

# Package: IPCAPS (via r-universe)

October 26, 2024

**Type** Package

**Title** Iterative Pruning to Capture Population Structure

**Version** 1.1.8

**Description** An unsupervised clustering algorithm based on iterative pruning is for capturing population structure. This version supports ordinal data which can be applied directly to SNP data to identify fine-level population structure and it is built on the iterative pruning Principal Component Analysis ('ipPCA') algorithm as explained in Intarapanich et al. (2009) <[doi:10.1186/1471-2105-10-382](https://doi.org/10.1186/1471-2105-10-382)>. The 'IPCAPS' involves an iterative process using multiple splits based on multivariate Gaussian mixture modeling of principal components and 'Expectation-Maximization' clustering as explained in Lebre et al. (2015) <[doi:10.18637/jss.v067.i06](https://doi.org/10.18637/jss.v067.i06)>. In each iteration, rough clusters and outliers are also identified using the function rubikclust() from the R package 'KRIS'.

**Depends** R (>= 3.5.0)

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.1.1

**Imports** stats,utils,graphics,grDevices,MASS,Matrix,expm,KRIS,fpc,LPCM,apcluster,Rmixmod

**Suggests** testthat

**BugReports** <https://gitlab.com/kris.ccp/ipcaps/-/issues>

**URL** <https://gitlab.com/kris.ccp/ipcaps>

**Collate** 'parallelization.R' 'check.stopping.R' 'clustering.mode.R' 'clustering.R' 'data.R' 'export.groups.R' 'get.node.info.R' 'ipcaps-package.R' 'process.each.node.R' 'output.template.R' 'save.html.R' 'save.eigenplots.html.R' 'save.plots.label.html.R' 'save.plots.cluster.html.R' 'save.plots.R' 'postprocess.R' 'preprocess.R' 'ipcaps.R' 'top.discriminator.R'

**NeedsCompilation** no

**Author** Kridsakorn Chaichoompu [aut, cre], Kristel Van Steen [aut],  
Fentaw Abegaz [aut], Sissades Tongsimma [aut], Philip Shaw  
[aut], Anavaj Sakuntabhai [aut], Luisa Pereira [aut]

**Maintainer** Kridsakorn Chaichoompu <kridsakorn@biostatgen.org>

**Repository** CRAN

**Date/Publication** 2021-01-25 17:00:15 UTC

## Contents

cal.eigen.fit . . . . .	2
check.stopping . . . . .	3
clustering . . . . .	4
clustering.mode . . . . .	5
diff.eigen.fit . . . . .	5
diff.xy . . . . .	6
do.glm . . . . .	6
export.groups . . . . .	7
get.node.info . . . . .	8
ipcaps . . . . .	9
label . . . . .	12
output.template . . . . .	13
pasre.categorical.data . . . . .	14
PC . . . . .	14
postprocess . . . . .	15
preprocess . . . . .	15
process.each.node . . . . .	17
raw.data . . . . .	18
replace.missing . . . . .	19
save.eigenplots.html . . . . .	19
save.html . . . . .	20
save.plots . . . . .	21
save.plots.cluster.html . . . . .	22
save.plots.label.html . . . . .	23
top.discriminator . . . . .	24
<b>Index</b>	<b>27</b>

---

cal.eigen.fit	<i>(Internal function) Calculae a vector of EigenFit values, internally used for parallelization</i>
---------------	--

---

## Description

(Internal function) Calculae a vector of EigenFit values, internally used for parallelization

**Usage**

```
cal.eigen.fit(eigen.value)
```

**Arguments**

eigen.value      A vector of Eigenvalues return from [svd](#) (\$d), `rARPACK::svds` (\$d), [eigen](#) (\$values) or `rARPACK::eigs` (\$values).

**Value**

A vector of all possible values for EigenFit.

---

check.stopping	<i>(Internal function) Check whether the IPCAPS process meets the stopping criterion.</i>
----------------	---

---

**Description**

(Internal function) Check whether the IPCAPS process meets the stopping criterion.

**Usage**

```
check.stopping(eigen.value, threshold)
```

**Arguments**

eigen.value      A vector of Eigenvalues return from [svd](#) (\$d), `rARPACK::svds` (\$d), [eigen](#) (\$values) or `rARPACK::eigs` (\$values).

threshold        A threshold or a cutoff to stop the IPCAPS process. Also see [ipcaps](#) (the parameter threshold).

**Value**

A list containing status, eigen.value, eigen.fit, threshold, and no.significant.PC as explained below:

- \$status is either 0 representing that the criterion is not met, or 1 representing that the criterion is met.
- \$eigen.value is a vector of Eigenvalues as the input parameter.
- \$eigen.fit is a vector of EigenFit values.
- \$threshold is a threshold as the input parameter.
- \$no.significant.PC is an estimated number of significant principal components (PC).

---

clustering

*(Internal function) Perform the clustering process of IPCAPS*


---

### Description

(Internal function) Perform the clustering process of IPCAPS

### Usage

```
clustering(
  dataframe,
  node = 1,
  result.dir,
  threshold,
  min.fst,
  method = "mix",
  min.in.group = 20,
  datatype = "snp",
  nonlinear = FALSE
)
```

### Arguments

dataframe	A data frame containing raw.data (matrix or data frame), label (vector), and index (vector). raw.data represents a matrix of subset of input data. label represents a vector of labels for all rows of raw.data. index represents a vector of indexes that are selected from the original input data.
node	An integer representing the current node number which is being analyzed.
result.dir	An output directory
threshold	A threshold or a cutoff to stop the IPCAPS process. Also see <a href="#">ipcaps</a> (the parameter threshold).
min.fst	A number represents a cutoff for minimum Fst value.
method	A clustering method selected from the <a href="#">ipcaps</a> function. See <a href="#">ipcaps</a> for available methods.
min.in.group	A integer represents a minimum number of group members.
datatype	To specify whether the input data are 'snp' or other type. Defalut = 'snp'.
nonlinear	(Unimplemented) To specify whether linear or non-linear method is used for IPCAPS analysis. If TRUE, non-linear method is used, otherwise linear method is used. Default = FALSE.

### Value

A list containing status, node, and new.index as explained below:

- \$status is either 0 representing that the criterion is not met, or 1 representing that the criterion is met.

- \$node is an integer representing the current node number which is being analyzed.
- \$new.index is a list of vectors containing a subset of indexes split from dataframe\$index according to a clustering result.

---

clustering.mode	<i>(Internal function) Select a clustering method to be used for the IPCAPS process.</i>
-----------------	--

---

### Description

(Internal function) Select a clustering method to be used for the IPCAPS process.

### Usage

```
clustering.mode(node, work.dir, method)
```

### Arguments

node	An integer representing the current node number which is being analyzed.
work.dir	A working directory.
method	A clustering method selected from the <a href="#">ipcaps</a> function. See <a href="#">ipcaps</a> for available methods.

### Value

A vector of cluster assignment, for which cluster each individual belongs.

### See Also

[ipcaps](#)

---

diff.eigen.fit	<i>(Internal function) Calculate a vector of different values from a vector of EigenFit values, internally used for parallelization</i>
----------------	---

---

### Description

(Internal function) Calculate a vector of different values from a vector of EigenFit values, internally used for parallelization

### Usage

```
## S3 method for class 'eigen.fit'  
diff(eigen.value)
```

**Arguments**

eigen.value      A vector of Eigenvalues return from `svd ($d)`, `rARPACK::svds ($d)`, `eigen ($values)` or `rARPACK::eigs ($values)`.

**Value**

A vector of different values from a vector of EigenFit values

---

<code>diff.xy</code>	<i>(Internal function) Check the different value of X and Y, internally used for parallelization</i>
----------------------	--

---

**Description**

(Internal function) Check the different value of X and Y, internally used for parallelization

**Usage**

```
## S3 method for class 'xy'
diff(x, y)
```

**Arguments**

<code>x</code>	The first number
<code>y</code>	The second number

**Value**

The different number of `x` and `y`.

---

<code>do.glm</code>	<i>(Internal function) Perform regression models, internally used for parallelization</i>
---------------------	---

---

**Description**

(Internal function) Perform regression models, internally used for parallelization

**Usage**

```
do.glm(X, PC, method = "linear")
```

**Arguments**

X	A vector of data
PC	A matrix of principal components
method	Specify a method to be used for regression model, which can be "linear", "poisson", and "negative.binomial". Default = "linear"

**Value**

A vector of residuals computed from regression model

---

export.groups	<i>Export the IPCAPS result to a text file</i>
---------------	--

---

**Description**

Export clustering result of [ipcaps](#) to text file called 'groups.txt'.

**Usage**

```
export.groups(result.dir)
```

**Arguments**

result.dir      A result directory as the \$output object returned from the [ipcaps](#) function.

**Details**

After running, this function exports the file called 'groups.txt' to the same result directory. If 'groups.txt' already exists in the result directory, the exported file is changed to 'groups1.txt', 'groups2.txt', 'groups3.txt', ..., accordingly.

**Value**

A data frame of exported data containing 4 columns; group, node, label, row.number, as described below for more details:

- group represents group membership of IPCAPS result.
- node represents node numbers of IPCAPS result.
- label represents labels of rows in original input data.
- row.number represents row numbers of original input data.

## Examples

```
# Importantly, bed file, bim file, and fam file are required
# Use the example files embedded in the package

BED.file <- system.file("extdata", "ipcaps_example.bed", package="IPCAPS")
LABEL.file <- system.file("extdata", "ipcaps_example_individuals.txt.gz", package="IPCAPS")

my.cluster <- ipcaps(bed=BED.file, label.file=LABEL.file, lab.col=2, out=tempdir())

#Here, to export the IPCAPS result to a text file
exported.data <- export.groups(my.cluster$output.dir)
print(dim(exported.data))
head(exported.data)
```

---

get.node.info	<i>Get the information for specified node</i>
---------------	---

---

## Description

Obtain the information for specified node from the output list of [ipcaps](#).

## Usage

```
get.node.info(cluster.obj, node)
```

## Arguments

cluster.obj	A list returned from the <a href="#">ipcaps</a> function.
node	An integer representing a node number to enquire information as shown in the HTML output files.

## Value

The return value is NULL if node's information does not exist or a list containing PCs, eigen.fit, index, and label as explained below:

- \$PCs is a matrix of principal components of this node.
- \$eigen.fit is a number representing the EigenFit value of this node.
- \$index is a vector of row number (individuals) of raw.data (input data).
- \$label is the vector of labels of all individuals that belongs to this node.



## Examples

```
# Importantly, bed file, bim file, and fam file are required
# Use the example files embedded in the package

BED.file <- system.file("extdata", "ipcaps_example.bed", package="IPCAPS")
LABEL.file <- system.file("extdata", "ipcaps_example_individuals.txt.gz", package="IPCAPS")

my.cluster <- ipcaps(bed=BED.file, label.file=LABEL.file, lab.col=2, out=tempdir())

#Here, to obtain the information of specified node, for example, node 3
node.info <- get.node.info(my.cluster, 3)
ls(node.info)
```

---

ipcaps

*Perform unsupervised clustering to capture population structure based on iterative pruning*

---

## Description

This version supports ordinal data which can be applied directly to SNP data to identify fine-scale population structure. It was built on the iterative pruning Principal Component Analysis (ipPCA) algorithm (Intarapanich et al., 2009; Limpiti et al., 2011). The IPCAPS involves an iterative process using multiple splits based on multivariate Gaussian mixture modeling of principal components and Clustering EM estimation (Lebet et al., 2015). In each iteration, rough clusters and outliers are also identified using our own method called rubikclust (R package **KRIS**).

## Usage

```
ipcaps(
  bed = NA,
  rdata = NA,
  files = NA,
  label.file = NA,
  lab.col = 1,
  out,
  plot.as.pdf = FALSE,
  method = "mix",
  missing = NA,
  covariate = NA,
  cov.col.first = NA,
  cov.col.last = NA,
  threshold = 0.18,
  min.fst = 8e-04,
  min.in.group = 20,
  no.plot = FALSE
)
```

**Arguments**

bed	A PLINK binary format consists of 3 files; bed, bim, and fam. To generate these files from PLINK, use option <code>-make-bed</code> . See more details at: <a href="http://zzz.bwh.harvard.edu/plink/data.shtml">http://zzz.bwh.harvard.edu/plink/data.shtml</a> .
rdata	In case of re-analysis, it is convenient to run IPCAPS using the file <code>rawdata.RData</code> generated by IPCAPS. This file contains a matrix of SNPs ( <code>raw.data</code> ) and a vector of labels ( <code>label</code> ).
files	IPCAPS supports SNPs encoded as 0, 1 and 2 (dosage encoding). Rows represent SNPs and columns represent subjects. Each column needs to be separated by a space or a tab. A big text file should be divided into smaller files to load faster. For instance, to input 3 files, use as: <code>files=c('input1.txt', 'input2.txt', 'input3.txt')</code> .
label.file	An additional useful information (called "labels" in IPCAPS) related subject, for example, geographic location or disease phenotype. These labels (one at a time) are used in displaying the clustering outcome of IPCAPS. A label file must contain at least one column. However, it may contain more than one column in which case each column need to be separated by a space or a tab.
lab.col	The label in the label file to be used in the tree-like display of IPCAPS clustering results.
out	To set an absolute path for IPCAPS output. If the specified output directory already exists, result files are saved in sub-directories <code>cluster_out</code> , <code>cluster_out1</code> , <code>cluster_out2</code> , etc.
plot.as.pdf	To export plots as PDF. When omitted, plots are saved as PNG.
method	The internal clustering method. It can be set to "mix" (rubikclust & mixmod), "mixmod" (Lebret et al., 2015), "clara" (R: Clustering Large Applications), "pam" (R: Partitioning Around Medoids (PAM) Object), "meanshift" (Wang, 2016), "apcluster" (Bodenhofer et al., 2016), and "hclust" (R: Hierarchical Clustering). Default = "mix".
missing	Symbol used for missing genotypes. Default = NA.
covariate	A file of covariates; one covariate per column. SNPs can be adjusted for these covariates via regression modeling and residual computation.
cov.col.first	Refer to a covariate file, the first covariate to be considered as confounding variable.
cov.col.last	Refer to a covariate file, the last covariate to be considered as confounding variable. All the variables in between the <code>cov.col.first</code> and <code>cov.col.last</code> will be considered in the adjustment process.
threshold	Cutoff value for EigenFit. Possible values range from 0.03 to 0.18. The smaller, the potentially finer the substructure can be. Default = 0.18.
min.fst	Minimum Fst between a pair of subgroups. Default = 0.0008.
min.in.group	Minimum number of individuals to constitute a cluster or subgroup. Default = 20.
no.plot	No plot is generated if this option is TRUE. This option is useful when the system does not support X Windows in the unix based system. Default = FALSE.

## Details

The computational time depends on the number of individuals. Consequentially, if large data sets are analyzed, it may be necessary first to split data into smaller files, and then load into computer memory (with parameter 'files'). Internally, the split files are merged to construct a com-prehensive matrix.

## Value

Returns the list object containing 2 internal objects; `output.dir` as class character and `cluster` as class `data.frame`. The object `output.dir` stores a result directory. The object `cluster` contains 4 columns, `group`, `node`, `label`, and `row.number`. The column `group` contains the assigned groups from IPCAPS. The column `node` contains node numbers in a tree as illustrated in the HTML result files. The column `label` contains the given labels that is set to the parameter `label`. The column `row.number` contains indices to an input data, which is matched to row numbers of input matrix. All clustering result files are saved in one directory (as specified by `out`) containing assigned groups, hierarchical trees of group membership, PC plots, and binary data for further analysis.

- `$snp` is a SNP matrix from BED file.
- `$snp.info` is a `data.frame` of SNP information from BIM file.
- `$ind.info` is a `data.frame` of individual information from FAM file.

If function return `NULL`, it means the input files are not in proper format.

## References

- Bodenhofer, U., Palme, J., Melkonian, C., and Kothmeier, A. (2016). `apcluster` : Affinity Propagation Clustering. Available at: <https://CRAN.R-project.org/package=apcluster> (Accessed March 7, 2017).
- Intarapanich, A., Shaw, P. J., Assawamakin, A., Wangkumhang, P., Ngamphiw, C. , Chaichoompu, K., et al. (2009). Iterative pruning PCA improves resolution of highly structured populations. *BMC Bioinformatics* 10, 382. doi:10.1186/1471-2105-10-382.
- Lebret, R., Iovleff, S., Langrognet, F., Biernacki, C., Celeux, G., and Govaert, G. (2015). `Rmixmod`: TheRPackage of the Model-Based Unsupervised, Supervised, and Semi-Supervised Classification-MixmodLibrary. *J. Stat. Softw.* 67. doi:10.18637/jss.v067.i06.
- Limpiti, T., Intarapanich, A., Assawamakin, A., Shaw, P. J., Wangkumhang, P., Piriyaongsa, J., et al. (2011). Study of large and highly stratified population datasets by combining iterative pruning principal component analysis and structure. *BMC Bioinformatics* 12, 255. doi:10.1186/1471-2105-12-255.
- Maechler, M., Rousseeuw, P., Struyf, A., Hubert, M., and Hornik, K. (2017). `cluster`: Cluster Analysis Basics and Extensions. R: Clustering Large Applications Available at: <https://stat.ethz.ch/R-manual/R-devel/library/cluster/html/clara.html> (Accessed March 7, 2017).
- R Core Team (2017). R: A Language and Environment for Statistical Computing. Vienna, Austria: R Foundation for Statistical Computing Available at: <https://www.R-project.org/>.
- R: Hierarchical Clustering Available at: <https://stat.ethz.ch/R-manual/R-devel/library/stats/html/hclust.html> (Accessed March 7, 2017).

R: Partitioning Around Medoids (PAM) Object Available at: <https://stat.ethz.ch/R-manual/R-devel/library/cluster/html/pam.object.html> (Accessed March 7, 2017).

Wang, M. C. and D. (2016). MeanShift: Clustering via the Mean Shift Algorithm. Available at: <https://CRAN.R-project.org/package=MeanShift> (Accessed March 7, 2017).

## Examples

```
#Use the BED format as input
#Importantly, bed file, bim file, and fam file are required
#Use the example files embedded in the package

BED.file <- system.file("extdata", "ipcaps_example.bed", package = "IPCAPS")
LABEL.file <- system.file("extdata", "ipcaps_example_individuals.txt.gz",
                          package = "IPCAPS")
my.cluster1 <- ipcaps(bed = BED.file, label.file = LABEL.file, lab.col = 2,
out = tempdir())

table(my.cluster1$cluster$label, my.cluster1$cluster$group)

# Alternatively, use a text file as input
# Use the example files embedded in the package

#text.file <- system.file("extdata", "ipcaps_example_rowVar_colInd.txt.gz",
#                          package="IPCAPS")
#LABEL.file <- system.file("extdata", "ipcaps_example_individuals.txt.gz",
#                          package="IPCAPS")

#my.cluster2 <- ipcaps(files = c(text.file), label.file = LABEL.file, lab.col = 2,
#                          out=tempdir())
#table(my.cluster2$cluster$label, my.cluster2$cluster$group)

# The other alternative way, use an R Data file as input
# Use the example file embedded in the package

#rdata.file <- system.file("extdata", "ipcaps_example.rda", package = "IPCAPS")

#my.cluster3 <- ipcaps(rdata = rdata.file, out = tempdir())
#table(my.cluster3$cluster$label, my.cluster3$cluster$group)
```

---

label	<i>Synthetic dataset containing population labels for the dataset</i>
	raw.data

---

## Description

A dataset contains a character vector of 1,004 elements containing labels or populations of 1,004 individuals which they belong. Five populations and outliers were labeled as "pop1", "pop2", "pop3", "pop4", "pop5", and "outlier".

**Usage**

```
data(ipcaps_example)
```

**Format**

A vector with 1,004 elements.

**See Also**

[raw.data](#) and [PC](#)

---

output.template	<i>(Internal object) The HTML output template for IPCAPS</i>
-----------------	--

---

**Description**

output.template contains \$lno\_title, \$lno\_data, \$lno\_leafnode, \$lno\_body, and \$template as explained below:

**lno\_title** A index number of \$template that a title of HTML file is replaced.

**lno\_data** A index number of \$template that a data section in the HTML file is replaced.

**lno\_leafnode** A index number of \$template that a leaf-node section in the HTML file is replaced.

**lno\_body** A index number of \$template that a body section in the HTML file is replaced.

**template** A vector of characters for HTML file.

**Usage**

```
output.template
```

**Format**

A list with with 5 objects

---

```
pasre.categorical.data
```

*(Internal function) Manipulate categorical input files*

---

### Description

(Internal function) Manipulate categorical input files

### Usage

```
pasre.categorical.data(files)
```

### Arguments

`files` IPCAPS supports categorical data, which rows represent features and columns represent subjects or individuals. Each column needs to be separated by a space or a tab. A big text file should be divided into smaller files to load faster. For instance, to input 3 files, use as: `files=c( 'input1.txt', 'input2.txt', 'input3.txt')`.

### Value

A data frame of input data

---

```
PC
```

*Synthetic dataset containing the top 10 principal components (PC) from the dataset `raw.data`*

---

### Description

A dataset contains a numeric matrix of 1,004 rows and 10 columns of top 10 PCs calculated from the dataset `raw.data`. The PCs were calculated using linear principal component analysis (PCA), see more details at `KRIS::cal.pc.linear`

### Usage

```
data(ipcaps_example)
```

### Format

A matrix with 10 columns and 1,004 rows

### See Also

[raw.data](#) and [label](#)

---

postprocess	<i>(Internal function) Perform the post-processing step of IPCAPS</i>
-------------	---

---

**Description**

(Internal function) Perform the post-processing step of IPCAPS

**Usage**

```
postprocess(result.dir, reanalysis = FALSE)
```

**Arguments**

result.dir	A result directory as the \$output object returned from the <a href="#">ipcaps</a> function.
reanalysis	(Unimplemented) To specify whether it is re-analysis or not. If TRUE, it is re-analysis, otherwise it is not. Default = FALSE.

**Value**

A data frame of clustering result containing 4 columns; group, node, label, row.number, as described below for more details:

- group represents group membership of IPCAPS result.
- node represents node numbers of IPCAPS result.
- label represents labels of rows in original input data.
- row.number represents row numbers of original input data.

---

preprocess	<i>(Internal function) Perform the pre-processing step of IPCAPS</i>
------------	--

---

**Description**

(Internal function) Perform the pre-processing step of IPCAPS

**Usage**

```
preprocess(
  files,
  label.file,
  lab.col,
  rdata.infile,
  bed.infile,
  cate.list,
  result.dir,
  threshold,
```

```

min.fst,
max.thread = NA,
reanalysis = FALSE,
method = "mix",
min.in.group = 20,
datatype = "snp",
nonlinear = FALSE,
missing.char = NA,
regression.file = NA,
regression.col.first = NA,
regression.col.last = NA,
reg.method = "linear",
plot.as.pdf = NA,
no.plot = NA
)

```

### Arguments

files	IPCAPS supports SNPs encoded as 0, 1 and 2 (dosage encoding). Rows represent SNPs and columns represent subjects. Each column needs to be separated by a space or a tab. A big text file should be divided into smaller files to load faster. For instance, to input 3 files, use as: <code>files=c('input1.txt', 'input2.txt', 'input3.txt')</code> .
label.file	An additional useful information (called "labels" in IPCAPS) related subject, for example, geographic location or disease phenotype. These labels (one at a time) are used in displaying the clustering outcome of IPCAPS. A label file must contain at least one column. However, it may contain more than one column in which case each column need to be separated by a space or a tab.
lab.col	The label in the label file to be used in the tree-like display of IPCAPS clustering results.
rdata.infile	In case of re-analysis, it is convenient to run IPCAPS using the file <code>rawdata.RData</code> generated by IPCAPS. This file contains a matrix of SNPs ( <code>raw.data</code> ) and a vector of labels ( <code>label</code> ).
bed.infile	A PLINK binary format consists of 3 files; <code>bed</code> , <code>bim</code> , and <code>fam</code> . To generate these files from PLINK, use option <code>-make-bed</code> . See more details at: <a href="http://zzz.bwh.harvard.edu/plink/data.shtml">http://zzz.bwh.harvard.edu/plink/data.shtml</a> .
cate.list	(Unimplemented) A list of categorical input file (text). For instance, to input 3 files, use as: <code>files=c('input1.txt', 'input2.txt', 'input3.txt')</code> .
result.dir	To set an absolute path for IPCAPS output. If the specified output directory already exists, result files are saved in sub-directories <code>cluster_out</code> , <code>cluster_out1</code> , <code>cluster_out2</code> , etc.
threshold	Cutoff value for EigenFit.
min.fst	Minimum Fst between a pair of subgroups.
max.thread	(Require the parallelization patch) Maximum number of concurrent threads.
reanalysis	(Unimplemented) To specify whether it is re-analysis or not. If TRUE, it is re-analysis, otherwise it is not. Default = FALSE.



method	The internal clustering method. It can be set to "mix" (rubikclust & mixmod), "mixmod" (Lebet et al., 2015), "clara" (R: Clustering Large Applications), "pam" (R: Partitioning Around Medoids (PAM) Object), "meanshift" (Wang, 2016), "apcluster" (Bodenhofer et al., 2016), and "hclust" (R: Hierarchical Clustering). Default = "mix".
min.in.group	Minimum number of individuals to constitute a cluster or subgroup.
datatype	To specify whether the input data are 'snp' or other type. Default = 'snp'.
nonlinear	(Unimplemented) To specify whether linear or non-linear method is used for IPCAPS analysis. If TRUE, non-linear method is used, otherwise linear method is used. Default = FALSE.
missing.char	Symbol used for missing genotypes. Default = NA.
regression.file	A file of covariates; one covariate per column. SNPs can be adjusted for these covariates via regression modeling and residual computation.
regression.col.first	Refer to a covariate file, the first covariate to be considered as confounding variable.
regression.col.last	Refer to a covariate file, the last covariate to be considered as confounding variable. All the variables in between the cov.col.first and cov.col.last will be considered in the adjustment process.
reg.method	(Fixed) Specify a method for regression analysis. Default = 'linear'.
plot.as.pdf	To export plots as PDF. When omitted, plots are saved as PNG.
no.plot	No plot is generated if this option is TRUE. This option is useful when the system does not support X Windows in the unix based system. Default = FALSE.

### Value

A data frame of input data.

---

process.each.node      *(Internal function) Perform the iterative process for each node*

---

### Description

(Internal function) Perform the iterative process for each node

### Usage

```
process.each.node(node, work.dir)
```

### Arguments

node	An integer representing the current node number which is being analyzed.
work.dir	A working directory.

**Value**

NULL

---

`raw.data`*Synthetic dataset containing single nucleotide polymorphisms (SNP)*

---

**Description**

The `raw.data` is the simulated dataset which consists of 3,000 independent SNPs and 1,004 individuals belonging to one of 5 populations (200 individuals each) and 4 outlying individuals. The matrix `raw.data` contains the number 0, 1, and 2 representing SNP in additive coding. The pairwise genetic distance between populations are listed below (see Balding, 1995):

	pop1	pop2	pop3	pop4	pop5
pop1		0.0040	0.0059	0.0085	0.0101
pop2	0.0040		0.0055	0.0082	0.0099
pop3	0.0059	0.0055		0.0104	0.0119
pop4	0.0085	0.0082	0.0104		0.0139
pop5	0.0101	0.0099	0.0119	0.0139	

**Usage**`data(ipcaps_example)`**Format**

A matrix with 3,000 columns and 1,004 rows

**References**

Balding, D.J., and Nichols, R.A. (1995). A method for quantifying differentiation between populations at multi-allelic loci and its implications for investigating identity and paternity. *Genetica* 96, 3-12.

**See Also**[label](#) and [PC](#)

---

replace.missing	<i>(Internal function) Replace missing values by specified values, internally used for parallelization</i>
-----------------	--

---

**Description**

(Internal function) Replace missing values by specified values, internally used for parallelization

**Usage**

```
replace.missing(X, missing = NA, rep)
```

**Arguments**

X	A vector of data
missing	The old characters representing a missing value, for example, 'NA' or '-'
rep	The new characters to replace the old characters, for example, 'NULL' or '-1'

**Value**

A vector of data with replaced character for miss values

---

save.eigenplots.html	<i>Generate HTML file for EigenFit plots</i>
----------------------	--

---

**Description**

Generate HTML file called 'tree\_scee.html' from the result of [ipcaps](#). This function is a part of workflow in [save.plots](#). The clustering result is shown as a tree rendering by the online Google Organizational Chart library. Note that the Internet is required to view the HTML file.

**Usage**

```
save.eigenplots.html(output.dir)
```

**Arguments**

output.dir	A result directory as the \$output object returned from the <a href="#">ipcaps</a> function.
------------	--

**Details**

After running, this function generates the file called 'tree\_scee.html' in the same result directory. All plots are generated and saved as image files in the sub-directory 'images'.

**Value**

NULL

**See Also**[save.html](#), [save.plots](#), [save.plots.cluster.html](#), and [save.plots.label.html](#)**Examples**

```
# Importantly, bed file, bim file, and fam file are required
# Use the example files embedded in the package

BED.file <- system.file("extdata", "ipcaps_example.bed", package="IPCAPS")
LABEL.file <- system.file("extdata", "ipcaps_example_individuals.txt.gz", package="IPCAPS")

my.cluster <- ipcaps(bed=BED.file, label.file=LABEL.file, lab.col=2, out=tempdir())

#Here, to generate HTML file
save.eigenplots.html(my.cluster$output.dir)
```

---

`save.html`*Generate HTML file for clustering result in text mode*

---

**Description**

Generate HTML file called 'tree\_text.html' from the result of [ipcaps](#). The clustering result is shown as a tree rendering by the online Google Organizational Chart library. Note that the Internet is required to view the HTML file.

**Usage**

```
save.html(output.dir)
```

**Arguments**

`output.dir` A result directory as the `$output` object returned from the [ipcaps](#) function.

**Details**

After running, this function generates the file called 'tree\_text.html' in the same result directory.

**Value**

NULL

**See Also**[save.plots](#), [save.plots.cluster.html](#), [save.eigenplots.html](#), and [save.plots.label.html](#)

## Examples

```
# Importantly, bed file, bim file, and fam file are required
# Use the example files embedded in the package

BED.file <- system.file("extdata", "ipcaps_example.bed", package="IPCAPS")
LABEL.file <- system.file("extdata", "ipcaps_example_individuals.txt.gz", package="IPCAPS")

my.cluster <- ipcaps(bed=BED.file, label.file=LABEL.file, lab.col=2, out=tempdir())

#Here, to generate HTML file
save.html(my.cluster$output.dir)
```

---

save.plots

*Workflow to generate HTML files for all kinds of plots*

---

## Description

Generate HTML files and all image files (plots) from the result of [ipcaps](#). The clustering result is shown as a tree rendering by the online Google Organizational Chart library. Note that the Internet is required to view the HTML files.

## Usage

```
save.plots(output.dir)
```

## Arguments

`output.dir` A result directory as the `$output` object returned from the [ipcaps](#) function.

## Details

After running, this function generates all plots and saves as image files in the sub-directory 'images'. It calls [save.plots.cluster.html](#), [save.eigenplots.html](#), and [save.plots.label.html](#) to generate all HTML files.

## Value

NULL

## See Also

[save.html](#), [save.plots.cluster.html](#), [save.eigenplots.html](#), and [save.plots.label.html](#)

## Examples

```
# Importantly, bed file, bim file, and fam file are required
# Use the example files embedded in the package

BED.file <- system.file("extdata", "ipcaps_example.bed", package="IPCAPS")
LABEL.file <- system.file("extdata", "ipcaps_example_individuals.txt.gz", package="IPCAPS")

my.cluster <- ipcaps(bed=BED.file, label.file=LABEL.file, lab.col=2, out=tempdir())

#Here, to generate all plots and HTML files
save.plots.label.html(my.cluster$output.dir)
```

---

save.plots.cluster.html

*Generate HTML file for scatter plots which all data points are highlighted by IPCAPS clusters*

---

## Description

Generate HTML file called 'tree\_scatter\_cluster.html' from the result of [ipcaps](#). This function is a part of workflow in [save.plots](#). The clustering result is shown as a tree rendering by the online Google Organizational Chart library. Note that the Internet is required to view the HTML file.

## Usage

```
save.plots.cluster.html(output.dir)
```

## Arguments

output.dir      A result directory as the \$output object returned from the [ipcaps](#) function.

## Details

After running, this function generates the file called 'tree\_scatter\_cluster.html' in the same result directory. All plots are generated and saved as image files in the sub-directory 'images'.

## Value

NULL

## See Also

[save.html](#), [save.plots](#), [save.eigenplots.html](#), and [save.plots.label.html](#)

## Examples

```
# Importantly, bed file, bim file, and fam file are required
# Use the example files embedded in the package

BED.file <- system.file("extdata", "ipcaps_example.bed", package="IPCAPS")
LABEL.file <- system.file("extdata", "ipcaps_example_individuals.txt.gz", package="IPCAPS")

my.cluster <- ipcaps(bed=BED.file, label.file=LABEL.file, lab.col=2, out=tempdir())

#Here, to generate HTML file
save.plots.cluster.html(my.cluster$output.dir)
```

---

save.plots.label.html *Generate HTML file for scatter plots which data points are highlighted by given labels*

---

## Description

Generate HTML file called 'tree\_scatter\_label.html' from the result of [ipcaps](#). This function is a part of workflow in [save.plots](#). The clustering result is shown as a tree rendering by the online Google Organizational Chart library. Note that the Internet is required to view the HTML file.

## Usage

```
save.plots.label.html(output.dir)
```

## Arguments

output.dir      A result directory as the \$output object returned from the [ipcaps](#) function.

## Details

After running, this function generates the file called 'tree\_scatter\_label.html' in the same result directory. All plots are generated and saved as image files in the sub-directory 'images'.

## Value

NULL

## See Also

[save.html](#), [save.plots](#), [save.plots.cluster.html](#), and [save.eigenplots.html](#)

**Examples**

```
# Importantly, bed file, bim file, and fam file are required
# Use the example files embedded in the package

BED.file <- system.file("extdata", "ipcaps_example.bed", package="IPCAPS")
LABEL.file <- system.file("extdata", "ipcaps_example_individuals.txt.gz", package="IPCAPS")

my.cluster <- ipcaps(bed=BED.file, label.file=LABEL.file, lab.col=2, out=tempdir())

#Here, to generate HTML file
save.plots.label.html(my.cluster$output.dir)
```

---

top.discriminator      *Detecting top discriminators between two groups*

---

**Description**

Detects top discriminators that contribute to group separation based on the fixation index (Fst).

**Usage**

```
top.discriminator(
  cluster.obj = NULL,
  group1,
  group2,
  bim.file,
  use.node.number = FALSE,
  num.top = 100,
  percentile = 0.9,
  use.percentile = FALSE,
  use.path = FALSE,
  result.path = NULL
)
```

**Arguments**

cluster.obj	The object which is returned from <code>ipcaps</code> . This parameter is used when <code>use.path</code> is FALSE.
group1	To specify the first group number to be compared. (also see <code>use.node.number</code> )
group2	To specify the second group number to be compared. (also see <code>use.node.number</code> )
bim.file	Option: In case that SNP information is not provided to <code>ipcaps</code> , an absolute path of SNP information file is required. It needs to be in PLINK format (bim). See more details at: <a href="http://zzz.bwh.harvard.edu/plink/data.shtml">http://zzz.bwh.harvard.edu/plink/data.shtml</a> .
use.node.number	To specify whether a group number or a node number is be used. If TRUE, a node nubmer is used instead. Default = FALSE.



num.top	A number of top Fst SNPs to be returned. This parameter is used when percentile is FALSE. Default = 100.
percentile	A percentile for top Fst SNPs to be returned. This parameter is used when percentile is TRUE. Default = 0.999.
use.percentile	A logical value to indicate whether percentile is used instead of num.top. This parameter is used when percentile is TRUE. Default = FALSE.
use.path	A logical value to indicate whether result.path is used instead of cluster.obj. Importantly, result.path needs to be set. This parameter only work with the IPCAPS's result from version 1.1.7 onward. Default = FALSE.
result.path	A path to an result directory of IPCAPS. This parameter is used when use.path is TRUE. This parameter only work with the IPCAPS's result from version 1.1.7 onward.

## Value

The returned value is a data.frame of SNP information sorting by Fst in descending order, which contains 7 columns, chr, SNP, centimorgans, position, allele1, allele2, and Fst. The column 1-6 are SNP information from the bim file. The column Fst contains estimated Fst between group1 and group2.

## Examples

```
# Importantly, bed file, bim file, and fam file are required
# Use the example files embedded in the package
BED.file <- system.file("extdata", "ipcaps_example.bed", package="IPCAPS")
LABEL.file <- system.file("extdata", "ipcaps_example_individuals.txt.gz", package="IPCAPS")
my.cluster <- ipcaps(bed=BED.file, label.file=LABEL.file, lab.col=2, out=tempdir())
table(my.cluster$cluster$label, my.cluster$cluster$group)
# 1 2 3 4 5 6
# outlier4 5 4 1 0 0 0
# pop1 0 0 0 0 250 0
# pop2 0 0 0 0 0 250
# pop3 0 0 0 250 0 0

#Identify top discriminators between groups, for example, group 4 and group 5
top.snp <- top.discriminator(my.cluster, 4, 5)
#or, specify the bim file
#top.snp <- top.discriminator(my.cluster, 4, 5, bim.file="ipcaps_example.bim")
head(top.snp)
# chr SNP centimorgans position allele1 allele2 Fst
#V5452 1 marker5452 0 54520000 A T 0.11337260
#V2348 1 marker2348 0 23480000 A T 0.11194490
#V8244 1 marker8244 0 82440000 A T 0.09556580
#V5972 1 marker5972 0 59720000 A T 0.08747794
#V3561 1 marker3561 0 35610000 A T 0.08725860
#V8419 1 marker8419 0 84190000 A T 0.08293494

#Alternatively, specify the previous result directory of IPCAPS and identify
#top discriminators between groups, for example, group 4 and group 5
previous.res.path <- my.cluster$output.dir
```

```
top.snp <-top.discriminator(result.path = previous.res.path, use.path = TRUE,
group1 = 4, group2 = 5)
head(top.snp)

#Identify top discriminators between groups, for example, group 4 and group 5
top.snp <-top.discriminator(my.cluster,4,5)
#or, specify the bim file
#top.snp <-top.discriminator(my.cluster,4,5,bim.file="ipcaps_example.bim")
dim(top.snp)
head(top.snp)

#Here, it is possible to select the top Fst SNPs based on a percentile.
top.snp <-top.discriminator(my.cluster,4,5, percentile = 0.9,
use.percentile = TRUE)
dim(top.snp)
head(top.snp)

#Identify top discriminators between groups, for example, node 7 and node 8
top.snp2 <-top.discriminator(my.cluster,7,8,use.node.number=TRUE)
head(top.snp2)
# chr SNP centimorgans position allele1 allele2 Fst
#V5452 1 marker5452 0 54520000 A T 0.11337260
#V2348 1 marker2348 0 23480000 A T 0.11194490
```

# Index

- \* **PC**
  - PC, [14](#)
- \* **label**
  - label, [12](#)
- \* **output.template**
  - output.template, [13](#)
- \* **raw.data**
  - raw.data, [18](#)
  
- cal.eigen.fit, [2](#)
- check.stopping, [3](#)
- clustering, [4](#)
- clustering.mode, [5](#)
  
- diff.eigen.fit, [5](#)
- diff.xy, [6](#)
- do.glm, [6](#)
  
- eigen, [3](#)
- export.groups, [7](#)
  
- get.node.info, [8](#)
  
- ipcaps, [3–5](#), [7](#), [8](#), [9](#), [15](#), [19–24](#)
  
- label, [12](#), [14](#), [18](#)
  
- output.template, [13](#)
  
- pasre.categorical.data, [14](#)
- PC, [13](#), [14](#), [18](#)
- postprocess, [15](#)
- preprocess, [15](#)
- process.each.node, [17](#)
  
- raw.data, [13](#), [14](#), [18](#)
- replace.missing, [19](#)
  
- save.eigenplots.html, [19](#), [20–23](#)
- save.html, [20](#), [20](#), [21–23](#)
- save.plots, [19](#), [20](#), [21](#), [22](#), [23](#)
- save.plots.cluster.html, [20](#), [21](#), [22](#), [23](#)
- save.plots.label.html, [20–22](#), [23](#)
- svd, [3](#)
  
- top.discriminator, [24](#)