fitglm

Create generalized linear regression model

Syntax

```
mdl = fitglm(tbl)
mdl = fitglm(X,y)
mdl = fitglm(___,modelspec)
mdl = fitglm(___,Name,Value)
```

Description

`mdl = fitglm(tbl)` returns a generalized linear model fit to variables in the table or dataset array `tbl`. By default, `fitglm` takes the last variable as the response variable. example

`mdl = fitglm(X,y)` returns a generalized linear model of the responses `y`, fit to the data matrix `X`. example

`mdl = fitglm(___,modelspec)` returns a generalized linear model of the type you specify in `modelspec`. example

`mdl = fitglm(___,Name,Value)` returns a generalized linear model with additional options specified by one or more `Name,Value` pair arguments. example

For example, you can specify which variables are categorical, the distribution of the response variable, and the link function to use.

Examples

collapse all

Fit a Logistic Regression Model

Make a logistic binomial model of the probability of smoking as a function of age, weight, and sex, using a two-way interactions model.

Open Live Script

Load the `hospital` dataset array.

```
load hospital
dsa = hospital;
```

Specify the model using a formula that allows up to two-way interactions between the variables age, weight, and sex. Smoker is the response variable.

```
modelspec = 'Smoker ~ Age*Weight*Sex - Age:Weight:Sex';
```

Fit a logistic binomial model.

```
mdl = fitglm(dsa,modelspec,'Distribution','binomial')
```

```
mdl =
Generalized linear regression model:
    logit(Smoker) ~ 1 + Sex*Age + Sex*Weight + Age*Weight
    Distribution = Binomial

Estimated Coefficients:
```

```
                      Estimate         SE         tStat       pValue
                    ___________    _________    ________    _______

    (Intercept)         -6.0492       19.749     -0.3063    0.75938
    Sex_Male            -2.2859       12.424    -0.18399    0.85402
    Age                 0.11691      0.50977     0.22934    0.81861
    Weight             0.031109      0.15208     0.20455    0.83792
    Sex_Male:Age       0.020734      0.20681     0.10025    0.92014
    Sex_Male:Weight     0.01216     0.053168     0.22871     0.8191
    Age:Weight      -0.00071959    0.0038964    -0.18468    0.85348


100 observations, 93 error degrees of freedom
Dispersion: 1
Chi^2-statistic vs. constant model: 5.07, p-value = 0.535
```

All of the p-values (under `pValue`) are large. This means none of the coefficients are significant. The large p-value for the test of the model, 0.535, indicates that this model might not differ statistically from a constant model.

GLM for Poisson Response

Create sample data with 20 predictors, and Poisson response using just three of the predictors, plus a constant.

Open Live Script

```
rng('default') % for reproducibility
X = randn(100,7);
mu = exp(X(:,[1 3 6])*[.4;.2;.3] + 1);
y = poissrnd(mu);
```

Fit a generalized linear model using the Poisson distribution.

```
mdl =  fitglm(X,y,'linear','Distribution','poisson')
```

```
mdl =
Generalized linear regression model:
    log(y) ~ 1 + x1 + x2 + x3 + x4 + x5 + x6 + x7
    Distribution = Poisson

Estimated Coefficients:
                  Estimate       SE        tStat       pValue
                 _________    ________    ________    __________

    (Intercept)    0.88723    0.070969      12.502    7.3149e-36
    x1             0.44413    0.052337      8.4858    2.1416e-17
    x2           0.0083388    0.056527     0.14752       0.88272
    x3             0.21518    0.063416      3.3932    0.00069087
    x4           -0.058386    0.065503    -0.89135       0.37274
    x5           -0.060824    0.073441     -0.8282       0.40756
    x6             0.34267    0.056778      6.0352    1.5878e-09
    x7             0.04316     0.06146     0.70225       0.48252
```

```
100 observations, 92 error degrees of freedom
Dispersion: 1
Chi^2-statistic vs. constant model: 119, p-value = 1.55e-22
```

The p-values of 2.14e-17, 0.00069, and 1.58e-09 indicate that the coefficients of the variables `x1`, `x3`, and `x6` are statistically significant.

Input Arguments

collapse all

`tbl` — Input data

table | dataset array

Input data including predictor and response variables, specified as a table or dataset array. The predictor variables and response variable can be numeric, logical, categorical, character, or string. The response variable can have a data type other than numeric only if `'Distribution'` is `'binomial'`.

- By default, `fitglm` takes the last variable as the response variable and the others as the predictor variables.
- To set a different column as the response variable, use the `ResponseVar` name-value pair argument.
- To use a subset of the columns as predictors, use the `PredictorVars` name-value pair argument.
- To define a model specification, set the `modelspec` argument using a formula or terms matrix. The formula or terms matrix specifies which columns to use as the predictor or response variables.

The variable names in a table do not have to be valid MATLAB® identifiers. However, if the names are not valid, you cannot use a formula when you fit or adjust a model; for example:

- You cannot specify `modelspec` using a formula.
- You cannot use a formula to specify the terms to add or remove when you use the `addTerms` function or the `removeTerms` function, respectively.
- You cannot use a formula to specify the lower and upper bounds of the model when you use the `step` or `stepwiseglm` function with the name-value pair arguments `'Lower'` and `'Upper'`, respectively.

You can verify the variable names in `tbl` by using the `isvarname` function. If the variable names are not valid, then you can convert them by using the `matlab.lang.makeValidName` function.

`X` — Predictor variables

matrix

Predictor variables, specified as an n-by-p matrix, where n is the number of observations and p is the number of predictor variables. Each column of `X` represents one variable, and each row represents one observation.

By default, there is a constant term in the model, unless you explicitly remove it, so do not include a column of 1s in `X`.

Data Types: `single` | `double`

y — Response variable
vector | matrix

Response variable, specified as a vector or matrix.

- If `'Distribution'` is not `'binomial'`, then `y` must be an n-by-1 vector, where n is the number of observations. Each entry in `y` is the response for the corresponding row of `X`. The data type must be single or double.
- If `'Distribution'` is `'binomial'`, then `y` can be an n-by-1 vector or n-by-2 matrix with counts in column 1 and `BinomialSize` in column 2.

Data Types: `single` | `double` | `logical` | `categorical`

`modelspec` — Model specification
`'linear'` (default) | character vector or string scalar naming the model | t-by-(p + 1) terms matrix | character vector or string scalar formula in the form `'y ~ terms'`

Model specification, specified as one of these values.

- A character vector or string scalar naming the model.

Value	Model Type
`'constant'`	Model contains only a constant (intercept) term.
`'linear'`	Model contains an intercept and linear term for each predictor.
`'interactions'`	Model contains an intercept, linear term for each predictor, and all products of pairs of distinct predictors (no squared terms).
`'purequadratic'`	Model contains an intercept term and linear and squared terms for each predictor.
`'quadratic'`	Model contains an intercept term, linear and squared terms for each predictor, and all products of pairs of distinct predictors.
`'poly`*`ijk`*`'`	Model is a polynomial with all terms up to degree *`i`* in the first predictor, degree *`j`* in the second predictor, and so on. Specify the maximum degree for each predictor by using numerals 0 though 9. The model contains interaction terms, but the degree of each interaction term does not exceed the maximum value of the specified degrees. For example, `'poly13'` has an intercept and x_1, x_2, x_2^2, x_2^3, x_1*x_2, and x_1*x_2^2 terms, where x_1 and x_2 are the first and second predictors, respectively.

- A t-by-(p + 1) matrix, or a Terms Matrix, specifying terms in the model, where t is the number of terms and p is the number of predictor variables, and +1 accounts for the response variable. A terms matrix is convenient when the number of predictors is large and you want to generate the terms programmatically.

- A character vector or string scalar Formula in the form

  ```
  'y ~ terms',
  ```

  where the `terms` are in Wilkinson Notation. The variable names in the formula must be variable names in `tbl` or variable names specified by `Varnames`. Also, the variable names must be valid MATLAB identifiers.

  The software determines the order of terms in a fitted model by using the order of terms in `tbl` or `X`. Therefore, the order of terms in the model can be different from the order of terms in the specified formula.

Example: `'quadratic'`

Name-Value Pair Arguments

Specify optional comma-separated pairs of `Name,Value` arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside quotes. You can specify several name and value pair arguments in any order as `Name1,Value1,...,NameN,ValueN`.

Example: `'Distribution','normal','link','probit','Exclude',[23,59]` specifies that the distribution of the response is normal, and instructs `fitglm` to use the probit link function and exclude the 23rd and 59th observations from the fit.

`'BinomialSize'` — Number of trials for binomial distribution

1 (default) | numeric scalar | numeric vector | character vector | string scalar

Number of trials for binomial distribution, that is the sample size, specified as the comma-separated pair consisting of `'BinomialSize'` and the variable name in `tbl`, a numeric scalar, or a numeric vector of the same length as the response. This is the parameter `n` for the fitted binomial distribution. `BinomialSize` applies only when the `Distribution` parameter is `'binomial'`.

If `BinomialSize` is a scalar value, that means all observations have the same number of trials.

As an alternative to `BinomialSize`, you can specify the response as a two-column matrix with counts in column 1 and `BinomialSize` in column 2.

Data Types: `single` | `double` | `char` | `string`

`'B0'` — Initial values for coefficient estimates

numeric vector

Initial values for the coefficient estimates, specified as a numeric vector. The default values are initial fitted values derived from the input data.

Data Types: `single` | `double`

`'CategoricalVars'` — Categorical variable list

string array | cell array of character vectors | logical or numeric index vector

Categorical variable list, specified as the comma-separated pair consisting of `'CategoricalVars'` and either a string array or cell array of character vectors containing categorical variable names in the table or dataset array `tbl`, or a logical or numeric index vector indicating which columns are categorical.

- If data is in a table or dataset array `tbl`, then, by default, `fitglm` treats all categorical values, logical values, character arrays, string arrays, and cell arrays of character vectors as categorical variables.
- If data is in matrix `X`, then the default value of `'CategoricalVars'` is an empty matrix `[]`. That is, no variable is categorical unless you specify it as categorical.

For example, you can specify the second and third variables out of six as categorical using either of the following:

Example: `'CategoricalVars',[2,3]`

Example: `'CategoricalVars',logical([0 1 1 0 0 0])`

Data Types: `single` | `double` | `logical` | `string` | `cell`

`'DispersionFlag'` — Indicator to compute dispersion parameter

`false` for `'binomial'` and `'poisson'` distributions (default) | `true`

Indicator to compute dispersion parameter for `'binomial'` and `'poisson'` distributions, specified as the comma-separated pair consisting of `'DispersionFlag'` and one of the following.

`true`	Estimate a dispersion parameter when computing standard errors. The estimated dispersion parameter value is the sum of squared Pearson residuals divided by the degrees of freedom for error (DFE).
`false`	Default. Use the theoretical value of 1 when computing standard errors.

The fitting function always estimates the dispersion for other distributions.

Example: `'DispersionFlag',true`

`'Distribution'` — Distribution of the response variable

`'normal'` (default) | `'binomial'` | `'poisson'` | `'gamma'` | `'inverse gaussian'`

Distribution of the response variable, specified as the comma-separated pair consisting of `'Distribution'` and one of the following.

`'normal'`	Normal distribution
`'binomial'`	Binomial distribution
`'poisson'`	Poisson distribution
`'gamma'`	Gamma distribution
`'inverse gaussian'`	Inverse Gaussian distribution

Example: `'Distribution','gamma'`

`'Exclude'` — Observations to exclude
logical or numeric index vector

Observations to exclude from the fit, specified as the comma-separated pair consisting of `'Exclude'` and a logical or numeric index vector indicating which observations to exclude from the fit.

For example, you can exclude observations 2 and 3 out of 6 using either of the following examples.

Example: `'Exclude',[2,3]`

Example: `'Exclude',logical([0 1 1 0 0 0])`

Data Types: `single` | `double` | `logical`

`'Intercept'` — Indicator for constant term
`true` (default) | `false`

Indicator for the constant term (intercept) in the fit, specified as the comma-separated pair consisting of `'Intercept'` and either `true` to include or `false` to remove the constant term from the model.

Use `'Intercept'` only when specifying the model using a character vector or string scalar, not a formula or matrix.

Example: `'Intercept',false`

`'Link'` — Link function
canonical link function (default) | scalar value | structure

Link function to use in place of the canonical link function, specified as the comma-separated pair consisting of `'Link'` and one of the following.

Link Function Name	**Link Function**	**Mean (Inverse) Function**
`'identity'`	$f(\mu) = \mu$	$\mu = Xb$
`'log'`	$f(\mu) = \log(\mu)$	$\mu = \exp(Xb)$
`'logit'`	$f(\mu) = \log(\mu/(1–\mu))$	$\mu = \exp(Xb) / (1 + \exp(Xb))$
`'probit'`	$f(\mu) = \Phi^{-1}(\mu)$, where Φ is the cumulative distribution function of the standard normal distribution.	$\mu = \Phi(Xb)$
`'comploglog'`	$f(\mu) = \log(–\log(1 – \mu))$	$\mu = 1 – \exp(–\exp(Xb))$
`'reciprocal'`	$f(\mu) = 1/\mu$	$\mu = 1/(Xb)$
p (a number)	$f(\mu) = \mu^p$	$\mu = Xb^{1/p}$

Link Function Name	Link Function	Mean (Inverse) Function
`S` (a structure) with three fields. Each field holds a function handle that accepts a vector of inputs and returns a vector of the same size: • `S.Link` — The link function • `S.Inverse` — The inverse link function • `S.Derivative` — The derivative of the link function	$f(\mu)$ = `S.Link`(μ)	μ = `S.Inverse`(Xb)

The link function defines the relationship $f(\mu) = X^*b$ between the mean response μ and the linear combination of predictors X^*b.

For more information on the canonical link functions, see Canonical Link Function.

Example: `'Link','probit'`

Data Types: `char` | `string` | `single` | `double` | `struct`

`'Options'` — Optimization options

`statset('fitglm')` (default) | structure

Optimization options, specified as a structure. This argument determines the control parameters for the iterative algorithm that `fitglm` uses.

Create the `'Options'` value by using the function `statset` or by creating a structure array containing the fields and values described in this table.

Field Name	Value	Default Value
`Display`	Amount of information displayed by the algorithm • `'off'` — Displays no information • `'final'` — Displays the final output	`'off'`
`MaxIter`	Maximum number of iterations allowed, specified as a positive integer	`100`
`TolX`	Termination tolerance for the parameters, specified as a positive scalar	`1e-6`

You can also enter `statset('fitglm')` in the Command Window to see the names and default values of the fields that `fitglm` accepts in the `'Options'` name-value argument.

Example: `'Options',statset('Display','final','MaxIter',1000)` specifies to display the final information of the iterative algorithm results, and change the maximum number of iterations allowed to 1000.

Data Types: `struct`

`'Offset'` — **Offset variable**
[ ] (default) | numeric vector | character vector | string scalar

Offset variable in the fit, specified as the comma-separated pair consisting of `'Offset'` and the variable name in `tbl` or a numeric vector with the same length as the response.

`fitglm` uses `Offset` as an additional predictor with a coefficient value fixed at 1. In other words, the formula for fitting is

$f(\mu)$ = `Offset` + $X*b$,

where f is the link function, μ is the mean response, and $X*b$ is the linear combination of predictors X. The `Offset` predictor has coefficient `1`.

For example, consider a Poisson regression model. Suppose the number of counts is known for theoretical reasons to be proportional to a predictor `A`. By using the log link function and by specifying `log(A)` as an offset, you can force the model to satisfy this theoretical constraint.

Data Types: `single` | `double` | `char` | `string`

`'PredictorVars'` — **Predictor variables**
string array | cell array of character vectors | logical or numeric index vector

Predictor variables to use in the fit, specified as the comma-separated pair consisting of `'PredictorVars'` and either a string array or cell array of character vectors of the variable names in the table or dataset array `tbl`, or a logical or numeric index vector indicating which columns are predictor variables.

The string values or character vectors should be among the names in `tbl`, or the names you specify using the `'VarNames'` name-value pair argument.

The default is all variables in `X`, or all variables in `tbl` except for `ResponseVar`.

For example, you can specify the second and third variables as the predictor variables using either of the following examples.

Example: `'PredictorVars',[2,3]`

Example: `'PredictorVars',logical([0 1 1 0 0 0])`

Data Types: `single` | `double` | `logical` | `string` | `cell`

`'ResponseVar'` — **Response variable**
last column in `tbl` (default) | character vector or string scalar containing variable name | logical or numeric index vector

Response variable to use in the fit, specified as the comma-separated pair consisting of `'ResponseVar'` and either a character vector or string scalar containing the variable name in the table or dataset array `tbl`, or a logical or numeric index vector indicating which column is the response variable. You typically need to use `'ResponseVar'` when fitting a table or dataset array `tbl`.

For example, you can specify the fourth variable, say `yield`, as the response out of six variables, in one of the following ways.

Example: `'ResponseVar','yield'`

Example: `'ResponseVar',[4]`

Example: `'ResponseVar',logical([0 0 0 1 0 0])`

Data Types: `single` | `double` | `logical` | `char` | `string`

`'VarNames'` — Names of variables

`{'x1','x2',...,'xn','y'}` (default) | string array | cell array of character vectors

Names of variables, specified as the comma-separated pair consisting of `'VarNames'` and a string array or cell array of character vectors including the names for the columns of `X` first, and the name for the response variable `y` last.

`'VarNames'` is not applicable to variables in a table or dataset array, because those variables already have names.

The variable names do not have to be valid MATLAB identifiers. However, if the names are not valid, you cannot use a formula when you fit or adjust a model; for example:

- You cannot use a formula to specify the terms to add or remove when you use the `addTerms` function or the `removeTerms` function, respectively.
- You cannot use a formula to specify the lower and upper bounds of the model when you use the `step` or `stepwiseglm` function with the name-value pair arguments `'Lower'` and `'Upper'`, respectively.

Before specifying `'VarNames',varNames`, you can verify the variable names in `varNames` by using the `isvarname` function. If the variable names are not valid, then you can convert them by using the `matlab.lang.makeValidName` function.

Example: `'VarNames',{'Horsepower','Acceleration','Model_Year','MPG'}`

Data Types: `string` | `cell`

`'Weights'` — Observation weights

`ones(n,1)` (default) | n-by-1 vector of nonnegative scalar values

Observation weights, specified as the comma-separated pair consisting of `'Weights'` and an n-by-1 vector of nonnegative scalar values, where n is the number of observations.

Data Types: `single` | `double`

Output Arguments

collapse all

`mdl` — **Generalized linear regression model**
`GeneralizedLinearModel` object

Generalized linear regression model, specified as a `GeneralizedLinearModel` object created using `fitglm` or `stepwiseglm`.

More About

collapse all

Terms Matrix

A terms matrix `T` is a t-by-(p + 1) matrix specifying terms in a model, where t is the number of terms, p is the number of predictor variables, and +1 accounts for the response variable. The value of `T(i,j)` is the exponent of variable `j` in term `i`.

For example, suppose that an input includes three predictor variables `x1`, `x2`, and `x3` and the response variable `y` in the order `x1`, `x2`, `x3`, and `y`. Each row of `T` represents one term:

- `[0 0 0 0]` — Constant term or intercept
- `[0 1 0 0]` — `x2`; equivalently, `x1^0 * x2^1 * x3^0`
- `[1 0 1 0]` — `x1*x3`
- `[2 0 0 0]` — `x1^2`
- `[0 1 2 0]` — `x2*(x3^2)`

The `0` at the end of each term represents the response variable. In general, a column vector of zeros in a terms matrix represents the position of the response variable. If you have the predictor and response variables in a matrix and column vector, then you must include `0` for the response variable in the last column of each row.

Formula

A formula for model specification is a character vector or string scalar of the form `'`*`y`* `~` *`terms`*`'`.

- *`y`* is the response name.
- *`terms`* represents the predictor terms in a model using Wilkinson notation.

To represent predictor and response variables, use the variable names of the table input `tbl` or the variable names specified by using `VarNames`. The default value of `VarNames` is `{'x1','x2',...,'xn','y'}`.

For example:

- `'y ~ x1 + x2 + x3'` specifies a three-variable linear model with intercept.
- `'y ~ x1 + x2 + x3 – 1'` specifies a three-variable linear model without intercept. Note that formulas include a constant (intercept) term by default. To exclude a constant term from the model, you must include `-1` in the formula.

A formula includes a constant term unless you explicitly remove the term using `-1`.

Wilkinson Notation

Wilkinson notation describes the terms present in a model. The notation relates to the terms present in a model, not to the multipliers (coefficients) of those terms.

Wilkinson notation uses these symbols:

- `+` means include the next variable.
- `–` means do not include the next variable.
- `:` defines an interaction, which is a product of terms.
- `*` defines an interaction and all lower-order terms.
- `^` raises the predictor to a power, exactly as in `*` repeated, so `^` includes lower-order terms as well.
- `()` groups terms.

This table shows typical examples of Wilkinson notation.

Wilkinson Notation	Terms in Standard Notation
`1`	Constant (intercept) term
`x1^k`, where `k` is a positive integer	`x1`, $x1^2$, ..., $x1^k$
`x1 + x2`	`x1`, `x2`
`x1*x2`	`x1`, `x2`, `x1*x2`
`x1:x2`	`x1*x2` only
`–x2`	Do not include `x2`
`x1*x2 + x3`	`x1`, `x2`, `x3`, `x1*x2`
`x1 + x2 + x3 + x1:x2`	`x1`, `x2`, `x3`, `x1*x2`
`x1*x2*x3 – x1:x2:x3`	`x1`, `x2`, `x3`, `x1*x2`, `x1*x3`, `x2*x3`
`x1*(x2 + x3)`	`x1`, `x2`, `x3`, `x1*x2`, `x1*x3`

For more details, see Wilkinson Notation.

Canonical Link Function

The default link function for a generalized linear model is the *canonical link function*.

Distribution	Canonical Link Function Name	Link Function	Mean (Inverse) Function
`'normal'`	`'identity'`	$f(\mu) = \mu$	$\mu = Xb$
`'binomial'`	`'logit'`	$f(\mu) = \log(\mu/(1 - \mu))$	$\mu = \exp(Xb) / (1 + \exp(Xb))$
`'poisson'`	`'log'`	$f(\mu) = \log(\mu)$	$\mu = \exp(Xb)$
`'gamma'`	`-1`	$f(\mu) = 1/\mu$	$\mu = 1/(Xb)$
`'inverse gaussian'`	`-2`	$f(\mu) = 1/\mu^2$	$\mu = (Xb)^{-1/2}$

Tips

- The generalized linear model `mdl` is a standard linear model unless you specify otherwise with the `Distribution` name-value pair.
- For methods such as `plotResiduals` or `devianceTest`, or properties of the `GeneralizedLinearModel` object, see `GeneralizedLinearModel`.

- After training a model, you can generate C/C++ code that predicts responses for new data. Generating C/C++ code requires MATLAB Coder™. For details, see Introduction to Code Generation.

Algorithms

- `fitglm` treats a categorical predictor as follows:
  - A model with a categorical predictor that has L levels (categories) includes $L - 1$ indicator variables. The model uses the first category as a reference level, so it does not include the indicator variable for the reference level. If the data type of the categorical predictor is `categorical`, then you can check the order of categories by using `categories` and reorder the categories by using `reordercats` to customize the reference level. For more details about creating indicator variables, see Automatic Creation of Dummy Variables.
  - `fitglm` treats the group of $L - 1$ indicator variables as a single variable. If you want to treat the indicator variables as distinct predictor variables, create indicator variables manually by using `dummyvar`. Then use the indicator variables, except the one corresponding to the reference level of the categorical variable, when you fit a model. For the categorical predictor `X`, if you specify all columns of `dummyvar(X)` and an intercept term as predictors, then the design matrix becomes rank deficient.
  - Interaction terms between a continuous predictor and a categorical predictor with L levels consist of the element-wise product of the $L - 1$ indicator variables with the continuous predictor.
  - Interaction terms between two categorical predictors with L and M levels consist of the $(L - 1)*(M - 1)$ indicator variables to include all possible combinations of the two categorical predictor levels.
  - You cannot specify higher-order terms for a categorical predictor because the square of an indicator is equal to itself.
- `fitglm` considers `NaN`, `''` (empty character vector), `""` (empty string), `<missing>`, and `<undefined>` values in `tbl`, `X`, and `Y` to be missing values. `fitglm` does not use observations with missing values in the fit. The `ObservationInfo` property of a fitted model indicates whether or not `fitglm` uses each observation in the fit.

Alternative Functionality

- Use `stepwiseglm` to select a model specification automatically. Use `step`, `addTerms`, or `removeTerms` to adjust a fitted model.

References

[1] Collett, D. *Modeling Binary Data*. New York: Chapman & Hall, 2002.

[2] Dobson, A. J. *An Introduction to Generalized Linear Models*. New York: Chapman & Hall, 1990.

[3] McCullagh, P., and J. A. Nelder. *Generalized Linear Models*. New York: Chapman & Hall, 1990.

Extended Capabilities

Tall Arrays

Calculate with arrays that have more rows than fit in memory.

This function supports tall arrays for out-of-memory data with some limitations.

- If any input argument to `fitglm` is a tall array, then all of the other inputs must be tall arrays as well. This includes nonempty variables supplied with the `'Weights'`, `'Exclude'`, `'Offset'`, and `'BinomialSize'` name-value pairs.
- The default number of iterations is 5. You can change the number of iterations using the `'Options'` name-value pair to pass in an options structure. Create an options structure using `statset` to specify a different value for `MaxIter`.
- For tall data, `fitglm` returns a `CompactGeneralizedLinearModel` object that contains most of the same properties as a `GeneralizedLinearModel` object. The main difference is that the compact object is sensitive to memory requirements. The compact object does not include properties that include the data, or that include an array of the same size as the data. The compact object does not contain these `GeneralizedLinearModel` properties:
  - `Diagnostics`
  - `Fitted`
  - `Offset`
  - `ObservationInfo`
  - `ObservationNames`
  - `Residuals`
  - `Steps`
  - `Variables`

  You can compute the residuals directly from the compact object returned by `GLM = fitglm(X,Y)` using

  ```
  RES = Y - predict(GLM,X);
  S = sqrt(GLM.SSE/GLM.DFE);
  histogram(RES,linspace(-3*S,3*S,51))
  ```

For more information, see Tall Arrays for Out-of-Memory Data.

GPU Arrays
Accelerate code by running on a graphics processing unit (GPU) using Parallel Computing Toolbox™.

This function fully supports GPU arrays. For more information, see Run MATLAB Functions on a GPU (Parallel Computing Toolbox).

See Also
`GeneralizedLinearModel` | `glmfit` | `predict` | `stepwiseglm`

Topics
Generalized Linear Model Workflow

Generalized Linear Models

Introduced in R2013b