Package 'ISOpureR'

October 12, 2022

Version 1.1.3

Date 2019-05-10

Title Deconvolution of Tumour Profiles

Author Gerald Quon [aut],
Catalina V Anghel [aut, trl],
Syed Haider [aut],
Francis Nguyen [aut],
Amit G Deshwar [aut],
Quaid D Morris [aut],
Paul C Boutros [aut, cre]

Maintainer Paul C Boutros <pboutros@mednet.ucla.edu>

Depends R (>= 3.1.1)

Imports Rcpp (>= 0.11.3), stats, futile.logger

LinkingTo Rcpp, RcppEigen (>= 0.3.2.2.0)

Suggests knitr

VignetteBuilder knitr

Description Deconvolution of mixed tumour profiles into normal and cancer for each patient, using the ISOpure algorithm in Quon et al. Genome Medicine, 2013 5:29. Deconvolution requires mixed tumour profiles and a set of unmatched ``basis" normal profiles.

License GPL-2

NeedsCompilation yes

Repository CRAN

Date/Publication 2019-05-11 00:00:03 UTC

R topics documented:

ISOpure.calculate.tac *Perform calculation for Tumour Adjacent Cell (TAC) profiles*

Description

Performs the mathematical calculations taking bulk tumor data and deconvolved profiles and returning deconvolved tumour adjacent cell profiles.

Usage

```
ISOpure.calculate.tac(tumor.profiles, deconvolved.profiles, purity.estimates)
```

Arguments

tumor.profiles	a GxD matrix representing gene expression profiles of heterogeneous (mixed) tumor samples, where G is the number of genes, D is the number of tumor samples.
deconvolved.profiles	a GxD matrix representing gene expression profiles of purified (ISOpure output) tumor samples, where G is the number of genes, D is the number of tumor samples.
purity.estimates	a vector D representing the purity estimates (output from ISOpure)

Value

a GxD matrix representing gene expression profiles of purified (ISOpure output) tumor adjacent cell signal, where G is the number of genes, D is the number of tumor samples.

Author(s)

Natalie Fox

ISOpure.model_optimize.cg_code.rminimize *Minimize a differentiable multivariate function*

Description

This function is a conjugate-gradient search with interpolation/extrapolation by Carl Edward Rasmussen. A description of the Matlab code can be found at http://learning.eng.cam.ac.uk/carl/code/minimize/ (accessed Jan. 21, 2014). This is a implementation in R.

Usage

```
ISOpure.model_optimize.cg_code.rminimize(X, f, df, run_length, ...)
```

Arguments

X	The starting point is given by X which must be either a scalar or a column vector or matrix, not a row matrix
f	The name of the function to be minimized, returning a scalar
df	The name of the function which returns the vector of partial derivatives of f wrt X, where again the partial derivatives must be in scalar or column vector/matrix form
run_length	Gives the length of the run: if it is positive, it gives the maximum number of line searches, if negative its absolute gives the maximum allowed number of function evaluations. Note, for ISOpureR, used only positive run_length.
...	Parameters to be passed on to the function f.

Details

The function returns when either its length is up, or if no further progress can be made (ie, we are at a (local) minimum, or so close that due to numerical problems, we cannot get any closer). NOTE: If the function terminates within a few iterations, it could be an indication that the function values and derivatives are not consistent (ie, there may be a bug in the implementation of your "f" function).

The Polack-Ribiere flavour of conjugate gradients is used to compute search directions, and a line search using quadratic and cubic polynomial approximations and the Wolfe-Powell stopping criteria is used together with the slope ratio method for guessing initial step sizes. Additionally a bunch of checks are made to make sure that exploration is taking place and that extrapolation will not be unboundedly large.

Value

A list with three components:

X	The found solution X
fX	A vector of function values fX indicating the progress made
i	The number of iterations

Author(s)

Catalina Anghel, Francis Nguyen, Carl Edward Rasmussen

Examples

```
# Example from Carl E. Rasmussen's webpage

rosenbrock <- function(x){
D <- length(x);
  y <- sum(100*(x[2:D] - x[1:(D-1)]^2)^2 + (1-x[1:(D-1)])^2);
  return(y);
  };
drosenbrock <- function(x){
D <- length(x);
df <- numeric(D);
```

```
df[1:D-1] <- -400*x[1:(D-1)]*(x[2:D]-x[1:(D-1)]^2) - 2*(1-x[1:(D-1)]);
   df[2:D] <- df[2:D] + 200*(x[2:D]-x[1:(D-1)]^2);
   return(df);
};

ISOpure.model_optimize.cg_code.rminimize(c(0,0), rosenbrock, drosenbrock, 25)
#
# [[1]]
# [1] 1 1
#
# [[2]]
#  [1] 1.000000e+00 7.716094e-01 5.822402e-01 4.049274e-01 3.246633e-01
#  [6] 2.896041e-01 7.623420e-02 6.786212e-02 3.378424e-02 1.089908e-03
# [11] 1.087952e-03 8.974308e-05 1.218382e-07 6.756019e-09 3.870791e-15
# [16] 1.035408e-21 6.248025e-27 5.719242e-30 4.930381e-32
#
# [[3]]
# [1] 20
```

`ISOpure.model_optimize.vv.vv_deriv_loglikelihood`
Compute the derivative of the loglikelihood relevant to vv for step 1

Description

Computes the derivative of the loglikelihood function relevant to optimizing vv for step 1

Usage

```
ISOpure.model_optimize.vv.vv_deriv_loglikelihood(ww, sum_log_theta, DD)
```

Arguments

`ww`	log(vv-1), a Kx1 matrix
`sum_log_theta`	the column sums of log(theta), a 1xK matrix
`DD`	the number of patients (a scalar)

Value

The negative derivative of the part of the loglikelihood function relevant to vv with respect to (log) vv

Author(s)

Gerald Quon, Catalina Anghel, Francis Nguyen

ISOpure.model_optimize.vv.vv_loglikelihood *Compute the loglikelihood relevant to vv for step 1*

Description

Computes the part of the loglikelihood function relevant to optimizing vv for step 1

Usage

```
ISOpure.model_optimize.vv.vv_loglikelihood(ww, sum_log_theta, DD)
```

Arguments

ww	log(vv-1), a Kx1 matrix
sum_log_theta	the column sums of log(theta), a 1xK matrix
DD	the number of patients (a scalar)

Value

The negative of the loglikelihood relevant to vv

Author(s)

Gerald Quon, Catalina Anghel, Francis Nguyen

ISOpure.step1.CPE *Perform first step of ISOpure purification algorithm*

Description

Performs the first step of the ISOpure purification algorithm, taking tumor data normal profiles and returning the a list, ISOpureS1model, with all the updated parameters.

Usage

```
ISOpure.step1.CPE(tumordata, BB, PP, MIN_KAPPA, logging.level)
```

Arguments

tumordata	a GxD matrix representing gene expression profiles of heterogeneous (mixed) tumor samples, where G is the number of genes, D is the number of tumor samples.
BB	represents B = [b_1 ... b_(K-1)] matrix (from Genome Medicine paper) a Gx(K-1) matrix, where (K-1) is the number of normal profiles $(\beta_1, ..., \beta_(K-1))$, G is the number of genes. These are the normal profiles representing normal cells that contaminate the tumor samples (i.e. normal samples from the same tissue location as the tumor). The minimum element of BB must be greater than 0 – i.e. every gene/transcript must be observed on some level in each normal sample.
PP	a GxM matrix, representing the expression profiles whose convex combination form the prior over the purified cancer profile learned.
MIN_KAPPA	(optional) The minimum value allowed for the strength parameter kappa placed over the reference cancer profile m (see Quon et al, 2013). By default, this is set to 1/min(BB), such that the log likelihood of the model is always finite. However, when the min(BB) is very small, this forces MIN_KAPPA to be very large, and can sometimes cause the reference profile m to look too much like a 'normal profile' (and therefore you may observe the tumor samples having low % cancer content estimates). If this is the case, you can try setting MIN_KAPPA=1, or some other small value. For reference, for the data presented in Quon et al., 2013, MIN_KAPPA is on the order of 10^5.
logging.level	(optional) A string that gives the logging threshold for futile.logger. The possible options are 'TRACE', 'DEBUG', 'INFO', 'WARN', 'ERROR', 'FATAL'. Currently the messages in ISOpureR are only in the categories 'INFO', 'WARN', and 'FATAL', and the default setting is 'INFO'. Setting a setting for the entire package will over-ride the setting for a particular function.

Value

ISOpureS1model, a list with the following important fields:

theta	a DxK matrix, giving the fractional composition of each tumor sample. Each row represents a tumor sample that was part of the input, and the first K-1 columns correspond to the fractional composition with respect to the Source Panel contaminants. The last column represents the fractional composition of the pure cancer cells. In other words, each row sums to 1, and element (i,j) of the matrix denotes the fraction of tumor i attributable to component j (where the last column refers to cancer cells, and the first K-1 columns refer to different 'normal cell' components). The 'cancer', or tumor purity, estimate of each tumor is simply the last column of theta.
alphapurities	tumor purities (alpha_i in paper), same as the last column of the theta variable, pulled out for user convenience.
mm	reference cancer profile, in the form of parameters of a multinomial or discrete distribution (sum of elements is 1). This is the same as the purified cancer profile that ISOLATE was designed to learn.
omega	a Mx1 vector describing the convex combination weights learned by ISOpure step 1 over the PPtranspose matrix, that when applied to the Site of Origin Panel,

forms the prior over the reference cancer profile. When ISOpure step 1 is used in a similar fashion to the ISOLATE algorithm, entry i indicates the "probability" that the normal profile in the i-th column of PP is the site of origin of the secondary tumors stored in tumordata.

`total_loglikelihood`
: log likelihood of the model

`vv`
: (internal parameter) hyper-parameters from Dirichlet distribution, representing both mean and strength of a Dirichlet distribution over theta

`kappa`
: (internal parameter) the strength parameter over the Dirichlet distribution over the reference cancer parameter, mm

`mm_weights, theta_weights, omega_weights`
: (internal parameters) used in the optimization of mm, theta, and omega (instead of performing constrained optimization on these positively constrained variables directly, we optimize their logs in an unconstrained fashion.)

`log_BBtranspose, PPtranspose, log_all_rates:`
: (internal parameters) used in the calculations of loglikelihood

`MIN_KAPPA`
: (internal parameter) as described in the Arguments section

Author(s)

Gerald Quon, Catalina Anghel, Francis Nguyen

References

G Quon, S Haider, AG Deshwar, A Cui, PC Boutros, QD Morris. *Computational purification of individual tumor gene expression profiles*. Genome Medicine (2013) 5:29, http://genomemedicine.com/content/5/3/29.

G Quon, QD Morris. *ISOLATE: a computational strategy for identifying the primary origin of cancers using high-thoroughput sequencing*. Bioinformatics 2009, 25:2882-2889 http://bioinformatics.oxfordjournals.org/content/25/21/2882.

ISOpure.step2.PPE *Perform second step of ISOpure purification algorithm*

Description

Performs the second step of the ISOpure purification algorithm, taking tumor data and normal profiles and returning the a list, ISOpureS2model, with all the updated parameters.

Usage

```
ISOpure.step2.PPE(tumordata, BB, ISOpureS1model, MIN_KAPPA, logging.level)
```

Arguments

tumordata	(same as for ISOpureS1) a GxD matrix representing gene expression profiles of heterogeneous (mixed) tumor samples, where G is the number of genes, D is the number of tumor samples.
BB	(same as for ISOpureS1) represents B = [b_1 ... b_(K-1)] matrix (from Genome Medicine paper) a Gx(K-1) matrix, where (K-1) is the number of normal profiles $(\beta_1, ..., \beta_(K-1))$, G is the number of genes. These are the normal profiles representing normal cells that contaminate the tumor samples (i.e. normal samples from the same tissue location as the tumor). The minimum element of BB must be greater than 0 – i.e. every gene/transcript must be observed on some level in each normal sample.
ISOpureS1model	output model list from ISOpureS1 code
MIN_KAPPA	(optional) The minimum value allowed for the strength parameter kappa placed over the reference cancer profile m (see Quon et al, 2013). By default, this is set to 1/min(BB), such that the log likelihood of the model is always finite. However, when the min(BB) is very small, this forces MIN_KAPPA to be very large, and can sometimes cause the reference profile m to look too much like a 'normal profile' (and therefore you may observe the tumor samples having low % cancer content estimates). If this is the case, you can try setting MIN_KAPPA=1, or some other small value. For reference, for the data presented in Quon et al., 2013, MIN_KAPPA is on the order of 10^5.
logging.level	(optional) A string that gives the logging threshold for futile.logger. The possible options are 'TRACE', 'DEBUG', 'INFO', 'WARN', 'ERROR', 'FATAL'. Currently the messages in ISOpureR are only in the categories 'INFO', 'WARN', and 'FATAL', and the default setting is 'INFO'. Setting a setting for the entire package will over-ride the setting for a particular function.

Value

ISOpureS2model, a list with the following important fields:

theta	a DxK matrix, giving the fractional composition of each tumor sample. Each row represents a tumor sample that was part of the input, and the first K-1 columns correspond to the fractional composition with respect to the Source Panel contaminants. The last column represents the fractional composition of the pure cancer cells. In other words, each row sums to 1, and element (i,j) of the matrix denotes the fraction of tumor i attributable to component j (where the last column refers to cancer cells, and the first K-1 columns refer to different 'normal cell' components). The 'cancer', or tumor purity, estimate of each tumor is simply the last column of theta.
alphapurities	(same as ISOpureS1) tumor purities (alpha_i in paper), same as the last column of the theta variable, pulled out for user convenience - not changed in step 2
cc_cancerprofiles	purified cancer profiles. This matrix is of the same dimensionality as tumordata, and is also on the same scale (i.e. although ISOpureS2 treats purified cancer profiles as parameters of a multinomial distribution, we re-scale them to be on the

same scale as the input tumor profiles – see Genome Medicine paper). Column i of cc_cancerprofiles corresponds to column i of tumordata.

`total_loglikelihood`
: log likelihood of the model

`omega`
: (internal parameter, same as ISOpureS1) prior over the reference cancer profile - not changed in step 2

`vv`
: (internal parameter) hyper-parameters from Dirichlet distribution, representing both mean and strength of a Dirichlet distribution over theta

`kappa`
: (internal parameter) the strength parameter over the Dirichlet distribution over cc, given the reference cancer parameter, mm

`mm_weights, theta_weights, omega_weights`
: (internal parameters) used in the optimization of mm, theta, and omega (instead of performing constrained optimization on these positively constrained variables directly, we optimize their logs in an unconstrained fashion.)

`log_BBtranspose, PPtranspose, log_all_rates:`
: (internal parameters) used in the calculations of loglikelihood

`MIN_KAPPA`
: (internal parameter) as described in the Arguments section

Author(s)

Gerald Quon, Catalina Anghel, Francis Nguyen

References

G Quon, S Haider, AG Deshwar, A Cui, PC Boutros, QD Morris. *Computational purification of individual tumor gene expression profiles*. Genome Medicine (2013) 5:29, http://genomemedicine.com/content/5/3/29.

G Quon, QD Morris. *ISOLATE: a computational strategy for identifying the primary origin of cancers using high-thoroughput sequencing*. Bioinformatics 2009, 25:2882-2889 http://bioinformatics.oxfordjournals.org/content/25/21/2882.

`ISOpure.util.logsum` *Log-sum-exp*

Description

Prevents underflow/overflow using the log-sum-exp trick

Usage

```
ISOpure.util.logsum(xx, dimen);
```

Arguments

`xx`
: A matrix of numerical values

`dimen`
: The dimension along which the long sum is taken (1 for row, 2 for column)

Value

Returns log(sum(exp(x),dimen)), the log sum of exps, summing over dimension dimen but in a way that tries to avoid underflow/overflow.

Author(s)

Gerald Quon and Catalina Anghel

Examples

```
x <-  c(1, 1e20, 1e40, -1e40, -1e20, -1);
x <- as.matrix(x);

# compute log sum exp without the function
log(sum(exp(x)))
#[1] Inf

# compute log sum exp with the function
ISOpure.util.logsum(x, 1)
#[1] 1e+40
```

ISOpure.util.matlab_greater_than

Greater than operator

Description

Greater than function that matches Matlab behaviour when one of the arguments is NA (i.e. returns FALSE instead of NA)

Usage

```
ISOpure.util.matlab_greater_than(a, b)
```

Arguments

a	A numeric value (including Inf) or NA
b	A numeric value or NA

Value

Logical: TRUE if a > b, FALSE if a <= b OR if one of a, b is NA or NaN

Author(s)

Catalina Anghel

Examples

```
ISOpure.util.matlab_greater_than(5,3)
#[1] TRUE
ISOpure.util.matlab_greater_than(3,5)
#[1] FALSE
ISOpure.util.matlab_greater_than(5,NA)
#[1] FALSE
ISOpure.util.matlab_greater_than(NA,5)
#[1] FALSE
ISOpure.util.matlab_greater_than(5,Inf)
#[1] FALSE
ISOpure.util.matlab_greater_than(Inf,5)
#[1] TRUE
```

ISOpure.util.matlab_less_than

Less than operator

Description

Less than function that matches Matlab behaviour when one of the arguments is NA (i.e. returns FALSE instead of NA)

Usage

```
ISOpure.util.matlab_less_than(a, b)
```

Arguments

a	A numeric value (including Inf) or NA
b	A numeric value (including Inf) or NA

Value

Logical: TRUE if a < b, FALSE if a >= b OR if one of a, b is NA or NaN

Author(s)

Catalina Anghel

Examples

```
ISOpure.util.matlab_less_than(5,3)
#[1] FALSE
ISOpure.util.matlab_less_than(3,5)
#[1] TRUE
ISOpure.util.matlab_less_than(5,NA)
#[1] FALSE
```

```
ISOpure.util.matlab_less_than(NA,5)
#[1] FALSE
ISOpure.util.matlab_less_than(5,Inf)
#[1] TRUE
ISOpure.util.matlab_less_than(Inf,5)
#[1] FALSE
```

`ISOpure.util.matlab_log`

Modified logarithm function

Description

Logarithm function that matches Matlab behaviour on negative entries (i.e. returns a complex number)

Usage

```
ISOpure.util.matlab_log(x)
```

Arguments

x	A numeric or complex value, vector, or matrix.

Value

Returns log(x) if all entries of x > 0. For complex or negative input, x, where x = a + bi, the function returns log(z) = log(abs(z)) + 1i*atan2(b,a) where atan(b,a) is on the half-closed interval, (-pi, pi], as for the Matlab log function.

Author(s)

Catalina Anghel

Examples

```
ISOpure.util.matlab_log(5)
#[1] 1.609438
ISOpure.util.matlab_log(-5)
#[1] 1.609438+3.141593i
ISOpure.util.matlab_log(complex(real=3, imaginary=4))
#[1] 1.609438+0.927295i
ISOpure.util.matlab_log(c(2,3,4,-7,1))
#[1] 0.6931472+0.000000i 1.0986123+0.000000i 1.3862944+0.000000i
#[4] 1.9459101+3.141593i 0.0000000+0.000000i
```

`ISOpure.util.repmat` *Tiles matrix horizontally or vertically*

Description

Tiles matrix horizontally or vertically in the same way as the Matlab repmat command

Usage

```
ISOpure.util.repmat(a, n, m)
```

Arguments

`a`	A matrix
`n`	Number of times the matrix should be tiled horizontally
`m`	number of times the matrix should be tiled vertically

Value

A matrix which has replicated and tiled the input matrix a by n rows and m columns

Author(s)

Catalina Anghel, Ohloh (now Black Duck Open Hub)

Examples

```
x <- matrix(runif(6), 3, 2)
x
#           [,1]      [,2]
# [1,] 0.5167029 0.7543404
# [2,] 0.9064936 0.4316977
# [3,] 0.3256870 0.5310625
ISOpure.util.repmat(x, 1, 2)
#           [,1]      [,2]      [,3]      [,4]
# [1,] 0.5167029 0.7543404 0.5167029 0.7543404
# [2,] 0.9064936 0.4316977 0.9064936 0.4316977
# [3,] 0.3256870 0.5310625 0.3256870 0.5310625
ISOpure.util.repmat(x, 2, 1)
#           [,1]      [,2]
# [1,] 0.5167029 0.7543404
# [2,] 0.9064936 0.4316977
# [3,] 0.3256870 0.5310625
# [4,] 0.5167029 0.7543404
# [5,] 0.9064936 0.4316977
# [6,] 0.3256870 0.5310625
ISOpure.util.repmat(x, 2, 3)
#           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
# [1,] 0.5167029 0.7543404 0.5167029 0.7543404 0.5167029 0.7543404
```

```
# [2,] 0.9064936 0.4316977 0.9064936 0.4316977 0.9064936 0.4316977
# [3,] 0.3256870 0.5310625 0.3256870 0.5310625 0.3256870 0.5310625
# [4,] 0.5167029 0.7543404 0.5167029 0.7543404 0.5167029 0.7543404
# [5,] 0.9064936 0.4316977 0.9064936 0.4316977 0.9064936 0.4316977
# [6,] 0.3256870 0.5310625 0.3256870 0.5310625 0.3256870 0.5310625
```

`ISOpureS1.model_core.compute_loglikelihood`
Compute loglikelihood given all model parameters for step 1

Description

Computes complete loglikelihood given all model parameters for step 1

Usage

```
ISOpureS1.model_core.compute_loglikelihood(tumordata, model)
```

Arguments

`tumordata`	a GxD matrix representing gene expression profiles of tumor samples
`model`	list containing all the parameters updated in ISOpure step one iterations

Value

The scalar value of the complete loglikelihood obtained given the model parameters

Author(s)

Gerald Quon, Catalina Anghel, Francis Nguyen

`ISOpureS1.model_core.new_model`
Initialize a model list to hold all the parameters

Description

Produces a list (the model) which initializes the parameters vv, log_BBtranspose, PPtranspose, kappa, theta, omega, log_all_rates for step 1

Usage

```
ISOpureS1.model_core.new_model(tumordata, kappa, INITIAL_VV, PPtranspose, BBtranspose)
```

Arguments

tumordata	a GxD matrix representing gene expression profiles of tumor samples
kappa	scalar strength parameter kappa placed over the reference cancer profile mm
INITIAL_VV	a vector with K components, the prior over mixing proportions, theta, with last entry weighed more heavily
PPtranspose	a (K-1)xG matrix, standardized so that all entries sum to 1, see ISOpure.step1.CPE.R
BBtranspose	a (K-1)xG matrix of the standardized normal profiles, so that they sum to 1

Value

model	a newly generated model list to hold all the parameters vv, log_BBtranspose, PPtranspose, kappa, theta, omega, log_all_rates

Author(s)

Gerald Quon, Catalina Anghel, Francis Nguyen

ISOpureS1.model_core.optmodel

Optimizes the ISOpure parameters for step 1

Description

Optimizes the ISOpure parameters for step 1 cyclically until convergence

Usage

```
ISOpureS1.model_core.optmodel(tumordata, model, NUM_ITERATIONS=35)
```

Arguments

tumordata	a GxD matrix representing gene expression profiles of tumor samples
model	list containing all the parameters to be optimized
NUM_ITERATIONS	(optional) minimum number of iterations of optimization algorithm, default is 35

Value

model	updated model list containing all the parameters

Author(s)

Gerald Quon, Catalina Anghel, Francis Nguyen

ISOpureS1.model_optimize.kappa.kappa_compute_loglikelihood

Compute loglikelihood relevant to kappa for step 1

Description

Computes the part of the loglikelihood function relevant to optimizing kappa for step 1

Usage

```
ISOpureS1.model_optimize.kappa.kappa_compute_loglikelihood(kappa, tumordata, model)
```

Arguments

kappa	a scalar kappa, the strength parameter in the prior over the reference cancer profile
tumordata	a GxD matrix representing gene expression profiles of tumour samples
model	list containing all the parameters to be optimized

Value

The part of the loglikelihood function relevant to optimizing kappa

Author(s)

Gerald Quon, Catalina Anghel, Francis Nguyen

ISOpureS1.model_optimize.kappa.kappa_deriv_loglikelihood

Compute derivative of loglikelihood with respect to kappa for step 1

Description

Computes the derivative of the part of the loglikelihood function relevant to optimizing kappa for step 1. Instead of performing constrained optimization on kappa directly, we optimize the log of kappa in an unconstrained fashion. Thus, if y=log(kappa) and L is the loglikelihood function w.r.t. y, to optimize L w.r.t. y, dL/dy = dL/dkappa * dkappa/dy, where dkappa/dy = exp(y)= exp(log(kappa)). The input into the derivative function is log(kappa - model\$MIN_KAPPA).

Usage

```
ISOpureS1.model_optimize.kappa.kappa_deriv_loglikelihood(log_kappa, tumordata, model)
```

Arguments

log_kappa	the scalar log(kappa - model\$MIN_KAPPA)
tumordata	a GxD matrix representing gene expression profiles of tumour samples
model	list containing all the parameters to be optimized

Value

The negative derivative of the part of the loglikelihood function relevant to kappa with respect to log kappa (a scalar given that for step 1 of ISOpure kappa is a scalar)

Author(s)

Gerald Quon, Catalina Anghel, Francis Nguyen

ISOpureS1.model_optimize.kappa.kappa_loglikelihood

Compute loglikelihood relevant to kappa for step 1

Description

Computes the part of the loglikelihood function relevant to optimizing kappa for step 1. Instead of performing constrained optimization on kappa directly, we optimize the log of kappa in an unconstrained fashion.

Usage

```
ISOpureS1.model_optimize.kappa.kappa_loglikelihood(log_kappa, tumordata, model)
```

Arguments

log_kappa	the scalar log(kappa - model\$MIN_KAPPA)
tumordata	a GxD matrix representing gene expression profiles of tumour samples
model	list containing all the parameters to be optimized

Value

The negative of the loglikelihood relevant to optimizing kappa

Author(s)

Gerald Quon, Catalina Anghel, Francis Nguyen

`ISOpureS1.model_optimize.mm.mm_deriv_loglikelihood` *Compute the derivative of the loglikelihood relevant to mm for step 1*

Description

Computes the derivative of the loglikelihood function relevant to optimizing the reference cancer profile, mm, for step 1

Usage

```
ISOpureS1.model_optimize.mm.mm_deriv_loglikelihood(ww, tumordata, model)
```

Arguments

`ww`	the mm_weights, with G entries
`tumordata`	a GxD matrix representing gene expression profiles of tumor samples
`model`	list containing all the parameters to be optimized

Value

The negative derivative the likelihood function relevant to optimizing mm. The derivative is taken not with respect to mm but with respect to unconstrained variables via a change of variables.

Author(s)

Gerald Quon, Catalina Anghel, Francis Nguyen

`ISOpureS1.model_optimize.mm.mm_loglikelihood` *Compute the loglikelihood relevant to mm for step 1*

Description

Computes the loglikelihood function relevant to optimizing the reference cancer profile, mm, for step 1

Usage

```
ISOpureS1.model_optimize.mm.mm_loglikelihood(ww, tumordata, model)
```

Arguments

`ww`	the mm_weights, with G entries
`tumordata`	a GxD matrix representing gene expression profiles of tumor samples
`model`	list containing all the parameters to be optimized

Value

The negative of the likelihood function relevant to optimizing mm.

Author(s)

Gerald Quon, Catalina Anghel, Francis Nguyen

`ISOpureS1.model_optimize.omega.omega_compute_loglikelihood`
Compute loglikelihood relevant to omega for step 1

Description

Computes the part of the loglikelihood function relevant to optimizing omega for step 1

Usage

```
ISOpureS1.model_optimize.omega.omega_compute_loglikelihood(omega, tumordata, model)
```

Arguments

`omega`	(K-1)x1 matrix representing the weights of the normal profiles B_i used to make the weighted combination that forms the mean parameter vector for the Dirichlet distribution over m
`tumordata`	a GxD matrix representing gene expression profiles of tumor samples
`model`	list containing all the parameters to be optimized

Value

The part of the loglikelihood function relevant to optimizing omega

Author(s)

Gerald Quon, Catalina Anghel, Francis Nguyen

ISOpureS1.model_optimize.omega.omega_deriv_loglikelihood

Compute the derivative of loglikelihood relevant to omega for step 1

Description

Compute the derivative of the part of the loglikelihood function relevant to omega with respect to (log) omega, in step 1. Instead of performing constrained optimization on omega directly, we optimize the log of omega in an unconstrained fashion.

Usage

```
ISOpureS1.model_optimize.omega.omega_deriv_loglikelihood(ww, tumordata, model)
```

Arguments

ww	(K-1)x1 matrix, log(omega), where the entries in omega are constrained to add to 1 where K-1 is the number of normal samples
tumordata	a GxD matrix representing gene expression profiles of tumor samples
model	list containing all the parameters to be optimized

Value

The negative derivative of the part of the loglikelihood function relevant to omega with respect to (log) omega

Author(s)

Gerald Quon, Catalina Anghel, Francis Nguyen

ISOpureS1.model_optimize.omega.omega_loglikelihood

Compute the loglikelihood relevant to omega for step 1

Description

Compute the the part of the loglikelihood function relevant to omega in step 1

Usage

```
ISOpureS1.model_optimize.omega.omega_loglikelihood(ww, tumordata, model)
```

Arguments

ww	(K-1)x1 matrix, log(omega), where the entries in omega are constrained to add to 1 where K-1 is the number of normal samples
tumordata	a GxD matrix representing gene expression profiles of tumor samples
model	list containing all the parameters to be optimized

Value

The negative of the loglikelihood function relevant to omega

Author(s)

Gerald Quon, Catalina Anghel, Francis Nguyen

ISOpureS1.model_optimize.opt_kappa

Optimize kappa in step 1

Description

This function optimizes kappa, the strength parameter in the prior over the reference cancer profile. Note that we don't directly optimize kappa because it has constraints (must be greater than the minimum determined in ISOpure.step1.CPE.)

Usage

```
ISOpureS1.model_optimize.opt_kappa(
tumordata,
model,
NUM_ITERATIONS_RMINIMIZE,
iter,
NUM_GRID_SEARCH_ITERATIONS
)
```

Arguments

tumordata	a GxD matrix representing gene expression profiles of tumour samples
model	list containing all the parameters to be optimized
NUM_ITERATIONS_RMINIMIZE	minimum number of iteration that the minimization algorithm runs
iter	the iteration number
NUM_GRID_SEARCH_ITERATIONS	number of times to try restarting with different initial values

Value

The model with the kappa parameter updated

Author(s)

Gerald Quon, Catalina Anghel, Francis Nguyen

`ISOpureS1.model_optimize.opt_mm`
Optimize the reference cancer profile, m, in step 1

Description

The goal of this function is to optimize the reference cancer profile mm. Because mm is constrained (must be parameters of multinomial/discrete distribution), we don't directly optimize the likelihood function w.r.t. mm, but we perform change of variables to do unconstrained optimization. We therefore store these unconstrained variables in the field "mm_weights", and update these variables.

Usage

```
ISOpureS1.model_optimize.opt_mm(
tumordata, model,
NUM_ITERATIONS_RMINIMIZE,
iter,
NUM_GRID_SEARCH_ITERATIONS
)
```

Arguments

`tumordata`	a GxD matrix representing gene expression profiles of tumour samples
`model`	list containing all the parameters to be optimized
`NUM_ITERATIONS_RMINIMIZE`	minimum number of iteration that the minimization algorithm runs
`iter`	the iteration number
`NUM_GRID_SEARCH_ITERATIONS`	number of times to try restarting with different initial values

Value

The model with mm_weights updated (and log_all_rates)

Author(s)

Gerald Quon, Catalina Anghel, Francis Nguyen

`ISOpureS1.model_optimize.opt_omega` *Optimize omega in step 1*

Description

This function optimizes omega, in fact the convex mixing weights that govern prior over the reference cancer profile.

Usage

```
ISOpureS1.model_optimize.opt_omega(
tumordata,
model,
NUM_ITERATIONS_RMINIMIZE,
iter,
NUM_GRID_SEARCH_ITERATIONS
)
```

Arguments

`tumordata`	a GxD matrix representing gene expression profiles of tumour samples
`model`	list containing all the parameters to be optimized
`NUM_ITERATIONS_RMINIMIZE`	minimum number of iteration that the minimization algorithm runs
`iter`	the iteration number
`NUM_GRID_SEARCH_ITERATIONS`	number of times to try restarting with different initial values

Value

The model with the omega_weights and omega parameters updated

Author(s)

Gerald Quon, Catalina Anghel, Francis Nguyen

ISOpureS1.model_optimize.opt_theta
Optimize theta in step 1

Description

This function optimizes theta, in fact theta_weights. Since thetas are constrained (must be parameters of multinomial/discrete distribution), we don't directly optimize the likelihood function w.r.t. theta, but we perform change of variables to do unconstrained optimization. We therefore store these unconstrained variables in the field "theta_weights", and update these variables.

Usage

```
ISOpureS1.model_optimize.opt_theta(
tumordata,
model,
NUM_ITERATIONS_RMINIMIZE,
iter,
NUM_GRID_SEARCH_ITERATIONS
)
```

Arguments

tumordata	a GxD matrix representing gene expression profiles of tumour samples
model	list containing all the parameters to be optimized
NUM_ITERATIONS_RMINIMIZE	minimum number of iteration that the minimization algorithm runs
iter	the iteration number
NUM_GRID_SEARCH_ITERATIONS	number of times to try restarting with different initial values

Value

The model with the theta parameter updated

Author(s)

Gerald Quon, Catalina Anghel, Francis Nguyen

`ISOpureS1.model_optimize.opt_vv` *Optimize vv in step 1*

Description

This function optimizes vv, the strength parameter in the prior over the reference cancer profile. Note that we don't directly optimize vv because it has constraints (must be >=1 to guarantee real-valued likelihoods).

Usage

```
ISOpureS1.model_optimize.opt_vv(
tumordata,
model,
NUM_ITERATIONS_RMINIMIZE,
iter,
NUM_GRID_SEARCH_ITERATIONS
)
```

Arguments

`tumordata`	a GxD matrix representing gene expression profiles of tumour samples
`model`	list containing all the parameters to be optimized
`NUM_ITERATIONS_RMINIMIZE`	minimum number of iteration that the minimization algorithm runs
`iter`	the iteration number
`NUM_GRID_SEARCH_ITERATIONS`	number of times to try restarting with different initial values

Value

The model with the vv parameter updated

Author(s)

Gerald Quon, Catalina Anghel, Francis Nguyen

ISOpureS1.model_optimize.theta.theta_deriv_loglikelihood

Compute the derivative of loglikelihood relevant to theta for step 1

Description

Computes the derivative of the loglikelihood function relevant to optimizing theta, not with respect to theta but with respect to unconstrained variables

Usage

```
ISOpureS1.model_optimize.theta.theta_deriv_loglikelihood(ww, tumordata, dd, model)
```

Arguments

ww	the theta weights corresponding to patient dd, a 1xK matrix
tumordata	a GxD matrix representing gene expression profiles of tumor samples
dd	the patient number
model	list containing all the parameters to be optimized

Value

The negative derivative of the loglikelihood function relevant to optimizing theta, not with respect to theta but with respect to unconstrained variables.

Author(s)

Gerald Quon, Catalina Anghel, Francis Nguyen

ISOpureS1.model_optimize.theta.theta_loglikelihood

Compute the loglikelihood relevant to theta for step 1

Description

Computes the part of the loglikelihood function relevant to optimizing theta for step 1

Usage

```
ISOpureS1.model_optimize.theta.theta_loglikelihood(ww, tumordata, dd, model)
```

Arguments

ww	the theta weights corresponding to patient dd, a 1xK matrix
tumordata	a GxD matrix representing gene expression profiles of tumor samples
dd	the patient number
model	list containing all the parameters to be optimized

Value

The negative of the loglikelihood relevant to theta

Author(s)

Gerald Quon, Catalina Anghel, Francis Nguyen

ISOpureS1.model_optimize.vv.vv_compute_loglikelihood
Compute loglikelihood relevant to vv for step 1

Description

Computes the part of the loglikelihood function relevant to optimizing vv for step 1.

Usage

```
ISOpureS1.model_optimize.vv.vv_compute_loglikelihood(vv, sum_log_theta, DD)
```

Arguments

vv	Kx1 matrix representing the weights of the normal profiles B_i used to make the weighted combination that forms the mean parameter vector for the Dirichlet distribution over m
sum_log_theta	the column sums of log(theta), a 1xK matrix
DD	the number of patients (a scalar)

Value

The negative of the loglikelihood relevant to optimizing vv

Author(s)

Gerald Quon, Catalina Anghel, Francis Nguyen

ISOpureS2.model_core.compute_loglikelihood

Compute loglikelihood given all model parameters for step 2

Description

Computes complete loglikelihood given all model parameters for step 2

Usage

```
ISOpureS2.model_core.compute_loglikelihood(tumordata, model)
```

Arguments

tumordata	a GxD matrix representing gene expression profiles of tumor samples
model	list containing all the parameters updated in ISOpure step two iterations

Value

The scalar value of the complete loglikelihood obtained given the model parameters

Author(s)

Gerald Quon, Catalina Anghel, Francis Nguyen

ISOpureS2.model_core.new_model

Compute loglikelihood given all model parameters for step 2

Description

Produces a list (the model) which initializes the parameters vv, log_BBtranspose, PPtranspose, kappa, theta, omega, log_all_rates for step 2

Usage

```
ISOpureS2.model_core.new_model(tumordata, kappa, INITIAL_VV, PPtranspose, BBtranspose)
```

Arguments

tumordata	a GxD matrix representing gene expression profiles of tumor samples
kappa	a 1xD matrix which represents strength parameter kappa over cc, given the reference profile mm
INITIAL_VV	a vector with K components, the prior over mixing proportions, theta, with last entry weighed more heavily
PPtranspose	the prior on the tumor-specific cancer profiles is just the reference cancer profile (1xG matrix) learned in ISOpureS1, standardized so that all entries sum to 1
BBtranspose	a (K-1)xG matrix of the standardized normal profiles, so that they sum to 1

Value

model	a newly generated model list to hold all the parameters

Author(s)

Gerald Quon, Catalina Anghel, Francis Nguyen

ISOpureS2.model_core.optmodel

Optimizes the ISOpure parameters for step 2

Description

Optimizes the ISOpure parameters for step 2 cyclically until convergence

Usage

```
ISOpureS2.model_core.optmodel(tumordata, model, NUM_ITERATIONS=35)
```

Arguments

tumordata	a GxD matrix representing gene expression profiles of tumor samples
model	list containing all the parameters to be optimized
NUM_ITERATIONS	(optional) minimum number of iterations of optimization algorithm, default is 35

Value

model	updated model list containing all the parameters

Author(s)

Gerald Quon, Catalina Anghel, Francis Nguyen

`ISOpureS2.model_optimize.cc.cc_deriv_loglikelihood`
Compute the derivative of loglikelihood relevant to the patient cancer profiles, cc, for step 2

Description

Computes the derivative of the part of the likelihood function relevant to optimizing cc.

Usage

```
ISOpureS2.model_optimize.cc.cc_deriv_loglikelihood(ww, tumordata, dd, model)
```

Arguments

`ww`	the cc_weights for patient dd, with G entries
`tumordata`	a GxD matrix representing gene expression profiles of tumor samples
`dd`	the patient number
`model`	list containing all the parameters to be optimized

Value

The negative derivative of the loglikelihood function relevant to optimizing cc for patient dd, the cancer profile for that patient. The derivative is taken not with respect to vv but with respect to unconstrained variables via a change of variables

Author(s)

Gerald Quon, Catalina Anghel, Francis Nguyen

`ISOpureS2.model_optimize.cc.cc_loglikelihood`
Compute the loglikelihood relevant to the patient cancer profiles, cc, for step 2

Description

Computes the part of the loglikelihood function relevant to optimizing cc for patient dd, the cancer profile for that patient

Usage

```
ISOpureS2.model_optimize.cc.cc_loglikelihood(ww, tumordata, dd, model)
```

Arguments

ww	the cc_weights for patient dd, with G entries
tumordata	a GxD matrix representing gene expression profiles of tumor samples
dd	the patient number
model	list containing all the parameters to be optimized

Value

The negative the part of the loglikelihood function relevant to optimizing cc for patient dd, the cancer profile for that patient.

Author(s)

Gerald Quon, Catalina Anghel, Francis Nguyen

ISOpureS2.model_optimize.kappa.kappa_compute_loglikelihood

Compute loglikelihood relevant to kappa for step 2

Description

Computes the part of the loglikelihood function relevant to optimizing kappa for step 2

Usage

```
ISOpureS2.model_optimize.kappa.kappa_compute_loglikelihood(kappa, model)
```

Arguments

kappa	a 1xK vector strength parameter in the prior over cc given the cancer profile mm
model	list containing all the parameters to be optimized

Value

The part of the loglikelihood function relevant to optimizing kappa

Author(s)

Gerald Quon, Catalina Anghel, Francis Nguyen

ISOpureS2.model_optimize.kappa.kappa_deriv_loglikelihood

Compute derivative of loglikelihood with respect to kappa for step 2

Description

Computes the derivative of the part of the loglikelihood function relevant to optimizing kappa for step 2. Instead of performing constrained optimization on kappa directly, we optimize the log of kappa in an unconstrained fashion.

Usage

```
ISOpureS2.model_optimize.kappa.kappa_deriv_loglikelihood(log_kappa, model)
```

Arguments

log_kappa	the 1xD matrix log(kappa - model\$MIN_KAPPA)
model	list containing all the parameters to be optimized

Value

The negative derivative of the part of the loglikelihood function relevant to kappa with respect to log kappa (a Dx1 matrix).

Author(s)

Gerald Quon, Catalina Anghel, Francis Nguyen

ISOpureS2.model_optimize.kappa.kappa_loglikelihood

Compute loglikelihood relevant to kappa for step 2

Description

Computes the part of the loglikelihood function relevant to optimizing kappa for step 2. Instead of performing constrained optimization on kappa directly, we optimize the log of kappa in an unconstrained fashion.

Usage

```
ISOpureS2.model_optimize.kappa.kappa_loglikelihood(log_kappa, model)
```

Arguments

log_kappa	the 1xD matrix log(kappa - model\$MIN_KAPPA)
model	list containing all the parameters to be optimized

Value

The negative of the loglikelihood relevant to optimizing kappa

Author(s)

Gerald Quon, Catalina Anghel, Francis Nguyen

`ISOpureS2.model_optimize.opt_cc` *Optimize the tumor-specific cancer profiles in step 2*

Description

Optimize the tumor-specific cancer profiles. Because cc is constrained (each cc_i are parameters of multinomial/discrete distribution), we don't directly optimize the likelihood function w.r.t. cc, but we perform change of variables to do unconstrained optimization. We therefore store these unconstrained variables in the field "cc_weights", and update these variables.

Usage

```
ISOpureS2.model_optimize.opt_cc(
tumordata,
model,
NUM_ITERATIONS_RMINIMIZE,
iter,
NUM_GRID_SEARCH_ITERATIONS)
```

Arguments

`tumordata`	a GxD matrix representing gene expression profiles of tumour samples
`model`	list containing all the parameters to be optimized
`NUM_ITERATIONS_RMINIMIZE`	minimum number of iteration that the minimization algorithm runs
`iter`	the iteration number
`NUM_GRID_SEARCH_ITERATIONS`	number of times to try restarting with different initial values

Value

The model with cc_weights and log_cc updated

Author(s)

Gerald Quon, Catalina Anghel, Francis Nguyen

`ISOpureS2.model_optimize.opt_kappa` *Optimize kappa in step 2*

Description

This function optimizes kappa, the strength parameter in the prior over the reference cancer profile. Note that we don't directly optimize kappa because it has constraints (must be greater than the minimum determined in ISOpure.step2.PPE.)

Usage

```
ISOpureS2.model_optimize.opt_kappa(
tumordata,
model,
NUM_ITERATIONS_RMINIMIZE,
iter,
NUM_GRID_SEARCH_ITERATIONS
)
```

Arguments

`tumordata`	a GxD matrix representing gene expression profiles of tumour samples
`model`	list containing all the parameters to be optimized
`NUM_ITERATIONS_RMINIMIZE`	minimum number of iteration that the minimization algorithm runs
`iter`	the iteration number
`NUM_GRID_SEARCH_ITERATIONS`	number of times to try restarting with different initial values

Value

The model with the kappa parameter (which is a 1xD vector) updated

Author(s)

Gerald Quon, Catalina Anghel, Francis Nguyen

`ISOpureS2.model_optimize.opt_theta` *Optimize theta in step 2*

Description

This function optimizes theta, in fact theta_weights. Since thetas are constrained (must be parameters of multinomial/discrete distribution), we don't directly optimize the likelihood function w.r.t. theta, but we perform change of variables to do unconstrained optimization. We therefore store these unconstrained variables in the field "theta_weights", and update these variables.

Usage

```
ISOpureS2.model_optimize.opt_theta(
tumordata,
model,
NUM_ITERATIONS_RMINIMIZE,
iter,
NUM_GRID_SEARCH_ITERATIONS
)
```

Arguments

`tumordata`	a GxD matrix representing gene expression profiles of tumour samples
`model`	list containing all the parameters to be optimized
`NUM_ITERATIONS_RMINIMIZE`	minimum number of iteration that the minimization algorithm runs
`iter`	the iteration number
`NUM_GRID_SEARCH_ITERATIONS`	number of times to try restarting with different initial values

Value

The model with the theta parameter updated (the first K-1 columns) corresponding to the normal sample contributions

Author(s)

Gerald Quon, Catalina Anghel, Francis Nguyen

ISOpureS2.model_optimize.opt_vv
Optimize vv in step 2

Description

This function optimizes vv, the strength parameter in the prior over the reference cancer profile. Note that we don't directly optimize vv because it has constraints (must be >=1 to guarantee real-valued likelihoods).

Usage

```
ISOpureS2.model_optimize.opt_vv(
tumordata,
model,
NUM_ITERATIONS_RMINIMIZE,
iter,
NUM_GRID_SEARCH_ITERATIONS
)
```

Arguments

tumordata	a GxD matrix representing gene expression profiles of tumour samples
model	list containing all the parameters to be optimized
NUM_ITERATIONS_RMINIMIZE	minimum number of iteration that the minimization algorithm runs
iter	the iteration number
NUM_GRID_SEARCH_ITERATIONS	number of times to try restarting with different initial values

Value

The model with the vv parameter updated

Author(s)

Gerald Quon, Catalina Anghel, Francis Nguyen

ISOpureS2.model_optimize.theta.theta_deriv_loglikelihood

Compute the derivative of loglikelihood relevant to theta for step 2

Description

Computes the derivative of the loglikelihood function relevant to optimizing theta, not with respect to theta but with respect to unconstrained variables

Usage

```
ISOpureS2.model_optimize.theta.theta_deriv_loglikelihood(ww, tumordata, dd, model)
```

Arguments

ww	the theta weights corresponding to patient dd, a 1xK matrix
tumordata	a GxD matrix representing gene expression profiles of tumor samples
dd	the patient number
model	list containing all the parameters to be optimized

Value

The negative derivative of the loglikelihood function relevant to optimizing theta, not with respect to theta but with respect to unconstrained variables.

Author(s)

Gerald Quon, Catalina Anghel, Francis Nguyen

ISOpureS2.model_optimize.theta.theta_loglikelihood

Compute the loglikelihood relevant to theta for step 2

Description

Computes the part of the loglikelihood function relevant to optimizing theta for step 2

Usage

```
ISOpureS2.model_optimize.theta.theta_loglikelihood(ww, tumordata, dd, model)
```

Arguments

ww	the theta weights corresponding to patient dd, a 1xK matrix
tumordata	a GxD matrix representing gene expression profiles of tumor samples
dd	the patient number
model	list containing all the parameters to be optimized

Value

The negative of the loglikelihood relevant to theta

Author(s)

Gerald Quon, Catalina Anghel, Francis Nguyen

ISOpureS2.model_optimize.vv.vv_compute_loglikelihood

Compute loglikelihood relevant to vv for step 2

Description

Computes the part of the loglikelihood function relevant to optimizing vv for step 2.

Usage

```
ISOpureS2.model_optimize.vv.vv_compute_loglikelihood(ww, sum_log_theta, D)
```

Arguments

ww	log(vv-1), a Kx1 matrix
sum_log_theta	the column sums of log(theta), a 1xK matrix
D	the number of patients (a scalar)

Value

The negative of the loglikelihood relevant to optimizing vv

Author(s)

Gerald Quon, Catalina Anghel, Francis Nguyen

Index