matchprobes

April 19, 2009

`basecontent` *Obtain the ATCG content of a gene*

Description

This function accepts a character vector representing the nucleotide sequences and computes the frequencies of each base (A, C, G, T).

Usage

```
basecontent(seq)
countbases(seq, dna = TRUE)
```

Arguments

`seq`	Character vector.
`dna`	Logical value indicating whether the sequence is DNA (`TRUE`) or RNA (`FALSE`)

Details

The base frequencies are calculated separately for each element of `x`. The elements of `x` can be in upper case, lower case or mixed.

Value

A matrix with 4 columns and `length(x)` rows. The columns are named `A`, `C`, `T`, `G`, and the values in each column are the counts of the corresponding bases in the elements of `x`. When `dna=FALSE`, the `T` column is replaced with a `U` column.

Author(s)

R. Gentleman, W. Huber, S. Falcon

See Also

`complementSeq`,`reverseSeq`, `revcompDNA`, `revcompRNA`

Examples

```
v<-c("AAACT", "GGGTT", "ggAtT")
basecontent(v)

countbases(v)
```

`combineAffyBatch` *A function to combine data from different Affymetrix genechip types into one AffyBatch.*

Description

The function takes a list of AffyBatches and their respective probe sequence information and merges the intensities from the matching probes only into one AffyBatch.

Usage

```
combineAffyBatch(batch, probepkg, newcdf, verbose=TRUE)
```

Arguments

`batch`	A `list` of AffyBatches.
`probepkg`	A character vector of the same length as `batch`, containing the names of the probe sequences libraries (for example, `hgu133aprobe`) that go with the AffyBatches.
`newcdf`	Character. The name of the name CDF environment that is to be created.
`verbose`	Logical. If TRUE, messages are printed to the console.

Details

Note that batch effects may play an important role, e.g. because of slightly but systematically different reagents or sample treatments in the experiments at led to the data in the different AffyBatches. Thus, please interprete the results of this function with caution.

Probe to probe-set mapping: after the common probes between all AffyBatches have been identified, their mapping into probe sets (=the new CDF environment) is obtained from the restriction of the mapping of the first element of `batch` (see variable `REFCHIP` in the code).

If you want to use this function, please have a look at its code and if necessary, check intermediate results.

Value

A list with two elements: `dat`, an `AffyBatch`, and `cdf`, an environment that contains the probe-set to probe mapping.

The phenoData slot of `dat` will be empty. This is because in most cases this requires some manual interference. You will need to construct a combined phenoData slot by yourself.

Author(s)

R. Gentleman, Wolfgang Huber

Examples

```
## see vignette!
```

`complementSeq` *Complementary sequence.*

Description

Function to obtain the complementary sequence.

Usage

```
complementSeq(seq, start=1, stop=0)
```

Arguments

`seq`	Character vector consisting of the letters A, C, G and T.
`start`	Numeric scalar: the sequence position at which to start complementing. If 1, start from the beginning.
`stop`	Numeric scalar: the sequence position at which to stop complementing. If 0, go until the end.

Details

The complemented sequence for each element of the input is computed and returned. The complement is given by the mapping: A -> T, C -> G, G -> C, T -> A.

An important special case is `start=13`, `stop=13`: If `seq` is a vector of 25mer sequences on an Affymetrix GeneChip, `complementSeq(seq, start=13, stop=13)` calculates the so-called *mismatch* sequences.

The function deals only with sequences that represent DNA. These can consist only of the letters `A`, `C`, `T` or `G`. Upper, lower or mixed case is allowed and honored.

Value

A character vector of the same length as `seq` is returned. Each component represents the transformed sequence for the input value.

Author(s)

R. Gentleman, W. Huber

See Also

`basecontent`, `reverseSeq`

Examples

```
seq <- c("AAACT", "GGGTT")
complementSeq(seq)

seq <- c("CGACTGAGACCAAGACCTACAACAG", "CCCGCATCATCTTTCCTGTGCTCTT")
complementSeq(seq, start=13, stop=13)
```

`getProbeDataAffy` *Read a data file describing the probe sequences on an Affymetrix genechip*

Description

Read a data file describing the probe sequences on an Affymetrix genechip

Usage

```
getProbeDataAffy(arraytype, datafile, pkgname = NULL, comparewithcdf = TRUE)
```

Arguments

`arraytype`	Character. Array type (e.g. 'HG-U133A')
`datafile`	Character with the filename of the input data file, or a connection (see example). If omitted a default name is constructed from `arraytype` (for details you will need to consult this function's source code).
`pkgname`	Character. Package name. If NULL the name is derived from `arraytype`.
`comparewithcdf`	Logical. If TRUE, run a consistency check against a CDF package of the same name (what used to be Laurent's "extraparanoia".)

Details

This function serves as an interface between the (1) representation of array probe information data in the packages that are generated by `makeProbePackage` and (2) the vendor- and possibly version-specific way the data are represented in `datafile`.

`datafile` is a tabulator-separated file with one row per probe, and column names `'Probe X'`, `'Probe Y'`, `'Probe Sequence'`, and `'Probe.Set.Name'`. See the vignette for an example.

Value

A list with three components

`dataEnv`	an environment which contains the data frame with the probe sequences and the other probe data.
`symVal`	a named list of symbol value substitutions which can be used to customize the man pages. See `createPackage`.
`pkgname`	a character with the package name; will be the same as the function parameter `pkgname` if it was specified; otherwise, the name is constructed from the parameter `arraytype`.

See Also

makeProbePackage

Examples

```
## Please refer to the vignette
```

`getProbeData_1lq` *Read a 1lq file for an Affymetrix genechip*

Description

Read a 1lq file for an Affymetrix genechip

Usage

```
getProbeData_1lq(arraytype, datafile, pkgname = NULL)
```

Arguments

`arraytype`	Character. Array type (e.g. 'Scerevisiaetiling)
`datafile`	Character. The filename of the input data file. If omitted a default name is constructed from `arraytype` (see this function's source code).
`pkgname`	Character. Package name. If NULL the name is derived from `arraytype`.

Details

This function serves as an interface between the (1) representation of array probe information data in the packages that are generated by `makeProbePackage` and (2) the vendor- and possibly version-specific way the data are represented in `datafile`.

Value

A list with three components

`dataEnv`	an environment which contains the data frame with the probe sequences and the other probe data.
`symVal`	a named list of symbol value substitutions which can be used to customize the man pages. See `createPackage`.
`pkgname`	a character with the package name; will be the same as the function parameter `pkgname` if it was specified; otherwise, the name is constructed from the parameter `arraytype`.

See Also

makeProbePackage

Examples

```
## makeProbePackage(
##     arraytype = "Scerevisiaetiling",
##     maintainer= "Wolfgang Huber <huber@ebi.ac.uk>",
##     version   = "1.1.0",
##     datafile  = "S.cerevisiae_tiling.1lq",
##     importfun = "getProbeData_1lq")
```

longestConsecutive *Obtain the length of the longest substring containing only 'letter'*

Description

This function accepts a character vector and computes the length of the longest substring containing only `letter` for each element of `x`.

Usage

```
longestConsecutive(seq, letter)
```

Arguments

`seq`	Character vector.
`letter`	Character vector of length 1, containing one single character.

Details

The elements of `x` can be in upper case, lower case or mixed. NAs are handled.

Value

An integer vector of the same length as `x`.

Author(s)

W. Huber

See Also

`complementSeq`,`basecontent`,`reverseSeq`

Examples

```
v = c("AAACTGTGFG", "GGGAATT", "CCAAAAAAAAAATT")
longestConsecutive(v, "A")
```

makeProbePackage *Make a package with probe sequence related data for microarrays*

Description

Make a package with probe sequence related data for microarrays

Usage

```
makeProbePackage(arraytype,
    importfun = "getProbeDataAffy",
    maintainer,
    version,
    species,
    pkgname = NULL,
    outdir  = ".",
    force = FALSE, quiet = FALSE,
    check = TRUE, build = TRUE, unlink = TRUE, ...)
```

Arguments

arraytype	Character. Name of array type (typically a vendor's name like "HG-U133A").
importfun	Character. Name of a function that can read the probe sequence data e.g. from a file. See getProbeDataAffy for an example.
maintainer	Character. Name and email address of the maintainer.
version	Character. Version number for the package.
species	Character. Species name in the format Genus_species (e.g., Homo_sapiens)
pkgname	Character. Name of the package. If missing, a name is created from arraytype.
outdir	Character. Path where the package is to be written.
force	Logical. If TRUE overrides possible warnings
quiet	Logical. If TRUE do not print statements on progress on the console
check	Logical. If TRUE call R CMD check on the package
build	Logical. If TRUE call R CMD build on the package
unlink	Logical. If TRUE unlink (remove) the check directory (only relevant if check=TRUE)
...	Further arguments that get passed along to importfun

Details

See vignette.

Important note for *Windows* users: Building and checking packages requires some tools outside of R (e.g. a Perl interpreter). While these tools are standard with practically every Unix, they do not come with MS-Windows and need to be installed separately on your computer. See http://www.murdoch-sutherland.com/Rtools. If you just want to build probe packages, you will not need the compilers, and the "Windows help" stuff is optional.

Examples

```
filename <- system.file("extdata", "HG-U95Av2_probe_tab.gz",
     package="matchprobes")
outdir   <- tempdir()
me       <- "Wolfgang Huber <huber@ebi.ac.uk>"
makeProbePackage("HG-U95Av2",
                 datafile   = gzfile(filename, open="r"),
                 outdir     = outdir,
                 maintainer = me,
                 version    = "0.0.1",
                 species    = "Homo_sapiens",
```

```
                  check       = FALSE,
                  force       = TRUE)
dir(outdir)
```

`matchprobes` *A function to match a query sequence to the sequences of a set of probes.*

Description

The `query` sequence, a character string (probably representing a transcript of interest), is scanned for the presence of exact matches to the sequences in the character vector `records`. The indices of the set of matches are returned.

The function is inefficient: it works on R's character vectors, and the actual matching algorithm is of time complexity `length(query)` times `length(records)`!

Usage

```
matchprobes(query, records, probepos=FALSE)
```

Arguments

`query`	A character vector. For example, each element may represent a gene (transcript) of interest. See Details.
`records`	A character vector. For example, each element may represent the probes on a DNA array.
`probepos`	A logical value. If TRUE, return also the start positions of the matches in the query sequence.

Details

`toupper` is applied to the arguments `query` and `records` before matching. The intention of this is to make the matching case-insensitive. The function is embarrassingly naive. The matching is done using the C library function `strstr`.

Value

A list. Its first element is a list of the same length as the input vector. Each element of the list is a numeric vector containing the indices of the probes that have a perfect match in the query sequence.

If `probepos` is TRUE, the returned list has a second element: it is of the same shape as described above, and gives the respective positions of the matches.

Author(s)

R. Gentleman, Laurent Gautier, Wolfgang Huber

Examples

```
if(require("hgu95av2probe")){
  data("hgu95av2probe")
  seq <- hgu95av2probe$sequence[1:20]
  target <- paste(seq, collapse="")
  matchprobes(target, seq, probepos=TRUE)
}
```

---

print.probetable *Print method for probetable objects*

Description

Prints class(x), nrow(x) and ncol(x), but not the elements of x. The motivation for having this method is that methods from the package base such as print.matrix and print.data.frame will try to print the values of all elements of x, which can take inconveniently much time and screen space if x is large.

Usage

```
## S3 method for class 'probetable':
print(x, ...)
```

Arguments

x	an object of S3-class probetable.
...	further arguments that get ignored.

See Also

print.matrix, print.data.frame

Examples

```
a = as.data.frame(matrix(runif(1e6), ncol=1e3))
class(a) = c("probetable", class(a))
print(a)
print(as.matrix(a[2:3, 4:6]))
```

---

reverseSeq *Reverse Sequence*

Description

Functions to obtain the reverse and reverse complement of a sequence

Usage

```
reverseSeq(seq)
revcompDNA(seq)
revcompRNA(seq)
```

Arguments

seq	Character vector. For revcompRNA and revcompDNA the sequence should consist of appropriate letter codes: [ACGUN] and ACGTN, respectively.

Details

The function reverses the order of the constituent character strings of its argument.

Value

A character vector of the same length as seq.

Author(s)

R. Gentleman, W. Huber, S. Falcon

See Also

basecontent,complementSeq

Examples

```
 w <-  c("hey there", "you silly fool")
 reverseSeq(w)

 w <- "able was I ere I saw Elba"
 reverseSeq(w)

rna1 <- "UGCA"
revcompRNA(rna1)

dna1 <- "TGCA"
revcompDNA(dna1)
```

Index

∗Topic **IO**
    getProbeData_llq, 5
    getProbeDataAffy, 4
    makeProbePackage, 6
∗Topic **manip**
    basecontent, 1
    combineAffyBatch, 2
    complementSeq, 3
    longestConsecutive, 6
    matchprobes, 8
    reverseSeq, 9
∗Topic **print**
    print.probetable, 9
∗Topic **utilities**
    getProbeData_llq, 5
    getProbeDataAffy, 4
    makeProbePackage, 6

AffyBatch, *2*

basecontent, 1, *3*, *6*, *10*

combineAffyBatch, 2
complementSeq, *1*, 3, *6*, *10*
countbases *(basecontent)*, 1
createPackage, *4, 5*

getProbeData_llq, 5
getProbeDataAffy, 4, *7*

longestConsecutive, 6

makeProbePackage, *4, 5*, 6
matchprobes, 8

print.data.frame, *9*
print.matrix, *9*
print.probetable, 9

revcompDNA, *1*
revcompDNA *(reverseSeq)*, 9
revcompRNA, *1*
revcompRNA *(reverseSeq)*, 9
reverseSeq, *1*, *3*, *6*, 9

toupper, *8*