

# Package ‘PopGenome’

March 21, 2012

**Type** Package

**Title** Population genetic analysis

**Version** 1.1

**Date** 2011-03-17

**Author** Bastian Pfeifer

**Maintainer** Bastian Pfeifer <Bastian.Pfeifer@uni-duesseldorf.de>

**Depends** R (>= 1.8.0), methods

**Description** PopGenome is an R-library for Population Genetic Analysis

**License** GPL-2

**LazyLoad** yes

**Repository** CRAN

**Date/Publication** 2012-03-21 12:52:46

## R topics documented:

detail.stats-methods . . . . .	2
F_ST.stats-methods . . . . .	3
GENOME-class . . . . .	5
get.status-methods . . . . .	8
linkage.stats-methods . . . . .	9
MKT-methods . . . . .	10
MS . . . . .	11
neutrality.stats-methods . . . . .	13
PopGenome . . . . .	15
readData . . . . .	16
set.populations-methods . . . . .	18
show.slots-methods . . . . .	19
sliding.window.transform-methods . . . . .	19
test.params-class . . . . .	20

**Index**[22](#)

---

detail.stats-methods    *Several statistics*

---

**Description**

This generic function calculates some mixed statistics.

**Usage**

```
## S4 method for signature 'GENOME'  
detail.stats(object,new.populations=FALSE,subsites=FALSE,biallelic.structure=TRUE,mismatch.distribu  
## S4 method for signature 'GENOME'  
get.detail(object)
```

**Arguments**

object            an object of class "GENOME"  
new.populations    list of populations. default:FALSE  
subsites           "syn" for synonymous sites and "nonsyn" for nonsynonymous sites. default:FALSE  
biallelic.structure    fixed and shared polymorphisms (stored in GENOME.class@region.stats). default:TRUE  
mismatch.distribution    statistics based on mismatch distribution. default:TRUE

**Value**

returned value is an modified object of class "GENOME"

---

Following Slots will be modified in the "GENOME" object

---

MDSD

MDG1

MDG2

region.stats    the slot biallelic.structure will be filled

**Examples**

```
# GENOME.class <- readData("\home\Alignments")
# GENOME.class
# GENOME.class <- detail.stats(GENOME.class)
# GENOME.class <- detail.stats(GENOME.class,list(1:7,8:12))
# show the result:
# mismatch.values <- get.detail(GENOME.class)
# GENOME.class@region.stats@biallelic.structure
# GENOME.class@region.stats@biallelic.structure[[1]]
```

---

F\_ST.stats-methods      *Fixation Index*

---

**Description**

A generic function to calculate some F-statistics and nucleotide/haplotype diversities.

**Usage**

```
## S4 method for signature 'GENOME'
F_ST.stats(object,new.populations=FALSE,subsites=FALSE,
detail=TRUE,mode="ALL")
## S4 method for signature 'GENOME'
get.diversity(object,between=FALSE)
## S4 method for signature 'GENOME'
get.F_ST(object,mode=FALSE,pairwise=FALSE)
```

**Arguments**

object	An object of class "GENOME"
new.populations	list of populations. default:FALSE
subsites	synonymous positions (subsites="syn"), nonsynonymous positions (subsites="nonsyn")
detail	detail statistics. Note: slower!
between	TRUE: show between diversities. FALSE: show within diversities
mode	mode="haplotype" or mode="nucleotide"
pairwise	show pairwise comparisons. default:FALSE

**Value**

	Slot	Reference	Description
1.	haplotype.F_ST	[1]	Fixation Index based on haplotype frequencies
2.	nucleotide.F_ST	[1]	Fixation Index based on minor.allele frequencies
3.	Nei.G_ST	[2]	Nei's Fixation Index
4.	Hudson.G_ST	[3]	
5.	Hudson.H_ST	[3]	
6.	Hudson.K_ST	[3]	
7.	nuc.diversity.within	[1]	Nucleotide diversity (within the population)
8.	hap.diversity.within	[1]	Haplotype diversity (within the population)
9.	Pi	[4]	Diversity from Nei (within the population)
10.	hap.F_ST.vs.all	[1]	Fixation Index for each population against the rest (haplotype)
11.	nuc.F_ST.vs.all	[1]	Fixation Index for each population against the rest (nucleotide)
12.	hap.diversity.between	[1]	Haplotype diversities between populations
13.	nuc.diversity.between	[1]	Nucleotide diversities between populations
14.	nuc.F_ST.pairwise	[1]	Fixation Index for every pair of population (nucleotide)
15.	hap.F_ST.pairwise	[1]	Fixation Index for every pair of population (haplotype)
16.	Nei.G_ST.pairwise	[2]	Fixation Index for every pair of population (Nei)
17.	region.stats		an object of class "region.stats" for detail statistics

**References**

- [1] Hudson, R. R., M. Slatkin, and W.P. Maddison (1992). *Estimating of levels of gene flow from DNA sequence data*. *Genetics* 13(2),583-589
- [2] Nei, M. (1973). *Analysis of gene diversity in subdivided populations*. *Proc.Natl. Acad. Sci. USA* 70: 3321-3323
- [3] Hudson, R. R., Boos, D.D. and N. L. Kaplan (1992). *A statistical test for detecting population subdivision*. *Mol. Biol. Evol.* 9: 138-151.
- [4] Nei, M. (1987). *Molecular Evolutionary Genetics*. Columbia Univ. Press, New York.

**Examples**

```
# GENOME.class <- readData("\home\Alignments")
# GENOME.class
# GENOME.class <- FST_stats(GENOME.class)
# GENOME.class <- FST_stats(GENOME.class,list(1:4,5:10),subsites="syn")
# GENOME.class <- FST_stats(GENOME.class,list(c("seq1","seq5","seq3"),c("seq2","seq8")))
# show the result:
# get.F_ST(GENOME.class)
# get.F_ST(GENOME.class, pairwise=TRUE)
```

```
# get.diversity(GENOME.class, between=TRUE)
# GENOME.class@Pi --> population specific view
# GENOME.class@region.stats
```

---

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

---

## Description

A class where all values are stored

## Objects from the Class

coming soon ...

## Slots

```
basepath: Object of class "character" ~~
project: Object of class "character" ~~
populations: Object of class "list" ~~
poppairs: Object of class "vector" ~~
outgroup: Object of class "vector" ~~
region.names: Object of class "character" ~~
genelength: Object of class "numeric" ~~
n.sites: Object of class "numeric" ~~
n.biallelic.sites: Object of class "numeric" ~~
n.gaps: Object of class "numeric" ~~
n.unknowns: Object of class "numeric" ~~
n.valid.sites: Object of class "numeric" ~~
n.polyallelic.sites: Object of class "numeric" ~~
trans.transv.ratio: Object of class "numeric" ~~
Pop_Neutrality: Object of class "list" ~~
Pop_FSTN: Object of class "list" ~~
Pop_FSTH: Object of class "list" ~~
Pop_Linkage: Object of class "list" ~~
Pop_Slide: Object of class "list" ~~
Pop_MK: Object of class "list" ~~
Pop_Detail: Object of class "list" ~~
FSTNLISTE: Object of class "list" ~~
nucleotide.F_ST: Object of class "matrix" ~~
```

nucleotide.F\_ST2: Object of class "matrix" ~~  
nuc.diversity.between: Object of class "matrix" ~~  
nuc.diversity.within: Object of class "matrix" ~~  
nuc.F\_ST.pairwise: Object of class "matrix" ~~  
nuc.F\_ST.vs.all: Object of class "matrix" ~~  
n.haplotypes: Object of class "matrix" ~~  
hap.diversity.within: Object of class "matrix" ~~  
hap.diversity.between: Object of class "matrix" ~~  
Pi: Object of class "matrix" ~~  
PIA\_nei: Object of class "matrix" ~~  
haplotype.counts: Object of class "matrix" ~~  
haplotype.F\_ST: Object of class "matrix" ~~  
hap.F\_ST.pairwise: Object of class "matrix" ~~  
Nei.G\_ST.pairwise: Object of class "matrix" ~~  
hap.F\_ST.vs.all: Object of class "matrix" ~~  
Nei.G\_ST: Object of class "matrix" ~~  
Hudson.G\_ST: Object of class "matrix" ~~  
Hudson.H\_ST: Object of class "matrix" ~~  
Hudson.K\_ST: Object of class "matrix" ~~  
Hudson.Snn: Object of class "matrix" ~~  
hap.pair.F\_ST: Object of class "matrix" ~~  
MKT: Object of class "matrix" ~~  
Tajima.D: Object of class "matrix" ~~  
SLIDE: Object of class "matrix" ~~  
Fay.Wu.H: Object of class "matrix" ~~  
Zeng.E: Object of class "matrix" ~~  
theta\_Tajima: Object of class "matrix" ~~  
theta\_Watterson: Object of class "matrix" ~~  
theta\_Fu.Li: Object of class "matrix" ~~  
theta\_Achaz.Watterson: Object of class "matrix" ~~  
theta\_Achaz.Tajima: Object of class "matrix" ~~  
theta\_Fay.Wu: Object of class "matrix" ~~  
theta\_Zeng: Object of class "matrix" ~~  
Fu.Li.F: Object of class "matrix" ~~  
Fu.Li.D: Object of class "matrix" ~~  
Yach: Object of class "matrix" ~~  
n.segregating.sites: Object of class "matrix" ~~

Rozas.R\_2: Object of class "matrix" ~~  
 Fu.F\_S: Object of class "matrix" ~~  
 Strobeck.S: Object of class "matrix" ~~  
 Kelly.Z\_nS: Object of class "matrix" ~~  
 Rozas.ZZ: Object of class "matrix" ~~  
 Rozas.ZA: Object of class "matrix" ~~  
 Wall.B: Object of class "matrix" ~~  
 Wall.Q: Object of class "matrix" ~~  
 MDSD: Object of class "matrix" ~~  
 MDG1: Object of class "matrix" ~~  
 MDG2: Object of class "matrix" ~~  
 genes: Object of class "list" ~~  
 region.data: Object of class "region.data" ~~  
 region.stats: Object of class "region.stats" ~~

## Methods

**detail.stats** signature(object = "GENOME"): Several Statistics  
**F\_ST.stats.2** signature(object = "GENOME"): ...  
**F\_ST.stats** signature(object = "GENOME"): Fixation Index  
**getBayes** signature(object = "GENOME"): ...  
**get.detail** signature(object = "GENOME"): ...  
**get.diversity** signature(object = "GENOME"): ...  
**get.F\_ST** signature(object = "GENOME"): ...  
**get.linkage** signature(object = "GENOME"): ...  
**get.MKT** signature(object = "GENOME"): ...  
**getMS** signature(object = "GENOME"): ...  
**get.neutrality** signature(object = "GENOME"): ...  
**get.status** signature(object = "GENOME"): ...  
**get.sum.data** signature(object = "GENOME"): ...  
**linkage.stats** signature(object = "GENOME"): Linkage Disequilibrium  
**MKT** signature(object = "GENOME"): MKT Test  
**neutrality.stats** signature(object = "GENOME"): Neutrality Statistics  
**popFSTN** signature(object = "GENOME"): ...  
**set.populations** signature(object = "GENOME"): Define the populations  
**show** signature(object = "GENOME"): ...  
**show.slots** signature(object = "GENOME"): ...  
**sliding.window.transform** signature(object = "GENOME"): Sliding Window  
**usage** signature(object = "GENOME"): ...

**Note**

test test

**Author(s)**

Bastian Pfeifer

**References**

~put references to the literature/web site here ~

**See Also**

bla bla

**Examples**

```
showClass("GENOME")
```

---

*get.status-methods*      *State of Calculations*

---

**Description**

coming soon ...

**Methods**

**object** = "GENOME" coming soon ...

**Examples**

```
# get.status(GENOME.class)
```

---

 linkage.stats-methods *Linkage Disequilibrium*


---

## Description

A generic function to calculate some linkage disequilibrium statistics.

## Usage

```
## S4 method for signature 'GENOME'
linkage.stats(object,new.populations=FALSE,subsites=FALSE,detail=FALSE)
## S4 method for signature 'GENOME'
get.linkage(object)
```

## Arguments

object	An object of class "GENOME"
new.populations	list of populations. default=FALSE
subsites	synonymous positions (subsites="syn"), nonsynonymous positions (subsites="nonsyn")
detail	if you want to calculate some detail statistics. slower! default:FALSE

## Value

Returned value is an modified object of class "GENOME"

---

Following slots will be modified in the "GENOME" object

---

	Slot	Reference	Description
1.	Wall.B	[2]	Wall \$B\$ statistic (only adjacent positions are considered)
2.	Wall.Q	[2]	Wall \$Q\$ statistic (only adjacent positions are considered)
3.	Kelly.Z_nS	[3]	Kelly \$Z_nS\$ statistic (if detail==TRUE)
4.	Rozas.ZA	[1]	Rozas \$ZA\$ statistic (adjacent positions, if detail==TRUE)
5.	Rozas.ZZ	[1]	Rozas \$ZZ\$ statistic (\$ZZ=ZA-Z_nS\$, if detail==TRUE)

## References

[1] Rozas, J., M.Gullaud, G.Blandin, and M.Aguade(2001). *DNA variation at the rp49 gene region of Drosophila simulans: evolutionary inferences from an unusual haplotype structure*. Genetics 158(3),1147-1155

[2] Wall, J.(1999). *Recombination and the power of statistical tests of neutrality*. Genet Res 74, 65-79

[3] Kelly,J.K. (1997). *A test of neutrality based on interlocus associations*. Genetics 146: 1197-1206

## Examples

```
# GENOME.class <- readData("\home\Alignments")
# GENOME.class
# GENOME.class <- linkage.stats(GENOME.class)
# GENOME.class <- linkage.stats(GENOME.class,list(1:4,5:10),subsites="syn")
# GENOME.class <- linkage.stats(GENOME.class,list(c("seq1","seq5","seq3"),
# c("seq2","seq8")))
# GENOME.class <- linkage.stats(GENOME.class,detail=TRUE)
# show the result:
# get.linkage(GENOME.class)
# GENOME.class@Wall.B --> population specific view
# GENOME.class@region.stats
```

---

MKT-methods

*McDonald & Kreitman Test*


---

## Description

This generic function calculates an approximate version of the McDonald & Kreitman Test.

## Usage

```
## S4 method for signature 'GENOME'
MKT(object,new.populations=FALSE)
## S4 method for signature 'GENOME'
get.MKT(object)
```

## Arguments

object            an object of class "GENOME"

new.populations   list of populations. default:FALSE

**Details**

This function assumes that in population genetic analysis the probability of two variants in one codon is very small. Due to this only single nucleotide polymorphisms (SNPs) are examined. When there was no gff-file specified, an alignment in the right reading frame is expected.

**Value**

Returned value is an modified object of class "GENOME"

---

Following slots will be modified in the "GENOME" object

---

MKT                    a matrix which includes following values:

	Columns	Description
1.	P_nonsyn	nonsynonymous sites
2.	P_syn	synonymous sites
3.	D_nonsyn	fixed nonsynonymous sites
4.	D_syn	fixed synonymous sites
5.	neutrality.index	$\$(P\_nonsyn/P\_syn)/(D\_nonsyn/D\_syn)\$$
6.	alpha	1-neutrality.index

**References**

McDonald, J. H.; Kreitman, M. (1991). *Adaptive protein evolution at the Adh locus in Drosophila*. Nature 351 (6328): 652-654

**Examples**

```
# GENOME.class <- readData("\home\Alignments")
# GENOME.class
# GENOME.class <- MKT(GENOME.class)
# GENOME.class <- MKT(GENOME.class,list(1:7,8:12))
# show the result:
# get.MKT(GENOME.class)
```

**Description**

This function uses Hudson's MS to compare simulated data with the observed data.

**Usage**

```
MS(GENO,niter=10,thetaID="user",params=FALSE,detail=FALSE,
  neutrality=FALSE,linkage=FALSE,F_ST=FALSE)
```

**Arguments**

GENO	an object of class "GENOME"
niter	number of samples per loci
thetaID	"Tajima","Watterson" or "user". default:"user"
neutrality	Calculate neutrality tests. default=FALSE
linkage	Calculate linkage disequilibrium. default=FALSE
F_ST	Calculate fixation index. default=FALSE
params	an object of class "test.params". see ?test.params
detail	detail statistics. Note:slower! default=FALSE

**Details**

You can choose different mutation rate estimators to generate simulation data. When thetaID="user", you have to define the theta values in an object of class "test.params". The "test.params" class can also be used to specify some additional parameter like migration and/or recombination rates... (?test.params).

**Value**

The function creates an object of class "cs.stats"

**Note**

The executable file ms from Hudson have to be stored in the current workspace.

**References**

Hudson, R. R. (2002). Generating samples under a Wright-Fisher neutral model of genetic variation. *Bioinformatics* 18: 337-338

**Examples**

```
# GENOME.class <- readData("../Alignments")
# GENOME.class <- neutrality.stats(GENOME.class,list(1:6))
# MS.class <- MS(GENOME.class,thetaID="Tajima",neutrality=TRUE)
# MS.class
# MS.class@obs.val
```

---

 neutrality.stats-methods

*Neutrality Statistics*


---

## Description

This generic function calculates some neutrality statistics.

## Usage

```
## S4 method for signature 'GENOME'
neutrality.stats(object,new.populations=FALSE,new.outgroup=FALSE,
  subsites=FALSE,detail=FALSE)
## S4 method for signature 'GENOME'
get.neutrality(object,theta=FALSE,stats=TRUE)
```

## Arguments

object	an object of class "GENOME"
new.populations	list of populations. default:FALSE
new.outgroup	vector of outgroup sequences. default:FALSE
subsites	"syn" for synonymous sites and "nonsyn" for nonsynonymous sites
detail	default:FALSE, TRUE for some detail statistics. Note:slower!
stats	show the results of each statistic. default:TRUE
theta	show the theta values. default:FALSE

## Value

Returned value is an modified object of class "GENOME"

---

Following slots will be modified in the "GENOME" object

---

	Slot	Reference	Description
1.	n.segregating.sites		Total number of segregating sites
2.	Tajima.D	[1]	Tajima D statistic 1989
3.	Fu.Li.F	[3]	Fu & Li F* statistic 1993
4.	Fu.Li.D	[3]	Fu & Li D* statistic 1993
5.	Fay.Wu.H	[6]	Fay & Wu H statistic 2000
6.	Zeng.E	[7]	Zeng E statistic 2006
7.	Strobeck.S	[5]	Strobeck S statistic 1987 (if detail==TRUE)

8.	Fu.F_S	[4]	Fu's F <sub>S</sub> statistic 1997 (if detail==TRUE)
9.	Rozas.R_2	[2]	Rozas R <sub>2</sub> statistic 2002
10.	theta_Tajima	[1]	
11.	theta_Watterson		
12.	theta_Fu.Li	[3]	
13.	theta_Achaz.Watterson		
14.	theta_Achaz.Tajima		
15.	theta_Fay.Wu	[6]	
16.	theta_Zeng	[7]	

## References

- [1] Tajima, F.(1989) *Statistical Method for Testing the Neutral Mutation Hypothesis by DNA Polymorphism*. Genetics, 123(3): 585-595.
- [2] Ramos-Onsins, S.E. and J.Rozas (2002). *Statistical Properties of New Neutrality Tests Against Population Growth*. Mol.Biol.Evol.19(12),2092-2100
- [3] Fu, Y.X. and W.H.Li (1993). *Statistical Tests of Neutrality of Mutations*. Genetics 133(3),693-709
- [4] Fu, Y.-X.(1997). *Statistical Tests of Neutrality of mutations against population growth, hitchhiking and background selection*. Genetics 147(2),915-925.
- [5] Strobeck, C. (1987). *Average number of nucleotide differences in a sample from a single sub-population: a test for population subdivision*. Genetics 117, 149-153
- [6] Fay, J.C. and C.-I. Wu (2000). *Hitchhiking under positive Darwinian selection*. Genetics 155 (3),1405-1413
- [7] Zeng, K., Y.-X. Fu, S. Shi, and C.-I. Wu (2006). *Statistical tests for detecting positive selection by utilizing high-frequency variants*. Genetics 174, 1431-1439

## Examples

```
# GENOME.class <- readData("\home\Alignments")
# GENOME.class
# GENOME.class <- neutrality.stats(GENOME.class)
# GENOME.class <- neutrality.stats(GENOME.class,list(1:4,5:10),subsites="syn")
# GENOME.class <- neutrality.stats(GENOME.class,list(c("seq1","seq5","seq3"),
# c("seq2","seq8")))
# GENOME.class <- neutrality.stats(GENOME.class,detail=TRUE)
# show the result:
# get.neutrality(GENOME.class)
# GENOME.class@Tajima.D --> population specific view
# detail = TRUE
```

```
# GENOME.class@region.stats
```

---

 PopGenome

*PopGenome*


---

## Description

R-package for Population genetic analysis

## Details

```
Package: PopGenome
Type: Package
Version: 1.0
Date: 2011-03-17
Depends: R (>= 1.8.0)
License: What license is it under?
LazyLoad: yes
Packaged: Tue Sep 20 15:33:01 2011; bapfe
Built: R 2.8.1; $x86_64$-unknown-linux-gnu; 2011-09-20 15:33:04; unix
```

## Index:

F_ST.stats	Fixation index
MKT	McDonald & Kreitman test
MS	Coalescent simulation
detail.stats	Several statistics
linkage.stats	Linkage disequilibrium
neutrality.stats	Neutrality statistics
readData	Reading alignments and calculating summary data
sliding.window.transform	Sliding window transformation.
test.params	Set parameter for coalescent simulation.

## Author(s)

Bastian Pfeifer Maintainer: Bastian Pfeifer <Bastian.Pfeifer@uni-duesseldorf.de>

## See Also

?readData [readData](#)

## Examples

```
# GENOME.class <- readData("../Alignments")
# GENOME.class <- neutrality.stats(GENOME.class)
# values      <- get.neutrality(GENOME.class)
# GENOME.class <- F_ST.stats(GENOME.class,list(1:5,6:10))
# values      <- get.F_ST(GENOME.class)
```

---

readData

*Reading alignments and calculating summary data*


---

## Description

This function reads alignments in FASTA format and calculates some summary data.

## Usage

```
readData(path,populations=FALSE,outgroup=FALSE,include.unknown=TRUE,
gffpath=FALSE)
```

```
## S4 method for signature 'GENOME'
get.sum.data(object)
```

## Arguments

object	object of class "GENOME"
path	the basepath of the alignments
outgroup	vector of outgroup sequences
include.unknown	if unknown positions should be considered. default:TRUE
populations	list of populations.default:FALSE
gffpath	the basepath of the corresponding gff-files. default:FALSE

## Details

When there is no gff-file specified, an alignment in the right reading frame is expected. Otherwise the examination of synonymous and nonsynonymous positions is useless.

When there is no population defined the whole alignment is considered.

**Value**

The function creates an object of class "GENOME"

---

Following Slots will be filled in the "GENOME" object

---

	Slot	Description
1.	n.sites	total number of sites
2.	n.biallelic.sites	number of biallelic sites
3.	n.gaps	number of sites with gaps
4.	n.unknowns	number of sites with unknown nucleotides
5.	n.valid.sites	number of valid sites
6.	n.polyallelic.sites	number of sites with >2 nucleotides
7.	trans.transv.ratio	transition/transversion ratio of biallelic sites
8.	region.data	some detail data informations

## Examples

```
# GENOME.class <- readData("../Alignments")
# GENOEM.class <- readData("../Alignments", include.unknown=F)
# GENOME.class
# show the result:
# get.sum.data(GENOME.class)
# GENOME.class@region.data
```

---

set.populations-methods

*Define populations*

---

## Description

This generic function defines the populations.

The advantage of this function is, that you don't have to specify the populations for each calculation.

The populations are set for each statistic module.

## Usage

```
## S4 method for signature 'GENOME'
set.populations(object,new.populations=FALSE)
```

## Arguments

object            an object of class "GENOME"

new.populations    list of populations. default:FALSE

**Examples**

```
# GENOME.class <- readData("\home\Alignments")
# pop.1 <- c("seq1","seq2")
# pop.2 <- c("seq3","seq4","seq1")
# GENOME.class <- set.populations(GENOME.class,list(pop.1,pop.2))
# GENOME.class <- neutrality.stats(GENOME.class)
```

---

```
show.slots-methods      Show Slots of class GENOME
```

---

**Description**

coming soon ...

**Methods**

**object** = "GENOME" coming soon ...

**Examples**

```
# show.slots(GENOME.class)
```

---

```
sliding.window.transform-methods
      Sliding Window Transformation.
```

---

**Description**

This generic function transforms the existing object of class "GENOME" in another object of class "GENOME", so that sliding window calculations are possible.

**Usage**

```
## S4 method for signature 'GENOME'
sliding.window.transform(object,width=7,jump=5,type=1)
```

**Arguments**

object	an object of class "GENOME"
width	window size. default:7
jump	jump size. default:5
type	1 scan only biallelic positions, 2 scan the whole alignment. default:1

**Value**

The function creates an object of class "GENOME"

**Note**

In this version of PopGenome Sliding Window applications will scan alignments separately. What can be the file set to be analyzed depends on how much workspace is available. In a future version of PopGenome this problem is circumvented by moving data swapped.

**Examples**

```
# GENOME.class <- readData("../Alignments")
# slide.GENOME.class <- sliding.window.transform(slide.GENOME.class)
# slide.GENOME.class <- neutrality.stats(slide.GENOME.class)
# values <- get.neutrality(slide.GENOME.class)
```

---

test.params-class	<i>Set parameter for Coalescent Simulation.</i>
-------------------	---

---

**Description**

This object can be passed to the function MS after having set parameter values. This class eases the process of passing on all necessary values to the MS function.

**Arguments**

theta	mutation parameter theta ( $4N\mu$ ), where N is the diploid population size and mu the mutation rate per locus. It needs to be provided as vector of length n.regions
seeds	specify 3 random number seeds. a vector of length 3 with positive values is expected
fixedSegsites	usually the number of segregating sites varies in each iteration. Please provide a single numeric value if the number of segregating sites needs to be fixed.
recombination	provide a vector of format: c(p, nsites), p = cross over parameter rate, nsites is the number of sites between recombination occurs
geneConv	in addition to recombination intra-locus non-cross-over exchange gene conversion can be included in simulation, expected format is c(f, gamma), f denote the ratio, g/r, where r is the probability per generation of crossing-over between adjacent sites. (see Wiuf and Hein 2000), gamma is the mean conversion tract length
growth	population size is measured by $N(t) = N_0 \exp^{\alpha * t}$ . provide alpha as integer value. negative values indicate that population was larger in the past than present
migration	specify the migration rate between populations. Please provide a single numeric value.

demography vector of length 3 or 4 with first value denoted as 'type'  
 valid 'types' for vectors of length 3 are as following:  
 - 1 to set a growth rate change alpha at a certain time t:  
 c(1, t, alpha)

- 2 set all subpop to size  $x * N_0$  and growth rate to zero:  
 c(2, t, x)

- 3 set all elements of migration matrix to  $x/(n_{pop}-1)$ :  
 c(3, t, x)

valid 'types' for vector of length 4 with the following values:  
 - 4 set growth rate of subpop i to alpha at time z:  
 c(4, t, i, alpha)

- 5 set subpop i size to  $x * N_0$  at time t and growth rate to zero:  
 c(5, t, i, x)

- 6 split subpopulation i into subpopulation i and a new subpopulation,  
 labeled npop + 1. Each ancestral lineage in subpopulation i is randomly  
 assigned to subpopulation i with probability p and subpopulation  
 npop + 1 with probability 1 - p. The size of subpopulation npop + 1 is  
 set to  $N_0$ . Migration rates to and from the new subpopulation are assumed  
 to be zero and the growth rate of the new subpopulation is set to zero:  
 c(6, t, i, p)

- 7 move all lineages in subpopulation i to subpopulation j at time t.  
 Migration rates from subpopulation i are set to zero:  
 c(7, t, i, j)

**Author(s)**

Bastian Pfeifer

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

```
# params          <- new("test.params")
# params@theta    <- rep(5,n.regions)
# params@migration <- 3
```

# Index

## \*Topic **classes**

GENOME-class, 5  
test.params-class, 20

## \*Topic **methods**

detail.stats-methods, 2  
F\_ST.stats-methods, 3  
get.status-methods, 8  
linkage.stats-methods, 9  
MKT-methods, 10  
MS, 11  
neutrality.stats-methods, 13  
readData, 16  
set.populations-methods, 18  
show.slots-methods, 19  
sliding.window.transform-methods,  
19

## \*Topic **package**

PopGenome, 15

detail.stats (GENOME-class), 5  
detail.stats, GENOME-method  
(detail.stats-methods), 2  
detail.stats-methods, 2

F\_ST.stats (GENOME-class), 5  
F\_ST.stats, GENOME-method  
(F\_ST.stats-methods), 3  
F\_ST.stats-methods, 3

GENOME-class, 5  
get.detail (GENOME-class), 5  
get.detail, GENOME-method  
(detail.stats-methods), 2  
get.detail-methods  
(detail.stats-methods), 2  
get.diversity (GENOME-class), 5  
get.diversity, GENOME-method  
(F\_ST.stats-methods), 3  
get.diversity-methods  
(F\_ST.stats-methods), 3

get.F\_ST (GENOME-class), 5  
get.F\_ST, GENOME-method  
(F\_ST.stats-methods), 3  
get.F\_ST-methods (F\_ST.stats-methods), 3  
get.linkage (GENOME-class), 5  
get.linkage, GENOME-method  
(linkage.stats-methods), 9  
get.linkage-methods  
(linkage.stats-methods), 9  
get.MKT (GENOME-class), 5  
get.MKT, GENOME-method (MKT-methods), 10  
get.MKT-methods (MKT-methods), 10  
get.neutrality (GENOME-class), 5  
get.neutrality, GENOME-method  
(neutrality.stats-methods), 13  
get.neutrality-methods  
(neutrality.stats-methods), 13  
get.status (GENOME-class), 5  
get.status, GENOME-method  
(get.status-methods), 8  
get.status-methods, 8  
get.sum.data (GENOME-class), 5  
get.sum.data, GENOME-method (readData),  
16  
get.sum.data-methods (readData), 16  
getBayes, GENOME-method (GENOME-class), 5  
getMS, GENOME-method (GENOME-class), 5

linkage.stats (GENOME-class), 5  
linkage.stats, GENOME-method  
(linkage.stats-methods), 9  
linkage.stats-methods, 9

MKT (GENOME-class), 5  
MKT, GENOME-method (MKT-methods), 10  
MKT-methods, 10  
MS, 11, 21

neutrality.stats (GENOME-class), 5

neutrality.stats, GENOME-method  
    (neutrality.stats-methods), 13  
neutrality.stats-methods, 13

popFSTN, GENOME-method (GENOME-class), 5  
PopGenome, 15

readData, 15, 16

set.populations (GENOME-class), 5  
set.populations, GENOME-method  
    (set.populations-methods), 18  
set.populations-methods, 18  
show, GENOME-method (GENOME-class), 5  
show.slots (GENOME-class), 5  
show.slots, GENOME-method  
    (show.slots-methods), 19  
show.slots-methods, 19  
sliding.window.transform  
    (GENOME-class), 5  
sliding.window.transform, GENOME-method  
    (sliding.window.transform-methods),  
    19  
sliding.window.transform-methods, 19

test.params (test.params-class), 20  
test.params-class, 20

usage, GENOME-method (GENOME-class), 5