

# API Documentation

API Documentation

September 14, 2016

## Contents

<b>Contents</b>	<b>1</b>
<b>1 Script script-FixedVar</b>	<b>2</b>
1.1 Functions . . . . .	2
1.2 Variables . . . . .	4
1.3 Class var . . . . .	5
1.3.1 Methods . . . . .	5
1.3.2 Properties . . . . .	7
1.4 Class genome . . . . .	7
1.4.1 Methods . . . . .	8
1.4.2 Properties . . . . .	8
<b>2 Script script-XMLtoTSV</b>	<b>10</b>
2.1 Functions . . . . .	10
2.2 Variables . . . . .	10
<b>3 Script script-phyloFixedVar</b>	<b>11</b>
3.1 Functions . . . . .	11
3.2 Variables . . . . .	16
3.3 Class var . . . . .	16
3.3.1 Methods . . . . .	17
3.3.2 Properties . . . . .	19
3.4 Class genome . . . . .	19
3.4.1 Methods . . . . .	20
3.4.2 Properties . . . . .	20
3.5 Class node . . . . .	21
3.5.1 Methods . . . . .	21
3.5.2 Properties . . . . .	22

# 1 Script script-FixedVar

## 1.1 Functions

### **get\_parser()**

Parse arguments

#### **Return Value**

arguments list

*(type=parser object)*

### **printObjNucl(*objetNucl*)**

Debug fonction, print all attributes of var object

#### **Parameters**

**objetNucl:** object var to print

*(type=var object)*

### **addANN(*objetNucl*, *annotation*)**

Add annotation informations to a var object

#### **Parameters**

**objetNucl:** var object to complete

*(type=var object)*

**annotation:** for one line in a VCF file, the info column

*(type=string)*

### **readVCFfile(*file*, *listeGenomeName*)**

Read a VCF file and make a list constituted by genome objects from genome in the VCF

#### **Parameters**

**file:** VCF file

*(type=file)*

**listeGenomeName:** list

*(type=list of genomes name not to extract)*

#### **Return Value**

list of genome objects

*(type=list)*

**selectPosSensible**(*listeGenomes*)

Select all position which all genome from a list have the same allele. Not necessary for only one genome.

**Parameters**

**listeGenomes:** list of genome objects  
(*type=list*)

**Return Value**

dictionnary with the position as key and the nucleic allele for value  
(*type=dictionnary*)

**genomeName\_to\_genomeObjet**(*GenomesNames, GenomesObjet*)

From a list of genomes name, this fuction find the genome objects associated to its name.

**Parameters**

**GenomesNames:** list of genome names  
(*type=list*)  
**GenomesObjet:** list of genome objects  
(*type=list*)

**Return Value**

list of genome objects associated to the genome names of the GenomesNames list  
(*type=list*)

**createDicoPos**(*listeGenomesObjets*)

Make a dictionnary with all variants position and the all the different alleles associated

**Parameters**

**listeGenomesObjets:** list of genome objects  
(*type=list*)

**Return Value**

dictionnary with position as key and the list of the different alleles at this position for value  
(*type=dictionnary*)

**selectPosSpecifique**(*dicoPosSensible, listeGenomesObjets*)

Select the alleles existing in the dictionary and not present for the genomes of the list (specificity filter)

**Parameters**

**dicoPosSensible:** dictionary with positions as key and the allele corresponding as value  
(*type=dictionnary*)

**listeGenomesObjets:** list of genome objects  
(*type=list*)

**Return Value**

dictionary with position specifique as key and the allele corresponding as value  
(*type=dictionnary*)

**fileToList**(*inputFile*)

Extract genomes name from a file with 1 name per line

**Parameters**

**inputFile:** input file name  
(*type=string*)

**Return Value**

list of genomes name  
(*type=list*)

**makeXML**(*GenomesObjet, genome\_to\_process, genome\_to\_compare*)

Make a XML objet from a the comparaison groups list and the lists of genomes to analyse and genomes to compare

**Parameters**

**GenomesObjet:** list of genome objects  
(*type=list*)

**genome\_to\_process:** genomes name to analyse  
(*type=list*)

**genome\_to\_compare:** genomes name to use for the comparison  
(*type=list*)

**Return Value**

root of the XML tree  
(*type=etree*)

**main**()

## 1.2 Variables

Name	Description
<code>--doc--</code>	<b>Value:</b> ...

### 1.3 Class var

object   
**script-FixedVar.var**

Create var object to stock variant information

#### 1.3.1 Methods

<b>__init__(self)</b>	
Initialize the var class	
<b>Parameters</b>	
<b>type:</b>	type of mutation ('SNP' or 'INDEL') ( <i>type=string</i> )
<b>pos:</b>	variant position regarding the reference genome ( <i>type=integer</i> )
<b>var:</b>	nucleic variation ( <i>type=string</i> )
<b>homoplasmy:</b>	homoplastic variant ('Yes' or 'No') ( <i>type=string</i> )
<b>region:</b>	type of genetic region ('intergenic' or 'intragenic') ( <i>type=string</i> )
<b>impact:</b>	variant impact ( <i>type=string</i> )
<b>geneID:</b>	for a intragenic region, the gene ID impacted by the variant ( <i>type=string @@param geneName : for a intragenic region, the gene name impacted by the variant</i> )
<b>transcritID:</b>	for a intragenic region, the transcrit ID impacted by the variant ( <i>type=string</i> )
Overrides: object.__init__	

  

<b>setType(self, type)</b>	
Set the type of mutation ('SNP' or 'INDEL')	
<b>Parameters</b>	
<b>type:</b>	type of mutation ('SNP' or 'INDEL') ( <i>type=string</i> )

---

**setPos**(*self*, *pos*)

---

Set the variant position regarding the reference genome

**Parameters**

**pos**: variant position regarding the reference genome  
(*type=integer*)

---

**setVar**(*self*, *var*)

---

Set the nucleic variation

**Parameters**

**var**: nucleic variation  
(*type=string*)

---

**setHomoplasy**(*self*, *homo*)

---

Set if the variant is homoplastic or not ('Yes' or 'No')

**Parameters**

**homo**: homoplastic variant ('Yes' or 'No')  
(*type=string*)

---

**setRegion**(*self*, *reg*)

---

Set the type of genetic region ('intergenic' or 'intragenic')

**Parameters**

**reg**: type of genetic region ('intergenic' or 'intragenic')  
(*type=string*)

---

**setImpact**(*self*, *imp*)

---

Set the genetic impact of the variant

**Parameters**

**imp**: variant impact  
(*type=string*)

---

**setGeneID**(*self*, *id*)

---

Set the gene ID impacted by the variant

**Parameters**

**id**: gene ID impacted by the variant  
(*type=string*)

<b>setGeneName</b> ( <i>self</i> , <i>name</i> )
--

Set the gene name impacted by the variant
---

<b>Parameters</b>
-------------------

<b>name:</b> gene name impacted by the variant
--

( <i>type=string</i> )
------------------------

<b>setTranscritID</b> ( <i>self</i> , <i>transcritID</i> )
--

Set the transcrit ID impacted by the variant
--

<b>Parameters</b>
-------------------

<b>transcritID:</b> transcrit ID impacted by the variant
--

( <i>type=string</i> )
------------------------

### *Inherited from object*

`--delattr--()`, `--format--()`, `--getattr--()`, `--hash--()`, `--new--()`, `--reduce--()`, `--reduce_ex--()`,  
`--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

### 1.3.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

## 1.4 Class genome

object └─ **script-FixedVar.genome**

Create genome object to stock all of its variants informations

### 1.4.1 Methods

**`--init--(self)`**

Initialize the genome class

**Parameters**

**name:** name of the genome

*(type=string)*

**variants:** list of var objects assigned to the genome

*(type=list)*

Overrides: object.\_\_init\_\_

**`setName(self, name)`**

Set the name of the genome

**Parameters**

**name:** name of the genome

*(type=string)*

**`setVar(self, variants)`**

Set a list of var objects assigned to the genome

**Parameters**

**variants:** list of var objects assigned to the genome

*(type=list)*

**`addVariant(self, variant)`**

Add a new var object in the genome variants list

**Parameters**

**variants:** list of var objects assigned to the genome

*(type=list)*

### *Inherited from object*

`--delattr--()`, `--format--()`, `--getattr--()`, `--hash--()`, `--new--()`, `--reduce--()`, `--reduce_ex--()`,  
`--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

### 1.4.2 Properties



Name	Description
Inherited from object __class__	

## 2 Script script-XMLtoTSV

### 2.1 Functions

**get\_parser()**

Parse arguments

**Return Value**

arguments list

*(type=parser object)*

**node\_to\_tsv(*node*, *output*)**

Write all node informations in a TSV file

**Parameters**

**node:** node of interest to process

*(type=etree)*

**output:** name of the output TSV file

*(type=string)*

**main()**

### 2.2 Variables

Name	Description
__doc__	<b>Value:</b> ...

### 3 Script script-phyloFixedVar

#### 3.1 Functions

##### **get\_parser()**

Parse arguments

##### **Return Value**

arguments list

*(type=parser object)*

##### **printObjNucl(*objetNucl*)**

Debug function, print all attributes of var object

##### **Parameters**

**objetNucl**: object var to print

*(type=var object)*

##### **createNode()**

Create a new empty node object

##### **Return Value**

empty node object

*(type=node object)*

##### **readTree(*file*)**

Read a newick file and stock informations in a node object

##### **Parameters**

**file**: newick file name

*(type=string)*

##### **Return Value**

root of the phylogeny tree

*(type=node object)*

**allNodesLeafs**(*node*, *liste*)

For a given node, stock all its descendants name in a list (recursive function)

**Parameters**

- node:** node object which descendants names will be found  
*(type=node object)*
- liste:** empty liste that will contain genomes name  
*(type=list)*

**groupCompare**(*node*, *dico*)

Complete a dictionnary which will contain a id as key and a list as value. The list is composed by 2 element :

- the first element is a list of all descendants genomes name of a node object,
- the second element is a list of all cousin(s) genomes names. \

This function is applied on all sons of the given node.

**Parameters**

- node:** kin node object to process  
*(type=node object)*
- dico:** dictionnary to complete  
*(type=dictionnary)*

**AllgroupCompare**(*node*, *dico*)

Launch the groupCompare funtion on all sons of a given node

**Parameters**

- node:** kin node object to process  
*(type=node object)*
- dico:** dictionnary to complete  
*(type=dictionnary)*

**addANN**(*objetNucl*, *annotation*)

Add annotation informations to a var object

**Parameters**

**objetNucl:** var object to complete  
(*type=var object*)

**annotation:** for one line in a VCF file, the info column  
(*type=string*)

**readVCFfile**(*file*)

Read a VCF file and make a list constituted by genome objects from genome in the VCF

**Parameters**

**file:** VCF file  
(*type=file*)

**Return Value**

list of genome objects  
(*type=list*)

**selectPosSensible**(*listeGenomes*)

Select all position which all genome from a list have the same allele. Not necessary for only one genome.

**Parameters**

**listeGenomes:** list of genome objects  
(*type=list*)

**Return Value**

dictionnary with the position as key and the nucleic allele for value  
(*type=dictionnary*)

**genomeName\_to\_genomeObjet**(*GenomesNames*, *GenomesObjet*)

From a list of genomes name, this fuction find the genome objects associated to its name.

**Parameters**

**GenomesNames:** list of genome names  
(*type=list*)

**GenomesObjet:** list of genome objects  
(*type=list*)

**Return Value**

list of genome objects associated to the genome names of the  
GenomesNames list  
(*type=list*)

**createDicoPos**(*listeGenomesObjets*)

Make a dictionnary with all variants position and the all the different alleles associated

**Parameters**

**listeGenomesObjets:** list of genome objects  
(*type=list*)

**Return Value**

dictionnary with position as key and the list of the different alleles at  
this position for value  
(*type=dictionnary*)

**selectPosSpecifique**(*dicoPosSensible*, *listeGenomesObjets*)

Select the alleles existing in the dictionnary and not present for the genomes of the list (specificity filter)

**Parameters**

**dicoPosSensible:** dictionnary with positions as key and the allele corresponding as value

(*type=dictionnary*)

**listeGenomesObjets:** list of genome objects

(*type=list*)

**Return Value**

dictionnary with position specifique as key and the allele corresponding as value

(*type=dictionnary*)

**allOtherGenomes**(*Genomes\_to\_exclude*, *allGenomesObjets*)

Make a liste of genome name from a list of genome objects without the genomes name given in argument

**Parameters**

**Genomes\_to\_exclude:** list of genomes name to exclude

(*type=list*)

**allGenomesObjets:** list of genome objects

(*type=list*)

**Return Value**

a list of genomes name without the given name

(*type=list*)

**addNodeNameNewick**(*filename*, *outputFilename*)

Make a new newick fill with a label for each node. The label is created like this :

- N + the number of the node

CX-CY with X and Y the number of combination

**Parameters**

**filename:** newick input file name  
(*type=string*)

**outputFilename:** newick output file name  
(*type=string*)

**makeXML**(*dico*, *GenomesObjet*)

Make a XML objet from a the comparaison groups list

**Parameters**

**dico:** dictionnary obtained with the AllgroupCompare function  
(*type=string*)

**GenomesObjet:** list of genome objects  
(*type=list*)

**Return Value**

root of the XML tree  
(*type=etree*)

**main**()

### 3.2 Variables

Name	Description
<code>--doc--</code>	<b>Value:</b> ...

### 3.3 Class var

object └─ **script-phyloFixedVar.var**



Create var object to stock variant information

### 3.3.1 Methods

**`__init__(self)`**

Initialize the var class

**Parameters**

**type:** type of mutation ('SNP' or 'INDEL')  
(*type=string*)

**pos:** variant position regarding the reference genome  
(*type=integer*)

**var:** nucleic variation  
(*type=string*)

**homoplasy:** homoplastic variant ('Yes' or 'No')  
(*type=string*)

**region:** type of genetic region ('intergenic' or 'intragenic')  
(*type=string*)

**impact:** variant impact  
(*type=string*)

**geneID:** for a intragenic region, the gene ID impacted by the variant  
(*type=string @@param geneName : for a intragenic region, the gene name impacted by the variant*)

**transcritID:** for a intragenic region, the transcrit ID impacted by the variant  
(*type=string*)

Overrides: object.\_\_init\_\_

**`setType(self, type)`**

Set the type of mutation ('SNP' or 'INDEL')

**Parameters**

**type:** type of mutation ('SNP' or 'INDEL')  
(*type=string*)

**setPos**(*self*, *pos*)

Set the variant position regarding the reference genome

**Parameters**

**pos**: variant position regarding the reference genome  
(*type=integer*)

**setVar**(*self*, *var*)

Set the nucleic variation

**Parameters**

**var**: nucleic variation  
(*type=string*)

**setHomoplasy**(*self*, *homo*)

Set if the variant is homoplastic or not ('Yes' or 'No')

**Parameters**

**homo**: homoplastic variant ('Yes' or 'No')  
(*type=string*)

**setRegion**(*self*, *reg*)

Set the type of genetic region ('intergenic' or 'intragenic')

**Parameters**

**reg**: type of genetic region ('intergenic' or 'intragenic')  
(*type=string*)

**setImpact**(*self*, *imp*)

Set the genetic impact of the variant

**Parameters**

**imp**: variant impact  
(*type=string*)

**setGeneID**(*self*, *id*)

Set the gene ID impacted by the variant

**Parameters**

**id**: gene ID impacted by the variant  
(*type=string*)

<b>setGeneName</b> ( <i>self</i> , <i>name</i> )
--

Set the gene name impacted by the variant
---

<b>Parameters</b>
-------------------

<b>name</b> : gene name impacted by the variant <i>(type=string)</i>
---

<b>setTranscritID</b> ( <i>self</i> , <i>transcritID</i> )
--

Set the transcrit ID impacted by the variant
--

<b>Parameters</b>
-------------------

<b>transcritID</b> : transcrit ID impacted by the variant <i>(type=string)</i>
---

### *Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

### 3.3.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 3.4 Class genome

object └─ **script-phyloFixedVar.genome**

Create genome object to stock all of its variants informations

### 3.4.1 Methods

**`--init--(self)`**

Initialize the genome class

**Parameters**

**name:** name of the genome

(*type=string*)

**variants:** list of var objects assigned to the genome

(*type=list*)

Overrides: object.\_\_init\_\_

**`setName(self, name)`**

Set the name of the genome

**Parameters**

**name:** name of the genome

(*type=string*)

**`setVar(self, variants)`**

Set a list of var objects assigned to the genome

**Parameters**

**variants:** list of var objects assigned to the genome

(*type=list*)

**`addVariant(self, variant)`**

Add a new var object in the genome variants list

**Parameters**

**variants:** list of var objects assigned to the genome

(*type=list*)

### *Inherited from object*

`--delattr--()`, `--format--()`, `--getattr--()`, `--hash--()`, `--new--()`, `--reduce--()`, `--reduce_ex--()`,  
`--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

### 3.4.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

### 3.5 Class node

object └─ **script-phyloFixedVar.node**

Create node object to stock the phylogeny tree

#### 3.5.1 Methods

<b><code>--init--(self)</code></b>
Initialize the node class
<b>Parameters</b>
<b>files:</b> list of sons nodes objects ( <i>type=list</i> )
<b>pere:</b> father node object ( <i>type=node object</i> )
<b>val:</b> genome name if the node is a leaf ( <i>type=string</i> )
<b>name:</b> name of the genome ( <i>type=string</i> )
<b>variants:</b> list of var objects assigned to the genome ( <i>type=list</i> )
Overrides: object. <code>--init--</code>

  

<b><code>nbFils(self)</code></b>
Compute number of sons
<b>Return Value</b>
number of sons ( <i>type=integer</i> )

**isLeaf(*self*)**

Check if the node is a leaf

**Return Value**

True if the node is a leaf, False otherwise

(*type=boolean*)

**setPere(*self*, *node*)**

Set the father node of the current node object

**Parameters**

**node**: father node object

(*type=node object*)

**setFils(*self*, *node*)**

Add a new son to the current node object

**Parameters**

**node**: son node object

(*type=node object*)

**setVal(*self*, *val*)**

Set the genome name of the node

**Parameters**

**val**: genome name

(*type=string*)

**listLeaf(*self*, *liste*)**

Give all genomes name of all sons node objects (recursive function)

**Parameters**

**liste**: empty list which will contain genomes name

(*type=list*)

***Inherited from object***

`--delattr__()`, `--format__()`, `--getattr__()`, `--hash__()`, `--new__()`, `--reduce__()`, `--reduce_ex__()`,  
`--repr__()`, `--setattr__()`, `--sizeof__()`, `--str__()`, `--subclasshook__()`

**3.5.2 Properties**

Name	Description
<i>Inherited from object</i> __class__	

## Index

- script-FixedVar (*script*), 2–9
  - script-FixedVar.addANN (*function*), 2
  - script-FixedVar.createDicoPos (*function*), 3
  - script-FixedVar.fileToList (*function*), 4
  - script-FixedVar.genome (*class*), 7–9
    - script-FixedVar.genome.addVariant (*method*), 8
    - script-FixedVar.genome.setName (*method*), 8
    - script-FixedVar.genome.setVar (*method*), 8
  - script-FixedVar.genomeName\_to\_genomeObjet (*function*), 3
  - script-FixedVar.get\_parser (*function*), 2
  - script-FixedVar.main (*function*), 4
  - script-FixedVar.makeXML (*function*), 4
  - script-FixedVar.printObjNucl (*function*), 2
  - script-FixedVar.readVCFfile (*function*), 2
  - script-FixedVar.selectPosSensible (*function*), 2
  - script-FixedVar.selectPosSpecifique (*function*), 3
  - script-FixedVar.var (*class*), 5–7
    - script-FixedVar.var.setGeneID (*method*), 6
    - script-FixedVar.var.setGeneName (*method*), 6
    - script-FixedVar.var.setHomoplasy (*method*), 6
    - script-FixedVar.var.setImpact (*method*), 6
    - script-FixedVar.var.setPos (*method*), 5
    - script-FixedVar.var.setRegion (*method*), 6
    - script-FixedVar.var.setTranscritID (*method*), 7
    - script-FixedVar.var.setType (*method*), 5
    - script-FixedVar.var.setVar (*method*), 6
- script-phyloFixedVar (*script*), 11–23
  - script-phyloFixedVar.addANN (*function*), 12
  - script-phyloFixedVar.addNodeNameNewick (*function*), 15
  - script-phyloFixedVar.AllgroupCompare (*function*), 12
  - script-phyloFixedVar.allNodesLeafs (*function*), 11
  - script-phyloFixedVar.allOtherGenomes (*function*), 15
  - script-phyloFixedVar.createDicoPos (*function*), 14
  - script-phyloFixedVar.createNode (*function*), 11
  - script-phyloFixedVar.genome (*class*), 19–21
    - script-phyloFixedVar.genome.addVariant (*method*), 20
    - script-phyloFixedVar.genome.setName (*method*), 20
    - script-phyloFixedVar.genome.setVar (*method*), 20
  - script-phyloFixedVar.genomeName\_to\_genomeObjet (*function*), 13
  - script-phyloFixedVar.get\_parser (*function*), 11
  - script-phyloFixedVar.groupCompare (*function*), 12
  - script-phyloFixedVar.main (*function*), 16
  - script-phyloFixedVar.makeXML (*function*), 16
  - script-phyloFixedVar.node (*class*), 21–23
    - script-phyloFixedVar.node.isLeaf (*method*), 21
    - script-phyloFixedVar.node.listLeaf (*method*), 22
    - script-phyloFixedVar.node.nbFils (*method*), 21
    - script-phyloFixedVar.node.setFils (*method*), 22
    - script-phyloFixedVar.node.setPere (*method*), 22
    - script-phyloFixedVar.node.setVal (*method*), 22



- 22
- script-phyloFixedVar.printObjNucl (*function*), 11
- script-phyloFixedVar.readTree (*function*), 11
- script-phyloFixedVar.readVCFfile (*function*), 13
- script-phyloFixedVar.selectPosSensible (*function*), 13
- script-phyloFixedVar.selectPosSpecifique (*function*), 14
- script-phyloFixedVar.var (*class*), 16–19
  - script-phyloFixedVar.var.setGeneID (*method*), 18
  - script-phyloFixedVar.var.setGeneName (*method*), 18
  - script-phyloFixedVar.var.setHomoplasy (*method*), 18
  - script-phyloFixedVar.var.setImpact (*method*), 18
  - script-phyloFixedVar.var.setPos (*method*), 17
  - script-phyloFixedVar.var.setRegion (*method*), 18
  - script-phyloFixedVar.var.setTranscritID (*method*), 19
  - script-phyloFixedVar.var.setType (*method*), 17
  - script-phyloFixedVar.var.setVar (*method*), 18
- script-XMLtoTSV (*script*), 10
  - script-XMLtoTSV.get\_parser (*function*), 10
  - script-XMLtoTSV.main (*function*), 10
  - script-XMLtoTSV.node\_to\_tsv (*function*), 10