

# Package ‘OSAT’

April 27, 2026

**Type** Package

**Title** OSAT: Optimal Sample Assignment Tool

**Version** 1.59.0

**Author** Li Yan

**Maintainer** Li Yan <li.yan@roswellpark.org>

**Depends** methods,stats

**Suggests** xtable, Biobase

**Description** A sizable genomics study such as microarray often involves the use of multiple batches (groups) of experiment due to practical complication. To minimize batch effects, a careful experiment design should ensure the even distribution of biological groups and confounding factors across batches. OSAT (Optimal Sample Assignment Tool) is developed to facilitate the allocation of collected samples to different batches. With minimum steps, it produces setup that optimizes the even distribution of samples in groups of biological interest into different batches, reducing the confounding or correlation between batches and the biological variables of interest. It can also optimize the even distribution of confounding factors across batches. Our tool can handle challenging instances where incomplete and unbalanced sample collections are involved as well as ideal balanced RCBD. OSAT provides a number of predefined layout for some of the most commonly used genomics platform. Related paper can be find at <http://www.biomedcentral.com/1471-2164/13/689> .

**URL** <http://www.biomedcentral.com/1471-2164/13/689>

**License** Artistic-2.0

**Collate** AllClasses.R AllGenerics.R gAssembly-class.R gSample-class.R gSetup-class.R gUtils.R

**biocViews** DataRepresentation, Visualization, ExperimentalDesign, QualityControl

**git\_url** <https://git.bioconductor.org/packages/OSAT>

**git\_branch** devel

**git\_last\_commit** 3e012f9

**git\_last\_commit\_date** 2025-10-29

**Repository** Bioconductor 3.24

**Date/Publication** 2026-04-26

## Contents

BeadChip-class . . . . .	2
BeadPlate-class . . . . .	3
create.optimized.setup . . . . .	4
example.setup . . . . .	5
gArray-class . . . . .	5
gAssembly-class . . . . .	6
gContainer-class . . . . .	7
get.experiment.setup . . . . .	8
gExperimentSetup-class . . . . .	9
gPlate-class . . . . .	10
gSample-class . . . . .	11
gSlide-class . . . . .	12
MSAroboticPlate-class . . . . .	12
multi.barplot . . . . .	13
multi.chisq.test . . . . .	14
optimal.block . . . . .	14
optimal.shuffle . . . . .	15
plot-methods . . . . .	16
predefined . . . . .	17
QC . . . . .	17
<b>Index</b>	<b>18</b>

---

BeadChip-class	<i>Class "BeadChip"</i>
----------------	-------------------------

---

### Description

A predefined class represent BeadChip from Illumina. Inherited from class gSlide.

### Predefined chips

IlluminaBeadChip: A slide with 6 rows and 2 columns. The 12 wells is numbered columns first.

GenotypingChip: A slide with 12 rows and one column.

### Extends

Class "gSlide", directly. Class "gArray", by class "gSlide", distance 2.

### Methods

**getLayout** signature(x = "BeadChip"): Return a data frame represent the layout of the chip.

**Examples**

```
newChip <- new("BeadChip", nRows=6, nColumns=1, byrow=FALSE, comment="mock chip")
newChip

GenotypingChip
IlluminaBeadChip
```

---

BeadPlate-class	Class "BeadPlate"
-----------------	-------------------

---

**Description**

A class represent a plate consistent of multiple chips. Inherit from class `gPlate`.

**Objects from the Class**

Objects can be created by calls of the form `new("BeadPlate", chip, nRows, nColumns, comment, ...)`.

**Slots**

`chip`: Object of class `"gSlide"`. The slide used in the plate.  
`nRows`: Object of class `"integer"`. Number of clide per row.  
`nColumns`: Object of class `"integer"`. Number of slide per column.  
`layout`: Object of class `"data.frame"`. A data frame represent the layout of the plate.  
`byrow`: Object of class `"logical"`. Order of slide on the plate.  
`metadata`: Object of class `"list"`. A brief description of the plate.

**Predefined plate**

```
IlluminaBeadChip96Plate
IlluminaBeadChip48Plate
IlluminaBeadChip24Plate
```

are plates that hold 2, 4, 8 `IlluminaBeadChip` chips and have 24, 48, 96 wells, respectively.

**Extends**

Class `"gPlate"`, directly. Class `"gArray"`, by class `"gPlate"`, distance 2.

**Examples**

```
library("OSAT")
newPlate <- new("BeadPlate", chip=IlluminaBeadChip, nRows=2L, nColumns=4L, comment="mock plate")
newPlate
```

---

```
create.optimized.setup
```

*Create optimized setup*

---

## Description

Create a optimized sample assignment.

## Usage

```
create.optimized.setup(fun = "default", sample, container, ...)
```

## Arguments

fun	The name of the optimization function. When omitted, it is the same as default or optimal.shuffle. Currently alternative optimization function is optimal.block. User can define their own optimization function.
sample	A gSample object contains sample related information.
container	A gContainer object contains experiment related container (chips, plate, exclusion etc) information.
...	Additional parameters passed to optimization function.

## Details

Currently two methods are available: optimal.shuffle (default) and optimal.block methods. The function is equivalent to create.experiment.setup() followed by corresponding optimization function of the same name.

## Value

A gExperimentSetup object is returned to store all related information.

## Examples

```
library("OSAT")
# data as an example
inPath <- system.file("extdata", package="OSAT")
pheno <- read.table(file.path(inPath, 'samples.txt'), header=TRUE, sep="\t")

## create object to hold sample information
gs <- setup.sample(pheno, optimal=c("SampleType", "Race", "AgeGrp"), strata=c("SampleType"))
gs

gc <- setup.container(IlluminaBeadChip96Plate, 6, batch='plates')
gc
# demonstration only. nSim=5000 or more are commonly used.
gSetup <- create.optimized.setup(sample=gs, container=gc, nSim=500)
```

---

example.setup	<i>R data object based on example file.</i>
---------------	---

---

**Description**

Data objects created using the example data for demonstration.

**Usage**

```
data(example.setup)
```

**Details**

See vignette for codes that created these data objects.

a data frame created from example sample information text file.

**pheno**: a class gSample object created based on example sample using function `setup.sample()` ..

**gc**: a class gContainer object using function `setup.container()`.

**gSetup**: a class gExperimentSetup object. Created using `optimal.shuffle()` method.

**gSetup2**: a class gExperimentSetup object. Created using `optimal.block()` method.

**Examples**

```
library("OSAT")
data(example.setup)
head(pheno)
gs
gc
gSetup
```

---

gArray-class	<i>Class "gArray"</i>
--------------	-----------------------

---

**Description**

The virtual class gArray is used to standardize the storage and access of array-like experiment arrangement. It represents the arrangement of the experiment units, such as slides and plates.

**Objects from the Class**

A virtual Class: No objects may be created from it.

**Slots**

**nRows**: Object of class "integer". Number of units per row.

**nColumns**: Object of class "integer". Number of units per column.

**layout**: Object of class "data.frame". A data frame represents the layout and order of position.

**byrow**: Object of class "logical". The order of position is filled by row or not. Default is FALSE, same as used in `matrix()`.

**metadata**: Object of class "list". A list that holds brief description of the class and other information.

## Methods

In the following code snippets, x is an gArray object.

**dim** signature(x = "gArray"): Return the dimension of the array.

**getLayout** signature(x = "gArray"): Return a data frame represent the layout positions of the array.

**metadata<-** signature(x = "gArray"): Set the metadata.

**metadata** signature(x = "gArray"): Get the metadata.

**ncol** signature(x = "gArray"): Return the number of columns.

**nrow** signature(x = "gArray"): Return the number of rows.

---

gAssembly-class	Class "gAssembly"
-----------------	-------------------

---

## Description

A class represent an assembly of experiment plates.

## Objects from the Class

Objects can be created by calls of the form `new("gAssembly", plate, n, comment)`.

## Slots

**plate:** Object of class "gArray". Plate used in the experiment.

**n:** Object of class "integer". Number of plates used.

**layout:** Object of class "data.frame". The layout of wells in the assembly.

**metadata:** Object of class "list". A list holds meta data of the assembly.

## Methods

**getLayout** signature(x = "gAssembly"): Return a data frame that represent the wells location in the assembly.

**metadata** signature(x = "gAssembly"): Get metadata.

**metadata<-** signature(x = "gAssembly"): Set metadata.

**show** signature(object = "gAssembly"): Print a brief summary of the assembly.

## See Also

["gContainer"](#).

---

gContainer-class	Class "gContainer"
------------------	--------------------

---

### Description

A class for storage information related to the experiment container setup.

### Usage

```
setup.container(plate, n, batch = "plates", exclude = NULL)
```

### Arguments

plate	A object of gPlate class or its expanded class.
n	Number of plates used in the experiment.
batch	The level where batch effect is considered.
exclude	A data frame indicate location of wells in the container that should be excluded from sample assignment in the experiment.

### Value

A gContainer object.

### Objects from the Class

Objects can be created by calls of the form `setup.container(plate, n, batch, exclude)`.

### Slots

**plate:** Object of class "gPlate". Plate used in the experiment.  
**n:** Object of class "integer". Number of plate used.  
**batch:** Object of class "character". On what level batch effect are considered. Could be "plates" or "chips".  
**exclude:** Object of class "data.frame". A data frame indicate wells that should be excluded from the sample Assignment.  
**data:** Object of class "list". A list holds summaries and other useful informaiton.  
**metadata:** Object of class "list". A list for a brief description and other useful informtion.

### Methods

**get.gAssembly** signature(x = "gContainer"): Return the assembly of the plates used in the experiments.  
**getLayout** signature(x = "gContainer"): Return a data frame that holds the layout of available wells in the container.  
**exclude<-** signature(object = "gContainer"): Exclude some wells/chips/plates from the container for randomization.  
**metadata<-** signature(x = "gContainer"): Set the metadata.  
**metadata** signature(x = "gContainer"): Get the metadata.  
**show** signature(object = "gContainer"): A brief summary of the container.

**Examples**

```
library("OSAT")
# a container consist of 6 predefined Illumina plates with 96 wells each
gc <- setup.container(IlluminaBeadChip96Plate, 6, batch='plates')
gc

# to exclude first wells on first chips of each plate.
(excludedWells <- data.frame(plates=1:6, chips=rep(1,6), wells=rep(1,6)) )
gc3 <- setup.container(IlluminaBeadChip96Plate, 6, batch='plates', exclude=excludedWells)
```

---

get.experiment.setup *Get experiment assignment after optimization.*

---

**Description**

Return a data frame that contains sample and assigned well position.

**Usage**

```
get.experiment.setup(x)
```

**Arguments**

x                    A gExperimentSetup object.

**Value**

A data frame is returned in the order of initial sample data frame, with additional columns indicate assigned well position.

**Examples**

```
library("OSAT")
# data as an example
inPath <- system.file("extdata", package="OSAT")
pheno <- read.table(file.path(inPath, 'samples.txt'), header=TRUE, sep="\t")

## create object to hold sample information
gs <- setup.sample(pheno, optimal=c("SampleType", "Race", "AgeGrp"), strata=c("SampleType"))
gs

gc <- setup.container(IlluminaBeadChip96Plate, 6, batch='plates')
gc
# demonstration only. nSim=5000 or more are commonly used.
gSetup <- create.optimized.setup(sample=gs, container=gc, nSim=500)

mySetup <- get.experiment.setup(gSetup)
```

---

gExperimentSetup-class

Class "gExperimentSetup"

---

## Description

A class stores all relevant informations related to the experimental sample assignment.

## Details

Function `create.experiment.setup()` create a block randomized experiment assignment, without optimization step.

Functions `optimal.shuffle()` and `optimal.block` optimize setup using different optimization methods.

Function `create.optimized.setup()` create a optimized setup directly.

## Slots

**expSetup:** Object of class "data.frame". A data frame represents the sample placement to well locations in the container.

**data:** Object of class "list". A list that hold a gSample object, a gContainer object and assignment link the two objects.

**summaryInfo:** Object of class "list". Some summary of the sample, container and assignment.

**metadata:** Object of class "list". Metadata of the object.

## Accessors

**get.gAssembly** signature(x = "gExperimentSetup"): Get the chip/plate assembly used for container.

**samples** signature(x = "gExperimentSetup"): Return the gSample object.

**get.gContainer** : ...

**metadata** signature(x = "gExperimentSetup"): Get the metadata.

**metadata<-** signature(x = "gExperimentSetup"): Set the metadata.

## Related Methods

**map.to.MSA** signature(x = "gExperimentSetup", y = "MSAr roboticPlate"): A method map the experiment setup to a set of MSA 96 wells robotic plates.

**plot** signature(x = "gExperimentSetup", y = "missing"): Visual

**show** signature(object = "gExperimentSetup"): ...

**summary** signature(object = "gExperimentSetup"): ...

**get.experiment.setup** : Return a data frame with linked sample and container information.

## See Also

[MSAr roboticPlate](#)

**Examples**

```

library("OSAT")
# data as an example
inPath <- system.file("extdata", package="OSAT")
pheno <- read.table(file.path(inPath, 'samples.txt'), header=TRUE, sep="\t")

## create object to hold sample information
gs <- setup.sample(pheno, optimal=c("SampleType", "Race", "AgeGrp"), strata=c("SampleType") )
gs

gc <- setup.container(IlluminaBeadChip96Plate, 6, batch='plates')
gc

gSetup0 <- create.experiment.setup(sample=gs, container=gc)

```

---

gPlate-class

*Class "gPlate"*


---

**Description**

A virtual class represent a experiment plate that is consistent of multiple slides.

**Objects from the Class**

A virtual Class: No objects may be created from it.

**Slots**

**chip:** Object of class "gSlide". A virtual class represent a experiment plate that is consistent of multiple slides.

**nRows:** Object of class "integer". Number of slide per row.

**nColumns:** Object of class "integer". Number of slide per column.

**layout:** Object of class "data.frame". A data frame represent the layout of the plate. Usually down to the wells level.

**byrow:** Object of class "logical". The order of the position is filled by row or not. Default is FALSE, same as used in mactrix().

**metadata:** Object of class "list". A list that holds brief description of the class and other information.

**Extends**

Class "gArray", directly.

---

gSample-class	Class gSample
---------------	---------------

---

### Description

Create a class used for storage of sample related information.

### Usage

```
setup.sample(x, optimal, strata)
summary(object, ...)
```

### Arguments

x	a data frame holds sample variables.
optimal	a vector of sample variable names to be treated as optimal variables.
strata	a vector of sample variable names to be treated as block variables. If omitted, the first element of optimal vector is treated as strata.
object	an object of class gSample.
...	additional arguments affecting the summary produced.

### Value

An object of gSample class for function `setup.sample()`. A list of two tables for function `summary(object)`:

strataTable	Sample frequency table by stata variables.
optimalTable	Sample frequency table by optimal variables.

### Slots

**rawData:** Object of class "data.frame" The original data frame that holds the sample information.

**optimal:** Object of class "character" A character vector of column names in the sample info data frame. Represent the optimal variables.

**strata:** Object of class "character" A character vector of column names in the sample info data frame. Represent the blocking variables.

**data:** Object of class "list" A list holds summaries and other useful information.

### Methods

**show** signature(object = "gSample"): ...

### Examples

```
inPath <- system.file("extdata", package="OSAT")
pheno <- read.table(file.path(inPath, 'samples.txt'), header=TRUE, sep="\t")
## create object to hold sample information
gs <- setup.sample(pheno, optimal=c("SampleType", "Race", "AgeGrp"), strata=c("SampleType"))
```

---

gSlide-class                      *Class "gSlide"*

---

### Description

A class represent a slide used in experiment. Inherited from Class "gArray". The layout of the slide is described by a data frame with columns of "rows", "columns", and "wells".

### Objects from the Class

Objects can be created by calls of the form `new("gSlide", nRows, nColumns, byrow, comment)`.

### Extends

Class "gArray", directly.

### Methods

**getLayout** signature(`x = "gSlide"`): ...  
**initialize** signature(`.Object = "gSlide"`): ...  
**show** signature(`object = "gSlide"`): ...

### Examples

```
newSlide <- new("gSlide", nRows=2, nColumns=2, byrow=FALSE,comment="mock slide")
newSlide
```

---

MSAr roboticPlate-class    *Class "MSAr roboticPlate"*

---

### Description

A class store layout information of MSA robotic loader plate.

### Extends

Class "gArray", directly.

### Methods

**map.to.MSA** signature(`x = "data.frame"`, `y = "MSAr roboticPlate"`): Return a data frame that assign samples (represented by the input data frame) to MSA robotic plate sequentially.  
**map.to.MSA** signature(`x = "gExperimentSetup"`, `y = "MSAr roboticPlate"`): Return a data frame that assign samples from an `gExperimentSetup` object to MSA robotic plate.  
**show** signature(`object = "MSAr roboticPlate"`): Shows the layout of the plate.

### Predefined objects

`MSA4.plate`: A predefined object of class `MSAr roboticPlate` that represent a 96 position plate.  
`BeadChip96ToMSA4MAPMap`: The loading order a MSA4 robotic loader used to load `BeadChips`.

**See Also**[gExperimentSetup](#)**Examples**

```

library("OSAT")
# data as an example
inPath <- system.file("extdata", package="OSAT")
pheno <- read.table(file.path(inPath, 'samples.txt'), header=TRUE, sep="\t")

## create object to hold sample information
gs <- setup.sample(pheno, optimal=c("SampleType", "Race", "AgeGrp"), strata=c("SampleType"))
gs

gc <- setup.container(IlluminaBeadChip96Plate, 6, batch='plates')
gc

gSetup <- create.optimized.setup(sample=gs, container=gc, nSim=100)
out <- map.to.MSA(gSetup, MSA4.plate)

```

multi.barplot

*multi.barplot***Description**

Plot multiple bar plots based on a single data frame.

**Usage**

```
multi.barplot(x, grpVar = "plates", varList, main = NULL, ...)
```

**Arguments**

x	A data frame.
grpVar	The variable for x-axis.
varList	A vector of variables, each will be used as y-axis for the multiple bar plot.
main	The overall title of the plot.
...	Additional parameters for the plot function.

**Value**

Use the side effect for plot. No return value.

**Examples**

```

## create a random assignment and check it
library("OSAT")
data(example.setup)
set.seed(10)
c1 <- getLayout(gc) # available wells
c1 <- c1[order(runif(nrow(c1))),] # shuffle randomly
ranomSetup <- cbind(pheno, c1[1:nrow(pheno), ]) # create a sample assignment

multi.barplot(ranomSetup, grpVar='plates', varList=c("SampleType", "Race", "AgeGrp"), main="A random case")

```

---

```
multi.chisq.test      multi.chisq.test
```

---

**Description**

Do a few Chi-square tests based on the same data frame.

**Usage**

```
multi.chisq.test(x, grpVar = "plates", varList, main = NULL)
```

**Arguments**

x	A data frame.
grpVar	Common variables. Default is 'plate'.
varList	A vector of variables.
main	The overall title.

**Examples**

```
## create a random assignment and check it
library("OSAT")
data(example.setup)
set.seed(10)
c1 <- getLayout(gc)           # available wells
c1 <- c1[order(runif(nrow(c1))),] # shuffle randomly
ranomSetup <- cbind(pheno, c1[1:nrow(pheno), ]) # create a sample assignment

multi.chisq.test(ranomSetup, grpVar='plates', varList=c("SampleType", "Race", "AgeGrp"))
```

---

```
optimal.block      optimal.block
```

---

**Description**

Optimize a sample assignment setup by selecting from multiple candidate setup.

**Usage**

```
optimal.block(x, nSim = 100)
```

**Arguments**

x	A gExperimentSetup object.
nSim	Number of candidate setup created.

## Details

Multiple (typically thousands of or more) sample assignment setups are first generated, based only on the list of specified blocking variable(s). Then, the optimal setup is chosen by selecting the setup of sample assignment (from the pool generated in blocking step) which minimizes the value of the objective function based on all variables considered.

## Value

A `gExperimentSetup` object, after optimization.

## See Also

[optimal.shuffle](#)

## Examples

```
library("OSAT")
# data as an example
inPath <- system.file("extdata", package="OSAT")
pheno <- read.table(file.path(inPath, 'samples.txt'), header=TRUE, sep="\t")

## create object to hold sample information
gs <- setup.sample(pheno, optimal=c("SampleType", "Race", "AgeGrp"), strata=c("SampleType") )
gs

gc <- setup.container(IlluminaBeadChip96Plate, 6, batch='plates')
gc

gSetup0 <- create.experiment.setup(sample=gs, container=gc)

g2 <- optimal.block(gSetup0, nSim=100)
```

---

optimal.shuffle

*optimal.shuffle*

---

## Description

Optimize a sample assignment setup by shuffling.

## Usage

```
optimal.shuffle(x, nSim, k)
```

## Arguments

x	A <code>gExperimentSetup</code> object.
nSim	Number of shuffling steps.
k	Number of samples been shuffled. Default k=2 when omitted..

## Details

Given any `gExperimentSetup` object, we randomly select `k` samples from different batches and shuffle them between batches to create a new sample assignment. `k = 2` by default but could be any number up to half of the sample size. Value of the objective function is calculated on the new setup and compared to that of the original one. If the value is smaller then the new assignment replaces the previous one. This procedure will continue until we reach a preset number of attempts (usually in the tens of thousands).

## Value

A class `gExperimentSetup` object, after optimized.

## See Also

[optimal.block](#)

## Examples

```
library("OSAT")
# data as an example
inPath <- system.file("extdata", package="OSAT")
pheno <- read.table(file.path(inPath, 'samples.txt'), header=TRUE, sep="\t")

## create object to hold sample information
gs <- setup.sample(pheno, optimal=c("SampleType", "Race", "AgeGrp"), strata=c("SampleType") )
gs

gc <- setup.container(IlluminaBeadChip96Plate, 6, batch='plates')
gc

gSetup0 <- create.experiment.setup(sample=gs, container=gc)
# demonstration only. nSim=5000 or more are commonly used.
g3 <- optimal.shuffle(gSetup0, nSim=500, k=2)
```

---

plot-methods

*Methods for Function plot*

---

## Description

Create bar plots based on the experiment setup.

## Methods

`signature(x = "gExperimentSetup", y = "missing")` Plot distribution of relevant variables over different batches.

---

predefined	<i>Show predefined objects in the package.</i>
------------	--

---

**Description**

Show predefined objects in the package.

**Usage**

```
predefined()
```

**Details**

Currently layout of two chips, three IlluminaBeadChip plates, and the MSA4 robotic loader are defined.

---

QC	<i>QC</i>
----	-----------

---

**Description**

Provide a visual summary of the sample placement per plates, and Chi-square test of dependence between plates and other considered variables from sample.

**Usage**

```
QC(object, main = NULL, ...)
```

**Arguments**

object	An object of class gExperimentSetup.
main	Mail title on the bar plot.
...	Additional plot parameters.

**Examples**

```
library("OSAT")
inPath <- system.file("extdata", package="OSAT")
pheno <- read.table(file.path(inPath, 'samples.txt'), header=TRUE, sep="\t")

## create object to hold sample information
gs <- setup.sample(pheno, optimal=c("SampleType", "Race", "AgeGrp"), strata=c("SampleType"))
## create object that represents the used in the experiment
gc <- setup.container(IlluminaBeadChip96Plate, 6, batch='plates')
## create an optimized setup.
# demonstration only. nSim=5000 or more are commonly used.
gSetup <- create.optimized.setup(sample=gs, container=gc, nSim=500)
QC(gSetup)
```

# Index

## \* classes

BeadChip-class, 2  
BeadPlate-class, 3  
gArray-class, 5  
gAssembly-class, 6  
gContainer-class, 7  
gExperimentSetup-class, 9  
gPlate-class, 10  
gSample-class, 11  
gSlide-class, 12  
MSAr roboticPlate-class, 12

## \* datasets

example.setup, 5

## \* methods

plot-methods, 16

BeadChip-class, 2

BeadChip96ToMSA4MAP  
(MSAr roboticPlate-class), 12

BeadPlate-class, 3

cid, gContainer-method  
(gContainer-class), 7

create.experiment.setup  
(gExperimentSetup-class), 9

create.optimized.setup, 4

dim, gArray-method (gArray-class), 5

example.setup, 5

exclude<- (gContainer-class), 7

exclude<-, gContainer-method  
(gContainer-class), 7

gArray, 2, 3, 10, 12

gArray-class, 5

gAssembly-class, 6

gc (example.setup), 5

gContainer, 6

gContainer-class, 7

GenotypingChip (BeadChip-class), 2

get.experiment.setup, 8

get.gAssembly, gContainer-method  
(gContainer-class), 7

get.gAssembly, gExperimentSetup-method  
(gExperimentSetup-class), 9

getLayout (gArray-class), 5

getLayout, gPlate-method  
(gPlate-class), 10

getLayout, BeadChip-method  
(BeadChip-class), 2

getLayout, gArray-method (gArray-class),  
5

getLayout, gAssembly-method  
(gAssembly-class), 6

getLayout, gContainer-method  
(gContainer-class), 7

getLayout, gSlide-method (gSlide-class),  
12

getLayout, MSAr roboticPlate-method  
(MSAr roboticPlate-class), 12

gExperimentSetup, 13

gExperimentSetup-class, 9

gPlate, 3

gPlate-class, 10

gs (example.setup), 5

gSample-class, 11

gSetup (example.setup), 5

gSetup2 (example.setup), 5

gSlide, 2

gSlide-class, 12

IlluminaBeadChip (BeadChip-class), 2

IlluminaBeadChip24Plate  
(BeadPlate-class), 3

IlluminaBeadChip48Plate  
(BeadPlate-class), 3

IlluminaBeadChip96Plate  
(BeadPlate-class), 3

initialize, gAssembly-method  
(gAssembly-class), 6

initialize, gContainer-method  
(gContainer-class), 7

initialize, gExperimentSetup-method  
(gExperimentSetup-class), 9

initialize, gPlate-method  
(gPlate-class), 10

- initialize,gSample-method  
(gSample-class), 11
- initialize,gSlide-method  
(gSlide-class), 12
- initialize,MSARoboticPlate-method  
(MSARoboticPlate-class), 12
- map.to.MSA (MSARoboticPlate-class), 12
- map.to.MSA,data.frame,MSARoboticPlate-method  
(MSARoboticPlate-class), 12
- map.to.MSA,gExperimentSetup,MSARoboticPlate-method  
(MSARoboticPlate-class), 12
- metadata (gArray-class), 5
- metadata,gArray-method (gArray-class), 5
- metadata,gAssembly-method  
(gAssembly-class), 6
- metadata,gContainer-method  
(gContainer-class), 7
- metadata,gExperimentSetup-method  
(gExperimentSetup-class), 9
- metadata<- (gArray-class), 5
- metadata<- ,gArray-method  
(gArray-class), 5
- metadata<- ,gAssembly-method  
(gAssembly-class), 6
- metadata<- ,gContainer-method  
(gContainer-class), 7
- metadata<- ,gExperimentSetup-method  
(gExperimentSetup-class), 9
- MSA4.plate (MSARoboticPlate-class), 12
- MSARoboticPlate, 9
- MSARoboticPlate-class, 12
- multi.barplot, 13
- multi.chisq.test, 14
- ncol,gArray-method (gArray-class), 5
- nrow,gArray-method (gArray-class), 5
- optimal.block, 14, 16
- optimal.shuffle, 15, 15
- pheno (example.setup), 5
- plot,gExperimentSetup,missing-method  
(plot-methods), 16
- plot-methods, 16
- predefined, 17
- QC, 17
- samples,gExperimentSetup-method  
(gExperimentSetup-class), 9
- setup.container (gContainer-class), 7
- setup.sample (gSample-class), 11
- show,gAssembly-method  
(gAssembly-class), 6
- show,gContainer-method  
(gContainer-class), 7
- show,gExperimentSetup-method  
(gExperimentSetup-class), 9
- show,gPlate-method (gPlate-class), 10
- show,gSample-method (gSample-class), 11
- show,gSlide-method (gSlide-class), 12
- show,MSARoboticPlate-method  
(MSARoboticPlate-class), 12
- summary (gSample-class), 11
- summary,gExperimentSetup-method  
(gExperimentSetup-class), 9
- summary,gSample-method (gSample-class),  
11