

VarisScan

Analysis of large-scale DNA sequence polymorphism data

Version 2.0

Albert J. Vilella **Departament de Genètica**
Angel Blanco-García **Universitat de Barcelona**
Stephan Hutter **Barcelona**
Julio Rozas **Spain**



www.ub.es/softevol/variscan

Last updated: March 36, 2006

1 Overview

VariScan is a software package for the analysis of DNA sequence polymorphisms at the whole genome-scale. The software, i) allows estimating a number of population genetic parameters; ii) conducts the analysis by using the sliding window or a wavelet transform-based methods to capture relevant information from DNA polymorphism data; iii) permits the visualization of the results by commonly used genome browsers.

The software is written in ANSI C, therefore it can be compiled on a wide variety of operating systems, and it has been tested on Linux, Mac OS X and Win32 platforms.

Authors

Albert J. Vilella	avilella@gmail.com
Angel Blanco-Garcia	ablancog@ub.edu
Stephan Hutter	hutter@zi.biologie.uni-muenchen.de
Julio Rozas	jrozas@ub.edu

VariScan references

VariScan version 1:

Vilella, A. J., Blanco-Garcia, A., Hutter, S. and Rozas, J. (2005). VariScan: Analysis of evolutionary patterns from large-scale DNA sequence polymorphism data. *Bioinformatics* **21**: 2791-2793.

VariScan Web Site

<http://www.ub.es/softevol/variscan>

Acknowledgements

The software package has been carried out by Spanish grants BMC2001-2906 from the Dirección General de Investigación Científica y Técnica conferred on M. Agudé, and TXT98-1802 from the Dirección General de Enseñanza Superior e Investigación Científica conferred on J. Rozas.

We thank P. Librado for their contribution in writing some scripts, and also thanks to M. Agudé, J. M. Aroca, B. Audit, M. Casas, J. Castresana, S. O. Kolokotronis, D. Posada, S. E. Ramos-Onsins and C. Segarra for their valuable comments and suggestions.

2 Installation

Linux / Mac OS X

The package includes executables for linux (**variscan**) and Mac OS X (**variscan**). For other Unix-based platforms you will have to compile it from the source files included in the *VariScan* package.

1) From the command terminal decompress the file **variscan-version.tar.gz**
tar xzf variscan-version.tar.gz

2) The package includes a compiled (executable-binary) file. You can also compile the software:
cd variscan
./autogen.sh
make

This step, which will take from a few seconds to a minute, will generate the **variscan** executable binary file.

3) The package also includes the executable for the *LastWave* software (see section 8).

Windows

The package includes (**src** directory), the source code, the project (**variscan.dev**) and makefile (**variscan.win**) files to be used, for instance, for the Dev-C++ (a free Integrated Development Environment for the C/C++ programming language). We are also distributing a Windows executable file (**variscan.exe** file included on **variscanWin-version.zip** file). The package software does not include any executable for *LastWave*.

3 Input Data Files

The input data files are multiple aligned DNA sequence data in a number of interleaved formats as MAF, MGA, XMFA, PHYLIP, or the HapMap genotype format.

MAF (<http://genome.ucsc.edu/goldenpath/help/maf.html>).

MGA (<http://bibiserv.techfak.uni-bielefeld.de/mga/>).

XMFA (<http://lagan.stanford.edu/>).

PHYLIP (<http://evolution.genetics.washington.edu/phylip.html>).

HapMap genotype format (<http://www.hapmap.org/downloads/encode1.html.en>).

MAF Format

An example of this format is included in `InputDataFiles/maf_example.maf`, and an example config file in `InputDataFiles/maf_example.conf`

```
##maf version=1 scoring=whatever
# stuff on how it is obtained can be in this line
a score=30778.0
s sequence1 1542 460 + 246127941 ctggagattctta-ttagtgatttgggctggggc-
ctggccatgtgtatttttta-aatttcactgatgattttgctgcatggccggtgttgagaatgactgCG-
CAAATTTGCCGGATTTCTTTGCTGTTCTGTCATGTAGTTTAAACGAGATTGCCAGCACCGGGTATCATTACCAT-----
-----
-----
TTTTCTTTTCGTTAACCTGCCGTGAGCCTTTCTTTGACCTCTCTTTCTGTTTCATGTGTATTGCTGTCTTTAGCCCAGACTTCCCGTGTCT
CTTTCCACCGGGCCTTTGAGAGGTCACAGGGTCTTGATGCTGTGGTCTTTCATCTGCAGGTGTCTGACTTCCAGCAACTGCTGGCC---
TGTGCCAGGGTGCAAGCTGAGC-ACTGGAGTGGAGTTTCTGTGGAGAGGACCATGCCTAGAGTGGGATGGGCCATTGTTTCATC-
TTCTGGCCCTGTTGTCT
s sequence2 27723223 600 - 149950539
CTAGGGAGTCTTAGTCAAAGGTTTGGACCAAGTCCCTGGCCATGCAGATCTTTGTAGAATCTCCACTCGTACTTCTGCATAACCAGAGTT
GAGCATCTTTGAGTCAAGTGTGCCAACTTTCTT-----
TGCTGTTAAATAAGGATGCCAACACCCGATGTCATTAACAGTCTCGTAGGTTGATTGATTTGTTGGCTGGCTCAAAAATGAGAG-
TTATTTTTTCATTTTGTTTTGAT-----
-----
tgattttttaagtcttgatctagatagcccagctgggttggagcttactatgtagtttaggttgcgtgcaactctcaatctccagctccca
atcctcaagtgtcctccaggtctatgtctactgtactcagGTAAAAAGGAG-TTTTCTGTCTGCTAATTTGCCACAGTCATTTT-----
-----CTATT-ACGTGTGTCTGCTGCCTCTAGCCCAGGCT-----TGCCCTTCTCTCC--TCTTCTGAGGTGTCATAGGGTCTGAC---
-----TTACCTGGTTTGGGGGAGTAGTTGGAA-----GCTGAGTGAGTG--
GTGGGGTTTTCTTATGCTAAAGACCTGCGTCCAGTATAGGAAGGCCATGTGCCCTCCACTCTGGCCCTTGTGGTCT
s sequence3 29160419 613 - 187371129
CTGGAGAGTCTTATTTGAAGGGTTGGACCAAGCCACTGGCCATGTAGATCTATTATAATCACTACTGGTGACTTTCATGTATAACCAGAGTT
GAGCATCTTTGAGTCAAATGTGCCAAATTTCT-----
TGCTGTTTAAATAAGGATGCCAACACTGCATATCATTAAACAGTCTTGTAGGTTGATTGATTAGTTGGCTGGCTGGGGAACGGGGGAGTATTT
TCATTTTGTTTTATTGTTTAAAGTATGATCTATATAGCCAGCTGGGCTGGAGCTTACTATGTAGTTTAGATTAGGCTGCAACTCTCAATCCT
CCTGTCTCCACTTTCCAGTGTCACTCCAGGCTATGCTACTGT-----
-----
ACCCAGCTAAAAAGTAGTTTTCTGCCTGCTAATTTGCCACAGTCTTTTC-----CTGTT-
ACATGTACCCACTGCCTCCTAGCCTAGGCT-----TGTCTTCTCTCC--TCTTCTGAAAGGTACAGGGTCTTGAC-----
----TTACCTGGGTTGGGGGAGGGGTTGGAAGCACACGCTGATTTGGATGCTGAGTGACTG--
GTGGAGTTTTCTTATGACAAAGACCTGTGTCCAGGATGGGATAGGCCACACGCTTCC-CTCTGGCCCTTGTGGTCT

a score=1111.0
s sequence1 2040 63 + 246127941
ATTGGAGGAAAGATGAGTGAGAGCATCAACTTCTCTCACAACTAGGCCAGTAAGTAGTGCTT
s sequence2 64593042 56 + 93529596 GTTGAGGGAAGATGAGTGAAGGATCAATTTCTCTGATGACCTGGGCCGGTAGGT-
-----
s sequence3 29162556 61 - 187371129 ATTGGAGGGAGGGTGAACAAAGAGATAGACT--
TCTGGCAACTGGGCCAGTAGGTAGTGCT
s sequence5 3040 63 + 246127941
TTAGGTGATAGATTAGTGATAGCATCTTCTCTCAACCTAGTCCAGTAAGTATTGCTT

a score=-999.0
s sequence1 2103 19 + 246127941 GTGCTCATCTCCTTGGCTG----
s sequence2 64593098 19 + 93529596 GTGGTGTCTCTTTGTCTG----
s sequence3 105237468 19 + 113649943 ----GGCCCTAATTGCTAAGGCA
s sequence4 105237468 19 + 113649943 ----GGCCCTAATGCTATAGGCA
s sequence5 105237468 19 + 113649943 ----GGCCCTAATTGCTAAGCGCA
```

MGA Format

An example is included in [InputDataFiles/mga_example.mga](#)

39 0 0 0 0 Exact: ttgttgatattctgttttttcttttttagttttccacat	39
!39 !39 !39 !39 Seq 1: g Seq 2: a Seq 3: a Seq 4: a	1
28 40 40 40 40 Exact: aaaaatagttgaaaacaatagcgggtgc	28
12:68-79 12:68-79 12:68-79 12:68-79 Seq 1: cccttaaaatgg Seq 2: a..... Seq 3: a.....a Seq 4: a.....a	12
23 80 80 80 80 Exact: cttttccacaggttggtggagaac	23
5:103-107 5:103-107 5:103-107 5:103-107 Seq 1: ccaaa Seq 2: t...g Seq 3: Seq 4:	5
27 108 108 108 108 Exact: ttaacagtgtaattttttttccacag	27
32:135-166 2:135-136 32:135-166 32:135-166 Seq 1: gttgtggaaaaactaactattatccatcgttc Seq 2: ----- Seq 3: a.....t.c.. Seq 4: a.....t.c..	32
216 167 137 167 167 Exact: tgtgaaaactagaatagtttatggtagaatagttctagaattatccacaagaaggaacc Exact: tagtatgactgaaaatgaacaattttttggaacagggcttggattagctcagagtca Exact: attaaaacaggcaacttatgaattttttgttcatgatgcccgctattaaaggtcgataa Exact: gcatattgcaactatttacttagatcaaatgaaaga	60 120 180 216

XMFA Format

An example is included in [InputDataFiles/xmfa_example.xmfa](#), and an example config file in [InputDataFiles/xmfa_example.conf](#)

```
>1:1-598 + chrY
TCCAAGTCGGCTTTATGTTTCTGCTCCAGGCATTCTAGATGCCCATGTCTAGGATCT
CTTTAGGCAGGAGAGAGGGTGATGGTGTAGGAGGACCCATTTCTTGGCTTGCAGATTCCA
ATAATAAAAAAGTCACAGATTTAAACCCCAAACCTTGTGAAATGCAGGTCTAGGGTTTT
AAAATATAATGAGAGTTAAATACTTTTTGATTTTTCTTCCAGAGATGGGGCAAGCTTC
CTCATCTGCTCGTTCATGGGTGATTTATATTTTCCCCACTCCATCCTTTTCTAAGGTAT
TTTTTTTTTAGGGACAATGGCTTTTGCAGAGTACTCAGTCCAGCTCCGGGGGACCCGG
TTGAGCCCTTACCGTCTGCCCTTAAACATCCAGACCTCAAGTTAGAGAGGGGAGTAACA
TTTGGGGGGTGCACACCTAGGAGGACCAATCCTTCTGGTTTCTTAGGGATGCAGGAA
TTTGGGGGGGGGGGCTCAGTGCTAAACCAAGTAGAGTCTGGGCAACAGATGACTG
AAGATGCTTTGAACACCCTAGCGTTATGTCGATCGATGCATCGTAGTGTGCTGATG
>2:5000-5598 - chr17
TGCAGATTGGCCTT-TGTTTCGTTTTTC-AAGCGTT-TAAA--CGCCTTGCCTAAGAATC
TTTT--GCAGGGAAGGGGATAGTGAAGTGGGAAAACCTGGCTCTTCTTTTCGAGATTCCA
GTAACAAACATGTCATAAATATAAACCGCCAAACTTGG--AGAGCGCAGGAATGAAGGTC
AAACACCAATGAGAGTTAGATGGTTTTGGGTTT-----GCT--
CTAGTCTGCACG-----GTGCTCCCGTCCCCTCACGTCCGTGCTTTTCTCAGGATG
ATGCCTTGCCAGAACACCGGTGTGCTGCAAGGTGCTCAGTCCAAATCGGGCTGCACCGC
TTCAGCTTTCCCATCCAGCCA--ACGCAGGAAGGCCTGGAGCTACAGAGTTTAGAGCCA
TCTCTCCGCTGCTCAT-----TAACCAACCATTCAGCT-----GTCTGTAGTGG
GTTTTTTTCTT----CTCTACATAAAATGAGGACAGTCCAGGCCCTTTG--TTAGACTG
AAGATGCTTTGAACACCCTAGCGTTATGTCGATCGATGCATCGTAGTGTGCTGATG
>3:19000-19598 - chr7
TCCAGACTGTCTTT-TGCTCCCTTTTTCCGAGCATT-TAAAAATACCATGCCTAAGAATC
TTTT--GCAGGGAAGGGGATAGCAGGCTGGGAAGGCATTTTCTTTCATTTTCGAGATTCTG
GTAATAAACATGTCATAAATATAAATGCCAAACTCCG--GAAATGCAGGTGTAGAGCGTC
AGATTCTATTTGGACTTAAATGATGTGGTGTTTT-----GCT--
CTAATTTCTACC-----GTGCTCTCCGTTCC-TCAAGTCCATGCATTTCTTAGGGTG
CTGCCTTTCCAGAGTACTGGTATGCTGCAGGGTCTCAGTCCACATCTGTCTGCACTAT
TTCAAAGTTTCCC-TCCAGCCC--ACACAACATGCTAGAGCTA--GAGGTTAGAACC
TCTGTCCA-TGCTCTT-----TAACCAACCATCCAGAT-----AGGTGTGGTGG
TTTTTTTTTTTTTTTCTCTGACTAAAATTAGGACAGTCCAGGCCTGTTG--TTAGACCA
AAGATGCTTTGAACACCCTAGCGTTATGTCGATCGATGCATCGTAGTGTGCTGATG
= score = 111
>1:1000-1060 + chrY
CACTCTAATAGTAAAGTTTCTTTTGTCTGTGCGAAGCTCTTAGTTTAATTAGATCCCAT
>2:6000-6060 + chr17
CACTCTAATACTAAACTTTCTTTTCTCTCCACA----CTTTACTTTAATTACATCCCAT
>3:20000-20060 - chr12
CACTCTAATAGTAAAGTTTCTT----TGTGCGAAGCTCTTAGTTTAATTAGATCCCAT
= score = 11
```

PHYLIP Format

An example is included in [InputDataFiles/phylip_example.phy](#). PHYLIP files should be interleaved (large sequential files might require a lot of RAM memory).

```
5 150
AY044121 aatctggtgatcttggaccgtattgagaaccccgggcattgctgagctgaaggcaatcaatcctaagg
AY044123 aatctggtgatcttggaccgtattgagaaccccgggcattgctgagctgaaggcaatcaatcctaagg
AY044124 aatctggtgatcttggaccgtattgagaaccccgggcattgctgagctgaaggcaatcaatcctaagg
AY044125 aatctggtgatcttggaccgtattgagaaccccgggcattgctgagctgaaggcaatcaatcctaagg
AY044126 aatctggtgatcttggaccgtattgagaaccccgggcattgctgagctgaaggcaatcaatcctaagg

gcccgtttactcagcctccaagggcgtgtgtcagcttcaccagctcattggctgtaagtata-----
gcccgtttactcagcctccaagggcgtgtgtcagcttcaccagctcattggctgtaagtata-----
gcccgtttactcagcctccaagggcgtgtgtcagcttcaccagctcattggctgtaagtata-----
gcccgtttactcagcctccaagggcgtgtgtcagcttcaccagctcattggctgtaagtata-----
gcccgtttactcagcctccaagggcgtgtgtcagcttcaccagctcattggctgtaagtata-----

accaagcact
accaagcact
accaagcact
accaagcact
accaagcact
```

HapMap genotype format

Included in `InputDataFiles/hapmap_example.hapmap`. Entries should be ordered by chromosome position (`pos` field).

```
rs# SNPalleles chrom pos strand genome_build center protLSID assayLSID panelLSID QC_code
NA06985 NA06991 NA06993 NA06993.dup NA06994 NA07000 NA07019 NA07022 NA07029 NA07034 NA07048
NA07055 NA07056 NA07345 NA07348 NA07357 NA10830 NA10831 NA10835 NA10838 NA10839 NA10846
NA10847 NA10851 NA10854 NA10855 NA10856 NA10857 NA10859 NA10860 NA10861 NA10863 NA11829
NA11830 NA11831 NA11832 NA11839 NA11840 NA11881 NA11882 NA11992 NA11993 NA11993.dup NA11994
NA11995 NA12003 NA12003.dup NA12004 NA12005 NA12006 NA12043 NA12044 NA12056 NA12057 NA12144
NA12145 NA12146 NA12154 NA12155 NA12156 NA12156.dup NA12234 NA12236 NA12239 NA12248
NA12248.dup NA12249 NA12264 NA12707 NA12716 NA12717 NA12740 NA12750 NA12751 NA12752 NA12753
NA12760 NA12761 NA12762 NA12763 NA12801 NA12802 NA12812 NA12813 NA12814 NA12815 NA12864
NA12865 NA12872 NA12873 NA12874 NA12875 NA12878 NA12891 NA12892
rs10171150 A/G Chr2 2091 + ncbi_b34 mcgill-gqic
urn:LSID:illumina.hapmap.org:Protocol:Golden_Gate_1.0.0:1 urn:LSID:mcgill-
gqic.hapmap.org:Assay:810448:1 urn:lsid:dcc.hapmap.org:Panel:CEPH-30-trios:1 QC+ GG GG GG GG
GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG
GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG
GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG
rs10193286 C/G Chr2 5491 + ncbi_b34 mcgill-gqic
urn:LSID:illumina.hapmap.org:Protocol:Golden_Gate_1.0.0:1 urn:LSID:mcgill-
gqic.hapmap.org:Assay:617205:1 urn:lsid:dcc.hapmap.org:Panel:CEPH-30-trios:1 QC+ GG GG GG GG
GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG
GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG
GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG
rs4632379 A/G Chr2 5672 + ncbi_b34 mcgill-gqic
urn:LSID:illumina.hapmap.org:Protocol:Golden_Gate_1.0.0:1 urn:LSID:mcgill-
gqic.hapmap.org:Assay:225142:1 urn:lsid:dcc.hapmap.org:Panel:CEPH-30-trios:1 QC+ AA AA AA AA
AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
rs3901618 A/G Chr2 5892 - ncbi_b34 mcgill-gqic
urn:LSID:illumina.hapmap.org:Protocol:Golden_Gate_1.0.0:1 urn:LSID:mcgill-
gqic.hapmap.org:Assay:423166:1 urn:lsid:dcc.hapmap.org:Panel:CEPH-30-trios:1 QC+ GG GG GG GG
GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG
GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG
GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG
rs7594188 C/T Chr2 11494 + ncbi_b34 mcgill-gqic
urn:LSID:illumina.hapmap.org:Protocol:Golden_Gate_1.0.0:1 urn:LSID:mcgill-
gqic.hapmap.org:Assay:346814:1 urn:lsid:dcc.hapmap.org:Panel:CEPH-30-trios:1 QC+ CC CC CC CC
CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC
CC CC CC CC CC CC CC CC CC CC CC NN CC CC
CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC
rs7594567 C/G Chr2 11833 + ncbi_b34 mcgill-gqic
urn:LSID:illumina.hapmap.org:Protocol:Golden_Gate_1.0.0:1 urn:LSID:mcgill-
gqic.hapmap.org:Assay:523981:1 urn:lsid:dcc.hapmap.org:Panel:CEPH-30-trios:1 QC+ CC CC CC CC
CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC
CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC
CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC
rs6725981 A/G Chr2 14445 + ncbi_b34 mcgill-gqic
urn:LSID:illumina.hapmap.org:Protocol:Golden_Gate_1.0.0:1 urn:LSID:mcgill-
gqic.hapmap.org:Assay:811673:1 urn:lsid:dcc.hapmap.org:Panel:CEPH-30-trios:1 QC+ GG GG GG GG
GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG
GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG
GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG GG
```

Phased/Unphased and SNPs data

Although some of the analysis can be conducted using unphased data (in general, genotypic data is phase-unknown), the gametic phase information is needed in some of the implemented methods (e.g., LD, Haplotype diversity); therefore, the gametic phase should be determined before using these methods in *VariScan*.

For instance, HapMap genotype format include genotype data information (SNPs data), which is currently unphased. Therefore, association-based statistic values (LD, Haplotype diversity, F_s , etc.) will be incorrect. If the data does not include genotype information for all contiguous sites of the studied genomic region, per-site based statistics (e.g. π), will be also incorrect. In addition, the results will also be biased if the typed SNPs were originally identified (discovery panel) in a small number of individuals.

4 VariScan Programs and Files

4.1 VariScan Programs

VariScan (linux, Mac OS X); **variscan.exe** (windows)

The main *VariScan* program. Computes all population genetic parameters and includes the sliding window option.

Input 1: Multiple aligned DNA sequence data.

Input 2: *VariScan* config file (*.conf).

Output: File with summary statistics and sliding window results (*.vs).

VariScanGUI.jar (linux, Mac OS X, windows)

This is the Graphical User Interface (GUI) of *VariScan*. This allows conducting all analyses with a friendly-user interface. Nevertheless, you can also run the software by a command-line (see below).

runMRA.PLS

Perl script to run *LastWave* software to obtain the MRA decomposition analysis.

The output from the *VariScan* program is the input for *LastWave*.

Input 1: *VariScan* output (*.vs).

Input 2: Config file (*.mraconf).

Output: File with MRA results (*.mra).

mra2bed.PLS

Perl script to obtain from the MRA decomposition output a file formatted according to the wiggle custom tracks at the UCSC genome browser.

Input 1: runMRA.PLS output (*.mra).

Output: Output file (*.bed).

mra2gbrowse.PLS

Perl script to obtain from the MRA decomposition output a file formatted according to the xyplot tracks at the Gbrowse genome browser (version higher than 1.62).

Input 1: runMRA.PLS output (*.mra).

Output: Output file (*.xyplot).

vs2bed.PLS

Perl script to obtain from the sliding window output a file formatted according to the wiggle custom tracks at the UCSC genome browser.

Input 1: *VariScan* output (*.vs).

Output: Output file (*.bed).

vs2gbrowse.PLS

Perl script to obtain from the sliding window output a file formatted according to the xyplot tracks at the Gbrowse genome browser (version higher than 1.62).

Input 1: *VariScan* output (*.vs).

Output: Output file (*.xyplot).

gff2bdf.pl

Perl script that parses through a generic/general feature format files (GFF) and creates a block data file (BDF) with the specified features that can be used by *VariScan*.

Input 1: GFF file.

Output: BDF file (*.bdf).

4.2 VariScan Config File (*.conf)

This file is used to specify the kind and range of analysis. Lines beginning after the symbol '#' are considered as comments.

Range of analysis

StartPos First site to be analysed (eg. **StartPos = 1**)
EndPos Last site to be analysed.
LastPos = 123000 Analysis until position 123000.
LastPos = 0 Analysis until the last position.
RefPos Indicates the site-position coordinates.
RefPos = 1 Numbers given in **StartPos** and **EndPos** are positions in the reference sequence.
RefPos = 0 Numbers given in **StartPos** and **EndPos** are positions on the multiple alignment file.

BlockDataFile command

Specifies if a BlockDataFile defining specific regions for the analysis should be used. A detailed description of this feature can be found in the appendix.

BlockDataFile = none No BlockDataFile should be used.
BlockDataFile = myfile.bdf The file myfile.bdf defines the regions to be analysed.

Sequence Names

Mandatory for MAF file formats. For other formats the parameter will be ignored. See the appendix for a detailed description of this parameter.

IndivNames = Name1 Name2 Name3 Name4

Include/Exclude Sequences

SeqChoice = 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1

There are 20 sequences in the multiple aligned data file. Individuals (sequences) 2 and 6 will be excluded for further analysis.

SeqChoice = all All sequences will be included. This option does not work for XMFA file format.

Outgroup command

Defines the outgroups in the multiple alignment file. A detailed description of this feature can be found in the appendix.

Outgroup = none There is no defined outgroup in the alignment.
Outgroup = first The first sequence of the alignment is the outgroup.
Outgroup = last The last sequence of the alignment is the outgroup.
Outgroup = 0 0 1 0 2 0 0 0 The alignment consists of 8 sequences, sequence #3 is the closest outgroup and sequence #5 is a more distantly related outgroup. All other sequences belong to the ingroup

Reference Sequence

Mandatory for **RefPos = 1**. Defines the individual used for the reference sequence. This sequence does not need to be included. For MAF files, the number entered in **RefSeq** must be the position of the individual in the **IndivNames** vector.

RefSeq = 3 Sequence (individual) Num#3 is the reference sequence

RunMode Parameter

This parameter specifies the types of analysis to be performed. *n*, denotes the minimum number of sequences needed for the analysis.

RunMode = 11	Summary statistics [π (π); Theta (θ)]; $n \geq 2$
RunMode = 12	Summary statistics (π ; Theta; Tajima's D ; Fu-Li's D^* and F^*); $n \geq 4$
RunMode = 21	Summary statistics with outgroup [π ; K (divergence)]; $n \geq 1$
RunMode = 22	Summary statistics with outgroup (π ; Fu-Li's D , F ; Fay-Wu's H); $n \geq 3$
RunMode = 31	Linkage disequilibrium (D , D' , r^2), Haplotype diversity and Fu's F_S ; $n \geq 2$
RunMode = 51	Transforms a multiple alignment file (other than MGA) to PHYLIP

Segregating Sites / Number of Mutations

This parameter specifies the use of the total number of segregating sites (S) or the total number of mutations (η) for some analysis (such as theta, Tajima's D , etc)

UseMuts = 1	Use the total number of mutations
UseMuts = 0	Use the total number of segregating sites

Singletons Usage in LD Analysis

This option defines if singleton variants are to be used when calculating linkage disequilibrium statistics.

UseLDSinglets = 0	Ignore singletons for LD analysis.
UseLDSinglets = 1	Use singletons for LD analysis.

Handling Gaps / Missing Data

This option determines how to treat alignment gaps and missing data. See the appendix for a more detailed description of these options.

CompleteDeletion = 1	All sites containing gaps (-), missing or ambiguous nucleotides will be excluded from analysis. Tajima's D , and Fu and Li's parameters will be calculated. Values given in FixNum and NumNuc will be ignored. This is mandatory for RunMode = 31
CompleteDeletion = 0	Sites with gaps or ambiguities will be included in the analysis as defined in the FixNum and NumNuc parameters.
FixNum = 1	Analysis in a fix number of sequences. Only sites containing a fix number (defined in NumNuc) of valid (after excluding gaps and missing data) sequences will be used. If the site contains more valid sequences <i>VariScan</i> will randomly choose NumNuc of them. This will allow calculating the standard Tajima's D , and Fu and Li's statistics using the same number of sequences.
FixNum = 0	Analysis in a minimum number of sequences. Only sites that contain at least (defined in NumNuc) valid sequences will be used. Since the sample size may vary from site to site, Tajima's and Fu and Li's statistics do not behave the standard way. See the appendix for details.
NumNuc = X	Defines the fixed or minimal number of sequences to be used in the analysis. Sites containing less valid sequences will be ignored. Depending on the X value, the program can calculate only specific statistics. For example, with NumNuc = 2 (valid for RunMode = 11 and RunMode = 12), only π and Theta will be calculated. With NumNuc = 4 (or higher), all statistics will be calculated.

Sliding Window Option

The following parameters determine a number of options to be used for the sliding window (SW) analysis.

SlidingWindow = 1	Perform the sliding window analysis
SlidingWindow = 0	Do not conduct the sliding window analysis.
WidthSW = X	Defines the number of sites of the sliding window width
JumpSW = X	Defines the number of sites of the sliding window slide (jump) of the. If WidthSW = JumpSW , the analysis will be performed on a non-overlapping windows.

The **WindowType** parameter specifies which sites are including in the **WidthSW** and **JumpSW** parameters. See the appendix for a more detailed description of this option.

- WindowType = 0** The number of sites refers to number of columns in the alignment.
- WindowType = 1** The number of sites refers to number of net sites (excluding discarded sites).
- WindowType = 2** The number of sites refers to the number of polymorphic sites.
- WindowType = 3** The number of sites refers to number of positions in the reference sequence.

4.3 MRA Config File (*.mraconf)

This file is used to specify the information to be extracted from the *VariScan* output file (*.vs; see also section 6), and the information needed for the LastWave software (mother wavelet, MRA levels, etc.).

Analysis

- stat_col = X** Column with statistic results to be analysed by MRA
- ref_col = Y** Column with the reference positions (usually Y = 3)

MRA decomposition levels

The MRA analysis divides the input signal into different components. The subdivisions values define these groups.

- sub_a = J** Low-frequency bands from (1 to J-1) levels
- sub_b = K** Frequency bands from (J to K-1) levels
- sub_c = L** Frequency bands from (K to L-1) levels.

The last group will include levels from L to the maximum.

If you want to obtain 4 subdivisions, set **sub_a < sub_b < sub_c**.

If you want to obtain 3 subdivisions, set **sub_a = sub_b < sub_c**.

If you want to obtain only 2 subdivisions, set **sub_a = sub_b = sub_c**.

LastWave directories

Specifies the directory paths to be used for LastWave. For example:

- lwpath = \$HOME/LastWave_2_0_4**
- lwsourcedir = \$HOME/LastWave_2_0_4/scripts**
- lwbinary = \$HOME/LastWave_2_0_4/bin/linux/lw**

Mother Wavelet filter

- filter = D4.o** Specifies the mother wavelet (filter) to be used in MRA analysis. Usually the Daubechies D4 filter (**D4.o**) will be used. More information in: **LastWave_2_0_4/scripts/wtrans1d/filters/** directory.

5 Analyses

VariScan implements several population genetic parameters including coalescent-based statistics (Kingman, 1982; Nei, 1987; Hudson, 1990; Nordborg, 2001; Rosenberg and Nordborg, 2002). In particular, *VariScan* computes (1) summary statistics of nucleotide polymorphism levels: the population mutational parameter (θ), nucleotide diversity (π), and haplotype diversity (Nei, 1987; Depaulis and Veuille, 1998); (2) linkage disequilibrium based-statistics: D' (Lewontin, 1964), r^2 (Hill and Robertson, 1968), and Z_{ns} (Kelly, 1997); (3) neutrality-based tests: Tajima's D (Tajima, 1989), Fu and Li's D^* and F^* (Fu and Li, 1993) and Fu's F_s (Fu, 1997). *VariScan* can estimate these parameters on a specific number of sequences, or considering different options of treating gaps and missing data. A detailed description of the output for each **RunMode** value can be found in the appendix.

Sliding Window (SW)

All of analyses implemented in *VariScan* can be conducted using the standard sliding window method that, in turn, can be used to obtain a graphical representation of the results (a detailed description is included in the appendix).

Multiresolution Analysis (MRA)

VariScan implements the wavelet transform (WT) analysis for capturing information from DNA sequence data (Arneodo et al., 1995; Sweldens, 1996; Liò, 2003). WT is a relatively new mathematical method, similar to the well-known Fourier transform (FT), very useful for the frequency analysis of signals that are localized in time and space. The method allows obtaining low and high frequency components from signals (for instance, the profile of nucleotide diversity along the DNA sequence), and therefore it could be useful in capturing features from DNA sequence data. We have used the discrete wavelet transform (DWT) to decompose the signal into several groups of coefficients, which contain information on the global and local features of DNA sequence data. Therefore, the method can be helpful in detecting relevant features from DNA polymorphism data at a genome-wide scale, such as conserved regions, peaks and valleys of nucleotide diversity, linkage disequilibrium clusters, etc, that in turn might reveal the distinctive footprint left by the action of natural selection.

Here, the signal is the raw profile of the statistic obtained along the DNA sequence. The signal, which can be envisaged as a one-dimensional vector (of length L), is further analysed using the **wtrans1d** module of LastWave v2.0 software (E. Bacry; <http://www.cmap.polytechnique.fr/~bacry/LastWave/>). The DWT analysis requires a signal to have a number of points equal to some power of two. For this purpose, and to avoid the boundary effect problem (McGill and Taswell, 1993), we used the mirror padding method. With this approach the signal is extended by mirroring both ends at the boundaries, to achieve a total length (L') as a power of two. After the WT analysis, the padding tags are discarded and the original signal (of length L) is recovered. We chose Daubechies' D4 as the default wavelet filter (Daubechies, 1992). This filter is

adequate for locating features as peaks and valleys from a signal, with a minimum degree of smoothness (Liò and Vanucci, 2000). The D4 filter, nevertheless, can be changed by the user. The signal is further decomposed to all analysing levels (MRA analysis) using the orthogonal wavelet decomposition method. The orthogonal property of Daubechies wavelets allows the reconstruction of the signal. The outcome is the reconstructed wavelet-transform profiles of the population genetic parameter along the sequence.

Visualization

VariScan permits the visualization of the results through available genomic browsers. For instance, *VariScan* can write the outcome on custom annotation track formats as the WIG format used in the Genome browser at UCSC (Kent et al., 2002) or the xyplot format in GBrowse (Stein et al., 2002), conferring a visual representation of the wavelet-transform profile integrated with current annotation tracks for the genome of interest. As a result, it is possible to relate statistic profile results (of nucleotide diversity, LD, etc) with present annotated genomic features (i.e., specific genes, intergenic regions, haplotype information, etc) from available genome projects.

Genome Browser at UCSC (<http://genome.ucsc.edu/cgi-bin/hgGateway>)

GBrowse (<http://www.gmod.org/>)

6 Running Example

6.1 Instructions for Linux (command-line mode)

This section consists of three parts:

- 1) Using *VariScan* to obtain a statistic profile
- 2) Using *LastWave* for the MRA analysis
- 3) Visualization of the results

It is supposed that you have downloaded and extracted both **tar.gz** files in your **\$HOME** directory, go to the *VariScan* directory, and follow this steps in a bash shell. All the output files (***.vs**, ***.mra**, and ***.bed**) are also included in the **example_outputs/** directory.

1) Using VariScan to obtain a statistic profile

Run *VariScan* to obtain estimates of the statistics (this will take a few seconds).

Running options

```
variscan input_data_file input_config_file > output.vs
```

Example:

```
[bash]$ cd variscan
[bash]$ ./variscan examples/test1.phy examples/test1.conf > myresults.vs
```

'test1.phy' is a phylip computer-simulated data file (10 sequences and 2000000 positions).
 'test1.conf' is the config file with the specified options to run **variscan**.
 'myresults.vs' File with the output

Running **variscan** you will obtain the output file ('myresults.vs').

2) Using LastWave for the MRA analysis

Run *LastWave* to obtain the MRA decomposition analysis (this will take from seconds to a few minutes).

The output from Step 1 ('myresults.vs') is the input for *LastWave*. In the present example we will use Pi values (3rd column for the reference positions, and the 11th column with Pi - π - results).

Specify the *LastWave* path:

```
[bash]$ export LWSOURCEDIR=$HOME/LastWave_2_0_4/scripts
[bash]$ export LWPATH=$HOME/LastWave_2_0_4/
```

Running options

```
perl runMRA.PLS input_file.vs file.mraconf
```

Example:

```
[bash]$ cd variscan
[bash]$ perl scripts/runMRA.PLS myresults.vs examples/test1.mraconf
```

'myresults.vs' Output from Step 1
 'test1.mraconf' is the config file with the specified options to the MRA analysis.

In this config file, you should specify (1) information to be extracted from the *VariScan* output (***.vs**); and (2) information needed for the *LastWave* software (mother wavelet, MRA levels, etc.) Now, in **test1.mraconf**, we have specified:

```
'stat_col = 11' Column with results of the desired statistic ( $\pi$  in our example)
'ref_col = 3'   Column with the reference position
'sub_a = 6'    Low-frequency bands from (1 to 5) levels
'sub_b = 9'    Frequency bands from (6 to 8) levels
'sub_c = 12'   Frequency bands from (9 to 11) levels. The last grouping will be (13 to 17)
```

Running **runMRA.PLS** you will obtain the output file (**'myresults.mra'**).

3) Visualization of the results (using the UCSC genome browser).

(this step will take from seconds to a few minutes).

Run **mra2bed.PLS** to obtain a file formatted according to the custom tracks (wiggle) at the UCSC genome browser.

Running options

```
perl mra2bed.PLS input_file.mra -chr chrXX
```

'-chr chrXX' Indicates the chromosome id for the UCSC browser

Example. Let me suppose that your polymorphism data is in chicken chromosome 13.

```
[bash]$ cd variscan
[bash]$ perl scripts/mra2bed.PLS myresults.mra -chr chr13
```

Running **mra2bed.PLS** you will obtain the output file (**'myresults.bed'**).

Open your web browser at <http://genome.ucsc.edu/cgi-bin/hgGateway>

Select the chicken genome (Feb. 2004).

Click on "Add Your Own Custom Tracks"

Submit the "bed" file (**'myresults.bed'**)

You will see four custom plots in the upper part of the panel, corresponding to the signal reconstruction of low-frequency bands, with information from: i) 1 to 5 MRA levels; ii) 6 to 8 MRA levels; iii) 9 to 11 MRA levels; and iv) 12 to 17 MRA levels.

4) Using the gff2bdf script.

gff2bdf.pl is a Perl script that parses through a generic/general feature format files (GFF) and creates a block data file (BDF) with the specified features that can be used by VariScan.

The script was designed to parse GFF files of version 3 as described in:

<http://flybase.net/annot/gff3.html>

but it should also be able to process files of version 2:

http://www.sanger.ac.uk/Software/formats/GFF/GFF_Spec.shtml

The script is invoked by:

```
perl gff2bdf.pl INPUT_FILE OUTPUT_FILE
```

where **INPUT_FILE** is the GFF file that should be parsed and **OUTPUT_FILE** is the BDF file created from the process.

The menu structure

The script comes with a menu that lets you choose the features of interest which should be filtered from the file. One built in feature is CDS, or coding sequence (menu item [1]). After choosing this feature you can define the minimal and maximal length of the coding regions you wish to filter as well as the strand the CDS should be located on. Then the script will parse through the GFF file and write all coding regions that correspond to your choices to the BDF file.

The custom feature option (menu item [3]) works similarly. But instead of looking for coding sequences, you can define the feature it should filter.

The non-coding regions option (menu item [2]) allows a more complex filtering operation and will be explained in detail.

Filtering non-coding sequences

When filtering non-coding regions, you will be given the choice to extract all non-coding regions (introns and intergenic regions) [1], only introns [2] or only intergenic regions [3].

Using options [1] and [2] works analogous to the filtering process as described above. When filtering intergenic regions, you will have additional options to choose from.

Filtering intergenic regions

When choosing to filter regions within defined size limits [1], you can define specific size limits as for the other features. After that, instead of defining the strand your feature should be located on, you can specify the strand location of the genes flanking the intergenic region.

You can also choose to look at blocks of a fixed size within an intergenic region [2]. You first have to set the size of the block (in bp) and then define where this block should be located within an intergenic region. You can either position it relative to the flanking genes or set it to be exactly in the middle in between the flanking genes. The orientation of the flanking genes can then also be specified.

Option [3] lets you define intergenic regions that have a specific distance to the genes that are flanking. The script will then only write intergenic regions that have the defined distance to the flanking genes. The orientation of the flanking genes can then also be specified.

6.2 Instructions for the VariScan GUI

Alternatively to running the *VariScan* package manually (the command line mode), you can use the graphical user interface (GUI) to control the programs.

The GUI is a front-end written in Java that lets you control *VariScan* and its associated programs in a simple point-and-click fashion. It runs on all systems that have Java Runtime Environment (JRE) version 1.4.2 or higher installed (tested on Windows XP, Mac OS X and Linux).

If you currently don't have a JRE installed, you can get it from:

<http://java.sun.com/>

Also note that if you want to run the *LastWave* or visualize the results on a genome browser (as GBrowse or UCSC genome browser) you must have installed the PERL programming language on your system. PERL is usually installed by default on Mac OS X and Unix/Linux systems. If you don't have PERL installed on your Windows computer you can get it from:

<http://www.activestate.com/Products/ActivePerl/>

Running the GUI

The *VariScan* GUI comes as a Java archive (JAR) called “**VariScanGUI.jar**”. In most systems JAR-files are automatically associated with the JRE, so simply double-clicking on the file should start the program. If this does not work, the GUI can be started in a terminal manually by issuing the following command:

```
java -jar VariScanGUI.jar
```

If this still doesn't work, you probably haven't set up the JRE correctly. Please make sure that Java is installed on your machine.

Setting up the GUI

When running the GUI for the first time, you must specify the directory location of the programs. This is done by choosing “**Options**” and then “**Settings**” from the menu bar. On this screen you locate the files and directories used by the *VariScan* package. Look for them by simply clicking the “**Browse**” button. There are 5 fields you have to specify:

VariScan scripts directory:

The directory where the PERL scripts used for running the *LastWave* and the genome browsers components (see section 4.1) are located. It is a directory called “**scripts**” inside the *VariScan* folder.

VariScan binary:

The actual *VariScan* executable, normally called “**variscan.exe**” on Windows systems, or simply “**variscan**” on other systems.

LastWave directory:

This is the main directory of the *LastWave* program. It is usually called “**LastWave_2_0_4**”.

LastWave source directory:

The directory containing *LastWave*-specific scripts. It is a directory called “**scripts**” inside your *LastWave* directory.

LastWave binary:

The actual *LastWave* executable. It is normally called “**lw**”. Precompiled executables for multiple systems can be found in the “**bin**” folder of the *LastWave* directory.

After defining these you can save the settings by clicking on the “**Save Settings**” button. This will ensure that the GUI will remember these settings the next time it is started.

If you want to run *LastWave*, you also have to make sure to set the **LWPATH** and **LWSOURCEDIR** environment variables (also see section 6.2 of the running example).

Running the programs

The programs are run in the “**Tasks**” / “**Run Analysis**” window. It is divided into three main parts, controlling the three components of the *VariScan* package. These three parts are also described in the running example (section 6).

VariScan analysis

This executes *VariScan* and calculates population genetic parameters from a sequence alignment file. The needed fields are:

Sequence alignment file:

The file containing the sequence alignment data; used as the input for *VariScan*. The sequence alignment file must be one of the formats as described in section 3.

VariScan config file (*.conf):

The file containing the configuration parameters for *VariScan*. If you don't have a config file yet or want to create a new one, simply click on "**Create**". This will open a file chooser dialogue box. Here you can enter the name of the file you wish to create. After you have loaded or created a config file you can edit it by clicking the "**Edit**" button. This will open up an editor that lets you change the parameters inside the file. The parameters are explained in detail in section 4.2. Clicking on "**Save**" saves the changes made to the file, clicking on "**Cancel**" discards all changes.

VariScan output file:

The file where the output should be written. If you want to save your output to a new file simply enter the name while in the file chooser.

LastWave analysis

Since there is no native Windows *LastWave* executable yet, this component will not work under Windows. If you want to run *LastWave* on a Windows machine you must do it manually from within the **cygwin** environment (see the *LastWave* documentation for details). This will be changed once a native Windows executable becomes available.

The *LastWave* component takes as input the output from *VariScan* and calculates a reconstructed wavelet-transform profile of a population genetic parameter. The following files are needed:

LastWave input file (*.vs):

This file is a *VariScan* output file where population genetic parameters using a sliding window were calculated.

LastWave config file (*.mraconf):

Configuration file for the wavelet transformation. Parameters are described in section 4.3. As for the *VariScan* config file you may also create and edit these files. Note that in the editor you cannot change the "*LastWave* directories" (see section 4.3). These parameters are taken directly from the "**Options**" / "**Settings**" of the GUI.

LastWave output file:

The file where the MRA output should be written. This file is automatically created based on the name of the input file and cannot be changed.

UCSC Genome Browser and GBrowser component

This component creates a file that can be visualized using genome browsers. You can either visualize results that were obtained by a multi resolution analysis using *LastWave*, or statistics calculated under the sliding window method from *VariScan*. It has the following fields:

Visualization input file:

This is either the output of a multi resolution analysis performed by *LastWave* (*.mra), or a *VariScan* output file (usually *.vs) where statistics were calculated using a sliding window analysis.

Input file type:

This specifies which of the two possible input formats is used.

VariScan statistic column(s):

When using a *VariScan* file as an input for the visualization process you will have to define which statistics (respective columns) within your file should be visualized. A description of the possible columns can be found at the end of the appendix. The input here is simply the number of the column. You can also choose to visualize multiple columns. Then the column numbers should be entered separated by a “:” character (for example enter “11:13:14” if you want to visualize the respective columns).

Visualization output file (for example, *.bed):

The file name to be used for visualization. This file is automatically created based on the name of the input file and cannot be changed.

Output file type:

Lets you choose between two formats of visualization files: Either a UCSC wiggle bed, or a GBrowse intensity xyplot.

Chromosome ID:

The chromosome name (identifier) for the analyzed chromosome (for example “13” if the data analyzed resides on chromosome 13).

Controlling the components

Each one of these components can be turned on, or of by clicking on the corresponding checkboxes. If more than one component is activated, they will be called sequentially: First *VariScan*, then *LastWave* and finally the browser visualization. This allows the user to perform the complete task of creating a visualization file from a sequence alignment file by a few mouse-clicks. To facilitate this process file names are automatically passed on to subsequent components if they are activated.

Clicking the “**Run**” button starts the process.

GUI known bugs and work in progress

General layout and appearance is still in development. It might change in future versions. Progress bar is not working. After clicking “**Run**” it seems as if the GUI freezes. The progress bar and status indicator at the bottom do not give feedback about the actual status of the program. The validity of the input files is not thoroughly checked. The GUI calls whatever files the user specifies. Nevertheless, if errors occur during the process the GUI will display the error output of the individual components.

7 References

- Aguadé, M., Rozas, J. and Segarra, C. (2004) Inferring the action of natural selection from DNA sequence comparisons: data from *Drosophila*. In Moya, A. and Font, E. (eds), *Evolution: from Molecules to Ecosystems*, Oxford Univ. Press, pp. 11-19.
- Ardell, D.H. (2004) SCANMS: adjusting for multiple comparisons in sliding window analysis of neutrality test statistics. *Bioinformatics*, 20, 1986-1988.
- Arneodo, A., Bacry, E., Graves, P.V. and Muzy, J.F. (1995) Characterizing long-range correlations in DNA sequences from wavelet analysis. *Phys. Rev. Lett.*, 74, 3293-3296.
- Daubechies, I. (1992) *Ten lectures on wavelets*, SIAM, Philadelphia.
- Depaulis, F. and Veuille, M. (1998) Neutrality tests based on the distribution of haplotypes under an infinite-site model. *Mol. Biol. Evol.*, 15, 1788-1790.
- Fares, M.A., Elena, S.F., Ortiz, J., Moya, A. and Barrio, E. (2002) A Sliding Window-Based Method to Detect Selective Constraints in Protein-Coding Genes and Its Application to RNA Viruses. *J. Mol. Evol.*, 55, 509-521.
- Filatov, D.A. (2002) ProSeq: A software for preparation and evolutionary analysis of DNA sequence data sets. *Mol. Ecol. Notes*, 2, 621-624.
- Fu, Y.-X. and Li, W.-H. (1993) Statistical tests of neutrality of mutations. *Genetics*, 133, 693-709.
- Fu, Y.-X. (1997) Statistical tests of neutrality of mutations against population growth, hitchhiking and background selection. *Genetics*, 147, 915-925.
- Gabriel, S.B., Schaffner, S.F., Nguyen, H., Moore, J.M., Roy, J., Blumenstiel, B., Higgins, J., DeFelice, M., Lochner, A., Faggart, M., Liu-Cordero, S.N., Rotimi, C., Adeyemo, A., Cooper, R., Ward, R., Lander, E.S., Daly, M.J., and Altshuler, D. (2002) The structure of haplotypes in the human genome. *Science*, 296, 2225-2229.
- Hill, W.G. and Robertson, A. (1968) Linkage disequilibrium in finite populations. *Theor. Appl. Genet.*, 38, 226-231.
- Höhl, M., Kurtz, S. and Ohlebusch, E. (2002) Efficient multiple genome alignment. *Bioinformatics*, 18, S312-S320.
- Hudson, R.R. (1990) Gene genealogies and the coalescent process. *Oxf. Surv. Evol. Biol.*, 7, 1-44.
- Innan, H., Padhukasahasram, B. and Nordborg, M. (2003) The Pattern of Polymorphism on Human Chromosome 21. *Genome Res.*, 13, 1158-1168.
- Kaplan, N.L., Hudson, R.R., Langley, C.H. (1989) The "hitchhiking effect" revisited. *Genetics*, 123, 887-899.
- Kelly, J.K. (1997) A test of neutrality based on interlocus associations. *Genetics*, 146, 1197-1206.
- Kent, W.J., Sugnet, C.W., Furey, T.S., Roskin, K.M., Pringle, T.H., Zahler, A.M. and Haussler, D. (2002) The Human Genome Browser at UCSC. *Genome Res.*, 12, 996-1006.

- Kim, Y. and Stephan, W. (2002) Detecting a local signature of genetic hitchhiking along a recombining chromosome. *Genetics*, 160, 765-777.
- Kimura, M. (1983) *The Neutral Theory of Molecular Evolution*. Cambridge University Press, Cambridge, MA.
- Kingman, J.F.C. (1982) On the genealogy of large populations. *J. Appl. Prob.*, 19A, 27-43.
- Kreitman, M. (2000) Methods to detect selection in populations with applications to the human. *Annu. Rev. Genom. Hum. Genet.*, 1, 539-559.
- Kreitman, M. and Hudson, R.R. (1991) Inferring the evolutionary histories of the *Adh* and *Adh-dup* loci in *Drosophila melanogaster* from patterns of polymorphism and divergence. *Genetics*, 127, 565-582.
- Lewontin, R.C. (1964) The interaction of selection and linkage. I. General considerations: heterotic models. *Genetics*, 49, 49-67.
- Liò, P. (2003) Wavelets in bioinformatics and computational biology: state of art and perspectives. *Bioinformatics*, 19, 2-9.
- Liò, P. and Vanucci, M. (2000) Finding pathogenicity islands and gene transfer events in genome data. *Bioinformatics*, 16, 932-940.
- McDonald, J.H. (1998) Improved tests for heterogeneity across a region of DNA sequence in the ratio of polymorphism to divergence. *Mol. Biol. Evol.*, 15, 377-384.
- Mallat, S.G. (1989) A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11, 674-693.
- Mallat, S. (1999) *A Wavelet Tour of Signal Processing*; 2 edition. Academic Press, San Diego.
- McGill, K. and Taswell, C. (1993) Wavelet transform algorithms for finite-duration discrete-time signals. In Meyer, Y. and Roques, S. (eds), *Proceedings of the International Conference on Wavelets and Applications*, Editions Frontières, Toulouse, France, pp. 221-224.
- Nei, M. (1987) *Molecular Evolutionary Genetics*. Columbia University Press, New York.
- Nordborg, M. (2001) Coalescent theory. In Balding, D., Bishop, M. and Cannings, C. (eds), *Handbook of Statistical Genetics*, John Wiley & Sons, Chichester, U.K., pp. 179-212.
- Olson, S. (2002) Seeking the signs of selection. *Science*, 298, 324-325.
- Patil, N., Berno, A.J., Hinds, D.A., Barrett, W.A., Doshi, J.M., Hacker, C.R., Kautzer, C.R., Lee, D.H., Marjoribanks, C., McDonough, D.P., Nguyen, B.T.N., Norris, M.C., Sheehan, J.B., Shen, N., Stern, D., Stokowski, R.P., Thomas, D.J., Trulson, M.O., Vyas, K.R., Frazer, K.A., Fodor, S.P.A. and Cox, D.R. (2001) Blocks of Limited Haplotype Diversity Revealed by High-Resolution Scanning of Human Chromosome 21. *Science*, 294, 1719-1723.
- Quesada, H., Ramírez, U.E.M., Rozas, J. and Aguadé, M. (2003) Large-Scale Adaptive Hitchhiking upon High Recombination. *Genetics*, 165, 895-900.
- Rosenberg, N.A. and Nordborg, M. (2002) Genealogical trees, coalescent theory, and the analysis of genetic polymorphisms. *Nat. Rev. Genet.*, 3, 380-390.

Rozas,J. and Rozas.R. (1995) DnaSP, DNA sequence polymorphism: an interactive program for estimating Population Genetics parameters from DNA sequence data. *Comput. Appl. Biosci.*, 11, 621-625.

Rozas,J., Sánchez-DelBarrio,J.C., Messeguer,X. and Rozas,R. (2003) DnaSP, DNA polymorphism analyses by the coalescent and other methods. *Bioinformatics*, 19, 2496-2497.

Sabeti,P.C., Reich,D.E., Higgins,J.M., Levine,H.Z., Richter,D.J., Schaffner,S.F., Gabriel,S.B., Platko,J.V., Patterson,N.J., McDonald,G.J., Ackerman,H.C., Campbell,S.J., Altshuler,D., Cooper,R., Kwiatkowski,D., Ward,R., and Lander,E.S. (2002) Detecting recent positive selection in the human genome from haplotype structure. *Nature*, 419, 832-837.

Schneider,S., Roessli,D., and Excoffier,L. (2000) Arlequin: A software for population genetics data analysis. Ver 2.000. Genetics and Biometry Lab, Dept. of Anthropology, University of Geneva.

Stein,L.D., Mungall,C., Shu,S., Caudy,M., Mangone,M., Day,A., Nickerson,E., Stajich,J.E., Harris,T.W., Arva,A. and Lewis,S. (2002) The generic genome browser: a building block for a model organism system database. *Genome Res.*, 12, 1599-1610.

Sweldens,W. (1996) Wavelets: what next? *Proc. IEEE*, 84, 680–685.

Tajima,F. (1989) Statistical method for testing the neutral mutation hypothesis by DNA polymorphism. *Genetics*, 123, 585-595.

Tajima,F. (1991) Determination of window size for analyzing DNA sequences. *J. Mol. Evol.*, 33, 470-473.

Vilella, A. J., Blanco-Garcia, A., Hutter, S. and Rozas, J. (2005). VariScan: Analysis of evolutionary patterns from large-scale DNA sequence polymorphism data. *Bioinformatics*, 21, 2791-2793.

8 Appendix

Block Data File Format

Block Data Files allow the user to specifically define certain blocks (genomic regions) within the alignment that should be analyzed. A block data file is a simple ASCII text file that contains the start and end coordinates of a block as well as a name or description for that block. Here is a short example:

```
1001 6000 Block #1 (rp49 coding region)
8501 13000 Block #2 (exon 3, gen XX)
13001 18000 Block #3
14001 16000 Block #4
```

There are three columns: The first column defines the start coordinate, the second column defines the end coordinate and the third column defines the name of a block. The columns are separated by whitespace. Note that the name column may also contain whitespaces.

VariScan will then only analyze these 4 blocks in the alignment. All other parts of the alignment will be ignored. If a sliding window analysis is to be performed, it will be done within each block separately.

The blocks defined within the block data file must be ordered by their start coordinates, otherwise *VariScan* will abort with an error message. Note that in our example blocks #3 and #4 overlap. This is allowed as long as the start coordinates are in ascending order.

When using a block data file the global **StartPos** and **EndPos** parameters are still in effect. If, for example, **StartPos** = 2001 and **EndPos** = 15000, then in our case Block #1 will only be analyzed from position 2001 to position 6000. Similarly, Block #3 will only be analyzed from positions 13001 to 15000 and Block #4 from positions 14001 to 15000.

As for the **StartPos** and **EndPos** parameters the coordinates defined in a block data file are sensitive to the **RefPos** parameter. If **RefPos** = 0, this means that all coordinates given in the block data file refer to total positions in the alignment. If **RefPos** = 1, then all coordinates are positions in the reference sequence.

Outgroup Information

When using **RunMode** 21 or 22 an outgroup must to be defined in order to carry out the analysis. This is done with the **Outgroup** parameter. By setting this parameter to **first** or **last** you can easily define the first or last sequence in the alignment to be the outgroup. If another sequence of the alignment is the outgroup, you can define it with a vector similar to the one used for the **SeqChoice** parameter. If, for example, you have an alignment of 8 individuals with the 4th individual being the outgroup and all other individuals belonging to the ingroup, then the **Outgroup** parameter should be set like this:

```
Outgroup = 0 0 0 1 0 0 0 0
```

All ingroup individuals are labelled with a 0, the outgroup is labelled with a 1. *VariScan* even allows you to define multiple outgroups within an alignment. In this case the outgroups are labelled with numbers describing their phylogenetic distance to the ingroup, i.e. the higher the number, the more distant the relationship. A short example:

Outgroup = 3 0 0 1 0 0 2 0

In this example we have an alignment with 5 ingroup individuals and 3 outgroups. Individual 4 is the closest outgroup followed by individual 7. Individual 1 is the most distant outgroup. Currently *VariScan* only makes use of the closest outgroup, i.e the outgroup with the lowest number. Other outgroups are ignored. This might change in future versions.

Outgroup and LD Analysis

When defining an outgroup the way the LD statistics are calculated changes. Without an outgroup the derived and ancestral state of a polymorphic site are inferred by the frequency of the segregating alleles. The major allele is set to the ancestral, the minor allele to the derived state. With this definition the coupling and repulsion phases for each pair wise comparison are calculated.

When an outgroup is defined this is different. Now it is possible to infer the ancestral state of a polymorphic site by using the outgroup for comparison. If it is not possible to infer the ancestral state by comparison to the outgroup, the site will be discarded for analysis. Since this might happen quite frequently, the number of sites used for calculating LD statistics is usually higher when not using an outgroup, even though the inclusion of an outgroup sequence adds more information to the analysis.

IndivNames and MAF files

MAF files are organized in blocks of information (paragraphs). If you look at the `'maf_example.maf'` given in the `data` directory, you will see three paragraphs. Notice that the number of aligned sequences differs from paragraph to paragraph; the first paragraph has 3 sequences (sequence1, sequence2 and sequence3), the second paragraph has 4 sequences and third one 5 sequences. This fluctuating number of sequences poses a problem when analyzing the data. The program does not know the total number of individuals present in the MAF file, by just looking at the first paragraph. So the individuals have to be defined in the `IndivNames` parameter. For the given example it has to look like this:

IndivNames = sequence1 sequence2 sequence3 sequence4 sequence5

This ensures that *VariScan* is aware that individuals are missing in paragraphs containing less than 5 sequences aligned. The program will then automatically generate a sequence for the missing individuals consisting solely of "N"s (because the sequence is unknown). For our example the actual alignment that *VariScan* will work with then looks like this for the first paragraph:

```
sequence1      ctggagattctta-ttagtgatttgggctggggc-ctggccatgtgtatt
...
sequence2      cTAGGGAGTCTTAGTCAAAGGTTTGGACCAAGTCCCTGGCCATGCAGATC
...
sequence3      CTGGAGAGTCTTATTTGAAGGGTTGGACCAAGCCACTGGCCATGTAGATC
...
sequence4      NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
...
sequence5      NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
...
```

This has multiple consequences for the analysis. If coordinates are supposed to be based on the reference sequence (by setting `RefPos = 1`), then the reference sequence (as defined in `RefSeq`) has to be present in ALL paragraphs of the MAF file. If this is not the case, *VariScan* will abort with

an error, as soon as it reaches a paragraph not containing the reference sequence. *VariScan* will also abort, if it finds a paragraph containing an individual that is not defined by the **IndivNames** vector.

If MAF files contain many paragraphs with a reduced number of individuals, this will lead to a large number of discarded sites if **CompleteDeletion = 1**. This can be avoided by using the **NumNuc** parameter (see below).

Handling Gaps / Missing Data

VariScan only uses positions in the alignment that match the criteria defined by the parameters **CompleteDeletion**, **NumNuc** and **FixNum**. If positions do not match these criteria, they will be discarded and not used for analysis. Here are some examples.

Let us assume **CompleteDeletion = 1**:

	D	D	D	DD	D
Seq1	CNAAAGAGCA	TAAAAA	AAAAA	AAAAA	AAAAA
Seq2	AATAAANACG	A--	AAAAANA		
Seq3	AAT-AA-ATA	A--	TACAANA		
Seq4	AAA-AAAAAA	AA-	TAAAAANA		
Seq5	CAA-AGACAA	AG-	AAACAAA		
Seq6	AATAAAAAAG	AAATAACAAA			

Every site containing a gap ("-") and/or an ambiguous nucleotide ("N") will be ignored in the analysis. These discarded sites are indicated by the "D"s. **CompleteDeletion = 1** is obligatory for **RunMode = 31**.

If **CompleteDeletion = 0**, sites containing gaps/ambiguities can be included in the analysis. This depends on value defined in **NumNuc**, which stands for "Number of valid Nucleotides". All sites containing the defined number or more will be kept for analysis. Let us assume that **NumNuc = 4**:

	D	D	D
Seq1	CNAAAGAGCA	TAAAAA	AAAAA
Seq2	AATAAANACG	A--	AAAAANA
Seq3	AAT-AA-ATA	A--	TACAANA
Seq4	AAA-AAAAAA	AA-	TAAAAANA
Seq5	CAA-AGACAA	AG-	AAACAAA
Seq6	AATAAAAAAG	AAATAACAAA	

In the example, site 12 contains 2 gaps but it also contains 4 valid nucleotides. In this case only the gaps are discarded, but the remaining 4 nucleotides are kept for analysis. Sites 4, 13 and 19 are completely discarded, because they contain less than 4 valid nucleotides.

Therefore, the sample size can vary among different sites. Most sites have a sample size of 6, while others have sample sizes of only 4 or 5. This biases the estimation of certain statistics, Tajima's *D*, Fu and Li's *D** and *F**. These statistics were developed assuming constant sample size over the whole alignment. For variable sample size among sites (as in our example) *VariScan* will calculate the statistics by using as sample size, the average sample size over all sites. Note, however, that these test-statistics do NOT behave like the standard ones. To obtain the confidence intervals for these statistics you should conduct coalescent simulations taking into account the average number of discarded nucleotides per site (this value is included in the *VariScan* output; see below).

With **CompleteDeletion = 1** you will overcome this problem: i.e., the sample size will be fixed among sites. Alternatively, the parameter **FixNum** can be used. If **FixNum = 0**, the

behaviour is as stated above. If **FixNum = 1**, *VariScan* will use, for every site, a fixed sample size, as defined by **NumNuc**. If a site has a larger sample size than **NumNuc**, *VariScan* will randomly discard the extra nucleotides.

Let's look at the example again. The parameters are **NumNuc = 4**, **FixNum = 1**:

```

                D      D      D
Seq1      CNAAGAGCA TAAAAAAAAA
Seq2      AATAAANACG A--AAAAANA
Seq3      AAT-AA-ATA A--TACAANA
Seq4      AAA-AAAAAA AA-TAAAAANA
Seq5      CAA-AGACAA AG-AAACAAA
Seq6      AATAAAAAAG AAATAACAAA
    
```

Sites 4, 13 and 19 are still discarded; they contain less than 4 valid nucleotides. Site 1 has 6 valid nucleotides, so 2 of those nucleotides will be randomly discarded. Site 2 contains one "N" which is automatically discarded. Out of the remaining 5 valid nucleotides 1 is randomly discarded to ensure the fixed sample size of 4. This procedure is repeated for all sites in the alignment. Since this leads to constant sample size over all analyzed sites, the standard statistics can be calculated.

This feature is especially useful, if **CompleteDeletion = 1** would lead to a large loss of sites. This could be the case, if e.g. only one sequence contains a long stretch of "N"s or gaps, while all other sequences carry valid nucleotides.

Sliding Window Option

VariScan allows you to perform a sliding window analysis along the alignment. Depending on the settings in the config file, the behaviour of the analysis will change.

The width (length) of the window

Let us assume you have set **WidthSW = 10**. The value of 10 has different meanings based on the value given in the **WindowType** parameter:

WindowType = 0:

This means that the units measuring the width of window are simply the number of consecutive sites in the alignment.

```

Seq1      CNAAG-CA TAAAAAAAAA
Seq2      AATAAANACG A--AAAAANA
Seq3      AAT-AANATA AA-TACAANA
Seq4      AAA-AANAAA AA-TAAAAANA
Seq5      NAA-AGACAA AG-AAACAAA
          *****
    
```

The window is indicated by asterisks (*). The first 10 sites of the alignment are covered.

WindowType = 1:

Here, the width of the window is based on the number of net sites, i.e., excluding all discarded sites. Discarded sites depend on the values of parameters **NumNuc** and **CompleteDeletion** (see handling gaps / missing data part in this appendix). Let us assume that **NumNuc = 4**, meaning all sites with less than 4 valid bases will be discarded:

```

                D  D
Seq1      CNAAG-CA TAAAAAAAAA
Seq2      AATAAANACG A-GAAAAANA
Seq3      AAT-AANATA AA-TACAANA
    
```

```
Seq4      AAA-AANAAA AA-TAAAAAA
Seq5      NAA-AGACAA AG-AAACAAA
          ***** **
```

The discarded sites are indicated by the "D"s. Site 4 contains only 2 valid bases and 3 gaps, while site 7 contains 2 valid bases and 3 ambiguous bases. The window now covers 12 sites in the alignment, but a total of 10 net sites.

WindowType = 2:

Here, the window will cover the number of polymorphic sites given by the **WidthSW** parameter. A polymorphic site is defined as a site containing more than one variant, excluding gaps or ambiguities. Note that polymorphic sites which are discarded are NOT taken into account. Let us again look at our alignment, with **NumNuc = 4**:

```
          P PD PDPPP PPDP P
Seq1      CNAA-GA-CA TAAAAAAAAA
Seq2      AATAAANACG A-GAAAAANA
Seq3      AAT-AANATA AAGTACAANA
Seq4      AAA-AANAAA AA-TAAAAAA
Seq5      NAA-AGACAA AG-AAACAAA
          ***** **
```

The polymorphic sites are indicated by "P"s. The window now covers 16 sites in the alignment, including 10 polymorphic sites. Note that site 13 is NOT counted as a polymorphic site (even though it contains "A" and "G"), because it is discarded (it contains less than 4 valid nucleotides).

WindowType = 3:

The window is now based on positions in the reference sequence. The window is set, so that the amount of sites in the reference sequence corresponds to the value given in **WidthSW**. Let us assume that Seq1 is our reference sequence:

```
          R R
Seq1      CNAA-GA-CA TAAAAAAAAA
Seq2      AATAAANACG A-GAAAAANA
Seq3      AAT-AANATA AA-TACAANA
Seq4      AAA-AANAAA AA-TAAAAAA
Seq5      NAA-AGACAA AG-AAACAAA
          ***** **
```

The reference sequence contains two gaps, indicated by "R"s. The window is extended by two sites and now covers 12 sites in the alignment, but exactly 10 positions in the reference sequence. Note that discarded or polymorphic sites do not influence the calculation of the window.

The slide of the window

The slide of window follows the same rules (and units) as those for the width. So let us assume **JumpSW = 5**. For **WindowType = 0** the window simply slides 5 sites along the alignment to the right. For **WindowType = 1** it slides over 5 net sites. For **WindowType = 2** it slides over 5 polymorphic sites, and for **WindowType = 3** it slides over 5 positions in the reference sequence.

The coordinates of the window

For each window, *VariScan* displays two sets of coordinates: The start, end and midpoint in the alignment, as well as the start, end and midpoint in the reference sequence.

The midpoint of the window is calculated differently for each WindowType. For **WindowType = 0** the midpoint simply is the middle point between the start and the end sites of the window (alignment coordinates). For **WindowType = 3** it is the middle point between the start and the end in the reference sequence. For **WindowType = 1** the midpoint is the coordinate for the middle net

site. So if you have set **WidthSW = 15**, it is the coordinate of the 8th net site. For **WindowType = 2** it is the coordinate of the middle polymorphic site (the 8th polymorphic site, if **WidthSW = 15**).

Here is an example of how the reference coordinates are calculated. For a window covering from sites 3 to 17 (in the alignment) and a **WindowType = 0**:

```

                S      M      E
Seq1      A----A--N-  AA--NAAA-A
Seq2      AAAAA-ATTT  --TTTT-AAA
Seq3      AA-ATTAA-A  TTTAAT-AAA
Seq4      AA-AATAAGA  TTTAATTAAA
Seq5      AA-ATCAA-A  TTTAACCACA
                *****
    
```

The start, midpoint and end of the window are indicated with "S", "M" and "E".
So the coordinates in the alignment are: Start 3, Mid 10, End 17.

For the reference sequence, the start position is the first reference base inside the window. In this case the second base. The last reference base in the window is the 8th base. As you can see, the midpoint falls on a gap in the reference sequence. In this case the last valid position before the midpoint is given, which here would be the 3rd base.
So the coordinates in the reference sequence are: Start 2, Mid 3, End 8.

Output of the analyses

Depending on the **RunMode** value, *VariScan* will calculate different statistics and will display them in columns. The results for each window (sliding window analysis) will be displayed in rows. *VariScan* will also indicate the total results for the whole analyzed region.

For all sliding window analyses:

#	Column	Description
1	RefStart	The coordinates in the reference sequence of the first position of a sliding window
2	RefEnd	The coordinates in the reference sequence of the last position of a sliding window
3	RefMid	The coordinates in the reference sequence of the midpoint of a sliding window
4	Start	The coordinates in the alignment of the first position of a sliding window
5	End	The coordinates in the alignment of the last position of a sliding window
6	Midpoint	The coordinates in the alignment of the midpoint of a sliding window
7	NumSites	Net size of window (Non-discarded sites used for analysis)
8	Missing	Average number of missing (discarded) nucleotides per site (only printed for CompleteDeletion = 0 and FixNum = 0)

RunMode = 11:

#	Column	Description
9	S	Number of segregating sites
10	Eta	Total (minimum number) number of mutations (η)
11	Pi	Nucleotide diversity (π); i.e., the average pairwise nucleotide differences per site
12	Theta	Waterson's estimator of nucleotide diversity per site (based on Eta for UseMuts = 1 , based on S for UseMuts = 0)

RunMode = 12:

#	Column	Description
9	S	Number of segregating sites
10	Eta	Total (minimum number) number of mutations (η)

11	Eta_E	Number of singletons (for UseMuts = 1) Number of sites containing singletons (for UseMuts = 0)
12	Pi	Nucleotide diversity (π)
13	Theta	Waterson's estimator of nucleotide diversity per site (based on Eta for UseMuts = 1 , based on <i>S</i> for UseMuts = 0)
14	Tajima_D	Tajima's <i>D</i> statistic
15	FuLi_Dstar	Fu & Li's <i>D</i> * statistic
16	FuLi_Fstar	Fu & Li's <i>F</i> * statistic

RunMode = 21:

#	Column	Description
9	S	Number of segregating sites
10	Eta	Total (minimum number) number of mutations (η)
11	S_Inter	Number of sites differing between ingroup and outgroup (including ingroup polymorphic sites)
12	Pi	Nucleotide diversity (π)
13	K	Divergence per site (Jukes-Cantor corrected)

RunMode = 22:

#	Column	Description
9	S	Number of segregating sites
10	Eta	Total (minimum number) number of mutations (η)
11	Eta_E	Number of external mutations, i.e. number of derived singletons
12	Pi	Nucleotide diversity (π)
13	FuLi_D	Fu & Li's <i>D</i> statistic
14	FuLi_F	Fu & Li's <i>F</i> statistic
15	FayWu_H	Fay & Wu's <i>H</i> statistic

RunMode = 31:

#	Column	Description
9	LD_sites	Number of polymorphic sites used for linkage disequilibrium analysis (!)
10	D	<i>D</i> value averaged over all comparisons in the window
11	D	Absolute <i>D</i> value averaged over all comparisons in the window
12	D'	<i>D'</i> value averaged over all comparisons in the window
13	D'	Absolute <i>D'</i> value averaged over all comparisons in the window
14	r^2	<i>r</i> ² value averaged over all comparisons in the window (i.e., <i>Z</i> _{ns} statistic)
15	h	Number of haplotypes
16	Hd	Haplotype diversity
17	Pi	Average pair wise distance per site(π)
18	Fu_Fs	Fu's <i>F</i> _S statistic

#, Column number (to be used in **stat_col** and **ref_col** parameters)

!, Only polymorphic sites that contain EXACTLY two variants are used for LD analysis. Sites containing 3 or more variants are ignored for LD analysis, but used when calculating haplotypes and Fu's *F*_S statistics.