

By setting `group_conc_ranges = FALSE` the different dilution series are kept separate.

```
set_concs_test <- setConcsForNlme(normalized_gdsc_example, group_conc_ranges = F)
unique(subset(set_concs_test, DRUG_ID_lib == 1510, select =  c("lib_drug", "dose", "CONC", "maxc", "x")))
#> # A tibble: 14 x 5
#>    lib_drug dose      CONC  maxc      x
#>    <chr>    <chr>    <dbl> <dbl>  <dbl>
#>  1 L9       D1     2.5         2.5  9
#>  2 L32      D1    10          10    9
#>  3 L9       D2     0.791       2.5  7.34
#>  4 L32      D2     3.16       10    7.34
#>  5 L9       D3     0.250       2.5  5.68
#>  6 L32      D3     1.00       10    5.68
#>  7 L9       D4     0.0791      2.5  4.02
#>  8 L32      D4     0.316      10    4.02
#>  9 L9       D5     0.0250      2.5  2.36
#> 10 L32      D5     0.100      10    2.36
#> 11 L9       D6     0.00791    2.5  0.696
```

```
#> 12 L32      D6     0.0316   10    0.696
#> 13 L9       D7     0.00250   2.5 -0.965
#> 14 L32      D7     0.0100   10   -0.965
```

Choosing `CL` and `drug` for the nlme model

Finally the data can be transformed into the format needed for input to the nlme fitting. The fitting will calculate a model for the dose response of a cell line to a specified drug treatment.

- The model uses a cell line input (`CL`) and drug input (`drug`) to uniquely identify each experiment.
- Here we use the `COSMIC_ID` column from the input data to be the `CL` identifier. If there is no `COSMIC_ID` for the cell line we might chose a different identifier, e.g., `CELL_ID` or `CELL_LINE_NAME`.
- The `drug_specifiers` argument controls the granularity for setting `drug` level in the dose response. The default values are `DRUG_ID_lib` plus `maxc`, which will fit for every drug with a separate model if there is a difference in maximum concentration.

We might try to fit a model for each `DRUG_ID` in the data set. Replicate data from different plates across the dataset will be included in the same model. Bearing in mind we have run `setConcsForNlme(..., group_conc_ranges = F)`, if we run `prepNlmeData()` a warning is generated.

```
nlme_data <- prepNlmeData(scaled_gdsc_example,
                          cl_id = "COSMIC_ID",
                          drug_specifiers = "DRUG_ID_lib")
#> Warning: `arrange_()` was deprecated in dplyr 0.7.0.
#> i Please use `arrange()` instead.
#> i See vignette('programming') for more help
#> i The deprecated feature was likely used in the gdscIC50 package.
#>   Please report the issue to the authors.
#> This warning is displayed once every 8 hours.
#> Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
#> Warning: `unite_()` was deprecated in tidyr 1.2.0.
#> i Please use `unite()` instead.
#> i The deprecated feature was likely used in the gdscIC50 package.
#>   Please report the issue to the authors.
#> This warning is displayed once every 8 hours.
#> Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
#> Warning: `count_()` was deprecated in dplyr 0.7.0.
#> i Please use `count()` instead.
#> i See vignette('programming') for more help
#> i The deprecated feature was likely used in the gdscIC50 package.
#>   Please report the issue to the authors.
#> This warning is displayed once every 8 hours.
#> Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
#> Warning: `distinct_()` was deprecated in dplyr 0.7.0.
#> i Please use `distinct()` instead.
#> i See vignette('programming') for more help
#> i The deprecated feature was likely used in the gdscIC50 package.
#>   Please report the issue to the authors.
#> This warning is displayed once every 8 hours.
#> Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
#> Warning in prepNlmeData(scaled_gdsc_example, cl_id = "COSMIC_ID",
#> drug_specifiers = "DRUG_ID_lib"): There is more than one maximum concentration
#> for drug 1510
```

It is possible to continue with this model specification, but there will be a single model with scale and xmid parameters for drug 1510 but when IC50s are calculated there will be a different IC50 value depending on the `maxc`.

There are two options:

- Rerun `setConcsForNlme()` but this time with `group_conc_ranges = T` again before running `prepNlmeData(..., drug_specifiers = "DRUG_ID_lib")`.
- Or use the default `drug_specifiers` setting of the `DRUG_ID_lib` and `maxc` columns from the input data. This will result in a separate model with different parameters fitted for `DRUG_ID` 1510 at the two different maximum concentration values.

Choosing the latter option:

```
nlme_data <- prepNlmeData(scaled_gdsc_example,
                          cl_id = "COSMIC_ID",
                          drug_specifiers = c("DRUG_ID_lib", "maxc"))
head(nlme_data, 3)
#> # A tibble: 3 x 17
#>   CELL_LINE_NAME     CL drug      maxc      x     y DRUG_ID_lib BARCODE SCAN_ID
#>   <chr>           <dbl> <chr>    <dbl>  <dbl> <dbl>       <dbl> <chr>     <dbl>
#> 1 MC-CAR         683665 1003_0.1   0.1 -0.965 0            1003 8860       8043
#> 2 MC-CAR         683665 1003_0.1   0.1 -0.965 0            1003 9068       8227
#> 3 MC-CAR         683665 1003_0.1   0.1  0.696 0.130        1003 8860       8043
#> # i 8 more variables: POSITION <int>, DRUGSET_ID <dbl>, norm_neg_pos <chr>,
#> #   CL_SPEC <chr>, drug_spec <chr>, time_stamp <dttm>, sw_version <chr>,
#> #   RESEARCH_PROJECT <chr>
```

Many of the columns in the `nlme_data` are carried over from the normalized data frame. The `CL_SPEC` and the `drug_spec` columns record the `cl_id` and `drug_specifiers` arguments to the function call `prepNlmeData()`. For the nlme fitting the essential columns are `drug`, `CL`, `x`, `y` and `maxc`. `y` is just the `normalized_intensity` as prepared above. Note how the `drug` column is a concatenation of the chosen `drug_spec`.

Table 1: Examples of `drug_specifier` for `prep_nlme_data`

drug_specifiers =	Fits a model for:
"DRUG_ID_lib"	Each DRUG_ID. Combines data from any replicates in a drug set and from any replicate plates.
c("DRUG_ID_lib", "maxc")	Each DRUG_ID but separates out instances where the dose titration starts from different maximum concentrations.
c("BARCODE", "DRUG_ID_lib")	Each DRUG_ID on each plate. Combines replicates within the drug set as applied to an individual plate but not between plates.
c("DRUGSET_ID", "lib_drug")	Each library drug in each drug set. Different library drugs could be replicates of the same DRUG_ID. Replicates are combined across plates.
c("BARCODE", "DRUGSET_ID", "lib_drug")	Each library drug in each drug set on every plate. Replicates only within a plate (BARCODE).
c("DRUG_ID_lib", "maxc", "DATE_CREATED")	Each DRUG_ID at different concentration ranges and on different plate dates.

Fitting the dose response model

gdscIC50 provides a wrapper for fitting the model using djvMixedIC50. For this demonstration, the parameter `isLargeData` is set to false to minimize the fitting time and ensure the model converges. For a larger data set this parameter should be set to `TRUE` such that the covariance between the position and scale parameter on the cell line level are assumed to be correlated. This further stabilizes the fit. In small bespoke screens this is set to false as the model otherwise struggles to converge.

```
nlme_model <- fitModelNlmeData(nlme_data, isLargeData = F)
```

The results

Next a data frame (actually a tibble) of the results is calculated from the fitted model.

```
nlme_stats <- calcNlmeStats(nlme_model, nlme_data)
head(data.frame(nlme_stats), 2)
#>       xmid     scal     CL     drug CELL_LINE_NAME maxc          x y
#> 1 13.51234 4.005647 683665 1003_0.1         MC-CAR  0.1 -0.9650242 0
#> 2 13.51234 4.005647 683665 1003_0.1         MC-CAR  0.1 -0.9650242 0
#>   DRUG_ID_lib BARCODE SCAN_ID POSITION DRUGSET_ID norm_neg_pos   CL_SPEC
#> 1        1003    8860    8043      388        159       NC-1+B COSMIC_ID
#> 2        1003    9068    8227      388        217       NC-1+B COSMIC_ID
#>          drug_spec          time_stamp      sw_version RESEARCH_PROJECT
#> 1 DRUG_ID_lib+maxc 2024-01-18 14:32:39 gdscIC50_0.99.4          GDSC_SA
#> 2 DRUG_ID_lib+maxc 2024-01-18 14:32:39 gdscIC50_0.99.4          GDSC_SA
#>         yhat        yres      RMSE   x_micromol      IC50       auc   AUCtrap
#> 1 0.02623084 -0.02623084 0.1497467 0.0001000527 0.8251303 0.8978178 0.8968996
#> 2 0.02623084 -0.02623084 0.1497467 0.0001000527 0.8251303 0.8978178 0.8968996
```

- The resulting data frame has a row for each data point `(x, y)` used in each model.
- For each data point there is the fitted value `yhat` and the residual `yres`.
- For each x value the corresponding value in μM is given.
- For the overall model the model parameters are `xmid` and `scal`.
- The `RMSE` - root mean square error - is calculated for each model fit.
- The `IC50` is calculated as the natural log of the μM value.
- Two calculated values are shown for the are under the curve: the integral value from the logistic curve `AUC`; and a numerical calculation using the trapezoid rule `AUCtrap` - the former is preferred over the latter.
- Both `AUC` and `AUCtrap` are calculated as a fraction of the area bounded by the maximum and minimum concentrations screened on the x axis and by the maximum and minimum response on the y axis. They are therefore always between 0 and 1, with values closer to 0 indicating a sensitive response.

Plotting a dose response

Individual dose response plots (ggplot objects) can be created from the fitted values data frame. First an example from the data of a sensitive response:

```
plotResponse(model_stats = nlme_stats, cell_line = 1503364, drug_identifier = "1032_2")
```

Secondly, an insensitive response:

```
plotResponse(model_stats = nlme_stats, cell_line = 687829, drug_identifier = "1003_0.1")
```

Preparing the IC50 matrix for analysis with GDSCtools

The Python package GDSCtools Cokelaer, T et al. 2017 is used to analyse the relationship between drug sensitivity and the genomics of the cell line model systems. GDSCtools provides several statistical methods for biomarker discovery. In particular it is used for the GDSC ANOVA analysis, the results of which are presented on http://www.cancerrxgene.org. The drug sensitivity data needs to be presented as a matrix for use with GDSCtools and saved as a csv.

```
IC50_matrix <- getIC50Matrix(nlme_stats)
AUC_matrix <- getIC50Matrix(nlme_stats, measure = "auc")
```

```
write.csv(IC50_matrix, "my_gdsctools_dir/IC50_matrix.csv")
```
