Package ‘WhopGenome’
March 13, 2017

Type Package
Title High-Speed Processing of VCF, FASTA and Alignment Data
Version 0.9.7
Date 2017-03-10
Author Ulrich Wittelsbuerger [aut, cre], Heng Li [ctb], Bob Handsaker [ctb]
Maintainer Ulrich Wittelsbuerger <ulrich.wittelsbuerger@uni-duesseldorf.de>
Depends R (>= 1.8.0)
Suggests RMySQL, DBI, AnnotationDbi
Description Provides very fast access to whole genome, population scale variation data
from VCF files and sequence data from FASTA-formatted files.
It also reads in alignments from FASTA, Phylip, MAF and other file formats.
Provides easy-to-use interfaces to genome annotation from UCSC and Bioconductor and gene ontology data
from AmiGO and is capable to read, modify and write PLINK .PED-format pedigree files.
License GPL (>= 2)
SystemRequirements zlib headers and library
NeedsCompilation yes
LazyLoad yes
Copyright inst/COPYRIGHTS
Repository CRAN
Date/Publication 2017-03-13 17:10:56

R topics documented:

  WhopGenome-package .................................................. 4
  bgzf_compress .......................................................... 5
  fai_build ............................................................. 6
  fai_close ............................................................. 7
  fai_open ............................................................. 8
  fai_query2 ........................................................... 9
  fai_query4 ........................................................... 10
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>fai_reopen</td>
<td>11</td>
</tr>
<tr>
<td>tabix_build</td>
<td>12</td>
</tr>
<tr>
<td>tabix_close</td>
<td>14</td>
</tr>
<tr>
<td>tabix_getregion</td>
<td>15</td>
</tr>
<tr>
<td>tabix_open</td>
<td>16</td>
</tr>
<tr>
<td>tabix_read</td>
<td>17</td>
</tr>
<tr>
<td>tabix_reopen</td>
<td>18</td>
</tr>
<tr>
<td>tabix_restartregion</td>
<td>19</td>
</tr>
<tr>
<td>tabix_setregion</td>
<td>20</td>
</tr>
<tr>
<td>vcf_addfilter</td>
<td>21</td>
</tr>
<tr>
<td>vcf_buildindex</td>
<td>23</td>
</tr>
<tr>
<td>vcf_clearfilters</td>
<td>24</td>
</tr>
<tr>
<td>vcf_close</td>
<td>25</td>
</tr>
<tr>
<td>vcf_countSNPs</td>
<td>26</td>
</tr>
<tr>
<td>vcf_describefilters</td>
<td>27</td>
</tr>
<tr>
<td>vcf_eor</td>
<td>28</td>
</tr>
<tr>
<td>vcf_getChrom</td>
<td>29</td>
</tr>
<tr>
<td>vcf_getcontignames</td>
<td>30</td>
</tr>
<tr>
<td>vcf_getfieldnames</td>
<td>31</td>
</tr>
<tr>
<td>vcf_getheaderline</td>
<td>32</td>
</tr>
<tr>
<td>vcf_getnumcontigs</td>
<td>32</td>
</tr>
<tr>
<td>vcf_getregion</td>
<td>33</td>
</tr>
<tr>
<td>vcf_isINDEL</td>
<td>34</td>
</tr>
<tr>
<td>vcf_isSNP</td>
<td>35</td>
</tr>
<tr>
<td>vcf_open</td>
<td>36</td>
</tr>
<tr>
<td>vcf_parseNextSNP</td>
<td>36</td>
</tr>
<tr>
<td>vcf_readLineDF</td>
<td>37</td>
</tr>
<tr>
<td>vcf_readLineRaw</td>
<td>38</td>
</tr>
<tr>
<td>vcf_readLineVec</td>
<td>39</td>
</tr>
<tr>
<td>VCF_read_snp_diplo_bial_int_altpresence</td>
<td>40</td>
</tr>
<tr>
<td>vcf_reopen</td>
<td>41</td>
</tr>
<tr>
<td>vcf_restartregion</td>
<td>42</td>
</tr>
<tr>
<td>vcf_rule.disable</td>
<td>43</td>
</tr>
<tr>
<td>vcf_rule.setaction</td>
<td>44</td>
</tr>
<tr>
<td>vcf_rule.setcolumn</td>
<td>45</td>
</tr>
<tr>
<td>vcf_rule.setcomparison</td>
<td>46</td>
</tr>
<tr>
<td>vcf_rule.setfield</td>
<td>48</td>
</tr>
<tr>
<td>vcf_rule.setrefvalues</td>
<td>49</td>
</tr>
<tr>
<td>vcf_selectsamples</td>
<td>50</td>
</tr>
<tr>
<td>vcf_setregion</td>
<td>52</td>
</tr>
<tr>
<td>VCF_snpmat_diplo_bial_geno_filtered</td>
<td>53</td>
</tr>
<tr>
<td>vcf_valid</td>
<td>55</td>
</tr>
<tr>
<td>whop.eg.abbrevForOrganism</td>
<td>56</td>
</tr>
<tr>
<td>whop.eg.chromosome</td>
<td>56</td>
</tr>
<tr>
<td>whop.eg.eg_lookup</td>
<td>57</td>
</tr>
<tr>
<td>whop.eg.eg_lookupAll</td>
<td>57</td>
</tr>
<tr>
<td>whop.eg.eg_lookupSingle</td>
<td>58</td>
</tr>
<tr>
<td>whop.eg.eg_RevLookup</td>
<td>59</td>
</tr>
</tbody>
</table>
R topics documented:

whop.egenzyme .................................................. 59
whop.eg.fromAccnum ......................................... 60
whop.eg.fromAlias ............................................ 60
whop.eg.fromEnsembl ......................................... 61
whop.eg.fromEnsemblProt .................................... 61
whop.eg.fromEnsemblTrans ................................... 62
whop.eg.fromEnzyme .......................................... 62
whop.eg.fromGO .................................................. 63
whop.eg.fromGO2AllEgs ......................................... 64
whop.eg.fromOmim .............................................. 64
whop.eg.fromPath ............................................... 65
whop.eg.fromPmid ............................................... 65
whop.eg.fromRefseq ........................................... 66
whop.eg.fromUnigene ......................................... 66
whop.eg.fromUniprot .......................................... 67
whop.eg.genename .............................................. 67
whop.eg.golds .................................................... 68
whop.eg.installdb ............................................. 68
whop.eg.keggpathways ......................................... 69
whop.eg.load_orgdb ............................................ 69
whop.eg.Organism ............................................. 70
whop.eg.orgdb_loaded ......................................... 70
whop.eg.region .................................................. 71
whop.eg.selectOrganism ....................................... 72
whop.eg.toAccnum .............................................. 72
whop.eg.toAlias ................................................. 73
whop.eg.toEnsembl ............................................ 73
whop.eg.toEnsemblProt ....................................... 74
whop.eg.toEnsemblTrans ...................................... 74
whop.eg.toEnzyme ............................................. 75
whop.eg.toGO .................................................... 76
whop.eg.toOmim .................................................. 76
whop.eg.toPath .................................................. 77
whop.eg.toPmid .................................................. 77
whop.eg.toRefseq .............................................. 78
whop.eg.toUnigene ............................................ 78
whop.eg.toUniprot ............................................. 79
whop.go.all_genes_for_term ................................ 79
whop.go.connect ............................................... 80
whop.go.goid_like ............................................. 81
whop.go.is_obsolete_byid ..................................... 81
whop.go.is_obsolete_byname .................................. 82
whop.go.load ................................................... 82
whop.go.match .................................................. 83
whop.go.terms_match .......................................... 83
whop.go.term_ancestors ....................................... 84
whop.go.term_ancestors_similar ................................ 84
whop.go.term_children ........................................ 85
WhopGenome-package

High-speed, high-specialisation population-scale whole-genome variation and sequence data access

Description

WhopGenome provides read access to Variant Call Format files with maximum speed by means of C functions with many specialised output formats and a configurable filtering engine. Allows indexing of FASTA files and any file format using tab-separated columns, such as GFF, VCF and METAL, in preparation to high-speed access. Can read specified subsections of indexed FASTA files very fast. It also provides many easy-to-use methods to access the UCSC Genome Browser SQL servers, the AmiGO gene ontology databases, PLINK .PED files and Bioconductor’s organism annotation databases.

Details

Package: WhopGenome
Type: Package
Version: 1.0
Date: 2013-01-24
License: GPL-2
bgzf_compress

- Open a VCF file with handle <- vcf_open("filename") - Set a region of interest (chromosome/contig ID,start position, end position) with vcf_setregion(handle,"X",200000, 300000 ) - Select (in this case the first 10) samples of interest: vcf_selectsamples( handle, vcf_getSampleNames(handle)[1:10] ) - Read from the file via resvec <- vcf_readLineVec(handle)

Author(s)
Ulrich Wittelsbuerger <ulrich.wittelsbuerger@uni-duesseldorf.de>

References
The 1000 Genomes Project http://1000genomes.org/
The Variant Call Format http://www.1000genomes.org/wiki/Analysis/Variant%20Call%20Format/vcf-variant-call-format-version-41

Examples
#vcfh <- .Call("VCF_open","/data/vcf/1000g/ALL.Chromosome1.consensus.vcf.gz",PACKAGE="WhopGenome")

bgzf_compress Compress file with bgzip

Description
Write contents of file <infilename> bgzip-compressed to file named <outfilename>.

Usage
bgzf_compress( infilename, outfilename )

Arguments
infilename Name of file to read data from for compression
outfilename Name of file to write compressed data to

Details
Compresses the file specified by <infilename> with the bgzip compression scheme, as developed by Bob Handsaker and modified by Heng Li, and creates a compressed file under the name given by <outfilename>.

Value
TRUE if call succeeds, FALSE if it fails).
Author(s)

Ulrich Wittelsbuerger

Examples

```r
##
## Example:
##
gfffile <- system.file("data", "ex.gff3", package = "WhopGenome")
gffgzfile <- paste(sep="", gfffile, ".gz")
file.remove(gffgzfile)
bgzf_compress(gfffile, gffgzfile)
file.exists(gffgzfile)
```

---

fai_build  
Build a .fai-index for the given FASTA file.

Description

Build a .fai-index for the given FASTA file.

Usage

`fai_build( filename )`

Arguments

filename  
Name of the FASTA file for which an index file should be built.

Details

Use `.Call("FAI_build", filename)` to eliminate the overhead of using the R wrapper function.

Value

TRUE if call succeeds, FALSE if it fails.

Author(s)

Ulrich Wittelsbuerger

See Also

fai_open
Examples

```r
## Example:
##
## faifile <- system.file("extdata", "ex.fasta", package = "WhopGenome")
faiindexfile <- paste( sep="", faifile, ".fai" ) # construct name of index file
file.remove( faiindexfile ) # remove existing index
fai_build( faifile ) # re-create index
stopifnot( file.exists(faiindexfile) ) # check whether index file exists
print("All OK")
```

---

**fai_close**

Closes a file previously opened with fai_open

## Description

Closes a file previously opened with fai_open

## Usage

```r
fai_close( faifh )
```

## Arguments

- **faifh**
  
  A FAIhandle as returned by fai_open

## Details

Use `.Call("FAI_close", faifh )` to eliminate the slight overhead of using the R wrapper function.

## Value

TRUE if call succeeds, FALSE if it fails.

## Author(s)

Ulrich Wittelsbuerger

## See Also

fai_open
Examples

```r
## Example:
##
## faifile <- system.file("extdata", "ex.fasta", package = "WhopGenome")
## faifh <- fai_open( faifile )
## stopifnot( !is.null(faifh) )
## fai_close( faifh )
```

---

**fai_open**  
*Open a fai-indexed FASTA file*

**Description**

Opens a FASTA file that has an associated .fai index file

**Usage**

```r
fai_open( filename )
```

**Arguments**

- `filename`: File name of the FASTA file. A file `filename.fai` is expected to reside in the same path.

**Details**

Use `.Call("FAI_open", filename)` to eliminate the slight overhead of using the R wrapper function.

**Value**

Returns a FAIhandle that is required for `fai_query3`, `fai_query5`, `fai_close`

**Author(s)**

Ulrich Wittelsbuerger

**See Also**

`fai_reopen`, `fai_query3`, `fai_query5`
**fai_query2**

Examples

```r
## Example :
##
## faifile <- system.file("extdata", "ex.fasta", package = "WhopGenome")
## faifh <- fai_open( faifile )
## stopifnot( !is.null(faifh) )
```

---

**fai_query2**  Extract a part of a FASTA sequence.

---

**Description**

Return a part of a FASTA sequence.

**Usage**

```r
fai_query2( faifh, regionstring )
fai_query3( faifh, regionstring , resultstring )
```

**Arguments**

- `faifh`  FAIhandle as returned by fai_open
- `regionstring`  String of the form sequencename:beginpos-endpos e.g. "MTR1mous:20-40" specifying the sequence and region
- `resultstring`  String variable to store results into

**Details**

Note: the fai_query3 and fai_query5 methods are DEPRECATED : to be as fast as possible, they modified a given variable’s contents (resultstring) which will cause issues in R’s internals!

Use `.Call("FAI_query2", faifh, regionstring )` to eliminate the overhead of using the R wrapper function. Use this function in combination with a while( ( seq = fai_query2(F,region) ) != FALSE ) if you need to loop. Only the string "FALSE" has a boolean value of FALSE, all others have a boolean value of TRUE.

**Value**

A string containing the (sub-)sequence, FALSE if it fails.

**Author(s)**

Ulrich Wittelsbuerger

**See Also**

fai_open
Examples

```r
## Example:
##
## faifile <- system.file("extdata", "ex.fasta", package = "WhopGenome")
## faifh <- fai_open( faifile )
## stopifnot( !is.null(faifh) )
## result = fai_query2( faifh, "1:100-200" )
## if( result != FALSE )
## {
##   print( result )
## }
## fai_close( faifh )
```

---

**fai_query4**

*Extract a part of a FASTA sequence.*

**Description**

Return a part of the a FASTA sequence.

**Usage**

```r
fai_query4( faifh, sequencename, beginpos, endpos )
```

**Arguments**

- `faifh` FAIhandle as returned by `fai_open`
- `sequencename` Identifier of a sequence in the fasta file
- `beginpos` Start position of the subsequence to extract
- `endpos` End position of the subsequence to extract
- `resultstring` Variable to store the results into

**Details**

Note: the `fai_query3` and `fai_query5` methods are DEPRECATED: to be as fast as possible, they modified a given variable’s contents (resultstring) which will cause issues in R’s internals! Use `.Call("FAI_query4", faifh, sequencename, beginpos, endpos )` to eliminate the overhead of using the R wrapper function. Use this function in combination with a while( ( seq = fai_query4(F.region) ) != FALSE ) if you need to loop. (This exploits the fact that only the string "FALSE" has a boolean value of FALSE, all others have a boolean value of TRUE.)

**Value**

A string containing the (sub-)sequence, FALSE if it fails.
Author(s)  
Ulrich Wittelsbuerger

See Also  
faifh

Examples

```r
##
## Example:
##
fafh <- faifh(file = "file")
faifh <- faifh(fafh)
faifh <- faifh()  # Close FAIhandle
```

---

### Description

Reopen a FAIhandle that has become stale, e.g. by restarting R or loading a workspace containing a FAIhandle variable.

### Usage

```r
fai_reopen(faifh)
```

### Arguments

- **faifh**: A FAIhandle to a .fai-indexed FASTA file

### Details

Use `.Call("FAI_reopen", faifh)` to eliminate the slight overhead of using the R wrapper function.

### Value

TRUE if call succeeds, FALSE if it fails.
Author(s)
Ulrich Wittelsbuerger

See Also
fai_open

Examples

```r
## Example:
##
faifile <- system.file("extdata", "ex.fasta", package = "WhopGenome")
faifh <- fai_open( faifile )
stopifnot( !is.null(faifh) )
result <- fai_query4( faifh , "1", 100 , 200 )
print( result )
fai_close( faifh )
fai_reopen( faifh )
result <- fai_query4( faifh , "1", 100 , 110 )
print( result )
```

---

**tabix_build**  
*Build a tabix index file for fast access to tab-separated-value formatted files.*

Description

Given a pre-sorted and compressed file in a compatible tab-separated-columns format, create a Tabix index file to perform fast queries on regions of data.

Usage

```r
tabix_build( filename, sc, bc, ec, meta, lineskip )
```

Arguments

- `filename`: Name of file to create index for
- `sc`: Number of sequence column
- `bc`: Number of start column
- `ec`: Number of end column
- `meta`: Symbol used to begin comment/meta-information lines
- `lineskip`: Number of lines to skip from the top
Details

Tabix is a tool that has been developed to quickly retrieve data on an arbitrary chromosomal region from files that store their data in tab-separated columns, such as VCF, BED, GFF and SAM. As long as there is a column for named groups (e.g. chromosomes) and another column giving a numerical order (e.g. chromosomal position), it can be used for other data as well. As a required preprocessing step, it creates an index file for a file which has been sorted by group names (e.g. chromosome) and location as well as gzip/bgzf-compressed. After sorting, compressing and indexing, specific portions of such a file can be very efficiently retrieved, e.g. using the other tabix_XXX functions.

Value

TRUE or FALSE.

Author(s)

Ulrich Wittelsbuerger

See Also

tabix_open, tabix_setregion, tabix_read

Examples

```r
##
## Example:
##
##
gfffile <- system.file("extdata", "ex.gff3", package = "WhopGenome")
gfffile

gffbasename <- tempfile()
file.copy( from=gfffile, to=gffbasename )
gffgzfile <- paste( sep="", gffbasename, ".gz" )
gffgzfile

##
##
gffindexfile <- paste( sep="", gffgzfile, ".tbi" )
gffindexfile
stopifnot( ! file.exists( gffindexfile ) )
print("Index file does not exist yet!")

###
### compress GFF file
###
bgzf_compress( gffbasename, gffgzfile )
stopifnot( file.exists( gffgzfile ) )
###
### build index
###
```
```r
# Close Tabix-indexed file

tabix_close(tabfh)

## Arguments

- `tabfh`: Tabix file handle

## Value

- None.

## Author(s)

- Ulrich Wittelsbuerger

## See Also

- `tabix_open` `tabix_read`

## Examples

```r
## Example :

```
```
tabix_getregion

Return the currently selected region of the given tabix file.

Description
Return the currently selected region of the given tabix file. The resulting value does not reflect the current read position inside that region, i.e., you cannot infer whether there are any lines left for reading from that region.

Usage

`tabix_getregion( tabfh )`

Arguments

- `tabfh` Tabix handle, once returned by `tabix_open`

Details

Use `.Call("tabix_getRegion", tabfh)` to eliminate the slight overhead of using the R wrapper function.

Value
Tabix file handle

Author(s)
Ulrich Wittelsbuerger

See Also
`tabix_open`

Examples

```r
## Example:
##
gffgzfile <- system.file("extdata", "ex.gff3.gz", package = "WhopGenome")
gffh <- tabix_open( gffgzfile )
gffh

tabix_setregion( gffh, "ex.1", 1, 400 )
tabix_getregion( gffh )
```
Description
Open Tabix-indexed file for subsequent access with other tabix_ methods

Usage
```
tabix_open(filename)
```

Arguments
- `filename`: String, name of tabix-indexed file to open

Details
As filename, specify the data file, not the index file ending in .tbi!

Value
Tabix file handle

Author(s)
Ulrich Wittelsbuerger

See Also
- `tabix_open`
- `tabix_read`

Examples
```r
# Example:
gffgzfile <- system.file("extdata", "ex.gff3.gz", package = "WhopGenome")
gffh <- tabix_open(gffgzfile)
gffh
tabix_close(gffh)
gffh
```
Description

Read a line from a tabix_open()ed file

Usage

tabix_read(tabfh)
tabix_readraw(tabfh)

Arguments

tabfh Tabix file handle as returned by tabix_open

Details

Instead of tabix_readraw() you can use .Call("tabix_readLine", tabfh) to eliminate the slight overhead of using the R wrapper function.

Value

A line of data from the indexed data file. tabix_read splits the line up into its fields and returns a vector. tabix_readraw returns the line as stored in the file.

Author(s)

Ulrich Wittelsbuerger

See Also

tabix_open

Examples

##
## Example : (NOT RUN)
##

print("Opening and reading")
gffgzfile <- system.file("extdata", "ex.gff3.gz", package = "WhopGenome")
if( file.exists(gffgzfile) )
{
gffgzfile
gfh <- tabix_open( gffgzfile )
gfh
stopifnot( !is.null(gfh) )
tabix_reopen

Reopen a Tabix-indexed file if the filehandle became invalid.

Description

Reopen a Tabix-indexed file if the filehandle became invalid.

Usage

```r
tabix_reopen( tabfh )
```

Arguments

- `tabfh` Tabix handle, once returned by `tabix_open`

Details

Use `.Call("tabix_reopen", tabfh)` to eliminate the slight overhead of using the R wrapper function.

Value

Tabix file handle

Author(s)

Ulrich Wittelsbuerger

See Also

`tabix_open`

Examples

```r
## Example :
##
##
##
## Example :
##
##
gffgzfile <- system.file("extdata", "ex.gff3.gz", package = "WhopGenome")
gffh <- tabix_open( gffgzfile )
gffh
```
tabix_restartregion

```
  tabix_restartregion( gffh )
gffh
  tabix_reopen( gffh )
gffh
```

---

**Description**

Reset the currently selected region so that the next read call will return the first line inside that region.

**Usage**

```
  tabix_restartregion( tabfh )
```

**Arguments**

`tabfh` Tabix handle, once returned by `tabix_open`

**Details**

Use `.Call("tabix_restartRegion", tabfh )` to eliminate the slight overhead of using the R wrapper function.

**Value**

Tabix file handle

**Author(s)**

Ulrich Wittelsbuerger

**See Also**

`tabix_open`

**Examples**

```
## Example :
##
gffgzfile <- system.file("extdata", "ex.gff3.gz", package = "WhopGenome" )
gffh <- tabix_open( gffgzfile )
gffh
##
```
## tabix_setregion

Reopen a Tabix-indexed file if the filehandle became invalid.

### Description

Reopen a Tabix-indexed file if the filehandle became invalid.

### Usage

```
   tabix_setregion( tabfh, tid, beginpos, endpos )
```

### Arguments

- **tabfh**: Tabix handle, once returned by `tabix_open`
- **tid**: A string naming one of the contig/chromosome identifiers stored in the Tabix indexed file
- **beginpos**: Earliest position from which subsequent `tabix_read/tabix_readraw` calls return lines
- **endpos**: Last position to return lines from with `tabix_read/tabix_readraw`

### Details

Use `.Call("tabix_setRegion", tabfh, tid, beginpos, endpos)` to eliminate the slight overhead of using the R wrapper function.

### Value

Tabix file handle

### Author(s)

Ulrich Wittelsbuerger

### See Also

`tabix_open`
vcf_addfilter

Examples

```r
## Example:
##
gffgzfile <- system.file("extdata", "ex.gff3.gz", package = "WhopGenome")
gfh <- tabix_open( gffgzfile )
gfh
  tabix_setregion( gfh, "ex.1", 1, 400 )
gfh
```

cf_addfilter Add a condition for SNP filtering from VCF files.

Description

Add a condition for filtering SNPs based on any column in a given VCF file.

Usage

`vcf_addfilter(vcf, columnnam, fieldnam, cmptype, cmpvalue1, cmpvalue2 = 0, action)`

Arguments

- `vcf`: VCF file handle
- `columnnam`: name of column containing the to-be-checked values
- `fieldnam`: name of the subfield or "" to check
- `cmptype`: Type of comparison to perform. See Details
- `cmpvalue1`: Comparison reference value 1 or lower bound
- `cmpvalue2`: Comparison reference value 2 or upper bound
- `action`: Action to take if comparison matches: NOP, SKIP, KEEP or fails: SKIP_NOT, KEEP_NOT

Details

Parameter 'columnnam': Name of a VCF column, in which the data of interest is stored. Parameter 'fieldnam': For the INFO and samples columns, the key under which the interesting data is stored. Example: `vcf_addfilter( vcf_file, "INFO", "H2", "DOES_EXIST", 0, 0, "DROP_NOT" )` would cause any subsequent calls to read functions that perform filtering to drop lines that do not have the "H2" key in the INFO column, which indicates that the SNP is not marked as being registered in HapMap2. The parameters <ref1> and <ref2> are not used by the "DOES_EXIST" operation.

Comparison types:
• DOES_EXIST Rule matches, if in column named by <columnnam> is a key with the same name as in <fieldnam>

for integer values:
• INT_CMP is value = ref1 ?
• INT_CMP_OO is value in open range (ref1, ref2)
• INT_CMP_OC is value in half-closed range (ref1, ref2]
• INT_CMP_CO is value in half-closed range [ref1, ref2)
• INT_CMP_CC is value in closed range [ref1, ref2]

for floating point values:
• FLT_CMP is value = ref1 ?
• FLT_CMP_OO is value in open range (ref1, ref2)
• FLT_CMP_OC is value in half-closed range (ref1, ref2]
• FLT_CMP_CO is value in half-closed range [ref1, ref2)
• FLT_CMP_CC is value in closed range [ref1, ref2]

Value
Success status: TRUE on success, FALSE if the rule could not be added.

Author(s)
Ulrich Wittelsbuerger

Examples

```r
## Example:
##
## vcf_file <- vcf_open( system.file( "extdata", "ex.vcf.gz", package="WhopGenome" ) )
vcf_setregion(vcf_file, "Y", 1, 100000 )
vcf_addfilter( vcf_file, "POS", ",", "INT_CMP_OO",
 as.integer(49005), as.integer(49007), "DROP" )
vcf_describefilters( vcf_file )
####
####
vcf_readLineVecFiltered( vcf_file )
vcf_readLineVecFiltered( vcf_file )
vcf_readLineVecFiltered( vcf_file )
#####
#####
vcf_clearfilters( vcf_file )
vcf_describefilters( vcf_file )
vcf_restartregion( vcf_file )
####
####
vcf_readLineVecFiltered( vcf_file )
```
vcf_buildindex

Build Tabix-index required for processing VCF files.

Description
Builds a Tabix-index for a VCF file that is already sorted and compressed.

Usage
vcf_buildindex( filename )

Arguments
filename Name of VCF file

Details
Given the name of a VCF file, builds a Tabix-index file (automatically named <filename>.tbi) in the directory where the given VCF file is located. Prerequisite is that the VCF file be sorted by chromosome and position as well as bgzip-compressed. Such files carry the extension .vcf.gz. Information on how to sort data in VCF files can be found at <http://vcftools.sourceforge.net/docs.html>. Using bgzf_compress, you can thereafter compress the file.

Value
Returns TRUE if the index could be created or FALSE if not.

Author(s)
Ulrich Wittelsbuerger

See Also

    tabix_build bgzf_compress

Examples

    ##
    ## Example:
    ##
vcf_clearfilters

Removes all filter steps.

Description

Removes all active filters, no pre-filtering of returned lines will take place. There is no function to undo this step.

Usage

vcf_clearfilters(vcffh)

Arguments

vcffh VCF file handle

Details

Use .Call("VCF_clearFilters", vcffh) to eliminate the overhead of using the R wrapper function.

Value

None.

Author(s)

Ulrich Wittelsbuerger

Examples

## Example:
##
## vcf_file <- vcf_open( system.file( "extdata", "ex.vcf.gz", package="WhopGenome" ) )
vcf_setregion(vcf_file, "Y", 1, 100000 )
vcf_addfilter( vcf_file, "POS", ",", "INT_CMP_OO",
as.integer(49005), as.integer(49007), "DROP" )
vcf_describefilters( vcf_file )
##
##
## vcf_readLineVecFiltered( vcf_file )
vcf_readLineVecFiltered( vcf_file )
vcf_readLineVecFiltered( vcf_file )

####################################
vcf_clearfilters( vcf_file )
vcf_describefilters( vcf_file )
vcf_restartregion( vcf_file )
##
### vcf_close

Closes a VCF file previously opened with vcf_open.

**Description**

Closes the VCF file described by the given handle and prevents subsequent use.

**Usage**

```r
vcf_close(vcf_filehandle)
```

**Arguments**

- `vcf_filehandle` A VCF filehandle returned by vcf_open

**Details**

Use `.Call("VCF_close", vcf_filehandle)` to eliminate the overhead of using the R wrapper function.

**Value**

None

**Author(s)**

Ulrich Wittelsbuerger

**See Also**

vcf_open

**Examples**

```r
## Example:
##
vcffile <- system.file("extdata", "ex.vcf.gz", package="WhopGenome")
vcffile
vcffile
vcffh <- vcf_open( vcffile )
vcffh
```
## vcf_countSNPs

Count how many entries in the selected region

### Description

Reads all data in the currently selected region of the given VCF file and counts how many loci with SNPs or biallelic SNPs respectively, are encountered.

### Usage

```r
vcf_countSNPs( vcffh )
vcf_countBiallelicSNPs( vcffh )
```

### Arguments

- **vcffh**: Handle to a VCF file, as returned by `vcf_open`

### Details

For certain cases, like pre-allocating variables, it can be useful to know how many SNPs are present in a certain region. In order to reduce the effort of this task and its impact on runtime to a minimum, the functions `vcf_countSNPs` and `vcf_countBiallelicSNPs` were implemented. Take note that they do not automatically 'restart' from the beginning of the selected region but continue from the current position. Use `vcf_restartregion` to make sure that all SNPs in the currently set region are counted.

### Value

An integer number is returned: the number of SNPs or biallelic SNPs.

### Author(s)

Ulrich Wittelsbuerger

### See Also

- `vcf_restartregion`
Examples

```r
## Example:
##
## vcf_file <- system.file( "extdata" , "ex.vcf.gz" , package="WhopGenome" )
## vcf_file
## vcf_fh <- vcf_open( vcf_file )
## vcf_fh
## vcf_countSNPs( vcf_fh )
```

---

**vcf_describefilters**  
*Prints description of current filter rules*

Description

Prints a better understandable description of the filter rules currently active for the given VCF file.

Usage

```r
vcf_describefilters(vcffh)
```

Arguments

- `vcffh` VCF file handle

Details

Use `.Call("VCF_describeFilterConfig", filename )` to eliminate the overhead of using the R wrapper function. Note the different naming of the library function!

Value

None.

Author(s)

Ulrich Wittelsbuerger

Examples

```r
## Example:
##
## vcf_file <- vcf_open( system.file( "extdata" , "ex.vcf.gz" , package="WhopGenome" ) )
## vcf_setregion(vcf_file, "Y", 1, 100000 )
## vcf_addfilter( vcf_file, "POS", "", "INT_CMP OO", as.integer(49005), as.integer(49007), "DROP" )
## vcf_describefilters( vcf_file )
###
```
Determine whether all lines in the selected region have been read.

**Description**

When reading SNP info within a region defined by `VCF_setRegion`, this function returns TRUE/FALSE to indicate whether or not all lines within that region have been read.

**Usage**

```r
cvf_eor( vcfh )
```

**Arguments**

- `vcfh`  
  Handle of a VCF file opened by `VCF_open`

**Details**

Use `.Call("VCF_eor", vcfh)` to eliminate the overhead of using the R wrapper function.

**Value**

TRUE if all SNPs inside the previously defined region have been read.

**Author(s)**

Ulrich Wittelsbuerger
Examples

```r
vcffile <- vcf_open( system.file( "extdata", "ex.vcf.gz" , package="WhopGenome" ) )
vcf_setregion(vcffile, "Y", 1, 100000 )
while( !vcf_eor(vcffile) )
{
  vcf_readLineVec( vcffile )
}
```

vcf_getChrom

Return a specific piece of information from the last line processed with vcf_parseNextSNP or vcf_parcenextline.

Description

Return a specific piece of information from the last line processed with vcf_parseNextSNP or vcf_parcenextline.

Usage

```r
vcf_getChrom( vcffh )
vcf_getPos( vcffh )
vcf_getID( vcffh )
vcf_getRef( vcffh )
vcf_getAlt( vcffh )
vcf_getQual( vcffh )
vcf_getFilter( vcffh )
vcf_getInfo( vcffh )
vcf_getInfoField( vcffh, fieldnam )
vcf_getFormat( vcffh )
vcf_getSample( vcffh, stridx )
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vcffh</td>
<td>VCF file handle</td>
</tr>
<tr>
<td>fieldnam</td>
<td>Name of a key of the key-value-pairs stored in the INFO subfield</td>
</tr>
<tr>
<td>stridx</td>
<td>Name of a sample</td>
</tr>
</tbody>
</table>

Details

Use `.Call("VCF_getChrom", filename )` to eliminate the overhead of using the R wrapper function. Replace getChrom by getPos, getID, getRef, getAlt, getQual, getFilter, getInfo, getInfoField, getSample and add the respective function arguments in the order given above to call the respective other function.

Value

None if the call failed, otherwise the respective data from the last read line is extracted.
vcf_getcontignames

Description

Return the contig/chromosome identifiers used in the VCF file

Usage

vcf_getcontignames(vcff)

Arguments

vcff VCF file handle

Details

vcf_setregion for example requires one of these identifiers to be able to successfully select a region for extraction. Use .Call("VCF_getContigNames", vcff) to eliminate the overhead of using the R wrapper function.

Value

Vector with contig and/or chromosome identifiers.
vcf_getfieldnames

Author(s)

Ulrich Wittelsbuerger

See Also

vcf_setregion

Examples

vcffile <- vcf_open( system.file( "extdata", "ex.vcf.gz", package="WhopGenome" ) )
vcf_getcontignames( vcffile )
# [1] "Y"

vcf_getfieldnames(vcff)

Description

Return a vector with the field names used in the VCF file.

Usage

vcf_getfieldnames(vcff)

Arguments

vcff VCF file handle

Details

Use .Call("VCF_getFieldNames", vcff ) to eliminate the overhead of using the R wrapper function.

Value

A vector of strings representing the field names present in the VCF file.

Author(s)

Ulrich Wittelsbuerger

Examples

vcffile <- vcf_open( system.file( "extdata", "ex.vcf.gz", package="WhopGenome" ) )
vcf_getfieldnames( vcffile )
### vcf_getheaderline

*Return one of the header lines of the VCF file*

**Description**

Return one of the header lines of the VCF file

**Usage**

```r
cvf_getheaderline(vcff, whichnum)
```

**Arguments**

- `vcff`: VCF file handle
- `whichnum`: Number of header line to retrieve

**Details**

Use `.Call("VCF_getHeaderLine", vcff, whichnum)` to eliminate the overhead of using the R wrapper function.

**Value**

A string containing the full header line.

**Author(s)**

Ulrich Wittelsbuerger

**Examples**

```r
vcffile <- vcf_open( system.file( "extdata", "ex.vcf.gz", package="WhopGenome" ) )
vcf_getheaderline( vcffile, as.integer(0) )
vcf_getheaderline( vcffile, as.integer(1) )
```

### vcf_getnumcontigs

*Get the number of different contigs/chromosomes stored in the file*

**Description**

Get the number of different contigs/chromosomes stored in the file

**Usage**

```r
cvf_getnumcontigs(vcff)
```
vcf_getregion

Arguments

vcff VCF file handle

Details

Use .Call("VCF_getNumContig", vcff) to eliminate the overhead of using the R wrapper function.

Value

The number of different contigs/chromosomes stored in the file.

Author(s)

Ulrich Wittelsbuerger

Examples

vcffile <- vcf_open( system.file( "extdata", "ex.vcf.gz", package="WhopGenome") )
vcf_getnumcontigs( vcffile )
# [1] 1

vcf_getregion Get description of currently selected chromosomal region.

Description

Returns a textual description like "chr3:10913000-20240100" representing the genomic range which is currently set.

Usage

vcf_getregion( vcffh )

Arguments

vcffh Handle to the VCF for which the currently active region should be retrieved

Details

Returns the string describing the region which is currently set for the given VCF file. The string has the form "<chromosome id>:<startpos>-<endpos>", e.g. "1:120300-130500", where "1" is the identifier of the chromosome or contig stored in the file. 120300 is the leftmost position in the sequence for which we want to get variation data and 130500 is the rightmost position. Because usually there is no variation data for every position, there is no guarantee that the first reported SNP will be at position 120300. Initially, before a region has been set by the user, the returned string is ":0-0".
**Value**

NULL if vcffh is not a valid VCF filehandle as returned by vcf_open. Otherwise, a region string.

**Author(s)**

Ulrich Wittelsbuerger

**Examples**

```r
cvfile <- vcf_open( system.file( "extdata", "ex.vcf.gz", package="WhopGenome" ) )
vcf_getregion( cvffile )
```

---

**vcf_isInDEL**

*Determines whether the last vcf_parse-call returned a InDel (instead of SNP)*

**Description**

Returns TRUE if the last call to vcf_parse/VCF_parse returned an InDel.

**Usage**

```r
vcf_isINDEL(vcff)
```

**Arguments**

- `vcff` VCF file handle

**Details**

Use `.Call("VCF_isInDel", vcff)` to eliminate the overhead of using the R wrapper function.

**Value**

TRUE or FALSE.

**Author(s)**

Ulrich Wittelsbuerger

**See Also**

vcf_isSNP

**Examples**

```r
cvffile <- vcf_open( system.file( "extdata", "ex.vcf.gz", package="WhopGenome" ) )
vcf_parseNextSNP( cvffile )
vcf_getPos( cvffile )
vcf_isINDEL( cvffile )
```
Determines whether the last vcf_parse-call returned a SNP (instead of InDel)

**Usage**

```r
vcf_isSNP(vcff)
```

**Arguments**

- `vcff`: VCF file handle

**Details**

Use `.Call("VCF_isSNP", vcff)` to eliminate the overhead of using the R wrapper function.

**Value**

None.

**Author(s)**

Ulrich Wittelsbuerger

**See Also**

- `vcf_isINDEL`

**Examples**

```r
vcffile <- vcf_open( system.file( "extdata", "ex.vcf.gz" , package="WhopGenome") )
vcf_parseNextSNP( vcffile )
vcf_getPos( vcffile )
vcf_isSNP( vcffile )
```
vcf_open

Open the specified VCF file and return a filehandle for subsequent access.

Description
Open the specified VCF file and return a filehandle for subsequent access.

Usage
vcf_open(filename)

Arguments
filename A filename of a tabix-indexed and gzip-compressed VCF file

Details
Use .Call("VCF_open", filename ) to eliminate the overhead of using the R wrapper function.

Value
A VCF file handle, used in most VCF functions

Author(s)
Ulrich Wittelsbuerger

Examples
vcffile <- vcf_open( system.file( "extdata", "ex.vcf.gz", package="WhopGenome") )

vcf_parseNextSNP

Read until next SNP or next line and buffer it

Description
Read until next SNP or next line and buffer it. Use the vcf_getXXX functions to access specific fields of the line

Usage
vcf_parseNextSNP(vcffh)
vcf_parseNextLine(vcffh)
Arguments

vcffh       VCF file handle

Details

Use .Call("VCF_parseNextSNP", vcffh ) and .Call("VCF_parseNextLine", vcffh ) respectively, to eliminate the overhead of using the R wrapper function.

Value

None.

Author(s)

Ulrich Wittelsbuerger

See Also

vcf_isSNP, vcf_open, vcf_getPos

Examples

vcffile <- vcf_open( system.file( "extdata", "ex.vcf.gz", package="WhopGenome" ) )
vcf_parseNextSNP( vcffile )
vcf_getPos( vcffile )

vcf_readLineDF

Read a line of data from the given VCF file and return it as a data frame

Description

Read a line of data from the given VCF file and return it as a data frame

Usage

vcf_readLineDF(vcffh)

Arguments

vcffh       VCF file handle

Details

Reads a line of data from the given VCF file, splits it up into its components (fields) and fills a data.frame with the contents of the fields and names the entries according to the header line of the VCF (e.g. CHROM, POS, ID, REF, ALT, ... ).
vcf_readLineRaw

Value
A data frame

Author(s)
Ulrich Wittelsbuerger

Examples
vcffile <- vcf_open( system.file( "extdata", "ex.vcf.gz", package="WhopGenome" ) )
d <- vcf_readLineDF( vcffile )

vcf_readLineRaw
Read a line of data from the given VCF file and return it as a string without postprocessing.

Description
Read a line of data from the given VCF file and return it as a string without postprocessing.

Usage
vcf_readLineRaw( vcffh )
vcf_readLineRawFiltered( vcffh )

Arguments
vcffh VCF file handle

Details
vcf_readLineRawFiltered applies the filtering rules (see vcf_describefilters) and does not return any lines that do not pass the filter rules.
Use .Call("VCF_readLineRaw", vcffh ) and .Call("VCF_readLineRawFiltered", vcffh ) respectively, to eliminate the overhead of using the R wrapper function.

Value
For the 1-argument versions: A raw string representing a line of data from the file or FALSE if no more lines to read

Author(s)
Ulrich Wittelsbuerger
vcf_readLineVec

**Examples**

```r
vcffile <- vcf_open( system.file( "extdata", "ex.vcf.gz" , package="WhopGenome" ) )
d <- vcf_readLineRaw( vcffile )
```

---

**vcf_readLineVec**  
*Read a line of data from the given VCF file and return the fields as vector elements*

**Description**

Read a line of data from the given VCF file and return the fields as vector elements

**Usage**

```r
vcf_readLineVec(vcffh)
vcf_readLineVecFiltered(vcffh)
```

**Arguments**

- `vcffh`: VCF file handle

**Details**

The latter version applies filtering set up with `vcf_addfilter`. Use `.Call("VCF_readLineTSV", vcffh)` or `.Call("VCF_readLineTSVFiltered", vcffh)` respectively to eliminate the overhead of using the R wrapper function.

**Value**

A vector where each element is a field from a line of data in the VCF

**Author(s)**

Ulrich Wittelsbuerger

**See Also**

`vcf_addfilter`, `vcf_describeFilters`

**Examples**

```r
vcffile <- vcf_open( system.file( "extdata", "ex.vcf.gz" , package="WhopGenome" ) )
vcf_readLineVec( vcffile )
```
VCF_read_snp_diplo_bial_int_altpresence

(OBSOLETE) Read batch of biallelic SNP data into matrices

Description

OBSOLETE: please refer to documentation for "VCF_snpmat_diplo_bial_geno_filtered" to find out about their replacements.

Reads biallelic SNP data in different representations into pre-allocated matrices.

Usage

VCF_read_snp_diplo_bial_int_altpresence( vcffh, mat )
VCF_read_snp_diplo_bial_int_nuclcodes( vcffh, mat )
VCF_read_snp_diplo_bial_str_01( vcffh, mat )
VCF_read_snp_diplo_bial_str_allelechars( vcffh, mat )
VCF_read_snp_diplo_bial_str_nuclcodes( vcffh, mat )

Arguments

vcffh        VCF file handle as returned by vcf_open
mat          A matrix of either integer or string type, corresponding to _str_ or _int_ named methods

Details

OBSOLETE: please refer to documentation for "VCF_snpmat_diplo_bial_geno_filtered" to find out about their replacements.

Prerequisites are: - a valid, open VCF file handle, passed as vcffh - a valid sample selection (vcf_getsamples, vcf_getselectedsamples, vcf_selectsamples) - a properly set region (vcf_setregion) - and a result matrix, mat.

The matrix will be filled with allele data in one of 4 encodings and needs to be of either integer or character data type, both depending on the called function (VCF_..._int_... or VCF_..._str_...). Each column corresponds to a SNP locus and each row to a sample. The number of matrix columns determines the maximum number of SNP loci that are parsed from the VCF. Column names are set to the position of the SNP, the row names are named after the samples they represent. There must be at least as many rows as selected samples. Unused rows will be filled with default (N) data. If there are not enough SNPs to fill all columns, the unused columns will be numbered with -1 and filled with N or -1.

VCF data is required to be diploid.

Representations:

- int_altpresence: 0 if genotype is REF/REF, 1 if not
vcf_reopen

- int_nuclcodes : integers, two-digit numbers: 11=TT, 12=TC, 13=TG, 14=TA, 15=TN, 21=CT, etc. (1=T, 2=C, 3=G, 4=A, 5=N)
- str_01 : string, either 00, 01, 10 or 11 : 00=ref/ref, 11=alt/alt, 10=alt/ref, 01=ref/alt
- str_allelechars : string, nucleotides of both chromosomes (no indication of reference allele)
- str_nuclcodes : string, two-digit numbers: 11=TT, 12=TC, 13=TG, 14=TA, 15=TN, 21=CT, etc. (1=T, 2=C, 3=G, 4=A, 5=N)

Value

TRUE or FALSE

Author(s)

Ulrich Wittelsbuerger

See Also

VCF_snpmat_diplo_bial_geno_filtered

Examples

warning("These functions are obsolete! Consult VCF_snpmat_diplo_bial_geno_filtered etc."")

---

vcf_reopen

Reopen a closed or stale VCF file handle.

Description

Allows re-opening a previously opened VCF file.

Usage

vcf_reopen(vcffh)

Arguments

vcffh VCF file handle as returned by vcf_open

Details

If a file handle was closed (vcf_close) or became stale (e.g. after an R crash), it can be reactivated with this function. Use .Call("VCF_reopen", vcffh ) to eliminate the overhead of using the R wrapper function.

Value

Returns the reopened file handle.
vcf_restartregion

Author(s)
Ulrich Wittelsbuerger

See Also
vcf_open

Examples
vcffile <- vcf_open( system.file( "extdata", "ex.vcf.gz", package="WhopGenome" ) )
vcffile
vcf_close( vcffile )
vcffile
vcf_reopen( vcffile )
vcffile

vcf_restartregion Let subsequent read calls return from the start of the currently set region.

Description
Once the read-functions reached the end of the previously set region, no more results are returned. If, for example for a two-pass algorithm, the same region should be scanned again from the start, this function is the key.

Usage
vcf_restartregion(vcffh)

Arguments
vcffh Handle of a VCF file, as returned by vcf_open()

Details
Alternative to calling vcf_setregion() with the same parameters again. Use .Call("VCF_restartRegion", vcffh ) to eliminate the overhead of using the R wrapper function.

Value
TRUE if the region could be rewound, FALSE if not.

Author(s)
Ulrich Wittelsbuerger
vcf_rule.disable

See Also

vcf_setregion, vcf_open

Examples

```r
## Example:
##
## vcf_file <- vcf_open( system.file( "extdata", "ex.vcf.gz", package="WhopGenome" ) )
## vcf_setregion(vcf_file, "y", 1, 100000 )

vcf_readLineVec( vcf_file )
vcf_readLineVec( vcf_file )
vcf_restartregion( vcf_file )
vcf_readLineVec( vcf_file )
vcf_readLineVec( vcf_file )
```

vcf_rule.disable

Disable and enable processing of a rule

Description

Filtering rules can be enabled and disabled. Disabled rules are ignored by any filtering VCF read function.

Usage

```r
vcf_rule.disable( vcfh, ruleidx )
vcf_rule.enable( vcfh, ruleidx )
```

Arguments

- **vcfh**: VCF file handle
- **ruleidx**: number of rule

Details

Certain VCF read functions support the fast pre-filtering mechanism of WhopGenome. The pre-filtering mechanism is a list of rules that describe how and what to check in SNP descriptions and is executed very quickly without using any R code. Every rule specifies the column of data (e.g. INFO, POS, FILTER), the key in the column (e.g. AF in the INFO column), the type of comparison, reference values to compare against and whether to keep or drop the line if the rule matches.

Value

TRUE if succeeded, FALSE if not
Author(s)
Ulrich Wittelsbuerger

Examples

```r
##
## Example:
##
## vcffile <- vcf_open( system.file( "extdata" , "ex.vcf.gz" , package="WhopGenome" ) )
vcf_setregion(vcffile, "Y", 1, 100000 )
vcf_addfilter( vcffile, "POS", ",", "INT_CMP_00", as.integer(49005), as.integer(49007), "DROP" )
vcf_describefilters( vcffile )
vcf_readLineVecFiltered( vcffile )

vcf_rule.disable( vcffile, 0 )
vcf_describefilters( vcffile )
vcf_restartregion( vcffile )
vcf_readLineVecFiltered( vcffile )
```

**vcf_rule.setaction**  
Sets the kind of action to take when a rule matches (or does not match).

**Description**

The value that rule number <ruleidx> should inspect is stored in the column named <column>, e.g. "INFO" or "POS".

**Usage**

```r
vcf_rule.setaction( vcffh, ruleidx, action )
```

**Arguments**

- vcffh  
  VCF file handle
- ruleidx  
  Filter rule to change
- action  
  name of an action, see below

**Details**

Recognised values for 'action':

- NOP do nothing
- SKIP drop line on match, read next line
- DROP keep line on match, do not test further
- KEEP keep line if not matching rule, do not test further

Each action has also a 'disabled' variant, causing it to be ignored.

- NOP_DISABLED
- SKIP.Disabled
- DROP.Disabled
- KEEP.Disabled
- SKIP_NOT
- DROP_NOT
- KEEP_NOT
- SKIP_IF_NOT
- DROP_IF_NOT
- KEEP_IF_NOT
- SKIP_NOT.disabled
- DROP_NOT.disabled
- KEEP_NOT.disabled
- SKIP_IF_NOT.disabled
- DROP_IF_NOT.disabled
- KEEP_IF_NOT.disabled
The _NOT / _IF_NOT variants effectively invert the comparison operation. (A == B) becomes (A != B), (1 <= A <= 100) becomes (A < 1 OR > 100).

Certain VCF read functions support the fast pre-filtering mechanism of WhopGenome. The pre-filtering mechanism is a list of rules that describe how and what to check in SNP descriptions and is executed very quickly without using any R code. Every rule specifies the column of data (e.g. INFO, POS, FILTER), the key in the column (e.g. AF in the INFO column), the type of comparison, reference values to compare against and whether to keep or drop the line if the rule matches.

**Value**

TRUE on success, FALSE if it failed.

**Author(s)**

Ulrich Wittelsbuerger

**Examples**

```r
## Example:
##
vcffile <- vcf_open( system.file( "extdata", "ex.vcf.gz", package="WhopGenome" ) )
vcf_addfilter( vcffile, "POS", "\", "INTCMP OO", as.integer(49005), as.integer(49007), "DROP" )
vcf_describefilters( vcffile )

vcf_rule.setcolumn( vcffile, 0, "ID" )
vcf_describefilters( vcffile )
```

**Description**

The value that rule number <ruleidx> should inspect is stored in the column named <column>, e.g. "INFO" or "POS".

**Usage**

`vcf_rule.setcolumn( vcffh, ruleidx, column )`

**Arguments**

- `vcffh` VCF file handle
- `ruleidx` Filter rule to change
- `column` name of column containing the to-be-checked values
Details

Certain VCF read functions support the fast pre-filtering mechanism of WhopGenome. The pre-filtering mechanism is a list of rules that describe how and what to check in SNP descriptions and is executed very quickly without using any R code. Every rule specifies the column of data (e.g. INFO, POS, FILTER), the key in the column (e.g. AF in the INFO column), the type of comparison, reference values to compare against and whether to keep or drop the line if the rule matches.

Value

TRUE on success, FALSE if it failed.

Author(s)

Ulrich Wittelsbuerger

Examples

```r
## Example:
##
vcffile <- vcf_open( system.file( "extdata", "ex.vcf.gz", package="WhopGenome" ) )
vcf_addfilter( vcffile, "POS", "", "INT_CMP_OO", as.integer(49005), as.integer(49007), "DROP" )
vcf_describefilters( vcffile )

vcf_rule.setcolumn( vcffile, 0, "ID" )
vcf_describefilters( vcffile )
```

---

**vcf_rule.setcomparison**

*Set comparison operation for filtering rule.*

Description

For filtering rule `<ruleidx>` the comparison operation is set to `<cmpop>`, which is one of the following strings:

<table>
<thead>
<tr>
<th>string</th>
<th>alternative</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;HASKEY&quot;</td>
<td>&quot;DOES_EXIST&quot;</td>
<td>key (specified as field) is present in column</td>
</tr>
<tr>
<td>&quot;INT=&quot;</td>
<td>&quot;INT_CMP&quot;</td>
<td>ref1 = value</td>
</tr>
<tr>
<td>&quot;INT()&quot;</td>
<td>&quot;INT_CMP_OO&quot;</td>
<td>ref1 &lt; value &lt; ref2</td>
</tr>
<tr>
<td>&quot;INT[]&quot;</td>
<td>&quot;INT_CMP_OC&quot;</td>
<td>ref1 &lt; value &lt;= ref2</td>
</tr>
<tr>
<td>&quot;INT[]&quot;</td>
<td>&quot;INT_CMP_CC&quot;</td>
<td>ref1 &lt;= value &lt;= ref2</td>
</tr>
<tr>
<td>&quot;FLT==&quot;</td>
<td>&quot;FLT_CMP&quot;</td>
<td>ref1 = value</td>
</tr>
</tbody>
</table>

- integer comparisons:
- floating point (real numbers):
vcf_rule.setcomparison

"FLT()" "FLT_CMP_OO" ref1 < value < ref2
"FLT(]" "FLT_CMP_OC" ref1 < value <= ref2
"FLT[)" "FLT_CMP_CO" ref1 <= value < ref2
"FLT[]" "FLT_CMP_CC" ref1 <= value <= ref2

Usage

vcf_rule.setcomparison( vcffh, ruleidx, cmpop )

Arguments

vcffh         VCF file handle
ruleidx       number of rule in list
cmpop         One of the above strings, naming the comparison operation to perform

Details

Certain VCF read functions support the fast pre-filtering mechanism of WhopGenome. The pre-
filtering mechanism is a list of rules that describe how and what to check in SNP descriptions and
is executed very quickly without using any R code. Every rule specifies the column of data (e.g.
INFO, POS, FILTER), the key in the column (e.g. AF in the INFO column), the type of comparison,
reference values to compare against and whether to keep or drop the line if the rule matches.

Value

TRUE on success, FALSE if it failed.

Author(s)

Ulrich Wittelsbuerger

Examples

##
## Example:
##
vcffile <- vcf_open( system.file( "extdata", "ex.vcf.gz", package="WhopGenome" ) )
vcf_addfilter( vcffile, "POS", "", "INT_CMP_OO",
     as.integer(49005), as.integer(49007), "DROP" )
vcf_describefilters( vcffile )

vcf_rule.setcomparison( vcffile , 0, "INT_CMP_CC" )
vcf_describefilters( vcffile )
vcf_rule.setfield

Set field or key of filtering rule.

Description

Filtering rule number <ruleidx> should inspect the value stored under the key <field>. This key is stored in the column defined for this rule (e.g. an INFO-column AF=0.34;RD=231;GQ=130 has keys AF,RD and GQ).

Usage

vcf_rule.setfield( vcffh, ruleidx, field )

Arguments

vcffh VCF file handle
ruleidx number of rule in list
field XXXX

Details

Certain VCF read functions support the fast pre-filtering mechanism of WhopGenome. The pre-filtering mechanism is a list of rules that describe how and what to check in SNP descriptions and is executed very quickly without using any R code. Every rule specifies the column of data (e.g. INFO, POS, FILTER), the key in the column (e.g. AF in the INFO column), the type of comparison, reference values to compare against and whether to keep or drop the line if the rule matches.

Value

TRUE on success, FALSE if it failed.

Author(s)

Ulrich Wittelsbuerger

Examples

```r
## Example:
##
## vcffile <- vcf_open( system.file( "extdata", "ex.vcf.gz", package="WhopGenome" ) )
##
## vcf_setregion(vcffile, "Y", 50000, 51000 )
##
## USELESS filter : # filter out SNPs with rule "DROP if (0.0 < INFO:AA < 0.5)"
```
# AA= ancestral allele, is a floating point number!
vcf_addfilter( vcffile, "INFO", "AA", "FLT_CMP_00", 0, 0.5, "DROP"
) vcf_describefilters( vcffile )

cvf_readLineVecFiltered( vcffile ) # pos 50001
vcf_readLineVecFiltered( vcffile ) # pos 50002
#
cvf_setregion(vcffile, "Y", 50000, 51000 )

#CORRECT rule:
# filter out SNP at pos 50001 with INFO:AF=0.285 with rule "DROP if (0.0 < INFO:AF < 0.5)"
#
vcf_rule.setfield( vcffile , 0, "AF" )
vcf_describefilters( vcffile )

vcf_readLineVecFiltered( vcffile ) # pos 50002
vcf_readLineVecFiltered( vcffile ) # pos 50003

vcf_rule.setrefvalues  Set reference values for a filtering rule’s comparison operation.

Description

Set the reference values 1 and 2 for the comparison operation of rule <ruleidx>. Some comparison operations need only the first <ref1> reference value and ignore <ref2>.

Usage

vcf_rule.setrefvalues( vcffh, ruleidx, ref1, ref2 )

Arguments

vcffh VCF file handle
ruleidx name of column containing the to-be-checked values
ref1 name of the subfield or "" to check
ref2 Type of comparison to perform. See Details

Details

Certain VCF read functions support the fast pre-filtering mechanism of WhopGenome. The pre-filtering mechanism is a list of rules that describe how and what to check in SNP descriptions and is executed very quickly without using any R code. Every rule specifies the column of data (e.g. INFO, POS, FILTER), the key in the column (e.g. AF in the INFO column), the type of comparison, reference values to compare against and whether to keep or drop the line if the rule matches.
**Value**

TRUE on success, FALSE if it failed.

**Author(s)**

Ulrich Wittelsbuerger

**Examples**

```r
## Example:
##
## vcffile <- vcf_open( system.file( "extdata", "ex.vcf.gz", package="WhopGenome" ) )
#
# vcf_setregion(vcffile, "Y", 50000, 51000 )
#
# USELESS filter : # filter out SNPs with rule "DROP if (0.0 < INFO:AF < 0.2)"
# pos 50001 has AF=0.285 , for which (0 < 0.285 < 0.2) is true
# vcf_addfilter( vcffile, "INFO", "AF", "FLT_CMP_00", 0, 0.2, "DROP" )
vcf_describefilters( vcffile )
vcf_readLineVecFiltered( vcffile ) # pos 50001
vcf_readLineVecFiltered( vcffile ) # pos 50002
#
# vcf_setregion(vcffile, "Y", 50000, 51000 )

#CORRECT rule:  
# filter out SNP at pos 50001 with INFO:AF=0.285 with rule "DROP if (0.2 < INFO:AF < 0.3)"
# vcf_ruleNsetrefvalues( vcffile , 0 , 0.2, 0.3 )
vcf_describefilters( vcffile )
vcf_readLineVecFiltered( vcffile ) # pos 50002
vcf_readLineVecFiltered( vcffile ) # pos 50003
```

---

**Description**

Set (vcf_selectsamples) or query (vcf_getselectedsamples) which individuals are included in the returned results, or get a list of selectable individuals.
vcf_selectsamples

Usage

vcf_selectsamples( vcffh, sampleslist )
vcf_getselectedsamples( vcffh )
vcf_getsamples( vcffh )

Arguments

vcffh VCFhandle type as returned by vcf_open
sampleslist A vector containing the identifiers of the individuals

Details

When reading variants from VCF files, it is possible to restrict the returned results to a certain subset of the available individuals (samples), e.g. members of a population or people with a certain trait. With vcf_selectsamples the currently selected subset of individuals can be set for a given VCF file. vcf_getselectedsamples returns the list of currently selected individuals and vcf_getsamples returns a list of all available identifiers in the file.

As with most other VCF functions, it is possible to call directly into the library to avoid some overhead. Use .Call("VCF_getSampleNames", vcffh ), .Call("VCF_getSelectedSamples", vcffh ) or .Call("VCF_selectSamples", vcffh, sampleslist ), respectively. Note the different names!

Value

A vector of strings representing the sample names selected or present in the VCF file.

Author(s)

Ulrich Wittelsbuerger

See Also

vcf_open

Examples

##
## Example:
##
vcffile <- vcf_open( system.file( "exdata", "ex.vcf.gz", package="WshopGenome" ) )
allsamplenames <- vcf_getsamples( vcffile )
vcf_selectsamples( vcffile, allsamplenames )
vcf_setregion

Set region from which to return genome variation data.

Description

Set region from which to return genome variation data.

Usage

vcf_setregion( vcffh, tid, from=NA, to=NA )

Arguments

vcffh  VCF file handle

tid     Either a chromosome identifier (from and to MUST be specified) or a region
        string (rendering from and to unnecessary)

from    Start position of the region from which to return data, if str is a chromosome
        identifier

to      End position of the region from which to return data, if str is a chromosome
        identifier

Details

Parameter 'regionstr' is of the form "chr:beg-end", e.g. "1:102910-210030" for chromosome 1,
positions >= 102910 and <= 210030. Use .Call("VCF_setRegion", vcffh, chromosomeid, from, to )
to eliminate the overhead of using the R wrapper function.

Value

TRUE or FALSE, whether the call succeeded or not.

Author(s)

Ulrich Wittelsbuerger

See Also

vcf_open vcf_getregion

Examples

## Example:
## vcffile <- vcf_open( system.file( "extdata" , "ex.vcf.gz" , package="WhopGenome" ) )

vcf_setregion(vcffile, "Y", 1, 100000 )
vcf_readLineVec( vcffile )
VCF_snpmat_diplo_bial_geno_filtered

Read SNP matrices in one of various representations.

Description

These functions read SNPs into matrices in a number of variations. All VCF_snpmat functions read data into the provided integer matrices, except for the <geno> format, which expects character/string-type matrices. The functions return TRUE if the call was successful, FALSE otherwise.

Each row corresponds to a sample, so make sure that the matrix you pass for <mat> has at least as many rows as selected samples.

Each column corresponds to a SNP. You can directly influence how many SNPs are read in at most by adjusting the number of columns of the matrices you pass. These functions try to read as many SNPs as possible from the currently active region and fill unused columns with the value -2.

If the given matrices have dimnames, the column names are set to the genomic position (from the VCF_column "POS") of the SNPs.

Usage

VCF_snpmat_diplo_bial_geno_filtered( vcffh, mat )
VCF_snpmat_diplo_anyal_geno_filtered( vcffh, mat )
VCF_snpmat_diplo_bial_geno_unfiltered( vcffh, mat )
VCF_snpmat_diplo_anyal_geno_unfiltered( vcffh, mat )
VCF_snpmat_diplo_bial_ishet_filtered( vcffh, mat )
VCF_snpmat_diplo_anyal_ishet_filtered( vcffh, mat )
VCF_snpmat_diplo_bial_ishet_unfiltered( vcffh, mat )
VCF_snpmat_diplo_anyal_ishet_unfiltered( vcffh, mat )
VCF_snpmat_diplo_bial_hasalt_filtered( vcffh, mat )
VCF_snpmat_diplo_bial_hasalt_unfiltered( vcffh, mat )
VCF_snpmat_diplo_anyal_hasalt_filtered( vcffh, mat )
VCF_snpmat_diplo_anyal_hasalt_unfiltered( vcffh, mat )
VCF_snpmat_diplo_bial_nucodes_filtered( vcffh, mat )
VCF_snpmat_diplo_bial_nucodes_unfiltered( vcffh, mat )
VCF_snpmat_diplo_anyal_nucodes_filtered( vcffh, mat )
VCF_snpmat_diplo_anyal_nucodes_unfiltered( vcffh, mat )
VCF_snpmat_anyplo_bial_nucodes_filtered( vcffh, mat )
VCF_snpmat_anyplo_bial_nucodes_unfiltered( vcffh, mat )
VCF_snpmat_anyplo_anyal_nucodes_filtered( vcffh, mat )
VCF_snpmat_anyplo_anyal_nucodes_unfiltered( vcffh, mat )

VCF_readIntoCodeMatrix( vcffh, mat )
read_snp_diplo_bial_int_altpresence( vcffh, mat )
read_snp_diplo_bial_int_nuclcodes( vcffh, mat )
read_snp_diplo_bial_str_allelechars( vcffh, mat )
read_snp_diplo_bial_str_01( vcffh, mat )
read_snp_diplo_bial_str_nuclcodes( vcffh, mat )

Arguments

vcffh VCF file handle as returned by VCF_open
mat Matrix to load data into

Details

The function names indicate what kind of data is read, how it is represented and whether filtering rules are applied. The names are constructed as follows: VCF_snpmat_[diploidy]_[allelicity]_[format]_[filtering]

For [diploidy] insert either diplo - SNPs from diploid data anyplo - SNPs of arbitrary ploidy.
For [allelicity] insert either bial - biallelic SNPs anyal - SNPs with an arbitrary number of alleles.
For [format] insert geno - genotype string ( typeof(mat) should be "character" ! ) ishet - 1 or 0 depending on whether the genotype is heterozygous or not hasalt - 1 or 0 depending on whether the genotype features the alternate allele (either homo- or heterozygous). nucodes - nucleotide code, where ACTGN- are represented by a number between 1 and 6.
For [filtering] insert filtered - drop lines not matching filtering rules unfiltered - do not drop any lines

Example: the function VCF_snpmat_diplo_bial_nucodes_filtered would read biallelic SNPs from diploid species data, turn their genotypes into numeric nucleotide codes and store them in an integer matrix. Only SNPs that passed the currently active filtering rules

For [format] geno, provide a matrix of type "character". For all other [format]s, provide a matrix of integer (not double!) type (typeof(mat) = "integer").

The following functions have become OBSOLETE:
VCF_readIntoCodeMatrix - use VCF_snpmat_diplo_bial_nucodes_filtered() instead.
read_snp_diplo_bial_int_altpresence - use VCF_snpmat_diplo_bial_hasalt_filtered() instead.
read_snp_diplo_bial_int_nuclcodes - use VCF_snpmat_diplo_bial_nucodes_filtered() instead.
read_snp_diplo_bial_str_allelechars( vcffh, mat ) - use VCF_snpmat_diplo_bial_geno_filtered() instead.
read_snp_diplo_bial_str_01( vcffh, mat ) - use VCF_snpmat_diplo_bial_hasalt_filtered() with integer matrix.
read_snp_diplo_bial_str_nuclcodes( vcffh, mat ) - use VCF_snpmat_diplo_bial_nucodes_filtered() with integer matrix.

Value
TRUE on success, FALSE if it failed.

Author(s)
Ulrich Wittelsbuerger
vcf_valid

See Also
vcf_addfilter

Examples

## Example:
vcffile <- vcf_open( system.file( "extdata" , "ex.vcf.gz" , package="WhopGenome" ) )
vcf_setregion(vcffile, "Y", 1, 100000 )

sn <- vcf_getsamples( vcffile )
vcf_selectsamples( vcffile , sn )

m <- matrix( data=as.integer(0) , nrow = length(sn) , ncol = 4 )

VCF_read_snp_diplo_bial_int_nuclcodes( vcffile , m )

vcf_valid Returns whether a VCF file handle is valid and usable.

Description
Returns whether a VCF file handle is valid and usable.

Usage
vcf_valid(vcffh)

Arguments
vcffh VCF handle

Value
TRUE or FALSE

Author(s)
Ulrich Wittelsbuerger

Examples
vcffile <- vcf_open( system.file( "extdata" , "ex.vcf.gz" , package="WhopGenome" ) )
vcf_valid( vcffile )
whop.eg.chromosome

Return the chromosome on which the gene identified by the given Entrez ID lies.

Description

Return the chromosome on which the gene identified by the given Entrez ID lies.

Usage

whop.eg.chromosome(id, db)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Entrez identifier</td>
</tr>
<tr>
<td>db</td>
<td>Organism database name, if not using currently activated one</td>
</tr>
</tbody>
</table>

whop.eg.abbrevForOrganism

Look up the organism prefix for the .org.eg.db databases from Bioconductor

Description

Look up the organism prefix for the .org.eg.db databases from Bioconductor

Usage

whop.eg.abbrevForOrganism(organismname)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>organismname</td>
<td>Name of organism</td>
</tr>
</tbody>
</table>

Details

Used internally.

Value

Database prefix

Author(s)

Ulrich Wittelsbuerger
whop.eg.eg_lookup

Value

Chromosome name

Author(s)

Ulrich Wittelsbuerger

---

Return all entries in an EG organism’s data table for all given identifiers

whop.eg.eg_lookup(ids, subdbname, db)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ids</td>
<td>Identifiers to look for</td>
</tr>
<tr>
<td>subdbname</td>
<td>Subtable to look in</td>
</tr>
<tr>
<td>db</td>
<td>Organism’s database if not using default currently active one</td>
</tr>
</tbody>
</table>

Value

Depends on table

Author(s)

Ulrich Wittelsbuerger

---

Return all entries in an EG organism’s data table for a given identifier

whop.eg.eg_lookupAll(id, subdbname, db)

Description

Return all entries in an EG organism’s data table for a given identifier

Usage

whop.eg.eg_lookupAll(id, subdbname, db)
**Arguments**

- **id**: Identifier(s) to look for in subtable
- **subdbname**: Organism annotation table name
- **db**: Optional, organism database if not using default active one

**Value**

Depends on table

**Author(s)**

Ulrich Wittelsbuerger

---

**Description**

Return the first entry in an EG organism’s data table for a given identifier

**Usage**

`whop.eg.eg_lookupSingle(id, subdbname, db)`

**Arguments**

- **id**: Identifiers to look for in subtable
- **subdbname**: Organism annotation table name
- **db**: Optional, organism database if not using default active one

**Value**

First entry with any of the given id(s) in the table

**Author(s)**

Ulrich Wittelsbuerger
whop.eg.eg_RevLookup

Perform a reverse lookup on one of the EG organism database’s sub-tables.

Description

Perform a reverse lookup on one of the EG organism database’s sub-tables.

Usage

whop.eg.eg_RevLookup(ids, subdbname, db)

Arguments

ids               Identifiers to look for in subtable
subdbname         Organism annotation table name
db                Optional, organism database if not using default active one

Value

Depends on data queried

Author(s)

Ulrich Wittelsbuerger

whop.eg.enzyme

Turn an Enzyme identifier into a Entrez identifier.

Description

Turn an Enzyme identifier into a Entrez identifier.

Usage

whop.eg.enzyme(id, db)

Arguments

id                Enzyme EC identifier
db                Organism database name, if not using currently activated one

Value

Entrez identifier(s)
**Author(s)**

Ulrich Wittelsbuerger

---

**whop.eg.fromAccnum**   *Turn a GenBank accession number into a Entrez identifier.*

**Description**

Turn a GenBank accession number into a Entrez identifier.

**Usage**

whop.eg.fromAccnum(id, db)

**Arguments**

id   GenBank accession

db   Organism database name, if not using currently activated one

**Value**

Entrez identifier(s)

---

**Author(s)**

Ulrich Wittelsbuerger

---

**whop.eg.fromAlias**   *Turn an Alias into a Entrez identifier.*

**Description**

Turn an Alias into a Entrez identifier.

**Usage**

whop.eg.fromAlias(id, db)

**Arguments**

id   Alias

db   Organism database name, if not using currently activated one

**Value**

Entrez identifier(s)
whop\_eg\_fromEnsembl

**Author(s)**

Ulrich Wittelsbuerger

---

Turn an Ensembl identifier into a Entrez identifier.

**Usage**

whop.eg.fromEnsembl(id, db)

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Ensembl identifier</td>
</tr>
<tr>
<td>db</td>
<td>Organism database name, if not using currently activated one</td>
</tr>
</tbody>
</table>

**Value**

Entrez identifier(s)

---

whop.eg.fromEnsemblProt

**Turn an Ensembl Protein identifier into a Entrez identifier.**

**Description**

Turn an Ensembl Protein identifier into a Entrez identifier.

**Usage**

whop.eg.fromEnsemblProt(id, db)

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Ensembl Protein identifier</td>
</tr>
<tr>
<td>db</td>
<td>Organism database name, if not using currently activated one</td>
</tr>
</tbody>
</table>
whop.eg.fromEnzyme

Value
Entrez identifier(s)

Author(s)
Ulrich Wittelsbuerger

whop.eg.fromEnsemblTrans

*Turn an Ensemble transcript identifier into a Entrez identifier.*

Description
Turn an Ensemble transcript identifier into a Entrez identifier.

Usage
whop.eg.fromEnsemblTrans(id, db)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Ensembl Transcript identifier</td>
</tr>
<tr>
<td>db</td>
<td>Organism database name, if not using currently activated one</td>
</tr>
</tbody>
</table>

Value
Entrez identifier(s)

Author(s)
Ulrich Wittelsbuerger

whop.eg.fromEnzyme

*Turn an Enzyme nomenclature identifier into a Entrez identifier.*

Description
Turn an Enzyme nomenclature identifier into a Entrez identifier.

Usage
whop.eg.fromEnzyme(id, db)
whop.eg.fromGO

Arguments
id
   Enzyme EC identifier
db
   Organism database name, if not using currently activated one

Value
Entrez identifier(s)

Author(s)
Ulrich Wittelsbuerger

whop.eg.fromGO  Turn a GO term identifier into a related Entrez identifier.

Description
Turn a GO term identifier into a related Entrez identifier.

Usage
whop.eg.fromGO(id, db)

Arguments
id
   GO term identifier
db
   Organism database to look in, if not using currently active one

Value
Entrez identifier(s)

Author(s)
Ulrich Wittelsbuerger
whop.eg.fromOmim

Return all Entrez identifiers related to a given GO term.

**Description**

Return all Entrez identifiers related to a given GO term.

**Usage**

whop.eg.fromGo2AllEgs(id, db)

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>GO term identifier</td>
</tr>
<tr>
<td>db</td>
<td>Organism database to look in, if not using currently active one</td>
</tr>
</tbody>
</table>

**Value**

Entrez identifier(s)

**Author(s)**

Ulrich Wittelsbuerger

---

whop.eg.fromOmim

Turn an OMIM identifier into a Entrez identifier.

**Description**

Turn an OMIM identifier into a Entrez identifier.

**Usage**

whop.eg.fromOmim(id, db)

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>OMIM identifier</td>
</tr>
<tr>
<td>db</td>
<td>Organism database to look in, if not using currently active one</td>
</tr>
</tbody>
</table>

**Value**

Entrez identifier(s)

**Author(s)**

Ulrich Wittelsbuerger
whop.eg.fromPath

whop.eg.fromPath  

Turn a KEGG pathway identifier into related Entrez identifiers.

Description

Turn a KEGG pathway identifier into related Entrez identifiers.

Usage

whop.eg.fromPath(id, db)

Arguments

id  
KEGG pathway identifier

db  
Organism database to look in, if not using currently active one

Value

Entrez identifier(s)

Author(s)

Ulrich Wittelsbuerger

whop.eg.fromPmid

whop.eg.fromPmid  

Turn an PMID identifier into a Entrez identifier.

Description

Turn an PMID identifier into a Entrez identifier.

Usage

whop.eg.fromPmid(id, db)

Arguments

id  
PMID identifier

db  
Organism database to look in, if not using currently active one

Value

Entrez identifier(s)

Author(s)

Ulrich Wittelsbuerger
whop.e fromRefseq  
*Turn a Refseq identifier into an Entrez identifier.*

**Description**

Turn a Refseq identifier into an Entrez identifier.

**Usage**

`whop.e fromRefseq(id, db)`

**Arguments**

- `id`  
  Refseq identifier
- `db`  
  Organism database to look in, if not using currently active one

**Value**

Entrez identifier(s)

**Author(s)**

Ulrich Wittelsbuerger

---

whop.e fromUnigene  
*Turn an Unigene identifier into a Entrez identifier.*

**Description**

Turn an Unigene identifier into an Entrez identifier.

**Usage**

`whop.e fromUnigene(id, db)`

**Arguments**

- `id`  
  Unigene identifier
- `db`  
  Organism database to look in, if not using currently active one

**Value**

Entrez identifier(s)

**Author(s)**

Ulrich Wittelsbuerger
whop.eg.fromUniprot  

**whop.eg.fromUniprot**  
*Turn an Uniprot identifier into a Entrez identifier.*

**Description**

Turn an Uniprot identifier into a Entrez identifier.

**Usage**

whop.eg.fromUniprot(id, db)

**Arguments**

<table>
<thead>
<tr>
<th>id</th>
<th>Uniprot identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>db</td>
<td>Organism database to look in, if not using currently active one</td>
</tr>
</tbody>
</table>

**Value**

Entrez identifier(s)

**Author(s)**

Ulrich Wittelsbuerger

---

whop.eg.genename  

**whop.eg.genename**  
*Find the gene name for a given Entrez identifier*

**Description**

Find the gene name for a given Entrez identifier

**Usage**

whop.eg.genename(id, db)

**Arguments**

<table>
<thead>
<tr>
<th>id</th>
<th>Entrez identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>db</td>
<td>Organism database name, if not using currently activated one</td>
</tr>
</tbody>
</table>

**Value**

Gene names

**Author(s)**

Ulrich Wittelsbuerger
whop.eg.goIds

*Returns GO term identifiers related to the given Entrez identifier.*

**Description**

Returns GO term identifiers related to the given Entrez identifier.

**Usage**

whop.eg.goIds(id, db)

**Arguments**

- **id**
  - Entrez identifier

- **db**
  - Organism database name, if not using currently activated one

**Value**

GO identifiers

**Author(s)**

Ulrich Wittelsbuerger

---

whop.eg.installdb

*Download and install the Bioconductor EG database for a given organism*

**Description**

Download and install the Bioconductor EG database for a given organism

**Usage**

whop.eg.installdb(organismname)

**Arguments**

- **organismname**
  - Organism name or abbreviation

**Details**

Attempts to automatically download and install an organism’s annotation database from Bioconductor
**whop.eg.keggpathways**

**Value**

Success status

**Author(s)**

Ulrich Wittelsbuerger

---

**whop.eg.keggpathways**  
*Look up KEGG pathway identifiers related to the given Entrez identifier.*

---

**Description**

Look up KEGG pathway identifiers related to the given Entrez identifier.

**Usage**

`whop.eg.keggpathways(id, db)`

**Arguments**

- `id`  
  Entrez identifier

- `db`  
  Organism database name, if not using currently activated one

**Value**

KEGG PATHWAY identifiers

**Author(s)**

Ulrich Wittelsbuerger

---

**whop.eg.load_orgdb**  
*Load and, if necessary, install a Bioconductor EG database for a given organism.*

---

**Description**

Load and, if necessary, install a Bioconductor EG database for a given organism.

**Usage**

`whop.eg.load_orgdb(organismname, install.if.missing = F)`
Arguments

organismname  Organism name or abbreviation
install.if.missing  Install database if not present locally?

Value

Success status

Author(s)

Ulrich Wittelsbuerger

whop.eg.Organism  Returns the organism’s name for which the current database-set contains information.

Description

Returns the organism’s name for which the current database-set contains information.

Usage

whop.eg.Organism()

Value

String: organism name

Author(s)

Ulrich Wittelsbuerger

whop.eg.orgdb_loaded  Find out whether a certain organism’s Bioconductor EG database has been loaded

Description

Find out whether a certain organism’s Bioconductor EG database has been loaded

Usage

whop.eg.orgdb_loaded(organismname)
whop.eg.region

Arguments

organismname  Organism's name

Value

TRUE or FALSE

Author(s)

Ulrich Wittelsbuerger

whop.eg.region  Look up the start and end of the gene identified by the given Entrez ID.

Description

Look up the start and end of the gene identified by the given Entrez ID.

Usage

whop.eg.region(id, db)

Arguments

id  Entrez identifier

db  Organism database name, if not using currently activated one

Value

Start and end positions

Author(s)

Ulrich Wittelsbuerger
whop.eg.selectOrganism

Select the organism to query with subsequent whop.eg calls and load the appropriate database(s).

Description

Select the organism to query with subsequent whop.eg calls and load the appropriate database(s).

Usage

whop.eg.selectOrganism(organismname, dontload = FALSE, install.if.missing = F)

Arguments

organismname  Organism to query
dontload      Whether to load the database
install.if.missing
              Whether to install the database, if it does not exist locally

Value

Success status

Author(s)

Ulrich Wittelsbuerger

whop.eg.toAccnum

Look up for an Entrez identifier the corresponding GenBank Accession number.

Description

Look up for an Entrez identifier the corresponding GenBank Accession number.

Usage

whop.eg.toAccnum(id, db)

Arguments

id          Entrez identifier
db          Organism database name, if not using currently activated one
whop.eg.toAlias

Value
Translated identifiers

Author(s)
Ulrich Wittelsbuerger

whop.eg.toAlias Look up the corresponding common alias for an Entrez identifier.

Description
Look up the corresponding common alias for an Entrez identifier.

Usage
whop.eg.toAlias(id, db)

Arguments
id Entrez identifier
db Organism database name, if not using currently activated one

Value
Translated identifiers

Author(s)
Ulrich Wittelsbuerger

whop.eg.toEnsembl Look up for an Entrez identifier the corresponding Ensembl identifiers.

Description
Look up for an Entrez identifier the corresponding Ensembl identifiers.

Usage
whop.eg.toEnsembl(id, db)

Arguments
id Entrez identifier
db Organism database name, if not using currently activated one
whop.eg.toEnsemblTrans

Value
Translated identifiers

Author(s)
Ulrich Wittelsbuerger

whop.eg.toEnsemblProt

Look up for an Entrez identifier the corresponding Ensembl Protein identifiers.

Description
Look up for an Entrez identifier the corresponding Ensembl Protein identifiers.

Usage
whop.eg.toEnsemblProt(id, db)

Arguments
id Entrez identifier
db Organism database name, if not using currently activated one

Value
Translated identifiers

Author(s)
Ulrich Wittelsbuerger

whop.eg.toEnsemblTrans

Look up for an Entrez identifier the corresponding Ensembl transcript identifiers.

Description
Look up for an Entrez identifier the corresponding Ensembl transcript identifiers.

Usage
whop.eg.toEnsemblTrans(id, db)
whop.eg.toEnzyme

Arguments

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Entrez identifier</td>
</tr>
<tr>
<td>db</td>
<td>Organism database name, if not using currently activated one</td>
</tr>
</tbody>
</table>

Value

Translated identifiers

Author(s)

Ulrich Wittelsbuerger

whop.eg.toEnzyme  Look up for an Entrez identifier the corresponding Enzyme identifiers.

Description

Look up for an Entrez identifier the corresponding Enzyme identifiers.

Usage

whop.eg.toEnzyme(id, db)

Arguments

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Entrez identifier</td>
</tr>
<tr>
<td>db</td>
<td>Organism database name, if not using currently activated one</td>
</tr>
</tbody>
</table>

Value

Translated identifiers

Author(s)

Ulrich Wittelsbuerger
whop.eg.toGO

*Look up for an Entrez identifier the corresponding GO terms.*

**Description**

Look up for an Entrez identifier the corresponding GO terms.

**Usage**

```
whop.eg.toGO(id, db)
```

**Arguments**

- **id**: Entrez identifier
- **db**: Organism database name, if not using currently activated one

**Value**

Translated identifiers

**Author(s)**

Ulrich Wittelsbuerger

---

whop.eg.toOmim

*Look up the OMIM identifier(s) corresponding to an Entrez identifier*

**Description**

Look up the OMIM identifier(s) corresponding to an Entrez identifier

**Usage**

```
whop.eg.toOmim(id, db)
```

**Arguments**

- **id**: Entrez identifier
- **db**: Organism database name, if not using currently activated one

**Value**

Translated identifiers

**Author(s)**

Ulrich Wittelsbuerger
whop.eg.toPath

Look up the Pathway identifier(s) corresponding to an Entrez identifier

Description
Look up the Pathway identifier(s) corresponding to an Entrez identifier

Usage
whop.eg.toPath(id, db)

Arguments
id Entrez identifier
db Organism database name, if not using currently activated one

Value
Translated identifiers

Author(s)
Ulrich Wittelsbuerger

whop.eg.toPmid

Look up the Uniprot identifier(s) corresponding to an Entrez identifier

Description
Look up the Uniprot identifier(s) corresponding to an Entrez identifier

Usage
whop.eg.toPmid(id, db)

Arguments
id Entrez identifier
db Organism database name, if not using currently activated one

Value
Translated identifiers

Author(s)
Ulrich Wittelsbuerger
whop.toRefseq

**Look up the Refseq identifier(s) corresponding to an Entrez identifier**

**Description**

Look up the Refseq identifier(s) corresponding to an Entrez identifier

**Usage**

whop.toRefseq(id, db)

**Arguments**

id  Entrez identifier
db  Organism database name, if not using currently activated one

**Value**

Translated identifiers

**Author(s)**

Ulrich Wittelsbuerger

whop.toUnigene

**Look up the Unigene identifier(s) corresponding to an Entrez identifier**

**Description**

Look up the Unigene identifier(s) corresponding to an Entrez identifier

**Usage**

whop.toUnigene(id, db)

**Arguments**

id  Entrez identifier
db  Organism database name, if not using currently activated one

**Value**

Translated identifiers

**Author(s)**

Ulrich Wittelsbuerger
**whop.eg.toUniprot**

*Look up the Uniprot identifier(s) corresponding to an Entrez identifier*

**Description**

Look up the Uniprot identifier(s) corresponding to an Entrez identifier

**Usage**

whop.eg.toUniprot(id, db)

**Arguments**

- **id**: Entrez identifier
- **db**: Organism database name, if not using currently activated one

**Value**

Translated identifiers

**Author(s)**

Ulrich Wittelsbuerger

---

**whop.go.all_genes_for_term**

*Returns all genes related to the given GO term*

**Description**

Returns all genes related to the given GO term

**Usage**

whop.go.all_genes_for_term(tomatch)

**Arguments**

- **tomatch**: GO term name

**Value**

- Genes

**Author(s)**

Ulrich Wittelsbuerger
whop.go.connect  

Establish a connection to the AmiGO database servers

**Description**

Establish a connection to the AmiGO database servers or an arbitrary one with the same database schema as the AmiGO DB.

**Usage**

```r
whop.go.connect(althost = NA, altport = NA, altuser = NA, altpass = NA,
altdb = NA, altdbdrivername=NA, dbdrvpkgnam=NA)
```

**Arguments**

- **althost**: Optional override for the hostname of the database server; default "mysql.ebi.ac.uk"
- **altport**: Optional override for the port to connect to on the database server; default 4085
- **altuser**: Optional override for the username to authenticate with; default "go_select"
- **altpass**: Optional override for the password to authenticate with; default "amigo"
- **altdb**: Optional override for the database name to connect to; default "go_latest"
- **altdbdrivername**: Optional override for the DBMS driver to use; default "MySQL"
- **dbdrvpkgnam**: Optional hint which R package provides the DBMS driver (e.g. " RMySQL" for the MySQL DBMS driver)

**Value**

Success status

**Author(s)**

Ulrich Wittelsbuerger

**References**

AmiGO database
whop.go.goid_like

Return GO terms with identifiers typographically similar to the given one

Description
Return GO terms with identifiers typographically similar to the given one

Usage
whop.go.goid_like(idmatch)

Arguments
idmatch   GO term

Value
GO terms

Author(s)
Ulrich Wittelsbuerger

whop.go.is_obsolete_byid
Check obsolescence of GO terms with similar accessions

Description
Returns all obsolete GO terms with similar accession

Usage
whop.go.is_obsolete_byid(idmatch)

Arguments
idmatch   accession

Value
GO terms

Author(s)
Ulrich Wittelsbuerger
whop.go.is_obsolete_byname

*Check obsolescence of GO terms with similar names*

**Description**
Check obsolescence of GO terms with similar names

**Usage**
whop.go.is_obsolete_byname(tomatch)

**Arguments**
tomatch GO term name

**Value**
All obsolete GO terms matching the description

**Author(s)**
Ulrich Wittelsbuerger

---

whop.go.load

*Load a GO term database from file*

**Description**
Load a GO term database from file

**Usage**
whop.go.load(filename = NA)

**Arguments**
filename Filename of a GO database

**Value**
TRUE if any data has been read, FALSE if not

**Author(s)**
Ulrich Wittelsbuerger
whop.go.match

Description
Return all GO terms matching the given one

Usage
whop.go.match(tofind)

Arguments
tofind GO term

Value
GO terms

Author(s)
Ulrich Wittelsbuerger

whop.go.terms_match

Description
Returns all terms with names similar to the given one.

Usage
whop.go.terms_match(tomatch)

Arguments
tomatch term

Value
GO terms

Author(s)
Ulrich Wittelsbuerger
whop.go.term_ancestors

*Description*

Returns all ancestors of the given GO term.

*Usage*

```python
whop.go.term_ancestors(tomatch)
```

*Arguments*

- `tomatch`: GO term

*Value*

GO terms

*Author(s)*

Ulrich Wittelsbuerger

---

whop.go.term_ancestors_similar

*Description*

Return ancestral GO terms of similarly named GO term.

*Usage*

```python
whop.go.term_ancestors_similar(tomatch)
```

*Arguments*

- `tomatch`: GO term

*Value*

GO terms

*Author(s)*

Ulrich Wittelsbuerger
**whop.go.term_children**  
*Return child terms of the given term*

**Description**
Return child terms of the given GO term

**Usage**
whop.go.term_children(tomatch)

**Arguments**
tomatch GO term

**Value**
Child terms

**Author(s)**
Ulrich Wittelsbuerger

---

**whop.go.term_synonyms**  
*Returns GO terms synonymous with the given term*

**Description**
Returns GO terms synonymous with the given term

**Usage**
whop.go.term_synonyms(tomatch)

**Arguments**
tomatch GO term

**Value**
GO terms

**Author(s)**
Ulrich Wittelsbuerger
**whop.kegg.pathway_url**  
*Produces a URL to the KEGG website for a certain pathway*

**Description**
For all KEGG pathway IDs given, a URL to the KEGG webpage for that pathway is returned.

**Usage**

```r
whop.kegg.pathway_url(pathwayids)
```

**Arguments**

- `pathwayids`  
  One or more KEGG pathway identifiers

**Value**

A string containing an URL or vector of URLs

**Author(s)**

Ulrich Wittelsbuerger

---

**whop.ped.daughtersOf**  
*Return all daughters of a given individual from a pedigree dataset*

**Description**

All individuals which are female and have at least one of the given IDs as either mother or father

**Usage**

```r
whop.ped.daughtersOf(p, lis)
```

**Arguments**

- `p`  
  The pedigree dataset
- `lis`  
  One or more individual IDs

**Value**

Table of rows from the pedigree

**Author(s)**

Ulrich Wittelsbuerger
whop.ped.entriesOf

Return all entries from a pedigree dataset matching the list of given identifiers.

Description
Returns pedigree data on all individuals given in parameter 2

Usage
whop.ped.entriesOf(p, invids)

Arguments
p The pedigree dataset
invids The identifiers of the individuals to extract

Value
Table of rows from the pedigree

Author(s)
Ulrich Wittelsbuerger

whop.ped.familyOf

Returns all members of an individuals family

Description
Returns all members of an individuals family

Usage
whop.ped.familyOf(p, lis)

Arguments
p The pedigree dataset
lis The individual(s) for which family members should be extracted

Value
Table of rows from the pedigree

Author(s)
Ulrich Wittelsbuerger
whop.ped.fathers  
*Return all fathers from a pedigree dataset*

**Description**

Returns pedigree data on all individuals which appear in the Paternal.ID column

**Usage**

```
whop.ped.fathers(p)
```

**Arguments**

- `p`  
The pedigree dataset

**Value**

Table of rows from the pedigree

**Author(s)**

Ulrich Wittelsbuerger

**See Also**

- `whop.ped.mothers`

whop.ped.females  
*Return all females from a pedigree dataset*

**Description**

Extracts all individuals with 'Sex' defined as female

**Usage**

```
whop.ped.females(p)
```

**Arguments**

- `p`  
The pedigree dataset

**Value**

Table of rows from the pedigree
whop.ped.fromPop

Author(s)
Ulrich Wittelsbuerger

See Also
whop.ped.males

---

whop.ped.fromPop  Return all individuals belonging to a given population

Description
All individuals with one of the given population IDs are returned as a pedigree table.

Usage
whop.ped.fromPop(p, popids)

Arguments
p  The pedigree dataset
popids  A vector with one or more population IDs

Value
Table of rows from the pedigree

Author(s)
Ulrich Wittelsbuerger

---

whop.ped.load  Load a pedigree dataset from a .PED file

Description
Returns a table with the pedigree data contained in the file

Usage
whop.ped.load(filename)

Arguments
filename  Name of the file containing the pedigree data
Details
Expects the given file to be of the PLINK .PED format, i.e. a file with tab-separated columns of which the first few are required to be of a certain order.

Value
Table with pedigree data

Author(s)
Ulrich Wittelsbuerger

References
PLINK .PED

See Also
whop.ped.save

whop.ped.males

Return only the male individuals from a pedigree dataset

Description
Extract all male individuals from a pedigree dataset that has been previously loaded with whop.ped.load()

Usage
whop.ped.males(p)

Arguments
p The pedigree dataset

Value
Table of rows from the pedigree

Author(s)
Ulrich Wittelsbuerger

See Also
whop.ped.females bgzf_compress
whop.ped.mothers

Get all mothers stored in a pedigree file

**Description**

All individuals which appear in the Maternal.ID column of the pedigree data

**Usage**

whop.ped.mothers(p)

**Arguments**

p The pedigree dataset

**Value**

Table of rows from the pedigree

**Author(s)**

Ulrich Wittelsbuerger

**See Also**

whop.ped.fathers

whop.ped.names

Get all individual names

**Description**

Returns a vector of strings, containing all Individual.IDs from the pedigree data

**Usage**

whop.ped.names(p)

**Arguments**

p The pedigree dataset

**Value**

A vector of strings, containing all Individual.IDs from the pedigree data

**Author(s)**

Ulrich Wittelsbuerger
whop.ped.save

Return the parents of individuals

Description

Looks for all individuals which are listed as parents of certain other individuals.

Usage

whop.ped.parentsOf(p, invids)

Arguments

p The pedigree dataset
invids One or more individuals’ identifiers from the dataset

Details

All individuals which appear in the Maternal.ID and Paternal.ID columns of the given individuals.

Value

Table of rows from the pedigree

Author(s)

Ulrich Wittelsbuerger

whop.ped.save

Save pedigree data to file

Description

Saves the pedigree dataset in p to a file.

Usage

whop.ped.save(p, filename)

Arguments

p The pedigree dataset
filename Name of the file to save into
**whop.ped.siblingsOf**

**Value**

None.

**Author(s)**

Ulrich Wittelsbuerger

**See Also**

whop.ped.load()

---

whop.ped.siblingsOf  
_Return list of siblings_

**Description**

From the dataset 'p', all individuals which list at least one entry in 'lis' as mother or father are returned.

**Usage**

whop.ped.siblingsOf(p, lis)

**Arguments**

- \( p \)  
The pedigree dataset
- \( lis \)  
One or more individual identifiers from the dataset

**Details**

All entries which list one of the individuals in parameter 'lis' as either mother or father are returned.

**Value**

Table of rows from the pedigree

**Author(s)**

Ulrich Wittelsbuerger
### whop.ped.sonsOf

*Returns all sons of the given individuals*

**Description**

All individuals in a pedigree data

**Usage**

```r
whop.ped.sonsOf(p, lis)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>The pedigree dataset to work on</td>
</tr>
<tr>
<td>lis</td>
<td>One or more individuals' identifiers from the dataset</td>
</tr>
</tbody>
</table>

**Details**

For each element in lis, finds all male individuals who refer to these elements as parent. Essentially combines a `whop.ped.males()` with a `whop.ped.siblingsOf()` call.

**Value**

Table of rows from the pedigree

**Author(s)**

Ulrich Wittelsbuerger

**See Also**

- `whop.ped.daughtersOf`

### whop.ucsc.geneInfo

*Return information from UCSC about a gene named precisely as specified*

**Description**

Information about a gene (and optionally required to be located on a certain chromosome) is returned.

**Usage**

```r
whop.ucsc.geneInfo(gen, chr = NA)
```
whop.ucsc.geneInfoSimilar

Arguments

gene name to query information about

chr If specified, the identifier of the chromosome, on which this gene is located

Details

Gene name, chromosome of origin, strand, and start and end positions of transcription site, coding sequence and exons are returned.

Value

geneName Gene name
name Gene identifier
chrom Chromosome, on which the gene is located
strand Whether this gene is located on the + or - strand
txStart Transcription start site
txEnd Transcription end

cdsStart Coding sequence start
cdsEnd coding sequence end

exonCount Number of exons of this gene
exonStarts comma-separated list of exon start position
exonEnds comma-separated list of exon end positions

Author(s)

Ulrich Wittelsbuerger

See Also

whop.ucsc.geneInfoSimilar

whop.ucsc.geneInfoSimilar

Return information UCSC has about any genes with similar names

Description

Information about any genes named similarly as specified in 'gen' (and optionally required to be located on chromosome 'chr') is returned.

Usage

whop.ucsc.geneInfoSimilar(gen, chr = NA)
whop.ucsc.genesForRegion

Arguments

- gen: Gene name to query information about
- chr: If specified, the identifier of the chromosome, on which this gene is located

Details

Gene name, chromosome of origin, strand, and start and end positions of transcription site, coding sequence and exons are returned.

Value

- genename: Gene name
- name: Gene identifier
- chrom: Chromosome, on which the gene is located
- strand: Whether this gene is located on the + or - strand
- txStart: Transcription start site
- txEnd: Transcription end
- cdsStart: Coding sequence start
- cdsEnd: coding sequence end
- exonCount: Number of exons of this gene
- exonStarts: comma-separated list of exon start position
- exonEnds: comma-separated list of exon end positions

Author(s)

Ulrich Wittelsbuerger

See Also

whop.ucsc.geneInfo

whop.ucsc.genesForRegion(chrom, beg, end)

Return a list of genes located in a certain region on a certain chromosome

Description

Details on all genes falling into the positions between 'beg' and 'end' on chromosome 'chrom' are returned.

Usage

whop.ucsc.genesForRegion(chrom, beg, end)
whop.ucsc.query

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>chrom</td>
<td>Chromosome on which to look in &quot;chr1&quot; notation</td>
</tr>
<tr>
<td>beg</td>
<td>First position of the region a gene may fall into</td>
</tr>
<tr>
<td>end</td>
<td>Last position of the region a gene may fall into</td>
</tr>
</tbody>
</table>

Details

Gene name, chromosome of origin, strand, and start and end positions of transcription site, coding sequence and exons are returned.

Value

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>geneName</td>
<td>Gene name</td>
</tr>
<tr>
<td>name</td>
<td>Gene identifier</td>
</tr>
<tr>
<td>chrom</td>
<td>Chromosome, on which the gene is located</td>
</tr>
<tr>
<td>strand</td>
<td>Whether this gene is located on the + or - strand</td>
</tr>
<tr>
<td>txStart</td>
<td>Transcription start site</td>
</tr>
<tr>
<td>txEnd</td>
<td>Transcription end</td>
</tr>
<tr>
<td>cdsStart</td>
<td>Coding sequence start</td>
</tr>
<tr>
<td>cdsEnd</td>
<td>Coding sequence end</td>
</tr>
<tr>
<td>exonCount</td>
<td>Number of exons of this gene</td>
</tr>
<tr>
<td>exonStarts</td>
<td>Comma-separated list of exon start positions</td>
</tr>
<tr>
<td>exonEnds</td>
<td>Comma-separated list of exon end positions</td>
</tr>
</tbody>
</table>

Author(s)

Ulrich Wittelsbuerger

whop.ucsc.query | Send a SQL query string to the UCSC Genome Browser SQL server

Description

The items given as parameters are concatenated into a SQL query string and sent to the UCSC Genome Browser SQL server.

Usage

whop.ucsc.query(...)

Arguments

... any number of strings and variables that will be pasted together to build the query string
Value

The returned value(s) from the UCSC Genome Browser.

Author(s)

Ulrich Wittelsbuerger

Examples

```r
##
## Example :
##
```
Index

*Topic package
  WhopGenome-package, 4

bgzf_compress, 5, 23, 90
fai_build, 6
fai_close, 7
fai_open, 8
fai_query2, 9
fai_query3 (fai_query), 9
fai_query4, 10
fai_query5 (fai_query), 10
fai_reopen, 11

read_snp_diplo_bial_int_altpresence
  (VCF_snpmat_diplo_bial_geno_filtered), vcf_getheaderline, 32
  53
read_snp_diplo_bial_int_nuclcodes
  (VCF_snpmat_diplo_bial_geno_filtered), 53
read_snp_diplo_bial_str_01
  (VCF_snpmat_diplo_bial_geno_filtered), 53
read_snp_diplo_bial_str_allelechars
  (VCF_snpmat_diplo_bial_geno_filtered), 53
read_snp_diplo_bial_str_nuclcodes
  (VCF_snpmat_diplo_bial_geno_filtered), 53

tabix_build, 12, 23
tabix_close, 14
tabix_getregion, 15
tabix_open, 14, 16, 16
tabix_read, 14, 16, 17
tabix_readraw (tabix_read), 17
tabix_reopen, 18
tabix_restartregion, 19
tabix_setregion, 20

vcf_addfilter, 21

vcf_buildindex, 23
vcf_clearfilters, 24
vcf_close, 25
vcf_countBiallelicSNPs (vcf_countSNPs), 26
vcf_countSNPs, 26
vcf_describefilters, 27
vcf_eor, 28
vcf_getAlt (vcf_getChrom), 29
vcf_getChrom, 29
vcf_getcontignames, 30
vcf_getfieldnames, 31
vcf_getFilter (vcf_getChrom), 29
vcf_getFormat (vcf_getChrom), 29
vcf_getID (vcf_getChrom), 29
vcf_getInfo (vcf_getChrom), 29
vcf_getInfoField (vcf_getChrom), 29
vcf_getnumcontigs, 32
vcf_getPos (vcf_getChrom), 29
vcf_getQual (vcf_getChrom), 29
vcf_getRef (vcf_getChrom), 29
vcf_getregion, 33, 52
vcf_getSample (vcf_getChrom), 29
vcf_getsamples (vcf_selectsamples), 50
vcf_getselectedsamples
  (vcf_selectsamples), 50
vcf_isINDEL, 34
vcf_isSNP, 30, 35
vcf_open, 36, 52
vcf_parseNextLine (vcf_parseNextSNP), 36
vcf_parseNextSNP, 36
VCF_read_snp_diplo_bial_int_altpresence, 40
VCF_read_snp_diplo_bial_int_nuclcodes
  (VCF_read_snp_diplo_bial_int_altpresence), 40
VCF_read_snp_diplo_bial_str_01
  (VCF_read_snp_diplo_bial_int_altpresence),
40
VCF_read.snp.diplo_bial_str_allelechars (VCF_read.snp.diplo.bial_int.altpresence),
VCF_snpmat.diplo.anyal.hasalt.unfiltered (VCF_snpmat.diplo.bial.geno.filtered),
40
VCF_read.snp.diplo.bial_str.nuclcodes (VCF_read.snp.diplo.bial_int.altpresence),
VCF_snpmat.diplo.anyal.ishet.filtered (VCF_snpmat.diplo.bial.geno.filtered),
40
VCF_readIntoCodeMatrix (VCF_snpmat.diplo.bial.geno.filtered),
VCF_snpmat.diplo.anyal.unfiltered (VCF_snpmat.diplo.bial.geno.filtered),
53
vcf_readLineDF, 37
vcf_readLineRaw, 38
vcf_readLineRawFiltered (vcf_readLineRaw), 38
vcf_readLineVec, 39
vcf_readLineVecFiltered (vcf_readLineVec), 39
vcf_reopen, 41
vcf_restartregion, 42
vcf_rule.disable, 43
vcf_rule.enable (vcf_rule.disable), 43
vcf_rule.setaction, 44
vcf_rule.setcolumn, 45
vcf_rule.setcomparison, 46
vcf_rule.setfield, 48
vcf_rule.setrefvalues, 49
vcf_selectsamples, 50
vcf_setregion, 52
VCF_snpmat.anypl.anyal.nuclcodes.filtered (VCF_snpmat.diplo.bial.geno.filtered),
VCF_snpmat.diplo.bial.ishet.filtered, 53
VCF_snpmat.anypl.anyal.nuclcodes.unfiltered (VCF_snpmat.diplo.bial.geno.filtered),
VCF_snpmat.diplo.bial.nuclcodes.filtered (VCF_snpmat.diplo.bial.geno.filtered),
53
VCF_snpmat.anypl.bial.nuclcodes.filtered (VCF_snpmat.diplo.bial.geno.filtered),
VCF_snpmat.diplo.bial.nuclcodes.unfiltered (VCF_snpmat.diplo.bial.geno.filtered),
53
VCF_snpmat.anypl.bial.nuclcodes.unfiltered (VCF_snpmat.diplo.bial.geno.filtered),
VCF_valid, 55
VCF_snpmat.diplo.anyal.geno.filtered (VCF_snpmat.diplo.bial.geno.filtered),
whop.eg.abbrevForOrganism, 56
VCF_snpmat.diplo.anyal.geno.unfiltered (VCF_snpmat.diplo.bial.geno.filtered),
whop.eg.chromosome, 56
VCF_snpmat.diplo.anyal.geno.unfiltered (VCF_snpmat.diplo.bial.geno.filtered), whop.eg.lookup, 57
whop.eg.lookupAll, 57
whop.eg.lookupSingle, 58
whop.eg.RevLookup, 59
VCF_snpmat.diplo.anyal.hasalt.filtered (VCF_snpmat.diplo.bial.geno.filtered),
whop.eg.enzyme, 59
VCF_snpmat.diplo.anyal.hasalt.filtered (VCF_snpmat.diplo.bial.geno.filtered),
whop.eg.fromAccnum, 60
whopNegNfromAlias, 60
whopNegNfromEnsembl, 61
whopNegNfromEnsemblProt, 61
whopNegNfromEnsemblTrans, 62
whopNegNfromEnzyme, 62
whopNegNfromGO, 63
whopNegNfromGO2Allele, 64
whopNegNfromOMIM, 64
whopNegNfromPath, 65
whopNegNfromPMid, 65
whopNegNfromRefSeq, 66
whopNegNfromUnigene, 66
whopNegNfromUniprot, 67
whopNegNgenename, 67
whopNegNgoids, 68
whopNegNinstallDB, 68
whopNegNkeggPathways, 69
whopNegNloadOrgDB, 69
whopNegNorganism, 70
whopNegNorgdb_loaded, 70
whopNegNregion, 71
whopNegNselectOrganism, 72
whopNegNtoAccNum, 72
whopNegNtoAlias, 73
whopNegNtoEnsembl, 73
whopNegNtoEnsemblProt, 74
whopNegNtoEnsemblTrans, 74
whopNegNtoEnzyme, 75
whopNegNtoGO, 76
whopNegNtoOMIM, 76
whopNegNtoPath, 77
whopNegNtoPMid, 77
whopNegNtoRefSeq, 78
whopNegNtoUnigene, 78
whopNegNtoUniprot, 79
whopNegNgeneInfo, 80
whopNegNgeneInfoSimilar, 80
whopNegNall_genes_for_term, 80
whopNegNconnect, 80
whopNegNis_obsolete_byid, 80
whopNegNis_obsolete_byname, 80
whopNegNload, 80
whopNegNmatch, 80
whopNegNterm_ancestors, 80
whopNegNterm_ancestors_similar, 80
whopNegNterm_children, 80
whopNegNterm_synonyms, 80
whopNegNterms_match, 80
whopNegNkeggPathway_url, 81
whopNegNdnaughtersOf, 81
whopNegNentriesOf, 81
whopNegNfamilys, 81
whopNegNfemales, 81
whopNegNfromPop, 81
whopNegNload, 81
whopNegNmales, 81
whopNegNmothers, 81
whopNegNnames, 81
whopNegNparentsOf, 81
whopNegNsave, 81
whopNegNsiblingsOf, 81
whopNegNsonsOf, 81
whopNegNucscgeneInfo, 82
whopNegNucscgeneInfoSimilar, 82
whopNegNucscgenesForRegion, 82
whopNegNucscquery, 82
WhopGenome (WhopGenome-package), 4
WhopGenome-package, 4