`siever` Package Vignette

Comparative analysis of pathogen genomes infecting vaccine versus placebo recipients.

This package contains R routines to implement the categorical sieve analysis methods described in Gilbert, Wu, Jobes (2008) and variants thereof, some of which are described in deCamp and Edlefsen (2014). See below for methods that are currently implemented.

These methods are used to examine the effect of vaccination on infection as a function of a genomic feature of the pathogen (such as a sequence feature that makes an HIV-1 virus a target of vaccine-induced immunity). The methods implemented in this R package support comparative analysis of categorical data (usually amino acids) at one or multiple loci when the vaccine-targeted category is known and used as a reference. The methods compare weights reflecting some distance measure between the vaccine-targeted category and the category of each infecting pathogen across treatment groups, with weights usually given by an amino acid substitution matrix.

The present version implements only the non-parametric pooled-variance t test from the original article. The statistic $Z_2^A(i)$ is given in equation 4 of Gilbert, Wu, Jobes (2008). We also implement extensions and variations of the method to support multiple observed sequences per subject (using an arbitrary user-supplied function to compute per-subject weights from per-sequence weights) and grouped testing of multiple sites (using another arbitrary user-supplied function to compute per-site-set statistics from per-site t statistics). With these extensions, the package implements the Simplified Mismatch Bootstrap (SMMB) and Expected GWJ (EGWJ) methods described in deCamp and Edlefsen (2014) and k-mer and whole-gene variants.

CHANGES LIST

1. renamed weightopt to SieverParameters
2. added test-stat accumulator to SieverParameters

Here is an example of the typical usage.

```
> library("seqinr") # TODO: USE biostrings
> library("siever")
>
>
> # resetAssumptions <- function() {
> #   ## options to be manipulated in later tests
> #   weight.matrix <<- NULL
> #   site.sets.list <<- NULL
> #   use.f.test <<- FALSE
> #   return.t.test.result <<- FALSE
> #   weights.across.sites.in.a.set.init <<- 0
> #   weights.across.sites.in.a.set.accumulation.fn <<- sum
> #   vaccine.sequence.sets.list <<-  NULL
> #   placebo.sequence.sets.list <<- NULL
> #   mimic.smmb <<-  FALSE
> #   instead.return.weights <<- FALSE
> #   ## fixed background manipulations of assumptions
> #   if(mimic.smmb) {
> #      weights.across.sequences.in.a.set.accumulation.fn <<- sum
> #   } else {
> #      weights.across.sequences.in.a.set.accumulation.fn <<- mean
> #   }
> #   if(is.null(weight.matrix)) {
> #      acceptable.chars <<-
> #         c( "A","C","D","E","F","G","H","I","K","L","M","N","P","Q","R","S","T","V","W","Y","-" )
> #   } else {
> #      acceptable.chars <<- setdiff( rownames( weight.matrix ), 'X' )
> #   }
> #
> # }
> #
> # ## Here we run filterAcceptable.  Do we need to have an "acceptableChars" in weightopt?  It's used a
> # seqAmtx   <- siever:::filterAcceptable(siever:::getSeqMtx("control_A.fasta"))
> # seqBmtx   <- siever:::filterAcceptable(siever:::getSeqMtx("control_B.fasta"))
> # seqHXBmtx <- siever:::filterAcceptable(siever:::getSeqMtx("control_HXB2.fasta"))
> #
> # resetAssumptions()
> #
> # data.mtx <- rbind(seqAmtx, seqBmtx)
> # vaccine.indices <- 1:nrow(seqAmtx)
> # test.type <- "tTest"
> #
> # wOpt <-
> #    weightopt(weight.mtx = weight.matrix,
> #              mimic.smmb = mimic.smmb,
> #              site.sets.list= site.sets.list,
> #              obs.vSeq.sets.list= vaccine.sequence.sets.list,
> #              obs.pSeq.sets.list= placebo.sequence.sets.list,
> #              acceptable.chars = acceptable.chars, # TODO: USED ANYMORE?  REMOVE?
> #              within.sites.acc.fn = weights.across.sites.in.a.set.accumulation.fn,
> #              within.obs.acc.fn = weights.across.sequences.in.a.set.accumulation.fn
> #              )
> #
> #   GWJsieve(data.mtx=data.mtx,
> #             vaccine.indices=vaccine.indices,
> #             insert.char.vector=as.vector( seqHXBmtx ),
> #             test.type = test.type,
> #             perm.num = 0, # TODO: RENAME.  THIS IS num.perms
> #             wOpt=wOpt)
```