Package ‘alphabetr’

January 28, 2017

Type Package

Title Algorithms for High-Throughput Sequencing of Antigen-Specific T Cells

Version 0.2.2

Description Provides algorithms for frequency-based pairing of alpha-beta T cell receptors.

License AGPL (>= 3)

LazyData TRUE

Imports Rcpp (>= 0.12.3), clue (>= 0.3-50), dplyr (>= 0.4.3), multicool (>= 0.1-9)

LinkingTo Rcpp

Suggests knitr, rmarkdown

VignetteBuilder knitr

RoxygenNote 5.0.1

URL http://github.com/edwardslee/alphabetr

BugReports http://github.com/edwardslee/alphabetr

NeedsCompilation yes

Author Edward Lee [aut, cre]

Maintainer Edward Lee <email.edwardlee@gmail.com>

Repository CRAN

Date/Publication 2017-01-28 17:31:24

R topics documented:

`bagpipe` *Identify candidate alpha/beta pairs.*

Description

`bagpipe()` uses the alphabetr resampling procedure on sequencing data to identify candidate alpha/beta pairs. The procedure takes a subsample of the data without replacement, calculates association scores with `chain_scores`, and then for each well, uses the Hungarian algorithm to determine the most likely pairings for the chains found in the well. Each time this is done is a replicate, and the number of replicates is specified as an option. A threshold is then used to filter the candidate pairs that appear in proportion of the replicates larger than the threshold, resulting in the final list of candidate pairs. Bagpipe is an acronym for **b**ootstrapping **a**lphabetr **g**enerated **pai**rs **p**rocedur**e** (based on older versions that utilized bootstrapping)

Usage

```
bagpipe(alpha, beta, replicates = 100, frac = 0.75, bootstrap = FALSE)
```

Arguments

`alpha`	Matrix recording which alpha chains appear in each well of the data. See `create_data`.
`beta`	Matrix recording which beta chains appear in the each well of the data. See `create_data`.
`replicates`	The number of times the resampling procedure is repeated, i.e. the number of replicates. At least 100 replicates is recommended.
`frac`	The fraction of the wells resampled in each replicate. Default is 75% of the wells
`bootstrap`	Legacy option. Calls a bootstrapping strategy (which resamples with replacement) instead of sampling a subset without replacement.

Value

A n x 5 matrix where n is the number of clones determined by `bagpipe()`. Each row represents the chains of the clone. Columns 1 and 2 represent the beta chain indices of the clone. Columns 3 and 4 represent the alpha chain indices of the clone. Column 5 represents the number of replicates that this clone was found in. Column 5 is used to filter our the clones that have not be determined by a certain "threshold" proportion of replicates.

Note that column 1 == column 2 and column 3 == column 4 since `bagpipe()` does not try to determine dual-alpha or dual-beta clones.

Examples

```
 # see the help for create_clones() and create_data()
 clones <- create_clones(numb_beta = 1000,
                         dual_alpha = .3,
                         dual_beta = .06,
                         alpha_sharing = c(0.80, 0.15, 0.05),
                         beta_sharing  = c(0.75, 0.20, 0.05))
 dat <- create_data(clones$TCR, plate = 5,
                    error_drop = c(.15, .01),
                    error_seq  = c(.05, .001),
                    error_mode = c("lognormal", "lognormal"),
                    skewed = 10,
                    prop_top = 0.6,
                    dist = "linear",
                    numb_cells = matrix(c(50, 480), ncol = 2))

 ## Not run:
 # normally want to set replicates to 100 or more
 pairs <- bagpipe(alpha = dat$alpha,
               beta  = dat$beta,
               replicates = 5,
               frac = 0.75,
               bootstrap = FALSE)

 # using a threshold of 0.3 of replicates
 pairs <- pairs[pairs[, 5] > 0.3, ]

## End(Not run)
```

`chain_scores` *Calculate association scores between alpha and beta chain pairs.*

Description

`chain_scores()` calculates association scores between every pair of alpha and beta chains based on the number of concurrent well appearances each alpha and beta pair makes, scaled inversely by the number of unique chains in that well. See Lee et. al. for more information about this procedure.

Usage

```
chain_scores(data_a, data_b)
```

Arguments

`data_a`	Matrix recording which alpha chains appear in each well of the data. See `create_clones`.
`data_b`	Matrix recording which beta chains appear in the each well of the data. See `create_clones`.

Value

A list containing the alpha and beta association scores. Accessed with `list$ascores` and `list$bscores` respectively.

Examples

```
# see the help for create_clones() and create_data()
clones <- create_clones(numb_beta = 1000,
                        dual_alpha = .3,
                        dual_beta  = .06,
                        alpha_sharing = c(0.80, 0.15, 0.05),
                        beta_sharing  = c(0.75, 0.20, 0.05))
dat <- create_data(clones$TCR, plate = 5,
                   error_drop = c(.15, .01),
                   error_seq  = c(.05, .001),
                   error_mode = c("lognormal", "lognormal"),
                   skewed = 10,
                   prop_top = 0.6,
                   dist = "linear",
                   numb_cells = matrix(c(50, 480), ncol = 2))

#this is done internally in bagpipe()
scores <- chain_scores(data_a = dat$alpha, data_b = dat$beta)
scores <- scores$ascores + t(scores$bscores)
```

`combine_freq_results` *Combines the frequency estimation results from single TCR clones and dual TCR clones*

Description

`combine_freq_results()` combines the results of the frequency estimation performed on single TCR clones (from the output of `bagpipe`) and the frequency estimation performed on dual clones. The code will find the rows of the single TCR frequency results that are represented by the dual clones and replace them with the appropriate dual clone entry.

Usage

```
combine_freq_results(single, dual)
```

Arguments

`single` Frequency estimation results of single TCR clones (usually from the first time `freq_estimate` is called)

`dual` Frequency estimation results of dual TCR-alpha clones

Value

A data.frame with the same structure as the output of `freq_estimate`. If two single "clones" in the `single` data.frame is represented by a dual clone in `dual`, then it is removed and replaced with one row represented by the dual clone.

`create_clones` *Create a synthetic set of clones with a specific underlying clonal structure*

Description

`create_clones()` creates a set of (beta1, beta2, alpha1, alpha2) quadruples that represent the indices of the chains of clones. The function will take a fixed number of unique beta chains that are in the T cell population, and then use the degree of beta and alpha sharing to determine the number of unique alpha chains in the populations. These chains will then be randomly assigned to each other, with a proportion of them being dual TCR clones (i.e. alpha1 != alpha2 and/or beta1 != beta2), forming our random list of clones with their chain indices.

Usage

```
create_clones(numb_beta, dual_beta, dual_alpha, alpha_sharing, beta_sharing)
```

Arguments

`numb_beta` The number of unique betas in the clonal population

`dual_beta` The proportion of clone that are dual TCRbeta clones, i.e. has two distinct beta chains

`dual_alpha` The proportion of clones that are dual TCRalpha clones, i.e. has two distinct alpha chains

`alpha_sharing` A vector where the ith position represents the proportion of alpha chains that are shared by i clones; alpha chains can be shared by up to 7 clones

`beta_sharing` A vector where the ith position represents the proportion of beta chains that are shared by i clones; beta chains can be shared by up to 5 clones

Value

A list of four different matrices. Each matrix is has dimensions n x 4, where n is the total number of clones and each row represents the chains of a clone. Column 1 and column 2 are the beta index/indices of the beta chain(s) used by the clone. Column 3 and 4 are the alpha index/indices of the alpha chain(s) used by the clone. If a clone has a single beta chain, then col 1 and col 2 will be equal. If a clone has a single alpha chain, then col 3 and col 4 will be equal.

Examples

```
# Creating a population containing 1000 beta chains; 10% of clones with
# dual-beta TCRs and 30% of clones with dual TCRs; 75% beta shared by one
# clone, 20% by two clones, 5% by three clones; 80% alpha chains shared by
# one clone, 15% by two clones, and 5% by three clones

clones <- create_clones(numb_beta = 1000,
                        dual_alpha = .3,
                        dual_beta  = .06,
                        alpha_sharing = c(0.80, 0.15, 0.05),
                        beta_sharing  = c(0.75, 0.20, 0.05))
```

create_data *Simulate sequencing data obtained from the alphabetr approach with a specified clonal structure*

Description

create_data() simulates an alphabetr sequencing experiment by sampling clones from a clonal structure specified by the user. The clones are placed on a frequency distribution where a fixed number clones represents the top proportion of the population in frequency and the other clones represent the rest of the population in frequency. Different error models and different sampling strategies can be simulated as well.

Usage

```
create_data(TCR, plates, error_drop, error_seq, error_mode, skewed, prop_top,
  dist = "linear", numb_cells)
```

Arguments

TCR	The specified clonal structure, which can be created from create_clones.
plates	The number of plates of data.
error_drop	A vector of length 2 with the mean of the drop error rate and the sd of the drop error rate.
error_seq	A vector of length 2 with the mean of the in-frame error rate and the sd of the in-frame error rate.

error_mode — A vector of two strings determining the "mode" of the error models. The first element sets the mode of the drop errors, and the second element sets the mode of the in-frame errors. The two modes available are "constant" for a constant error rate and "lognormal" for error rates drawn from a lognormal distribution. If the mode is set to "constant" the sd specified in error_drop and/or error_seq will be ignored.

skewed — Number of clones represent the top proportion of the population by frequency (which is specified by prop_top).

prop_top — The proportion of the population in frequency represented by the number of clones specified by skewed.

dist — The distribution of frequency of the top clones. Currently only "linear" is available.

numb_cells — A two column matrix determining the sampling strategy of the experiment. The first column represents the number of cells per well, and the second column represents the number of wells with that sample size. The sum of column 2 should equal 96 times the number of plates.

Value

A list of length 2. The first element is a matrix representing the data of the alpha chains, and the second element is a matrix representing the data of beta chains. The matrix represents the sequencing data by representing the wells of the data by rows and the chain indices by column. Entry [i, j] of the matrix represents if chain j is found in well i (yes == 1, no == 0). e.g. if alpha chain 25 is found in well 10, then [10, 25] of the alpha matrix will be 1.

Examples

```
# see the help for create_clones() for details of this function call
clones <- create_clones(numb_beta = 1000,
                        dual_alpha = .3,
                        dual_beta = .06,
                        alpha_sharing = c(0.80, 0.15, 0.05),
                        beta_sharing  = c(0.75, 0.20, 0.05))

# creating a data set with 5 plates, lognormal error rates, 10 clones
# making up the top 60% of the population in frequency, and a constant
# sampling strategy of 50 cells per well for 480 wells (five 96-well plates)
dat <- create_data(clones$TCR, plate = 5,
                   error_drop = c(.15, .01),
                   error_seq  = c(.05, .001),
                   error_mode = c("lognormal", "lognormal"),
                   skewed = 10,
                   prop_top = 0.6,
                   dist = "linear",
                   numb_cells = matrix(c(50, 480), ncol = 2))
```

create_data_singlecells

Simulate sequencing data obtained single-cell sequencing

Description

`create_data_singlecells()` simulates a single-cell sequencing experiment by sampling clones from a clonal structure specified by the user and using the same error models and frequency distributions used in `create_data`. These functions are almost identical except this one simulates the sampling and sequencing of single T cells.

Usage

```
create_data_singlecells(TCR, plates = 5, error_drop = c(0.15, 0.01),
  error_seq = c(0.05, 0.01), error_mode = c("constant", "constant"),
  skewed = 15, prop_top = 0.5, dist = "linear")
```

Arguments

`TCR` The specified clonal structure, which can be created from `create_clones`.

`plates` The number of plates of data. The number of single-cells is 96 times `plates`.

`error_drop` A vector of length 2 with the mean of the drop error rate and the sd of the drop error rate.

`error_seq` A vector of length 2 with the mean of the in-frame error rate and the sd of the in-frame error rate.

`error_mode` A vector of two strings determining the "mode" of the error models. The first element sets the mode of the drop errors, and the second element sets the mode of the in-frame errors. The two modes available are "constant" for a constant error rate and "lognormal" for error rates drawn from a lognormal distribution. If the mode is set to "constant" the sd specified in `error_drop` and/or `error_seq` will be ignored.

`skewed` Number of clones represent the top proportion of the population by frequency (which is specified by `prop_top`).

`prop_top` The proportion of the population in frequency represented by the number of clones specified by `skewed`.

`dist` The distribution of frequency of the top clones. Currently only "linear" is available.

Value

A list of length 3. The first element is a matrix representing the data of the alpha chains ($alpha), and the second element is a matrix representing the data of beta chains ($beta). The matrix represents the sequencing data by representing the wells of the data by rows and the chain indices by column. Entry [i, j] of the matrix represents if chain j is found in well i (yes == 1, no == 0). e.g. if alpha chain 25 is found in well 10, then [10, 25] of the alpha matrix will be 1.

The third element of the list ($drop) is a matrix that records the index of the **clone** sampled in the well (col 1), records if a drop error occurred (col 2), and record if an in-frame error occurred (col 3).

Examples

```
# see the help for create_clones() for details of this function call
clones <- create_clones(numb_beta = 1000,
                     dual_alpha = .3,
                     dual_beta  = .06,
                     alpha_sharing = c(0.80, 0.15, 0.05),
                     beta_sharing  = c(0.75, 0.20, 0.05))

# creating a data set with 480 single cells, lognormal error rates, 10 clones
# making up the top 60% of the population in frequency, and a constant
# sampling strategy of 50 cells per well for 480 wells (five 96-well plates)
dat <- create_data_singlecells(clones$TCR, plate = 5,
                               error_drop = c(.15, .01),
                               error_seq  = c(.05, .001),
                               error_mode = c("lognormal", "lognormal"),
                               skewed = 10,
                               prop_top = 0.6,
                               dist = "linear")
```

---

dual_discrim_dual_likelihood

Calculate likelihood of two beta-sharing candidate alpha-beta pairs deriving from a dual clone

---

Description

dual_discrim_dual_likelihood() is used within dual_top to calculate the likelihood that two alpha-beta pairs identified by bagpipe sharing the same beta chain derive from a single dual-alpha clone (instead of two distinct clones sharing the same beta).

Usage

```
dual_discrim_dual_likelihood(est, err, numb_cells, numb_wells, binomials)
```

Arguments

est	Frequency estimate of the putative dual-alpha clone
err	Mean drop error rate
numb_cells	Vector containing the number of cells per well
numb_wells	Vector containing the number of wells with the sample sizes given by numb_cells
binomials	Calculations of the needed binomial coefficients; this is faster in R than in Rcpp (from my own tests)

Value

A numeric containing the negative log likelihood

dual_discrim_shared_likelihood

Calculate likelihood of two beta-sharing candidate alpha-beta pairs deriving from a dual clone

Description

`dual_discrim_shared_likelihood()` is used within `dual_top` to calculate the likelihood that two alpha-beta pairs identified by `bagpipe` sharing the same beta chain derive from a two distinct clones sharing the same beta chain dual-alpha clone (instead of a single dual-alpha clone)

Usage

```
dual_discrim_shared_likelihood(est1, est2, err, numb_cells, numb_wells,
  binomials, multinomials)
```

Arguments

`est1` Frequency estimate of the first alpha-beta clone

`est2` Frequency estimate of the second alpha-beta clone

`err` Mean drop error rate

`numb_cells` Vector containing the number of cells per well

`numb_wells` Vector containing the number of wells with the sample sizes given by `numb_cells`

`binomials` Calculations of the needed binomial coefficients; this is faster in R than in Rcpp (from my own tests)

`multinomials` Calculations of the needed multinomial coefficients; this is way faster in R due to vectorization

Value

A numeric containing the negative log likelihood

`dual_eval` *Calculate dual depths and false dual rates for simulated alphabetr experiments*

Description

`dual_eval()` is used in simulation situations to compare the duals determined by `dual_top` and `dual_tail` (which can be combined with `rbind()`) to the duals in the simulated T cell population.

Usage

```
dual_eval(duals, pair, TCR, number_skewed, TCR_dual)
```

Arguments

`duals` A 4 column matrix (col 1 + 2 = beta indices, col 3 + 4 = alpha indices) containing the indices of dual-alpha clones. The output of `dual_top` and `dual_tail` are in this form (and the outputs of these two functions can combined by using `rbind()`)

`pair` The output of `bagpipe`

`TCR` The clonal structure of the simulated T cell population. This is obtained by subsetting the `TCR` element of the output of `create_clones`

`number_skewed` The number of clones represent the top proportion of the T cell population by frequency (this is the same `number_skewed` argument used when `create_clones` is called)

`TCR_dual` The dual clones of the simulated T cell population. This is obtained by subsetting the `dual_alph` element of the output of `create_clones`

Value

A data.frame with the following columns:

- `fdr`, the false dual rate
- `numb_cand_duals`, the number of duals identified
- `adj_depth_top`, the adjusted dual depth of top clones
- `abs_depth_top`, the absolute dual depth of top clones
- `numb_correct_top`, the number of correctly identified dual clones in the top
- `numb_duals_ans_top`, the number of top dual clones in the simulated T cell population
- `numb_poss_top`, the number of top dual clones whose beta and both alpha chains were identified by `bagpipe()`
- `numb_unestimated_top`, number of top dual clones whose frequencies could not be calculated (usually because the clones appeared in every well of the data)
- `adj_depth_tail`, the adjusted dual depth of tail clones

- abs_depth_tail, the absolute dual depth of tail clones
- numb_correct_tail, the number of correctly identified tail clones
- numb_duals_ans_tail, the number of dual tail clones in the simulated T cell population
- numb_poss_tail, the number of tail dual cloens whose beta and both alpha chains were identified by bagpipe()
- numb_unestimated_tail, the number of tail clones whose frequencies could not be calculated

dual_tail *Discriminate between beta-sharing clones and dual-alpha TCR clones (optimized for rare clones)*

Description

dual_tail() distinguishes between clones that share a common beta chain and dual TCR clones with two productive alpha chains. The procedure tests the null hypothesis that two candidate alpha, beta pairs with the same beta represent two separate clones by using the frequency estimates to calculate the number of wells that both clones are expected to be in. This is compared to the actual number of wells that both clones appear in, and if the actual number is greater than the expected number, than the pairs are chosen to represent a dual TCR clone.

Usage

```
dual_tail(alpha, beta, freq_results, numb_cells)
```

Arguments

alpha	Matrix recording which alpha chains appear in each well of the data. See create_data.
beta	Matrix recording which beta chains appear in the each well of the data. See create_data.
freq_results	Output of freq_estimate.
numb_cells	Vector containing the number of cells sampled in the wells of each column of the plates.

Value

A n x 3 matrix where n is the number of candidate clones, column 1 is the beta index of the clone, and column 2-3 are the alpha indices of the clone

dual_top *Discriminate between beta-sharing clones and dual-alpha TCR clones (optimized for common clones)*

Description

`dual_top()` distinguishes between clones that share a common beta chain and dual TCR clones with two productive alpha chains. The procedure calculates the likelihood that two (alpha, beta) pairs (with common a beta chain) come from two distinct clones sharing the same beta chain vs the likelihood that the two pairs derive from a dual TCR-alpha clone. A significant difference between the two likelihoods is indicative of a dual alpha clone, and these clones are returned as dual clones.

Usage

```
dual_top(alpha, beta, pair, error, numb_cells)
```

Arguments

`alpha`	Matrix recording which alpha chains appear in each well of the data. See `create_data`.
`beta`	Matrix recording which beta chains appear in the each well of the data. See `create_data`.
`pair`	A matrix where each row is a beta/alpha pair, column 1 and 2 are the beta indices, and column 3 and 4 are the alpha indices, and column 5 is the proportion of replicates the clone was found in (or equal to -1 if the clone is dual)
`error`	The mean error "dropped" chain rate due to PCR or sequencing errors.
`numb_cells`	The number of cells per well in each column of the plates. Should be a vector of 12 elements.

Value

A matrix of dual-alpha clones, where col 1 and 2 are beta indices of the clone (which should be equal) and col 3 and 4 are alpha indices of the clone (which are different).

freq_estimate *Estimation of frequencies of clones identified by* `alphabetr`

Description

`freq_estimate()` estimates the frequencies of clones with confidence intervals by using a maximum likelihood approach. The function looks at the wells that a chains of a clone appear in and determines the most likely frequency that explains the data.

Usage

```
freq_estimate(alpha, beta, pair, error = 0.15, numb_cells)
```

Arguments

alpha	Matrix recording which alpha chains appear in each well of the data. See create_data.
beta	Matrix recording which beta chains appear in the each well of the data. See create_data.
pair	A matrix where each row is a beta/alpha pair, column 1 and 2 are the beta indices, and column 3 and 4 are the alpha indices, and column 5 is the proportion of replicates the clone was found in (or equal to -1 if the clone is dual)
error	The mean error "dropped" chain rate due to PCR or sequencing errors.
numb_cells	The number of cells per well in each column of the plates. Should be a vector of 12 elements.

Value

A data frame with frequency estimates and confidence intervals

Examples

```
 ## Not run:
 # obtained from the output of bagpipe()
 pairs <- pairs[pairs[, 5] > 0.3, ]
 freq  <- freq_estimate(alpha = dat$alpha,
                        beta = dat$beta,
                        pair = pairs,
                        numb_cells = matrix(c(50, 480), ncol = 2))

## End(Not run)
```

freq_eval *Calculate the precision, CV, and accuracy of frequency estimates*

Description

freq_eval() will evaluated how well freq_estimate performed by calculating the precision and CV of the frequency estimates for the top clones and by determining the proportion of the top clones whose true clonal frequency lies in the 95-percent CI determined by freq_estimate

Usage

```
freq_eval(freq, number_skewed, TCR, numb_clones, prop_top)
```

Arguments

`freq`	The output of `freq_estimate`
`number_skewed`	The number of clones represent the top proportion of the T cell population by frequency (this is the same `number_skewed` argument used when `create_clones` is called)
`TCR`	The clonal structure of the simulated T cell population. This is obtained by subsetting the `TCR` element of the output of `create_clones`
`numb_clones`	Total number of distinct clones in the parent population
`prop_top`	The proportion of the population in frequency represented by the number of clones specified by `skewed`.

Value

A list with the precision, cv, and accuracy of the frequency estimation.

`likelihood_dual` *Calculate likelihood curve of frequency estimates for a dual-alpha or dual-beta TCR clone*

Description

Calculate likelihood curve of frequency estimates for a dual-alpha or dual-beta TCR clone

Usage

```
likelihood_dual(est, err, numb_wells, numb_cells, numb_sample)
```

Arguments

`est`	Clonal frequency estimate
`err`	Mean drop error rate
`numb_wells`	A vector with the number of wells with the distinct sample sizes
`numb_cells`	A vector of the distinct sample sizes, i.e. the number of cells per well
`numb_sample`	A vector with the number of wells of the sample size of the same position of `numb_cells` that contains both alpha and the beta chain of the clone (for dual-alpha clones) or both beta and the alpha chain of the clone (for dual-beta clones)

Value

A numeric with the negative log likelihood

likelihood_dualdual *Calculate likelihood curve of frequency estimates for a dual-alpha and dual-beta TCR clone*

Description

Calculate likelihood curve of frequency estimates for a dual-alpha and dual-beta TCR clone

Usage

```
likelihood_dualdual(est, err, numb_wells, numb_cells, numb_sample)
```

Arguments

`est`	Clonal frequency estimate
`err`	Mean drop error rate
`numb_wells`	A vector with the number of wells with the distinct sample sizes
`numb_cells`	A vector of the distinct sample sizes, i.e. the number of cells per well
`numb_sample`	A vector with the number of wells of the sample size of the same position of `numb_cells` that contains both alpha and both betachains the clone

Value

A numeric with the negative log likelihood

likelihood_single *Calculate likelihood curve of frequency estimates for a single TCR clone*

Description

Calculate likelihood curve of frequency estimates for a single TCR clone

Usage

```
likelihood_single(est, err, numb_wells, numb_cells, numb_sample)
```

Arguments

`est`	Clonal frequency estimate
`err`	Mean drop error rate
`numb_wells`	A vector with the number of wells with the distinct sample sizes
`numb_cells`	A vector of the distinct sample sizes, i.e. the number of cells per well
`numb_sample`	A vector with the number of wells of the sample size of the same position of `numb_cells` that contains the alpha and beta chains the clone

Value

A numeric with the negative log likelihood

`read_alphabetr` *Read in alphabetr sequencing data into the binary matrix form needed by bagpipe()*

Description

`read_alphabetr()` will read in two different forms of a csv file to convert sequencing data using the alphabetr approach into the binary matrices required by `bagpipe`. The csv file(s) can have one of two forms. (1) A single csv file with three columns: column 1 containing whether the sequence is "TCRA" or "TCRB"; column 2 containing the well number; and column 3 containing the CDR3 sequence (2) Two CSV files, one for TCRA and one for TCRB, with two columns: column 1 containing the well number and column 2 containing the CDR3 sequence

Usage

```
read_alphabetr(data = NULL, data_alpha = NULL, data_beta = NULL)
```

Arguments

`data`	To read in a 3-column csv file containing both TCRA and TCRB sequencing information
`data_alpha`	To read in a 2-column csv file containing TCRA sequencing information. Must be used in conjunction with the `data_beta` argument and cannot be used with the `data` argument.
`data_beta`	To read in a 2-column csv file containing TCRB sequencing information. Must be used in conjunction with the `data_alpha` argument and cannot eb used with the `data` argument.

Value

A list of two binary matrices that represent the sequencing data and two character vectors that give the CDR3 sequences associated with each chain index.

Examples

```
## Not run:
  dat <- read_alphabetr(data = "alphabetr_data.csv")

  # saving the alpha and beta binary matrices
  data_alpha <- dat$alpha
  data_beta  <- dat$beta

  # finding the cdr3 sequences of alpha_2 and beta_4 respectively
  cdr3_alpha2 <- dat$alpha_lib[2]
  cdr3_beta4  <- dat$beta_lib[4]

## End(Not run)
```

Index