

# Package ‘Cyclops’

June 18, 2015

**Type** Package

**Title** Cyclic Coordinate Descent for Logistic, Poisson and Survival Analysis

**Version** 1.1.0

**Author** Marc A. Suchard [aut, cre],  
Martijn J. Schuemie [aut],  
Trevor R. Shaddox [aut]

**Maintainer** Marc A. Suchard <msuchard@ucla.edu>

**Description** This model fitting tool incorporates cyclic coordinate descent and majorization-minimization approaches to fit a variety of regression models found in large-scale observational healthcare data. Implementations focus on computational optimization and fine-scale parallelization to yield efficient inference in massive datasets.

**License** Apache License 2.0

**LazyData** Yes

**URL** <https://github.com/ohdsi/cyclops>

**BugReports** <https://github.com/ohdsi/cyclops/issues>

**Depends** R (>= 3.1.0)

**Imports** Matrix,  
Rcpp (>= 0.11.3),  
bit,  
ff,  
ffbase,  
RcppParallel

**LinkingTo** Rcpp,  
BH (>= 1.51.0),  
RcppEigen (>= 0.3.2),  
RcppParallel

**Suggests** testthat,  
survival,  
gnm,  
ggplot2

## R topics documented:

coef.cyclopsFit . . . . .	2
confint.cyclopsFit . . . . .	3
convertToCyclopsData . . . . .	4
coverage . . . . .	6
createControl . . . . .	6
createCyclopsData . . . . .	8
createPrior . . . . .	9
cyclops . . . . .	10
fitCyclopsModel . . . . .	11
fitCyclopsSimulation . . . . .	12
getCovariateIds . . . . .	12
getCovariateTypes . . . . .	13
getHyperParameter . . . . .	13
getNumberOfCovariates . . . . .	14
getNumberOfRows . . . . .	14
getNumberOfStrata . . . . .	14
isInitialized . . . . .	15
isSorted . . . . .	15
logLik.cyclopsFit . . . . .	16
mse . . . . .	17
Multitype . . . . .	17
oxford . . . . .	18
predict.cyclopsFit . . . . .	18
print.cyclopsData . . . . .	19
print.cyclopsFit . . . . .	19
readCyclopsData . . . . .	20
simulateCyclopsData . . . . .	21
summary.cyclopsData . . . . .	22
vcov.cyclopsFit . . . . .	23
<b>Index</b>	<b>24</b>

---

coef.cyclopsFit	<i>Extract model coefficients</i>
-----------------	-----------------------------------

---

### Description

coef.cyclopsFit extracts model coefficients from an Cyclops model fit object

### Usage

```
## S3 method for class 'cyclopsFit'
coef(object, ...)
```

### Arguments

object	Cyclops model fit object
...	Other arguments

**Value**

Named numeric vector of model coefficients.

---

confint.cyclopsFit	<i>Confidence intervals for Cyclops model parameters</i>
--------------------	--

---

**Description**

confint.cyclopsFit profiles the data likelihood to construct confidence intervals of arbitrary level. Usually it only makes sense to do this for variables that have not been regularized TODO: Profile data likelihood or joint distribution of remaining parameters.

**Usage**

```
## S3 method for class 'cyclopsFit'
confint(object, parm, level = 0.95,
        overrideNoRegularization = FALSE, includePenalty = TRUE, ...)
```

**Arguments**

object	A fitted Cyclops model object
parm	A specification of which parameters require confidence intervals, either a vector of numbers of covariateId names
level	Numeric: confidence level required
overrideNoRegularization	Logical: Enable confidence interval estimation for regularized parameters
includePenalty	Logical: Include regularized covariate penalty in profile
...	Additional argument(s) for methods

**Value**

A matrix with columns reporting lower and upper confidence limits for each parameter. These columns are labelled as  $(1 - \text{level}) / 2$  and  $1 - (1 - \text{level}) / 2$  in percent (by default 2.5 percent and 97.5 percent)

**Examples**

```
#Generate some simulated data:
sim <- simulateCyclopsData(nstrata = 1, nrows = 1000, ncovars = 2, eCovarsPerRow = 0.5,
                           model = "poisson")
cyclopsData <- convertToCyclopsData(sim$outcomes, sim$covariates, modelType = "pr",
                                    addIntercept = TRUE)

#Define the prior and control objects to use cross-validation for finding the
#optimal hyperparameter:
prior <- createPrior("laplace", exclude = 0, useCrossValidation = TRUE)
control <- createControl(cvType = "auto", noiseLevel = "quiet")

#Fit the model
fit <- fitCyclopsModel(cyclopsData, prior = prior, control = control)
```

```
#Find out what the optimal hyperparameter was:
getHyperParameter(fit)

#Extract the current log-likelihood, and coefficients
logLik(fit)
coef(fit)

#We can only retrieve the confidence interval for unregularized coefficients:
confint(fit, c(0))
```

---

convertToCyclopsData	<i>Convert data from two data frames or ffdF objects into a CyclopsData object</i>
----------------------	--

---

## Description

convertToCyclopsData loads data from two data frames or ffdF objects, and inserts it into a Cyclops data object.

## Usage

```
convertToCyclopsData(outcomes, covariates, modelType = "lr",
  addIntercept = TRUE, checkSorting = TRUE, checkRowIds = TRUE,
  quiet = FALSE)

## S3 method for class 'ffdf'
convertToCyclopsData(outcomes, covariates, modelType = "lr",
  addIntercept = TRUE, checkSorting = TRUE, checkRowIds = TRUE,
  quiet = FALSE)

## S3 method for class 'data.frame'
convertToCyclopsData(outcomes, covariates,
  modelType = "lr", addIntercept = TRUE, checkSorting = TRUE,
  checkRowIds = TRUE, quiet = FALSE)
```

## Arguments

outcomes	A data frame or ffdF object containing the outcomes with predefined columns (see below).
covariates	A data frame or ffdF object containing the covariates with predefined columns (see below).
modelType	Cyclops model type. Current supported types are "pr", "cpr", "lr", "clr", or "cox"
addIntercept	Add an intercept to the model?
checkSorting	Check if the data are sorted appropriately, and if not, sort.
checkRowIds	Check if all rowIds in the covariates appear in the outcomes.
quiet	If true, (warning) messages are suppressed.

## Details

These columns are expected in the outcome object:

stratumId	(integer)	(optional) Stratum ID for conditional regression models
rowId	(integer)	Row ID is used to link multiple covariates (x) to a single outcome (y)
y	(real)	The outcome variable
time	(real)	For models that use time (e.g. Poisson or Cox regression) this contains time (e.g. number of days)

These columns are expected in the covariates object:

stratumId	(integer)	(optional) Stratum ID for conditional regression models
rowId	(integer)	Row ID is used to link multiple covariates (x) to a single outcome (y)
covariateId	(integer)	A numeric identifier of a covariate
covariateValue	(real)	The value of the specified covariate

Note: If checkSorting is turned off, the outcome table should be sorted by stratumId (if present) and then rowId except for Cox regression when the table should be sorted by stratumId (if present), - time, y, and rowId. The covariate table should be sorted by covariateId, stratumId (if present), rowId except for Cox regression when the table should be sorted by covariateId, stratumId (if present), - time, y, and rowId.

## Value

An object of type cyclopsData

## Methods (by class)

- `ffdf`: Convert data from two `ffdf`
- `data.frame`: Convert data from two `data.frame`

## Examples

```
#Convert infert dataset to Cyclops format:
covariates <- data.frame(stratumId = rep(infert$stratum, 2),
                        rowId = rep(1:nrow(infert), 2),
                        covariateId = rep(1:2, each = nrow(infert)),
                        covariateValue = c(infert$spontaneous, infert$induced))
outcomes <- data.frame(stratumId = infert$stratum,
                      rowId = 1:nrow(infert),
                      y = infert$case)

#Make sparse:
covariates <- covariates[covariates$covariateValue != 0, ]

#Create Cyclops data object:
cyclopsData <- convertToCyclopsData(outcomes, covariates, modelType = "clr",
                                   addIntercept = FALSE)

#Fit model:
fit <- fitCyclopsModel(cyclopsData, prior = createPrior("none"))
```

---

coverage	<i>Coverage</i>
----------	-----------------

---

### Description

coverage computes the coverage on confidence intervals

### Usage

```
coverage(goldStandard, lowerBounds, upperBounds)
```

### Arguments

goldStandard	Numeric vector
lowerBounds	Numeric vector. Lower bound of the confidence intervals
upperBounds	Numeric vector. Upper bound of the confidence intervals

### Value

The proportion of times goldStandard falls between lowerBound and upperBound

---

createControl	<i>Create a Cyclops control object</i>
---------------	--

---

### Description

createControl creates a Cyclops control object for use with [fitCyclopsModel](#).

### Usage

```
createControl(maxIterations = 1000, tolerance = 1e-06,
  convergenceType = "gradient", cvType = "grid", fold = 10,
  lowerLimit = 0.01, upperLimit = 20, gridSteps = 10, cvRepetitions = 1,
  minCVData = 100, noiseLevel = "silent", threads = 1, seed = NULL,
  resetCoefficients = FALSE, startingVariance = -1, useKKTswindle = FALSE,
  tuneSwindle = 10, selectorType = "default")
```

### Arguments

maxIterations	Integer: maximum iterations of Cyclops to attempt before returning a failed-to-converge error
tolerance	Numeric: maximum relative change in convergence criterion from successive iterations to achieve convergence
convergenceType	String: name of convergence criterion to employ (described in more detail below)
cvType	String: name of cross validation search. Option "auto" selects an auto-search following BBR. Option "grid" selects a grid-search cross validation

fold	Numeric: Number of random folds to employ in cross validation
lowerLimit	Numeric: Lower prior variance limit for grid-search
upperLimit	Numeric: Upper prior variance limit for grid-search
gridSteps	Numeric: Number of steps in grid-search
cvRepetitions	Numeric: Number of repetitions of X-fold cross validation
minCVData	Numeric: Minimum number of data for cross validation
noiseLevel	String: level of Cyclops screen output ("silent", "quiet", "noisy")
threads	Numeric: Specify number of CPU threads to employ in cross-validation; default = 1 (auto = -1)
seed	Numeric: Specify random number generator seed. A null value sets seed via <a href="#">Sys.time</a> .
resetCoefficients	Logical: Reset all coefficients to 0 between model fits under cross-validation
startingVariance	Numeric: Starting variance for auto-search cross-validation; default = -1 (use estimate based on data)
useKKTswindle	Logical: Use the Karush-Kuhn-Tucker conditions to limit search
tuneSwindle	Numeric: Size multiplier for active set
selectorType	String: name of exchangeable sampling unit. If missing, then default for model is used. Option "byPid" selects entire strata. Option "byRow" selects single rows Todo: Describe convergence types

## Value

A Cyclops control object of class inheriting from "cyclopsControl" for use with [fitCyclopsModel](#).

## Examples

```
#Generate some simulated data:
sim <- simulateCyclopsData(nstrata = 1, nrows = 1000, ncovars = 2, eCovarsPerRow = 0.5,
                           model = "poisson")
cyclopsData <- convertToCyclopsData(sim$outcomes, sim$covariates, modelType = "pr",
                                   addIntercept = TRUE)

#Define the prior and control objects to use cross-validation for finding the
#optimal hyperparameter:
prior <- createPrior("laplace", exclude = 0, useCrossValidation = TRUE)
control <- createControl(cvType = "auto", noiseLevel = "quiet")

#Fit the model
fit <- fitCyclopsModel(cyclopsData, prior = prior, control = control)

#Find out what the optimal hyperparameter was:
getHyperParameter(fit)

#Extract the current log-likelihood, and coefficients
logLik(fit)
coef(fit)

#We can only retrieve the confidence interval for unregularized coefficients:
confint(fit, c(0))
```

---

createCyclopsData	Create a Cyclops data object
-------------------	------------------------------

---

## Description

createCyclopsData creates a Cyclops data object from an R formula or data matrices.

## Usage

```
createCyclopsData(formula, sparseFormula, indicatorFormula, modelType, data,
  subset, weights, offset, time = NULL, pid = NULL, y = NULL,
  type = NULL, dx = NULL, sx = NULL, ix = NULL, model = FALSE,
  method = "cyclops.fit")
```

## Arguments

formula	An object of class " <b>formula</b> " that provides a symbolic description of the numerically dense model response and terms.
sparseFormula	An object of class " <b>formula</b> " that provides a symbolic description of numerically sparse model terms.
indicatorFormula	An object of class " <b>formula</b> " that provides a symbolic description of {0,1} model terms.
modelType	character string: Valid types are listed below.
data	An optional data frame, list or environment containing the variables in the model.
subset	Currently unused
weights	Currently unused
offset	Currently unused
time	Currently undocumented
pid	Optional vector of integer stratum identifiers. If supplied, all rows must be sorted by increasing identifiers
y	Currently undocumented
type	Currently undocumented
dx	Optional dense " <b>Matrix</b> " of covariates
sx	Optional sparse " <b>Matrix</b> " of covariates
ix	Optional {0,1} " <b>Matrix</b> " of covariates
model	Currently undocumented
method	Currently undocumented

## Details

This function creates a Cyclops model data object from R "**formula**" or directly from numeric vectors and matrices to define the model response and covariates. If specifying a model using a "**formula**", then the left-hand side define the model response and the right-hand side defines dense covariate terms. Objects provided with "sparseFormula" and "indicatorFormula" must be include left-hand side responses and terms are coersed into sparse and indicator representations for computational efficiency.

Items to discuss: \* Only use formula or (y,dx,...) \* stratum() in formula \* offset() in formula \* when "stratum" (renamed from pid) are necessary \* when "time" are necessary



**Value**

A list that contains a Cyclops model data object pointer and an operation duration

**Models**

Currently supported model types are:

"ls"	Least squares
"pr"	Poisson regression
"lr"	Logistic regression
"clr"	Conditional logistic regression
"cpr"	Conditional Poisson regression
"sccs"	Self-controlled case series
"cox"	Cox proportional hazards regression

**Examples**

```
## Dobson (1990) Page 93: Randomized Controlled Trial :
counts <- c(18, 17, 15, 20, 10, 20, 25, 13, 12)
outcome <- gl(3, 1, 9)
treatment <- gl(3, 3)
cyclopsData <- createCyclopsData(
  counts ~ outcome + treatment,
  modelType = "pr")
cyclopsFit <- fitCyclopsModel(cyclopsData)

cyclopsData2 <- createCyclopsData(
  counts ~ outcome,
  indicatorFormula = ~ treatment,
  modelType = "pr")
summary(cyclopsData2)
cyclopsFit2 <- fitCyclopsModel(cyclopsData2)
```

---

createPrior	<i>Create a Cyclops prior object</i>
-------------	--------------------------------------

---

**Description**

createPrior creates a Cyclops prior object for use with [fitCyclopsModel](#).

**Usage**

```
createPrior(priorType, variance = 1, exclude = c(), graph = NULL,
  useCrossValidation = FALSE, forceIntercept = FALSE)
```

**Arguments**

priorType	Character: specifies prior distribution. See below for options
variance	Numeric: prior distribution variance
exclude	A vector of numbers or covariateId names to exclude from prior

graph                    Child-to-parent mapping for a hierarchical prior  
 useCrossValidation                    Logical: Perform cross-validation to determine prior variance.  
 forceIntercept   Logical: Force intercept coefficient into prior

### Value

A Cyclops prior object of class inheriting from "cyclopsPrior" for use with fitCyclopsModel.

### Prior types

We specify all priors in terms of their variance parameters. Similar fitting tools for regularized regression often parameterize the Laplace distribution in terms of a rate "lambda" per observation. See "glmnet", for example.

$\text{variance} = 2 * (\text{nobs} * \text{lambda})^2$  or  $\text{lambda} = \sqrt{2 / \text{variance}} / \text{nobs}$

### Examples

```
#Generate some simulated data:
sim <- simulateCyclopsData(nstrata = 1, nrows = 1000, ncovars = 2, eCovarsPerRow = 0.5,
                           model = "poisson")
cyclopsData <- convertToCyclopsData(sim$outcomes, sim$covariates, modelType = "pr",
                                    addIntercept = TRUE)

#Define the prior and control objects to use cross-validation for finding the
#optimal hyperparameter:
prior <- createPrior("laplace", exclude = 0, useCrossValidation = TRUE)
control <- createControl(cvType = "auto", noiseLevel = "quiet")

#Fit the model
fit <- fitCyclopsModel(cyclopsData, prior = prior, control = control)

#Find out what the optimal hyperparameter was:
getHyperParameter(fit)

#Extract the current log-likelihood, and coefficients
logLik(fit)
coef(fit)

#We can only retrieve the confidence interval for unregularized coefficients:
confint(fit, c(0))
```

---

cyclops

*Cyclops: Cyclic coordinate descent for logistic, Poisson and survival analysis*

---

### Description

The Cyclops package incorporates cyclic coordinate descent and majorization-minimization approaches to fit a variety of regression models found in large-scale observational healthcare data. Implementations focus on computational optimization and fine-scale parallelization to yield efficient inference in massive datasets.

---

fitCyclopsModel	<i>Fit a Cyclops model</i>
-----------------	----------------------------

---

## Description

fitCyclopsModel fits a Cyclops model data object

## Usage

```
fitCyclopsModel(cyclopsData, prior, control, weights = NULL,
  forceNewObject = FALSE, returnEstimates = TRUE,
  startingCoefficients = NULL)
```

## Arguments

cyclopsData	A Cyclops data object
prior	A prior object. More details are given below.
control	Cyclops control object, see " <a href="#">control</a> "
weights	Vector of 0/1 weights for each data row
forceNewObject	Logical, forces the construction of a new Cyclops model fit object
returnEstimates	Logical, return regression coefficient estimates in Cyclops model fit object
startingCoefficients	Vector of starting values for optimization

## Details

This function performs numerical optimization to fit a Cyclops model data object.

## Value

A list that contains a Cyclops model fit object pointer and an operation duration

## Prior

Currently supported prior types are:

"none"	Useful for finding MLE
"laplace"	L_1 regularization
"normal"	L_2 regularization

## References

Suchard MA, Simpson SE, Zorych I, Ryan P, Madigan D. Massive parallelization of serial inference algorithms for complex generalized linear models. *ACM Transactions on Modeling and Computer Simulation*, 23, 10, 2013.

Simpson SE, Madigan D, Zorych I, Schuemie M, Ryan PB, Suchard MA. Multiple self-controlled case series for large-scale longitudinal observational databases. *Biometrics*, 69, 893-902, 2013.

Mittal S, Madigan D, Burd RS, Suchard MA. High-dimensional, massive sample-size Cox proportional hazards regression for survival analysis. *Biostatistics*, 15, 207-221, 2014.

### Examples

```
## Dobson (1990) Page 93: Randomized Controlled Trial :
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)
cyclopsData <- createCyclopsData(counts ~ outcome + treatment, modelType = "pr")
cyclopsFit <- fitCyclopsModel(cyclopsData, prior = createPrior("none"))
coef(cyclopsFit)
confint(cyclopsFit, c("outcome2", "treatment3"))
predict(cyclopsFit)
```

---

fitCyclopsSimulation    *Fit simulated data*

---

### Description

fitCyclopsSimulation fits simulated Cyclops data using Cyclops or a standard routine. This function is useful for simulation studies comparing the performance of Cyclops when considering large, sparse datasets.

### Usage

```
fitCyclopsSimulation(sim, useCyclops = TRUE, model = "logistic",
  coverage = TRUE, includePenalty = FALSE)
```

### Arguments

sim	A simulated Cyclops dataset generated via simulateCyclopsData
useCyclops	Logical: use Cyclops or a standard routine
model	String: Fitted regression model type
coverage	Logical: report coverage statistics
includePenalty	Logical: include regularized regression penalty in computing profile likelihood based confidence intervals

---

getCovariateIds    *Get covariate identifiers*

---

### Description

getCovariateIds returns a vector of integer covariate identifiers in a Cyclops data object

### Usage

```
getCovariateIds(object)
```

### Arguments

object	A Cyclops data object
--------	-----------------------

---

getCovariateTypes	<i>Get covariate types</i>
-------------------	----------------------------

---

**Description**

getCovariateTypes returns a vector covariate types in a Cyclops data object

**Usage**

```
getCovariateTypes(object, covariateLabel)
```

**Arguments**

object                    A Cyclops data object  
 covariateLabel   Integer vector: covariate identifiers to return

---

getHyperParameter	<i>Get hyperparameter</i>
-------------------	---------------------------

---

**Description**

getHyperParameter returns the current hyper parameter in a Cyclops model fit object

**Usage**

```
getHyperParameter(object)
```

**Arguments**

object                    A Cyclops model fit object

**Examples**

```
#Generate some simulated data:
sim <- simulateCyclopsData(nstrata = 1, nrows = 1000, ncovars = 2, eCovarsPerRow = 0.5,
                           model = "poisson")
cyclopsData <- convertToCyclopsData(sim$outcomes, sim$covariates, modelType = "pr",
                                    addIntercept = TRUE)

#Define the prior and control objects to use cross-validation for finding the
#optimal hyperparameter:
prior <- createPrior("laplace", exclude = 0, useCrossValidation = TRUE)
control <- createControl(cvType = "auto", noiseLevel = "quiet")

#Fit the model
fit <- fitCyclopsModel(cyclopsData, prior = prior, control = control)

#Find out what the optimal hyperparameter was:
getHyperParameter(fit)

#Extract the current log-likelihood, and coefficients
```

```
logLik(fit)
coef(fit)
```

```
#We can only retrieve the confidence interval for unregularized coefficients:
confint(fit, c(0))
```

---

```
getNumberOfCovariates
```

*Get total number of covariates*


---

### Description

getNumberOfCovariates returns the total number of covariates in a Cyclops data object

### Usage

```
getNumberOfCovariates(object)
```

### Arguments

object                      A Cyclops data object

---

```
getNumberOfRows
```

*Get total number of rows*


---

### Description

getNumberOfRows returns the total number of outcome rows in a Cyclops data object

### Usage

```
getNumberOfRows(object)
```

### Arguments

object                      A Cyclops data object

---

```
getNumberOfStrata
```

*Get number of strata*


---

### Description

getNumberOfStrata return the number of unique strata in a Cyclops data object

### Usage

```
getNumberOfStrata(object)
```

### Arguments

object                      A Cyclops data object

---

isInitialized	<i>Check if a Cyclops data object is initialized</i>
---------------	--

---

**Description**

isInitialized determines if an Cyclops data object is properly initialized and remains in memory. Cyclops data objects do not serialized/deserialize their back-end memory across R sessions.

**Usage**

```
isInitialized(object)
```

**Arguments**

object	Cyclops data object to test
--------	-----------------------------

---

isSorted	<i>Check if data are sorted by one or more columns</i>
----------	--

---

**Description**

isSorted checks wether data are sorted by one or more specified columns.

**Usage**

```
isSorted(data, columnNames, ascending = rep(TRUE, length(columnNames)))
```

```
## S3 method for class 'data.frame'
isSorted(data, columnNames, ascending = rep(TRUE,
  length(columnNames)))
```

```
## S3 method for class 'ffdf'
isSorted(data, columnNames, ascending = rep(TRUE,
  length(columnNames)))
```

**Arguments**

data	Either a data.frame of ffdf object.
columnNames	Vector of one or more column names.
ascending	Logical vector indicating the data should be sorted ascending or descending according the specified columns.

**Details**

This function currently only supports checking for sorting on numeric values.

**Value**

True or false

**Methods (by class)**

- data.frame: Check if a data.frame is sorted by one or more columns
- ffdof: Check if a ffdof is sorted by one or more columns

**Examples**

```
x <- data.frame(a = runif(1000), b = runif(1000))
x <- round(x, digits=2)
isSorted(x, c("a", "b"))

x <- x[order(x$a, x$b),]
isSorted(x, c("a", "b"))

x <- x[order(x$a, -x$b),]
isSorted(x, c("a", "b"), c(TRUE, FALSE))
```

---

logLik.cyclopsFit	<i>Extract log-likelihood</i>
-------------------	-------------------------------

---

**Description**

logLik returns the current log-likelihood of the fit in a Cyclops model fit object

**Usage**

```
## S3 method for class 'cyclopsFit'
logLik(object, ...)
```

**Arguments**

object	A Cyclops model fit object
...	Additional arguments

**Examples**

```
#Generate some simulated data:
sim <- simulateCyclopsData(nstrata = 1, nrows = 1000, ncovars = 2, eCovarsPerRow = 0.5,
                           model = "poisson")
cyclopsData <- convertToCyclopsData(sim$outcomes, sim$covariates, modelType = "pr",
                                    addIntercept = TRUE)

#Define the prior and control objects to use cross-validation for finding the
#optimal hyperparameter:
prior <- createPrior("laplace", exclude = 0, useCrossValidation = TRUE)
control <- createControl(cvType = "auto", noiseLevel = "quiet")

#Fit the model
fit <- fitCyclopsModel(cyclopsData, prior = prior, control = control)

#Find out what the optimal hyperparameter was:
getHyperParameter(fit)

#Extract the current log-likelihood, and coefficients
```



```
logLik(fit)
coef(fit)

#We can only retrieve the confidence interval for unregularized coefficients:
confint(fit, c(0))
```

---

mse	<i>Mean squared error</i>
-----	---------------------------

---

### Description

mse computes the mean squared error between two numeric vectors

### Usage

```
mse(goldStandard, estimates)
```

### Arguments

goldStandard	Numeric vector
estimates	Numeric vector

### Value

```
MSE(goldStandard, estimates)
```

---

Multitype	<i>Create a multitype outcome object</i>
-----------	--

---

### Description

Multitype creates a multitype outcome object, usually used as a response variable in a hierarchical Cyclops model fit.

### Usage

```
Multitype(y, type)
```

### Arguments

y	Numeric: Response count(s)
type	Numeric or factor: Response type

### Value

An object of class Multitype with length equal to the length of y and type.

### Examples

```
Multitype(c(0,1,0), as.factor(c("A","A","B")))
```

---

oxford	<i>Oxford self-controlled case series data</i>
--------	--

---

### Description

A dataset containing the MMR vaccination / meningitis in Oxford example from Farrington and Whitaker. There are 10 patients comprising 38 unique exposure intervals.

### Usage

```
data(oxford)
```

### Format

A data frame with 38 rows and 6 variables:

**indiv** patient identifier  
**event** number of events in interval  
**interval** interval length in days  
**agegr** age group  
**exgr** exposure group  
**loginterval** log interval length ...

### Source

<http://statistics.open.ac.uk/sccs/r.htm>

---

predict.cyclopsFit	<i>Model predictions</i>
--------------------	--------------------------

---

### Description

predict.cyclopsFit computes model response-scale predictive values for all data rows

### Usage

```
## S3 method for class 'cyclopsFit'
predict(object, ...)
```

### Arguments

object	A Cyclops model fit object
...	Additional arguments

---

print.cyclopsData	<i>Print a Cyclops data object</i>
-------------------	------------------------------------

---

**Description**

print.cyclopsData displays information about a Cyclops data model object.

**Usage**

```
## S3 method for class 'cyclopsData'  
print(x, show.call = TRUE, ...)
```

**Arguments**

x	A Cyclops data model object
show.call	Logical: display last call to construct the Cyclops data model object
...	Additional arguments

---

print.cyclopsFit	<i>Print a Cyclops model fit object</i>
------------------	---

---

**Description**

print.cyclopsFit displays information about a Cyclops model fit object

**Usage**

```
## S3 method for class 'cyclopsFit'  
print(x, show.call = TRUE, ...)
```

**Arguments**

x	A Cyclops model fit object
show.call	Logical: display last call to update the Cyclops model fit object
...	Additional arguments

---

readCyclopsData	<i>Read Cyclops data from file</i>
-----------------	------------------------------------

---

## Description

readCyclopsData reads a Cyclops-formatted text file.

## Usage

```
readCyclopsData(fileName, modelType)
```

## Arguments

fileName	Name of text file to be read. If fileName does not contain an absolute path, the name is relative to the current working directory, <a href="#">getwd</a> .
modelType	character string: Valid types are listed below.

## Details

This function reads a Cyclops-formatted text file and returns a Cyclops data object. The first line of the file may start with '#', indicating that it contains header options. Valid header options are:

row_label	(assume file contains a numeric column of unique row identifiers)
stratum_label	(assume file contains a numeric column of stratum identifiers)
weight	(assume file contains a column of row-specific model weights, currently unused)
offset	(assume file contains a dense column of linear predictor offsets)
bbr_outcome	(assume logistic outcomes are encoded -1/+1 following BBR)
log_offset	(assume file contains a dense column of values $x_i$ for which $\log(x_i)$ is the offset)
add_intercept	(automatically include an intercept column of all 1s for each entry)
indicator_only	(assume all covariates 0/1-valued and only covariate name is given)
sparse	(force all BBR formatted covariates to be represented as sparse, instead of sparse-indicator, columns .. really only for debugging)
dense	(force all BBR formatted covariates to be represented as dense columns.. really only for debugging)

Successive lines of the file are white-space delimited and follow the format:

```
[Row ID] {Stratum ID} [Weight] <Outcome> {Censored} {Offset} <BBR covariates>
```

- [optional]
- <required>
- {required or optional depending on model}

Bayesian binary regression (BBR) covariates are white-space delimited and generally in a sparse '<name>:<value>' format, where 'name' must (currently) be numeric and 'value' is non-zero. If option 'indicator\_only' is specified, then format is simply '<name>'. 'Row ID' and 'Stratum ID' must be numeric, and rows must be sorted such that equal 'Stratum ID' are consecutive. 'Stratum ID' is required for 'clr' and 'sccs' models. 'Censored' is required for a 'cox' model. 'Offset' is (currently) required for a 'sccs' model.

**Value**

A list that contains a Cyclops model data object pointer and an operation duration

**Models**

Currently supported model types are:

"ls"	Least squares
"pr"	Poisson regression
"lr"	Logistic regression
"clr"	Conditional logistic regression
"cpr"	Conditional Poisson regression
"sccs"	Self-controlled case series
"cox"	Cox proportional hazards regression

**Examples**

```
## Not run:
dataPtr = readCyclopsData(system.file("extdata/infert_ccd.txt", package="Cyclops"), "clr")

## End(Not run)
```

---

simulateCyclopsData	<i>Simulation Cyclops dataset</i>
---------------------	-----------------------------------

---

**Description**

simulateCyclopsData generates a simulated large, sparse data set for use by fitCyclopsSimulation.

**Usage**

```
simulateCyclopsData(nstrata = 200, nrows = 10000, ncovars = 20,
  effectSizeSd = 1, zeroEffectSizeProp = 0.9, eCovarsPerRow = ncovars/100,
  model = "survival")
```

**Arguments**

nstrata	Numeric: Number of strata
nrows	Numeric: Number of observation rows
ncovars	Numeric: Number of covariates
effectSizeSd	Numeric: Standard derivation of the non-zero simulated regression coefficients
zeroEffectSizeProp	Numeric: Expected proportion of zero effect size
eCovarsPerRow	Number: Effective number of non-zero covariates per data row
model	String: Simulation model. Choices are: logistic, poisson or survival

**Value**

A simulated data set

## Examples

```
#Generate some simulated data:
sim <- simulateCyclopsData(nstrata = 1, nrows = 1000, ncovars = 2, eCovarsPerRow = 0.5,
                           model = "poisson")
cyclopsData <- convertToCyclopsData(sim$outcomes, sim$covariates, modelType = "pr",
                                   addIntercept = TRUE)

#Define the prior and control objects to use cross-validation for finding the
#optimal hyperparameter:
prior <- createPrior("laplace", exclude = 0, useCrossValidation = TRUE)
control <- createControl(cvType = "auto", noiseLevel = "quiet")

#Fit the model
fit <- fitCyclopsModel(cyclopsData, prior = prior, control = control)

#Find out what the optimal hyperparameter was:
getHyperParameter(fit)

#Extract the current log-likelihood, and coefficients
logLik(fit)
coef(fit)

#We can only retrieve the confidence interval for unregularized coefficients:
confint(fit, c(0))
```

---

summary.cyclopsData	<i>Cyclops data object summary</i>
---------------------	------------------------------------

---

## Description

summary.cyclopsData summarizes the data held in an Cyclops data object.

## Usage

```
## S3 method for class 'cyclopsData'
summary(object, ...)
```

## Arguments

object	A Cyclops data object
...	Additional arguments

## Value

Returns a data.frame that reports simply summarize statistics for each covariate in a Cyclops data object.

---

vcov.cyclopsFit	<i>Calculate variance-covariance matrix for a fitted Cyclops model object</i>
-----------------	---

---

**Description**

vcov.cyclopsFit returns the variance-covariance matrix for all covariates of a Cyclops model object

**Usage**

```
## S3 method for class 'cyclopsFit'  
vcov(object, control, overrideNoRegularization = FALSE,  
      ...)
```

**Arguments**

object	A fitted Cyclops model object
control	A Cyclops <a href="#">control</a> object
overrideNoRegularization	Logical: Enable variance-covariance estimation for regularized parameters
...	Additional argument(s) for methods

**Value**

A matrix of the estimates covariances between all covariate estimates.

# Index

`coef.cyclopsFit`, [2](#)  
`confint.cyclopsFit`, [3](#)  
`control`, [11](#), [23](#)  
`convertToCyclopsData`, [4](#)  
`coverage`, [6](#)  
`createControl`, [6](#)  
`createCyclopsData`, [8](#)  
`createPrior`, [9](#)  
`cyclops`, [10](#)  
`cyclops-package (cyclops)`, [10](#)  
  
`fitCyclopsModel`, [6](#), [7](#), [9](#), [11](#)  
`fitCyclopsSimulation`, [12](#)  
`formula`, [8](#)  
  
`getCovariateIds`, [12](#)  
`getCovariateTypes`, [13](#)  
`getHyperParameter`, [13](#)  
`getNumberOfCovariates`, [14](#)  
`getNumberOfRows`, [14](#)  
`getNumberOfStrata`, [14](#)  
`getwd`, [20](#)  
  
`isInitialized`, [15](#)  
`isSorted`, [15](#)  
  
`logLik.cyclopsFit`, [16](#)  
  
`Matrix`, [8](#)  
`mse`, [17](#)  
`Multitype`, [17](#)  
  
`oxford`, [18](#)  
  
`predict.cyclopsFit`, [18](#)  
`print.cyclopsData`, [19](#)  
`print.cyclopsFit`, [19](#)  
  
`readCyclopsData`, [20](#)  
  
`simulateCyclopsData`, [21](#)  
`summary.cyclopsData`, [22](#)  
`Sys.time`, [7](#)  
  
`vcov.cyclopsFit`, [23](#)