

# Package ‘BoxShuffler’

January 14, 2025

**Title** Optimize Randomized Sample Placement in Box-like spaces

**Version** 0.0.1.0

**Description** This script generates ``boxes" of samples under specific conditions while resolving adjacency conflicts and calculating entropy metrics. The primary purpose is to optimize sample placement in experimental setups to balance diversity and randomness.

**License** GPL (>= 3)

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.2

**Imports** dplyr,  
ggplot2,  
parallel,  
openxlsx

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

**Config/testthat/edition** 3

**VignetteBuilder** knitr

## Contents

---

calculate\_box\_metrics *Calculate Metrics for Multiple Boxes*

---

## Description

Computes aggregated metrics (entropy, unique genotypes, Simpson’s Diversity Index, randomness percentage) across multiple boxes.

**Usage**

```
calculate_box_metrics(boxes, genotypes)
```

**Arguments**

`boxes`                    A list of boxes (matrices) to analyze.  
`genotypes`                A vector of all possible genotypes.

**Value**

A list containing average and standard deviation of entropy, genotype counts, and Simpson's Diversity Index.

**Examples**

```
boxes <- list(
  list(box = matrix(c("cond1_G1_1", "cond1_G2_1", NA, NA), nrow = 2, byrow = TRUE)),
  list(box = matrix(c("cond1_G1_2", "cond1_G3_1", NA, NA), nrow = 2, byrow = TRUE))
)
genotypes <- c("G1", "G2", "G3")
calculate_box_metrics(boxes, genotypes)
```

---

compute_box_metrics	<i>Compute Box Metrics</i>
---------------------	----------------------------

---

**Description**

Computes entropy, unique genotype count, and Simpson's Diversity Index for a box.

**Usage**

```
compute_box_metrics(box)
```

**Arguments**

`box`                      A matrix representing the box.

**Value**

A list containing entropy, unique genotype count, and Simpson's Diversity Index.

**Examples**

```
box <- matrix(c("cond1_G1_1", "cond1_G2_1", "cond1_G1_2", NA), nrow = 2, ncol = 2)
compute_box_metrics(box)
```

---

exportBox	<i>Export Box Results</i>
-----------	---------------------------

---

**Description**

Exports box results to Excel and PDF formats for visualization and analysis.

**Usage**

```
exportBox(experiment, boxes, file_name = "experiment_output", font_size = NULL)
```

**Arguments**

experiment	The experiment number to export.
boxes	A list of box generation results.
file_name	Base name for output files.
font_size	Optional font size for PDF plots.

**Value**

Generates Excel and PDF files with the results.

**Examples**

```
samples <- data.frame(Genotype = c("G1", "G2", "G3"), Repls = c(2, 2, 1))
conditions <- c("cond1", "cond2")
boxes <- generateBoxes(samples, "Genotype", "Repls", conditions, 2, 2, 10, 1)
temp_file <- file.path(tempdir(), "experiment_results")
exportBox(1, boxes, file_name = temp_file)
```

---

generateBoxes	<i>Generate Boxes</i>
---------------	-----------------------

---

**Description**

Generates boxes of genotypic samples based on given conditions and parameters.

**Usage**

```
generateBoxes(
  samples,
  genotypes,
  reps,
  conditions,
  boxRows,
  boxCols,
  nIterations,
  nCores,
  availableBoxes = NULL
)
```

**Arguments**

<code>samples</code>	Data frame with genotype and replicate information.
<code>genotypes</code>	Column name for genotype data.
<code>reps</code>	Column name for replicate data.
<code>conditions</code>	Experimental conditions to consider.
<code>boxRows</code>	Number of rows in each box.
<code>boxCols</code>	Number of columns in each box.
<code>nIterations</code>	Number of iterations to attempt box generation.
<code>nCores</code>	Number of cores for parallel execution.
<code>availableBoxes</code>	Optional; maximum number of boxes allowed.

**Value**

A list of generated boxes with conflict resolution applied.

**Examples**

```
samples <- data.frame(Genotype = c("G1", "G2", "G3"), Reps = c(2, 2, 1))
conditions <- c("cond1", "cond2")
boxes <- generateBoxes(samples, "Genotype", "Reps", conditions, 2, 2, 10, 1)
```

---

generateMaps

*Generate Maps from Updated Sample Placement*


---

**Description**

This function reads a modified "Sample Placement" sheet and generates updated PDF maps for each box. It processes the user's modifications to the box layout and creates visualizations for all the updated boxes.

**Usage**

```
generateMaps(
  data,
  samples,
  rep,
  cond,
  box,
  row,
  col,
  file_name = "updated_maps",
  font_size = NULL
)
```

**Arguments**

data	A data frame (e.g., read from the "Sample Placement" sheet).
samples	The column name containing sample names.
rep	The column name containing replicate numbers.
cond	The column name containing condition labels.
box	The column name containing box identifiers.
row	The column name containing row numbers.
col	The column name containing column numbers.
file_name	The base name for the output PDF file.
font_size	Optional font size for the labels on the maps.

**Value**

A PDF file with visualizations for all boxes.

**Examples**

```
mock_data <- data.frame(
  Sample = c("G1", "G2", "G3", "G4", "G5"),
  Replicate = c(1, 2, 1, 3, 1),
  Condition = c("cond1", "cond1", "cond2", "cond2", "cond3"),
  Box = c(1, 1, 1, 2, 2),
  Row = c(1, 2, 3, 1, 2),
  Column = c(1, 2, 3, 1, 2)
)

temp_file <- file.path(tempdir(), "updated_map")
generateMaps(
  data = mock_data,
  samples = "Sample",
  rep = "Replicate",
  cond = "Condition",
  box = "Box",
  row = "Row",
```

```
col = "Column",
file_name = temp_file
)
```

---

is_adjacent	<i>Check for Adjacency Conflicts</i>
-------------	--------------------------------------

---

**Description**

Checks if a specific cell in a box matrix has adjacency conflicts based on the given genotype label.

**Usage**

```
is_adjacent(box, row, col, label)
```

**Arguments**

box	A matrix representing the box.
row	Row index of the cell to check.
col	Column index of the cell to check.
label	The genotype label in the cell.

**Value**

Logical value: TRUE if there is a conflict, FALSE otherwise.

**Examples**

```
box <- matrix(NA, nrow = 5, ncol = 5)
box[1, 1] <- "cond1_G1_1"
is_adjacent(box, 1, 2, "cond1_G1_2")
```

---

resolve_conflicts	<i>Resolve Adjacency Conflicts</i>
-------------------	------------------------------------

---

**Description**

Resolves adjacency conflicts in a box by swapping labels between cells.

**Usage**

```
resolve_conflicts(box, max_iterations = 1000)
```

**Arguments**

`box` A matrix representing the box.

`max_iterations` Maximum number of iterations to attempt resolution.

**Value**

A matrix with resolved conflicts.

**Examples**

```
box <- matrix(c(NA, "cond1_G1_1", "cond1_G1_2", NA), nrow = 2, ncol = 2)
resolve_conflicts(box, max_iterations = 100)
```

---

scoreBoxes	<i>Score Boxes</i>
------------	--------------------

---

**Description**

Evaluates generated boxes and computes metrics for each experiment.

**Usage**

```
scoreBoxes(box_results)
```

**Arguments**

`box_results` A list of box generation results.

**Value**

A data frame with metrics for each experiment, including entropy, unique genotype counts, and Simpson's Diversity Index.

**Examples**

```
box_results <- list(list(
  iteration = 1,
  boxes = list(
    cond1 = list(
      list(box = matrix(c("cond1_G1_1", "cond1_G2_1", NA, NA), nrow = 2, ncol = 2))
    )
  )
))
scoreBoxes(box_results)
```

---

validate_conflicts	<i>Validate Adjacency Conflicts</i>
--------------------	-------------------------------------

---

**Description**

Validates if any adjacency conflicts exist in a box.

**Usage**

```
validate_conflicts(box)
```

**Arguments**

box	A matrix representing the box.
-----	--------------------------------

**Value**

Logical value: TRUE if no conflicts exist, FALSE otherwise.

**Examples**

```
box <- matrix(c(NA, "cond1_G1_1", "cond1_G1_2", NA), nrow = 2, ncol = 2)
validate_conflicts(box)
```