

Package ‘ccems’

June 14, 2009

Type Package

Title Combinatorially Complex Equilibrium Model Selection

Version 1.03

Date 2009-06-08

Author Tom Radivoyevitch

Maintainer Tom Radivoyevitch <txr24@case.edu>

Description Dissociation constants of quasi-equilibriums of enzymes and protein-ligand binding equilibriums in general are systematically scanned though possibilities of being infinity and/or equal to others. The automatically generated space of models is then fitted to data.

Depends odesolve,snow,PolynomF

Imports

Suggests nws

Enhances

License GPL-2

LazyLoad yes

LazyData yes

URL <http://epbi-radivot.cwru.edu/ccems>

Repository CRAN

Date/Publication 2009-06-14 19:07:54

R topics documented:

ccems-package	2
ems	6
fitModel	9
mkg	11
mkGrids	14
mkKd2Kj	17
mkModel	19
mkSpurs	23
RNR	25
simulateData	27
TK1	29

Index	33
--------------	-----------

ccems-package	<i>Combinatorially Complex Equilibrium Model Selection</i>
---------------	--

Description

This package performs model selections of equilibriums in general and quasi-equilibriums of enzyme complexes in particular. Estimates of dissociation constants K that best describe a dataset are found by systematically scanning though all possibilities of K being infinity and/or plausibly equal to other K . The automatically generated space of models is then fitted to data. Automation enables searches of spaces too large to be specified by hand, e.g. spaces generated by combinatorially complex equilibriums.

Details

```

Package:    ccems
Type:      Package
Depends:   odesolve, snow, PolynomF
Suggests:  nws
License:   GPL-2
LazyLoad:  yes
LazyData:  yes
URL:      http://epbi-radivot.cwru.edu/ccems

```

Index:

RNR	Ribonucleotide Reductase Data
TK1	Thymidine Kinase 1 Data
ems	Equilibrium Model Selection
fitModel	Fit Model
mkGrids	Make Grid Model Space
mkKd2Kj	Make Kd2Kj Mappings

<code>mkModel</code>	Make Specific Model
<code>mkSpurs</code>	Make Spur Model Space
<code>mkg</code>	Make Generic Model
<code>simulateData</code>	Simulate Data

This package automatically generates and fits biochemical equilibrium models using as outputs either average protein mass data or enzyme reaction rate data. It is currently limited to systems where one central hub protein mediates all of the interactions and total concentrations of the reactants are approximately known exactly, e.g. as in systems that were reconstituted from purified reactants. It is limited further in that multiple sites for the same ligand must be filled in a predetermined sequence.

Equilibriums can be specified by any acyclic spanning subgraph of its nodes, where edges are dissociation constants. Here, hub protein oligomerization is viewed as a curtain rod from which threads of ligand bound states/complexes hang: each notch down a thread corresponds to one additional ligand bound to the hub j -mer where j increases as one moves to the right on the curtain rod. At the top of each thread is a head-node that sits on the rod. The head nodes must be specified, as some j values may be absent and some ligand sites (other than the thread defining site) may be assumed to be saturated in some j -mers. The last node in each thread will be referred to as a tail node. If a ligand has more than one binding site, the tail of the thread of one site (other than the last one filled) is the head of the thread of the site filled next. Thus, head nodes must be stated only for the first site filled.

In the examples below, E is the concentration of thymidine kinase 1 (TK1) tetramers, S is thymidine, t is dTTP, X is ATP and R is the large subunit of ribonucleotide reductase (RNR). The examples are ordered by cpu consumption: the first takes ~ 0.5 min on 1 core, the second ~ 1.5 minutes on 2 cores, and the third ~ 2 days on 16 cores. The first fits activity data to a single thread model. It is the fastest example because it uses rational polynomials for the system model because $[E]$ is small enough that total $[S]$ approximates free $[S]$. In the second example there is only one ligand binding site (the s -site) and the hub protein forms at most a dimer. Thus, the thread topology of the acyclic graph used (to explore K equality hypotheses) has only two head nodes and two threads. The head node of the monomer thread is the free hub protein $R1t0$ and the head node of the dimer thread is the ligand free dimer $R2t0$. As there is only one site, the s -site, there are only two threads, one for the monomer and one for the dimer. Threads contain the names of only their non-head nodes since their heads have already been specified. This structure is assigned to `topology` which is then passed to the function `mkg` to produce a generic model object g . Together with the data, this generic model object is then passed to the function `ems` (equilibrium model selection) which generates the model space, fits it to the data, and returns the `topN` (typically 5, 10 or 20) best (lowest AIC) models. The third example is more complicated than the second because ATP has multiple R1 binding sites and because R also tetramerizes and hexamerizes with increases in $[ATP]$. This problem motivated the development of this R package. It is an example of a problem whose solution is enabled by this software because its model space is too large to specify by hand. A linux cluster is needed to execute this example.

The user must have working directory write privileges so that the subdirectories `models` and `results` can be created to hold model C code (generated by `mkg`) and html output (generated by `ems`), respectively.

Note

This work was supported by the National Cancer Institute (K25CA104791).


```

        m=c("R1t1"),          # monomer      1
        d=c("R2t1","R2t2")  # dimer      2
    )
)

g <- mkg(topology,TCC=TRUE)
data(RNR)
d1 <- subset(RNR, (year==2001) & (fg==1) & (G==0) & (t>0), select=c(R,t,m,year))
d2 <- subset(RNR, year==2006, select=c(R,t,m,year))
dd <- rbind(d1,d2)
names(dd)[1:2] <- paste(strsplit(g$id,split=" ")[[1]],"T",sep="")#e.g. to form "RT"
rownames(dd) <- 1:dim(dd)[1] # lose big number row names of parent dataframe
## top10=ems(dd,g,cpusPerHost=c("localhost "=2),maxTotalPs=2,ptype="SOCK")

## CLUSTER EXAMPLE: This ATP induced R1 hexamerization example runs 1.8 days
##                   on a 16 core (4 quad proc machines) ROCKS Linux cluster.

library(ccems)
topology <- list(
  heads=c("R1X0","R2X2","R4X4","R6X6"),
  sites=list( # s-sites are already filled only in (j>1)-mers
    a=list( #a-site thread
      m=c("R1X1"),          # monomer 1
      d=c("R2X3","R2X4"),  # dimer 2
      t=c("R4X5","R4X6","R4X7","R4X8"), # tetramer 3
      h=c("R6X7","R6X8","R6X9","R6X10","R6X11","R6X12") # hexamer 4
    ), # tails of a-site threads are heads of h-site threads
    h=list( # h-site
      m=c("R1X2"),          # monomer 5
      d=c("R2X5","R2X6"),  # dimer 6
      t=c("R4X9","R4X10","R4X11","R4X12"), # tetramer 7
      h=c("R6X13","R6X14","R6X15","R6X16","R6X17","R6X18") # hexamer 8
    )
  )
)

g=mkg(topology,TCC=TRUE)
dd=subset(RNR, (year==2002) & (fg==1) & (X>0), select=c(R,X,m,year))
names(dd)[1:2] <- paste(strsplit(g$id,split=" ")[[1]],"T",sep="")#i.e. c("RT","XT")

## 29 choose 3(2) is 3654(406), so 3654 + 406 + 29 + 1 = 4090 spurs, but after
## subtracting those without at least one hexamer complex, and after adding
## grids, the total number of models is 3410. Of these 3406 converged, see below.
## Not run:
cpusPerHost=c("localhost" = 4,"compute-0-0 "=4,"compute-0-1 "=4,"compute-0-2 "=4)
top10=ems(dd,g,cpusPerHost=cpusPerHost, maxTotalPs=3, ptype="SOCK",KIC=100)

# The following are the last few lines of the output. The first line shows that a
# one parameter model is best(shown are best AICs of models with 0, 1, 2 or 3
# parameters). The next shows that it took 1.8 days on 16 cpus to fit 3406 models.
# And the block that follows shows that the top 5 models are all spur graph models.
# The html file RXglobSOCK.htm in the results directory contains this information
# and more (e.g. parameter estimates and CI).
```

```

#
# [1] 1000000.00000      -33.16309      -31.73658      -29.99075
#
# Time difference of 2623.881 mins
# Fitted = 3406, out of a total of 3410
#
# ... making HTML file ...
# 1 Model 20; nbp= 1; id=IIIIIIIIIIJIIIIIIIIIIIIIIIIIIII; AIC=-33.1631
# 2 Model 108; nbp= 2; id=IIIIJIIIIJIIIIIIIIIIIIIIIIIIII; AIC=-31.7366
# 3 Model 21; nbp= 1; id=IIIIIIIIIIJIIIIIIIIIIIIIIIIIIII; AIC=-31.5144
# 4 Model 109; nbp= 2; id=IIIIJIIIIJIIIIIIIIIIIIIIIIIIII; AIC=-31.4678
# 5 Model 145; nbp= 2; id=IIIIIIIIJIIIIJIIIIIIIIIIIIIIIIIIII; AIC=-31.4431
## End(Not run)

```

ems

Equilibrium Model Selection

Description

This is the main automation function of this package. It generates a space of combinatorially complex equilibrium models and fits them to data.

Usage

```

ems(d, g, cpusPerHost=c("localhost" = 1), ptype="",
    spurChunkSize=1000, nSpurChunks=1,
    maxTotalPs=5, minTotalPs=NULL, extend2maxP=TRUE,
    smart=FALSE, doTights=FALSE, doGrids=TRUE,
    doSpurs=TRUE, topN=10, showConstr=FALSE,
    atLeastOne=TRUE, atLeastOneOfEach=FALSE,
    KIC=1, kIC=1, fullGrid=FALSE,
    transform=c("boxCox", "relResid", "none", "sqrt", "log"), lam=0.5,
    m1=-90, p=-1, forceM1=FALSE, forceP=FALSE)

```

Arguments

d	The data as a dataframe.
g	The list output of <code>mkg</code> .
cpusPerHost	This is an integer vector where names are host names and values are their cpu numbers.
ptype	Parallelization type: "" for single cpus; "SOCK" and "NWS" (networkspaces) for snow options.
spurChunkSize	The <code>batchSize</code> of spur model chunks, see mkSpurs
nSpurChunks	The number of spur model chunks requested (this may increase internally if <code>extend2maxP = TRUE</code> or <code>smart=TRUE</code>).

maxTotalPs	The maximum number of parameters of models that will be fitted (internally, larger models may be generated but not fitted).
minTotalPs	The minimum number of parameters of models in the model space. If NULL no minimum is imposed.
extend2maxP	This logical is TRUE if nSpurChunks should be extended (if needed) to reach maxTotalPs.
smart	Set to TRUE to stop when models with lastCompleted parameters (see mkSpurs) have an AIC that is bigger than that of the lastCompleted-1 parameter models.
doTights	Set to TRUE if spur models with infinitely tight binding single edges (with K=0) are wanted in the model space.
doGrids	Leave TRUE (the default) if grid models are wanted, set to FALSE if not (e.g. if only spur models are wanted).
doSpurs	Leave TRUE if the spur model space is wanted, set to FALSE if not (e.g. if only grid models are wanted).
topN	The number of best models of the current batch of models that will be carried over to compete with the next batch; such carryovers are needed to allow fits of model spaces that are too large to reside in memory at one time. This number is also the number of best models summarized in html in the results folder after fitting each batch.
showConstr	Set to TRUE if constrained (fixed and tracking) parameters are to be included in the html report in results.
atLeastOne	Leave TRUE if only models with at least one complex of maximal size are to be considered. Set FALSE if there is no prior knowledge supportive of the assertion that the largest oligomer must be in the model.
atLeastOneOfEach	Set TRUE if only models with at least one complex of each oligomer size are to be considered. This is useful when the data are multivariate proportions (i.e. mass distribution data) and each j-mer is clearly present.
KIC	The initial condition of all K parameters optimized. The default is KIC=1 (in uM).
kIC	The initial condition of all k parameters optimized. The default is kIC=1 (in 1/seconds per occupied active site).
fullGrid	Set TRUE if a full binary K model is wanted, else grids that are equivalent to spurs are eliminated from the model space.
transform	If not "none" data and model are transformed before forming residuals. This is used to stabilize enzyme activity variances. Other options are "boxCox" for Box-Cox transformations, in which case lam below is used as lambda, "relResid" to divide the residuals by the data, and square root and natural log transformations using "sqrt" and "log", respectively.
lam	The lambda parameter of the Box-Cox transformation, if used.
m1	The hub protein's monomer mass in kDa. The default is 90 for the big (R1) subunit of ribonucleotide reductase (RNR). This only matters if the data is mass data. Negative numbers imply fixed values and positive numbers imply starting values to be fitted to the data.

<code>p</code>	Probability that hub can oligomerize, i.e. is not damaged. Set to a positive value if additional rows are to be added to the output dataframe to include models with <code>p</code> freely estimated. Set negative to hold fixed. Value is the initial or fixed value.
<code>forceM1</code>	Set <code>TRUE</code> to force all models to estimate M1, i.e. to not generate models with M1 fixed.
<code>forceP</code>	Set <code>TRUE</code> to force all models to estimate <code>p</code> , i.e. to not generate models with fixed <code>p</code> .

Details

This is the highest level function in `ccems`. The other functions serve this function, though they may also be used to fit individual models manually.

Value

A list of the `topN` best (lowest AIC) models. This should be assigned to a variable to avoid large screen dumps. An html report, the `topN` fitted models, and a brief summary of all fitted models, are saved to `results` and are the main outputs and use of this function.

Note

Spur and grid graph models have network topologies that either radiate from the hub or can be overlaid on a city block lay out, respectively. Though head node spur graph edges can be superimposed in curtain rods (see `ccems`) to give these graphs a grid appearance, curtain rods are really sets of nested arches. Thus curtains could be called spur-grid hybrid K equality graphs or simply hybrids (i.e. a term that is more tolerant than grid). Another option is to tolerate spur edges to head nodes in a broadened definition of the term grid. Advantages include an emphasis on parallel edges and thus equality aspects of the graph (compared to the term hybrid), more compactness (compared to the term K equality) and usage inertia. Readers are thus asked to accept this broadened definition of the term grid, i.e. to allow head node spur edges in grid graphs.

This work was supported by the National Cancer Institute (K25CA104791).

Author(s)

Tom Radivoyevitch (txr24@case.edu)

References

Radivoyevitch, T. (2009) Automated model generation and analysis methods for combinatorially complex biochemical equilibriums. (In preparation)

See Also

[ccems](#), [mkg](#)

Examples

```

library(ccems)
topology <- list(
  heads=c("R1t0", "R2t0"),
  sites=list(
    s=list(
      m=c("R1t1"),          # s-site   thread #
                        # monomer     1
      d=c("R2t1", "R2t2") # dimer    2
    )
  )
)
g <- mkg(topology)
data(RNR)
d1 <- subset(RNR, (year==2001) & (fg==1) & (G==0) & (t>0), select=c(R,t,m,year))
d2 <- subset(RNR, year==2006, select=c(R,t,m,year))
dd <- rbind(d1,d2)
names(dd)[1:2] <- c("RT", "tT")
rownames(dd) <- 1:dim(dd)[1] # lose big number row names of parent dataframe
# the call above ends sooner if maxTotalPs is reached
## Not run:

top <- ems(dd,g,maxTotalPs=1) # this takes roughly one minute
## End(Not run)

```

fitModel

Fit Model

Description

This function fits a model/hypothesis created by `mkModel`. It is typically passed to `lapply` or `clusterApplyLB` to fit a list of model objects, typically within `ems`.

Usage

```
fitModel(model)
```

Arguments

`model` The output list of `mkModel`.

Details

The main output of this function is the `report` component of its value (see below) which is also echoed to the screen during computations.

Value

The input argument `model` extended to include the following fields:

<code>echk</code>	A matrix that checks the TCC solver and <code>model\$fback</code> . Matrix column names that end in Q should match their sans-Q counterparts.
<code>eSS</code>	The expected steady state concentrations of complexes and free reactants. For each row of the data dataframe there is a row in this matrix. Its contents are the TCC solver's expected free reactant concentrations and the result of applying <code>model\$fback</code> to them to create expected complex concentrations.
<code>res</code>	The residuals of the fit.
<code>nData</code>	The number of data points/rows in the data dataframe <code>model\$d</code> .
<code>SSE</code>	The initial and final sum of squared errors (i.e. residual sum of squares).
<code>AIC</code>	The initial and final Akaike Information Criterion values, corrected for small samples. Since nonlinear least squares is used $AIC = N \cdot \log(SSE/N) + 2 \cdot P + 2 \cdot P \cdot (P+1) / (N-P-1) + N \cdot \log(2 \cdot \pi) + N$ where $N = nData$ and P is the number of estimated parameters (including the variance).
<code>nOptParams</code>	The number of optimized parameters, i.e. the length of the parameter vector sent to <code>optim</code> .
<code>hess</code>	This is TRUE if the determinant of the Hessian of the log-likelihood evaluated at the optimum is greater than zero, i.e. if the hessian can be inverted to create a parameter estimate covariance matrix.
<code>CI</code>	Confidence intervals. Unlike those in <code>model\$report</code> these are numeric rather than strings and these are not expressed as concentrations raised to integer powers (in cases of complete dissociation constants).
<code>cpu</code>	The amount of computing time (in minutes) taken to fit the model.
<code>report</code>	An extension of <code>model\$params</code> to include parameter point estimates and confidence intervals (see CI above). The <code>final</code> column holds numerics and the <code>pointEstimate</code> column holds strings of the same numbers expressed as powers in cases of complete dissociation constants.

Note

This work was supported by the National Cancer Institute (K25CA104791).

Author(s)

Tom Radivoyevitch (txr24@case.edu)

References

Radivoyevitch, T. (2008) Equilibrium model selection: dTTP induced R1 dimerization. *BMC Systems Biology* **2**, 15.

See Also

[mkModel,ems,ccems](#)

Examples

```

library(ccems)
topology <- list(
  heads=c("R1t0", "R2t0"),
  sites=list(
    s=list(
      m=c("R1t1"),      # s-site   thread #
                        # monomer    1
      d=c("R2t1", "R2t2") # dimer     2
    )
  )
)
g <- mkg(topology, TCC=TRUE)
data(RNR)
d1 <- subset(RNR, (year==2001) & (fg==1) & (G==0) & (t>0), select=c(R,t,m,year))
d2 <- subset(RNR, year==2006, select=c(R,t,m,year))
dRt <- rbind(d1, d2)
names(dRt)[1:2] <- paste(strsplit(g$id, split=" ")[[1]], "T", sep="") #e.g. to form "RT"
rownames(dRt) <- 1:dim(dRt)[1] # lose big number row names of parent dataframe

## Not run:
models <- list(
  mkModel(g, "IIJJ", dRt, Kjparams=c(R2t0=Inf, R1t1=Inf, R2t1=1, R2t2=1)),
  mkModel(g, "IIIJ", dRt, Kjparams=c(R2t0=Inf, R1t1=Inf, R2t1=Inf, R2t2=1))
)
# the next line fits the list of two models above in series on a single processor
fmodels <- lapply(models, fitModel)
## End(Not run)
# Note that fitModel always delivers a summary of the fit to the screen as a byproduct.
# The output of the call is assigned to avoid scrolling up through the returned large
# fitted list of models in order to find this summary.

```

mkg

Make Generic Model

Description

This function maps network topology information into a generic full spur graph model. If TCC is TRUE (default) it also automatically generates and compiles total concentration constraint C code and stores it in the `models` subdirectory.

Usage

```
mkg(strct, TCC=TRUE, activity=FALSE, free=FALSE)
```

Arguments

`strct` The thread topology of the equilibrium network (see [ccems](#)). The slots of this list structure are: `heads` which is a character vector of the head nodes of threads of the first site in `sites`; and `sites` which is a list of sites where each site is

a list of character vectors of non-head thread nodes. Naming conventions are given below under `Z`.

<code>TCC</code>	This is <code>TRUE</code> if total concentration constraints (TCCs) are to be used. If so, model C code is automatically generated and compiled. If <code>TCC</code> is <code>FALSE</code> a rational polynomial model is automatically generated as an R function component of the output. Use of rational polynomial models assumes that the approximations $[X_{\text{free}}] = [X_{\text{T}}]$ are reasonable for all non-hub species X (i.e. ligands).
<code>activity</code>	If <code>TRUE</code> data fitted is enzyme activity data. In this case the substrate variable should be named S.
<code>free</code>	Set <code>TRUE</code> if non-hub reactants are to be treated as free concentrations.

Details

This function is typically the first `ccems` function called in a script. It creates a list of objects generic to the entire model space, i.e. a model space kernel.

Value

A list comprised of the inputs and the following:

<code>id</code>	The biochemical equilibrium system ID. This is the set of single character reactant representations collapsed into one string.
<code>hubChar</code>	The first character of the first name in <code>struct</code> . As of ver 1.02 this is no longer a passed option.
<code>Z</code>	The names of the hub protein complexes expressed as single character reactant symbols followed by the number of copies of it in the complex. For example, <code>R2t1</code> is a dimer of R with one bound ligand t, <code>R2t2</code> is saturated dimer, and <code>R2t0</code> is ligand free dimer. The choice of the symbol <code>Z</code> for hub complexes derives from its common use as a complex number.
<code>nZ</code>	The length of <code>Z</code> . An <code>n</code> in front of a name often implies a length.
<code>atomS</code>	A character vector of atom/reactant names where the term atom refers to reactant molecules being indivisible in the system of interest. A capital S at the end of a name implies a string.
<code>nAtomS</code>	The length of <code>atomS</code> .
<code>species</code>	A character vector equal to <code>c(atomS, Z)</code> , i.e. the names of all of the chemical species.
<code>nSpecies</code>	The length of <code>species</code> .
<code>reactantS</code>	A list with component names equal to complex names and values equal to vectors of the <code>atomS</code> that comprise them.
<code>W</code>	A dataframe of copy numbers/weights of atoms (column names) in each species (row names).
<code>KdS</code>	Subscript names of the dissociation constants, one for each element of <code>Z</code> . Binary K sit in positions of their products and use <code>"_"</code> to separate their reactants; spur edges (e.g. head nodes) take their names from <code>Z</code> .
<code>hdS</code>	The names of the head nodes.

<code>hds</code>	The positions of the head nodes within the Z vector of complex nodes.
<code>sstime</code>	The amount of ODE integration time used to reach steady state when solving TCCs (preset to 1e6). Since integration uses variable step sizes which rapidly become large near steady state, overkill is OK.
<code>rtol</code>	An <code>lsoda</code> integration relative error tolerance parameter: <code>rtol=1e-5</code> .
<code>atol</code>	An <code>lsoda</code> integration absolute error tolerance parameter: <code>atol=1e-7</code> .
<code>parmsTCC</code>	The parameters of the total concentration constraints passed to the <code>.dll</code> (or <code>.so</code>) when using <code>lsoda</code> . These include the total concentrations (i.e. system inputs).
<code>initialStateTCC</code>	The initial conditions (default zero) used to solve the TCC ODEs.
<code>dfThread</code>	A dataframe of the thread within site and oligo structure.
<code>threads</code>	A nested list of the thread structure with threads at the top of the list and their contents and memberships below.
<code>threadsWithinSites</code>	A list of threads within sites, i.e. the list indices are sites.
<code>nodesWithinSites</code>	A list of non-head nodes within sites.
<code>usedLets</code>	A vector of the single characters used to label Kd equivalent threads. Currently, if there is more than one thread, entire threads are either completely equal or all free to be different. If the curtain contains a single thread, then that thread can have patches of contiguous blocks that are equal. Thus, a curtain with n threads has n used letters, but a curtain with one thread has as many used letters as there are nodes in that one thread.
<code>fback</code>	A function that maps a vector of free concentrations into a vector of complex concentrations given the current set of complete dissociation constant estimates of the optimization algorithm.
<code>code</code>	If <code>TCC = TRUE</code> this is the C code of the TCC ODE right hand side. This code is automatically compiled for use by <code>lsoda</code> .

The TCC flag, which passes unchanged from input to output, is used by `simulateData` to determine how the expected response surface is to be generated.

Note

Kd and Kj are generic grid and spur edge names, respectively. The name `KdS` above is thus a grid-like name. That it is used to describe hybrid models that include head node spur edges is consistent with their allowance in a generalized definition of grid graphs.

In the description of `KdS` above "reactants" is used generically to mean a complex or a ligand, but by far, it refers most often to purified substances of known controlled amounts (i.e. the experimentally manipulated variables or the reactants that are initially present). This common usage is synonymous with the use of "atoms" above, which emphasizes their indivisible building block nature in the non-covalent binding equilibriums of interest.

This work was supported by the National Cancer Institute (K25CA104791).

Author(s)

Tom Radivoyevitch

References

Radivoyevitch, T. (2008) Equilibrium model selection: dTTP induced R1 dimerization. *BMC Systems Biology* **2**, 15.

See Also[ccems,ems](#)**Examples**

```
library(ccems)
topology <- list(
  heads=c("R1t0", "R2t0"),
  sites=list(
    s=list(
      m=c("R1t1"), # s-site monomer 1
      d=c("R2t1", "R2t2") # dimer 2
    )
  )
)
mkg(topology)
```

mkGrids

*Make Grid Model Space***Description**

This function takes `g` from `mkg` and maps it into a dataframe of grid model definitions. It also augments and returns the input list `g`.

Usage

```
mkGrids (g,maxTotalPs=NULL,minTotalPs=NULL,contig=TRUE,
         atLeastOne=TRUE,atLeastOneOfEach=FALSE,
         KIC=1,kIC=1,fullGrid=FALSE,
         m1=-90,p=-1,forceM1=FALSE,forceP=FALSE)
```

Arguments

<code>g</code>	The output of <code>mkg</code> .
<code>maxTotalPs</code>	The maximum number of parameters of models in the model space. If <code>NULL</code> all models are found (but see <code>fullGrid</code> below).
<code>minTotalPs</code>	The minimum number of parameters of models in the model space. If <code>NULL</code> no minimum is imposed.

<code>contig</code>	Set to <code>TRUE</code> to allow Kd equalities of threads only in contiguous runs. Tentatively, this should always be <code>TRUE</code> .
<code>atLeastOne</code>	Leave <code>TRUE</code> if only models with at least one complex of maximal size are to be considered. Set <code>FALSE</code> if there is no prior knowledge supportive of the assertion that the largest oligomer must be in the model.
<code>atLeastOneOfEach</code>	Set <code>TRUE</code> if only models with at least one complex of each oligomer size are to be considered. This is useful when the data are multivariate proportions (i.e. mass distribution data) and each j-mer is clearly present.
<code>KIC</code>	The initial condition of all K parameters optimized. The default is <code>IC=1</code> (in <code>uM</code>).
<code>kIC</code>	The initial condition of all k parameters optimized. The default is <code>kIC=1</code> (in <code>1/seconds per occupied active site</code>).
<code>fullGrid</code>	Set <code>TRUE</code> if a full binary K model is wanted, else grids that are equivalent to spurs are eliminated from the model space. This is used to obtain grid reparameterizations of full spur models to form model averages across common binary K parameters.
<code>m1</code>	The hub protein's monomer mass in <code>kDa</code> . The default is 90 for the big (R1) subunit of ribonucleotide reductase (RNR). This only matters if the data is mass data. Negative numbers imply fixed values and positive numbers imply starting values to be fitted to the data.
<code>p</code>	Probability that hub can oligomerize, i.e. is not damaged. Set to a positive value if additional rows are to be added to the output dataframe to include models with <code>p</code> freely estimated. Set negative to hold fixed. Value is the initial or fixed value.
<code>forceM1</code>	Set <code>TRUE</code> to force all models to estimate M1, i.e. to not generate models with M1 fixed.
<code>forceP</code>	Set <code>TRUE</code> to force all models to estimate p, i.e. to not generate models with fixed p.

Details

In a run of equal threads, the first head node of the run is the leader (optimized parameter) and the remaining nodes in the run are followers (i.e. constrained to track the leader in parameter estimate optimizations); a default in `ems` is that only leader estimates are reported in `html` in the `results` subdirectory. In contrast to its counterpart `mkSpurs`, a means of traversing the grid model space incrementally with increasing numbers of parameters remains to be found and implemented, i.e. `mkGrids` does not have `state` inputs and outputs and the whole space is found in one batch. When this drawback limits research due to too much memory usage, attempts will be made to identify an approach that, similar to what has been implemented for `mkSpurs`, specifies chunks of grid models, fits them, and then specifies the next chunk based on knowledge of where the previous chunk stopped. Note that setting `maxTotalPs` to smaller values will not help this foreseen memory problem as the entire grid model space dataframe is first generated and then, only later, truncated to `maxTotalPs`.

Value

A list with components

chunk	The entire K equality model space requested by the arguments. In this dataframe each row specifies a model. If the <code>activity</code> field of <code>g</code> is <code>TRUE</code> this dataframe includes <code>k</code> columns. The row names encode the equality constraints. Therein <code>.</code> separates K models from <code>k</code> models, <code>I</code> stands for infinity, <code>J</code> stands for freely estimated (in spur components), and other letters are the same when parameters that correspond to their positions equal each other. Unmatched other letters are freely estimated and thus just like <code>J</code> 's but in <code>k</code> and binary <code>K</code> components of the model names.
Keqs	A list of K equality constraints indexed by model names where each element is a vector of character strings whose names are followers and values are leaders.
keqs	Similar to <code>Keqs</code> but for activity constraints. This is <code>NULL</code> if <code>activity</code> in <code>g</code> is <code>FALSE</code> .

Note

This work was supported by the National Cancer Institute (K25CA104791).

Author(s)

Tom Radivoyevitch (txr24@case.edu)

References

Radivoyevitch, T. (2008) Equilibrium model selection: dTTP induced R1 dimerization. *BMC Systems Biology* **2**, 15.

See Also

[mkSpurs](#), [ccems](#)

Examples

```
library(ccems)
topology <- list(
  heads=c("R1t0", "R2t0"),
  sites=list(
    s=list(
      m=c("R1t1"), # s-site   thread #
      d=c("R2t1", "R2t2") # monomer   1
                        # dimer     2
    )
  )
)
g <- mkg(topology)
gridL <- mkGrids(g)
print(gridL$chunk)
topology <- list(
  heads=c("R1X0", "R2X2", "R4X4", "R6X6"),
  sites=list( # s-sites are already filled only in (j>1)-mers
    a=list( #a-site                                     thread
      m=c("R1X1"), # monomer   1
      d=c("R2X3", "R2X4"), # dimer     2
    )
  )
)
```

```

        t=c("R4X5", "R4X6", "R4X7", "R4X8"),           # tetramer  3
        h=c("R6X7", "R6X8", "R6X9", "R6X10", "R6X11", "R6X12") # hexamer   4
    ), # tails of a-site threads are heads of h-site threads
    h=list( # h-site
        m=c("R1X2"), # monomer  5
        d=c("R2X5", "R2X6"), # dimer  6
        t=c("R4X9", "R4X10", "R4X11", "R4X12"), # tetramer  7
        h=c("R6X13", "R6X14", "R6X15", "R6X16", "R6X17", "R6X18") # hexamer  8
    )
)
)

g <- mkg(topology)
gridL <- mkGrids(g, maxTotalPs=2)
print(gridL$chunk)

gridL <- mkGrids(g, maxTotalPs=4)
# the next line should be run separately since its output is large.
# print(gridL$chunk)

```

mkKd2Kj

*Make Kd2Kj Mappings***Description**

This function takes the `g` output of `mkg` and converts it into a list of functions that maps Kd (grid edge) values into Kj (spur edge) values. Such functions are needed so that one generic full spur graph model can be used by all of the specific hypotheses of the model space.

Usage

```
mkKd2Kj(g)
```

Arguments

`g` The output `g` of `mkg`.

Details

Suppose a hub protein ligand has two binding sites (call them a- and h-sites) that are filled in (a,h) order. The names of the Kd2Kj functions produced by calls to this function are then strings of 0's and 1's where a '1' implies that the a-site thread is infinite while its corresponding (within the same j-mer) h-site thread is finite and a '0' implies that the a-site thread is finite. The relevance of this is that a '1' implies that a finite tail Kj parameter must be inserted into an infinite a-site thread to serve as the head of a downstream finite h-site thread; a '0' implies that no action need be taken. Thus, for monomers, dimers, tetramers and hexamers, '0000' is used when no bridge spur edge insertions are needed and '1111' is used when all h-site threads are finite and all a-site threads are infinite.

More sites can also be handled. Suppose we have three ordered (s, a, h) across j-mers (m, d, t, h). The digits then follow binary codings of the s- and a-site threads being infinity: '0' is neither (and

thus no insertions), '1' is a finite a-site thread, '2' is an infinite s-site thread, and '3' (binary 11) is that both are infinite (this case is similar to '1' in that only one a-site insertion is needed). If there were 4 binding sites the digits would range from '0' to '7' (binary 111).

Value

A list of functions that map vectors of Kd's into vectors of Kj's where, within stoichiometric integer factors in the numerators and denominators, an output Kj value is the product of the input Kd's that sit between the output Kj node and the root node (free hub). Here Kd's whose products are not heads are binary and those which are heads are already spur edges, so they pass through an identity mapping. Thus, the Kd inputs are of hybrid/generalized grid graphs (i.e. hanging thread network topologies) and the Kj outputs are of pure spur full model graphs (i.e. the generic model output of `mkG` that is used by the entire model space).

Note

Two functions with two different names can be identical. This happens when the monomer a-site thread is the same whether it is finite as a finite thread or finite because it was asked to support the subsequent monomer (e.g. h-site) thread; in longer threads these two scenarios yield different models: one where the thread nodes are all finite and one where only the tail node is finite while the rest are infinite.

This work was supported by the National Cancer Institute (K25CA104791).

Author(s)

Tom Radivoyevitch (txr24@case.edu)

References

Radivoyevitch, T. (2008) Equilibrium model selection: dTTP induced R1 dimerization. *BMC Systems Biology* **2**, 15.

See Also

[mkGrids](#), [ccems](#)

Examples

```
library(ccems)
topology <- list(
  heads=c("R1t0", "R2t0"),
  sites=list(
    s=list(
      m=c("R1t1"), # s-site monomer 1
      d=c("R2t1", "R2t2") # dimer 2
    )
  )
)
g <- mkg(topology, TCC=TRUE) # generic model
mkKd2Kj(g) # when there is only one function its name is "default"
```

```

topology <- list(
  heads=c("R1X0","R2X2","R4X4","R6X6"),
  sites=list(
    # s-sites are already filled only in (j>1)-mers
    a=list( #a-site
      m=c("R1X1"),
      d=c("R2X3","R2X4"),
      t=c("R4X5","R4X6","R4X7","R4X8"),
      h=c("R6X7","R6X8","R6X9","R6X10", "R6X11", "R6X12")
    ), # tails of a-site threads are heads of h-site threads
    h=list( # h-site
      m=c("R1X2"),
      d=c("R2X5", "R2X6"),
      t=c("R4X9", "R4X10","R4X11", "R4X12"),
      h=c("R6X13", "R6X14", "R6X15","R6X16", "R6X17", "R6X18")# hexamer
    )
  )
)

g <- mkg(topology)
mkKd2Kj(g)

## Not run:
topology <- list(
  heads=c("R1X0","R2X0","R4X0","R6X0"),
  # no requirement that s-sites are filled in oligomers
  sites=list(
    s=list(
      # s-site
      m=c("R1X1"),
      d=c("R2X1","R2X2"),
      t=c("R4X1","R4X2","R4X3","R4X4"),
      h=c("R6X1","R6X2","R6X3","R6X4", "R6X5", "R6X6")
    ),
    a=list(
      # a-site
      m=c("R1X2"),
      d=c("R2X3","R2X4"),
      t=c("R4X5","R4X6","R4X7","R4X8"),
      h=c("R6X7","R6X8","R6X9","R6X10", "R6X11", "R6X12")
    ), # tails of a-site threads are heads of h-site threads
    h=list( # h-site
      m=c("R1X3"),
      d=c("R2X5", "R2X6"),
      t=c("R4X9", "R4X10","R4X11", "R4X12"),
      h=c("R6X13", "R6X14", "R6X15","R6X16", "R6X17", "R6X18")# hexamer
    )
  )
)

g <- mkg(topology,TCC=TRUE)
#gridL <- mkGrids(g,maxTotalPs=3)
mkKd2Kj(g) # this does not take much time to run but it will fill
# the buffer with many function definitions.
## End(Not run)

```

mkModel

*Make Specific Model***Description**

This function takes a generic model input list g and augments it to include data and a specific hypothesis. The hypotheses come in the form of claims that certain K_j are so large that the data cannot discriminate them from being infinity, that certain K_d are so close in value that the data cannot distinguish them from being equal, and that the protein proportion that is active is so close to 1 that the data cannot discriminate it from being 1. Hypotheses regarding enzyme activity parameters k being essentially equal to others on an average per occupied catalytic site basis can also be made.

Usage

```
mkModel(g, mid, d=NULL, Kjparams=NULL, Kdparams=NULL, Keq=NULL,
        Kd2KjLst=NULL, pparams=c(p=-1, m1=-90), kparams=NULL,
        keq=NULL, tightLogic=TRUE,
        transform=c("boxCox", "relResid", "none", "sqrt", "log"), lam=0.5,
        indx=NULL, nParams=NULL)
```

Arguments

g	The output of <code>mkg</code> .
mid	The name of the specific hypothesis/model. The convention is that I stands for infinity, J for a freely estimated spur graph edge, H for a freely estimated grid head node, and D, E, F, etc. (i.e. other characters) for grid K_d that are equal to each other. In the latter case the same letter in two different positions indicates equality between them; positions in model name strings are the binary reaction product node positions in $g\$Z$, see <code>mkg</code>). When k constraints exist they preceded by the K name (a period separates them) and they follow single thread certain conventions (k constraints for larger curtains remain to be worked out). The model name, which becomes $g\$mid$, should not be confused with the name of the biochemical system $g\$id$.
d	The data as a dataframe.
$Kjparams$	If the hypothesis is a spur model, this is a numeric vector of its initial complete dissociation constant parameter values.
$Kdparams$	If the hypothesis is a generalized grid model these are the initial dissociation constant parameter values: head node spur edges are distinguished from thread edges by not having "_" in their names.
Keq	This character vector specifies which K_d parameters are equal to each other. Names are followers and values leaders in the sense of parameters constrained to track each other.
$Kd2KjLst$	This is a list of functions (see <code>mkKd2Kj</code>) that maps generalized grid K_d parameters into full spur model K_j parameters. An appropriate component of this list is assigned to $g\$Kd2Kj$. Such functions are needed so that one generic full

spur graph model, typically compiled in C, can be used by all of the specific hypotheses/models of the model space.

<code>pparams</code>	The <code>p</code> component of this vector is the fraction of hub protein that is active. The negative default of <code>-1</code> keeps it fixed at 1. An initial value of <code>+1</code> indicates that it is to be optimized. The <code>m1</code> component is monomer mass in kDa. The default is 90 for the big (R1) subunit of ribonucleotide reductase (RNR). It is negative to stay fixed and positive to be the starting values in the parameter optimization.
<code>kparams</code>	These are the enzyme activity parameter initial values, if the data is reaction rate data, else it should be <code>NULL</code> .
<code>keq</code>	These are the equality constraints (if any) that are being placed on the enzyme activity parameters.
<code>tightLogic</code>	If <code>tightLogic</code> is true, instead of taking <code>Kj</code> to <code>.001</code> as an approximation of infinitely tight binding, logic is used to model <code>Kj = 0</code> exactly.
<code>transform</code>	If not "none" data and model are transformed before forming residuals. This is used to stabilize enzyme activity variances. Other options are "boxCox" for Box-Cox transformations, in which case <code>lam</code> below is used as lambda, "relResid" to divide the residuals by the data, and square root and natural log transformations using "sqrt" and "log", respectively.
<code>lam</code>	The lambda parameter of the Box-Cox transformation, if used.
<code>indx</code>	This is an integer index of the model. The current hypothesis is the <code>indx</code> th model of the model space.
<code>nParams</code>	The number of model parameters, i.e. the first column of a chunk data frame.

Details

Infinite initial `Kj` parameters remain fixed at `Inf` and are passed from R to C properly to eliminate corresponding polynomial terms in the total concentration constraints.

Value

The input object `g` augmented to include the arguments `d`, `mid` and `indx` and the following:

<code>params</code>	A dataframe specification of the parameter's initial values and whether they are optimized or fixed or constrained to track others. Final value placeholders are initialized to initial values.
<code>Kparams</code>	This is either <code>Kjparams</code> or <code>Kdparams</code> . It is the one of the two which is not <code>NULL</code> . In cases of hybrids <code>Kdparams</code> is used.
<code>codeS</code>	This is the name of the component of <code>Kd2KjLst</code> that is relevant to the current hypothesis. It is a string of digits whose binary representation indicates which threads are infinite.
<code>Kd2Kj</code>	This is the component of <code>Kd2KjLst</code> that is relevant to the current hypothesis, i.e. <code>Kd2Kj = Kd2KjLst[[codeS]]</code> .
<code>fits</code>	A string indicator of the status of the model fitting. It is initialized here to "not fitted yet".
<code>typeY</code>	The type of output. This is <code>m</code> for average mass and <code>v</code> for reaction velocity.

posY This is the column number of the output measurement in the data dataframe d.
 posReactantsD

These are the column numbers of the total reactant concentrations (system inputs) in the data dataframe d.

The value returned by this function is a model object that is ready to be fitted by `fitModel`.

Note

E-shaped topologies found in the BMC SB 2008 reference are not supported in `ccems`. This work was supported by the National Cancer Institute (K25CA104791).

Author(s)

Tom Radivoyevitch (txr24@case.edu)

References

Radivoyevitch, T. (2008) Equilibrium model selection: dTTP induced R1 dimerization. *BMC Systems Biology* **2**, 15.

See Also

[mkg](#), [mkKd2Kj](#) and [fitModel](#).

Examples

```
library(ccems)
topology <- list(
  heads=c("R1t0", "R2t0"),
  sites=list(
    s=list(
      m=c("R1t1"), # s-site monomer 1
      d=c("R2t1", "R2t2") # dimer 2
    )
  )
)
g <- mkg(topology)

data(RNR)
dRt <- subset(RNR, (year==2001) & (fg==1) & (G==0) & (t>0), select=c(R,t,m,year))
names(dRt)[1:2] <- c("RT", "tT")

## first a simple spur model
mkModel(g, "IIIJ", dRt, Kjparams=c(R2t0=Inf, R1t1=Inf, R2t1=Inf, R2t2=1))
Kmapping=mkKd2Kj(g)
mdl=mkModel(g, "HDFE", dRt, Kdparams=c(R2t0=1, R0t0_t=1, R2t0_t=1, R2t1_t=1),
  Keq=c(R2t1_t="R2t0_t"), Kd2KjLst=Kmapping)
fmdl <- fitModel(mdl)

## or mkGrids can be used to achieve the same thing as follows
gridL <- mkGrids(g, maxTotalPs=3)
```

```

chunk <- gridL$chunk
Keqs <- gridL$Keqs
mdl <- mkModel(g, "HDFP", dRt, Kdparams=chunk["HDFP", 2:(g$nZ+1)],
              Keq=Keqs[["HDFP"]], Kd2KjLst=Kmapping,
              pparams=chunk["HDFP", "p", drop=FALSE], indx=chunk["HDFP", "indx"])
print(mdl)
print(chunk)
print(Keqs)

```

mkSpurs

*Make Spur Model Space***Description**

This function takes `g` from `mkg` and maps it into a dataframe of spur graph model definitions.

Usage

```

mkSpurs(g, state=list(globMdlIndex=0, globCmbIndex=0, relCmbIndex=0,
                     config=NULL, maxnKjPs=NULL, maxTotalPs=NULL, minTotalPs=NULL,
                     batchSize=500, doTights=FALSE,
                     atLeastOne=TRUE, atLeastOneOfEach=FALSE,
                     KIC=1, kIC=1, m1=-90, p=-1, forceM1=FALSE, forceP=FALSE)

```

Arguments

<code>g</code>	The generic model output list of <code>mkg</code> .
<code>state</code>	The current state of model space generation. This is a list with several components. <code>globMdlIndex</code> is the global model space index. When <code>mkSpurs</code> is called within <code>ems</code> the <code>globMdlIndex</code> of the spurs begins where it ends for grids. <code>globCmbIndex</code> is the global spur model index. This index is used to trim the last <code>batchSize</code> , if needed, to avoid attempts to extend the spur space size beyond its upper limit of $2^{g\$nZ}$. <code>relCmbIndex</code> is the most critical component of the state. This is the column number of the current matrix output of <code>combn</code> , i.e. it is a relative index. The spur space chunk returned by <code>mkSpurs</code> begins just after this column. <code>config</code> is a vector of the integer positions in <code>g\$Z</code> of the last model's finite <code>K</code> (i.e. the last model of the previous chunk). <code>config</code> is the <code>relCmbIndex</code> th column of the current <code>combn</code> matrix. Its length is the current number of <code>K</code> parameters in the model unless <code>relCmbIndex=0</code> , in which case the number of parameters is one more than this (in this case the end of the last batch coincides with the end of a <code>combn</code> matrix).
<code>maxnKjPs</code>	The maximum number of <code>Kj</code> parameters of models in the model space. Full chunks are created and then trimmed, so decreases in the value of this option will not solve "out of memory" problems.
<code>maxTotalPs</code>	This is the maximum number of freely estimated <code>k</code> or <code>K</code> parameters.

<code>minTotalPs</code>	The minimum number of parameters of models in the model space. If <code>NULL</code> no minimum is imposed.
<code>batchSize</code>	This is the number of <code>K</code> infinity models fitted per batch. Chunk sizes are bigger than this if <code>doTights</code> is <code>TRUE</code> and/or if <code>pRows</code> is <code>TRUE</code> , and/or if activity parameter constraints split models further. Values less than ~1000 are recommended for quad core 8 GB motherboards.
<code>doTights</code>	This should be <code>TRUE</code> if infinitely tight binding models of single edge spur graphs are to be created.
<code>atLeastOne</code>	Leave <code>TRUE</code> if only models with at least one complex of maximal size are to be considered. Set <code>FALSE</code> if there is no prior knowledge supportive of the assertion that the largest oligomer must be in the model.
<code>atLeastOneOfEach</code>	Set <code>TRUE</code> if only models with at least one complex of each oligomer size are to be considered. This is useful when the data are multivariate proportions (i.e. mass distribution data) and each <code>j</code> -mer is clearly present.
<code>KIC</code>	The initial condition of all <code>K</code> parameters optimized. The default is <code>IC=1</code> (in <code>uM</code>).
<code>kIC</code>	The initial condition of all <code>k</code> parameters optimized. The default is <code>kIC=1</code> (in <code>1/seconds per occupied active site</code>).
<code>m1</code>	The hub protein's monomer mass in <code>kDa</code> . The default is <code>90</code> for the big (<code>R1</code>) subunit of ribonucleotide reductase (<code>RNR</code>). This only matters if the data is mass data. Negative numbers imply fixed values and positive numbers imply starting values to be fitted to the data.
<code>p</code>	Probability that hub can oligomerize, i.e. is not damaged. Set to a positive value if additional rows are to be added to the output dataframe to include models with <code>p</code> freely estimated. Set negative to hold fixed. Value is the initial or fixed value.
<code>forceM1</code>	Set <code>TRUE</code> to force all models to estimate <code>M1</code> , i.e. to not generate models with <code>M1</code> fixed.
<code>forceP</code>	Set <code>TRUE</code> to force all models to estimate <code>p</code> , i.e. to not generate models with fixed <code>p</code> .

Details

This function is complicated by the fact that one readily runs out of memory with 29 complexes and thus roughly 500,000,000 spur models (in this case 8 GB RAM allows at most 16 bytes per model!). Thus, chunks of the spur model space must be created, fitted and summarized in sizes small enough to fit into memory. As both an input and an output, `state` links successive calls to this function. It keeps track of where we are in the spur graph model space and it allows searches through the low parameter number models without first defining all of the higher parameter number models (and thus consuming all of the RAM in the process).

Value

A list with components

`chunk` A dataframe where each row is a spur model.

state The state, defined in the same way as the input argument `state` since the output state of one call is the input `state` of the next call.

maxReached This is TRUE if the maximum number of parameter has been reached.

lastCompleted When all of the models with j K parameters have been specified, `lastCompleted` equals j . This is useful in `ems` when `smart = TRUE` as it defines how far into the model space the fitting process has gone so far in terms of numbers of model parameters.

Note

This work was supported by the National Cancer Institute (K25CA104791).

Author(s)

Tom Radivoyevitch (txr24@case.edu)

References

Radivoyevitch, T. (2008) Equilibrium model selection: dTTP induced R1 dimerization. *BMC Systems Biology* **2**, 15.

See Also

[mkGrids](#), [ccems](#), [combn](#)

Examples

```
library(ccems)
topology <- list(
  heads=c("R1t0", "R2t0"),
  sites=list(
    s=list(
      m=c("R1t1"),      # s-site   thread #
                        # monomer    1
      d=c("R2t1", "R2t2") # dimer    2
    )
  )
)
g <- mkg(topology)
mkSpurs(g, p=0.95, doTights=TRUE)
```

Description

This is a compilation of ribonucleotide reductase (RNR) literature data into one data frame.

Usage

data (RNR)

Format

A data frame with 546 observations on the following 23 variables.

- R** The total concentration of R1 (big R for the big subunit of RNR).
- r** The total concentration of R2 dimer (small r for the small subunit of RNR).
- U** The total concentration of UDP.
- C** The total concentration of CDP.
- G** The total concentration of GDP.
- A** The total concentration of ADP.
- t** The total concentration of dTTP.
- t \bar{t}** The concentration of free dTTP.
- g** The total concentration of dGTP.
- g \bar{g}** The concentration of free dGTP.
- a** The total concentration of dATP.
- a \bar{a}** The concentration of free dATP.
- x** The total concentration of ATP.
- Mg** The total concentration of magnesium.
- v** The velocity of the reduction reaction in 1/seconds; v in Scott et al should be multiplied by .00151 to create these units, save figure 7 (GDP reduction) which is in percentages of the maximum velocity.
- m** The weight averaged average mass.
- n** The number of ligands bound. Exception is Ingemarson et al. 1996 where it is a possibly negative BIAcore experiment parameter.
- fg** The figure number in the original paper.
- year** The year of the original paper.
- jrn1** The journal of the data source: Bioc is Biochemistry, JBC is Journal of Biological Chemistry.
- vol** The volume number of the article.
- page** The page number of the article.
- frstAut** The first author of the article. This is a factor with levels IngemarsonR, KashlanOB, Rofougaran, sanath, and ScottCP.
- organism** An element of c ("human", "mouse", "yeast", "ecoli"). This dataset is predominantly mouse RNR data.

Details

All concentrations are in micromolar. To some extent, capital letters are used for ribonucleotides and small letters for deoxyribonucleotides. Free concentrations, where given, were model dependent and were thus converted back to total concentrations, i.e. to make them model independent. Values of zero that lead a sequence of measurements of increasing concentrations were set to $1e-2$ (essentially zero) to allow $[x]>0$ subsetting.

Source

The figure number, year, first author, journal, volume and page number are all included in the dataframe. The main reference is Kashlan OB, Scott CP, Lear JD, Cooperman BS: A comprehensive model for the allosteric regulation of mammalian ribonucleotide reductase. Functional consequences of ATP- and dATP-induced oligomerization of the large subunit. *Biochemistry* 2002, 41(2):462-474. The first seven rows, by first author Sanath (Sanath Wijerathna of the Dealwis Lab at CWRU), are unpublished.

References

Radvoyevitch, T. (2008) Equilibrium model selection: dTTP induced R1 dimerization. *BMC Systems Biology* 2, 15.

Examples

```
library(ccems)
RNR[1:40,]
d=subset(RNR, (year==2002) & (fg==1) & (X>0), select=c(R,X,m,year))
plot(d$X,d$m,xlab="Total [ATP]", ylab="Weight averaged R1 mass",
     main="Kashlan et al. Biochemistry, 2002, Fig. 1 (DLS data)")
```

simulateData

Simulate Data

Description

This function generates expected values of responses at the total concentrations of the dataframe `g$d` or at points specified in `predict` if `predict` is not `NULL`.

Usage

```
simulateData(g, init = FALSE, predict = NULL, typeYP = NULL)
```

Arguments

`g` A specific model/hypothesis created by `mkModel`.

`init` This is `TRUE` only in first calls to this function by the parameter estimate optimization algorithm. When `TRUE` the initial AIC value is set.

predict	A dataframe of total concentrations of the reactants at which response predictions are desired.
typeYP	The type of output data desired for the predictions. Options are "m" and "v" for average mass and reaction velocity, respectively. The option "P" (proportions) for mass distribution data is still in development.

Details

This function is the workhorse core of the nonlinear least squares algorithm, so its speed is critical which is why it uses compiled C code when `g$TCC = TRUE`. In addition to model fitting, this function can also be used to predict system response surfaces over grids of physiologically relevant total concentrations of the reactants. It is assumed that the model used to formulate predictions is based on data, and that the output type of the predictions is the same as the output type used to build the model.

Value

The input model object augmented to include the following fields if `predict` is `NULL`.

echk	A matrix that checks the TCC solver and <code>g\$fback</code> . Matrix column names that end in Q should match their sans-Q counterparts.
eSS	The expected steady state concentrations of species (complexes and free reactants). For each row of the data dataframe there is a row in this matrix. Its contents are the TCC solver solution (free reactant expected concentrations) and the result of applying <code>g\$fback</code> to them to create expected complex concentrations.
res	The residuals of the fit.
nData	The number of data points (i.e. rows) in the data dataframe <code>g\$d</code> .
SSE	The initial and final sum of squared errors (i.e. residual sum of squares).
AIC	The initial and final Akaike Information Criterion values, corrected for small samples. Since nonlinear least squares is used $AIC = N \cdot \log(SSE/N) + 2 \cdot P + 2 \cdot P \cdot (P+1) / (N-P-1) + N \cdot \log(2 \cdot \pi) + N$ where $N = nData$ and P is the number of estimated parameters (including the variance).
predict	The input argument <code>predict</code> with an additional expected system response column named "EY".

Note

The function `fitModel` augments the input model object by the same six fields above because it calls this function iteratively.

Measurements are often made at total concentrations that are substantially higher than physiological values due to signal-to-noise limitations. Thus, predictions in physiologically relevant (and thus important) regions tend to be weak.

This work was supported by the National Cancer Institute (K25CA104791).

Author(s)

Tom Radivoyevitch (txr24@case.edu)

References

Radvoyevitch, T. (2008) Equilibrium model selection: dTTP induced R1 dimerization. *BMC Systems Biology* **2**, 15.

See Also

The experimental design example `expDesign` in the `docs` directory.

Examples

```
library(ccems)
topology <- list(
  heads=c("R1t0", "R2t0"),
  sites=list(
    s=list(
      m=c("R1t1"),          # s-site   thread #
                        # monomer     1
      d=c("R2t1", "R2t2") # dimer    2
    )
  )
)
g <- mkg(topology, TCC=TRUE)
d=subset(RNR, (year==2001) & (fg==1) & (t>0) & (G==0), select=c(R, t, m, year))
names(d)[1:2] <-c("RT", "tT")
mdl=mkModel(g, "IIIJ", d, Kjparams=c(R2t0=Inf, R1t1=Inf, R2t1=Inf, R2t2=1),
           pparams=c(p=1))
fmdl <- fitModel(mdl)
pt=c(.1, 1:20)
predict <- data.frame(RT = rep(7.6, length(pt)), tT = pt)
df <- simulateData(fmdl, predict=predict, typeYP="m")$predict
plot(d$tT, d$m, type="p", xlab="[dTTP] (uM)", ylab="Weight averaged R1 mass",
     main="Scott et al. Biochemistry, 2001, Fig. 1 (DLS data)")
lines(df$tT, df$EY)
```

 TK1

Thymidine Kinase 1 Data

Description

Human thymidine kinase 1 (i.e. cytosolic) data.

Usage

```
data(TK1)
```

Format

A data frame with the following columns.

E The total concentration of TK1 enzyme.

S The total concentration of dT (the nucleoside substrate).

x The total concentration of ATP.

v The velocity of the kinase reaction.

fg The figure number in the original paper.

year The year of the original paper.

jrn1 The journal of the data source: PEP is Protein Expression and Purification, EJB is European Journal of Biochemistry, BBRC is Biochem Biophys Res Commun, JBC is Journal of Biological Chemistry.

vol The volume number of the article.

page The page number of the article.

frstAut The first author of the article. This is a factor.

index The articles indexed as 1 through 5.

k The measured/average activity in 1/sec per enzyme molecule present.

Details

All concentrations are in micromolar.

Source

The figure number, year, first author, journal, volume and page number are all included in the dataframe.

Examples

```
## Note that two windows devices will end up exactly on top of each other.
## Please move device 3 below device 2 to compare their residual plots.
library(ccems)
## Warning: the next line clears all existing figures!!
if (!is.null(dev.list())) for (i in 2:max(dev.list())) dev.off(i);
for (j in 1:2) {
  if (.Platform$OS.type=="windows")
    windows(width = 10, height = 4, restoreConsole = TRUE,
            ypos=ifelse(j==2, -50, 0)) else X11(width=10,height=4)
  par(mfcol=c(2, 5), mar=c(4, 4, 2, 1)+.1)
  for (i in 1:5) {
    d=subset(TK1, index==i, select=c(E, S, k, frstAut, year))
    if (j==1)
      hillda<-nls(k~kmax*(S/S50)^h/(1+(S/S50)^h), d, start=list(kmax=5, S50=1, h=1))
    if (j==2)
      hillda<-nls(k~kmax*(S/S50)^h/(1+(S/S50)^h), d,
                 start=list(kmax=5, S50=1, h=1), weights=1/k^2)
```

```

print(hillda)
plot(d$S,d$k,xlab="Total [dT]",log="xy", ylab="k (1/sec)",
      main=paste(d[1,"frstAut"],d[1,"year"]))
mtext(paste("N =",length(d$k)),line=-3,side=1,font=1,cex=0.7)
mtext(paste("Hill Coeff = ",format(hillda$m$getPars()["h"],digits=3),sep=""),
      ,line=-2,side=1,font=1,cex=0.7)
## Note that the specific activity is ~16 fold higher in non-Birringer data
lgx=log(d$S)
upr=range(lgx)[2]
lwr=range(lgx)[1]
del=(upr-lwr)/50
fineX=exp(seq(lwr,upr,by=del))
lines(fineX,predict(hillda,list(S=fineX)),col="black",lwd=1)
plot(hillda$m$fitted(),hillda$m$resid(),xlab="Fitted Value",
      ylab="Residual",mar=c(2,2,0,1)+.1)
## Note that variance increases with the mean in non-Birringer data
## and that the 2000 and 1993 Hill fits are poor at low k (and [S])
}
}

## Not run:
if (.Platform$OS.type=="windows") # now create a window for ccems fits
  windows(width = 10, height = 4,restoreConsole = TRUE, ypos=50) else
  X11(width=10,height=4)
library(ccems)
topology <- list(
  heads=c("E1S0"), # E1S0 = substrate free E
  sites=list(
    c=list( # c for catalytic site
      t=c("E1S1","E1S2","E1S3","E1S4")
    ) # t for tetramer
  )
) # in transform below, TK1 is 25kDa => 25mg/umole
g <-mkg(topology, activity=TRUE,TCC=FALSE)

getKk <- function(x) {t(x$report[c(paste("E1S",0:3,"_S",sep=""),
  paste("kE1S",1:4,sep="")), "final",drop=FALSE])}
getAIC <- function(x) { x$report["AIC","final"]}
getSSE <- function(x) { x$report["SSE","final"]}
outs=list(NULL)
par(mfcol=c(2,5),mar=c(4,4,2,1)+.1)
for (i in 1:5) {
d=subset(TK1,index==i,select=c(E,S,k,frstAut,year))
plot(d$S,d$k,xlab="Total [dT]",log="xy", ylab="k (1/sec)",
      main=paste(d[1,"frstAut"],d[1,"year"]))
names(d)[1:2]= c("ET","ST")
tops=ems(d,g,maxTotalPs=3,doSpurs=FALSE)# takes ~15 sec for each dataset
lgx=log(d$ST)
upr=range(lgx)[2]
lwr=range(lgx)[1]
del=(upr-lwr)/50
fineX=exp(seq(lwr,upr,by=del))
predict <- data.frame(ET = rep(d$ET[1],length(fineX)), ST = fineX)

```

```

df <- simulateData(tops[[1]],predict=predict,typeYP="k")$predict
lines(df$ST,df$EY)
Kk=lapply(tops,getKk)
nms=names(Kk)
rowList=data.frame(NULL)
for (j in nms) {
  rowList=rbind(rowList,Kk[[j]])
}
rownames(rowList)<-nms
aic=sapply(tops,getAIC)
sse=sapply(tops,getSSE)
eDelAIC=exp(-(aic-min(aic)))
wgts=eDelAIC/sum(eDelAIC)
print(sum(wgts))
df=data.frame(aic,sse,wgts,rowList)
M=as.matrix(rowList)
ma=exp(wgts%*%log(M)) # average in space of gibbs free energy changes
dataID=paste(d[1,"frstAut"],d[1,"year"],sep="")
outs[[dataID]]$df=df
outs[[dataID]]$ma=ma
plot(tops[[1]]$d$EY,tops[[1]]$res,xlab="Fitted Value",
      ylab="Residual",main=tops[[1]]$mid)
## Note that the 2000 and 1993 fits are now improved
}
outs=outs[-1] # remove leading NULL
print(outs) # compare model averages across datasets
par(mfrow=c(1,1))
## End(Not run)

```

Index

*Topic **datasets**

RNR, [25](#)

TK1, [29](#)

*Topic **math**

simulateData, [27](#)

*Topic **models**

ems, [6](#)

fitModel, [9](#)

mkg, [11](#)

mkGrids, [14](#)

mkKd2Kj, [17](#)

mkModel, [19](#)

mkSpurs, [22](#)

*Topic **package**

ccems-package, [2](#)

ccems, [8](#), [10](#), [11](#), [13](#), [16](#), [18](#), [25](#)

ccems (*ccems-package*), [2](#)

ccems-package, [2](#)

combn, [25](#)

ems, [3](#), [6](#), [10](#), [13](#)

fitModel, [9](#), [22](#)

mkg, [3](#), [8](#), [11](#), [20](#), [22](#)

mkGrids, [14](#), [18](#), [25](#)

mkKd2Kj, [17](#), [20](#), [22](#)

mkModel, [10](#), [19](#)

mkSpurs, [6](#), [15](#), [16](#), [22](#)

RNR, [25](#)

simulateData, [27](#)

TK1, [29](#)