

---

# BenchMapL

*Release 1*

**Baudeau**

**Jun 12, 2022**



**CONTENTS:**

<b>1</b>	<b>Requirement</b>	<b>1</b>
1.1	Python Module . . . . .	1
1.2	OS version . . . . .	1
<b>2</b>	<b>API</b>	<b>3</b>
2.1	Workflow . . . . .	3
2.2	Workflow2 . . . . .	8
	<b>Python Module Index</b>	<b>9</b>
	<b>Index</b>	<b>11</b>



## **REQUIREMENT**

For the good fonctionnement of the Workflow please make sure the different package are installed

### **1.1 Python Module**

Import:

- Conda
- Snakemake
- matplotlib
- pysam
- numpy
- upsetplot

### **1.2 OS version**

- Linux



## 2.1 Workflow

`BenchPlot.common_error_gp1(results, output)`

plot upsetplot of unmapped reads groups

**Parameters**

- **results** (*list*) – list of each resu of the groups
- **output** (*str*) – name of the output

`BenchPlot.common_error_gp2(results, output)`

plot upsetplot of poorly located reads groups

**Parameters**

- **results** (*list*) – list of each resu of the groups
- **output** (*str*) – name of the output

`BenchPlot.countdiff(bamFP)`

fill the different result for the input bam

**Parameters**

**bamFP** (*pybam object*) – bam file in pybam format

**Returns**

the resu object

**Return type**

resu

`BenchPlot.find_output(key, name=False, outpath=None)`

find the output for each groups

**Parameters**

- **key** (*str*) – name of the group
- **name** (*bool or str, optional*) – name of the selected output, defaults to False
- **outpath** (*list, optional*) – list of all the possible output name, defaults to None

**Returns**

True if outpout is correct else false

**Return type**

boolean

**BenchPlot.get\_MapOrNot**(*results*)

return number of mapped and unmapped reads for each result of a groups

**Parameters**

**results** (*list*) – list of resu object

**Returns**

lists of the number of mapped and unmapped reads for each resu

**Return type**

list

**BenchPlot.get\_all\_cor**(*results*)

return list of list of each cor for each result of a groups

**Parameters**

**results** (*list*) – list of resu object

**Returns**

list of list of each cor

**Return type**

list

**BenchPlot.get\_label**(*results*)

extracts name of all the resu object of the groups

**Parameters**

**results** (*list*) – list of resu object

**Returns**

list of name

**Return type**

list

**BenchPlot.get\_nbHuman**(*results*)

return number of missaligned reads for each result of a groups

**Parameters**

**results** (*list*) – list of resu object

**Returns**

list of the number of missaligned reads for each resu

**Return type**

list

**BenchPlot.is\_human**(*read*, *resu*)

tell if read is an human reads and increm missaligned

**Parameters**

- **read** (*pybam object*) – pybam object
- **resu** (*resu object*) – resu object

**Returns**

False if missaligned else True

**Return type**

boolean



**BenchPlot**.multiple\_cor(*results*, *output*)

plot of the number of reads correctly localised by categories

**Parameters**

- **results** (*list*) – list of each resu of the groups
- **output** (*str*) – name of the output

**BenchPlot**.parsenamesimu(*text*)

return expected start position end position and number of reads mapped

**Parameters**

**text** (*str*) – name of the read

**Returns**

expected start position, expected end postion and number of mapped reads

**Return type**

tuple

**BenchPlot**.plot\_histoMU(*results*, *output*)

plot histogram of unmapped /mapped reads

**Parameters**

- **results** (*list*) – list of each resu of the groups
- **output** (*str*) – name of the output

**BenchPlot**.plot\_histoNotHuman(*results*, *output*)

plot histogram of the number of human reads mapped

**Parameters**

- **results** (*list*) – list of each resu of the groups
- **output** (*str*) – name of the output

**BenchPlot**.read\_align(*resu*, *read*, *threshold=0*)

increm resu cor for a read

**Parameters**

- **resu** (*resu*) – class resu
- **read** (*pybam object*) – read object from pybam
- **threshold** (*int*, *optional*) – specified cor number, defaults to 0

**Returns**

result of the operation true if read is increm else false

**Return type**

boolean

**class** BenchPlot.result

class for kept the different result from a bam

**addGroup1**(*add*)

add reads in group1 reads (unmapped reads)

**Parameters**

**add** (*str*) – id of the unmapped reads

**addGroup2**(*add*)

add reads in group2 reads (misslocalised reads)

**Parameters**

**add** (*str*) – id of the reads

**corpercent**(*cor*)

percent of cor reads

**Parameters**

**cor** (*int*) – number of cor reads

**Returns**

percent of correct reads

**Return type**

int

**increm\_cor**(*num=0*)

increm cor

**Parameters**

**num** (*int*, *optional*) – num of the cor, defaults to 0

**increm\_mapped**()

increment mapped

**increm\_unmapped**()

increm mapped

**mapperpercentM**()

return percent of mapped reads

**Returns**

percent of mapped reads

**Return type**

int

**mapperpercentU**()

return percent of unmapped reads

**Returns**

percent of unmapped reads

**Return type**

int

**missalign**()

increm missaligned

**setname**(*name*)

set name

**Parameters**

**name** (*str*) – name of the result

**plot\_default.do\_something**(*data\_path*, *out\_path*, *myparam*)

main function

**Parameters**

- **data\_path** (*list*) – snakemake.input
- **out\_path** (*list*) – snakemake.output
- **myparam** (*list*) – config files

`plot_default.group_input(files)`

generate a dict for each group of tools/datasets

**Parameters**

**files** (*list[string]*) – list of bam file from the different tools/dataset

**Returns**

dictionnary of the different datasets groups with each files in a list

**Return type**

dict

`plot_default.parsepath(path)`

find the name of the tool

**Parameters**

**path** (*string*) – a path

**Returns**

the string between the / and \_ character

**Return type**

string

`plot_params.do_something(data_path, out_path, myparam)`

main function

**Parameters**

- **data\_path** (*list*) – snakemake.input
- **out\_path** (*list*) – snakemake.output
- **myparam** (*list*) – config files

`plot_params.group_input(files)`

generate a dict for each group of tools/datasets/command

**Parameters**

**files** (*list[string]*) – list of bam file from the different tools/dataset

**Returns**

dictionnary of the different datasets groups with each files in a list

**Return type**

dict

`plot_params.parsepath(path)`

check the command and the tools

**Parameters**

**path** (*string*) – a path

**Returns**

the string name of the command

**Return type**

string

## 2.2 Workflow2

`select_human_nanosim.do_something(data_path, out_path)`

main function for snakemake

### Parameters

- **data\_path** (*string*) – snakemake.input
- **out\_path** (*string*) – snakemake.output

`select_human_nanosim.select_reads()`

select reads and add them in a list

### Returns

list of reads

:rtype:list

`select_human_pbsim.do_something(data_path, out_path)`

main function for snakemake

### Parameters

- **data\_path** (*string*) – snakemake.input
- **out\_path** (*string*) – snakemake.output

`select_human_pbsim.select_reads()`

select reads and add them in a list

### Returns

list of reads

:rtype:list

`convert_name.do_something()`

main function for rename all the reads in a fasta file

`convert_name.findr(x)`

filter for find the strand of a reads

### Parameters

**x** (*string*) – name of the reads

### Returns

F foward, R reverse

### Return type

string

`convert_name.openfile(file)`

open a file

### Parameters

**file** (*string*) – name of a file

### Returns

list of all the lines of the files

### Return type

list

## PYTHON MODULE INDEX

### b

BenchPlot, [3](#)

### c

convert\_name, [8](#)

### p

plot\_default, [6](#)

plot\_params, [7](#)

### s

select\_human\_nanosim, [8](#)

select\_human\_pbsim, [8](#)



## A

addGroup1() (*BenchPlot.result method*), 5  
 addGroup2() (*BenchPlot.result method*), 5

## B

BenchPlot  
     module, 3

## C

common\_error\_gp1() (*in module BenchPlot*), 3  
 common\_error\_gp2() (*in module BenchPlot*), 3  
 convert\_name  
     module, 8  
 corpercent() (*BenchPlot.result method*), 6  
 countediff() (*in module BenchPlot*), 3

## D

do\_something() (*in module convert\_name*), 8  
 do\_something() (*in module plot\_default*), 6  
 do\_something() (*in module plot\_params*), 7  
 do\_something() (*in module select\_human\_nanosim*), 8  
 do\_something() (*in module select\_human\_pbsim*), 8

## F

find\_output() (*in module BenchPlot*), 3  
 findr() (*in module convert\_name*), 8

## G

get\_all\_cor() (*in module BenchPlot*), 4  
 get\_label() (*in module BenchPlot*), 4  
 get\_MapOrNot() (*in module BenchPlot*), 3  
 get\_nbHuman() (*in module BenchPlot*), 4  
 group\_input() (*in module plot\_default*), 7  
 group\_input() (*in module plot\_params*), 7

## I

increment\_cor() (*BenchPlot.result method*), 6  
 increment\_mapped() (*BenchPlot.result method*), 6  
 increment\_unmapped() (*BenchPlot.result method*), 6  
 is\_human() (*in module BenchPlot*), 4

## M

mappercntM() (*BenchPlot.result method*), 6  
 mappercntU() (*BenchPlot.result method*), 6  
 missalign() (*BenchPlot.result method*), 6  
 module  
     BenchPlot, 3  
     convert\_name, 8  
     plot\_default, 6  
     plot\_params, 7  
     select\_human\_nanosim, 8  
     select\_human\_pbsim, 8  
 multiple\_cor() (*in module BenchPlot*), 4

## O

openfile() (*in module convert\_name*), 8

## P

parsequencesimu() (*in module BenchPlot*), 5  
 parsepath() (*in module plot\_default*), 7  
 parsepath() (*in module plot\_params*), 7  
 plot\_default  
     module, 6  
 plot\_histoMU() (*in module BenchPlot*), 5  
 plot\_histoNotHuman() (*in module BenchPlot*), 5  
 plot\_params  
     module, 7

## R

read\_align() (*in module BenchPlot*), 5  
 result (*class in BenchPlot*), 5

## S

select\_human\_nanosim  
     module, 8  
 select\_human\_pbsim  
     module, 8  
 select\_reads() (*in module select\_human\_nanosim*), 8  
 select\_reads() (*in module select\_human\_pbsim*), 8  
 setname() (*BenchPlot.result method*), 6