# Package: RMeDPower2 (via r-universe)

June 6, 2026

**Type** Package

**Title** Design and Modeling for Repeated Measures Studies

**Version** 1.0.2

**Description** Provides complete functionality to analyse data from repeated measures experiments with hierarchical or crossed experimental designs. Supports testing modeling assumptions, identifying outlier observations and experimental units, estimating statistical power, and performing sample size calculations. Uses linear mixed effects models via 'lme4' and simulation-based power analysis via 'simr'. Handles both normal and non-normal error distributions including binomial and Poisson families. For more details see Shin et al. (2022) <doi:10.1101/2022.07.18.500490>, Bates et al. (2015) <doi:10.18637/jss.v067.i01>, Green and MacLeod (2016) <doi:10.1111/2041-210X.12504>, Hartig (2024) <doi:10.32614/CRAN.package.DHARMa>, Nieuwenhuis et al. (2012) <doi:10.32614/RJ-2012-011>, Millard (2013) <doi:10.1007/978-1-4614-8456-1> and Kuznetsova et al. (2017) <doi:10.18637/jss.v082.i13>.

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.3.2

**VignetteBuilder** knitr

**Depends** R (>= 4.0)

**Imports** lme4, dplyr, simr, magrittr, ggplot2, ggtext, quantreg, tibble, lmerTest, DHARMa, influence.ME, EnvStats, jsonlite, methods, stats, grDevices, graphics

**Suggests** knitr, rmarkdown, testthat (>= 3.0.0)

**NeedsCompilation** no

**Author** Min-Gyoung Shin [aut], Reuben Thomas [aut, cre]

**Maintainer** Reuben Thomas <reuben.thomas@gladstone.ucsf.edu>

**Config/pak/sysreqs**
cmake make libicu-dev libjpeg-dev libpng-dev libuv1-dev libxml2-dev libssl-dev zlib1g-dev

**Repository** https://reubenthomas.r-universe.dev

**Date/Publication** 2026-06-06 16:50:02 UTC

**RemoteUrl** https://github.com/cran/RMeDPower2

**RemoteRef** HEAD

**RemoteSha** f2b8bcc50660c71a44444c5bd0fa67b992ccb9a1

# Contents



---

`calculatePower` *calculatePower*

---

## Description

This functions makes statistical power estimates given the data, the underlying design for it and the assumed probability model of the error distribution

## Usage

```
calculatePower(data, design, model, power_param)
```

### Arguments

| | |
|---|---|
| data | Input data frame with columns having all the necessary information regarding the dependent and independent variables of interest |
| design | an object of class RMeDesign with the necessary design information about the data |
| model | an object of class ProbabilityModel giving the error distribution of the data |
| power_param | an object of class PowerParams giving the target parameter of interest and the other necessary parameter to perform the power estimation |

### Value

A power curve as a ggplot object or a power calculation result printed in a text file

### Examples

```
template_dir <- system.file("input_templates/cell_assay_data", package = "RMeDPower2")
data <- plate_assay_pilot_data
design <- readDesign(file.path(template_dir,"design_cell_assay.json"))
model <- readProbabilityModel(file.path(template_dir,"prob_model.json"))
power_param <- readPowerParams(file.path(template_dir,"power_param.json"))
power_res <- calculatePower(data, design, model, power_param)
```

---

## diagnoseDataModel *diagnoseDataModel*

---

### Description

This function can be used to generate diagnostic QC plots for given model assumptions related to the input data, identify potential outlier observations and/or outlier experimental units

### Usage

```
diagnoseDataModel(data, design, model)
```

### Arguments

| | |
|---|---|
| data | Input data frame with columns having all the necessary information regarding the dependent and independent variables of interest |
| design | an object of class RMeDesign with the necessary design information about the data |
| model | an object of class ProbabilityModel giving the error distribution of the data |

## Value

A list with four elements. 1) models: representing the names of the models evaluated based on differnt modifications of the response column. The models would include one called natural_scale, another model called natural_scale_wo_outliers if outliers had beeen identified, another model called log_scale if the respose column is continuous and the model on the log-transformed values of the responses are what was evaluated and finally log_scale_wo_outliers model if there were outliers identified in the log_scale model. 2) Data_updated representing the updated data frame with additional columns for the modified response column corresponding to each of the models evaluated. 3) cooks_result: cooks distance of each of the experimental columns for each of the models evaluated. For models based on the binomial probability distribution, cooks distance is only reported for the first experimental column on account the increased computation time for evaluating this metric for the other experimental columns. 4) plots_info: is a list with two elements plots and captions. plots is a named list and captions is a character vector, both of the same length as the number of models evaluated. Each element of the plots list is yet another list of QC/diagnostic plots related to the corresponding model fit, while the captions is a vector of captions for each of the QC plots output

## Examples

```
template_dir <- system.file("input_templates/cell_assay_data", package = "RMeDPower2")
data <- plate_assay_pilot_data
design <- readDesign(file.path(template_dir,"design_cell_assay.json"))
model <- readProbabilityModel(file.path(template_dir,"prob_model.json"))
diagnose_res <- diagnoseDataModel(data, design, model)
```

---

`getEstimatesOfInterest` *getEstimatesOfInterest*

---

## Description

This function performs the estimations of interest and also visualizes the resulting association

## Usage

```
getEstimatesOfInterest(data, design, model, print_plots = TRUE)
```

## Arguments

| | |
|---|---|
| `data` | Input data frame with columns having all the necessary information regarding the dependent and independent variables of interest |
| `design` | an object of class RMeDesign with the necessary design information about the data |
| `model` | an object of class ProbabilityModel giving the error distribution of the data |
| `print_plots` | Whether or not to print the plots, irrespective of this argument ggplot versions of evaluated association between the response_column and the condition_column. TRUE - print the plot, FALSE - do not print the plot |

## Value

a list with three elements - 1. an object of class summary.merMod and 2. the output from the get_residuals functions. This output consists of a list with 3 elements. 1. The updated input data with an additional column with the model residuals of the individual observations. 2. A plot representing the purported association between the response column and the condition column. 3. The corresponding caption for this figure. 3. an object of class merMod

## Examples

```
template_dir <- system.file("input_templates/cell_assay_data", package = "RMeDPower2")
data <- plate_assay_pilot_data
design <- readDesign(file.path(template_dir,"design_cell_assay.json"))
model <- readProbabilityModel(file.path(template_dir,"prob_model.json"))
res <- getEstimatesOfInterest(data, design, model)
```

---

# mouse_behavior_MWM_assay_data

*Mouse behavior data from a Morris Water Maze assay*

---

## Description

Example behavioral dataset containing measurements from a mouse Morris Water Maze (MWM) assay. The data represent repeated measures across trials and subjects and are suitable for illustrating repeated measures power analysis.

## Usage

```
data(mouse_behavior_MWM_assay_data)
```

## Format

A data frame of behavioral measurements with information on mouse, trial, and experimental condition.

---

# mouse_brain_electro_physiology_data

*Mouse brain electrophysiology data*

---

## Description

Example dataset containing electrophysiological measurements from mouse brain recordings. The data are used to demonstrate power analyses in experiments with repeated measurements and complex correlation structures.

### Usage

```
data(mouse_brain_electro_physiology_data)
```

### Format

A data frame of electrophysiological measurements and associated experimental annotations.

---

## plate_assay_full_data *Full plate assay dataset*

---

### Description

Full plate assay dataset corresponding to the pilot data but including the complete experimental run. This dataset is used in examples demonstrating power calculations under more realistic sample sizes and hierarchies.

### Usage

```
data(plate_assay_full_data)
```

### Format

A data frame containing the full plate assay data. Column definitions follow those of `plate_assay_pilot_data`.

### See Also

`plate_assay_pilot_data`, `plate_assay_pilot_data_wo_repeats`

---

## plate_assay_pilot_data *Pilot plate assay data*

---

### Description

Pilot dataset from plate-based assays used in the RMeDPower2 documentation and examples. The data represent repeated measurements across plates and experimental units and are intended for illustrating experimental design specification, model diagnostics, and power calculations.

### Usage

```
data(plate_assay_pilot_data)
```

### Format

A data frame with observations from a pilot plate assay. Column names and structure are documented in the package vignette and example code.

## See Also

plate_assay_pilot_data_wo_repeats, plate_assay_full_data

---

`plate_assay_pilot_data_wo_repeats` *Pilot plate assay data without repeated measurements*

---

## Description

Version of plate_assay_pilot_data where repeated measurements have been removed, suitable for power analyses that assume a single observation per experimental unit at each time point or condition.

## Usage

```
data(plate_assay_pilot_data_wo_repeats)
```

## Format

A data frame containing the pilot plate assay data without repeated measurements. See the vignette for details on columns and preprocessing.

## See Also

plate_assay_pilot_data, plate_assay_full_data

---

`PowerParams-class` *PowerParams-class*

---

## Description

Objects of PowerParams class store information required for sample size estimation for given data

## Arguments

| | |
|---|---|
| power_curve | 1: Power simulation over a range of sample sizes or levels. 0: Power calculation over a single sample size or a level. |
| nsimn | The number of simulations to run. Default=1000 |
| target_columns | Name of the experimental parameters to use for the power calculation. |
| levels | 1: Amplify the number of corresponding target parameter. 0: Amplify the number of samples from the corresponding target parameter, ex) If target_columns = c("experiment","cell_line") and if you want to expand the number of experiment and sample more cells from each cell line, set levels = c(1,0). |

| | |
|---|---|
| max_size | Maximum levels or sample sizes to test. Default: the current level or the current sample size x 5. ex) If max_levels = c(10,5), it will test upto 10 experiments and 5 cell lines. |
| breaks | Levels /sample sizes of the variable to be specified along the power curve. Default: max(1, round( the number of current levels / 5 )) |
| effect_size | If you know the effect size of your condition variable, the effect size can be provided as a parameter. If the effect size is not provided, it will be estimated from your data |
| alpha | Threshold for Type I error |
| ICC | Intra-Class Coefficients (ICC) for each parameter |

## Value

an object of class ProbabilityModel

## Examples

```
power_param=new("PowerParams")
```

---

# ProbabilityModel-class

*ProbabilityModel-class*

---

## Description

Objects of ProbabilityModel class store information on the assumed probability distribution for the model

## Arguments

| | |
|---|---|
| error_is_non_normal | Default: the observed variable is continuous Categorical response variable will be implemented in the future. TRUE: Categorical , FALSE: Continuous (default). |
| family_p | The type of distribution family to specify when the response is categorical. If family is "binary" then binary(link="log") is used, if family is "poisson" then poisson(link="logit") is used, if family is "poisson_log" then poisson(link=") log") is used. |

## Value

an object of class ProbabilityModel

## Examples

```
model=new("ProbabilityModel")
```

## readDesign *readDesign*

### Description

This functions reads the underlying design for the data

### Usage

```
readDesign(jsonfile)
```

### Arguments

| | |
|---|---|
| jsonfile | the jsonfile with the necessary design parameters: condition_column, experimental_columns, response_column, total_column, condition_is_categorical, covariate, method, crossed_columns, error_is_non_normal, family_p, outlier_alpha, na.action |

### Value

an object of class RMeDesign

### Examples

```
template_dir <- system.file("input_templates/cell_assay_data", package = "RMeDPower2")
design <- readDesign(file.path(template_dir,"design_cell_assay.json"))
```

## readPowerParams *readPowerParams*

### Description

This functions reads the underlying design for the data

### Usage

```
readPowerParams(jsonfile)
```

### Arguments

| | |
|---|---|
| jsonfile | the jsonfile with the necessary parameters for statistical power estimation: target_columns, power_curve, nsimn, levels, max_size, alpha, breaks, effect_size, icc |

### Value

an object of class PowerParams

## Examples

```
template_dir <- system.file("input_templates/cell_assay_data", package = "RMeDPower2")
power_param <- readPowerParams(file.path(template_dir,"power_param.json"))
```

---

`readProbabilityModel` *readProbabilityModel*

---

## Description

This functions reads the underlying design for the data

## Usage

```
readProbabilityModel(jsonfile)
```

## Arguments

| | |
|---|---|
| `jsonfile` | the jsonfile with the necessary parameters for probability model: error_is_non_normal, family_p |

## Value

an object of class ProbabilityModel

## Examples

```
template_dir <- system.file("input_templates/cell_assay_data", package = "RMeDPower2")
model <- readProbabilityModel(file.path(template_dir,"prob_model.json"))
```

---

`RMeDesign-class` *RMeDesign-class*

---

## Description

Objects of RMeDesign class store information on the relevant repeated measures design for the given data

## Arguments

| | |
|---|---|
| `data` | Input data |
| `condition_column` | Name of the condition variable (ex variable with values such as control/case). The input file has to have a corresponding column name |
| `experimental_columns` | Name of the variable related to experimental design such as "experiment", "plate", and "cell_line". They should be in order, for example, "experiment" should always come first . |
| `response_column` | Name of the variable observed by performing the experiment. ex) intensity. |
| `total_column` | Set this column only when family_p="binomial" and it is equal to the total number of observations (number of cases plus number of controls) for a given number of cases |
| `outlier_alpha` | numeric scalar between 0 and 1 indicating the Type I error associated with the test of outliers |
| `condition_is_categorical` | Specify whether the condition variable is categorical. TRUE: Categorical, FALSE: Continuous. |
| `covariate` | The name of the covariate to control in the regression model |
| `method` | The method used to detect outliers. "rosner" (default) runs Rosner's test and "cook" runs Cook's distance. |
| `crossed_columns` | Name of experimental variables that may appear repeatedly with the same ID. For example, cell_line C1 may appear in multiple experiments, but plate P1 cannot appear in more than one experiment |
| `error_is_non_normal` | Default: the observed variable is continuous Categorical response variable will be implemented in the future. TRUE: Categorical , FALSE: Continuous (default). |
| `family_p` | The type of distribution family to specify when the response is categorical. If family is "binary" then binary(link="log") is used, if family is "poisson" then poisson(link="logit") is used, if family is "poisson_log" then poisson(link=") log") is used. |
| `na.action` | "complete": missing data is not allowed in all columns (default), "unique": missing data is not allowed only in condition, experimental, and response columns. Selecting "complete" removes an entire row when there is one or more missing values, which may affect the distribution of other features. |
| `include_interaction` | logical - TRUE or FALSE - Whether to include condition * covariate interaction |
| `random_slope_variable` | Variable for random slopes (typically one of "condition_column" or "covariate" and assuming that they are numeric variables). A random slope term is added for each of the variables specified in the experimental columns in addition to their corresponding random intercept terms. The random slope and intercept terms for each experimental_columns variable are assumed to be uncorrelated. |

covariate_is_categorical
: Specify whether the covariate variable is categorical. TRUE: Categorical, FALSE: Continuous.

## Value

an object of class RMeDesign

## Examples

```
design=new("RMeDesign")
```

---

# snRNAseq_cluster_count_data

*Single-nucleus RNA-seq cluster-level count data*

---

## Description

Example dataset containing cluster-level count summaries from a single-nucleus RNA-seq experiment. The data are intended to illustrate how RMeDPower2 can be applied to hierarchical omics experiments with counts aggregated at the cluster level.

## Usage

```
data(snRNAseq_cluster_count_data)
```

## Format

A data frame of cluster-level counts and associated annotations.

## See Also

snRNAseq_gene_count_data

---

# snRNAseq_gene_count_data

*Single-nucleus RNA-seq gene-level count data*

---

## Description

Example dataset containing gene-level count summaries from a single-nucleus RNA-seq experiment. This dataset can be used to demonstrate power calculations for differential expression-type analyses across experimental conditions.

## Usage

```
data(snRNAseq_gene_count_data)
```

## Format

A data frame of gene-level counts and associated annotations.

## See Also

snRNAseq_cluster_count_data

# Index