# Package 'DNABarcodeCompatibility'

April 5, 2026

**Type** Package

**Title** A Tool for Optimizing Combinations of DNA Barcodes Used in Multiplexed Experiments on Next Generation Sequencing Platforms

**Version** 1.27.0

**Maintainer** Céline Trébeau <ctrebeau@pasteur.fr>

**Description** The package allows one to obtain optimised combinations of DNA barcodes to be used for multiplex sequencing. In each barcode combination, barcodes are pooled with respect to Illumina chemistry constraints. Combinations can be filtered to keep those that are robust against substitution and insertion/deletion errors thereby facilitating the demultiplexing step. In addition, the package provides an optimiser function to further favor the selection of barcode combinations with least heterogeneity in barcode usage.

**License** file LICENSE

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 6.1.1

**Imports** dplyr, tidyr, numbers, purrr, stringr, stats, utils, methods, Rcpp (>= 0.11.2), BH

**LinkingTo** Rcpp, BH

**Depends** R (>= 3.6.0)

**Suggests** knitr, rmarkdown, BiocStyle, testthat

**VignetteBuilder** knitr

**biocViews** Preprocessing, Sequencing

**URL** https://dnabarcodecompatibility.pasteur.fr/

**BugReports** https://gitlab.pasteur.fr/ida-public/dnabarcodecompatibility/-/issues

**NeedsCompilation** yes

**git_url** https://git.bioconductor.org/packages/DNABarcodeCompatibility

**git_branch** devel

**git_last_commit** 0fc748e

**git_last_commit_date** 2025-10-29

**Repository** Bioconductor 3.23

**Date/Publication** 2026-04-05

**Author** Céline Trébeau [cre] (ORCID: <https://orcid.org/0000-0001-6795-5379>),
Jacques Boutet de Monvel [aut] (ORCID:
<https://orcid.org/0000-0001-6182-3527>),
Fabienne Wong Jun Tai [ctb],
Raphaël Etournay [aut] (ORCID: <https://orcid.org/0000-0002-2441-9274>)

# Contents



---

`DNABarcodeCompatibility-package`

*DNABarcodeCompatibility: to find optimised sets of compatible barcodes with least heterogeneity in barcode usage for multiplex experiments performed on next generation sequencing platforms.*

---

## Description

The DNABarcodeCompatibility package provides six functions to load DNA barcodes, and to generate, filter and optimise sets of barcode combinations for multiplex sequencing experiments. In particular, barcode combinations are selected to be compatible with respect to Illumina chemistry constraints, and can be filtered to keep those that are robust against substitution and insertion/deletion errors thereby facilitating the demultiplexing step. In addition, the package provides an optimiser function to further favor the selection of compatible barcode combinations with least heterogeneity in barcode usage.

distance *Calculate distance between two barcodes.*

## Description

The function calculates the distance between two barcodes. The user may choose one of several distance metrics ("hamming", "seqlev", "levenshtein", "phaseshift").

## Usage

```
distance(sequence1, sequence2, metric=c("hamming","seqlev","levenshtein", "phaseshift"), cost_sub=1,
```

## Arguments

| | |
|---|---|
| sequence1 | The first sequence (a string) |
| sequence2 | The second sequence (a string) |
| metric | The distance metric which should be calculated. |
| cost_sub | The cost weight given to a substitution. |
| cost_indel | The cost weight given to insertions and deletions. |

## Value

The distance between the two sequences.

## References

Buschmann, T. and Bystrykh, L. V. (2013) Levenshtein error-correcting barcodes for multiplexed DNA sequencing. BMC bioinformatics, 14(1), 272. Available from http://www.biomedcentral.com/1471-2105/14/272.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions and reversals. In Soviet physics doklady (Vol. 10, p. 707).

Hamming, R. W. (1950). Error detecting and error correcting codes. Bell System technical journal, 29(2), 147-160.

## Examples

```
distance("AGGT", "TTCC")
distance("AGGT", "TTCC", metric="seqlev")
```

`distance_filter` *Select barcode combinations with error correction properties*

## Description

Filters a list of barcode combinations for a given distance metric (hamming or seqlev) and threshold in order to produce a list of barcodes satisfying the distance constraints.

## Usage

```
distance_filter(index_df, combinations_m, metric, d)
```

## Arguments

| | |
|---|---|
| `index_df` | A dataframe containing barcodes identifiers, corresponding DNA sequences along with GC content and presence of homopolymers. |
| `combinations_m` | A matrix of compatible barcode combinations. |
| `metric` | The type of distance (hamming or seqlev or phaseshift). |
| `d` | The minimum value of the distance. |

## Details

The "hamming" distance is suitable for correcting substitution errors. The "seqlev" distance is suitable for correcting both substitution and insertion/deletion errors.

## Value

A filtered matrix containing the identifiers of the barcodes satisfying the distance constraints.

## References

Buschmann, T. 2015. The Systematic Design and Application of Robust DNA Barcodes.

Buschmann, T. 2017. DNABarcodes: an R package for the systematic construction of DNA sample tags. Bioinformatics 33, 920–922.

## See Also

`get_all_combinations`, `get_random_combinations`

## Examples

```
barcodes <- DNABarcodeCompatibility::IlluminaIndexes
m <- get_all_combinations(barcodes, 2, 4)
distance_filter(barcodes, m, "hamming", 3)
```

`experiment_design` *Find a set of barcode combinations with least heterogeneity in barcode usage for single and dual indexing*

## Description

This function uses the Shannon Entropy to identify a set of compatible barcode combinations with least heterogeneity in barcode usage, in the context of single and dual indexing. It performs either an exhaustive or a random-greedy search of compatible DNA-barcode combinations depending on the size of the DNA-barcode population and of the number of samples to be multiplexed.

## Usage

```
experiment_design(file1, sample_number, mplex_level, platform = 4,
file2 = NULL, export = NULL, metric = NULL, d = 3, thrs_size_comb,
max_iteration, method)
```

## Arguments

| | |
|---|---|
| `file1` | The input data file that contains 2 columns separated by a space or a tabulation, namely the sequence identifiers and corresponding DNA sequence |
| `sample_number` | Number of libraries to be sequenced. |
| `mplex_level` | The number at which the barcodes will be multiplexed. |
| `platform` | An integer representing the number of channels (1, 2, 4) of the desired Illumina platform: 1 for iSeq; 2 for NextSeq, NovaSeq, MiniSeq; 4 for HiSeq and MiSeq. 0 represents any other platform than Illumina. |
| `file2` | The input data file that contains 2 columns separated by a space or a tabulation, namely the sequence identifiers and corresponding DNA sequence; used for dual-indexing, see details below. |
| `export` | If not NULL, results are saved in a csv file at the specified path. |
| `metric` | The type of distance (hamming or seqlev). |
| `d` | The minimum value of the distance. |
| `thrs_size_comb` | The maximum size of the set of compatible combinations to be used for the greedy optimization. |
| `max_iteration` | The maximum number of iterations during the optimizing step. |
| `method` | The choice of the greedy search: 'greedy_exchange' or 'greedy_descent'. |

## Details

By specifying the total number of libraries and the number of libraries to be multiplexed, this function returns an optimal set of DNA-barcode combinations to be used for sequencing.

In the case of **single indexing**, one must only provide one input file containing a list of DNA barcodes (file1 argument). The file2 argument being optional, the program runs the optimisation for

single indexing if this argument is left empty. The output shows the sample ID with its respective single barcode.

In the case of **dual indexing**, one must provide two input files containing DNA barcodes as two separate sets of barcodes. The program will detect these two files and automatically switch to the 'dual indexing' mode. The program then runs the optimisation for each barcode set separately. The output shows the sample ID with its respective pair of barcodes.

The inputs of the algorithm are a list of n distinct barcodes, the number N of required libraries, and the multiplex level k; N = ak, where a is the number of lanes of the flow cells to be used for the experiment.

- Step 1:

This step consists of identifying a set of compatible barcode combinations. Given the number of barcodes and the multiplex level, the total number of barcode combinations (compatible or not) reads:

$$\binom{n}{k}$$

If this number is not too large, the algorithm will perform an exhaustive search and output all compatible combinations of k barcodes. Otherwise, it will proceed by picking up combinations at random, in order to identify a large enough set of compatible barcode combinations.

- Step 2:

Finds an optimized set of barcode combinations in which barcode redundancy is minimized (see details in optimize_combinations)

## Value

A dataframe containing compatible DNA-barcode combinations organized by lanes of the flow cell.

## Examples

```
write.table(DNABarcodeCompatibility::IlluminaIndexesRaw,
txtfile <- tempfile(), row.names = FALSE, col.names = FALSE, quote=FALSE)
experiment_design(file1=txtfile, sample_number=18, mplex_level=3,
platform=4)
```

---

# file_loading_and_checking

*Loading and checking DNA barcodes.*

---

## Description

Loads the file containing DNA barcodes and analyze barcode content.

### Usage

```
file_loading_and_checking(file)
```

### Arguments

| | |
|---|---|
| file | The input data file that contains 2 columns separated by a space or a tabulation, namely the sequence identifiers and corresponding DNA sequence. |

### Details

This function loads the DNA barcodes from the input file and checks barcodes for unicity (identifier and sequence), DNA content, and equal size. It also calculates the fraction of G and C relative to A and T, as referred to as "GC content", and it detects the presence of homopolymers of length >= 3.

### Value

A dataframe containing sequence identifiers, nucleotide sequence, GC content, presence of homopolymers.

### Examples

```
write.table(DNABarcodeCompatibility::IlluminaIndexesRaw,
txtfile <- tempfile(), row.names = FALSE, col.names = FALSE, quote=FALSE)
file_loading_and_checking(txtfile)
```

---

## get_all_combinations *Get all compatible combinations.*

---

### Description

Finds the exhaustive set of compatible barcode combinations.

### Usage

```
get_all_combinations(index_df, mplex_level, platform)
```

### Arguments

| | |
|---|---|
| index_df | A dataframe containing barcodes identifiers, corresponding DNA sequences along with GC content and presence of homopolymers. |
| mplex_level | The number at which the barcodes will be multiplexed. Illumina recommends to not multiplex more than 96 libraries. |
| platform | An integer representing the number of channels (1, 2, 4) of the desired Illumina platform: 1 for iSeq; 2 for NextSeq, NovaSeq, MiniSeq; 4 for HiSeq and MiSeq. 0 represents any other platform than Illumina. |

## Details

Be aware that the total number of combinations may become prohibitively large for large barcode sets and large multiplexing numbers.

## Value

A matrix containing the identifiers of compatible barcode combinations.

## See Also

`get_random_combinations`

## Examples

```
get_all_combinations(DNABarcodeCompatibility::IlluminaIndexes, 2, 4)
```

---

`get_random_combinations`

*Get a large set of compatible combinations.*

---

## Description

Finds a randomly generated set of at most 1000 combinations of compatible barcodes.

## Usage

```
get_random_combinations(index_df, mplex_level, platform)
```

## Arguments

| | |
|---|---|
| `index_df` | A dataframe containing barcodes identifiers, corresponding DNA sequences along with GC content and presence of homopolymers. |
| `mplex_level` | The number at which the barcodes will be multiplexed. Illumina recommends to not multiplex more than 96 libraries. |
| `platform` | An integer representing the number of channels (1, 2, 4) of the desired Illumina platform: 1 for iSeq; 2 for NextSeq, NovaSeq, MiniSeq; 4 for HiSeq and MiSeq. 0 represents any other platform than Illumina. |

## Details

This function is suited if the total number of possible combinations is too high for an exhaustive search to be possible in a reasonable amount of time.

## Value

A matrix containing the identifiers of compatible barcode combinations.

## See Also

get_all_combinations

## Examples

```
get_random_combinations(DNABarcodeCompatibility::IlluminaIndexes, 3, 4)
```

---

# IlluminaIndexes *Barcode dataset from Illumina with features.*

---

## Description

48 barcodes from Illumina TruSeq Small RNA kits along with percentage in CG content and presence of homopolymers of size >= 3

## Usage

```
IlluminaIndexes
```

## Format

A data frame with 48 rows and 4 variables:

**Id** barcode identifier

**sequence** DNA sequence

**GC_content** percentage of G and C relative to A and T

**homopolymer** presence of nucleotide repetitions (>= 3) ...

---

# IlluminaIndexesRaw *Barcode dataset from Illumina.*

---

## Description

48 barcodes from Illumina TruSeq Small RNA kits

## Usage

```
IlluminaIndexesRaw
```

## Format

A data frame with 48 rows and 2 variables:

**V1** barcode identifier

**V2** DNA sequence ...

`optimize_combinations` *Find a set of barcode combinations with least heterogeneity in barcode usage*

## Description

This function uses the Shannon Entropy to identify a set of compatible barcode combinations with least heterogeneity in barcode usage.

## Usage

```
optimize_combinations(combination_m, nb_lane, index_number,
thrs_size_comb, max_iteration, method)
```

## Arguments

| | |
|---|---|
| `combination_m` | A matrix of compatible barcode combinations. |
| `nb_lane` | The number of lanes to be use for sequencing (i.e. the number of libraries divided by the multiplex level). |
| `index_number` | The total number of distinct DNA barcodes in the dataset. |
| `thrs_size_comb` | The maximum size of the set of compatible combinations to be used for the greedy optimization. |
| `max_iteration` | The maximum number of iterations during the optimizing step. |
| `method` | The choice of the greedy search: 'greedy_exchange' or 'greedy_descent'. |

## Details

N/k compatible combinations are then selected using a Shannon entropy maximization approach. It can be shown that the maximum value of the entropy that can be attained for a selection of N barcodes among n, with possible repetitions, reads:

$$S_{max} = -(n-r)\frac{\lfloor N/n \rfloor}{N}\log(\frac{\lfloor N/n \rfloor}{N}) - r\frac{\lceil N/n \rceil}{N}\log(\frac{\lceil N/n \rceil}{N})$$

where r denotes the rest of the division of N by n, while

$$\lfloor N/n \rfloor$$

and

$$\lceil N/n \rceil$$

denote the lower and upper integer parts of N/n, respectively.

**Case 1:** number of lanes < number of compatible DNA-barcode combinations

This function seeks for compatible DNA-barcode combinations of highest entropy. In brief this function uses a randomized greedy descent algorithm to find an optimized selection. Note that the resulting optimized selection may not be globally optimal. It is actually close to optimal and much

improved in terms of non-redundancy of DNA barcodes used, compared to a randomly chosen set of combinations of compatible barcodes.

**Case 2:** number of lanes >= number of compatible DNA-barcode combinations

In such a case, there are not enough compatible DNA-barcode combinations and redundancy is inevitable.

## Value

A matrix containing an optimized set of combinations of compatible barcodes.

## See Also

get_all_combinations, get_random_combinations, experiment_design

## Examples

```
m <- get_random_combinations(DNABarcodeCompatibility::IlluminaIndexes, 3, 4)
optimize_combinations(m, 12, 48)
```

# Index