

# Package ‘iCNV’

December 2, 2021

**Title** Integrated Copy Number Variation detection

**Version** 1.14.0

**Author** Zilu Zhou, Nancy Zhang

**Maintainer** Zilu Zhou <zhouzilu@penmedicine.upenn.edu>

**Description** Integrative copy number variation (CNV) detection from multiple platform and experimental design.

**Depends** R (>= 3.3.1), CODEX

**License** GPL-2

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 6.0.1

**Imports** fields, ggplot2, truncnorm, tidyr, data.table, dplyr,  
grDevices, graphics, stats, utils, rlang

**Suggests** knitr, rmarkdown, WES.1KG.WUGSC

**VignetteBuilder** knitr

**biocViews** ImmunoOncology, ExomeSeq, WholeGenome, SNP,  
CopyNumberVariation, HiddenMarkovModel

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

**git\_branch** RELEASE\_3\_14

**git\_last\_commit** 8294aeb

**git\_last\_commit\_date** 2021-10-26

**Date/Publication** 2021-12-02

## R topics documented:

bambaf_from_vcf . . . . .	2
bed_generator . . . . .	3
chr . . . . .	4
filenm . . . . .	4
get_array_input . . . . .	5

icNV_detection . . . . .	5
icnv_output_to_gb . . . . .	7
icnv_res0 . . . . .	8
ngs_baf . . . . .	8
ngs_baf.chr . . . . .	9
ngs_baf.id . . . . .	9
ngs_baf.nm . . . . .	10
ngs_baf.pos . . . . .	10
ngs_plr . . . . .	11
ngs_plr.pos . . . . .	11
normObj . . . . .	12
output_list . . . . .	12
plotHMMscore . . . . .	13
plotindi . . . . .	14
plot_intensity . . . . .	15
projname . . . . .	15
qcObj . . . . .	16
sampname . . . . .	16
sampname_qc . . . . .	17
snp_baf . . . . .	17
snp_baf.pos . . . . .	18
snp_lrr . . . . .	18
snp_lrr.pos . . . . .	19
<b>Index</b>	<b>20</b>

---

bamfaf_from_vcf	<i>Get BAM baf information from vcf</i>
-----------------	---

---

## Description

If your vcf follow the format in the example, you could use this function to extract NGS baf from vcf files. Remember to load library before hands. Save 6 lists, each list has N entry. N = # of individuals (or vcf file) ngs\_baf.nm: name of the bamfiles; ngs\_baf.chr: the chromosome; ngs\_baf.pos: the position of the variants; ngs\_baf: the BAF of the variants; ngs\_baf.id: the ID of the variants; filenm:the file name

## Usage

```
bamfaf_from_vcf(dir = ".", vcf_list, chr = NULL, projname = "")
```

## Arguments

dir	The directory to all the vcf stored; default is right in this folder. Type character. Default `.`
vcf_list	All the vcf names stored in vcf.list; could use command:"ls *.vcf > vcf.list" to generate. Type character.

chr Specify the chromosome you want to generate. Must be of int from 1-22. If not specify, this function will generate all chromosomes. Default NULL

projname Name of the project. Type character. Default ""

**Value**

void

**Examples**

```
dir <- system.file("extdata", package="iCNV")
bambaf_from_vcf(dir, 'bam_vcf.list', projname='icnv.demo.')
bambaf_from_vcf(dir, 'bam_vcf.list', chr=22, projname='icnv.demo.')
```

---

bed\_generator *Generate BED file for WGS dataset.*

---

**Description**

Default position generated from USCS genome browser

**Usage**

```
bed_generator(chr = numeric(), hg = numeric(), start = NULL, end = NULL,
  by = 1000)
```

**Arguments**

chr Specify the chromosome you want to generate. Must be of int from 1-22. Type integer.

hg Specify the coordinate you want to generate from. Start and end position of hg19 and hg38 have been pre-implemented. Type integer.

start The start position of your BED file. Default NULL

end The end position of your BED file. Default NULL

by The chunk of your DNA for each bin. Type integer. Default 1000.

**Value**

void

**Examples**

```
bed_generator(chr=22, hg=38)
bed_generator(22, 38, 5001, 10000, by=500)
```

---

chr	<i>chromosome of the example</i>
-----	----------------------------------

---

**Description**

data chromosome

**Usage**

chr

**Format**

integer

**Value**

22

---

filenm	<i>Name of the file</i>
--------	-------------------------

---

**Description**

Example NGS VCF files for the 1000 Genome Project, value stored at filenm

**Usage**

filenm

**Format**

vector with NGS vcf file names

**Value**

File names for the NGS vcf

---

get_array_input	<i>Get array information from given format</i>
-----------------	--

---

### Description

If your array input file follow the format in the example, you could use this function to extract array LRR and baf. Remember to load library before hands. Save 4\*[# of chr] lists, each list has N entry. N = # of individuals snp\_lrr: SNP LRR intensity; snp\_lrr.pos: the position of the SNPs snp\_baf: the BAF of the SNPs; snp\_baf.pos: the position of the SNPs

### Usage

```
get_array_input(dir = character(), pattern = character(), chr = NULL,
               projname = "")
```

### Arguments

dir	A string. The directory path to the folder where store signal intensity file according to chr. Type character
pattern	A string. The pattern of all the intensity file. Type character
chr	Specify the chromosome you want to generate. Must be of int from 1-22. If not specify, this function will generate files for all chromosomes. Default NULL
projname	Name of the project. Type character

### Value

void

### Examples

```
dir <- system.file("extdata", package="iCNV")
pattern <- paste0('*.csv.arrayicnv$')
get_array_input(dir,pattern,chr=22,projname='icnv.demo.')
```

---

iCNV_detection	<i>CNV detection</i>
----------------	----------------------

---

### Description

Copy number variation detection tool for germline data. Able to combine intensity and BAF from SNP array and NGS data.

**Usage**

```
iCNV_detection(ngs_plr = NULL, snp_lrr = NULL, ngs_baf = NULL,
  snp_baf = NULL, ngs_plr.pos = NULL, snp_lrr.pos = NULL,
  ngs_baf.pos = NULL, snp_baf.pos = NULL, maxIt = 50, visual = 0,
  projname = "iCNV.", CN = 0, mu = c(-3, 0, 2), cap = FALSE)
```

**Arguments**

ngs_plr	A list of NGS intensity data. Each entry is an individual. If no NGS data, no need to specify.
snp_lrr	A list of SNP array intensity data. Each entry is an individual. If no SNP array data, no need to specify.
ngs_baf	A list of NGS BAF data. Each entry is an individual. If no NGS data, no need to specify.
snp_baf	A list of SNP array BAF data. Each entry is an individual. If no SNP array data, no need to specify.
ngs_plr.pos	A list of NGS intensity position data. Each entry is an individual with dimension= (#of bins or exons, 2(start and end position)). If no NGS data, no need to specify.
snp_lrr.pos	A list of SNP array intensity position data. Each entry is an individual with length=#of SNPs. If no SNP array data, no need to specify.
ngs_baf.pos	A list of NGS BAF position data. Each entry is an individual with length=#of BAFs. If no NGS data, no need to specify.
snp_baf.pos	A list of SNP array BAF position data. Each entry is an individual with length=#of BAFs. If no SNP array data, no need to specify.
maxIt	An integer number indicate the maximum number of EM iteration if not converged during parameter inference. Type integer. Default 50.
visual	An indicator variable with value 0,1,2. 0 indicates no visualization, 1 indicates basic visualization, 2 indicates complete visualization (Note visual 2 only work for single platform and integer CN inferred). Type integer. Default 0
projname	A string as the name of this project. Type character. Default 'iCNV.'
CN	An indicator variable with value 0,1 for whether wants to infer exact copy number. 0 no exact CN, 1 exact CN. Type integer. Default 0.
mu	A length three vector specify means of intensity in mixture normal distribution (Deletion, Diploid, Duplication). Default c(-3,0,2)
cap	A boolean decides whether we cap insane intensity value due to double deletion or mutiple amplification. Type logical. Default False

**Value**

(1) CNV inference, contains CNV inference, Start and end position for each inference, Conditional probability for each inference, mu for mixture normal, sigma for mixture normal, probability of CNVs, Z score for each inference.

(2) exact copy number for each CNV inference, if CN=1.

**Examples**

```
# icnv call without genotype (just infer deletion, duplication)
projname <- 'icnv.demo.'
icnv_res0 <- iCNV_detection(ngs_plr,snp_lrr,
                           ngs_baf,snp_baf,
                           ngs_plr.pos,snp_lrr.pos,
                           ngs_baf.pos,snp_baf.pos,
                           projname=projname,CN=0,mu=c(-3,0,2),cap=TRUE,visual = 1)
# icnv call with genotype inference and complete plot
projname <- 'icnv.demo.geno.'
icnv_res1 <- iCNV_detection(ngs_plr,snp_lrr,
                           ngs_baf,snp_baf,
                           ngs_plr.pos,snp_lrr.pos,
                           ngs_baf.pos,snp_baf.pos,
                           projname=projname,CN=1,mu=c(-3,0,2),cap=TRUE,visual = 2)
```

---

icnv_output_to_gb	<i>Convert icnv.output to input for Genome Browser.</i>
-------------------	---

---

**Description**

We could add the output to custom tracks on Genome Browser. Remember to choose human assembly matches your input data. We color coded the CNVs to make it as consistent as IGV. To show color, click 'User Track after submission', and edit config to 'visibility=2 itemRgb="On"'. Color see Github page for more example.

**Usage**

```
icnv_output_to_gb(chr = numeric(), icnv.output)
```

**Arguments**

chr	CNV chromosome. Type integer.
icnv.output	output from output_list_function

**Value**

matrix for Genome browser

**Examples**

```
icnv.output <- output_list(icnv_res=icnv_res0,sampleid=sampname_qc, CN=0, min_size=10000)
gb_input <- icnv_output_to_gb(chr=22,icnv.output)
write.table(gb_input,file='icnv_res_gb_chr22.tab',quote=FALSE,col.names=FALSE,row.names=FALSE)
```

---

`icnv_res0`*Example iCNV calling results.*

---

**Description**

iCNV calling result of all the samples

**Usage**`icnv_res`**Format**

A list containing the calling result of CNVs:

**1st item** HMM call result without Copy number

**2nd item** exact copy number

**Value**

iCNV calling result

---

`ngs_baf`*BAF list from NGS*

---

**Description**

10 samples BAF value extracted from VCF files, location stored at ngs\_baf.pos

**Usage**`ngs_baf`**Format**

A list of ten, which each entry is the BAF value for a individual

**Value**

BAF value

---

`ngs_baf.chr`*BAF chromosome from NGS*

---

**Description**

46 samples BAF chromosome. Pre-computed using whole exome sequencing data of 46 HapMap samples.

**Usage**`ngs_baf.chr`**Format**

A list of 46, which each entry is the BAF chromosome for a individual position

**Value**

BAF chromosome

---

`ngs_baf.id`*BAF variants id from NGS*

---

**Description**

46 samples BAF ids. Pre-computed using whole exome sequencing data of 46 HapMap samples.

**Usage**`ngs_baf.id`**Format**

A list of 46, which each entry is the BAF variants id a individual position

**Value**

BAF variants id

---

`ngs_baf.nm`*BAF variants sample name from NGS*

---

**Description**

46 samples BAF names.

**Usage**`ngs_baf.nm`**Format**

A list of 46, which each entry is the sample name

**Value**

BAF variants sample names

---

`ngs_baf.pos`*BAF position list from NGS*

---

**Description**

10 samples BAF position extracted from VCF files, value stored at ngs\_baf

**Usage**`ngs_baf.pos`**Format**

A list of ten, which each entry is the BAF positions for a individual

**Value**

BAF position

---

`ngs_plr`*Normalized Poisson likelihood ratio list from NGS*

---

**Description**

10 samples PLR value from BAM calculated by CODEX, exon position stored at ngs\_plr.pos

**Usage**`ngs_plr`**Format**

A list of ten, which each entry is the PLR value for a individual, calculated from CODEX

**Value**

PLR value

---

`ngs_plr.pos`*Exon location list from NGS*

---

**Description**

10 samples exon position extracted from BED files, value stored at ngs\_plr

**Usage**`ngs_plr.pos`**Format**

A list of ten, which each entry is the Exon positions for a individual

**Value**

Exon position

---

`normObj`*Demo data pre-stored for normObj.*

---

**Description**

Pre-stored normObj data for demonstration purposes.

**Usage**

```
normObj
```

**Details**

Pre-computed using whole exome sequencing data of 46 HapMap samples.

**Value**

normObj demo data (list) pre-computed.

**Author(s)**

Zilu Zhou <zhouzil@penmedicine.upenn.edu>

**Examples**

```
Yhat <- normObjDemo$Yhat
AIC <- normObjDemo$AIC
BIC <- normObjDemo$BIC
RSS <- normObjDemo$RSS
K <- normObjDemo$K
```

---

`output_list`*Generate output list.*

---

**Description**

Generate human readable output from result calculated by iCNV\_detection function

**Usage**

```
output_list(icnv_res, sampleid = NULL, CN = 0, min_size = 0)
```

**Arguments**

icnv_res	CNV inference result. Output from iCNV_detection()
sampleid	the name of the sample, same order as the input
CN	An indicator variable with value 0,1 for whether exact copy number inferred in iCNV_detection. 0 no exact CN, 1 exact CN. Type integer. Default 0.
min_size	A integer which indicate the minimum length of the CNV you are interested in. This could remove super short CNVs due to noise. Type integer. Default 0. Recommend 1000.

**Value**

output CNV list of each individual

**Examples**

```
icnv.output <- output_list(icnv_res=icnv_res0,sampleid=sampname_qc, CN=0)
```

---

plotHMMscore	<i>Plot CNV inference score.</i>
--------------	----------------------------------

---

**Description**

Plot out CNV inference score. Each row is a sample, each column is a SNP or, exon (WES) or bin (WGS). Red color indicate score favor duplication whereas blue favor deletion.

**Usage**

```
plotHMMscore(icnv_res, h = NULL, t = NULL, title = "score plot",
  output = NULL, col = "")
```

**Arguments**

icnv_res	CNV inference result. Result from iCNV_detection() (i.e. iCNV_detection(...))
h	start position of this plot. Default Start of the whole chromosome
t	end position of this plot. Default End of the whole chromosome
title	of this plot. Character value. Type character Default "score plot"
output	generated from output_list_function. If it isn't null, only CNVs in output file will be highlighted. Default NULL
col	Specify if would like to plot in DGV color scheme ('DGV',red for deletion, blue for duplication and grey for diploid) or default color scheme (blue for deletion, red for duplicatin and and green for diploid) Type character. Default "

**Value**

void

**Examples**

```
plotHMMscore(icnv_res0,h=21000000, t=22000000, title='my favorite subject')
plotHMMscore(icnv_res0,h=21000000, t=22000000, title='my favorite subject',col='DGV')
```

---

plotindi	<i>Individual sample plot</i>
----------	-------------------------------

---

**Description**

Plot relationship between platforms and features for each individual. Only work for multi-platform inference.

**Usage**

```
plotindi(ngs_plr, snp_lrr, ngs_baf, snp_baf, ngs_plr.pos, snp_lrr.pos,
         ngs_baf.pos, snp_baf.pos, icnvres, I = numeric(), h = NULL, t = NULL)
```

**Arguments**

ngs_plr	A list of NGS intensity data. Each entry is an individual. If no NGS data, no need to specify.
snp_lrr	A list of SNP array intensity data. Each entry is an individual. If no SNP array data, no need to specify.
ngs_baf	A list of NGS BAF data. Each entry is an individual. If no NGS data, no need to specify.
snp_baf	A list of SNP array BAF data. Each entry is an individual. If no SNP array data, no need to specify.
ngs_plr.pos	A list of NGS intensity position data. Each entry is an individual with dimension= (#of bins or exons, 2(start and end position)). If no NGS data, no need to specify.
snp_lrr.pos	A list of SNP array intensity position data. Each entry is an individual with length=#of SNPs. If no SNP array data, no need to specify.
ngs_baf.pos	A list of NGS BAF position data. Each entry is an individual with length=#of BAFs. If no NGS data, no need to specify.
snp_baf.pos	A list of SNP array BAF position data. Each entry is an individual with length=#of BAFs. If no SNP array data, no need to specify.
icnvres	CNV inference result. The output from iCNV_detection()
I	Indicating the position of the individual to plot. Type integer.
h	start position of this plot. Default Start of the whole chromosome
t	end position of this plot. Default End of the whole chromosome

**Value**

void

**Examples**

```
plotindi(ngs_plr, snp_lrr, ngs_baf, snp_baf,
         ngs_plr.pos, snp_lrr.pos, ngs_baf.pos, snp_baf.pos,
         icnv_res0, I=1)
```

---

plot_intensity	<i>plot out the NGS plr or array lrr.</i>
----------------	---

---

**Description**

For quality checking purpose during intermediate steps

**Usage**

```
plot_intensity(intensity, chr = numeric())
```

**Arguments**

intensity	Specify the ngs_plr object generated by CODEX or SNP array.
chr	Specify the chromosome you want to generate. Must be of int from 1-22. Type integer

**Value**

void

**Examples**

```
chr <- 22
plot_intensity(ngs_plr, chr)
plot_intensity(snp_lrr, chr)
```

---

projname	<i>name of project</i>
----------	------------------------

---

**Description**

name of project

**Usage**

```
projname
```

**Format**

string

**Value**

name of project

---

qcObj	<i>Demo data pre-stored for qcObj.</i>
-------	--

---

**Description**

Pre-stored qcObj data for demonstration purposes.

**Usage**

qcObj

**Details**

Pre-computed using whole exome sequencing data of 46 HapMap samples.

**Value**

qcObj demo data (list) pre-computed.

**Author(s)**

Zilu Zhou <zhouzil@penmedicine.upenn.edu>

**Examples**

```
Y_qc <- qcObj$Y_qc
smpname_qc <- qcObj$smpname_qc
gc_qc <- qcObj$gc_qc
mapp_qc <- qcObj$mapp_qc
ref_qc <- qcObj$ref_qc
```

---

smpname	<i>CODEX sample name</i>
---------	--------------------------

---

**Description**

46 samples BAM names.

**Usage**

smpname

**Format**

A vector of 46, which each entry is the sample name

**Value**

CODEX sample names

---

smpname_qc	<i>QCed sample name</i>
------------	-------------------------

---

**Description**

QCed sample name

**Usage**

smpname\_qc

**Format**

string

**Value**

name of samples after QC

---

snp_baf	<i>BAF list from Array</i>
---------	----------------------------

---

**Description**

10 samples BAF value extracted from standard format files, location stored at snp\_baf.pos

**Usage**

snp\_baf

**Format**

A list of ten, which each entry is the BAF value for a individual

**Value**

BAF value

---

snp_baf.pos	<i>BAF position list from Array</i>
-------------	-------------------------------------

---

**Description**

10 samples BAF position extracted from standard format, value stored at snp\_baf

**Usage**

snp\_baf.pos

**Format**

A list of ten, which each entry is the BAF positions for a individual

**Value**

BAF position

---

snp_lrr	<i>Normalized log R ratio list from Array</i>
---------	---

---

**Description**

10 samples LRR value from standard format, SNP position stored at snp\_lrr.pos

**Usage**

snp\_lrr

**Format**

A list of ten, which each entry is the LRR value for a individual

**Value**

LRR value

---

snp_lrr.pos	<i>SNP position list from Array</i>
-------------	-------------------------------------

---

**Description**

10 samples SNP position extracted from standard format, value stored at snp\_lrr

**Usage**

snp\_lrr.pos

**Format**

A list of ten, which each entry is the SNP positions for a individual

**Value**

SNP position

# Index

- \* **BAF**,
  - icNV\_detection, 5
- \* **CNV**,
  - icNV\_detection, 5
- \* **Intensity**
  - icNV\_detection, 5
- \* **Platform**
  - icNV\_detection, 5
- \* **datasets**
  - chr, 4
  - filenm, 4
  - icnv\_res0, 8
  - ngs\_baf, 8
    - ngs\_baf.chr, 9
    - ngs\_baf.id, 9
    - ngs\_baf.nm, 10
    - ngs\_baf.pos, 10
    - ngs\_plr, 11
    - ngs\_plr.pos, 11
  - normObj, 12
  - projname, 15
  - qcObj, 16
  - sampname, 16
  - sampname\_qc, 17
  - snp\_baf, 17
    - snp\_baf.pos, 18
  - snp\_lrr, 18
    - snp\_lrr.pos, 19
- \* **integration**,
  - icNV\_detection, 5
- bambaf\_from\_vcf, 2
- bed\_generator, 3
- chr, 4
- filenm, 4
- get\_array\_input, 5
- icNV\_detection, 5
- icnv\_output\_to\_gb, 7
- icnv\_res0, 8
- ngs\_baf, 8
  - ngs\_baf.chr, 9
  - ngs\_baf.id, 9
  - ngs\_baf.nm, 10
  - ngs\_baf.pos, 10
  - ngs\_plr, 11
  - ngs\_plr.pos, 11
- normObj, 12
- output\_list, 12
- plot\_intensity, 15
- plotHMMscore, 13
- plotindi, 14
- projname, 15
- qcObj, 16
- sampname, 16
- sampname\_qc, 17
- snp\_baf, 17
  - snp\_baf.pos, 18
- snp\_lrr, 18
  - snp\_lrr.pos, 19